
AWS Systems Manager

User Guide



AWS Systems Manager: User Guide

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

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

Table of Contents

What is AWS Systems Manager?	1
How it works	1
Capabilities	2
Application management	2
Change management	3
Node management	3
Operations management	5
Quick Setup	5
Shared resources	6
Accessing Systems Manager	6
Supported AWS Regions	6
Systems Manager pricing	7
Systems Manager service name history	7
About SSM Agent	7
About resource groups	9
Supported operating systems	9
Linux	10
macOS	11
Raspberry Pi OS (formerly Raspbian)	12
Windows Server	12
Prerequisites	13
Related content and references	14
Setting up Systems Manager	16
Setting up for EC2 instances	16
Step 1: Sign up for AWS	17
Step 2: Create an Admin IAM user for AWS	17
Step 3: Create non-Admin IAM users and groups for Systems Manager	18
Step 4: Create an IAM instance profile for Systems Manager	21
Step 5: Attach an IAM instance profile to an Amazon EC2 instance	26
Step 6: (Optional) Create a VPC endpoint	28
Step 7: (Optional) Create Systems Manager service roles	32
Step 8: (Optional) Set up integrations with other AWS services	34
Setting up hybrid environments	34
Step 1: Complete general Systems Manager setup steps	36
Step 2: Create an IAM service role for a hybrid environment	36
Step 3: Create a managed-instance activation for a hybrid environment	41
Step 4: Install SSM Agent for a hybrid environment (Linux)	45
Step 5: Install SSM Agent for a hybrid environment (Windows)	50
Setting up edge devices	52
Step 1: Complete general Systems Manager setup steps	54
Step 2: Create an IAM service role for edge devices	55
Step 3: Set up AWS IoT Greengrass	58
Step 4: Update the AWS IoT Greengrass token exchange role and install SSM Agent on your edge devices	59
Getting started	60
Step 1: Install or upgrade AWS command line tools	60
Installing or upgrading and then configuring the AWS CLI	61
Installing or upgrading and then configuring the AWS Tools for PowerShell	61
Step 2: Practice installing or updating SSM Agent on an instance	62
Step 3: Try Systems Manager tutorials and walkthroughs	63
Operations management	64
Application management	64
Change management	64
Node management	65

Shared resources	67
Working with SSM Agent	68
SSM Agent technical reference	68
SSM Agent credentials precedence	69
About the local ssm-user account	70
SSM Agent and the Instance Metadata Service (IMDS)	70
Keeping SSM Agent up-to-date	70
SSM Agent rolling updates by AWS Regions	71
Installing SSM Agent on VMs and on-premises instances	71
Validating on-premises servers, edge devices, and virtual machines using a hardware fingerprint	71
SSM Agent on GitHub	72
AMIs with SSM Agent preinstalled	72
SSM Agent version 3.0	75
Working with SSM Agent on EC2 instances for Linux	76
Manually installing SSM Agent on EC2 instances for Linux	76
Verifying the signature of the SSM Agent	113
Configuring SSM Agent to use a proxy (Linux)	117
Uninstalling SSM Agent from Linux instances	120
Working with SSM Agent on EC2 instances for macOS	121
Manually installing SSM Agent on EC2 instances for macOS	121
Configure SSM Agent to use a proxy (macOS)	122
Uninstall SSM Agent from macOS instances	122
Working with SSM Agent on EC2 instances for Windows Server	123
Manually installing SSM Agent on EC2 instances for Windows Server	123
Configure SSM Agent to use a proxy for Windows Server instances	125
Working with SSM Agent on edge devices	127
Checking SSM Agent status and starting the agent	128
Checking the SSM Agent version number	129
Viewing SSM Agent logs	132
Allowing SSM Agent debug logging	133
Restricting access to root-level commands through SSM Agent	134
Automating updates to SSM Agent	135
Automatically updating SSM Agent	136
Subscribing to SSM Agent notifications	137
SSM Agent communications with AWS managed S3 buckets	138
Required bucket permissions	138
Example	142
Troubleshooting SSM Agent	142
SSM Agent is out of date	143
View SSM Agent log files	143
Agent log files don't rotate (Windows)	143
Unable to connect to SSM endpoints	144
Quick Setup	145
What are the benefits of Quick Setup?	145
Who should use Quick Setup?	145
Getting started with Quick Setup	145
IAM roles and permissions	145
Configure the home AWS Region	147
Availability of Quick Setup in AWS Regions	147
Using Quick Setup	148
Configuration details	148
Editing and deleting your configuration	149
Configuration compliance	149
Troubleshooting Quick Setup results	149
Quick Setup Host Management	150
AWS Config recording	152

Deploy AWS Config conformance packs	153
Configure DevOps Guru with Quick Setup	154
Deploy Distributor packages with Quick Setup	155
Operations Management	157
Incident Manager	157
Explorer	157
What are the features of Explorer?	158
How does Explorer relate to OpsCenter?	159
What is OpsData?	159
Is there a charge to use Explorer?	160
Getting started	160
Using Explorer	171
Exporting OpsData	177
Troubleshooting	179
OpsCenter	180
OpsCenter integration	181
How can OpsCenter benefit my organization?	185
What are the features of OpsCenter?	186
How does OpsCenter work with Amazon EventBridge? Which service should I use?	187
Does OpsCenter integrate with my existing case management system?	188
Is there a charge to use OpsCenter?	188
Does OpsCenter work with my on-premises and hybrid managed nodes?	188
What are the quotas for OpsCenter?	188
Getting started with OpsCenter	189
Creating OpsItems	195
Working with OpsItems	209
Reducing duplicate OpsItems	214
Working with Incident Manager incidents	219
Remediating OpsItem issues	220
Viewing OpsCenter summary reports	224
Supported resources reference	224
Receiving Security Hub findings	227
Auditing and logging OpsCenter activity	229
CloudWatch dashboards	229
Application Management	2
Application Manager	231
What are the benefits of using Application Manager?	232
What are the features of Application Manager?	232
Is there a charge to use Application Manager?	234
What are the resource quotas for Application Manager?	234
Getting started	234
Working with Application Manager	241
AWS AppConfig	256
Parameter Store	256
How can Parameter Store benefit my organization?	256
Who should use Parameter Store?	256
What are the features of Parameter Store?	257
What is a parameter?	258
Setting up Parameter Store	260
Working with Parameter Store	279
Parameter Store walkthroughs	342
Auditing and logging Parameter Store activity	350
Troubleshooting Parameter Store	350
Change Management	352
Change Manager	352
How Change Manager works	353
How can Change Manager benefit my operations?	353

Who should use Change Manager?	354
What are the main features of Change Manager?	354
Is there a charge to use Change Manager?	355
What are the primary components of Change Manager?	356
Setting up Change Manager	357
Working with Change Manager	371
Auditing and logging Change Manager activity	396
Troubleshooting Change Manager	397
Automation	397
How can Automation benefit my organization?	397
Who should use Automation?	399
What is an automation?	399
Setting up Automation	401
Working with automations	407
Automation actions reference	477
Working with runbooks	539
Automation runbook reference	604
Automation walkthroughs	604
Understanding automation statuses	690
Troubleshooting Systems Manager Automation	691
Change Calendar	695
Who should use Change Calendar?	695
Benefits of Change Calendar	695
Setting up Change Calendar	696
Working with Change Calendar	697
Adding Change Calendar dependencies to Automation runbooks	705
Troubleshooting Change Calendar	705
Maintenance Windows	706
Setting up Maintenance Windows	707
Working with maintenance windows (console)	724
Maintenance Windows tutorials (AWS CLI)	732
Maintenance window walkthroughs	777
Maintenance window scheduling and active period options	790
Registering maintenance window tasks without targets	794
Troubleshooting maintenance windows	795
Node Management	798
Fleet Manager	798
Who should use Fleet Manager?	798
How can Fleet Manager benefit my organization?	798
What are the features of Fleet Manager?	799
Getting started with Fleet Manager	799
Working with Fleet Manager	803
Compliance	836
Getting started with Compliance	837
Creating a resource data sync for Compliance	838
Working with Compliance	839
Deleting a resource data sync for Compliance	842
Remediating compliance issues using EventBridge	843
Compliance walkthrough (AWS CLI)	844
Inventory	848
Learn more about Inventory	851
Setting up Inventory	858
Configuring inventory collection	866
Working with inventory data	870
Working with custom inventory	885
Viewing inventory history and change tracking	895
Stopping data collection and deleting inventory data	896

Inventory walkthroughs	897
Troubleshooting Inventory	909
Hybrid Activations	912
Session Manager	912
How can Session Manager benefit my organization?	913
Who should use Session Manager?	914
What are the main features of Session Manager?	914
What is a session?	915
Setting up Session Manager	916
Working with Session Manager	964
Auditing session activity	977
Logging session activity	978
Session document schema	981
Troubleshooting Session Manager	987
Run Command	991
Setting up Run Command	992
Sending commands	995
Handling exit codes with scripts	1011
Understanding command statuses	1013
Run Command walkthroughs	1019
Troubleshooting Run Command	1033
State Manager	1034
How can State Manager benefit my organization?	1034
Who should use State Manager?	1035
What are the features of State Manager?	1035
Is there a charge to use State Manager?	1036
How do I get started with State Manager?	1036
About State Manager	1037
Working with associations	1039
State Manager walkthroughs	1066
Patch Manager	1093
Patch Manager prerequisites	1094
How it works	1096
About SSM documents for patching managed nodes	1124
About patch baselines	1156
Using Kernel Live Patching on Amazon Linux 2 managed nodes	1168
Working with Patch Manager (console)	1174
Working with Patch Manager (AWS CLI)	1210
Patch Manager walkthroughs	1234
Troubleshooting Patch Manager	1245
Distributor	1252
How can Distributor benefit my organization?	1253
Who should use Distributor?	1253
What are the features of Distributor?	1253
What is a package?	1254
Setting up Distributor	1255
Working with Distributor	1257
Auditing and logging Distributor activity	1284
Troubleshooting Distributor	1285
Shared Resources	1287
SSM documents	1287
How can SSM documents benefit my organization?	1287
Who should use SSM documents?	1288
What are the types of SSM documents?	1288
SSM document schema features and examples	1293
SSM document syntax	1308
Systems Manager Command document plugin reference	1314

Viewing SSM Command document content	1351
Creating SSM documents	1352
Deleting custom SSM documents	1360
Comparing SSM document versions	1361
Sharing SSM documents	1361
Searching for SSM documents	1371
Running Systems Manager Command documents from remote locations	1373
Security	1377
Data protection	1377
Data encryption	1378
Internetwork traffic privacy	1380
Identity and access management	1380
Audience	1380
Authenticating with identities	1381
Managing access using policies	1382
How AWS Systems Manager works with IAM	1384
Identity-based policy examples	1391
AWS managed policies	1398
Troubleshooting	1408
Using service-linked roles	1410
Inventory, Maintenance Windows, and Explorer data role	1410
Explorer account discovery role	1412
OpsData and OpsItems creation role	1415
Operational insights creation role	1418
Logging and monitoring	1421
Compliance validation	1422
Resilience	1423
Infrastructure security	1423
Configuration and vulnerability analysis	1424
Security best practices	1424
Systems Manager preventative security best practices	1424
Systems Manager monitoring and auditing best practices	1426
Monitoring	1428
Monitoring tools	1428
Sending node logs to CloudWatch Logs (CloudWatch agent)	1429
Migrate Windows Server node log collection to the CloudWatch agent	1430
Store CloudWatch agent configuration settings in Parameter Store	1436
Rolling back to log collection with SSM Agent	1436
Sending SSM Agent logs to CloudWatch Logs	1438
Monitoring Automation metrics using Amazon CloudWatch	1440
Automation metrics	1441
Monitoring Run Command metrics using Amazon CloudWatch	1441
Systems Manager Run Command metrics and dimensions	1442
Logging AWS Systems Manager API calls with AWS CloudTrail	1442
Systems Manager information in CloudTrail	1442
Understanding Systems Manager log file entries	1443
Logging Automation action output with CloudWatch Logs	1445
Configuring Amazon CloudWatch Logs for Run Command	1447
Specifying CloudWatch Logs when you send commands	1448
Viewing command output in CloudWatch Logs	1449
Monitoring with Amazon EventBridge	1449
Configuring EventBridge for Systems Manager events	1450
Amazon EventBridge event examples for Systems Manager	1452
Amazon EventBridge target examples for Systems Manager	1461
Monitoring Systems Manager status changes using Amazon SNS notifications	1462
Configure Amazon SNS notifications for AWS Systems Manager	1462
Example Amazon SNS notifications for AWS Systems Manager	1467

Use Run Command to send a command that returns status notifications	1468
Use a maintenance window to send a command that returns status notifications	1471
Product and service integrations	1475
Integration with AWS services	1475
Compute	1475
Internet of Things (IoT)	1476
Storage	1477
Developer Tools	1478
Security, Identity, and Compliance	1478
Cryptography and PKI	1479
Management and Governance	1480
Networking and Content Delivery	1483
Analytics	1484
Application Integration	1485
AWS Management Console	1485
Running scripts from Amazon S3	1485
Referencing AWS Secrets Manager secrets from Parameter Store parameters	1488
Integration with other products and services	1492
Running scripts from GitHub	1493
Using Chef InSpec profiles with Systems Manager Compliance	1498
How it works	1499
Running an InSpec compliance scan	1499
Integration examples from the community	1502
Blog posts	1502
Tagging Systems Manager resources	1505
Taggable Systems Manager resources	1505
Tagging Systems Manager documents	1506
Creating documents with tags	1506
Adding tags to existing documents	1507
Removing tags from SSM documents	1509
Tagging maintenance windows	1510
Creating maintenance windows with tags	1511
Adding tags to existing maintenance windows	1511
Removing tags from maintenance windows	1513
Tagging managed nodes	1515
Creating or activating managed nodes with tags	1515
Adding tags to existing managed nodes	1515
Removing tags from managed nodes	1517
Tagging OpsItems	1519
Creating OpsItems with tags	1519
Adding tags to existing OpsItems	1519
Removing tags from Systems Manager OpsItems	1521
Tagging automations	1522
Adding tags to automations (console)	1522
Adding tags to automations (command line)	1523
Removing tags from automations	1524
Tagging Systems Manager parameters	1525
Creating parameters with tags	1525
Adding tags to existing parameters	1526
Removing tags from SSM parameters	1527
Tagging patch baselines	1529
Creating patch baselines with tags	1529
Adding tags to existing patch baselines	1529
Removing tags from patch baselines	1531
AWS Systems Manager reference	1534
EventBridge event patterns and types for Systems Manager	1534
Event type: Automation	1535

Event type: Change Calendar	1536
Event type: Configuration Compliance	1536
Event type: Inventory	1536
Event type: State Manager	1537
Event type: Maintenance Window	1537
Event type: Parameter Store	1538
Event type: Run Command	1539
Cron and rate expressions	1540
General information about cron and rate expressions	1541
Cron and rate expressions for associations	1544
Cron and rate expressions for maintenance windows	1545
ec2messages, ssessages, and other API operations	1546
Creating formatted date and time strings for Systems Manager	1547
Formatting date and time strings for Systems Manager	1547
Creating custom date and time strings for Systems Manager	1548
Use cases and best practices	1550
Deleting Systems Manager resources and artifacts	1552
Choosing between State Manager and Maintenance Windows	1554
State Manager and Maintenance Windows: Key use cases	1554
Document history	1558
Updates prior to June 2018	1631
Document conventions	1643
AWS glossary	1644

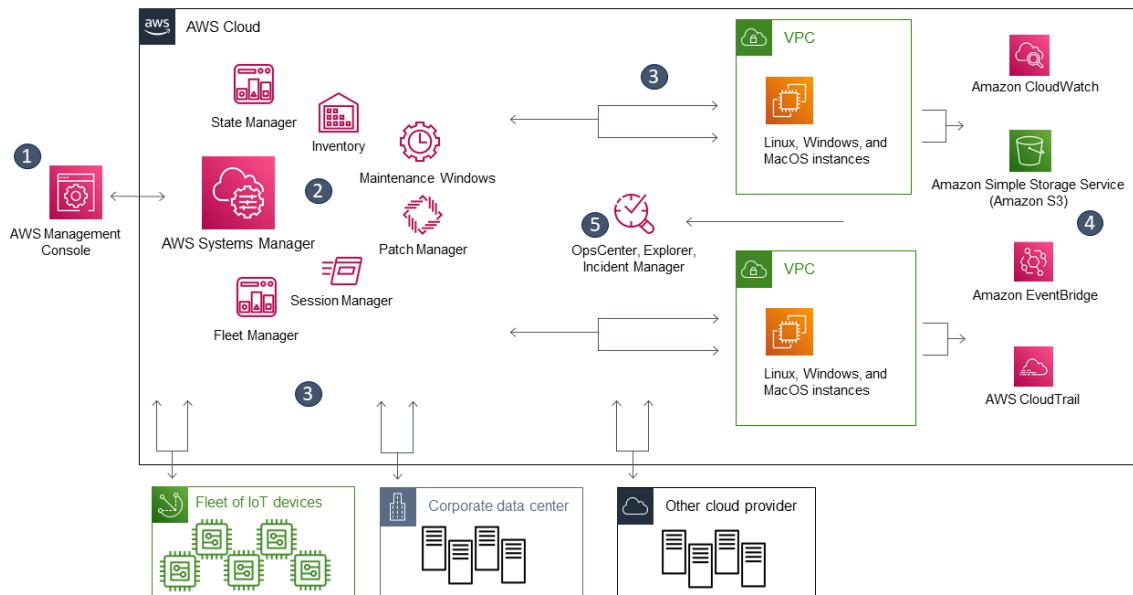
What is AWS Systems Manager?

AWS Systems Manager is a collection of capabilities to help you manage your applications and infrastructure running in the AWS Cloud. Systems Manager simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.

How Systems Manager works

The following diagram describes how some Systems Manager capabilities perform actions on your resources. The diagram doesn't cover all capabilities. Each enumerated interaction is described after the diagram.

Diagram 1: General example of Systems Manager process flow



1. **Access Systems Manager** – Use one of the available options for [accessing Systems Manager](#).
2. **Choose a Systems Manager capability** – Determine which capability can help you perform the action you want to perform on your resources. The diagram shows only a few of the capabilities that IT administrators and DevOps personnel use to manage their applications and resources.
3. **Verification and processing** – Systems Manager verifies that your AWS Identity and Access Management (IAM) user, group, or role has permission to perform the action you specified. If the target of your action is a managed node, the Systems Manager Agent (SSM Agent) running on the node performs the action. For other types of resources, Systems Manager performs the specified action or communicates with other AWS services to perform the action on behalf of Systems Manager.
4. **Reporting** – Systems Manager, SSM Agent, and other AWS services that performed an action on behalf of Systems Manager report status. Systems Manager can send status details to other AWS services, if configured.

5. **Systems Manager operations management capabilities** – If enabled, Systems Manager operations management capabilities such as Explorer, OpsCenter, and Incident Manager aggregate operations data or create artifacts in response to events or errors with your resources. These artifacts include operational work items (OpsItems) and incidents. Systems Manager operations management capabilities provide operational insight into your applications and resources and automated remediation solutions to help troubleshoot problems.

Systems Manager capabilities

Systems Manager groups capabilities into the following categories. Choose the tabs under each category to learn more about each capability.

Topics

- [Application management \(p. 2\)](#)
- [Change management \(p. 3\)](#)
- [Node management \(p. 3\)](#)
- [Operations management \(p. 5\)](#)
- [Quick Setup \(p. 5\)](#)
- [Shared resources \(p. 6\)](#)

Application management

Application Manager

[Application Manager \(p. 231\)](#) helps DevOps engineers investigate and remediate issues with their AWS resources in the context of their applications and clusters. In Application Manager, an *application* is a logical group of AWS resources that you want to operate as a unit. This logical group can represent different versions of an application, ownership boundaries for operators, or developer environments, to name a few. Application Manager support for container clusters includes both Amazon Elastic Kubernetes Service (Amazon EKS) and Amazon Elastic Container Service (Amazon ECS) clusters. Application Manager aggregates operations information from multiple AWS services and Systems Manager capabilities to a single AWS Management Console.

AppConfig

[AppConfig \(p. 256\)](#) helps you create, manage, and deploy application configurations and feature flags. AppConfig supports controlled deployments to applications of any size. You can use AppConfig with applications hosted on Amazon EC2 instances, AWS Lambda containers, mobile applications, or edge devices. To prevent errors when deploying application configurations, AppConfig includes validators. A validator provides a syntactic or semantic check to verify that the configuration you want to deploy works as intended. During a configuration deployment, AppConfig monitors the application to verify that the deployment is successful. If the system encounters an error or if the deployment invokes an alarm, AppConfig rolls back the change to minimize impact for your application users.

Parameter Store

[Parameter Store \(p. 256\)](#) provides secure, hierarchical storage for configuration data and secrets management. You can store data such as passwords, database strings, Amazon Elastic Compute Cloud (Amazon 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.

Change management

Change Manager

[Change Manager \(p. 352\)](#) is an enterprise change management framework for requesting, approving, implementing, and reporting on operational changes to your application configuration and infrastructure. From a single *delegated administrator account*, if you use AWS Organizations, you can manage changes across multiple AWS accounts in multiple AWS Regions. Alternatively, using a *local account*, you can manage changes for a single AWS account. Use Change Manager for managing changes to both AWS resources and on-premises resources.

Automation

Use [Automation \(p. 397\)](#) to automate common maintenance and deployment tasks. You can use Automation to create and update Amazon Machine Images (AMIs), 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 \(p. 695\)](#) helps you set up date and time ranges when actions you specify (for example, in [Systems Manager Automation \(p. 397\)](#) runbooks) can or can't be performed in your AWS account. In Change Calendar, these ranges are called *events*. When you create a Change Calendar entry, you're creating a [Systems Manager document \(p. 1287\)](#) 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. You can add events manually in the Change Calendar interface or import events from a supported third-party calendar using an `.ics` file.

Maintenance Windows

Use [Maintenance Windows \(p. 706\)](#) to set up recurring schedules for managed instances to run administrative tasks such as installing patches and updates without interrupting business-critical operations.

Node management

A *managed node* is any machine configured for Systems Manager. Systems Manager supports Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and on-premises servers or virtual machines (VMs), including VMs in other cloud environments.

Compliance

Use [Compliance \(p. 836\)](#) to scan your fleet of managed nodes for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and AWS Regions, and then drill down into specific resources that aren't compliant. By default, 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.

Fleet Manager

[Fleet Manager \(p. 798\)](#) is a unified user interface (UI) experience that helps you remotely manage your nodes. With Fleet Manager, you can view the health and performance status of your entire fleet from one console. You can also gather data from individual devices and instances to perform common troubleshooting and management tasks from the console. This includes viewing directory and file contents, Windows registry management, operating system user management, and more.

Inventory

[Inventory \(p. 848\)](#) automates the process of collecting software inventory from your managed nodes. You can use Inventory to gather metadata about applications, files, components, patches, and more.

Session Manager

Use [Session Manager \(p. 912\)](#) to manage your edge devices and Amazon Elastic Compute Cloud (Amazon EC2) instances through an interactive one-click browser-based shell or through the AWS CLI. Session Manager provides secure and auditable edge device and instance management without needing to open inbound ports, maintain bastion hosts, or manage SSH keys. Session Manager also allows you to comply with corporate policies that require controlled access to edge devices and instances, strict security practices, and fully auditable logs with edge device and instance access details, while still providing end users with simple one-click cross-platform access to your edge devices and EC2 instances. To use Session Manager, you must enable the advanced-instances tier. For more information, see [Turning on the advanced-instances tier \(p. 805\)](#).

Run Command

Use [Run Command \(p. 991\)](#) to remotely and securely manage the configuration of your managed nodes at scale. Use Run Command to perform on-demand changes such as updating applications or running Linux shell scripts and Windows PowerShell commands on a target set of dozens or hundreds of managed nodes.

State Manager

Use [State Manager \(p. 1034\)](#) to automate the process of keeping your managed nodes in a defined state. You can use State Manager to guarantee that your managed nodes are bootstrapped with specific software at startup, joined to a Windows domain (Windows Server nodes only), or patched with specific software updates.

Patch Manager

Use [Patch Manager \(p. 1093\)](#) to automate the process of patching your managed nodes 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 applications released by Microsoft.)

This capability allows you to scan managed nodes for missing patches and apply missing patches individually or to large groups of managed nodes by using tags. Patch Manager uses *patch baselines*, which can include rules for auto-approving patches within days of their release, and 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, or you can patch your managed nodes on demand at any time.

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 managed node. For Linux, you also have the ability to update any package on the managed node, not just those that are classified as operating system security updates. You can also generate patch reports that are sent to an S3 bucket of your choice. For a single managed node, reports include details of all patches for the machine. For a report on all managed nodes, only a summary of how many patches are missing is provided.

Distributor

Use [Distributor \(p. 1252\)](#) to create and deploy packages to managed nodes. With Distributor, you can package your own software—or find AWS-provided agent software packages, such as [AmazonCloudWatchAgent](#)—to install on Systems Manager managed nodes. After you install a package for the first time, you can use Distributor to uninstall and reinstall a new package version, or perform an in-place update that adds new or changed files. Distributor publishes resources, such as software packages, to Systems Manager managed nodes.

Hybrid Activations

To set up servers and VMs in your hybrid environment as managed instances, create a managed instance [activation \(p. 34\)](#). After you complete the activation, you receive an activation code and ID. This code and ID combination functions like an Amazon Elastic Compute Cloud (Amazon EC2) access ID and secret key to provide secure access to the Systems Manager service from your managed instances.

You can also create an activation for edge devices if you want to manage them by using Systems Manager.

Operations management

Incident Manager

[Incident Manager \(p. 157\)](#) is an incident management console that helps users mitigate and recover from incidents affecting their AWS hosted applications.

Incident Manager increases incident resolution by notifying responders of impact, highlighting relevant troubleshooting data, and providing collaboration tools to get services back up and running. Incident Manager also automates response plans and allows responder team escalation.

Explorer

[Explorer \(p. 157\)](#) 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 AWS Regions. In Explorer, OpsData includes metadata about your Amazon 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 OpsCenter, a capability of Systems Manager, to run Automation runbooks and resolve those issues.

OpsCenter

[OpsCenter \(p. 180\)](#) 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 runbooks that you can use to 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 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.

Quick Setup

Use [Quick Setup \(p. 145\)](#) to configure frequently used AWS services and features with recommended best practices. You can use Quick Setup in an individual AWS account or across multiple AWS accounts and AWS Regions by integrating with AWS Organizations. Quick Setup simplifies setting up services, including Systems Manager, by automating common or recommended tasks. These tasks include, for example, creating required AWS Identity and Access Management (IAM) instance profile roles and setting up operational best practices, such as periodic patch scans and inventory collection.

Shared resources

Documents

A [Systems Manager document \(p. 1287\)](#) (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 runbooks, 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.

Accessing Systems Manager

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

Systems Manager console

The [Systems Manager console](#) is a browser-based interface to access and use Systems Manager.

AWS IoT Greengrass V2 console

You can view and manage edge devices that are configured for AWS IoT Greengrass in the [Greengrass console](#).

AWS command line tools

By using the AWS command line tools, you can issue commands at your system's command line to perform Systems Manager and other AWS tasks. The tools are supported on Linux, macOS, and Windows. Using the AWS Command Line Interface (AWS 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](#) and the [AWS Tools for Windows PowerShell](#). For information about installing and using the AWS CLI, see the [AWS Command Line Interface User Guide](#). For information about installing and using the Tools for Windows PowerShell, see the [AWS Tools for Windows PowerShell User Guide](#).

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 Windows 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](#).

Supported AWS Regions

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 verify 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.region.amazonaws.com`
- **AWS CloudFormation resource types:** `AWS::SSM::Document`
- **AWS Config rule identifier:** `EC2_INSTANCE_MANAGED_BY_SSM`
- **AWS Command Line Interface (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:region:account-id:patchbaseline/pb-07d8884178EXAMPLE`

About SSM Agent

AWS Systems Manager Agent (SSM Agent) is Amazon software that runs on Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and on-premises servers and virtual machines (VMs). 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 AWS Systems Manager. Some Amazon Machine Images (AMIs) are configured to launch instances with [SSM Agent \(p. 68\)](#) preinstalled. (You can also configure a custom AMI to preinstall SSM Agent.) For more information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

On other AMIs; AWS IoT Greengrass core devices; and on-premises servers, edge devices, and virtual machines in your hybrid environment, you must install the agent manually, as described in the following table.

Important

An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#).

Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

Operating system type	SSM Agent installation
Linux	SSM Agent is installed by default on Amazon Linux, Amazon Linux 2, SUSE Linux Enterprise Server (SLES) 12 and 15, Ubuntu Server 16.04, 18.04 LTS, and 20.04 <i>base</i> Amazon EC2 AMIs. You must manually install SSM Agent on other versions of Amazon EC2 for Linux, including non-base images. For more information, see Working with SSM Agent on EC2 instances for Linux (p. 76) .
macOS	SSM Agent is installed by default on macOS 10.14.6 (Mojave), 10.15.7 (Catalina), and 11.x (BigSur) AMIs for Amazon EC2. For more information, see Working with SSM Agent on EC2 instances for macOS (p. 121) .
Windows	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 Working with SSM Agent on EC2 instances for Windows Server (p. 123) .
Edge devices	Systems Manager supports the following types of edge devices: <ul style="list-style-type: none">• AWS IoT Greengrass core devices• AWS IoT devices• Non-AWS IoT devices Setup requirements differ based on the type of edge device. For more information, see Setting up AWS Systems Manager for edge devices (p. 52) .
On-premises servers and VMs	You must manually install SSM Agent on on-premises servers and virtual machines (VMs) in your hybrid environment. The SSM Agent download and installation process for these machines is different than the process used for Amazon EC2 instances. For more information, see the following topics: <ul style="list-style-type: none">• Install SSM Agent for a hybrid environment (Windows) (p. 50)

Operating system type	SSM Agent installation
	<ul style="list-style-type: none">• Install SSM Agent for a hybrid environment (Linux) (p. 45)

About resource groups

With Systems Manager, you can associate AWS resources by assigning *resource tags*. You can then view operational data for these resources as a *resource group*. Resource groups help you monitor and troubleshoot your resources.

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

- A group of AWS IoT Greengrass core devices
- A group of Amazon 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 Amazon Simple Storage Service (Amazon S3) bucket to store patching operation log output
- A Systems Manager *maintenance window* that specifies the schedule for the patching operation

After tagging your resources, you can view the patch status of those resources in a Systems Manager consolidated dashboard. If a problem arises with any of the resources, you can take corrective action immediately.

Supported operating systems

You can manage Amazon Elastic Compute Cloud (Amazon EC2) instances and on-premises servers and virtual machines (VMs), including VMs hosted by other cloud providers, by using Systems Manager. These nodes must be running one of the following operating systems.

Note

If you plan to manage and configure AWS IoT Greengrass core devices by using Systems Manager, those devices must meet the requirements for AWS IoT Greengrass. For more information, see [Setting up AWS IoT Greengrass core devices](#) in the *AWS IoT Greengrass Version 2 Developer Guide*.

If you plan to manage and configure AWS IoT and non-AWS edge devices, those devices must meet the requirements listed here and be configured as on-premises managed nodes for Systems Manager. For more information, see [Setting up AWS Systems Manager for edge devices \(p. 52\)](#).

Operating system types

- [Linux \(p. 10\)](#)
- [macOS \(p. 11\)](#)
- [Raspberry Pi OS \(formerly Raspbian\) \(p. 12\)](#)
- [Windows Server \(p. 12\)](#)

Linux

Amazon Linux

Versions	x86	x86_64	ARM64
2012.03 – 2018.03	✓	✓	

Note

Beginning with version 2015.03, Amazon Linux is released in x86_64 versions.

Amazon Linux 2

Versions	x86	x86_64	ARM64
2.0 and all later versions		✓	✓

CentOS

Versions	x86	x86_64	ARM64
6.x ¹	✓	✓	
7.1 and later 7.x versions		✓	✓
8.0-8.5 versions		✓	✓

¹ SSM Agent no longer officially supports these versions and no longer updates the agent for these versions of CentOS. SSM Agent version 3.0.1390.0 and earlier is supported for CentOS 6.

CentOS Stream

Versions	x86	x86_64	ARM64
8		✓	✓

Debian Server

Versions	x86	x86_64	ARM64
Jessie (8)		✓	
Stretch (9)		✓	✓
Buster (10)		✓	✓
Bullseye (11)		✓	✓

Oracle Linux

Versions	x86	x86_64	ARM64
7.5-7.8		✓	

Versions	x86	x86_64	ARM64
8.1-8.5		✓	

Red Hat Enterprise Linux (RHEL)

Versions	x86	x86_64	ARM64
6.x ¹	✓	✓	
7.0-7.5		✓	
7.6-8.5		✓	✓

¹ SSM Agent no longer officially supports these versions and no longer updates the agent for these versions of RHEL. SSM Agent version 3.0.1390.0 and earlier is supported for RHEL 6.

Rocky Linux

Versions	x86	x86_64	ARM64
8.4-8.5		✓	✓

SUSE Linux Enterprise Server (SLES)

Versions	x86	x86_64	ARM64
12 and later 12.x versions		✓	
15 and later 15.x versions		✓	✓

Ubuntu Server

Versions	x86	x86_64	ARM64
12.04 LTS and 14.04 LTS	✓	✓	
16.04 LTS and 18.04 LTS		✓	✓
20.04 LTS and 20.10 STR		✓	✓

macOS

Version	x86	x86_64	ARM64
10.14.x (Mojave)		✓	
10.15.x (Catalina)		✓	

Version	x86	x86_64	ARM64
11.x (BigSur)		✓	

Note

macOS support is limited to the following AWS Regions:

- US East (N. Virginia) (us-east-1)
- US East (Ohio) (us-east-2)
- US West (Oregon) (us-west-2)
- Europe (Ireland) (eu-west-1)
- Asia Pacific (Singapore) (ap-southeast-1)

For more information about Amazon EC2 support for macOS, see [Amazon EC2 Mac instances](#) in the [Amazon EC2 User Guide for Linux Instances](#)

Raspberry Pi OS (formerly Raspbian)

Version	ARM32
8 (Jessie)	✓
9 (Stretch)	✓

Related content

[Manage Raspberry Pi devices using AWS Systems Manager](#)

Windows Server

SSM Agent requires Windows PowerShell 3.0 or later to run certain AWS Systems Manager documents (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](#).

Version	x86	x86_64	ARM64
2008 ¹	✓	✓	
2008 R2 ¹		✓	
2012 and 2012 R2		✓	
2016		✓	
2019		✓	
2022		✓	

¹ As of January 14, 2020, Windows Server 2008 is no longer supported for feature or security updates from Microsoft. Legacy Amazon Machine Images (AMIs) for Windows Server 2008 and 2008 R2 still

include version 2 of SSM Agent preinstalled, but Systems Manager no longer officially supports 2008 versions and no longer updates the agent for these versions of Windows Server. In addition, [SSM Agent version 3.0 \(p. 75\)](#) might not be compatible with all operations on Windows Server 2008 and 2008 R2. The final officially supported version of SSM Agent for Windows Server 2008 versions is 2.3.1644.0.

Systems Manager prerequisites

The prerequisites for using AWS Systems Manager to manage your Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and 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 \(p. 16\)](#)
- [Setting up AWS Systems Manager for hybrid environments \(p. 34\)](#)
- [Setting up AWS Systems Manager for edge devices \(p. 52\)](#)

This topic provides an overview of these prerequisites.

To complete prerequisites for using Systems Manager

1. Create an AWS account and configure the required AWS Identity and Access Management (IAM) roles.
2. Verify that Systems Manager is supported in the AWS Regions where you want to use the service.
3. Verify that your machines run a supported [operating system](#).
4. For edge devices, verify that your devices are configured to run the AWS IoT Greengrass Core software. For edge devices that don't run AWS IoT Greengrass Core software, the machines must be configured as on-premises machines for Systems Manager.
5. For Amazon EC2 instances, create an IAM instance profile and attach it to your machines.
6. For on-premises servers, edge devices, and VMs, create an IAM service role.
7. (Recommended) Create a VPC endpoint in Amazon Virtual Private Cloud (Amazon VPC) to use with Systems Manager.

If you don't use a VPC endpoint, configure your managed instances to allow [HTTPS \(port 443\)](#) outbound traffic to the Systems Manager endpoints. For information, see [\(Optional\) Create a VPC endpoint](#).

8. For on-premises servers, edge devices, VMs, and Amazon EC2 instances created from Amazon Machine Images (AMIs) that aren't supplied by AWS, ensure that a Transport Layer Security (TLS) certificate is installed.
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 the SSM Agent on each of your managed nodes.
11. For Amazon EC2 instances, verify the instance can reach the Instance Metadata Service (IMDS). Systems Manager relies on EC2 instance metadata to function correctly.

Note

SSM Agent initiates all connections to the Systems Manager service in cloud. For this reason, you don't need to configure your firewall to allow inbound traffic to your managed nodes for Systems Manager.

If your managed nodes don't display in Systems Manager after you've follow these steps, see [Troubleshooting managed node availability \(p. 816\)](#).

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. If you plan to use computing resources provided by AWS and on-premises servers and virtual machines (VMs), you also need to understand Amazon Elastic Compute Cloud 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:

- [Amazon Elastic Compute Cloud](#)
- [Getting Started with Amazon EC2 Linux Instances](#)
- [Getting Started with Amazon EC2 Windows Instances](#)
- [What is Amazon EC2? \(Linux\)](#)
- [What is Amazon EC2? \(Windows\)](#)
- [Amazon EC2 Mac instances](#) in the *Amazon EC2 User Guide for Linux Instances*

For more information about IAM, see the following:

- [AWS Identity and Access Management \(IAM\)](#)
- [Getting Started with IAM](#)
- [What is IAM?](#)

Related content and references

For more information about Systems Manager, see the following references, guides, blogposts, and sites.

Related API references

- [AWS Systems Manager API Reference](#) – Provides descriptions, syntax, and usage examples for each of the Systems Manager actions and data types.
- [AWS AppConfig API Reference](#) – Provides descriptions, syntax, and usage examples for each of the AWS AppConfig actions and data types.

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](#), and other posts tagged with [#Systems Manager](#).
 - [Systems Manager issues in AWS re:Post](#) – Follow announcements, or post or answer a question in the re:Post Community.
 - [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.
 - [AWS Systems Manager Service Level Agreement](#) – The Systems Manager Service Level Agreement (SLA) is a policy governing the use of Systems Manager and applies separately to each AWS account using Systems Manager.

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

- [**Classes & Workshops**](#) – Links to role-based and specialty courses, in addition to 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 webpage 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.

Setting up AWS Systems Manager

Complete the tasks in this section to set up and configure roles, user accounts, permissions, and initial resources for AWS Systems Manager. The tasks described in this section are typically performed by AWS account and systems administrators. After these steps are complete, users in your organization can use Systems Manager to configure, manage, and access your *managed nodes*. A managed node is any machine configured for Systems Manager. Systems Manager supports the following types of managed nodes: Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and on-premises servers and virtual machines (VMs) in a hybrid environment.

Note

If you plan to use Amazon EC2 instances *and* your own computing resources in a hybrid environment, follow the steps in [Setting up AWS Systems Manager for EC2 instances \(p. 16\)](#).

That topic presents steps in the best order for completing Systems Manager setup for EC2 instances and hybrid machines.

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're ready to use all Systems Manager capabilities.

Topics

- [Setting up AWS Systems Manager for EC2 instances \(p. 16\)](#)
- [Setting up AWS Systems Manager for hybrid environments \(p. 34\)](#)
- [Setting up AWS Systems Manager for edge devices \(p. 52\)](#)

Setting up AWS Systems Manager for EC2 instances

Complete the tasks in this section to set up and configure roles, user accounts, permissions, and initial resources for AWS Systems Manager. The tasks described in this section are typically performed by AWS account and systems administrators. After these steps are complete, users in your organization can use Systems Manager to configure, manage, and access Amazon Elastic Compute Cloud (Amazon EC2) instances.

Note

If you plan to use Systems Manager to manage and configure on-premises machines, follow the setup steps in [Setting up AWS Systems Manager for hybrid environments \(p. 34\)](#). If you plan to use both Amazon EC2 instances *and* your own computing resources in a hybrid environment, follow the steps here first. This section presents steps in the recommended 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're ready to use all Systems Manager capabilities.

Contents

- [Step 1: Sign up for AWS \(p. 17\)](#)
- [Step 2: Create an Admin IAM user for AWS \(p. 17\)](#)
- [Step 3: Create non-Admin IAM users and groups for Systems Manager \(p. 18\)](#)

- Step 4: Create an IAM instance profile for Systems Manager (p. 21)
- Step 5: Attach an IAM instance profile to an Amazon EC2 instance (p. 26)
- Step 6: (Optional) Create a VPC endpoint (p. 28)
- Step 7: (Optional) Create Systems Manager service roles (p. 32)
- Step 8: (Optional) Set up integrations with other AWS services (p. 34)

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

1. Open <https://portal.aws.amazon.com/billing/signup>.
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 \(p. 17\)](#).

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. As a best practice, create only the credentials that the user needs. For example, for a user who requires access only through the AWS Management Console, do not create access keys. Optionally, you can configure [multi-factor authentication \(MFA\)](#) for the user. MFA requires the user to provide a one-time-use code each time he or she signs into the AWS Management Console.

To create an IAM user with restricted permissions, see [Step 3: Create non-Admin IAM users and groups for Systems Manager \(p. 18\)](#).

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

1. Sign in to the [IAM console](#) as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

Note

We strongly recommend that you adhere to the best practice of using the **Administrator** IAM user that follows 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 users**.
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 \(p. 18\)](#).

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.

Note

You can grant users or groups full Systems Manager access using the AWS Identity and Access Management (IAM) policy **AmazonSSMFullAccess**, as described later in this section. In practice, however, you might want to limit users or groups to only some Systems Manager features. In the sections for many Systems Manager capabilities, such as Session Manager and Maintenance Windows, we provide instructions for limiting access to actions and resources for that capability only.

For information about using IAM policies to control user access to Systems Manager capabilities and resources, see [AWS Systems Manager identity-based policy examples \(p. 1391\)](#).

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 \(p. 19\)](#)
- [Task 2: Create users and assign permissions \(p. 20\)](#)

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 AWS Identity and Access Management (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 AWS Systems Manager, as described in [Control user session access to instances \(p. 926\)](#).

For additional examples of custom IAM policies for Systems Manager, see [Customer managed policy examples \(p. 1395\)](#).

For comprehensive information about using IAM policies for Systems Manager access, see [Identity and access management for AWS Systems Manager \(p. 1380\)](#).

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 **User groups**, and then choose **Create group**.
2. On the **Create user group** page, for **User group name**, enter a name for the group, such as **SSMUserGroup**.
3. For **Add users to the group**, check the box next to any existing user you want to add to the group.
4. For **Attach permissions policies**, 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. 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 [What are resource groups?](#) in the *AWS Resource Groups User Guide*.

- To provide users in this group with full access to the Systems Manager console, choose the **AmazonSSMFullAccess** policy.

-or-

If you want users in this group only to view Systems Manager data, and not create or update resources, choose the **AmazonSSMReadOnlyAccess** policy.

- To provide users with access to the **Built-In Insights** and **Dashboard by CloudWatch** pages in the Systems Manager console, select the check boxes next to these managed policies:
 - **AWSHealthFullAccess**
 - **AWSConfigUserAccess**

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.

- **CloudWatchReadOnlyAccess**
- 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.

This policy provides read-only access to Amazon 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.
5. Verify that the correct policies are added to this group, and then choose **Create group**.

Continue to [Task 2: Create users and assign permissions \(p. 20\)](#).

Task 2: Create users and assign permissions

Create AWS Identity and Access Management (IAM) users for the individuals who require access to AWS Systems Manager. Then, 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 Management Console](#).

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

To create users and assign permissions

1. In the navigation pane of the IAM console, choose **Users**, and then choose **Add users**.
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 **Access key - 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 **Password - 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. Optionally, you can 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** page, choose **Add user to group**.
7. In the group list, choose the user groups 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. 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. Your users need this information 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 won't 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 message 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://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 isn't included in the generated email message. Provide the password 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. Select the name of the user to add to a group. On the **Summary** page, choose **Add permissions**.
3. For **Add user to group**, select the check 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 \(p. 21\)](#).

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. 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 Quick Setup, a capability of AWS Systems Manager, to quickly configure an instance profile on all instances in your AWS account. Quick Setup also creates an IAM service role (or *assume role*), which allows 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 \(p. 145\)](#).

Note the following details about creating an IAM instance profile:

- If you're 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, configure your servers and VMs to use an IAM service role. For more information, see [Create an IAM service role for a hybrid environment \(p. 36\)](#).

- If you change the IAM instance profile, it might take some time for the instance credentials to refresh. SSM Agent won't 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 AWS Systems Manager. To provide permissions for communication between instances and the Systems Manager API, we recommend creating custom policies that reflect 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`, allows an instance to use 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 allows 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're using a virtual private cloud (VPC) endpoint to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS 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 Amazon Simple Storage Service (Amazon S3) buckets. These buckets are publicly accessible.

In a private VPC endpoint environment, however, explicitly provide access to the following buckets.

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

For more information, see [Step 6: \(Optional\) Create a VPC endpoint \(p. 28\)](#), [SSM Agent communications with AWS managed S3 buckets \(p. 138\)](#), and [AWS PrivateLink and VPC endpoints](#) in the *Amazon VPC User Guide*.

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

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

- To access scripts to use in commands you run that you store in the S3 bucket.
- 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. However, 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 User Guide*.

Policy: AmazonSSMDirectoryServiceAccess

Required only if you plan to join Amazon EC2 instances 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 features such as Amazon EventBridge or Amazon 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 EventBridge and CloudWatch Logs features is optional. However, 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 EventBridge 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:

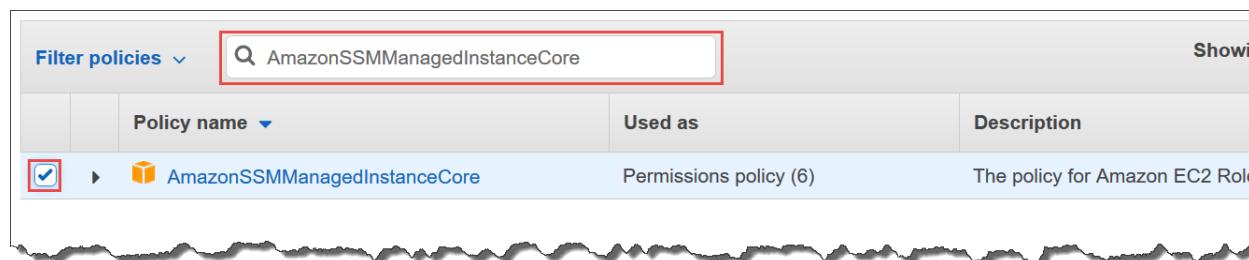
- [Restricting access to Systems Manager parameters using IAM policies \(p. 260\)](#)
- [Setting up Automation \(p. 401\)](#)
- [Verify or create an IAM role with Session Manager permissions \(p. 920\)](#)
- [Setting up Run Command \(p. 992\)](#)

Task 1: Add permissions to a Systems Manager instance profile (console)

Depending on whether you're 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 **Use case**, choose **EC2**, and then choose **Next**.
5. On the **Add permissions policies** page, do the following:
 - Use the **Search** field to locate the **AmazonSSMManagedInstanceCore** policy. Select the check box next to its name.



The console retains your selection even if you search for other policies.

- If you created a custom S3 bucket policy in the previous procedure, [Task 2: \(Optional\) Create a custom policy for S3 bucket access \(p. 24\)](#), search for it and select the check 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 check box next to its name.
 - If you plan to use EventBridge or CloudWatch Logs to manage or monitor your instance, search for **CloudWatchAgentServerPolicy** and select the check box next to its name.
6. Choose **Next**.
7. For **Role name**, enter a name for your new instance profile, such as **SSMInstanceProfile**.
- Note**
Make a note of the role name. You will choose this role when you create new instances that you want to manage by using Systems Manager.
8. (Optional) For **Description**, enter a description for this instance profile.
9. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this role, and then choose **Create role**. The system returns you to the **Roles** page.

To add instance profile permissions for Systems Manager to an existing role (console)

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**, and then choose the existing role you want to associate with an instance profile for Systems Manager operations.
3. On the **Permissions** tab, choose **Add permissions, Attach policies**.
4. On the **Attach policy** page, do the following:
 - Select the check box next to the required **AmazonSSMManagedInstanceCore** managed policy.
 - If you have created a custom S3 bucket policy, select the check box next to its name. For information about custom S3 bucket policies for an instance profile, see [Task 2: \(Optional\) Create a custom policy for S3 bucket access \(p. 24\)](#).
 - If you plan to join instances to an Active Directory managed by AWS Directory Service, select the check box next to **AmazonSSMDirectoryServiceAccess**.
 - If you plan to use EventBridge or CloudWatch Logs to manage or monitor your instance, select the check box next to **CloudWatchAgentServerPolicy**.
5. Choose **Attach policies**.

For information about how to update a role to include a trusted entity or further restrict access, see [Modifying a role](#) in the *IAM User Guide*.

Continue to [Step 5: Attach an IAM instance profile to an Amazon EC2 instance \(p. 26\)](#).

Task 2: (Optional) Create a custom policy for S3 bucket access

Creating a custom policy for Amazon S3 access is required only if you're 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 following policy, see [SSM Agent communications with AWS managed S3 buckets \(p. 138\)](#).

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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        1{
            "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:::aws-ssm-distributor-file-region/*",
                "arn:aws:s3:::aws-ssm-document-attachments-region/*",
                "arn:aws:s3:::patch-baseline-snapshot-region/*"
            ]
        },
        2{
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl", 3
                "s3:GetEncryptionConfiguration" 4
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET" 5
            ]
        }
    ]
}
```

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

² The second Statement element is required only if you're using an S3 bucket that you created to use in your Systems Manager operations.

³ The PutObjectAcl access control list permission is required only if you plan to support cross-account access to S3 buckets in other accounts.

⁴ The GetEncryptionConfiguration element is required if your S3 bucket is configured to use encryption.

⁵ If your S3 bucket is configured to use encryption, then the S3 bucket root (for example, arn:aws:s3:::**DOC-EXAMPLE-BUCKET**) must be listed in the Resource section. Your IAM user, group, or role must be configured with access to the root bucket.

4. If you're 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 aren't using a VPC endpoint in your operations, you can delete the first `Statement` element.

5. If you're using an S3 bucket of your own in your Systems Manager operations, do the following:

In the second `Statement` element, replace `DOC-EXAMPLE-BUCKET` 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 [Amazon S3 actions](#) in the *Amazon Simple Storage Service User 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. See the following example for permissions on buckets.

```
"Resource": [  
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*",  
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET2/*"  
]
```

-or-

If you aren't using an S3 bucket of your own in your Systems Manager operations, you can delete the second `Statement` element.

6. Choose **Next: Tags**.
7. (Optional) Add tags by choosing **Add tag**, and entering the preferred tags for the policy.
8. Choose **Next: Review**.
9. For **Name**, enter a name to identify this policy, such as **SSMInstanceProfileS3Policy**.
10. Choose **Create policy**.

Step 5: Attach an IAM instance profile to an Amazon EC2 instance

This topic describes how to attach the AWS Identity and Access Management (IAM) instance profile for AWS Systems Manager that you created in [Step 4: Create an IAM instance profile for Systems Manager \(p. 21\)](#) to Amazon EC2 instances. You can attach the instance profile to new Amazon EC2 instances when you launch them, or to existing Amazon EC2 instances.

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, 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, 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:

- Amazon Linux Base AMIs dated 2017.09 and later
- Amazon Linux 2
- Amazon Linux 2 ECS-Optimized Base AMIs
- macOS 10.14.x (Mojave), 10.15.x (Catalina), and 11.x (Big Sur)
- SUSE Linux Enterprise Server (SLES) 12 and 15
- Ubuntu Server 16.04, 18.04, and 20.04
- Windows Server 2008-2012 R2 AMIs published in November 2016 or later
- Windows Server 2016, 2019, and 2022

For information about manually installing SSM Agent on other Linux operating systems, see [Working with SSM Agent on EC2 instances for Linux \(p. 76\)](#).

TLS certificate requirement for instances

A Transport Layer Security (TLS) certificate must be installed on each managed instance you use with Systems Manager. These certificates are used to encrypt calls to other AWS services. A TLS certificate is already installed on each EC2 instance created from any AMI. On instances created from AMIs not supplied by Amazon, and on your own on-premises servers and VMs, install the certificate yourself. For more information, see [Install a TLS certificate on and VMs on-premises servers \(p. 819\)](#).

Topics

- [Launch an instance that uses the Systems Manager instance profile \(console\) \(p. 27\)](#)
- [Attach the Systems Manager instance profile to an existing instance \(console\) \(p. 28\)](#)

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**, **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** dropdown list, select the instance profile you created using the procedure in [Step 4: Create an IAM instance profile for Systems Manager \(p. 21\)](#).
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:** [Launch an instance using the Launch Instance Wizard](#) in the *Amazon EC2 User Guide for Linux Instances*
 - **Windows Server:** [Launch 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, 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. Navigate to and choose your EC2 instance from the list.
4. In the **Actions** menu, choose **Security, Modify 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 \(p. 21\)](#).
6. Choose **Save**.

For more information about attaching IAM roles to instances, choose one of the following, depending on your selected operating system type:

- [Attach an IAM role to an instance](#) in the *Amazon EC2 User Guide for Linux Instances*
- [Attach an IAM role to an instance](#) in the *Amazon EC2 User Guide for Windows Instances*

Continue to [Step 6: \(Optional\) Create a VPC endpoint \(p. 28\)](#).

Step 6: (Optional) Create a VPC 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). By using an interface VPC endpoint (interface endpoint), you can connect to services powered by AWS PrivateLink. AWS PrivateLink is a technology that allows you to privately access Amazon Elastic Compute Cloud (Amazon EC2) and Systems Manager APIs by using private IP addresses.

AWS PrivateLink restricts all network traffic between your managed instances, Systems Manager, and Amazon EC2 to the Amazon network. This means that your managed instances don't have access to the Internet. If you use AWS PrivateLink, you don't need an internet gateway, a NAT device, or a virtual private gateway.

You aren't required to configure AWS PrivateLink, but it's recommended. For more information about AWS PrivateLink and VPC endpoints, see [AWS PrivateLink and VPC endpoints](#).

Note

The alternative to using a VPC endpoint is to allow 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`

SSM Agent initiates all connections to the Systems Manager service in the cloud. For this reason, you don't need to configure your firewall to allow inbound traffic to your instances for Systems Manager.

For more information about calls to these endpoints, see [Reference: ec2messages, ssmmessages, and other API operations \(p. 1546\)](#).

About Amazon VPC

You can use Amazon Virtual Private Cloud (Amazon VPC) 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 the scalable infrastructure of AWS. 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 \(p. 29\)](#)
- [Creating VPC endpoints for Systems Manager \(p. 30\)](#)
- [Create an interface VPC endpoint policy \(p. 31\)](#)

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 unless you allow traffic from your instance to the public AWS Directory Service endpoints. This plugin requires the AWS Directory Service, and AWS Directory Service doesn't have PrivateLink endpoint support. Support for joining a Windows Server instance to a domain from other methods of joining instances to a domain depends 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 don't support cross-Region requests—ensure that you create your endpoint in the same AWS 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, `DOC-EXAMPLE-BUCKET.s3-us-west-2.amazonaws.com`. For more information about Region-specific endpoints for Amazon S3, see [Amazon S3 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 [VPC peering connections \(Quotas\)](#) in the [Amazon Virtual Private Cloud User Guide](#).

VPC gateway endpoint connections can't be extended out of a VPC. Resources on the other side of a VPC peering connection in your VPC can't 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 [VPC endpoints \(Quotas\)](#) 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 aren't allowed, then the managed instance can't connect to the SSM and EC2 endpoints.

Amazon S3 buckets

Your VPC endpoint policy must allow access to at least 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 from which the code is retrieved when a patch baseline document is run. If the code can't be downloaded, the patch baseline command will fail.

Note

If you use an on-premises firewall and plan to use Patch Manager, that firewall must also allow access to the appropriate patch baseline endpoint.

To provide access to the buckets in your AWS Region, include the following permission in your endpoint policy.

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

See the following example.

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

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`
- The S3 buckets listed in [SSM Agent communications with AWS managed S3 buckets \(p. 138\)](#).

Amazon CloudWatch Logs

If you don't allow your instances to access the internet, create a VPC endpoint for CloudWatch Logs to use features that send logs to CloudWatch Logs. For more information about creating an endpoint for CloudWatch Logs, see [Creating a VPC endpoint for CloudWatch Logs](#) in the *Amazon CloudWatch Logs User Guide*.

DNS in hybrid environment

For information about configuring DNS to work with AWS PrivateLink endpoints in hybrid environments, see [Private DNS for interface endpoints](#) in the *Amazon VPC User Guide*. 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 AWS Systems Manager. This topic links to procedures in the *Amazon VPC User Guide*.

To create VPC endpoints for Systems Manager

In the first step of this procedure, 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're 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 [Create 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 Amazon EBS volumes fails, which causes the Systems Manager command to fail.
 - `com.amazonaws.region.ssmmessages` – This endpoint is required only if you're connecting to your instances through a secure data channel using Session Manager. For more information, see [AWS Systems Manager Session Manager \(p. 912\)](#) and [Reference: ec2messages, ssmmessages, and other API operations \(p. 1546\)](#).
 - `com.amazonaws.region.kms` – This endpoint is optional. However, it can be created if you want to use AWS Key Management Service (AWS KMS) encryption for Session Manager or Parameter Store parameters.
 - `com.amazonaws.region.logs` – This endpoint is optional. However, it can be created if you want to use Amazon CloudWatch Logs (CloudWatch Logs) for Session Manager, Run Command, or SSM Agent logs.
2. Follow the steps in [Create 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 perform patching operations. Systems Manager also uses this endpoint 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. If the security group associated with your instances restricts outbound traffic, you must add a rule to allow traffic to the prefix list for Amazon S3. For more information, see [Modify your security group](#) in the *AWS PrivateLink Guide*.

Create an interface VPC endpoint policy

You can create policies for VPC interface endpoints for AWS Systems Manager in which you can specify:

- The principal that can perform actions
- The actions that can be performed
- The resources that can have actions performed on them

For more information, see [Control access to services with VPC endpoints](#) in the *Amazon VPC User Guide*.

Example: Allow users to only get and list command invocations

```
{  
    "Statement": [  
        {  
            "Sid": "SSMCommandsReadOnly",  
            "Principal": "*",  
            "Action": [  
                "ssm>ListCommands",  
                "ssm>ListCommandInvocations",  
                "ssm>GetCommandInvocation"  
            ],  
            "Effect": "Allow",  
            "Resource": "*"  
        }  
    ]  
}
```

Continue to [Step 7: \(Optional\) Create Systems Manager service roles \(p. 32\)](#).

Step 7: (Optional) Create Systems Manager service roles

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

Service role — A service role is a type of AWS Identity and Access Management (IAM) role 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 so that it can 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.

You can use the Systems Manager service-linked role 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 later in this topic.
- The Explorer capability uses the service-linked role to enable viewing OpsData and OpsItems from multiple accounts. This service-linked role also allows Explorer to create a managed rule when you enable Security Hub as a data source from Explorer or OpsCenter.

For more information about the service-linked role, see [Using service-linked roles for Systems Manager \(p. 1410\)](#).

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 permissions on a resource, but you want the user to run an Automation workflow that requires elevated permissions. In this scenario, you can create a service role with elevated permissions 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 \(p. 401\)](#)
- [Method 2: Use IAM to configure roles for Automation \(p. 403\)](#)

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 runbooks 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? \(p. 708\)](#)
- [\(Optional\) Create a custom service role for Maintenance Windows \(console\) \(p. 709\).](#)

Service role for Amazon SNS 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 \(p. 1462\)](#).

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 a *hybrid environment*, create an AWS Identity and Access Management (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 \(p. 36\)](#).

Continue to [Step 8: \(Optional\) Set up integrations with other AWS services \(p. 34\)](#).

Step 8: (Optional) Set up integrations with other AWS services

AWS Systems Manager integrates with many 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 \(p. 1488\)](#)
- [Using Chef InSpec profiles with Systems Manager Compliance \(p. 1498\)](#)

Tip

For a complete list of AWS services that integrate with Systems Manager, see [Integration with AWS services \(p. 1475\)](#).

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 EventBridge and Amazon Simple Notification Service – Using EventBridge, you can set up rules to detect when changes happen to AWS resources that you specify. You can configure EventBridge to log status execution changes of the commands that 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 an EventBridge event to invoke other actions in your AWS environment. For more information, see the following topics:

- [Understanding command statuses \(p. 1013\)](#)
- [Configuring EventBridge for Systems Manager events \(p. 1450\)](#)
- [Monitoring session activity using Amazon EventBridge \(console\) \(p. 977\)](#)

Amazon S3

Run Command command output in the Systems Manager console is truncated after 2,500 characters. To access complete command output logs, you can store Systems Manager output in an Amazon Simple Storage Service (Amazon S3) bucket. Then, you can 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 User Guide](#).

Amazon 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 \(p. 1447\)](#).

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, edge devices, and virtual machines (VMs) that are configured for AWS 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 Systems Manager to configure and manage their organization's on-premises machines.

Note

- Systems Manager supports edge devices that are configured as on-premises machines, including AWS IoT devices and non-AWS IoT devices. The process for setting up these types of edge devices is described here.

Systems Manager also supports edge devices that use AWS IoT Greengrass Core software. The setup process and requirements for AWS IoT Greengrass core devices is different than AWS IoT and non-AWS edge devices. To get started with AWS IoT Greengrass devices, see [Setting up AWS Systems Manager for edge devices \(p. 52\)](#).

- macOS isn't supported for Systems Manager hybrid environments.

If you plan to use Systems Manager to manage Amazon Elastic Compute Cloud (Amazon EC2) instances, or to use both Amazon EC2 instances and your own resources in a hybrid environment, follow the steps in [Setting up AWS Systems Manager for EC2 instances \(p. 16\)](#) first.

Configuring your hybrid environment for Systems Manager allows 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 machines by using AWS Identity and Access Management (IAM).
- Centralize auditing and your view into the actions performed on your machines 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 \(p. 1442\)](#).

- Centralize monitoring by configuring Amazon EventBridge and Amazon Simple Notification Service (Amazon SNS) to send notifications about service execution success.

For information about using EventBridge to monitor Systems Manager events, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#).

About managed nodes

After you finish configuring your on-premises servers, edge devices, and VMs for Systems Manager as described in this section, your hybrid machines are listed in the AWS Management Console and described as *managed nodes*. In the console, the IDs of your hybrid managed nodes are distinguished from Amazon EC2 instances with the prefix "mi-". Amazon EC2 instance IDs use the prefix "i-".

For more information, see [Managed nodes \(p. 804\)](#).

About instance tiers

Systems Manager offers a standard-instances tier and an advanced-instances tier for managed nodes in your hybrid environment. The standard-instances tier allows you to register a maximum of 1,000 on-premises machines per AWS account per AWS Region. If you need to register more than 1,000 on-premises machines in a single account and Region, then use the advanced-instances tier. Advanced instances also allow you to connect to your hybrid machines by using AWS Systems Manager Session Manager. Session Manager provides interactive shell access to your managed nodes.

For more information, see [Configuring instance tiers \(p. 805\)](#).

Topics

- [Step 1: Complete general Systems Manager setup steps \(p. 36\)](#)

- [Step 2: Create an IAM service role for a hybrid environment \(p. 36\)](#)
- [Step 3: Create a managed-instance activation for a hybrid environment \(p. 41\)](#)
- [Step 4: Install SSM Agent for a hybrid environment \(Linux\) \(p. 45\)](#)
- [Step 5: Install SSM Agent for a hybrid environment \(Windows\) \(p. 50\)](#)

Step 1: Complete general Systems Manager setup steps

If you haven't already done so, complete the following general setup steps for AWS Systems Manager in the [Setting up AWS Systems Manager for EC2 instances \(p. 16\)](#) section of this user guide. The other steps in that section are required only if you plan to manage Amazon Elastic Compute Cloud (Amazon EC2) instances.

- [Sign up for AWS \(p. 17\)](#)
- [Create an Admin IAM user for AWS \(p. 17\)](#)
- [Create non-Admin IAM users and groups for Systems Manager \(p. 18\)](#)
- [\(Optional\) Create a VPC endpoint \(p. 28\)](#)
- [\(Optional\) Create Systems Manager service roles \(p. 32\)](#)
- [\(Optional\) Set up integrations with other AWS services \(p. 34\)](#)

After ensuring that you have completed those steps, continue to [Step 2: Create an IAM service role for a hybrid environment \(p. 36\)](#).

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

Servers and virtual machines (VMs) in a hybrid environment require an AWS Identity and Access Management (IAM) role to communicate with the AWS Systems Manager service. The role grants AWS Security Token Service (AWS STS) [AssumeRole](#) trust to the Systems Manager service. You only need to create a service role for a hybrid environment once for each AWS account. However, you might choose to create multiple service roles for different hybrid activations if machines in your hybrid environment require different permissions.

The following procedures describe how to create the required service role using the Systems Manager console or your preferred command line tool.

Create an IAM service role (console)

Use the following procedure to create a service role for hybrid activation. Please note that this procedure uses the [AmazonSSMManagedInstanceCore](#) policy for Systems Manager core functionality. Depending on your use case, you might need to add additional policies to your service role for your on-premises machines to be able to access other capabilities or AWS services. For example, without access to the required AWS managed Amazon Simple Storage Service (Amazon S3) buckets, Patch Manager patching operations fail.

To create a 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**.
5. In the list of policies, select the box next to **AmazonSSMManagedInstanceCore**, and then choose **Next**.
6. In **Role name**, enter a name that identifies this role as a hybrid activation service role. For example: **my-hybrid-service-role**.
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. (Optional) Change the default role description to reflect the purpose of this role. For example: **Provides permissions for on-premises machines**.
9. Choose **Create role**. The system returns you to the **Roles** page.

Create an IAM service role (command line)

Use the following procedure to create a service role for hybrid activation. Please note that this procedure uses the **AmazonSSMManagedInstanceCore** policy Systems Manager core functionality. Depending on your use case, you might need to add additional policies to your service role for your on-premises machines to be able to access other capabilities or AWS services.

S3 bucket policy requirement

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

- **Case 1:** You're using a VPC endpoint to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink.
- **Case 2:** You plan to use an Amazon 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 Amazon S3 bucket. Before proceeding, follow the steps in [Create a custom S3 bucket policy for an instance profile \(p. 24\)](#). 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. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. On your local machine, 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. Be sure to specify your AWS account and the AWS Region in the ARN where you created your hybrid activation.

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

```
        "Condition":{  
            "StringEquals":{  
                "aws:SourceAccount":"123456789012"  
            },  
            "ArnEquals":{  
                "aws:SourceArn":"arn:aws:ssm:us-east-2:123456789012:/*"  
            }  
        }  
    }  
}
```

3. Open the AWS CLI, and in the directory where you created the JSON file, run 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 & macOS

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

Windows

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

4. Run the [attach-role-policy](#) command as follows to allow 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 Amazon Elastic Compute Cloud (Amazon EC2) instances. For more information about the AWS policies used in the following commands, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#).

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

Linux & macOS

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

Windows

```
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 allow AWS Systems Manager Agent (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 & macOS

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

Windows

```
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 & macOS

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

Windows

```
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 services such as Amazon EventBridge or Amazon CloudWatch Logs.

```
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. Install and configure the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. On your local machine, 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. Be sure to specify your AWS account and the AWS Region in the ARN where you created your hybrid activation.

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

```

    "Statement": [
        {
            "Sid":"",
            "Effect":"Allow",
            "Principal":{
                "Service":"ssm.amazonaws.com"
            },
            "Action":"sts:AssumeRole",
            "Condition":{
                "StringEquals":{
                    "aws:SourceAccount":"123456789012"
                },
                "ArnEquals":{
                    "aws:SourceArn":"arn:aws:ssm:region:123456789012:*"
                }
            }
        }
    ]
}

```

3. Open PowerShell in administrative mode, and in the directory where you created the JSON file, run [New-IAMRole](#) as follows to create a service role. This example creates a role named SSMServiceRole. You can choose another name if you prefer.

```

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

```

4. Use [Register-IAMRolePolicy](#) as follows to allow 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 \(p. 21\)](#).

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

```

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 allow 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.

```

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.

```

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 services such as Amazon EventBridge or Amazon CloudWatch Logs.

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

Continue to [Step 3: Create a managed-instance activation for a hybrid environment \(p. 41\)](#).

Step 3: 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 AWS Systems Manager 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 Fleet Manager page and **Managed Instances** tab of the Systems Manager console, by using the AWS CLI command [deregister-managed-instance](#), or by using the API call [DeregisterManagedInstance](#).

About activation tags

If you create an activation by using either the AWS Command Line Interface (AWS CLI) or AWS Tools for Windows PowerShell, you can specify tags. Tags are optional metadata that you assign to a resource. Tags allow 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 optional tags.

```
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're 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 nodes \(p. 1515\)](#).

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 nodes in a hybrid environment \(p. 815\)](#).

Topics

- [Create an activation \(console\) \(p. 42\)](#)
- [Create a managed instance activation \(command line\) \(p. 43\)](#)

Create an activation (console)

To create a managed-instance activation

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Hybrid Activations**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Hybrid Activations**.

3. Choose **Create activation**.
4. (Optional) In the **Activation description** field, enter a description for this activation. The description is optional, but we recommend that you enter a description if you plan to activate large numbers of servers and VMs.
5. In the **Instance limit** field, specify the total number of on-premises servers or VMs that you want to register with AWS as part of this activation. The default value is 1 instance.
6. In the **IAM role name** section, choose a service role option that allows your servers and VMs to communicate with AWS Systems Manager in the cloud:
 - a. Choose **Use the system created default command execution role** to use a role and managed policy created by AWS.
 - b. Choose **Select an existing custom IAM role that has the required permissions** to use the optional custom role you created earlier. This role must have a trust relationship policy that specifies "Service": "ssm.amazonaws.com". If your IAM role doesn't specify this principle in a trust relationship policy, you receive the following error:

```
An error occurred (ValidationException) when calling the CreateActivation operation: Not existing role: arn:aws:iam:<accountid>:role/SSMRole
```

For more information about creating this role, see [Step 2: Create an IAM service role for a hybrid environment \(p. 36\)](#).

7. In the **Activation expiry date** field, specify an expiration date for the activation. The expiry date must be in the future, and not more than 30 days into the future. The default value is 24 hours.

Note

If you want to register additional managed instances after the expiry date, you must create a new activation. The expiry date has no impact on registered and running instances.

8. (Optional) In the **Default instance name** field, specify a name.
9. Choose **Create activation**. Systems Manager immediately returns the Activation Code and ID to the console.

Create a managed instance activation (command line)

The following procedure describes how to use the AWS Command Line Interface (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 haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to create an activation.

Note

- In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).
- The role you specify for the `iam-role` parameter must have a trust relationship policy that specifies "Service": "ssm.amazonaws.com". If your AWS Identity and Access Management (IAM) role doesn't specify this principle in a trust relationship policy, you receive the following error:

```
An error occurred (ValidationException) when calling the CreateActivation
operation: Not existing role:
arn:aws:iam::<accountid>:role/SSMRole
```

For more information about creating this role, see [Step 2: Create an IAM service role for a hybrid environment \(p. 36\)](#).

- For `--expiration-date`, provide a date in timestamp format, such as "2021-07-07T00:00:00", for when the activation code expires. You can specify a date up to 30 days in advance. If you don't provide an expiration date, the activation code expires in 24 hours.

Linux & macOS

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

Windows

```
aws ssm create-activation ^
--default-instance-name name ^
--iam-role iam-service-role-name ^
```

```
--registration-limit number-of-managed-instances ^
--region region ^
--expiration-date "timestamp" ^
--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 ^
-ExpirationDate "timestamp" ^
-Tag @{"Key"="key-name-1";"Value"="key-value-1"},@{"Key"="key-name-2";"Value"="key-value-2"}
```

Here is an example.

Linux & macOS

```
aws ssm create-activation \
--default-instance-name MyWebServers \
--iam-role service-role/AmazonEC2RunCommandRoleForManagedInstances \
--registration-limit 10 \
--region us-east-2 \
--expiration-date "2021-07-07T00:00:00" \
--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 ^
--expiration-date "2021-07-07T00:00:00" ^
--tags "Key=Environment,Value=Production" "Key=Department,Value=Finance"
```

PowerShell

```
New-SSMActivation -DefaultInstanceName MyWebServers ^
-IamRole service-role/AmazonEC2RunCommandRoleForManagedInstances ^
-RegistrationLimit 10 ^
-Region us-east-2 ^
-ExpirationDate "2021-07-07T00:00:00" ^
-Tag
@{ "Key"="Environment";"Value"="Production"},@{ "Key"="Department";"Value"="Finance"}
```

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

Step 4: Install SSM Agent for a hybrid environment (Linux)

This topic describes how to install AWS Systems Manager SSM Agent on Linux machines in a hybrid environment. If you plan to use Windows Server machines in a hybrid environment, see the next step, [Step 5: Install SSM Agent for a hybrid environment \(Windows\) \(p. 50\)](#).

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 [Working with SSM Agent on EC2 instances for Linux \(p. 76\)](#).

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 3: Create a managed-instance activation for a hybrid environment \(p. 41\)](#). 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 `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 East (Ohio) Region (`us-east-2`), use the following URL:

```
https://s3.us-east-2.amazonaws.com/amazon-ssm-us-west-1/latest/linux_amd64/amazon-ssm-agent.rpm
```

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

- **x86_64**

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

- **x86**

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

- **ARM64**

```
https://s3.region.amazonaws.com/amazon-ssm-region/latest/linux_arm64/amazon-ssm-agent.rpm
```

Ubuntu Server

- **x86_64**

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

- **ARM64**

```
https://s3.region.amazonaws.com/amazon-ssm-region/latest/debian_arm64/amazon-ssm-agent.deb
```

- **x86**

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

Debian Server

- **x86_64**

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

- **ARM64**

```
https://s3.region.amazonaws.com/amazon-ssm-region/latest/debian_arm64/  
amazon-ssm-agent.deb
```

Raspberry Pi OS (formerly Raspbian)

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

## 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. If you use an HTTP or HTTPS proxy, you must set the `http_proxy` or `https_proxy` environment variables in the current shell session. If you aren't using a proxy, you can skip this step.

For an HTTP proxy server, enter the following commands at the command line:

```
export http_proxy=http://hostname:port
export https_proxy=https://hostname:port
```

For an HTTPS proxy server, enter the following commands at the command line:

```
export http_proxy=http://hostname:port
export https_proxy=https://hostname:port
```

3. 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, then press **Enter**.

### Note

Note the following important details:

- `sudo` isn't necessary if you're a root user.
- Each command block specifies `sudo -E amazon-ssm-agent`. The `-E` is only necessary if you set an HTTP or HTTPS proxy environment variable.
- Even though the following URLs show 'ec2-downloads-windows', these are the correct URLs for Linux operating systems.

**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*.

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

```
mkdir /tmp/ssm
curl https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-
agent.rpm -o /tmp/ssm/amazon-ssm-agent.rpm
sudo yum install -y /tmp/ssm/amazon-ssm-agent.rpm
sudo stop amazon-ssm-agent
sudo -E 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

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

## RHEL 8.x and CentOS 8.x

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

## Debian Server

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

## Raspberry Pi OS (formerly Raspbian)

```
mkdir /tmp/ssm
sudo curl https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_arm/amazon-
ssm-agent.deb -o /tmp/ssm/amazon-ssm-agent.deb
sudo dpkg -i /tmp/ssm/amazon-ssm-agent.deb
sudo service amazon-ssm-agent stop
sudo -E 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 -E 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
curl https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_amd64/amazon-ssm-agent.deb -o /tmp/ssm/amazon-ssm-agent.deb
sudo dpkg -i /tmp/ssm/amazon-ssm-agent.deb
sudo service amazon-ssm-agent stop
sudo -E 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>.

On Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS, 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` directory to the `/etc/amazon/ssm/` directory. Log and library files haven't changed (`/var/lib/amazon/ssm`, `/var/log/amazon/ssm`).

```
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
```

### Important

The *candidate* channel in the Snap store contains the latest version of SSM Agent; not the stable channel. If you want to track SSM Agent version information on the candidate channel, run the following command on your Ubuntu Server 18.04 and 16.04 LTS 64-bit instances.

```
sudo snap switch --channel=candidate amazon-ssm-agent
```

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 Systems Manager service. The server or VM is now a managed instance. Amazon EC2 instances configured for Systems Manager are also managed instances. In the Systems Manager console, however, your on-premises instances are distinguished from Amazon EC2 instances with the prefix "mi-".

[Continue to Step 5: Install SSM Agent for a hybrid environment \(Windows\) \(p. 50\).](#)

## Setting up private key auto rotation

To strengthen your security posture, you can configure AWS Systems Manager Agent (SSM Agent) to rotate the hybrid environment private key automatically. You can access this feature using SSM Agent version 3.0.1031.0 or later. Turn on this feature using the following procedure.

### To configure SSM Agent to rotate the hybrid environment private key

1. Navigate to `/etc/amazon/ssm/` on a Linux machine or `C:\Program Files\Amazon\SSM` for a Windows machine.
2. Copy the contents of `amazon-ssm-agent.json.template` to a new file named `amazon-ssm-agent.json`. Save `amazon-ssm-agent.json` in the same directory where `amazon-ssm-agent.json.template` is located.
3. Find `Profile`, `KeyAutoRotateDays`. Enter the number of days that you want between automatic private key rotations.
4. Restart SSM Agent.

Every time you change the configuration, restart SSM Agent.

You can customize other features of SSM Agent using the same procedure. For an up-to-date list of the available configuration properties and their default values, see [Config Property Definitions](#).

## Deregister and reregister a managed instance

You can deregister a managed instance by calling the [DeregisterManagedInstance](#) API operation 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"
```

You can reregister a managed instance after you deregistered it. Use the following procedure to reregister a managed instance. After you complete the procedure, your managed instance is displayed again in the list of managed instances.

### To reregister a managed instance on a Linux hybrid machine

1. Connect to your instance.
2. Run the following command. Be sure to 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 Region you want to download the SSM Agent from.

```
echo "yes" | sudo amazon-ssm-agent -register -code "activation-code" -id "activation-id" -region "region" && sudo systemctl restart amazon-ssm-agent
```

## Troubleshooting SSM Agent installation in a Linux hybrid environment

Use the following information to help you troubleshoot problems installing SSM Agent in a Linux hybrid environment.

### You receive DeliveryTimedOut error

**Problem:** While configuring an Amazon EC2 instance in one AWS account as a managed instance for a separate AWS account, you receive `DeliveryTimedOut` after running the commands to install SSM Agent on the target instance.

**Solution:** `DeliveryTimedOut` is the expected response code for this scenario. The command to install SSM Agent on the target instance changes the instance ID of the source instance. Because the instance ID has changed, the source instance isn't able to reply to the target instance that the command failed, completed, or timed out while executing.

## Unable to load instance associations

**Problem:** After running the install commands, you see the following error in the SSM Agent error logs:

```
Unable to load instance associations, unable to retrieve associations unable
to retrieve associations error occurred in RequestManagedInstanceRoleToken:
MachineFingerprintDoesNotMatch: Fingerprint doesn't match
```

You see this error when the machine ID doesn't persist after a reboot.

**Solution:** To solve this problem, run the following command. This command forces the machine ID to persist after a reboot.

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

# Step 5: 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 previous step, [Step 4: Install SSM Agent for a hybrid environment \(Linux\) \(p. 45\)](#).

### 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 [Working with SSM Agent on EC2 instances for Windows Server \(p. 123\)](#).

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 3: Create a managed-instance activation for a hybrid environment \(p. 41\)](#). 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. If you use an HTTP or HTTPS proxy, you must set the `http_proxy` or `https_proxy` environment variables in the current shell session. If you aren't using a proxy, you can skip this step.

For an HTTP proxy server, set this variable:

```
http_proxy=http://hostname:port
https_proxy=https://hostname:port
```

For an HTTPS proxy server, set this variable:

```
http_proxy=http://hostname:port
https_proxy=https://hostname:port
```

3. Open Windows PowerShell in elevated (administrative) mode.

4. Copy and paste the following command block into 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.

`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*.

64-bit

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

32-bit

```
$code = "activation-code"
$id = "activation-id"
$region = "region"
$dir = $env:TEMP + "\ssm"
New-Item -ItemType directory -Path $dir -Force
cd $dir
(New-Object System.Net.WebClient).DownloadFile("https://amazon-ssm-$region.s3.amazonaws.com/latest/windows_386/AmazonSSMAgentSetup.exe", $dir + "\AmazonSSMAgentSetup.exe")
Start-Process .\AmazonSSMAgentSetup.exe -ArgumentList @("/q", "/log", "install.log", "CODE=$code", "ID=$id", "REGION=$region") -Wait
Get-Content ($env:ProgramData + "\Amazon\SSM\InstanceData\registration")
Get-Service -Name "AmazonSSMAgent"
```

5. Press **Enter**.

The command does the following:

- Downloads and installs SSM Agent onto the server or VM.
- Registers the server or VM with the Systems Manager 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 Fleet Manager console, by using the AWS CLI command [describe-instance-information](#), or by using the API command [DescribeInstancePatches](#).

## Setting up private key auto rotation

To strengthen your security posture, you can configure AWS Systems Manager Agent (SSM Agent) to rotate the hybrid environment private key automatically. You can access this feature using SSM Agent version 3.0.1031.0 or later. Turn on this feature using the following procedure.

### To configure SSM Agent to rotate the hybrid environment private key

1. Navigate to `/etc/amazon/ssm/` on a Linux machine or `C:\Program Files\Amazon\SSM` for a Windows machine.
2. Copy the contents of `amazon-ssm-agent.json.template` to a new file named `amazon-ssm-agent.json`. Save `amazon-ssm-agent.json` in the same directory where `amazon-ssm-agent.json.template` is located.
3. Find `Profile`, `KeyAutoRotateDays`. Enter the number of days that you want between automatic private key rotations.
4. Restart SSM Agent.

Every time you change the configuration, restart SSM Agent.

You can customize other features of SSM Agent using the same procedure. For an up-to-date list of the available configuration properties and their default values, see [Config Property Definitions](#).

## Deregister and reregister a managed instance

You can deregister a managed instance by calling the [DeregisterManagedInstance](#) API operation 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"
```

You can reregister a managed instance after you deregistered it. Use the following procedure to reregister a managed instance. After you complete the procedure, your managed instance is displayed again in the list of managed instances.

### To reregister a managed instance on a Windows hybrid machine

1. Connect to your instance.
2. Run the following command. Be sure to 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 Region you want to download the SSM Agent from.

```
'yes' | & 'C:\Program Files\Amazon\SSM\amazon-ssm-agent.exe' -register -
code activation-code -id activation-id -region region; Restart-Service AmazonSSMAgent
```

## Setting up AWS Systems Manager for edge devices

This section describes the setup tasks that account and system administrators perform to enable configuration and management of AWS IoT Greengrass core devices. After you complete these tasks, users who have been granted permissions by the AWS account administrator can use AWS Systems Manager to configure and manage their organization's AWS IoT Greengrass core devices.

### Note

- SSM Agent for AWS IoT Greengrass isn't supported on macOS and Windows 10. You can't use Systems Manager capabilities to manage and configure edge devices that use these operating systems.
- Systems Manager also supports edge devices that aren't configured as AWS IoT Greengrass core devices. To use Systems Manager to manage AWS IoT Core devices and non-AWS edge devices, you must configure them as on-premises machines in a hybrid environment. For more information, see [Setting up AWS Systems Manager for hybrid environments \(p. 34\)](#).
- To use Session Manager and Microsoft application patching with your edge devices, you must enable the advanced-instances tier. For more information, see [Turning on the advanced-instances tier \(p. 805\)](#).

### Before you begin

Verify that your edge devices meet the following requirements.

- Your edge devices must meet the requirements to be configured as AWS IoT Greengrass core devices. For more information, see [Setting up AWS IoT Greengrass core devices in the AWS IoT Greengrass Version 2 Developer Guide](#).
- Your edge devices must be compatible with AWS Systems Manager Agent (SSM Agent). For more information, see [Supported operating systems \(p. 9\)](#).
- Your edge devices must be able to communicate with the Systems Manager service in the cloud. Systems Manager doesn't support disconnected edge devices.

### About setting up edge devices

Setting up AWS IoT Greengrass devices for Systems Manager involves the following processes.

| Step                                                                                                                         | Details                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Step 1: Complete general Systems Manager setup steps (p. 54)</a>                                                 | Complete all of the general requirements for setting up and configuring Systems Manager. If you completed these steps already, see Step 2.                                                                                                                                                                                                                                                             |
| <a href="#">Step 2: Create an IAM service role for edge devices (p. 55)</a>                                                  | Create an AWS Identity and Access Management (IAM) service role that enables your AWS IoT Greengrass devices to communicate with Systems Manager. If you previously configured on-premises servers and virtual machines in a hybrid environment for Systems Manager then you might have completed this step already.                                                                                   |
| <a href="#">Step 3: Set up AWS IoT Greengrass (p. 58)</a>                                                                    | You must set up your edge devices as AWS IoT Greengrass core devices. The setup process involves verifying supported operating systems and system requirements, as well as installing and configuring the AWS IoT Greengrass Core software on your devices. For more information, see <a href="#">Setting up AWS IoT Greengrass core devices in the AWS IoT Greengrass Version 2 Developer Guide</a> . |
| <a href="#">Step 4: Update the AWS IoT Greengrass token exchange role and install SSM Agent on your edge devices (p. 59)</a> | The final step for setting up and configuring your AWS IoT Greengrass core devices for Systems Manager requires you to update the AWS IoT                                                                                                                                                                                                                                                              |

| Step | Details                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | <p>Greengrass IAM service role, also called the <i>token exchange role</i>, and deploy AWS Systems Manager Agent (SSM Agent) to your AWS IoT Greengrass devices. Both processes are described in detail in the <i>AWS IoT Greengrass Version 2 Developer Guide</i>. For more information, see <a href="#">Install AWS Systems Manager SSM Agent</a>.</p> <p>AWS Systems Manager Agent (SSM Agent) makes it possible for Systems Manager to update, manage, and configure your edge devices. To deploy SSM Agent to your AWS IoT Greengrass devices, use Greengrass to deploy the <code>aws.greengrass.SystemsManagerAgent</code> component to your devices. After you deploy SSM Agent to your devices, AWS IoT Greengrass automatically registers your devices with Systems Manager. No additional registration is necessary. You can begin using Systems Manager capabilities to access, manage, and configure your AWS IoT Greengrass devices.</p> |

#### Note

For information about uninstalling SSM Agent from an edge device, see [Uninstall the AWS Systems Manager Agent](#) in the *AWS IoT Greengrass Version 2 Developer Guide*.

#### Topics

- [Step 1: Complete general Systems Manager setup steps \(p. 54\)](#)
- [Step 2: Create an IAM service role for edge devices \(p. 55\)](#)
- [Step 3: Set up AWS IoT Greengrass \(p. 58\)](#)
- [Step 4: Update the AWS IoT Greengrass token exchange role and install SSM Agent on your edge devices \(p. 59\)](#)

## Step 1: Complete general Systems Manager setup steps

If you haven't already done so, complete the following general setup steps for AWS Systems Manager.

- [Sign up for AWS \(p. 17\)](#)
- [Create an Admin IAM user for AWS \(p. 17\)](#)
- [Create non-Admin IAM users and groups for Systems Manager \(p. 18\)](#)
- [\(Optional\) Create a VPC endpoint \(p. 28\)](#)
- [\(Optional\) Create Systems Manager service roles \(p. 32\)](#)
- [\(Optional\) Set up integrations with other AWS services \(p. 34\)](#)

After you've completed these steps, continue to [Step 2: Create an IAM service role for edge devices \(p. 55\)](#).

## Step 2: Create an IAM service role for edge devices

AWS IoT Greengrass core devices require an AWS Identity and Access Management (IAM) service role to communicate with AWS Systems Manager. The role grants AWS Security Token Service (AWS STS) [AssumeRole](#) trust to the Systems Manager service. You only need to create the service role once for each AWS account. You will specify this role for the `RegistrationRole` parameter when you configure and deploy the SSM Agent component to your AWS IoT Greengrass devices. If you already created this role while setting up on-premises servers or virtual machines in a hybrid environment, you can skip this step.

### Note

Users in your company or organization who will use Systems Manager on your edge devices must be granted permission in IAM to call the Systems Manager API. For more information, see [Create non-Admin IAM users and groups for Systems Manager \(p. 18\)](#).

### S3 bucket policy requirement

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

- **Case 1:** You're using a VPC endpoint to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink.
- **Case 2:** You plan to use an Amazon 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 Amazon S3 bucket. Before proceeding, follow the steps in [Create a custom S3 bucket policy for an instance profile \(p. 24\)](#). The information about S3 bucket policies in that topic also applies to your service role.

### Note

If your devices are protected by a firewall and you plan to use Patch Manager, the firewall must allow access to the patch baseline endpoint `arn:aws:s3:::patch-baseline-snapshot-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](#).

### AWS CLI

#### To create an IAM service role for an AWS IoT Greengrass environment (AWS CLI)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. On your local machine, 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.

### Note

Make a note of the name. You will specify it when you deploy SSM Agent to your AWS IoT Greengrass core devices.

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

```
}
```

3. Open the AWS CLI, and in the directory where you created the JSON file, run the [create-role](#) command to create the service role.

#### Linux & macOS

```
aws iam create-role \
--role-name SSMSERVICEROLE \
--assume-role-policy-document file://SSMSERVICE-TRUST.json
```

#### Windows

```
aws iam create-role ^
--role-name SSMSERVICEROLE ^
--assume-role-policy-document file://SSMSERVICE-TRUST.json
```

4. Run the [attach-role-policy](#) command as follows to allow the service role you just created to create a session token. The session token gives your edge devices permission to run commands using Systems Manager.

#### Note

The policies you add for a service profile for edge devices are the same policies used to create an instance profile for Amazon Elastic Compute Cloud (Amazon EC2) instances. For more information about the AWS policies used in the following commands, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#).

(Required) Run the following command to allow an edge device to use AWS Systems Manager service core functionality.

#### Linux & macOS

```
aws iam attach-role-policy \
--role-name SSMSERVICEROLE \
--policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
```

#### Windows

```
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 allow AWS Systems Manager Agent (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 & macOS

```
aws iam attach-role-policy \
--role-name SSMSERVICEROLE \
--policy-arn arn:aws:iam::account_ID:policy/my_bucket_policy_name
```

#### Windows

```
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 from edge devices. The service role needs this policy only if you join your edge devices to a Microsoft AD directory.

#### Linux & macOS

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

#### Windows

```
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 edge devices. This command makes it possible to read information on a device and write it to CloudWatch. Your service role needs this policy only if you will use services such as Amazon EventBridge or Amazon CloudWatch Logs.

```
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 an AWS IoT Greengrass environment (AWS Tools for Windows PowerShell)

1. Install and configure the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. On your local machine, 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.

##### Note

Make a note of the name. You will specify it when you deploy SSM Agent to your AWS IoT Greengrass core devices.

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

3. Open PowerShell in administrative mode, and in the directory where you created the JSON file, run `New-IAMRole` as follows to create a service role.

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

4. Use [Register-IAMRolePolicy](#) as follows to allow the service role you created to create a session token. The session token gives your edge devices permission to run commands using Systems Manager.

**Note**

The policies you add for a service role for edge devices in an AWS IoT Greengrass 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 \(p. 21\)](#).

(Required) Run the following command to allow an edge device to use AWS Systems Manager service core functionality.

```
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 allow 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.

```
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 from edge devices. The service role needs this policy only if you join your edge devices to a Microsoft AD directory.

```
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 edge devices. This command makes it possible to read information on a device and write it to CloudWatch. Your service role needs this policy only if you will use services such as Amazon EventBridge or Amazon CloudWatch Logs.

```
Register-IAMRolePolicy `
 -RoleName $SSMServiceRole`
 -PolicyArn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
```

[Continue to Step 3: Set up AWS IoT Greengrass \(p. 58\).](#)

## Step 3: Set up AWS IoT Greengrass

Set up your edge devices as AWS IoT Greengrass core devices. The setup process involves verifying supported operating systems and system requirements, as well as installing and configuring the AWS IoT Greengrass Core software on your devices. For more information, see [Setting up AWS IoT Greengrass core devices](#) in the [AWS IoT Greengrass Version 2 Developer Guide](#).

Continue to [Step 4: Update the AWS IoT Greengrass token exchange role and install SSM Agent on your edge devices \(p. 59\)](#).

## Step 4: Update the AWS IoT Greengrass token exchange role and install SSM Agent on your edge devices

The final step for setting up and configuring your AWS IoT Greengrass core devices for Systems Manager requires you to update the AWS IoT Greengrass AWS Identity and Access Management (IAM) device service role, also called the *token exchange role*, and deploy AWS Systems Manager Agent (SSM Agent) to your AWS IoT Greengrass devices. Both processes are described in detail in the *AWS IoT Greengrass Version 2 Developer Guide*. For more information, see [Install the AWS Systems Manager Agent](#).

After you deploy SSM Agent to your devices, AWS IoT Greengrass automatically registers your devices with Systems Manager. No additional registration is necessary. You can begin using Systems Manager capabilities to access, manage, and configure your AWS IoT Greengrass devices.

**Note**

Your edge devices must be able to communicate with the Systems Manager service in the cloud. Systems Manager doesn't support disconnected edge devices.

# 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 \(p. 16\)](#), [Setting up AWS Systems Manager for hybrid environments \(p. 34\)](#), or [Setting up AWS Systems Manager for edge devices \(p. 52\)](#).

## 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're managing Amazon Elastic Compute Cloud (Amazon 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 \(p. 16\)](#)
  - [Setting up AWS Systems Manager for hybrid environments \(p. 34\)](#)
  - [Setting up AWS Systems Manager for edge devices \(p. 52\)](#)

When you're ready, continue with the following steps.

## Topics

- [Step 1: Install or upgrade AWS command line tools \(p. 60\)](#)
- [Step 2: Practice installing or updating SSM Agent on an instance \(p. 62\)](#)
- [Step 3: Try Systems Manager tutorials and walkthroughs \(p. 63\)](#)

## Step 1: Install or upgrade AWS command line tools

This topic is for users who have *programmatic access* to use AWS Systems Manager (or any other AWS service), and who want to run AWS Command Line Interface (AWS CLI) or AWS Tools for Windows PowerShell commands from their local machines. (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 \(p. 18\)](#).)

### Tip

As an alternative to running commands from your local machine, you can use AWS CloudShell. CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. You can run AWS CLI commands against AWS services using your preferred shell (Bash, PowerShell, or Z shell). And you can do this without needing to download or install command line tools. For more information, see the [AWS CloudShell User Guide](#).

## Topics

- [Installing or upgrading and then configuring the AWS CLI \(p. 61\)](#)
- [Installing or upgrading and then configuring the AWS Tools for PowerShell \(p. 61\)](#)

# Installing or upgrading and then configuring the AWS CLI

The AWS CLI is an open source tool that allows you to interact with AWS services using commands in your command-line shell. With minimal configuration, the AWS CLI allows you to start running commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from the command prompt in your terminal program.

For more information about the AWS CLI, see the [AWS Command Line Interface 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](#).

## Important

Starting January 10th, 2020, AWS CLI version 1.17 and later no longer support Python 2.6 or Python 3.3. Since that date, the installer for the AWS CLI requires Python 2.7, Python 3.4, or a later version.

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

1. Follow the instructions in [Installing the AWS Command Line Interface version 2](#) 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 AWS CLI periodically to verify 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're 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 AWS 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.

```
aws ssm help
```

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

# Installing or upgrading and then configuring the AWS Tools for PowerShell

The AWS Tools for PowerShell are a set of PowerShell modules that are built on the functionality exposed by the AWS SDK for .NET. The AWS Tools for PowerShell allow you to script operations on

your AWS resources from the PowerShell command line. The cmdlets provide an idiomatic PowerShell experience for specifying parameters and handling results even though they are implemented using the various AWS service HTTP query APIs.

For information about the 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 Tools for PowerShell, see the [Systems Manager section of the AWS Tools for PowerShell Cmdlet Reference](#).

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

1. Follow the instructions in [Installing the Tools for PowerShell](#) in the *AWS Tools for Windows PowerShell User Guide* to install or upgrade Tools for PowerShell on your local machine.

**Tip**

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

2. To configure Tools for PowerShell, see [Using AWS Credentials](#) in the *AWS Tools for Windows PowerShell User Guide*.

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

```
Set-AWSCredential `~
-AccessKey AKIAIOSFODNN7EXAMPLE `~
-SecretKey wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY `~
-StoreAs MyProfileName
```

**Important**

When you configure 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 Tools for 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 runs on Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and on-premises servers and virtual machines (VMs). 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 Amazon Elastic Compute Cloud (Amazon EC2) instance.

**Note**

If you're working with your own on-premises servers or VMs, see the following topics:

- [Install SSM Agent for a hybrid environment \(Windows\) \(p. 50\)](#)
- [Install SSM Agent for a hybrid environment \(Linux\) \(p. 45\)](#)

macOS isn't supported for AWS Systems Manager hybrid environments.

**Prerequisites**

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

- Create an Amazon EC2 instance profile for Systems Manager: [Create an IAM instance profile for Systems Manager \(p. 21\)](#)
- Attach the instance profile to an Amazon EC2 instance when you create the instance: [Attach an IAM instance profile to an EC2 instance \(p. 26\)](#)
- Attach the instance profile to an existing Amazon EC2 instance: [Attach an IAM role to an instance](#) in the [Amazon EC2 User Guide](#)

**Windows Server instance**

To practice installing or updating SSM Agent on an Amazon EC2 instance for Windows Server, follow the steps in [Manually installing SSM Agent on EC2 instances for Windows Server \(p. 123\)](#).

**Linux instance**

To practice installing or updating SSM Agent on an Amazon EC2 instance for Linux, follow the steps for your Linux operating system type in [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).

**macOS instance**

To practice installing or updating SSM Agent on an Amazon EC2 instance for macOS, follow the steps in [Working with SSM Agent on EC2 instances for macOS \(p. 121\)](#).

## 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 AWS Systems Manager.

Because Systems Manager is a collection of multiple 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.

Usually, you don't 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.

Sometimes, additional configuration, setup, or experience with Systems Manager are required before you try a tutorial or walkthrough. We've identified those tutorials and walkthroughs as "Advanced".

**Categories**

- [Operations management \(p. 64\)](#)
- [Application management \(p. 64\)](#)
- [Change management \(p. 64\)](#)
- [Node management \(p. 65\)](#)
- [Shared resources \(p. 67\)](#)

## Operations management

### Explorer

[Explorer \(p. 157\)](#), a capability of AWS Systems Manager, 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 AWS Regions.

- [Customizing the display and using filters \(p. 173\)](#)

### OpsCenter

[OpsCenter \(p. 180\)](#), a capability of AWS Systems Manager, provides a central location where operations engineers and IT professionals can view, investigate, and resolve operational work items (OpsItems) related to AWS resources.

- [Configuring CloudWatch to create OpsItems from alarms \(p. 196\)](#)
- [Configuring EventBridge to automatically create OpsItems for specific events \(p. 203\)](#)

## Application management

### Application Manager

[Application Manager \(p. 231\)](#), a capability of AWS Systems Manager, helps DevOps engineers investigate and remediate issues with their AWS resources in the context of their applications. Application Manager aggregates operations information from multiple AWS services and Systems Manager capabilities to a single AWS Management Console.

- [Viewing overview information about an application \(p. 242\)](#)

### Parameter Store

[Parameter Store \(p. 256\)](#), a capability of AWS Systems Manager, provides a centralized store to manage your configuration data, whether plaintext data such as database strings or secrets such as passwords. This allows you to separate your secrets and configuration data from your code. You can tag and organize parameters into hierarchies to help manage them.

- [Create a Systems Manager parameter \(console\) \(p. 281\)](#)
- [Create a Systems Manager parameter \(AWS CLI\) \(p. 282\)](#)
- [Working with parameter hierarchies \(p. 326\)](#)
- [Advanced: Create a SecureString parameter and join a node to a Domain \(PowerShell\) \(p. 342\)](#)

## Change management

### Change Manager

[Change Manager \(p. 352\)](#), a capability of AWS Systems Manager, is an enterprise change management framework for requesting, approving, implementing, and reporting on operational changes to your application configuration and infrastructure.

- Advanced: [Try out the AWS managed Hello World change template \(p. 372\)](#)

## Automation

[Automation \(p. 397\)](#), a capability of AWS Systems Manager, 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 to share best practices with the rest of your organization.

- Advanced: [Walkthrough: Patch a Linux AMI \(console\) \(p. 654\)](#)
- Advanced: [Walkthrough: Patch a Linux AMI \(AWS CLI\) \(p. 657\)](#)
- Advanced: [Walkthrough: Patch a Windows Server AMI \(p. 661\)](#)
- Advanced: [Walkthrough: Simplify AMI patching using Automation, AWS Lambda, and Parameter Store \(p. 665\)](#)
- Advanced: [Walkthrough: Patch an AMI and update an Auto Scaling group \(p. 671\)](#)
- Advanced: [Walkthrough: Run the EC2Rescue tool on unreachable instances \(p. 677\)](#)
- Advanced: [Walkthrough: Reset passwords and SSH keys on EC2 instances \(p. 681\)](#)
- Advanced: [Walkthrough: Using Automation with Jenkins \(p. 687\)](#)

## Change Calendar

[Change Calendar \(p. 695\)](#), a capability of AWS Systems Manager, helps you set up date and time ranges when actions you specify are or are not permitted to occur in your AWS account.

- [Creating a change calendar \(p. 697\)](#)
- [Creating a Change Calendar event \(p. 699\)](#)

## Maintenance Windows

[Maintenance Windows \(p. 706\)](#), a capability of AWS Systems Manager, helps 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 \(AWS CLI\) \(p. 733\)](#)
- [Tutorial: Update a maintenance window \(AWS CLI\) \(p. 769\)](#)
- [Tutorial: View information about maintenance windows \(AWS CLI\) \(p. 755\)](#)
- [Tutorial: View information about tasks and task executions \(AWS CLI\) \(p. 766\)](#)

# Node management

## Fleet Manager

[Fleet Manager \(p. 798\)](#), a capability of AWS Systems Manager, is a unified user interface (UI) experience that helps you remotely manage your managed nodes. With Fleet Manager, you can view the health and performance status of your entire managed-node fleet from one console.

- [Working with the file system \(p. 826\)](#)

## Compliance

[Compliance \(p. 836\)](#), a capability of AWS Systems Manager, scans your fleet of managed nodes for patch compliance and configuration inconsistencies.

- [Compliance walkthrough \(AWS CLI\) \(p. 844\)](#)

## Inventory

[Inventory \(p. 848\)](#), a capability of AWS Systems Manager, collects information about your managed nodes and the software installed on them, helping you to understand your system configurations and installed applications.

- Advanced: [Walkthrough: Assign custom inventory metadata to a managed node \(p. 898\)](#)
- Advanced: [Walkthrough: Configure your managed nodes for Inventory by using the CLI \(p. 899\)](#)
- Advanced: [Walkthrough: Use resource data sync to aggregate inventory data \(p. 903\)](#)

## Session Manager

[Session Manager \(p. 912\)](#), a capability of AWS Systems Manager, helps you manage your Amazon EC2 instances and AWS IoT Greengrass core devices 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 \(p. 964\)](#)

## Run Command

[Run Command \(p. 991\)](#), a capability of AWS Systems Manager, provides secure remote management of your managed nodes at scale without logging into them. Run Command helps you manage your nodes without using bastion hosts, SSH, or remote PowerShell. Run Command also provides a simple way of automating common administrative tasks across groups of managed nodes such as registry edits, user management, and software and patch installations.

- [Walkthrough: Use the AWS CLI with Run Command \(p. 1019\)](#)
- [Walkthrough: Use the AWS Tools for Windows PowerShell with Run Command \(p. 1025\)](#)

## State Manager

[State Manager \(p. 1034\)](#), a capability of AWS Systems Manager, helps you maintain your fleet of managed nodes in a consistent configuration and state that you define. Using State Manager, you can control configuration details such as server configurations, antivirus definitions, firewall settings, and more.

- [Walkthrough: Creating associations that run MOF files \(p. 1066\)](#)
- [Walkthrough: Automatically update SSM Agent \(CLI\) \(p. 1089\)](#)
- [Walkthrough: Automatically update PV drivers on EC2 instances for Windows Server \(console\) \(p. 1092\)](#)

## Patch Manager

[Patch Manager \(p. 1093\)](#), a capability of AWS Systems Manager, helps you select and deploy operating system and software patches automatically across large groups of managed nodes.

- [Working with custom patch baselines \(console\) \(p. 1194\)](#)

- [Working with patch groups \(p. 1205\)](#)
- [Walkthrough: Patch a server environment \(AWS CLI\) \(p. 1238\)](#)

### Distributor

[Distributor \(p. 1252\)](#), a capability of AWS Systems Manager, helps you package your own software—or find AWS provided agent software packages, such as AmazonCloudWatchAgent—to install on Systems Manager managed nodes.

- [Create a package \(p. 1260\)](#)
- [Add a package to Distributor \(p. 1268\)](#)

## Shared resources

### Documents

[Documents \(p. 1287\)](#), a capability of AWS Systems Manager, helps you create and manage *SSM documents*. An SSM document defines the actions that Systems Manager performs on your managed nodes. Systems Manager includes more than a dozen preconfigured 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 \(console\) \(p. 1355\)](#)
- [Create an SSM document \(command line\) \(p. 1355\)](#)

# Working with SSM Agent

AWS Systems Manager Agent (SSM Agent) is Amazon software that runs on Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and on-premises servers and virtual machines (VMs). 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 that your managed nodes communicate with ec2messages.\* endpoints. For more information, see [Reference: ec2messages, ssmmessages, and other API operations \(p. 1546\)](#). For information about porting SSM Agent logs to Amazon CloudWatch Logs, see [Monitoring AWS Systems Manager \(p. 1428\)](#).

## Contents

- [SSM Agent technical reference \(p. 68\)](#)
- [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#)
- [SSM Agent version 3.0 \(p. 75\)](#)
- [Working with SSM Agent on EC2 instances for Linux \(p. 76\)](#)
- [Working with SSM Agent on EC2 instances for macOS \(p. 121\)](#)
- [Working with SSM Agent on EC2 instances for Windows Server \(p. 123\)](#)
- [Working with SSM Agent on edge devices \(p. 127\)](#)
- [Checking SSM Agent status and starting the agent \(p. 128\)](#)
- [Checking the SSM Agent version number \(p. 129\)](#)
- [Viewing SSM Agent logs \(p. 132\)](#)
- [Restricting access to root-level commands through SSM Agent \(p. 134\)](#)
- [Automating updates to SSM Agent \(p. 135\)](#)
- [Subscribing to SSM Agent notifications \(p. 137\)](#)
- [SSM Agent communications with AWS managed S3 buckets \(p. 138\)](#)
- [Troubleshooting SSM Agent \(p. 142\)](#)

## SSM Agent technical reference

Use the information in this topic to help you implement AWS Systems Manager Agent (SSM Agent) and understand how the agent works.

### Topics

- [SSM Agent credentials precedence \(p. 69\)](#)
- [About the local ssm-user account \(p. 70\)](#)
- [SSM Agent and the Instance Metadata Service \(IMDS\) \(p. 70\)](#)
- [Keeping SSM Agent up-to-date \(p. 70\)](#)
- [SSM Agent rolling updates by AWS Regions \(p. 71\)](#)
- [Installing SSM Agent on VMs and on-premises instances \(p. 71\)](#)
- [Validating on-premises servers, edge devices, and virtual machines using a hardware fingerprint \(p. 71\)](#)

- [SSM Agent on GitHub \(p. 72\)](#)

## SSM Agent credentials precedence

This topic describes important information about how SSM Agent is granted permission to perform actions on your resources. The content is primarily focused on SSM Agent running Amazon Elastic Compute Cloud (Amazon EC2) instances and servers or VMs in your hybrid environment. For edge devices, you must configure your devices to use AWS IoT Greengrass Core software, configure an AWS Identity and Access Management (IAM) service role, and deploy SSM Agent to your devices by using AWS IoT Greengrass. For more information, see [Setting up AWS Systems Manager for edge devices \(p. 52\)](#).

When SSM Agent is installed on an instance, it requires permissions in order to communicate with the Systems Manager service. On Amazon Elastic Compute Cloud (Amazon EC2) instances, these permissions are provided in an instance profile that is attached to the instance. On a hybrid instance, SSM Agent normally gets the needed permissions from the shared credentials file, located at `/root/.aws/credentials` (Linux and macOS) or `%USERPROFILE%\aws\credentials` (Windows Server). The needed permissions are added to this file during the hybrid activation process.

In rare cases, however, an instance might end up with permissions added to more than one of the locations where SSM Agent checks for permissions to run its tasks.

As one example, you might configure an instance to be managed by Systems Manager. For an EC2 instance, that configuration includes attaching an instance profile. For an on-premises server or virtual machine (VM), that means credentials are provided through the hybrid activation process. But, then you decide to also use that instance for developer or end-user tasks and install the AWS Command Line Interface (AWS CLI) on it. This installation results in additional permissions being added to a credentials file on the instance.

When you run a Systems Manager command on the instance, SSM Agent might try to use credentials different from the ones you expect it to use, such as from a credentials file instead of an instance profile. This is because SSM Agent looks for credentials in the order prescribed for the *default credential provider chain*.

### Note

On Linux and macOS, SSM Agent runs as the root user. Therefore, the environment variables and credentials file that SSM Agent looks for in this process are those of the root user only (`/root/.aws/credentials`). SSM Agent doesn't look at the environment variables or credentials file of any other user accounts on the instance during the search for credentials.

The default provider chain looks for credentials in the following order:

1. Environment variables, if configured (`AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`).
2. Shared credentials file (`$HOME/.aws/credentials` for Linux and macOS or `%USERPROFILE%\aws\credentials` for Windows Server) with permissions provided by, for example, a hybrid activation or an AWS CLI installation.
3. An AWS Identity and Access Management (IAM) role for tasks if an application is present that uses an Amazon Elastic Container Service (Amazon ECS) task definition or `RunTask` API operation.
4. An instance profile attached to an Amazon EC2 instance.

For related information, see the following topics:

- Instance profiles for EC2 instances – [Create an IAM instance profile for Systems Manager \(p. 21\)](#) and [Attach an IAM instance profile to an EC2 instance \(p. 26\)](#)
- Hybrid activations – [Create a managed-instance activation for a hybrid environment \(p. 41\)](#)
- AWS CLI credentials – [Configuration and credential file settings](#) in the *AWS Command Line Interface User Guide*

- Default credential provider chain – [Specifying Credentials](#) in the *AWS SDK for Go Developer Guide*

**Note**

This topic in the *AWS SDK for Go Developer Guide* describes the default provider chain in terms of the SDK for Go; however, the same principles apply to evaluating credentials for SSM Agent.

## 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 the `/etc/sudoers.d` directory (Linux and macOS) or to the Administrators group (Windows Server). 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 starts in Session Manager, a capability of AWS Systems Manager. You can change the permissions by moving `ssm-user` to a less-privileged group or by changing the `sudoers` file. The `ssm-user` account isn't 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 isn't created automatically on Windows Server machines that are being used as domain controllers. To use Session Manager on a Windows Server domain controller, create the `ssm-user` account manually if it isn't already present, and assign Domain Administrator permissions to the user.

**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 role with Session Manager permissions \(p. 920\)](#).

## 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). Your instance must be able to access IPv4 address of the instance metadata service: 169.254.169.254. For more information, see [Instance metadata and user data](#) in the *Amazon EC2 User Guide for Linux Instances*.

## 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. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

**Note**

An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

Amazon Machine Images (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.

## SSM Agent rolling updates by AWS Regions

After an SSM Agent update is made available in its GitHub repository, it can take up to two weeks until the updated version is rolled out to all 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 turn on allow downgrade to proceed" error when trying to deploy a new version of SSM Agent in a Region.

To determine the version of SSM Agent available to you, you can run a `curl` command.

To view the version of the agent available in the global download bucket, run the following command.

```
curl https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/VERSION
```

To view the version of the agent available in a specific Region, run the following command, substituting `region` with the Region you're working in, such as `us-east-2` for the US East (Ohio) Region.

```
curl https://s3.region.amazonaws.com/amazon-ssm-region/latest/VERSION
```

You can also open the `VERSION` file directly in your browser without a `curl` command.

## Installing SSM Agent on VMs and on-premises instances

For information about installing SSM Agent on on-premises servers, edge devices, and virtual machines (VMs) in a hybrid environment, see [Install SSM Agent for a hybrid environment \(Linux\) \(p. 45\)](#) and [Install SSM Agent for a hybrid environment \(Windows\) \(p. 50\)](#).

## Validating on-premises servers, edge devices, and virtual machines using a hardware fingerprint

When running on-premises servers, edge devices, and virtual machines (VMs) in a hybrid environment, SSM Agent gathers a number of system attributes (referred to as the *hardware hash*) and uses these attributes to compute a *fingerprint*. The fingerprint is an opaque string that the agent passes to certain Systems Manager APIs. This unique fingerprint associates the caller with a particular on-premises managed node. The agent stores the fingerprint and hardware hash on the local disk in a location called the *Vault*.

The agent computes the hardware hash and fingerprint when the on-premises server, edge device, or VM is registered for use with Systems Manager. Then, the fingerprint is passed back to the Systems Manager service when the agent sends a `RegisterManagedInstance` command.

Later, when sending a `RequestManagedInstanceRoleToken` command, the agent checks the fingerprint and hardware hash in the Vault to make sure that the current machine attributes match with the stored hardware hash. If the current machine attributes do match the hardware hash stored in the Vault, the agent passes the fingerprint from the Vault to `RegisterManagedInstance`, resulting in a successful call.

If the current machine attributes don't match the stored hardware hash, SSM Agent computes a new fingerprint, stores the new hardware hash and fingerprint in the Vault, and passes the new fingerprint to `RequestManagedInstanceRoleToken`. This causes `RequestManagedInstanceRoleToken` to fail, and the agent won't be able to obtain a role token to connect to the Systems Manager service.

This failure is by design and is used as a verification step to prevent multiple on-premises managed nodes from communicating with the Systems Manager service as the same managed node.

When comparing the current machine attributes to the hardware hash stored in the Vault, the agent uses the following logic to determine whether the old and new hashes match:

- If the SID (system/machine ID) is different, then no match.
- Otherwise, if the IP address is the same, then match.
- Otherwise, the percentage of machine attributes that match is computed and compared with the user-configured similarity threshold to determine whether there is a match.

The similarity threshold is stored in the Vault, as part of the hardware hash.

The similarity threshold can be set after an instance is registered using a command like the following.

On Linux instances:

```
sudo amazon-ssm-agent -fingerprint -similarityThreshold 1
```

On Windows Server instances using PowerShell:

```
cd "C:\Program Files\Amazon\SSM\"
.\\amazon-ssm-agent.exe -fingerprint -similarityThreshold 1
```

**Important**

If one of the components used to calculate the fingerprint changes, this can cause the agent to hibernate. To help avoid this hibernation, set the similarity threshold to a low value, such as 1.

## SSM Agent on GitHub

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, Amazon Web Services doesn't provide support for running modified copies of this software.

## Amazon Machine Images (AMIs) with SSM Agent preinstalled

AWS Systems Manager Agent (SSM Agent) is preinstalled on some Amazon Machine Images (AMIs) provided by AWS.

In most cases, SSM Agent is preinstalled on AMIs provided by AWS for the following operating systems (OSs):

- Amazon Linux Base AMIs dated 2017.09 and later
- Amazon Linux 2
- Amazon Linux 2 ECS-Optimized Base AMIs
- macOS 10.14.x (Mojave), 10.15.x (Catalina), and 11.x (Big Sur)
- SUSE Linux Enterprise Server (SLES) 12 and 15
- Ubuntu Server 16.04, 18.04, and 20.04
- Windows Server 2008-2012 R2 AMIs published in November 2016 or later

- Windows Server 2016, 2019, and 2022

In rare cases, the agent might not be preinstalled, or it might be installed but not running on an Amazon Elastic Compute Cloud (Amazon EC2) instance created from one of the AWS managed AMIs for these OSs. Therefore, before you use Systems Manager with instances created from any of these AMIs, we recommend connecting to one of the instances to verify that SSM Agent is installed and running.

**Note**

Instances launched from AMIs that are not included in the preceding list might not have SSM Agent preinstalled. You can use the following procedure to verify whether the agent is installed on your instance. If you find that the agent is not installed, you can manually install the agent on [Linux](#), [\(p. 76\)](#) [macOS](#) [\(p. 121\)](#), and [Windows Server](#) [\(p. 123\)](#) instances. You must also manually install SSM Agent on [virtual machines](#) [\(p. 34\)](#) in your hybrid environment or on-premises servers, and on your [edge devices](#) [\(p. 52\)](#).

SSM Agent might be preinstalled on AMIs found in AWS Marketplace or Community AMIs; however, AWS doesn't support these AMIs.

### To verify installation of SSM Agent on an instance

1. After waiting a few minutes after the instance is launched, connect to it by using your preferred method, such as SSH for Linux or Remote Desktop for Windows Server.
2. Run the command for the operating system type of your instance to check SSM Agent status.

#### Amazon Linux

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

#### Amazon Linux 2

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

#### macOS

Check for an agent log file at `/var/log/amazon/ssm/amazon-ssm-agent.log`.

#### SUSE Linux Enterprise Server

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

#### Ubuntu Server 16.04 (32-bit)

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

#### Ubuntu Server 16.04 64-bit instances (deb package installation)

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

#### Ubuntu Server 16.04, 18.04, and 20.04 LTS & and 20.10 STR 64-bit (Snap package installation)

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

#### Windows Server

*Run in PowerShell:*

```
Get-Service AmazonSSMAgent
```

**Tip**

To view commands for checking agent status on all operating system types supported by Systems Manager, see [Checking SSM Agent status and starting the agent](#) [\(p. 128\)](#).

3. Check the results of the command to determine the status of the agent.

#### Result 1: Installed and running

In most cases, the command reports that the agent is running.

See the following Amazon Linux 2 example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: enabled)
 Active: active (running) since Wed 2021-10-20 19:09:29 UTC; 4min 6s ago
 --truncated--
```

See the following Windows Server example.

| Status  | Name           | DisplayName      |
|---------|----------------|------------------|
| -----   | -----          | -----            |
| Running | AmazonSSMAgent | Amazon SSM Agent |

### Result 2: Installed but not running

In rare cases, the command reports that the agent is installed but not running.

See the following Amazon Linux 2 example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: enabled)
 Active: inactive (dead) since Wed 2021-10-20 22:16:41 UTC; 18s ago
 --truncated--
```

See the following Windows Server example.

| Status  | Name           | DisplayName      |
|---------|----------------|------------------|
| -----   | -----          | -----            |
| Stopped | AmazonSSMAgent | Amazon SSM Agent |

If the agent is installed but not running, activate it manually using the command for your operating system type.

#### Amazon Linux

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

#### Amazon Linux 2

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

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

#### macOS

```
sudo launchctl load -w /Library/LaunchDaemons/com.amazon.aws.ssm.plist
```

```
sudo launchctl start com.amazon.aws.ssm
```

#### SUSE Linux Enterprise Server

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

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

### Ubuntu Server 16.04 (32-bit)

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

### Ubuntu Server 16.04 64-bit instances (deb package installation)

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

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

### Ubuntu Server 16.04, 18.04, and 20.04 LTS & and 20.10 STR 64-bit (Snap package installation)

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

## Windows Server

*Run in PowerShell:*

```
Start-Service AmazonSSMAgent
```

### Result 3: Not installed

In rare cases, the command reports that the agent isn't installed.

See the following Amazon Linux 2 example.

```
Unit amazon-ssm-agent.service could not be found.
```

See the following Windows Server example.

```
Get-Service : Cannot find any service with service name 'AmazonSSMAgent'.
--truncated--
```

If the agent isn't installed, install it manually by using the procedure for your operating system type:

- [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#)
- [Manually installing SSM Agent on EC2 instances for macOS \(p. 121\)](#)
- [Manually installing SSM Agent on EC2 instances for Windows Server \(p. 123\)](#)

## SSM Agent version 3.0

On September 21, 2020, AWS Systems Manager released AWS Systems Manager Agent (SSM Agent) version 3.0. Note the following important details about this release:

- Version 3.0 is backward compatible with version 2.x.
- If you configured your managed nodes to automatically update SSM Agent by using target version `$LATEST` (the default configuration for auto-update), then Systems Manager automatically updates the agent on your nodes to version 3.0 and removes version 2.x.

If you manually download SSM Agent, the system installs version 2.x.

- When you update to version 3.0, the system renames `amazon-ssm-agent` on your managed instance to `ssm-agent-worker`. The update then installs a new binary named `amazon-ssm-agent`. The new binary functions as a process manager for `ssm-agent-worker`. The `ssm-agent-worker` binary communicates directly with Systems Manager to process requests.

**Important**

Customers running SSM Agent in secure environments must add `ssm-agent-worker` to the allow list in your computing environment before upgrading.

- Version 2.x is still supported, but it will be deprecated.
- Amazon Machine Images (AMIs) install version 2.x until 2.x is deprecated.
- As of January 14, 2020, Windows Server 2008 is no longer supported for feature or security updates from Microsoft. Legacy Amazon Machine Images (AMIs) for Windows Server 2008 and 2008 R2 still include version 2 of SSM Agent preinstalled, but Systems Manager no longer officially supports 2008 versions and no longer updates the agent for these versions of Windows Server. In addition, [SSM Agent version 3.0 \(p. 75\)](#) might not be compatible with all operations on Windows Server 2008 and 2008 R2. The final officially supported version of SSM Agent for Windows Server 2008 versions is 2.3.1644.0.
- With version 3.0, SSM Agent start and update events are logged on the instance. For information about viewing SSM Agent log files, see [Viewing SSM Agent logs \(p. 132\)](#).
- You can use Amazon CloudWatch to perform actions on SSM Agent when the system logs a start or update event. For more information, see [Create a CloudWatch alarm based on a static threshold](#) in the [Amazon CloudWatch User Guide](#).
- With version 3.0, there is no change to the following:
  - Minimum AWS Identity and Access Management (IAM) permissions, the credential chain, or the `ssm-user` creation process
  - Supported platforms, log location, or debug logging
  - Command processing or SSM plugin support
  - The proxy configuration process
  - Windows Registry keys

## Working with SSM Agent on EC2 instances for Linux

AWS Systems Manager Agent (SSM Agent) processes Systems Manager requests and configures your machine as specified in the request. Use the procedures in following topics to install, configure, or uninstall SSM Agent on Linux operating systems.

**Topics**

- [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#)
- [Verifying the signature of the SSM Agent \(p. 113\)](#)
- [Configuring SSM Agent to use a proxy \(Linux\) \(p. 117\)](#)
- [Uninstalling SSM Agent from Linux instances \(p. 120\)](#)

## Manually installing SSM Agent on EC2 instances for Linux

Before you manually install AWS Systems Manager Agent (SSM Agent) on an Amazon Elastic Compute Cloud (Amazon EC2) Linux operating system, review the following information.

**SSM Agent installation file URLs**

You can access the installation files for SSM Agent that are stored in any commercial AWS Region. We also provide installation files in a globally available Amazon Simple Storage Service (Amazon S3) bucket that you can use as an alternative or backup source of files.

If you're manually installing the agent on a instance or two, you can use the commands in the **Quick installation** procedures we provide to save time.

If you're creating a script or template to use for installing the agent on multiple instances, we recommend that you use the installation files in or near an AWS Region where you're geographically located. For bulk installations, this can increase the speed of your downloads and reduce latency. In these cases, we recommend using the **Create custom installation commands** procedures in the installation topics.

### Amazon Machine Images with the agent preinstalled

SSM Agent is preinstalled on some Amazon Machine Images (AMIs) provided by AWS. For information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

### Installation on other machine types

If you need to install the agent on an on-premises server or a virtual machine (VM) so that it can be used with Systems Manager, see [Install SSM Agent for a hybrid environment \(Linux\) \(p. 45\)](#). For information about installing the agent on edge devices, see [Setting up AWS Systems Manager for edge devices \(p. 52\)](#).

### Keeping the 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. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

### Choose your operating system

To view the procedure for manually installing SSM Agent on the specified operating system, choose a link from the following list:

#### Note

For a list of supported versions of each of the following operating systems, see [Supported operating systems \(p. 9\)](#).

- [Amazon Linux \(p. 78\)](#)
- [Amazon Linux 2 \(p. 81\)](#)
- [CentOS \(p. 83\)](#)
- [CentOS Stream \(p. 89\)](#)
- [Debian Server \(p. 91\)](#)
- [Oracle Linux \(p. 93\)](#)
- [Red Hat Enterprise Linux \(p. 95\)](#)
- [Rocky Linux \(p. 101\)](#)
- [SUSE Linux Enterprise Server \(p. 103\)](#)
- [Ubuntu Server \(p. 106\)](#)

## Manually installing SSM Agent on Amazon Linux instances

### Important

This topic provides commands for working with SSM Agent on **Amazon Linux** instances. Some of these commands aren't supported on **Amazon Linux 2** instances. Before continuing, verify that you're viewing the correct topic for your instance type. For commands to run on Amazon Linux 2 instances, see [Manually installing SSM Agent on Amazon Linux 2 instances \(p. 81\)](#).

In most cases, the Amazon Machine Images (AMIs) for Amazon Linux that are provided by AWS come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For more information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

In the event that SSM Agent isn't preinstalled on a new Amazon Linux instance, or if you need to manually reinstall the agent, use the information on this page to help you.

### Before you begin

Before you install SSM Agent on an Amazon Linux instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).
- Managed nodes created from an Amazon Linux AMI that use a proxy must run a current version of the Python requests module to support Patch Manager operations. For more information, see [Upgrading the Python requests module on Amazon Linux instances that use a proxy server \(p. 119\)](#).
- If you use a yum command to update SSM Agent on a managed node 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.

### Topics

- [Quick installation commands for SSM Agent on Amazon Linux \(p. 78\)](#)
- [Create custom agent installation commands for Amazon Linux in your Region \(p. 80\)](#)

## Quick installation commands for SSM Agent on Amazon Linux

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

### To install SSM Agent on Amazon Linux using quick copy and paste commands

- Connect to your Amazon Linux instance using your preferred method, such as SSH.
- Copy the command for your instance's architecture and run it on the instance.

#### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for Amazon Linux.

#### x86\_64

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

#### x86

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

#### ARM64

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

3. (Recommended) Run the command for your instance's architecture to verify that the agent is running.

#### x86\_64 and x86

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

#### ARM64

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

In most cases, the command reports that the agent is running, as shown in the following examples.

#### x86\_64 and x86

```
amazon-ssm-agent start/running, process 12345
```

#### ARM64

```
amazon-ssm-agent.service - amazon-ssm-agent
Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: enabled)
Active: active (running) since Wed 2021-10-20 19:09:29 UTC; 4min 6s ago
 --truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following examples.

#### x86\_64 and x86

```
amazon-ssm-agent stop/waiting
```

#### ARM64

```
amazon-ssm-agent.service - amazon-ssm-agent
Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: enabled)
Active: inactive (dead) since Wed 2021-10-20 22:16:41 UTC; 18s ago
 --truncated--
```

To activate the agent in these cases, run the command for your instance's architecture.

#### x86\_64 and x86

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

#### ARM64

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

### Create custom agent installation commands for Amazon Linux in your Region

When you install SSM Agent on multiple instances using a script or template, we recommend using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

#### Tip

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on Amazon Linux \(p. 78\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).

#### x86\_64

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

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/linux_amd64/amazon-ssm-agent.rpm
```

#### x86

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

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/linux_386/amazon-ssm-agent.rpm
```

#### ARM64

```
sudo yum install -y https://s3.region.amazonaws.com/amazon-ssm-region/latest/linux_arm64/amazon-ssm-agent.rpm
```

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/linux_arm64/amazon-ssm-agent.rpm
```

## Manually installing SSM Agent on Amazon Linux 2 instances

### Important

This topic provides commands for working with SSM Agent on **Amazon Linux 2** instances. Some of these commands aren't supported on Amazon Linux instances. Before continuing, ensure you're viewing the correct topic for your instance type. For commands to run on Amazon Linux instances, see [Manually installing SSM Agent on Amazon Linux instances \(p. 78\)](#).

In most cases, the Amazon Machine Images (AMIs) for Amazon Linux 2 that are provided by AWS come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For more information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

In the event that SSM Agent isn't preinstalled on a new Amazon Linux 2 instance, or if you need to manually reinstall the agent, use the information on this page to help you.

### Before you begin

Before you install SSM Agent on an Amazon Linux 2 instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).
- If you use a `yum` command to update SSM Agent on a managed node 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.

### Topics

- [Quick installation commands for SSM Agent on Amazon Linux 2 \(p. 81\)](#)
- [Create custom agent installation commands for Amazon Linux 2 in your Region \(p. 82\)](#)

## Quick installation commands for SSM Agent on Amazon Linux 2

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

### To install SSM Agent on Amazon Linux 2 using quick copy and paste commands

1. Connect to your Amazon Linux 2 instance using your preferred method, such as SSH.
2. Copy the command for your instance's architecture and run it on the instance.

#### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for Amazon Linux 2.

#### x86\_64

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

#### ARM64

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

3. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: enabled)
Active: active (running) since Wed 2021-10-20 19:09:29 UTC; 4min 6s ago
 --truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: enabled)
Active: inactive (dead) since Wed 2021-10-20 22:16:41 UTC; 18s ago
 --truncated--
```

To activate the agent in these cases, run the following command.

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

## Create custom agent installation commands for Amazon Linux 2 in your Region

When you install SSM Agent on multiple instances using a script or template, we recommend using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

### Tip

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on Amazon Linux \(p. 78\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).

### x86\_64

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

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_amd64/amazon-ssm-agent.rpm
```

### ARM64

```
sudo yum install -y https://s3.region.amazonaws.com/amazon-ssm-region/latest/
linux_arm64/amazon-ssm-agent.rpm
```

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/linux_arm64/amazon-ssm-agent.rpm
```

## Manually installing SSM Agent on CentOS instances

The Amazon Machine Images (AMIs) for CentOS that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For a list of AWS managed AMIs on which the agent might be preinstalled, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

Use the information in this section to help you manually install or reinstall SSM Agent on a CentOS instance.

### Before you begin

Before you install SSM Agent on a CentOS instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).
- If you use a `yum` command to update SSM Agent on a managed node 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.

### Topics

- [Install SSM Agent on CentOS 8.x \(p. 83\)](#)
- [Install SSM Agent on CentOS 7.x \(p. 85\)](#)
- [Install SSM Agent on CentOS 6.x \(p. 87\)](#)

## Install SSM Agent on CentOS 8.x

The Amazon Machine Images (AMIs) for CentOS 8 that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. Use the information on this page to help you install or reinstall the agent on CentOS 8 instances.

### Before you begin

Before you install SSM Agent on a CentOS 8 instance, note the following:

- Ensure that either Python 2 or Python 3 is installed on your CentOS 8 instance. This is required in order for SSM Agent to work properly.

### Topics

- [Quick installation commands for SSM Agent on CentOS 8 \(p. 83\)](#)
- [Create custom agent installation commands for CentOS 8 in your Region \(p. 84\)](#)

## Quick installation commands for SSM Agent on CentOS 8

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

## To install SSM Agent on CentOS 8.x

1. Connect to your CentOS 8 instance using your preferred method, such as SSH.
2. Copy the command for your instance's architecture and run it on the instance.

### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for CentOS 8.

x86\_64 instances

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

ARM64 instances

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

3. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor>
 Active: active (running) since Tue 2022-04-19 15:48:54 UTC; 19s ago
 --truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; disabled; vendor>
 Active: inactive (dead)
 --truncated--
```

To activate the agent in these cases, run the following commands.

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

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

## Create custom agent installation commands for CentOS 8 in your Region

When you install SSM Agent on multiple instances using a script or template, we recommend using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

### Tip

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on CentOS 8 \(p. 83\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the *Amazon Web Services General Reference*.

#### x86\_64

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

See the following example.

```
sudo dnf install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_amd64/amazon-ssm-agent.rpm
```

#### ARM64

```
sudo dnf install -y https://s3.region.amazonaws.com/amazon-ssm-region/latest/
linux_arm64/amazon-ssm-agent.rpm
```

See the following example.

```
sudo dnf install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_arm64/amazon-ssm-agent.rpm
```

## Install SSM Agent on CentOS 7.x

The Amazon Machine Images (AMIs) for CentOS 7 that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. Use the information on this page to help you install or reinstall the agent on CentOS 7 instances.

### Topics

- [Quick installation commands for SSM Agent on CentOS 7 \(p. 85\)](#)
- [Create custom agent installation commands for CentOS 7 in your Region \(p. 86\)](#)

### Quick installation commands for SSM Agent on CentOS 7

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

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

1. Connect to your CentOS 7 instance using your preferred method, such as SSH.
2. Copy the command for your instance's architecture and run it on the instance.

#### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for CentOS 7.

#### x86\_64 instances

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

### ARM64 instances

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

3. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: disabled)
 Active: active (running) since Tue 2022-04-19 15:57:27 UTC; 6s ago
 --truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: disabled)
 Active: inactive (dead) since Tue 2022-04-19 15:58:44 UTC; 2s ago
 --truncated--
```

To activate the agent in these cases, run the following commands.

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

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

### Create custom agent installation commands for CentOS 7 in your Region

When you install SSM Agent on multiple instances using a script or template, we recommend using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

**Tip**

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on CentOS 7 \(p. 85\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).

#### x86\_64

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

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_amd64/amazon-ssm-agent.rpm
```

#### ARM64

```
sudo yum install -y https://s3.region.amazonaws.com/amazon-ssm-region/latest/
linux_arm64/amazon-ssm-agent.rpm
```

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_arm64/amazon-ssm-agent.rpm
```

## Install SSM Agent on CentOS 6.x

The Amazon Machine Images (AMIs) for CentOS 6 that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. Use the information on this page to help you install or reinstall the agent on CentOS 6 instances.

### Topics

- [Quick installation commands for SSM Agent on CentOS 6 \(p. 87\)](#)
- [Create custom agent installation commands for CentOS 6 in your Region \(p. 88\)](#)

### Quick installation commands for SSM Agent on CentOS 6

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

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

1. Connect to your CentOS 6 instance using your preferred method, such as SSH.
2. Copy the command for your instance's architecture and run it on the instance.

##### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for CentOS 6.

The following commands specify the version directory `3.0.1390.0` instead of a `latest` directory. This is because SSM Agent version 3.1 and later are not supported for CentOS 6.

x86\_64 instances

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/
SSMAgent/3.0.1390.0/linux_amd64/amazon-ssm-agent.rpm
```

ARM64 instances

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/
SSMAgent/3.0.1390.0/linux_386/amazon-ssm-agent.rpm
```

3. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent start/running, process 1744
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent stop/waiting
```

To activate the agent in these cases, run the following command.

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

## Create custom agent installation commands for CentOS 6 in your Region

When you install SSM Agent on multiple instances using a script or template, we recommended using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

### Tip

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on CentOS 6 \(p. 87\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the *Amazon Web Services General Reference*.

### Note

The following commands specify the version directory `3.0.1390.0` instead of a `latest` directory. This is because SSM Agent version 3.1 and later are not supported for CentOS 6.

x86\_64

```
sudo yum install -y https://s3.region.amazonaws.com/amazon-ssm-region/3.0.1390.0/
linux_amd64/amazon-ssm-agent.rpm
```

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/3.0.1390.0/
linux_amd64/amazon-ssm-agent.rpm
```

x86

```
sudo yum install -y https://s3.region.amazonaws.com/amazon-ssm-region/3.0.1390.0/
linux_386/amazon-ssm-agent.rpm
```

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/3.0.1390.0/
linux_386/amazon-ssm-agent.rpm
```

## Manually install SSM Agent on CentOS Stream instances

The Amazon Machine Images (AMIs) for CentOS Stream that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For a list of AWS managed AMIs on which the agent might be preinstalled, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

Use the information in this section to help you manually install or reinstall SSM Agent on a CentOS Stream instance.

### Before you begin

Before you install SSM Agent on a CentOS Stream instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).

### Topics

- [Quick installation commands for SSM Agent on CentOS Stream \(p. 89\)](#)
- [Create custom agent installation commands for CentOS Stream in your Region \(p. 90\)](#)

## Quick installation commands for SSM Agent on CentOS Stream

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

### Before you begin

Before you install SSM Agent on a CentOS Stream instance, note the following:

- Ensure that either Python 2 or Python 3 is installed on your CentOS Stream 8 instance. This is required in order for SSM Agent to work properly.

### To install SSM Agent on CentOS Stream

1. Connect to your CentOS Stream instance using your preferred method, such as SSH.
2. Copy the command for your instance's architecture and run it on the instance.

#### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for CentOS Stream.

x86\_64 instances

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

ARM64 instances

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

3. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor>
 Active: active (running) since Tue 2022-04-19 16:40:41 UTC; 9s ago
 Main PID: 4898 (amazon-ssm-agen)
 Tasks: 14 (limit: 4821)
 Memory: 34.6M
 CGroup: /system.slice/amazon-ssm-agent.service
 ##4898 /usr/bin/amazon-ssm-agent
 ##4954 /usr/bin/ssm-agent-worker
 --truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor>
 Active: inactive (dead) since Tue 2022-04-19 16:42:05 UTC; 2s ago
 --truncated--
```

To activate the agent in these cases, run the following commands.

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

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

## Create custom agent installation commands for CentOS Stream in your Region

When you install SSM Agent on multiple instances using a script or template, we recommend using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

**Tip**

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on CentOS Stream \(p. 89\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).

**x86\_64**

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

See the following example.

```
sudo dnf install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_amd64/amazon-ssm-agent.rpm
```

## ARM64

```
sudo dnf install -y https://s3.region.amazonaws.com/amazon-ssm-region/latest/
linux_arm64/amazon-ssm-agent.rpm
```

See the following example.

```
sudo dnf install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_arm64/amazon-ssm-agent.rpm
```

## Manually installing SSM Agent on Debian Server instances

The Amazon Machine Images (AMIs) for Debian Server that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For a list of AWS managed AMIs on which the agent might be preinstalled, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

Use the information in this section to help you manually install or reinstall SSM Agent on a Debian Server instance.

### Before you begin

Before you install SSM Agent on a Debian Server instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).

### Topics

- [Quick installation commands for SSM Agent on Debian Server \(p. 91\)](#)
- [Create custom agent installation commands for Debian Server in your Region \(p. 92\)](#)

## Quick installation commands for SSM Agent on Debian Server

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

### To install SSM Agent on Debian Server

- Connect to your Debian Server instance using your preferred method, such as SSH.
- Run the following command to create a temporary directory on the instance.

```
mkdir /tmp/ssm
```

- Run the following command to change to the temporary directory.

```
cd /tmp/ssm
```

- Copy the command for your instance's architecture and run it on the instance.

#### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for Debian Server.

For Debian Server 8, only the `x86_64` architecture is supported.

x86\_64 instances

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

ARM64 instances

```
wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_arm64/amazon-ssm-agent.deb
```

5. Run the following command.

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

6. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
 Active: active (running) since Tue 2022-04-19 16:25:03 UTC; 4s ago
 Main PID: 628 (amazon-ssm-agent)
 CGroup: /system.slice/amazon-ssm-agent.service
 ##628 /usr/bin/amazon-ssm-agent
 ##650 /usr/bin/ssm-agent-worker
 --truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
 Active: inactive (dead) since Tue 2022-04-19 16:26:30 UTC; 5s ago
 Main PID: 628 (code=exited, status=0/SUCCESS)
 --truncated--
```

To activate the agent in these cases, run the following commands.

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

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

## Create custom agent installation commands for Debian Server in your Region

When you install SSM Agent on multiple instances using a script or template, we recommend using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

**Tip**

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on Debian Server \(p. 91\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the *Amazon Web Services General Reference*.

**Note**

For Debian Server 8, only the x86\_64 architecture is supported.

x86\_64

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

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

See the following example.

```
wget https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/debian_amd64/amazon-ssm-agent.deb
```

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

ARM64

```
sudo yum install -y https://s3.region.amazonaws.com/amazon-ssm-region/latest/debian_arm64/amazon-ssm-agent.deb
```

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

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/debian_arm64/amazon-ssm-agent.deb
```

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

## Manually installing SSM Agent on Oracle Linux instances

The Amazon Machine Images (AMIs) for Oracle Linux that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For a list of AWS managed AMIs on which the agent might be preinstalled, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

Use the information in this section to help you manually install or reinstall SSM Agent on an Oracle Linux instance.

### Before you begin

Before you install SSM Agent on an Oracle Linux instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).
- If you use a `yum` command to update SSM Agent on a managed node 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.

### Topics

- [Quick installation commands for SSM Agent on Oracle Linux \(p. 94\)](#)
- [Create custom agent installation commands for Oracle Linux in your Region \(p. 95\)](#)

## Quick installation commands for SSM Agent on Oracle Linux

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

### To install SSM Agent on Oracle Linux using quick copy and paste commands

1. Connect to your Oracle Linux instance using your preferred method, such as SSH.
2. Copy the following command and run it on the instance.

#### Note

Even though URL in the following command includes an `ec2-downloads-windows` directory, these are the correct global installation files for Oracle Linux.

#### x86\_64

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

3. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: enabled)
Active: active (running) since Wed 2021-10-20 19:09:29 UTC; 4min 6s ago
 --truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: enabled)
Active: inactive (dead) since Wed 2021-10-20 22:16:41 UTC; 18s ago
 --truncated--
```

To activate the agent in these cases, run the following commands.

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

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

## Create custom agent installation commands for Oracle Linux in your Region

When you install SSM Agent on multiple instances using a script or template, we recommended using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

**Tip**

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on Oracle Linux \(p. 94\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).

### x86\_64

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

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_amd64/amazon-ssm-agent.rpm
```

## Manually installing SSM Agent on Red Hat Enterprise Linux instances

The Amazon Machine Images (AMIs) for Red Hat Enterprise Linux (RHEL) that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For a list of AWS managed AMIs on which the agent might be preinstalled, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

Use the information in this section to help you manually install or reinstall SSM Agent on a RHEL instance.

### Before you begin

Before you install SSM Agent on a RHEL instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).
- If you use a `yum` command to update SSM Agent on a managed node 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.

### Topics

- [Install SSM Agent on RHEL 8.x \(p. 96\)](#)

- [Install SSM Agent on RHEL 7.x \(p. 98\)](#)
- [Install SSM Agent on RHEL 6.x \(p. 99\)](#)

## Install SSM Agent on RHEL 8.x

The Amazon Machine Images (AMIs) for RHEL 8 that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. Use the information on this page to help you install or reinstall the agent on RHEL 8 instances.

### Before you begin

Before you install SSM Agent on a RHEL 8 instance, note the following:

- 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.

### Topics

- [Quick installation commands for SSM Agent on RHEL 8 \(p. 96\)](#)
- [Create custom agent installation commands for RHEL 8 in your Region \(p. 97\)](#)

### Quick installation commands for SSM Agent on RHEL 8

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

#### To install SSM Agent on RHEL 8.x

1. Connect to your RHEL 8 instance using your preferred method, such as SSH.
2. Copy the command for your instance's architecture and run it on the instance.

##### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for RHEL 8.

x86\_64 instances

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

ARM64 instances

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

3. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor>
 Active: active (running) since Tue 2022-04-19 16:40:41 UTC; 9s ago
 Main PID: 4898 (amazon-ssm-agen)
 Tasks: 14 (limit: 4821)
```

```
Memory: 34.6M
CGroup: /system.slice/amazon-ssm-agent.service
 ##4898 /usr/bin/amazon-ssm-agent
 ##4954 /usr/bin/ssm-agent-worker
--truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor>
 Active: inactive (dead) since Tue 2022-04-19 16:42:05 UTC; 2s ago
 --truncated--
```

To activate the agent in these cases, run the following commands.

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

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

## Create custom agent installation commands for RHEL 8 in your Region

When you install SSM Agent on multiple instances using a script or template, we recommended using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (us-east-2).

### Tip

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on RHEL 8 \(p. 96\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).

### x86\_64

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

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_amd64/amazon-ssm-agent.rpm
```

### ARM64

```
sudo yum install -y https://s3.region.amazonaws.com/amazon-ssm-region/latest/
linux_arm64/amazon-ssm-agent.rpm
```

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_arm64/amazon-ssm-agent.rpm
```

## Install SSM Agent on RHEL 7.x

The Amazon Machine Images (AMIs) for RHEL 7 that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. Use the information on this page to help you install or reinstall the agent on RHEL 7 instances.

### Topics

- [Quick installation commands for SSM Agent on RHEL 7 \(p. 98\)](#)
- [Create custom agent installation commands for RHEL 7 in your Region \(p. 99\)](#)

### Quick installation commands for SSM Agent on RHEL 7

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

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

1. Connect to your RHEL 7 instance using your preferred method, such as SSH.
2. Copy the command for your instance's architecture and run it on the instance.

##### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for RHEL 7.

##### x86\_64 instances

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

##### ARM64 instances

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

3. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: disabled)
 Active: active (running) since Tue 2022-04-19 16:47:36 UTC; 22s ago
 Main PID: 1342 (amazon-ssm-ag)
 CGroup: /system.slice/amazon-ssm-agent.service
 ##1342 /usr/bin/amazon-ssm-agent
 ##1362 /usr/bin/ssm-agent-worker
 --truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor
 preset: disabled)
```

```
Active: inactive (dead) since Tue 2022-04-19 16:48:56 UTC; 5s ago
Process: 1342 ExecStart=/usr/bin/amazon-ssm-agent (code=exited, status=0/SUCCESS)
Main PID: 1342 (code=exited, status=0/SUCCESS)
--truncated--
```

To activate the agent in these cases, run the following commands.

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

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

## Create custom agent installation commands for RHEL 7 in your Region

When you install SSM Agent on multiple instances using a script or template, we recommended using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

### Tip

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on RHEL 7 \(p. 98\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the *Amazon Web Services General Reference*.

### x86\_64

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

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_amd64/amazon-ssm-agent.rpm
```

### ARM64

```
sudo yum install -y https://s3.region.amazonaws.com/amazon-ssm-region/latest/
linux_arm64/amazon-ssm-agent.rpm
```

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_arm64/amazon-ssm-agent.rpm
```

## Install SSM Agent on RHEL 6.x

The Amazon Machine Images (AMIs) for RHEL 6 that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. Use the information on this page to help you install or reinstall the agent on RHEL 6 instances.

### Topics

- [Quick installation commands for SSM Agent on RHEL 6 \(p. 100\)](#)
- [Create custom agent installation commands for RHEL 6 in your Region \(p. 100\)](#)

## Quick installation commands for SSM Agent on RHEL 6

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

### To install SSM Agent on RHEL 6.x

1. Connect to your RHEL 6 instance using your preferred method, such as SSH.
2. Copy the command for your instance's architecture and run it on the instance.

#### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for RHEL 6.

The following commands specify the version directory `3.0.1390.0` instead of a `latest` directory. This is because SSM Agent version 3.1 and later are not supported for RHEL 6.

#### x86\_64 instances

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/
SSMAgent/3.0.1390.0/linux_amd64/amazon-ssm-agent.rpm
```

#### ARM64 instances

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/
SSMAgent/3.0.1390.0/linux_386/amazon-ssm-agent.rpm
```

3. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent start/running, process 1788
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent stop/waiting
```

To activate the agent in these cases, run the following command.

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

## Create custom agent installation commands for RHEL 6 in your Region

When you install SSM Agent on multiple instances using a script or template, we recommended using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

**Tip**

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on RHEL 6 \(p. 100\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the *Amazon Web Services General Reference*.

**Note**

The following commands specify the version directory `3.0.1390.0` instead of a `latest` directory. This is because SSM Agent version 3.1 and later are not supported for RHEL 6.

x86\_64

```
sudo yum install -y https://s3.region.amazonaws.com/amazon-ssm-region/3.0.1390.0/linux_amd64/amazon-ssm-agent.rpm
```

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/3.0.1390.0/linux_amd64/amazon-ssm-agent.rpm
```

x86

```
sudo yum install -y https://s3.region.amazonaws.com/amazon-ssm-region/3.0.1390.0/linux_386/amazon-ssm-agent.rpm
```

See the following example.

```
sudo yum install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/3.0.1390.0/linux_386/amazon-ssm-agent.rpm
```

## Manually install SSM Agent on Rocky Linux instances

The Amazon Machine Images (AMIs) for Rocky Linux that are provided by AWS do not come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For a list of AWS managed AMIs on which the agent might be preinstalled, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

Use the information in this section to help you manually install or reinstall SSM Agent on an Rocky Linux instance.

### Before you begin

Before you install SSM Agent on a Rocky Linux instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).

### Topics

- [Quick installation commands for SSM Agent on Rocky Linux \(p. 102\)](#)
- [Create custom agent installation commands for Rocky Linux in your Region \(p. 103\)](#)

## Quick installation commands for SSM Agent on Rocky Linux

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

### Before you begin

Before you install SSM Agent on a Rocky Linux instance, note the following:

- Ensure that either Python 2 or Python 3 is installed on your Rocky Linux instance. This is required in order for SSM Agent to work properly.

### To install SSM Agent on Rocky Linux

1. Connect to your Rocky Linux instance using your preferred method, such as SSH.
2. Copy the command for your instance's architecture and run it on the instance.

#### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for Rocky Linux.

x86\_64 instances

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

ARM64 instances

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

3. (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor>
 Active: active (running) since Tue 2022-04-19 16:40:41 UTC; 9s ago
 Main PID: 4898 (amazon-ssm-agen)
 Tasks: 14 (limit: 4821)
 Memory: 34.6M
 CGroup: /system.slice/amazon-ssm-agent.service
 ##4898 /usr/bin/amazon-ssm-agent
 ##4954 /usr/bin/ssm-agent-worker
 --truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
 Loaded: loaded (/etc/systemd/system/amazon-ssm-agent.service; enabled; vendor>
 Active: inactive (dead) since Tue 2022-04-19 16:42:05 UTC; 2s ago
 --truncated--
```

To activate the agent in these cases, run the following commands.

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

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

## Create custom agent installation commands for Rocky Linux in your Region

When you install SSM Agent on multiple instances using a script or template, we recommended using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

**Tip**

You can also replace a global URL in the procedure [???](#) (p. 102) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the *Amazon Web Services General Reference*.

### x86\_64

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

See the following example.

```
sudo dnf install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_amd64/amazon-ssm-agent.rpm
```

### ARM64

```
sudo dnf install -y https://s3.region.amazonaws.com/amazon-ssm-region/latest/
linux_arm64/amazon-ssm-agent.rpm
```

See the following example.

```
sudo dnf install -y https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/
linux_arm64/amazon-ssm-agent.rpm
```

## Manually install SSM Agent on SUSE Linux Enterprise Server instances

In most cases, the Amazon Machine Images (AMIs) for SUSE Linux Enterprise Server (SLES) that are provided by AWS come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For more information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled](#) (p. 72).

In the event that SSM Agent isn't preinstalled on a new SLES instance, or if you need to manually reinstall the agent, use the information on this page to help you.

### Before you begin

Before you install SSM Agent on a SLES instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).

### Topics

- [Quick installation commands for SSM Agent on SLES \(p. 104\)](#)
- [Create custom agent installation commands for SLES in your Region \(p. 105\)](#)

## Quick installation commands for SSM Agent on SLES

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

### To install SSM Agent on SLES using quick copy and paste commands

- Connect to your SLES instance using your preferred method, such as SSH.
- Option 1:** Use a `zypper` command:

- Run the following command:

```
sudo zypper install amazon-ssm-agent
```

- Enter `y` in response to any prompts.

#### Option 2: Use an `rpm` command.

- Create a temporary directory on the instance.

```
mkdir /tmp/ssm
```

- Change to the temporary directory.

```
cd /tmp/ssm
```

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

#### Note

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for SLES.

#### x86\_64 instances:

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

#### ARM64 instances:

```
wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_arm64/amazon-ssm-agent.rpm
```

- Run the following command.

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

- (Recommended) Run the following command to verify that the agent is running.

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

In most cases, the command reports that the agent is running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor
preset: disabled)
Active: active (running) since Mon 2022-02-21 23:13:28 UTC; 7s ago
Main PID: 2102 (amazon-ssm-agen)
Tasks: 15 (limit: 512)
CGroup: /system.slice/amazon-ssm-agent.service
##2102 /usr/sbin/amazon-ssm-agent
##2107 /usr/sbin/ssm-agent-worker
--truncated--
```

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

```
amazon-ssm-agent.service - amazon-ssm-agent
Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; disabled; vendor
preset: disabled)
Active: inactive (dead)
--truncated--
```

To activate the agent in these cases, run the following commands.

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

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

## Create custom agent installation commands for SLES in your Region

When you install SSM Agent on multiple instances using a script or template, we recommended using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

**Tip**

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on Amazon Linux \(p. 78\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).

**x86\_64**

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

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

See the following example.

```
wget https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/linux_amd64/amazon-ssm-agent.rpm
```

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

## ARM64

```
wget https://s3.region.amazonaws.com/amazon-ssm-region/latest/linux_arm64/amazon-ssm-agent.rpm
```

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

See the following example.

```
wget https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/linux_arm64/amazon-ssm-agent.rpm
```

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

## Manually installing SSM Agent on Ubuntu Server instances

### Important

Before you install SSM Agent on a 64-bit version of Ubuntu Server, ensure sure that you are using the correct installation tools. Beginning with Amazon Machine Images (AMIs) that are identified with 20180627, SSM Agent is pre-installed on version 16.04 using Snap packages. On instances created from earlier AMIs, SSM Agent must be installed using deb installer packages. For more information, see [Determining the correct SSM Agent version to install on 64-bit Ubuntu Server 16.04 instances \(p. 112\)](#)

In most cases, the Amazon Machine Images (AMIs) for Ubuntu Server that are provided by AWS come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For more information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

In the event that SSM Agent isn't preinstalled on a new Ubuntu Server instance, or if you need to manually reinstall the agent, use the information in this section to help you.

### Before you begin

Before you install SSM Agent on an Ubuntu Server instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#).

### Topics

- [Install SSM Agent on Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS 64-bit \(Snap\) \(p. 107\)](#)
- [Install SSM Agent on Ubuntu Server 16.04 and 14.04 64-bit \(deb\) \(p. 108\)](#)
- [Install SSM Agent on Ubuntu Server 16.04 and 14.04 32-bit \(p. 110\)](#)
- [Determining the correct SSM Agent version to install on 64-bit Ubuntu Server 16.04 instances \(p. 112\)](#)

## Install SSM Agent on Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS 64-bit (Snap)

### Before you begin

Before you install SSM Agent on an Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS 64-bit (Snap), note the following:

Version 16.04 installation by Snaps or deb installers

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.

SSM Agent installer files locations

On Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS (with Snap), 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` directory to the `/etc/amazon/ssm/` directory. Log and library files haven't changed (`/var/lib/amazon/ssm`, `/var/log/amazon/ssm`).

Using the Snap candidate channel

The *candidate* channel in the Snap store contains the latest version of SSM Agent (including all of the latest bug fixes); not the stable channel. To learn more about the differences between the candidate and stable channels, see **Risk-levels** at <https://snapcraft.io/docs/channels>.

If you want to track SSM Agent version information on the candidate channel, run the following command on your Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS 64-bit instances.

```
sudo snap switch --channel=candidate amazon-ssm-agent
```

Snaps recommended on versions 18.04 and later

On Ubuntu Server 20.10 STR & 20.04 and 18.04 LTS, we recommend you only use Snaps. Also verify that only one instance of the agent is installed and running on your instances. If you want to use SSM Agent without Snaps, uninstall the SSM Agent. Then install the SSM Agent as a debian package and ensure you don't have any Snaps installed that overlap with the list of packages you want managed as debian packages.

Maximum timeout exceeded error message

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:

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

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

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

### To install SSM Agent on Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS 64-bit instances (with Snap package)

1. SSM Agent is installed, by default, on Ubuntu Server 20.04, 18.04, and 16.04 LTS 64-bit AMIs with an identifier of 20180627 or later.

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>.

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

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

```
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.

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

4. Check the status of the agent.

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

## Install SSM Agent on Ubuntu Server 16.04 and 14.04 64-bit (deb)

### Important

Before you install SSM Agent on a 64-bit version of Ubuntu Server, ensure sure that you are using the correction installation tools. Beginning with Amazon Machine Images (AMIs) that are identified with 20180627, SSM Agent is pre-installed on version 16.04 using Snap packages. On instances created from earlier AMIs, SSM Agent must be installed using deb installer packages. For more information, see [Determining the correct SSM Agent version to install on 64-bit Ubuntu Server 16.04 instances \(p. 112\)](#). 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 might fail.

In most cases, the Amazon Machine Images (AMIs) Ubuntu Server 16.04 that are provided by AWS come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For more information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

In the event that SSM Agent isn't preinstalled on a new Ubuntu Server 16.04 instance prior to version 20180627, you are installing on Ubuntu Server 14.04, or you need to manually reinstall the agent, use the information on this page to help you.

### Quick installation commands for SSM Agent on Ubuntu Server 16.04 and 14.04 64-bit (deb)

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

#### To install SSM Agent on Ubuntu Server 16.04 and 14.04 64-bit (deb) using quick copy and paste commands

1. Connect to your Ubuntu Server instance using your preferred method, such as SSH.
2. Run the following command to create a temporary directory on the instance.

```
mkdir /tmp/ssm
```

3. Change to the temporary directory.

```
cd /tmp/ssm
```

4. Run the following commands.

**Note**

Even though URLs in the following commands include an `ec2-downloads-windows` directory, these are the correct global installation files for Ubuntu Server 16.04 and 14.04 64-bit.

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

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

5. (Recommended) 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
```

In most cases, the command reports that the agent is running.

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

6. 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
```

### Create custom installation commands for SSM Agent on Ubuntu Server 16.04 and 14.04 64-bit (deb) in your Region

When you install SSM Agent on multiple instances using a script or template, we recommend using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

**Tip**

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on Ubuntu Server 16.04 and 14.04 64-bit \(deb\) \(p. 108\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).

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

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

See the following example.

```
wget https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/debian_amd64/amazon-ssm-agent.deb
```

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

## Install SSM Agent on Ubuntu Server 16.04 and 14.04 32-bit

In most cases, the Amazon Machine Images (AMIs) Ubuntu Server 16.04 that are provided by AWS come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For more information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

In the event that SSM Agent isn't preinstalled on a new Ubuntu Server 16.04 instance, you are installing on Ubuntu Server 14.04, or you need to manually reinstall the agent, use the information on this page to help you.

### Quick installation commands for SSM Agent on Ubuntu Server 16.04 and 14.04 32-bit (deb)

Use the following steps to manually install SSM Agent on a single instance. This procedure uses globally available installation files.

#### To install SSM Agent on Ubuntu Server 16.04 and 14.04 32-bit (deb) using quick copy and paste commands

1. Connect to your Ubuntu Server instance using your preferred method, such as SSH.
2. Run the following command to create a temporary directory on the instance.

```
mkdir /tmp/ssm
```

3. Change to the temporary directory.

```
cd /tmp/ssm
```

4. Run the following commands.

#### Note

Even though URLs in the following commands include an ec2-downloads-windows directory, these are the correct global installation files for Ubuntu Server 16.04 and 14.04 32-bit.

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

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

5. (Recommended) 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
```

In most cases, the command reports that the agent is running.

In rare cases, the command reports that the agent is installed but not running, as shown in the following example.

6. 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
```

## Create custom installation commands for SSM Agent on Ubuntu Server 16.04 and 14.04 32-bit (deb) in your Region

When you install SSM Agent on multiple instances using a script or template, we recommended using installation files that are stored in the AWS Region you're working in.

For the following commands, we provide examples that use a publicly accessible Amazon S3 bucket in the US East (Ohio) Region (`us-east-2`).

**Tip**

You can also replace a global URL in the procedure [Quick installation commands for SSM Agent on Ubuntu Server 16.04 and 14.04 34-bit \(deb\) \(p. 110\)](#) earlier in this topic with a custom Regional URL you construct.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).

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

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

See the following example.

```
wget https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/debian_386/amazon-ssm-agent.deb
```

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

## Determining the correct SSM Agent version to install on 64-bit Ubuntu Server 16.04 instances

### Important

Before you install SSM Agent on a 64-bit version of Ubuntu Server, ensure sure that you are using the correction installation tools. Beginning with Amazon Machine Images (AMIs) that are identified with 20180627, SSM Agent is pre-installed on version 16.04 using Snap packages. On instances created from earlier AMIs, SSM Agent must be installed using deb installer packages. For more information, see [Determining the correct SSM Agent version to install on 64-bit Ubuntu Server 16.04 instances \(p. 112\)](#)

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

You can verify the source AMI ID creation date for an instance using either of the following methods. These procedures apply only to AWS managed AMIs.

### Verify a source AMI ID creation date (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the left navigation pane, choose **Instances**.
3. Select an instance.
4. On the **Description** tab, check for a YYYYMMDD identifier in the value in the **AMI ID** field. For example: `ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20180627`.

### Verify a source AMI ID creation date (AWS CLI)

- Run the following command.

```
aws ec2 describe-images --image-ids ami-id
```

*ami-id* represents the ID of an AMI provided by AWS, such as `ami-07c8bc5c1ce9598c3`.

If successful, the command returns information like the following, in which you can check the `CreationDate` and `Name` fields for information.

```
{
 "Images": [
 {
 "Architecture": "x86_64",
 "CreationDate": "2020-07-24T20:40:27.000Z",
 "ImageId": "ami-07c8bc5c1ce9598c3",
 -- truncated --
 "ImageOwnerAlias": "amazon",
 "Name": "amzn2-ami-hvm-2.0.20200722.0-x86_64-gp2",
 "RootDeviceName": "/dev/xvda",
 "RootDeviceType": "ebs",
 "SriovNetSupport": "simple",
 "VirtualizationType": "hvm"
 }
]
}
```

## Verifying the signature of the SSM Agent

The AWS Systems Manager Agent (SSM Agent) deb and rpm installer packages for Linux instances are cryptographically signed. You can use a public key to verify that the agent package is original and unmodified. If there is any damage or alteration to the files, the verification fails. You can verify the signature of the installer package using either RPM or GPG. The following information is for SSM Agent versions 3.1.1141.0 or later.

To find the correct signature file for your instance's architecture and operating system, see the following table.

*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*.

| Architecture | Operating system                                               | Signature file URL                                                                                                | Agent download file name          |
|--------------|----------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| x86_64       | Amazon Linux, Amazon Linux 2, CentOS, RHEL, Oracle Linux, SLES | <code>https://s3.region.amazonaws.com/amazon-ssm-region/latest/linux_amd64/amazon-ssm-agent.rpm.sig</code>        | <code>amazon-ssm-agent.rpm</code> |
|              |                                                                | <code>https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm.sig</code>  |                                   |
| x86_64       | Debian Server, Ubuntu Server                                   | <code>https://s3.region.amazonaws.com/amazon-ssm-region/latest/debian_amd64/amazon-ssm-agent.deb.sig</code>       | <code>amazon-ssm-agent.deb</code> |
|              |                                                                | <code>https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_amd64/amazon-ssm-agent.deb.sig</code> |                                   |
| x86          | Amazon Linux, Amazon Linux 2, CentOS, RHEL                     | <code>https://s3.region.amazonaws.com/amazon-ssm-region/latest/linux_386/</code>                                  | <code>amazon-ssm-agent.rpm</code> |

| Architecture | Operating system                           | Signature file URL                                                                                                                                                                                                                                                                                                                                                                                                                 | Agent download file name          |
|--------------|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
|              |                                            | amazon-ssm-agent.rpm.sig<br><a href="https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_386/amazon-ssm-agent.rpm.sig">https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_386/amazon-ssm-agent.rpm.sig</a>                                                                                                                                                                                      |                                   |
| x86          | Ubuntu Server                              | <a href="https://s3.region.amazonaws.com/amazon-ssm-region/latest/debian_386/amazon-ssm-agent.deb.sig">https://s3.region.amazonaws.com/amazon-ssm-region/latest/debian_386/amazon-ssm-agent.deb.sig</a><br><a href="https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_386/amazon-ssm-agent.deb.sig">https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_386/amazon-ssm-agent.deb.sig</a>     | amazon-ssm- <del>agent</del> .deb |
| ARM64        | Amazon Linux, Amazon Linux 2, CentOS, RHEL | <a href="https://s3.region.amazonaws.com/amazon-ssm-region/latest/linux_arm64/amazon-ssm-agent.rpm.sig">https://s3.region.amazonaws.com/amazon-ssm-region/latest/linux_arm64/amazon-ssm-agent.rpm.sig</a><br><a href="https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_arm64/amazon-ssm-agent.rpm.sig">https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_arm64/amazon-ssm-agent.rpm.sig</a> | amazon-ssm- <del>agent</del> .rpm |

## GPG

### To verify the SSM Agent package on a Linux server

1. Copy the following public key, and save it to a file named `amazon-ssm-agent.gpg`.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQENBGIx/F/8BCADV014neDCfkpdj79/XVeQVy0Wz9LSiB/iksc1jTPaCgD/9ojdQ
10LfEFyLoeTEhX5WBu0Ry7oKw9AK51kscMjTHwdFnzXsT4tAoSXxh7lbgdfhpVm
bJ0bVArrzKIQ8JOE2lrn6LgVcGTtbPGURNNNRD1nZEgZm6wni+ZoplXmsj0WD7f
```

```
I5zhk/e+OyrsolpNWBJB0vf6JXVV2MauZKG1wRR4pZoSw5yPOa0rZDtOTtPbUX5C
lWGLtdQ3848YvgjMzK9GeEqK9n6yQx5potLvxJ6TCzsZTwXXF5LyPuv2y6U22075
JjMMX7noNnVnipKMj+17x5fis+X+gaff/PbTABEBAAG0J1NTTSBBZ2VudCA8c3Nt
LWFnZW50LXNpZ251ckBhbWF6b24uY29tPokBPwQTAQIAKQUCYjEX/wIbLwUJAsaY
gAcLCQgHAWIBBhUIAgkKCwQWAgnMBAh4BAheAAAOJEN2BphdWuqVJUKoIANHALkLq
xsUco2JwymOorf+1icVtL8MSdi871lhxfIGWaGN5CkzrkBAJ1Iyf/C+hVcLzR9rQ
DWIJakLWE3XPb4g8fWyr5V1oOYbcGLCky0fL500pWEnF2ecQMMSpwkdv9zx7qUoo
PssEpuwz5kIOYp2ENy21IPkMGpny8MCbzQ+sHysLWiJ/b0AWX9giPuMe5vTO3djm
CPtyA5CeG3BMawPoADQvjxB+DnWCg1Hs1gdzpZiSSusuZ8u3xKaehEMiB/Li2BO9
yZMAeG6iok4Dn01ZVvp9mftZKIm+T5WBX5x+TBhQ1b30MQcN61kFEe0Gll3ReTu
CPEuDwAb4WruFkaJAhwEEAECAYFAmIxGAAACgkQfdCXo9rX9fy5yQ/+PIBXWQc4
D/a6/nEaGM/FrLDLgPSieBCbU4TpVb7qPg6gJUX8CA+h8cZ06wDgcdi9sJ3MwTnQ
Ze1ozZ8AJroRP6XhwVeNEeedBbmr7irSg81IdyXZed0G0T+7SX/MDEyup16vRxW
k2UyBCXYqnxBHxeTKf9GxH0nODpcGPGBYqjfmsB3nj2wZNQg8SWWz6oEwcXv218B
FJyj7W2bQsbMXoH1ILP28Ec5QN1r8cC1b1nQsmx4120XSKFWvi8trG2+dDb58LR
1afsEW8OhJwsJcba1YIMznxMbWpfyZww2S6g7rFahm1wKCxMkHIZ+Fca6axKoK9Y
KJaEPn9rbhh11XsgKBNI1P1h0eGmQTAvM01dWI9895fiaK3pQkCxV7in6dTxi8Jy
7iBbORStxsospBJzLf+0Ca3yvILxySg1Q2EuOKuN2VW7N/13IfkJ85DVjjQgh6A
T4L6ViK/0L6ww5n8tboKB/J90UDGF2idxhqe8WenIogAU3y4ZGUyzcZHMg91Rke
hdLYGtqRATdWuwFQbwjPeBNovulqKOPXU9BLEezz8gMtd6/aW/UQA33xuZlh9590
DHhGwWDXEJzhrIlFAljkB7rsIhhjrg/R2usSIi78i1jFkGsVqRET2/avn7/kBcgL
yIK43DugjkN04nzHfULMJmEm02uVumgSJzQ=
=rGBs
-----END PGP PUBLIC KEY BLOCK-----
```

- Import the public key into your keyring, and note the returned key value.

```
gpg --import amazon-ssm-agent.gpg
```

- Verify the fingerprint. Be sure to replace **key-value** with the value from the preceding step. We recommend that you use GPG to verify the fingerprint even if you use RPM to verify the installer package.

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

This command returns output similar to the following.

```
pub 2048R/56BAA549 2022-03-15 [expires: 2023-09-05]
 Key fingerprint = 2BC7 C7C2 67BB D505 EAA4 91E6 DD81 A617 56BA A549
uid SSM Agent <ssm-agent-signer@amazon.com>
```

The fingerprint should match the following.

```
2BC7 C7C2 67BB D505 EAA4 91E6 DD81 A617 56BA A549
```

If the fingerprint doesn't match, don't install the agent. Contact AWS Support.

- Download the signature file according to your instance's architecture and operating system if you haven't already done so.
- Verify the installer package signature. Be sure to replace the **signature-filename** and **agent-download-filename** with the values you specified when downloading the signature file and agent.

```
gpg --verify signature-filename agent-download-filename
```

This command returns output similar to the following.

```
gpg: Signature made Mon 21 Mar 2022 05:52:47 PM UTC using RSA key ID 693ECA21
gpg: Good signature from "SSM Agent <ssm-agent-signer@amazon.com>"
gpg: WARNING: This key is not certified with a trusted signature!
```

```
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 2BC7 C7C2 67BB D505 EAA4 91E6 DD81 A617 56BA A549
```

If the output includes the phrase `BAD signature`, check whether you performed the procedure correctly. If you continue to get this response, contact AWS Support and don't install the agent. The warning message about the trust doesn't mean that the signature isn't valid, only that you haven't verified the public key. A key is trusted only if you or someone who you trust has signed it. If the output includes the phrase `Can't check signature: No public key`, verify you downloaded SSM Agent version 3.1.1141.0 or later.

## RPM

### To verify the SSM Agent package on a Linux server

1. Copy the following public key, and save it to a file named `amazon-ssm-agent.gpg`.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQENBGIxF/8BCAdv014neDCfkpdj79/XVeQVY0Wz9LSiB/iksc1jTPaCgD/9ojdQ
10LfFEyLoeTEhX5WBu0Ry70Kw9AK51kscMjTHwdFnzXst4tAoSXxh7lbgdfhpVm
bJ0bVArrzKIQ8JOE2lrn6LgVcGTTbPGURNNNRD1nZEqZm6wni+ZoplsXmsj0wD7f
I5zhk/e+OyrsolpNWBJB0vf6JXVV2MauZKG1wRR4pZoSv5yPOa0rZDtOTtPbUX5C
1WGLtdQ3848YvgjMzK9GeEqK9n6yQx5pot1vxJ6TCzsZTwXXF5LyPuv2y6U22075
JjMMX7noNnVnipKMj+17x5fis+X+gaffF/PbTABEBAAG0J1NTTSBBZ2VudCA8c3Nt
LWFNzW50LXNpZ251ckBhbWF6b24uY29tPokBPwQTAQIAKQUCYjEX/wIbLwUJAsaY
gAcLCQgHAWIBBHUIAgkKCwOWAgMBAh4BAheAAA0JEN2BphdWuqVJUKoIANHALkLq
xsUco2JwymOorf+1icVtL8MSdi871lhxf1GwaGN5CkzrkBAJ1Iyf/C+hVcLzR9rQ
DWIJakLWE3XPB4g8fWyr5VlOoYbcGLCky0fL500pWEf2ecQMMSpwkdv9zx7qUoo
PssEpuwz5kIOYp2ENy21IPkMGpny8MCbzQ+sHysLWiJ/b0aWX9giPuMe5vTO3djM
CPtyA5CeG3BMawPOaDQvjxkB+DnWCg1Hs1gdzpZiSsusuz8u3xKaehEMiB/Li2B09
yZMAeG6iok4Dn01ZVVpU9mfTzKIm/T5WBX5x+TBhQ1b30McN61kFEEoG1l3ReTu
CPEuDwAb4WruFkaJAhwEEAECAAYFamIxGAAACgkQfdCx09rX9fy5yQ/+PIBXWQc4
D/a6/nEaGM/FrLDLgPSieBCbU4TpVb7qPg6gJUX8CA+h8cZ06wDgcdi9sJ3MwTnQ
Ze1OzZ8AJroRP6XhwVeNEbeedBbmr7irSg81IdyXZed0G0T+7SX/MDEyup16vRxW
k2UyBCXYqnxBHxeTKf9GxH0n0DpcGPGByqjfmsB3nj2wZN0g8SWWz6oEWcXv218B
FJyj7W2bQsbMXoH1ILP28Ec5QN1r8cC1b1nQsmx4120XSKFWvi8trG2+dDb58LR
1af8ew8OhJwsJcba1YIMznxMbWpfyZww2S6g7rFahm1wKCxMkHIZ+Fca6axKoK9Y
KJaEPn9rbhh11XsgKBNIIP1h0eGmQTAvm01dWI985fiaK3pQkCxV7in6dTxi8Jy
7iJbbStxssobBJzLf+0Ca3yvILxySg1Q2EuOKuN2VW7N/13iffJ85DVjjQgh6A
T4L6ViK/0L6ww5n8tboKB/Jz9UDGF2idxhqe8WenIogAU3y4ZGUyzcZHmg91Rke
hdLYGtqRATdWuwFQbwjPeBNovulqKOPXU9BLEezz8gMtd6/aW/UQA33xuZlh959o
DHhGwWDXEJzhr1lFAljkB7rsIhhjrg/R2usSIi78i1jFkGsVqRET2/avn7/kBcgL
yIk43DugjkN04nzHfULMJmEm02uVumgSJzQ=
=rGEs
-----END PGP PUBLIC KEY BLOCK-----
```

2. Import the public key into your keyring, and note the returned key value.

```
rpm --import amazon-ssm-agent.gpg
```

3. Verify the fingerprint. Be sure to replace `key-value` with the value from the preceding step. We recommend that you use GPG to verify the fingerprint even if you use RPM to verify the installer package.

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

This command returns output similar to the following.

```
pub 2048R/56BAA549 2022-03-15 [expires: 2023-09-05]
Key fingerprint = 2BC7 C7C2 67BB D505 EAA4 91E6 DD81 A617 56BA A549
uid SSM Agent <ssm-agent-signer@amazon.com>
```

The fingerprint should match the following.

```
2BC7 C7C2 67BB D505 EAA4 91E6 DD81 A617 56BA A549
```

If the fingerprint doesn't match, don't install the agent. Contact AWS Support.

4. Verify the installer package signature. Be sure to replace the *signature-filename* and *agent-download-filename* with the values you specified when downloading the signature file and agent.

```
rpm --checksig agent-download-filename
```

This command returns output similar to the following.

```
amazon-ssm-agent-3.1.1141.0-1.amzn2.x86_64.rpm: rsa sha1 (md5) pgp md5 OK
```

If pgp is missing from the output and you have imported the public key, then the agent isn't signed. If the output contains the phrase NOT OK (MISSING KEYS: (MD5) *key-id*), check whether you performed the procedure correctly and verify you downloaded SSM Agent version 3.1.1141.0 or later. If you continue to get this response, contact AWS Support and don't install the agent.

## Configuring SSM Agent to use a proxy (Linux)

You can configure AWS Systems Manager Agent (SSM Agent) to communicate through an HTTP proxy by creating an override configuration file and adding `http_proxy`, `https_proxy`, and `no_proxy` settings to the file. An override file also preserves the proxy settings if you install newer or older versions of SSM Agent. This section includes procedures for creating an override file in both `upstart` and `systemd` environments.

### Note

Managed nodes created from an Amazon Linux AMI that use a proxy must run a current version of the Python `requests` module to support Patch Manager operations. For more information, see [Upgrading the Python requests module on Amazon Linux instances that use a proxy server \(p. 119\)](#).

### Topics

- [Configure SSM Agent to use a proxy \(upstart\) \(p. 117\)](#)
- [Configure SSM Agent to use a proxy \(systemd\) \(p. 118\)](#)
- [Upgrading the Python requests module on Amazon Linux instances that use a proxy server \(p. 119\)](#)

## Configure SSM Agent to use a proxy (upstart)

Use the following procedure to create an override configuration file for an `upstart` environment.

### To configure SSM Agent to use a proxy (upstart)

1. Connect to the managed 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, add one of the following configurations.

**For an HTTP proxy server:**

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

**For an HTTPS proxy server:**

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

**Note**

Add the `no_proxy` setting to the file and specify the IP address listed here. It's the instance metadata endpoint for Systems Manager. Without this IP address, calls to Systems Manager fail.

3. Save the file with the name `amazon-ssm-agent.override` in the following location: `/etc/init/`
4. Stop and restart SSM Agent using the following commands.

```
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)

Use the following procedure to configure SSM Agent to use a proxy in a `systemd` environment.

**Note**

Some of the steps in this procedure contain explicit instructions for Ubuntu Server instances where SSM Agent was installed using Snap.

1. Connect to the instance where you installed SSM Agent.
2. Run one of the follow commands, depending on the operating system type.
  - On Ubuntu Server instances where SSM Agent is installed by using a snap:

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

On other operating systems:

```
sudo systemctl edit 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, add one of the following configurations.

**For an HTTP proxy server:**

```
[Service]
```

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

**For an HTTPS proxy server:**

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

**Note**

Add the `no_proxy` setting to the file and specify the IP address listed here. It's the instance metadata endpoint for Systems Manager. Without this IP address, calls to Systems Manager fail.

4. Save your changes. The system automatically creates one of the following files, depending on the operating system type.

- On Ubuntu Server instances where SSM Agent is installed by using a snap:

```
/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-
agent.service.d/override.conf
```

- On Amazon Linux 2 instances:

```
/etc/systemd/system/amazon-ssm-agent.service.d/override.conf
```

- On other operating systems:

```
/etc/systemd/system/amazon-ssm-agent.service.d/amazon-ssm-agent.override
```

5. Restart SSM Agent by using one of the following commands, depending on the operating system type.

- On Ubuntu Server instances installed by using a snap:

```
sudo systemctl daemon-reload && sudo systemctl restart snap.amazon-ssm-agent.amazon-
ssm-agent
```

- On other operating systems:

```
sudo systemctl daemon-reload && sudo systemctl restart amazon-ssm-agent
```

**Note**

For more information about working with `.override` files in systemd environments, see [Modifying Existing Unit Files](#) in the *Red Hat Enterprise Linux 7 System Administrator's Guide*.

## Upgrading 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, a capability of AWS Systems 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 Systems Manager document (SSM document) [AWS-RunShellScript](#) in Run Command, a capability of AWS Systems Manager, 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 isn't installed, run the following command to install it.

```
pip install requests
```

- If pip itself isn't 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
```

## Uninstalling SSM Agent from Linux instances

Use the following commands to uninstall AWS Systems Manager Agent (SSM Agent).

### Amazon Linux, Amazon Linux 2, CentOS, Oracle Linux, and Red Hat Enterprise Linux

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

### Debian Server

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

### SUSE Linux Enterprise Server (SLES)

- `zypper` command installations:

```
sudo zypper remove amazon-ssm-agent
```

- `rpm` command installations:

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

### Ubuntu Server

- `deb` package installations:

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

- **snap package installations:**

```
sudo snap remove amazon-ssm-agent
```

## Working with SSM Agent on EC2 instances for macOS

AWS Systems Manager (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 for macOS.

**Note**

SSM Agent is preinstalled, by default, on Amazon Machine Images (AMIs) for macOS. You don't need to install SSM Agent on an Amazon Elastic Compute Cloud (Amazon EC2) instance for macOS unless you have uninstalled it.

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 doesn't provide support for running modified copies of this software.

**Note**

To view details about the different versions of SSM Agent, see the [release notes](#).

Before you manually install SSM Agent on a macOS operating system, review the following information.

- SSM Agent is installed by default on the following EC2 instances and Amazon Machine Images:
  - macOS 10.14.x (Mojave)
  - macOS 10.15.x (Catalina)
  - macOS 11.x (BigSur)

SSM Agent doesn't need to be manually installed on macOS EC2 instances unless it has been uninstalled.

- Systems Manager doesn't support macOS in hybrid environments.
- An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

**Topics**

- [Manually installing SSM Agent on EC2 instances for macOS \(p. 121\)](#)
- [Configure SSM Agent to use a proxy \(macOS\) \(p. 122\)](#)
- [Uninstall SSM Agent from macOS instances \(p. 122\)](#)

## Manually installing SSM Agent on EC2 instances for macOS

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

## To install SSM Agent on macOS

1. Download the agent installer file using the following command.

```
sudo wget https://s3.region.amazonaws.com/amazon-ssm-region/latest/darwin_amd64/amazon-ssm-agent.pkg
```

Here is an example.

```
sudo wget https://s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/latest/darwin_amd64/amazon-ssm-agent.pkg
```

2. Use the following command to run the SSM Agent installer.

x86\_64:

```
sudo installer -pkg amazon-ssm-agent.pkg -target /
```

3. Check the status of the agent.

To determine if SSM Agent is running, check the agent log at `/var/log/amazon/ssm/amazon-ssm-agent.log`.

4. Run the following command to start the service if the the agent log indicates that "amazon-ssm-agent is stopped."

```
sudo launchctl load -w /Library/LaunchDaemons/com.amazon.aws.ssm.plist && sudo launchctl start com.amazon.aws.ssm
```

### Important

An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

## Configure SSM Agent to use a proxy (macOS)

You can configure AWS Systems Manager Agent (SSM Agent) to communicate through an HTTP proxy by adding web proxy, secure web proxy, and bypass proxy settings to the Network configuration on macOS Amazon Elastic Compute Cloud (Amazon EC2) instances. For more information, consult your macOS user documentation.

## Uninstall SSM Agent from macOS instances

macOS doesn't natively support uninstallation of PKG files. To uninstall AWS Systems Manager Agent (SSM Agent) from an Amazon Elastic Compute Cloud (Amazon EC2) instance for macOS, you can use the AWS managed script from the following location.

<https://github.com/aws/amazon-ssm-agent/blob/mainline/Tools/src/update/darwin/uninstall.sh>

# Working with SSM Agent on EC2 instances for Windows Server

AWS Systems Manager Agent (SSM Agent) is preinstalled, by default, on the following Amazon Machine Images (AMIs) for Windows Server:

- Windows Server 2008-2012 R2 AMIs published in November 2016 or later
- Windows Server 2016, 2019, and 2022

## Important

As of January 14, 2020, Windows Server 2008 is no longer supported for feature or security updates from Microsoft. Legacy Amazon Machine Images (AMIs) for Windows Server 2008 and 2008 R2 still include version 2 of SSM Agent preinstalled, but Systems Manager no longer officially supports 2008 versions and no longer updates the agent for these versions of Windows Server. In addition, [SSM Agent version 3.0 \(p. 75\)](#) might not be compatible with all operations on Windows Server 2008 and 2008 R2. The final officially supported version of SSM Agent for Windows Server 2008 versions is 2.3.1644.0.

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 Amazon Elastic Compute Cloud (Amazon 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. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

To view details about the different versions of SSM Agent, see the [release notes](#).

## Topics

- [Manually installing SSM Agent on EC2 instances for Windows Server \(p. 123\)](#)
- [Configure SSM Agent to use a proxy for Windows Server instances \(p. 125\)](#)

## Manually installing SSM Agent on EC2 instances for Windows Server

AWS Systems Manager Agent (SSM Agent) is preinstalled, by default, on the following Amazon Machine Images (AMIs) for Windows Server:

- Windows Server 2008-2012 R2 AMIs published in November 2016 or later
- Windows Server 2016, 2019, and 2022

If your managed 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 allows you to use SSM features published after November 2016. For information about how to install the latest version of the EC2Config service, see [Install 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. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

If necessary, you can manually download and install the latest version of SSM Agent on your Amazon Elastic Compute Cloud (Amazon 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 \(Windows\) \(p. 50\)](#).

**To manually install the latest version of SSM Agent on EC2 instances for Windows Server**

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.region.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 allow 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

```
$progressPreference = 'silentlyContinue'
Invoke-WebRequest `
 https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/AmazonSSMAgentSetup.exe `
 -Outfile $env:USERPROFILE\Desktop\SSMAgent_latest.exe
```

32-bit

```
$progressPreference = 'silentlyContinue'
Invoke-WebRequest `
 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](https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/AmazonSSMAgentSetup.exe)

Run the downloaded `AmazonSSMAgentSetup.exe` file to install SSM Agent.

3. Start or restart SSM Agent by sending the following command in PowerShell:

```
Restart-Service AmazonSSMAgent
```

### Important

SSM Agent requires Windows PowerShell 3.0 or later to run certain AWS Systems Manager documents (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

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

If your instance is a Windows Server 2008-2012 R2 instance created *before* November 2016, then EC2Config processes AWS 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 AWS Systems Manager Agent (SSM Agent) side-by-side with EC2Config. This side-by-side version of SSM Agent is compatible with your instances created from earlier Windows Amazon Machine Images (AMIs) and allows you to use Systems Manager features published after November 2016. For information about how to install the latest version of the EC2Config service, see [Install the latest version of EC2Config](#) in the *Amazon EC2 User Guide for Windows Instances*. If you don't upgrade to the latest version of EC2Config and use EC2Config to process Systems Manager requests, configure proxy settings for EC2Config. For information about configuring EC2Config to use a proxy, see [Configure proxy settings for the EC2Config service](#) in the *Amazon EC2 User Guide for Windows Instances*.

### Note

As of January 14, 2020, Windows Server 2008 is no longer supported for feature or security updates from Microsoft. Legacy Amazon Machine Images (AMIs) for Windows Server 2008 and 2008 R2 still include version 2 of SSM Agent preinstalled, but Systems Manager no longer officially supports 2008 versions and no longer updates the agent for these versions of Windows Server. In addition, [SSM Agent version 3.0 \(p. 75\)](#) might not be compatible with all operations on Windows Server 2008 and 2008 R2. The final officially supported version of SSM Agent for Windows Server 2008 versions is 2.3.1644.0.

## 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.

```
$serviceKey = "HKLM:\SYSTEM\CurrentControlSet\Services\AmazonSSMAgent"
$keyInfo = (Get-Item -Path $serviceKey).GetValue("Environment")
$proxyVariables = @("http_proxy=hostname:port", "https_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 [Viewing SSM Agent logs \(p. 132\)](#).

```
2020-02-24 15:31:54 INFO Getting IE proxy configuration for current user: The operation completed successfully.
2020-02-24 15:31:54 INFO Getting WinHTTP proxy default configuration: The operation completed successfully.
2020-02-24 15:31:54 INFO Proxy environment variables:
2020-02-24 15:31:54 INFO http_proxy: hostname:port
2020-02-24 15:31:54 INFO https_proxy: hostname:port
2020-02-24 15:31:54 INFO no_proxy: 169.254.169.254
2020-02-24 15:31:54 INFO Starting Agent: amazon-ssm-agent - v2.3.871.0
2020-02-24 15:31:54 INFO OS: windows, Arch: amd64
```

## 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.

```
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.

### Important

SSM Agent communicates using the HTTPS protocol. For this reason, you must configure the `HTTPS_proxy` parameter by using one of the following settings options.

SSM Agent proxy settings are evaluated in the following order.

1. AmazonSSMAgent Registry settings (`HKEY_LOCAL_MACHINE\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. Internet Explorer settings (HTTP, secure, exceptions)
5. WinHTTP proxy settings (`http=`, `https=`, `bypass-list=`)

## SSM Agent proxy settings and Systems Manager services

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

For Run Command, configure `WinINet` proxy settings on your Windows Server instances. The `[System.Net.WebRequest]` commands provided are per-session. To apply these configurations to subsequent network commands that are run in Run Command, these commands must precede other PowerShell commands in the same `aws:runPowerShellScript` plugin input.

The following PowerShell commands return the current `WinINet` proxy settings, and apply your proxy settings to `WinINet`.

```
[System.Net.WebRequest]::DefaultWebProxy

$proxyServer = "http://hostname:port"
$proxyBypass = "169.254.169.254"
$WebProxy = New-Object System.Net.WebProxy($proxyServer,$true,$proxyBypass)

[System.Net.WebRequest]::DefaultWebProxy = $WebProxy
```

For Patch Manager, 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 proxy-server="hostname:port" bypass-list="169.254.169.254"
```

For more information about using Run Command, see [Sending commands using Systems Manager Run Command \(p. 995\)](#).

## Working with SSM Agent on edge devices

Systems Manager supports installing and running SSM Agent on AWS IoT Greengrass core devices, AWS IoT, and non-AWS IoT devices. The SSM Agent installation and configuration process for AWS IoT Greengrass core devices is different than AWS IoT and non-AWS IoT devices. For more information, see [Setting up AWS Systems Manager for edge devices \(p. 52\)](#).

### Note

For information about uninstalling SSM Agent from an edge device, see [Uninstall the AWS Systems Manager Agent](#) in the *AWS IoT Greengrass Version 2 Developer Guide*.

# Checking SSM Agent status and starting the agent

This topic lists the commands to check whether AWS Systems Manager Agent (SSM Agent) is running on each supported operating system. It also provides the commands to start the agent if it isn't running.

| <b>Operating system</b>                     | <b>Command to check SSM Agent status</b>                                             | <b>Command to start SSM Agent</b>                                                                                                           |
|---------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Amazon Linux                                | <code>sudo status amazon-ssm-agent</code>                                            | <code>sudo start amazon-ssm-agent</code>                                                                                                    |
| Amazon Linux 2                              | <code>sudo systemctl status amazon-ssm-agent</code>                                  | <code>sudo systemctl enable amazon-ssm-agent</code><br><code>sudo systemctl start amazon-ssm-agent</code>                                   |
| CentOS 6.x                                  | <code>sudo status amazon-ssm-agent</code>                                            | <code>sudo start amazon-ssm-agent</code>                                                                                                    |
| CentOS 7.x and CentOS 8.x                   | <code>sudo systemctl status amazon-ssm-agent</code>                                  | <code>sudo systemctl enable amazon-ssm-agent</code><br><code>sudo systemctl start amazon-ssm-agent</code>                                   |
| Debian Server 8, 9, and 10                  | <code>sudo systemctl status amazon-ssm-agent</code>                                  | <code>sudo systemctl enable amazon-ssm-agent</code><br><code>sudo systemctl start amazon-ssm-agent</code>                                   |
| macOS                                       | Check the agent log file at<br><code>/var/log/amazon/ssm/amazon-ssm-agent.log</code> | <code>sudo launchctl load -w /Library/LaunchDaemons/com.amazon.aws.ssm.plist</code><br><code>sudo launchctl start com.amazon.aws.ssm</code> |
| Oracle Linux                                | <code>sudo systemctl status amazon-ssm-agent</code>                                  | <code>sudo systemctl enable amazon-ssm-agent</code><br><code>sudo systemctl start amazon-ssm-agent</code>                                   |
| Red Hat Enterprise Linux (RHEL) 6.x         | <code>sudo status amazon-ssm-agent</code>                                            | <code>sudo start amazon-ssm-agent</code>                                                                                                    |
| Red Hat Enterprise Linux (RHEL) 7.x and 8.x | <code>sudo systemctl status amazon-ssm-agent</code>                                  | <code>sudo systemctl enable amazon-ssm-agent</code><br><code>sudo systemctl start amazon-ssm-agent</code>                                   |
| SUSE Linux Enterprise Server (SLES)         | <code>sudo systemctl status amazon-ssm-agent</code>                                  | <code>sudo systemctl enable amazon-ssm-agent</code><br><code>sudo systemctl start amazon-ssm-agent</code>                                   |

| Operating system                                                                             | Command to check SSM Agent status                                                 | Command to start SSM Agent                                                                                |
|----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| Ubuntu Server 14.04 (all) and 16.04 (32-bit)                                                 | <code>sudo status amazon-ssm-agent</code>                                         | <code>sudo start amazon-ssm-agent</code>                                                                  |
| Ubuntu Server 16.04 64-bit instances (deb package installation)                              | <code>sudo systemctl status amazon-ssm-agent</code>                               | <code>sudo systemctl enable amazon-ssm-agent</code><br><code>sudo systemctl start amazon-ssm-agent</code> |
| Ubuntu Server 16.04, 18.04, and 20.04 LTS & and 20.10 STR 64-bit (Snap package installation) | <code>sudo systemctl status snap.amazon-ssm-agent.amazon-ssm-agent.service</code> | <code>sudo snap start amazon-ssm-agent</code>                                                             |
| Windows Server                                                                               | <i>Run in PowerShell:</i><br><code>Get-Service AmazonSSMAgent</code>              | <i>Run in PowerShell Administrator mode:</i><br><code>Start-Service AmazonSSMAgent</code>                 |

### Related content

- [Working with SSM Agent on EC2 instances for Linux \(p. 76\)](#)
- [Working with SSM Agent on EC2 instances for Windows Server \(p. 123\)](#)
- [Checking the SSM Agent version number \(p. 129\)](#)

## Checking the SSM Agent version number

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

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

### To check the version number of SSM Agent installed on a managed node

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Note the **Agent version**.

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

Choose from the following tabs to get the currently installed SSM Agent version from within an operating system.

## Amazon Linux and Amazon Linux 2

### Note

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

1. Log in to your managed node.
2. Run the following command.

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

This command returns output similar to the following.

```
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Installed Packages
Name : amazon-ssm-agent
Arch : x86_64
Version : 3.0.655.0
```

## CentOS

1. Log in to your managed node.
2. Run the following command for CentOS 6 and 7.

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

This command returns output similar to the following.

```
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Installed Packages
Name : amazon-ssm-agent
Arch : x86_64
Version : 3.0.655.0
```

## Debian Server

1. Log in to your managed node.
2. Run the following command.

```
apt list amazon-ssm-agent
```

This command returns output similar to the following.

```
apt list amazon-ssm-agent
Listing... Done
amazon-ssm-agent/n now 3.0.655.0-1 amd64 [installed,local]

3.0.655.0 is the version of SSM agent
```

## macOS

1. Log in to your managed node.
2. Run the following command.

```
pkgutil --pkg-info com.amazon.aws.ssm
```

#### RHEL

1. Log in to your managed node.
2. Run the following command for RHEL 6 and 7.

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

This command returns output similar to the following.

```
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Installed Packages
Name : amazon-ssm-agent
Arch : x86_64
Version : 3.0.655.0
```

#### SLES

1. Log in to your managed node.
2. Run the following command for SLES 12 and 15.

```
zypper info amazon-ssm-agent
```

This command returns output similar to the following.

```
Loading repository data...
Reading installed packages...
Information for package amazon-ssm-agent:

Repository : @System
Name : amazon-ssm-agent
Version : 3.0.655.0-1
```

#### Ubuntu Server

##### Note

To check if your Ubuntu Server 16.04 instance uses deb or Snap packages, see [Manually installing SSM Agent on Ubuntu Server instances \(p. 106\)](#).

1. Log in to your managed node.
2. Run the following command for Ubuntu Server 16.04 and 14.04 64-bit (with deb installer package).

```
apt list amazon-ssm-agent
```

This command returns output similar to the following.

```
apt list amazon-ssm-agent
Listing... Done
amazon-ssm-agent/nod 3.0.655.0-1 amd64 [installed,local]
```

3.0.655.0 is the version of SSM agent

Run the following command for Ubuntu Server 20.10 STR and 20.04, 18.04, and 16.04 LTS 64-bit instances (with Snap package).

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

This command returns output similar to the following.

```
snap list amazon-ssm-agent
Name Version Rev Tracking Publisher Notes
amazon-ssm-agent 3.0.529.0 3552 latest/stable/... aws# classic-
3.0.529.0 is the version of SSM agent
```

## Windows

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

```
& "C:\Program Files\Amazon\SSM\amazon-ssm-agent.exe" --version
```

This command returns output similar to the following.

```
SSM Agent version: 3.1.804.0
```

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 [Automating updates to SSM Agent \(p. 135\)](#).

# Viewing SSM Agent logs

AWS Systems Manager Agent (SSM Agent) writes information about executions, commands, scheduled actions, errors, and health statuses to log files on each managed node. You can view log files by manually connecting to a managed node, or you can automatically send logs to Amazon CloudWatch Logs. For more information about sending logs to CloudWatch Logs, see [Monitoring AWS Systems Manager \(p. 1428\)](#).

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

### Linux and macOS

/var/log/amazon/ssm/

### Windows

%PROGRAMDATA%\Amazon\SSM\Logs\

For Linux managed nodes, the SSM Agent `stderr` and `stdout` files are written to the following directory: /var/lib/amazon/ssm/.

For Windows managed nodes, the SSM Agent `stderr` and `stdout` files are written to the following directory: `%PROGRAMDATA%\Amazon\SSM\InstanceData\`.

For information about allowing SSM Agent debug logging, see [Allowing SSM Agent debug logging \(p. 133\)](#).

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.

## Allowing SSM Agent debug logging

Use the following procedure to allow SSM Agent debug logging on your managed nodes.

Linux and macOS

### To allow SSM Agent debug logging on Linux and macOS managed nodes

1. Either use Session Manager, a capability of AWS Systems Manager, to connect to the managed node where you want to allow debug logging, or log on to the managed node. For more information, see [Working with Session Manager \(p. 964\)](#).
2. Locate the `seelog.xml.template` file.

**Linux:**

On most Linux managed node types, the file is located in the directory `/etc/amazon/ssm/seelog.xml.template`.

On Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS, the file is located in the directory `/snap/amazon-ssm-agent/current/seelog.xml.template`. Copy this file from the `/snap/amazon-ssm-agent/current/` directory to the `/etc/amazon/ssm/` directory before making any changes.

**macOS:**

On macOS instance types, the file is located in the directory `/opt/aws/ssm/seelog.xml.template`.

3. Change the file name from `seelog.xml.template` to `seelog.xml`.

#### Note

On Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS, the file `seelog.xml` must be created in the directory `/etc/amazon/ssm/`. You can create this directory and file by running the following commands.

```
sudo mkdir -p /etc/amazon/ssm
```

```
sudo cp -p /snap/amazon-ssm-agent/current/seelog.xml.template /etc/amazon/ssm/seelog.xml
```

4. 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.

```
<seelog type="adaptive" mininterval="2000000" maxinterval="100000000"
critmsgcount="500" minlevel="debug">
```

5. (Optional) Restart SSM Agent using the following command.

**Linux:**

```
sudo service amazon-ssm-agent restart
```

macOS:

```
sudo /opt/aws/ssm/bin/amazon-ssm-agent restart
```

Windows

### To allow SSM Agent debug logging on Windows Server managed nodes

1. Either use Session Manager to connect to the managed node where you want to allow debug logging, or log on to the managed nodes. For more information, see [Working with Session Manager \(p. 964\)](#).
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.

```
<seelog type="adaptive" mininterval="2000000" maxinterval="100000000"
critmsgcount="500" minlevel="debug">
```

4. Locate the following entry.

```
filename="{{LOCALAPPDATA}}\Amazon\SSM\Logs\amazon-ssm-agent.log"
```

Change this entry to use the following path.

```
filename="C:\ProgramData\Amazon\SSM\Logs\amazon-ssm-agent.log"
```

5. Locate the following entry.

```
filename="{{LOCALAPPDATA}}\Amazon\SSM\Logs\errors.log"
```

Change this entry to use the following path.

```
filename="C:\ProgramData\Amazon\SSM\Logs\errors.log"
```

6. Restart SSM Agent using the following PowerShell command in Administrator mode.

```
Restart-Service AmazonSSMAgent
```

## Restricting access to root-level commands through SSM Agent

AWS Systems Manager Agent (SSM Agent) runs on Amazon Elastic Compute Cloud (Amazon EC2) instances using root permissions (Linux) or SYSTEM permissions (Windows Server). Because these are the highest level of system access permissions, 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 AWS Identity and Access Management (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 managed nodes 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 managed nodes that are tagged with specific tags or a combination of tags.

For example, say you have two fleets of EC2 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 doesn't 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 tags \(p. 993\)](#)
- [Restrict session access based on instance tags \(p. 936\)](#)

## Automating updates to SSM Agent

AWS releases a new version of AWS Systems Manager Agent (SSM Agent) when we add or update Systems Manager capabilities. If your managed nodes 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 managed nodes using any of the following methods.

| Method                                                        | Details                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| One-click automated update on all managed nodes (Recommended) | <p>You can configure all managed nodes in your AWS account to automatically check for and download new versions of SSM Agent. To do this, choose <b>Agent auto update</b> on the <b>Managed instances</b> page in the Systems Manager console, as described later in this topic.</p> <p><b>Note</b><br/>One-click automated updates aren't supported for macOS Amazon Elastic Compute Cloud (Amazon EC2) instances.</p> |
| Global or selective update                                    | <p>You can use State Manager, a capability of AWS Systems Manager, to create an association that automatically downloads and installs SSM Agent on your managed nodes. If you want to limit the disruption to your workloads, you can create a Systems Manager maintenance window to perform the installation during designated</p>                                                                                     |

| Method                                          | Details                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                 | time periods. Both methods allow you to create either a global update configuration for all of your managed nodes or selectively choose which instances get updated. For information about creating a State Manager association, see <a href="#">Walkthrough: Automatically update SSM Agent (CLI) (p. 1089)</a> . For information about using a maintenance window, see <a href="#">Walkthrough: Create a maintenance window to update SSM Agent (AWS CLI) (p. 777)</a> and <a href="#">Walkthrough: Create a maintenance window to automatically update SSM Agent (console) (p. 782)</a> . |
| Global or selective update for new environments | If you're getting started with Systems Manager, we recommend that you use the <b>Update Systems Manager (SSM) Agent every two weeks</b> option in Quick Setup, a capability of AWS Systems Manager. Quick Setup allows you to create either a global update configuration for all of your managed nodes or selectively choose which managed nodes get updated. For more information, see <a href="#">AWS Systems Manager Quick Setup (p. 145)</a> .                                                                                                                                          |

If you prefer to update SSM Agent on your managed nodes manually, you can subscribe to notifications that AWS publishes when a new version of the agent is released. For information, see [Subscribing to SSM Agent notifications \(p. 137\)](#). After you subscribe to notifications, you can use Run Command to manually update one or more managed nodes with the latest version. For more information, see [Update SSM Agent by using Run Command \(p. 997\)](#).

## Automatically updating SSM Agent

You can configure Systems Manager to automatically update SSM Agent on all Linux-based and Windows-based managed nodes in your AWS account. (One-click automated updates aren't supported for macOS EC2 instances.) If you turn on 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 managed nodes 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 node 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the **Settings** tab, and then choose **Auto update SSM Agent** under **Agent auto update**.

To change the version of SSM Agent your fleet updates to, choose **Edit** under **Agent auto update** on the **Settings** tab. Then enter the version number of SSM Agent you want to update to in **Version** under **Parameters**. If not specified, the agent updates to the latest version.

To stop automatically deploying updated versions of SSM Agent to all managed nodes in your account, choose **Delete** under **Agent auto update** on the **Settings** tab. This action deletes the State Manager association that automatically updates SSM Agent on your managed nodes.

## Subscribing to SSM Agent notifications

Amazon Simple Notification Service (Amazon SNS) can notify you when new versions of AWS Systems Manager Agent (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

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/v3/home>.
2. From the Region selector in the navigation bar, choose **US East (N. Virginia)**, if it isn't selected already. You must select this AWS Region because the Amazon SNS notifications for SSM Agent that you're 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**, enter an email address that you can use to receive the notifications. If you choose **SMS**, enter an area code and number.
  - d. Choose **Create subscription**.
6. If you chose **Email**, you will receive an email message asking you to confirm your subscription. Open the message, 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**.

# SSM Agent communications with AWS managed S3 buckets

In the course of performing various Systems Manager operations, AWS Systems Manager Agent (SSM Agent) accesses a number of Amazon Simple Storage Service (Amazon S3) buckets. These S3 buckets are publicly accessible, and by default, SSM Agent connects to them using [HTTP](#) calls.

However, if you're using a virtual private cloud (VPC) endpoint in your Systems Manager operations, you must provide explicit permission in an Amazon Elastic Compute Cloud (Amazon EC2) instance profile for Systems Manager, or in a service role for instances in a hybrid environment. Otherwise, your resources can't access these public buckets.

To grant your managed nodes access to these buckets when you are using a VPC endpoint, you create a custom Amazon S3 permissions policy, and then attach it to your instance profile (for EC2 instances) or your service role (for AWS IoT Greengrass core devices and for on-premises servers, edge devices, and virtual machines in a hybrid environment).

#### 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 \(p. 21\)](#)
- [Create an IAM service role for a hybrid environment \(p. 36\)](#)

#### Contents

- [Required bucket permissions \(p. 138\)](#)
- [Example \(p. 142\)](#)

## Required bucket permissions

The following table describes each of the S3 buckets that SSM Agent might need to access for Systems Manager operations.

#### 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](#).

Amazon S3 permissions required by SSM Agent

| S3 bucket ARN                                                                         | Description                                                                                                                                                                                                                                          |
|---------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>arn:aws:s3:::aws-windows-downloads-<a href="#"><code>region</code></a>/*</code> | Required for some SSM documents that support only Windows operating systems.                                                                                                                                                                         |
| <code>arn:aws:s3:::amazon-ssm-<a href="#"><code>region</code></a>/*</code>            | Required for updating SSM Agent installations. These buckets contain the SSM Agent installation packages, and the installation manifests that are referenced by the <code>AWS-UpdateSSMAgent</code> document and plugin. If these permissions aren't |

| S3 bucket ARN                                                          | Description                                                                                                                                                                                                                                                                                                                            |
|------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                        | provided, the SSM Agent makes an HTTP call to download the update.                                                                                                                                                                                                                                                                     |
| <code>arn:aws:s3:::amazon-ssm-packages-<i>region</i>/*</code>          | Required for using versions of SSM Agent prior to 2.2.45.0 to run the SSM document <code>AWS-ConfigureAWSPackage</code> .                                                                                                                                                                                                              |
| <code>arn:aws:s3:::<i>region</i>-birdwatcher-prod/*</code>             | Provides access to the distribution service used by version 2.2.45.0 and later of SSM Agent. This service is used to run the document <code>AWS-ConfigureAWSPackage</code> .<br><br>This permission is needed for all AWS Regions <i>except</i> the Africa (Cape Town) Region (af-south-1) and the Europe (Milan) Region (eu-south-1). |
| <code>arn:aws:s3:::aws-ssm-distributor-file-<i>region</i>/*</code>     | Provides access to the distribution service used by version 2.2.45.0 and later of SSM Agent. This service is used to run the SSM document <code>AWS-ConfigureAWSPackage</code> .<br><br>This permission is needed <i>only</i> for the Africa (Cape Town) Region (af-south-1) and the Europe (Milan) Region (eu-south-1).               |
| <code>arn:aws:s3:::aws-ssm-document-attachments-<i>region</i>/*</code> | Provides access to the S3 bucket containing the packages for Distributor, a capability of AWS Systems Manager, that are owned by AWS.                                                                                                                                                                                                  |

| S3 bucket ARN                                                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>arn:aws:s3:::patch-baseline-snapshot-<i>region</i>/*</code> | <p>Provides access to the S3 bucket containing patch baseline snapshots. This is required if you use any of the following SSM documents:</p> <ul style="list-style-type: none"> <li>• AWS-RunPatchBaseline</li> <li>• AWS-RunPatchBaselineAssociation</li> <li>• AWS-RunPatchBaselineWithHooks</li> <li>• AWS-ApplyPatchBaseline (a legacy SSM document)</li> </ul> <p><b>Note</b><br/> In the Middle East (Bahrain) Region (me-south-1) only, this S3 bucket uses a different naming convention. For this AWS Region only, use the following bucket instead.</p> <ul style="list-style-type: none"> <li>• <code>patch-baseline-snapshot-me-south-1-uduvl7q8</code></li> </ul> <p>In the Africa (Cape Town) Region (af-south-1) only, this S3 bucket uses a different naming convention. For this AWS Region only, use the following bucket instead.</p> <ul style="list-style-type: none"> <li>• <code>patch-baseline-snapshot-af-south-1-tbxdb5b9</code></li> </ul> |

| S3 bucket ARN                                                                                                                                                                                                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>For Linux and Windows Server managed nodes:<br/> <code>arn:aws:s3:::aws-ssm-<i>region</i>/*</code></p> <p>For Amazon EC2 instances for macOS:<br/> <code>arn:aws:s3:::aws-patchmanager-macos-<i>region</i>/*</code></p> | <p>Provides access to the S3 bucket containing modules required for use with certain Systems Manager documents (SSM documents). For example:</p> <ul style="list-style-type: none"> <li><code>arn:aws:s3:::aws-ssm-us-east-2/*</code></li> <li><code>aws-patchmanager-macos-us-east-2/*</code></li> </ul> <p><b>Exceptions</b></p> <p>The S3 bucket names in a few AWS Regions use an extended naming convention, as shown by their ARNs. For these Regions, use the following ARNs instead:</p> <ul style="list-style-type: none"> <li><b>Middle East (Bahrain) Region (me-south-1):</b> <code>aws-patch-manager-me-south-1-a53fc9dce</code></li> <li><b>Africa (Cape Town) Region (af-south-1):</b> <code>aws-patch-manager-af-south-1-bdd5f65a9</code></li> <li><b>Europe (Milan) Region (eu-south-1):</b> <code>aws-patch-manager-eu-south-1-c52f3f594</code></li> <li><b>Asia Pacific (Osaka) Region (ap-northeast-3):</b> <code>aws-patch-manager-ap-northeast-3-67373598a</code></li> </ul> <p><b>SSM documents</b></p> <p>The following are some commonly used SSM documents stored in these buckets.</p> <p>In <code>arn:aws:s3:::aws-ssm-<i>region</i>/</code>:</p> <ul style="list-style-type: none"> <li><code>AWS-RunPatchBaseline</code></li> <li><code>AWS-RunPatchBaselineAssociation</code></li> <li><code>AWS-RunPatchBaselineWithHooks</code></li> <li><code>AWS-InstanceRebootWithHooks</code></li> <li><code>AWS-ConfigureWindowsUpdate</code></li> <li><code>AWS-FindWindowsUpdates</code></li> <li><code>AWS-PatchAsgInstance</code></li> <li><code>AWS-PatchInstanceWithRollback</code></li> <li><code>AWS-UpdateSSMAgent</code></li> <li><code>AWS-UpdateEC2Config</code></li> </ul> <p>In <code>arn:aws:s3:::aws-patchmanager-macos-<i>region</i>/</code>:</p> <ul style="list-style-type: none"> <li><code>AWS-RunPatchBaseline</code></li> <li><code>AWS-RunPatchBaselineAssociation</code></li> <li><code>AWS-RunPatchBaselineWithHooks</code></li> </ul> |

| S3 bucket ARN | Description                                                                                                                                          |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <ul style="list-style-type: none"><li>• AWS-InstanceRebootWithHooks</li><li>• AWS-PatchAsgInstance</li><li>• AWS-PatchInstanceWithRollback</li></ul> |

## 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). In most cases, you need to provide these permissions explicitly in an instance profile or service role only when using a VPC endpoint.

### 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 don't 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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": "*",
 "Action": "s3:GetObject",
 "Resource": [
 "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:::aws-ssm-document-attachments-us-east-2/*",
 "arn:aws:s3:::patch-baseline-snapshot-us-east-2/*",
 "arn:aws:s3:::aws-ssm-us-east-2/*",
 "arn:aws:s3:::aws-patchmanager-macos-us-east-2/*"
]
 }
]
}
```

## Troubleshooting SSM Agent

If you experience problems running operations on your managed nodes, there might be a problem with AWS Systems Manager Agent (SSM Agent). Use the following information to help you view SSM Agent log files and troubleshoot the agent.

### Topics

- [SSM Agent is out of date \(p. 143\)](#)
- [View SSM Agent log files \(p. 143\)](#)
- [Agent log files don't rotate \(Windows\) \(p. 143\)](#)
- [Unable to connect to SSM endpoints \(p. 144\)](#)

## SSM Agent is out of date

An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

## View SSM Agent log files

SSM Agent logs information in the following files. The information in these files can also help you troubleshoot problems. For more information about SSM Agent log files, including how to turn on debug logging, see [Viewing SSM Agent logs \(p. 132\)](#).

### Note

If you choose to view these logs by using Windows File Explorer, be sure to allow 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 and macOS

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

For Linux managed nodes, you might find more information in the messages file written to the following directory: /var/log.

## Agent log files don't rotate (Windows)

If you specify date-based log file rotation in the seelog.xml file (on Windows Server managed nodes) and the logs don't rotate, specify the fullname=true parameter. Here is an example of a seelog.xml configuration file with the fullname=true parameter specified.

```
<seelog type="adaptive" mininterval="2000000" maxinterval="100000000" critmsgcount="500" minlevel="debug">
 <exceptions>
 <exception filepattern="test*" minlevel="error" />
 </exceptions>
 <outputs formatid="fmtinfo">
 <console formatid="fmtinfo" />
 <rollingfile type="date" datepattern="200601021504" maxrolls="4" filename="C:\ProgramData\Amazon\SSM\Logs\amazon-ssm-agent.log" fullname=true />
 <filter levels="error,critical" formatid="fmterror">
 <rollingfile type="date" datepattern="200601021504" maxrolls="4" filename="C:\ProgramData\Amazon\SSM\Logs\errors.log" fullname=true />
 </filter>
 </outputs>
 <formats>
 <format id="fmterror" format="%Date %Time %LEVEL [%FuncShort @ %File.%Line] %Msg%n" />
 </formats>
 </seelog>
```

```
<format id="fmtdebug" format="%Date %Time %LEVEL [%FuncShort @ %File.%Line] %Msg%n" />
<format id="fmtinfo" format="%Date %Time %LEVEL %Msg%n" />
</formats>
</seelog>
```

## Unable to connect to SSM endpoints

SSM Agent must be able to connect to the following endpoints:

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

### 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*.

SSM Agent won't work if it can't communicate with the preceding endpoints, even if you use AWS provided Amazon Machine Images (AMIs) such as Amazon Linux or Amazon Linux 2. Your network configuration must have open internet access or you must have custom virtual private cloud (VPC) endpoints configured. If you don't plan on creating a custom VPC endpoint, check your internet gateways or NAT gateways. For more information about how to manage VPC endpoints, see [Step 6: \(Optional\) Create a VPC endpoint \(p. 28\)](#).

# AWS Systems Manager Quick Setup

Use Quick Setup, a capability of AWS Systems Manager, to quickly configure frequently used Amazon Web Services services and features with recommended best practices. Quick Setup simplifies setting up services, including Systems Manager, by automating common or recommended tasks. These tasks include, for example, creating required AWS Identity and Access Management (IAM) instance profile roles and setting up operational best practices, such as periodic patch scans and inventory collection. To get started with Quick Setup, open the [Systems Manager console](#). In the navigation pane, choose **Quick Setup**.

## What are the benefits of Quick Setup?

Benefits of Quick Setup include the following:

- **Simplify service and feature configuration**

Quick Setup walks you through configuring operational best practices and automatically deploys those configurations. The Quick Setup dashboard displays a real-time view of your configuration deployment status.

- **Deploy configurations automatically across multiple accounts**

You can use Quick Setup in an individual AWS account or across multiple AWS accounts and AWS Regions by integrating with AWS Organizations. Using Quick Setup across multiple accounts helps to ensure that your organization maintains consistent configurations.

- **Eliminate configuration drift**

Configuration drift occurs whenever a user makes any change to a service or feature that conflicts with the selections made through Quick Setup. Quick Setup periodically checks for configuration drift and attempts to remediate it.

## Who should use Quick Setup?

Quick Setup is most beneficial for customers who already have some experience with the services and features they're setting up, and want to simplify their setup process. If you're unfamiliar with the AWS service you're configuring with Quick Setup, we recommend that you learn more about the service. Review the content in the relevant User Guide before you create a configuration with Quick Setup.

## Getting started with Quick Setup

To get started with Quick Setup, a capability of AWS Systems Manager, you must choose a home AWS Region and then onboard with Quick Setup. The home Region is where Quick Setup creates the AWS resources that are used to deploy your configurations. The home Region can't be changed after you select it.

### IAM roles and permissions

During onboarding, Quick Setup creates the following AWS Identity and Access Management (IAM) roles on your behalf:

- **AWS-QuickSetup-StackSet-Local-ExecutionRole** – Grants AWS CloudFormation permissions to use any template.
- **AWS-QuickSetup-StackSet-Local-AdministrationRole** – Grants permissions to AWS CloudFormation to assume **AWS-QuickSetup-StackSet-Local-ExecutionRole**.

If you're onboarding a management account, Quick Setup also creates the following roles on your behalf:

- **AWS-QuickSetup-SSM-RoleForEnablingExplorer** – Grants permissions to the **AWS-EnableExplorer** automation runbook. The **AWS-EnableExplorer** runbook configures Explorer, a capability of Systems Manager, to display information for multiple AWS accounts and AWS Regions.
- **AWSServiceRoleForAmazonSSM** – A service-linked role that grants access to AWS resources managed and used by Systems Manager.
- **AWSServiceRoleForAmazonSSM\_AccountDiscovery** – A service-linked role that grants permissions to Systems Manager to call AWS services to discover AWS account information when synchronizing data. For more information, see [About the AWSServiceRoleForAmazonSSM\\_AccountDiscovery role \(p. 163\)](#).

When onboarding a management account, Quick Setup enables trusted access between AWS Organizations and CloudFormation to deploy Quick Setup configurations across your organization. To enable trusted access, your management account must have administrator permissions. After onboarding, you no longer need administrator permissions. For more information, see [Enable trusted access with Organizations](#).

**Note**

Quick Setup uses AWS CloudFormation stack sets to deploy changes. Stack sets aren't deployed to your organization's management account. For more information, see [Considerations when creating a stack set with service-managed permissions](#).

If your IAM user, group, or role has access to the API operations listed in the following table, you can use all features of Quick Setup. There are two tabs of API operations, the first tab is permissions required by all accounts and the second tab contains the additional permissions you need for the management account of your organization.

Non-management account

```
"iam:CreateRole",
"iam:AttachRolePolicy",
"iam:PutPolicy",
"iam:GetRole",
"iam>ListRoles",
"iam:PassRole",
:ssm>ListAssociations",
:ssm>ListDocuments",
:ssm:GetDocument",
:ssm:DescribeAssociation",
:ssm:DescribeAutomationExecutions",
:cloudformation:DescribeStackSet",
:cloudformation:DescribeStackInstance",
:cloudformation:DescribeStacks",
:cloudformation:DescribeStackResources",
:cloudformation>ListStackSetOperations",
:cloudformation>ListStackSets",
:cloudformation>ListStacks",
:cloudformation>ListStackInstances",
:cloudformation>ListStackSetOperationResults",
:cloudformation:TagResource",
:cloudformation>DeleteStackSet",
:cloudformation:UpdateStackSet",
:cloudformation>CreateStackSet",
```

```
"cloudformation>DeleteStackInstances",
"cloudformation>CreateStackInstances"
```

#### Management account

```
"ssm:createResourceDataSync",
:ssm:listResourceDataSync",
:ssm:getOpsSummary",
:ssm:createAssociation",
:ssm:createDocument",
:ssm:startAssociationsOnce",
:ssm:startAutomationExecution",
:ssm:updateAssociation",
:ssm:listAssociations",
:ssm:listDocuments",
:ssm:getDocument",
:ssm:describeAssociation",
:ssm:describeAutomationExecutions",
:organizations>ListRoots",
:organizations>DescribeOrganization",
:organizations>ListOrganizationalUnitsForParent"
:organizations>EnableAWSAccess",
"cloudformation:describe"
```

## Configure the home AWS Region

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Quick Setup**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Quick Setup** in the navigation pane.

3. Choose a **home Region**.
4. Choose **Get started**.

To start using Quick Setup, choose a service or feature in the list of available configuration types. A *configuration type* in Quick Setup is specific to an AWS service or feature. When you choose a configuration type, you choose the options that you want to configure for that service or feature. By default, configuration types help you set up the service or feature to use recommended best practices.

After setting up a configuration, you can view details about it and its deployment status across organizational units (OUs) and Regions. You can also view State Manager association status for the configuration. State Manager is a capability of AWS Systems Manager. In the **Configuration details** pane, you can view a summary of the Quick Setup configuration. This summary includes details from all accounts and any detected configuration drift.

## Availability of Quick Setup in AWS Regions

Quick Setup for organizations is available in the following AWS Regions, unless otherwise stated:

- US East (Ohio)
- US East (N. Virginia)

- US West (N. California)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- South America (São Paulo)

Quick Setup for individual AWS accounts is available in all AWS Regions where Systems Manager is supported. For a list of supported Regions, see the **Region** column in [Systems Manager service endpoints](#) in the [Amazon Web Services General Reference](#).

## Using Quick Setup

Quick Setup, a capability of AWS Systems Manager, displays the results of each configuration in the **Configurations** table on the Quick Setup home page. From this page, you can **View details** of each configuration, delete configurations from the **Actions** drop down, or **Create** configurations. The **Configurations** table contains the following information:

- **Configuration type** – The configuration type chosen when creating the configuration.
- **Deployment type** – The AWS account and AWS Regions that you deployed the configuration to.
- **Organizational units** – Displays the organizational units (OUs) that the configuration is deployed to if you chose a **Custom** set of targets. Organizational units and custom targets are only available to the management account of your organization.
- **Regions** – The Regions that the configuration is deployed to if you chose a **Custom** set of targets or targets within your **Current account**.
- **Deployment status** – The deployment status indicates if AWS CloudFormation successfully deployed the target or stack instance. The target and stack instances contain the configuration options that you chose during configuration creation.
- **Association status** – The association status is the state of all associations created by the configuration that you created. The associations for all targets must run successfully; otherwise, the status is **Failed**.

Quick Setup creates and runs a State Manager association for each configuration target. State Manager is a capability of AWS Systems Manager.

## Configuration details

The **Configuration details** page displays information about the deployment of the configuration and its related associations. From this page, you can edit configuration options, update targets, or delete the configuration. You can also view the details of each configuration deployment to get more information about the associations.

Each configuration type has some of the following status graphs:

### Configuration deployment status

Displays the number of deployments that have succeeded, failed, or are running or pending. Deployments occur in the specified target accounts and Regions that contain nodes affected by the configuration.

### Configuration association status

Displays the number of State Manager associations that have succeeded, failed, or are pending. Quick Setup creates an association in each deployment for the configuration options selected.

The **Configuration details** table displays information about the deployment of your configuration. You can view more details about each deployment by selecting the deployment and then choosing **View details**. The details page of each deployment displays the associations deployed to the nodes in that deployment.

## Editing and deleting your configuration

You can edit configuration options of a configuration from the **Configuration details** page by choosing **Actions** and then **Edit configuration options**. When you add new options to the configuration, Quick Setup runs your deployments and creates new associations. When you remove options from a configuration, Quick Setup runs your deployments and removes any related associations.

#### Note

You can edit Quick Setup configurations for your account at anytime. To edit an **Organization** configuration, the **Configuration status** must be **Success** or **Failed**.

You can also update the targets included in your configurations by choosing **Actions** and **Add OUs**, **Add Regions**, **Remove OUs**, or **Remove Regions**. If your account isn't configured as the management account or you created the configuration for only the current account, you can't update the target organizational units (OUs). Removing a Region or OU removes the associations from those Regions or OUs.

You can delete a configuration from Quick Setup by choosing the configuration, then **Actions**, and then **Delete configuration**. Or, you can delete the configuration from the **Configuration details** page under the **Actions** dropdown and then **Delete configuration**. Quick Setup then prompts you to **Remove all OUs and Regions** which might take some time to complete. Deleting a configuration also deletes all related associations. This two-step deletion process removes all deployed resources from all accounts and Regions and then deletes the configuration.

## Configuration compliance

You can view whether your instances are compliant with the associations created by your configurations in either Explorer or Compliance, which are both capabilities of AWS Systems Manager. To learn more about compliance, see [Working with Compliance \(p. 839\)](#). To learn more about viewing compliance in Explorer, see [AWS Systems Manager Explorer \(p. 157\)](#).

## Troubleshooting Quick Setup results

### Failed deployment

A deployment fails if the CloudFormation stack set failed during creation. Use the following steps to investigate a deployment failure.

1. Navigate to the [AWS CloudFormation console](#).
2. Choose the stack created by your Quick Setup configuration. The **Stack name** includes **QuickSetup** followed by the type of configuration you chose, such as **SSMHostMgmt**.

### Note

CloudFormation sometimes deletes failed stack deployments. If the stack isn't available in the **Stacks** table, choose **Deleted** from the filter list.

3. View the **Status** and **Status reason**. For more information about stack statuses, see [Stack status codes](#) in the *AWS CloudFormation User Guide*.
4. To understand the exact step that failed, view the **Events** tab and review each event's **Status**.
5. Review [Troubleshooting](#) in the *AWS CloudFormation User Guide*.
6. If you are unable to resolve the deployment failure using the CloudFormation troubleshooting steps, delete the configuration and reconfigure it.

### Failed association

The **Configuration details** table on the **Configuration details** page of your configuration shows a **Configuration status** of **Failed** if any of the associations failed during set up. Use the following steps to troubleshoot a failed association.

1. In the **Configuration details** table, choose the failed configuration and then choose **View Details**.
2. Copy the **Association name**.
3. Navigate to **State Manager** and paste the association name into the search field.
4. Choose the association and choose the **Execution history** tab.
5. Under **Execution ID**, choose the association execution that failed.
6. The **Association execution targets** page lists all of the nodes where the association ran. Choose the **Output** button for an execution that failed to run.
7. 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 solve the problem, then you can try to recreate the association. To recreate the association, first delete the failing association in State Manager. After deleting the association, edit the configuration and choose the option you deleted and choose **Update**.

### Note

To investigate **Failed** associations for an **Organization** configuration, you must sign in to the account with the failed association and use the following failed association procedure, previously described. The **Association ID** isn't a hyperlink to the target account when viewing results from the management account.

### Drift status

When viewing a configuration's details page, you can view the drift status of each deployment. Configuration drift occurs whenever a user makes any change to a service or feature that conflicts with the selections made through Quick Setup. If an association has changed after the initial configuration, the table displays a warning icon that indicates the number of items that have drifted. You can determine what caused the drift by hovering over the icon.

When an association is deleted in State Manager, the related deployments display a drift warning. To fix this, edit the configuration and choose the option that was removed when the association was deleted. Choose **Update** and wait for the deployment to complete.

## Quick Setup Host Management

Use Quick Setup, a capability of AWS Systems Manager, to quickly configure required security roles and commonly used Systems Manager capabilities on your Amazon Elastic Compute Cloud (Amazon EC2)

instances. You can use Quick Setup in an individual account or across multiple accounts and AWS Regions by integrating with AWS Organizations. These capabilities help you manage and monitor the health of your instances while providing the minimum required permissions to get started.

If you're unfamiliar with Systems Manager services and features, we recommend that you review the *AWS Systems Manager User Guide* before creating a configuration with Quick Setup. For more information about Systems Manager, see [What is AWS Systems Manager? \(p. 1\)](#).

**Note**

You can't create multiple Quick Setup Host Management configurations that target the same AWS Region.

To set up host management, perform the following tasks in the AWS Systems Manager Quick Setup console.

### To set up host management with Quick Setup

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Quick Setup**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Quick Setup** in the navigation pane.

3. Choose **Create**.
4. Choose **Host management**, and then choose **Next**.
5. In the **Configuration options** section, choose the options that you want to allow for your configuration.
  - **Update Systems Manager (SSM) Agent every two weeks** – Enables Systems Manager to check 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 managed node to the latest released version.

We encourage you to choose this option to ensure that your nodes 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 \(p. 68\)](#).

- **Collect inventory from your instances every 30 minutes** – Enables Quick Setup to configure collection of the following types of metadata:
  - **AWS components** – EC2 driver, agents, versions, and more.
  - **Applications** – Application names, publishers, versions, and more.
  - **Node 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 nodes only).
  - **Windows roles** – Name, display name, path, feature type, installed state, and more (Windows Server nodes only).
  - **Windows updates** – Hotfix ID, installed by, installed date, and more (Windows Server nodes only).

For more information about Inventory, a capability of AWS Systems Manager, see [AWS Systems Manager Inventory \(p. 848\)](#).

**Note**

The **Inventory collection** option can take up to 10 minutes to complete, even if you only selected a few nodes.

- **Scan Instances for missing patches daily** – Enables Patch Manager, a capability of Systems Manager, to scan your nodes daily and generate a report in the **Compliance** page. The report shows how many nodes are patch-compliant according to the *default patch baseline*. The report includes a list of each node and its compliance status.

For more information about patching operations and patch baselines, see [AWS Systems Manager Patch Manager \(p. 1093\)](#). To view compliance information, see the Systems Manager **Compliance** page.

- **Install and configure the CloudWatch agent** – Installs the basic configuration of the unified CloudWatch agent on your Amazon EC2 instances. The agent collects metrics and log files from your instances for Amazon CloudWatch. This information is consolidated so you can quickly determine the health of your instances. For more information about the CloudWatch agent basic configuration, see [CloudWatch agent predefined metric sets](#). There might be added cost. For more information, see [Amazon CloudWatch pricing](#).
  - **Update the CloudWatch agent once every 30 days** – Enables Systems Manager to check every 30 days for a new version of the CloudWatch agent. If there is a new version, Systems Manager updates the agent on your instance. 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.
6. In the **Targets** section, choose whether to set up host management for your **Entire organization**, **Custom** organizational units (OUs), or the **Current account** you're signed in to:
- **Entire organization** – In the **Instance profile options** section, choose whether you want to add the required IAM policies to the existing instance profiles attached to your instances, or to allow Quick Setup to create the IAM policies and instance profiles with the permissions needed for the configuration you choose.
  - **Custom** – In the **Target OUs** section, select the OUs where you want to set up host management. Next, in the **Target Regions** section, select the Regions where you want to set up host management. Then, in the **Instance profile options** section, choose whether you want to add the required IAM policies to the existing instance profiles attached to your instances, or to allow Quick Setup to create the IAM policies and instance profiles with the permissions needed for the configuration you choose.
  - **Current account** – Select **Current Region** or **Choose Regions**. Next, select how you want to target instances. Then, if you selected **Current Region**, continue to step 7. If you selected **Choose Regions** choose the **Target Regions** where you want to set up host management and then continue to step 7.
7. Choose **Create**.

## AWS Config recording

With Quick Setup, a capability of AWS Systems Manager, you can quickly create a configuration recorder powered by AWS Config. Use the configuration recorder to detect changes in your resource configurations and capture the changes as configuration items. If you're unfamiliar with AWS Config, we recommend learning more about the service by reviewing the content in the [AWS Config Developer Guide](#) before creating a configuration with Quick Setup. For more information about AWS Config, see [What is AWS Config?](#) in the [AWS Config Developer Guide](#).

By default, the configuration recorder records all supported resources in the AWS Region where AWS Config is running. You can customize the configuration so that only the resource types you specify are recorded. For more information, see [Selecting which resources AWS Config records](#) in the [AWS Config Developer Guide](#).

You're charged service usage fees when AWS Config starts recording configurations. For pricing information, see [AWS Config pricing](#).

**Note**

Deleting the Quick Setup **Config recording** configuration type doesn't stop the configuration recorder. Changes continue to be recorded, and service usage fees apply until you stop the configuration recorder. To learn more about managing the configuration recorder, see [Managing the Configuration Recorder](#) in the *AWS Config Developer Guide*.

To set up AWS Config recording, perform the following tasks in the AWS Systems Manager console.

**To set up AWS Config recording with Quick Setup**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Quick Setup**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Quick Setup** in the navigation pane.

3. Choose **Create**.
4. Choose **Config recording**, and then choose **Next**.
5. In the **Configuration options** section, choose the AWS resource types you want to record and whether you want to include global resources.
6. Choose the Region you want AWS Config to use when recording changes made to global resources. The value you specify determines where API calls originate from when AWS Config gathers information about global resources in your configuration. The Region you choose must be a Region you specify later in **Targets**.
7. Create a new Amazon Simple Storage Service (Amazon S3) bucket, or choose an existing bucket you want to send configuration snapshots to.
8. Choose the notification option you prefer. AWS Config uses Amazon Simple Notification Service (Amazon SNS) to notify you about important AWS Config events related to your resources. If you choose the **Use existing SNS topics** option, you must provide the AWS account ID and name of the existing Amazon SNS topic in that account you want to use. If you target multiple AWS Regions, the topic names must be identical in each Region.
9. In the **Schedule** section, choose how frequently you want Quick Setup to remediate changes made to resources that differ from your configuration. The **Default** option runs once. If you don't want Quick Setup to remediate changes made to resources that differ from your configuration, choose **Disable remediation** under **Custom**.
10. In the **Targets** section, choose whether to allow AWS Config recording for your entire organization, some of your organizational units (OUs), or the account you're logged in to.

If you choose **Entire organization**, continue to step 12.

If you choose **Custom**, continue to step 11.

11. In the **Target OUs** section, select the check boxes of the OUs and Regions where you want to use AWS Config recording.
12. Choose **Create**.

## Deploy AWS Config conformance packs

A conformance pack is a collection of AWS Config rules and remediation actions. With Quick Setup, you can deploy a conformance pack as a single entity in an account and an AWS Region or across an organization in AWS Organizations. This helps you manage configuration compliance of your AWS

resources at scale, from policy definition to auditing and aggregated reporting, by using a common framework and packaging model.

To deploy conformance packs, perform the following tasks in the AWS Systems Manager Quick Setup console.

**Note**

You must enable AWS Config recording before deploying this configuration. For more information, see [Conformance packs](#) in the *AWS Config Developer Guide*.

### To deploy conformance packs with Quick Setup

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Quick Setup**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Quick Setup** in the navigation pane.

3. Choose **Create**.
4. Choose **Conformance packs**, and then choose **Next**.
5. In the **Configuration options** section, choose the conformance packs you want to deploy.
6. In the **Targets** section, choose whether to deploy conformance packs to your entire organization, some AWS Regions, or the account you're currently logged in to.

If you choose **Entire organization**, continue to step 8.

If you choose **Custom**, continue to step 7.

7. In the **Target Regions** section, select the check boxes of the Regions you want to deploy conformance packs to.
8. Choose **Create**.

## Configure DevOps Guru with Quick Setup

You can quickly configure DevOps Guru options by using Quick Setup. Amazon DevOps Guru is a machine learning (ML) powered service that makes it easy to improve an application's operational performance and availability. DevOps Guru detects behaviors that are different from normal operating patterns so you can identify operational issues long before they impact your customers. DevOps Guru automatically ingests operational data from your AWS applications and provides a single dashboard to visualize issues in your operational data. You can get started with DevOps Guru to improve application availability and reliability with no manual setup or machine learning expertise.

Configuring DevOps Guru with Quick Setup is available in the following AWS Regions:

- US East (N. Virginia)
- US East (Ohio)
- US West (Oregon)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (Stockholm)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)

For pricing information, see [Amazon DevOps Guru pricing](#).

To set up DevOps Guru, perform the following tasks in the AWS Systems Manager Quick Setup console.

### To set up DevOps Guru with Quick Setup

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Quick Setup**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Quick Setup** in the navigation pane.

3. Choose **Create**.
4. Choose **DevOps Guru**, and then choose **Next**.
5. In the **Configuration options** section, choose the AWS resource types you want to analyze and your notification preferences.

If you don't select the **Analyze all AWS resources in all the accounts in my organization** option, you can choose AWS resources to analyze later in the DevOps Guru console. DevOps Guru analyzes different AWS resource types (such as Amazon Simple Storage Service (Amazon S3) buckets and Amazon Elastic Compute Cloud (Amazon EC2) instances), which are categorized into two pricing groups. You pay for the AWS resource hours analyzed, for each active resource. A resource is only active if it produces metrics, events, or log entries within an hour. The rate you're charged for a specific AWS resource type depends on the price group.

If you select the **Enable SNS notifications** option, an Amazon Simple Notification Service (Amazon SNS) topic is created in each AWS account in the organizational units (OUs) you target with your configuration. DevOps Guru uses the topic to notify you about important DevOps Guru events, such as the creation of a new insight. If you don't enable this option, you can add a topic later in the DevOps Guru console.

If you select the **Enable AWS Systems Manager OpsItems** option, operational work items (OpsItems) will be created for related Amazon EventBridge events and Amazon CloudWatch alarms.

6. In the **Schedule** section, choose how frequently you want Quick Setup to remediate changes made to resources that differ from your configuration. The **Default** option runs once. If you don't want Quick Setup to remediate changes made to resources that differ from your configuration, choose **Disabled** under **Custom**.
7. In the **Targets** section, choose whether to allow DevOps Guru to analyze resources in some of your organizational units (OUs), or the account you're currently logged in to.

If you choose **Custom**, continue to step 8.

If you choose **Current account**, continue to step 9.

8. In the **Target OUs** and **Target Regions** sections, select the check boxes of the OUs and Regions where you want to use DevOps Guru.
9. Choose the Regions where you want to use DevOps Guru in the current account.
10. Choose **Create**.

## Deploy Distributor packages with Quick Setup

Distributor is a capability of AWS Systems Manager. A Distributor package is a collection of installable software or assets that can be deployed as a single entity. With Quick Setup, you can deploy a Distributor package in an AWS account and an AWS Region or across an organization in AWS Organizations.

Currently, only the Amazon Elastic File System (Amazon EFS) utilities package can be deployed with Quick Setup. For more information about Distributor, see [AWS Systems Manager Distributor \(p. 1252\)](#).

To deploy Distributor packages, perform the following tasks in the AWS Systems Manager Quick Setup console.

### To deploy Distributor packages with Quick Setup

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Quick Setup**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Quick Setup** in the navigation pane.

3. Choose **Create**.
4. Choose **Distributor**, and then choose **Next**.
5. In the **Configuration options** section, choose the package you want to deploy.
6. In the **Targets** section, choose whether to deploy the package to your entire organization, some of your organizational units (OUs), or the account you're currently logged in to.

If you choose **Entire organization**, continue to step 8.

If you choose **Custom**, continue to step 7.

7. In the **Target OUs** section, select the check boxes of the OUs and Regions you want to deploy the package to.
8. Choose **Create**.

# Operations Management

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

## Topics

- [AWS Systems Manager Incident Manager \(p. 157\)](#)
- [AWS Systems Manager Explorer \(p. 157\)](#)
- [AWS Systems Manager OpsCenter \(p. 180\)](#)
- [Amazon CloudWatch dashboards hosted by Systems Manager \(p. 229\)](#)

## AWS Systems Manager Incident Manager

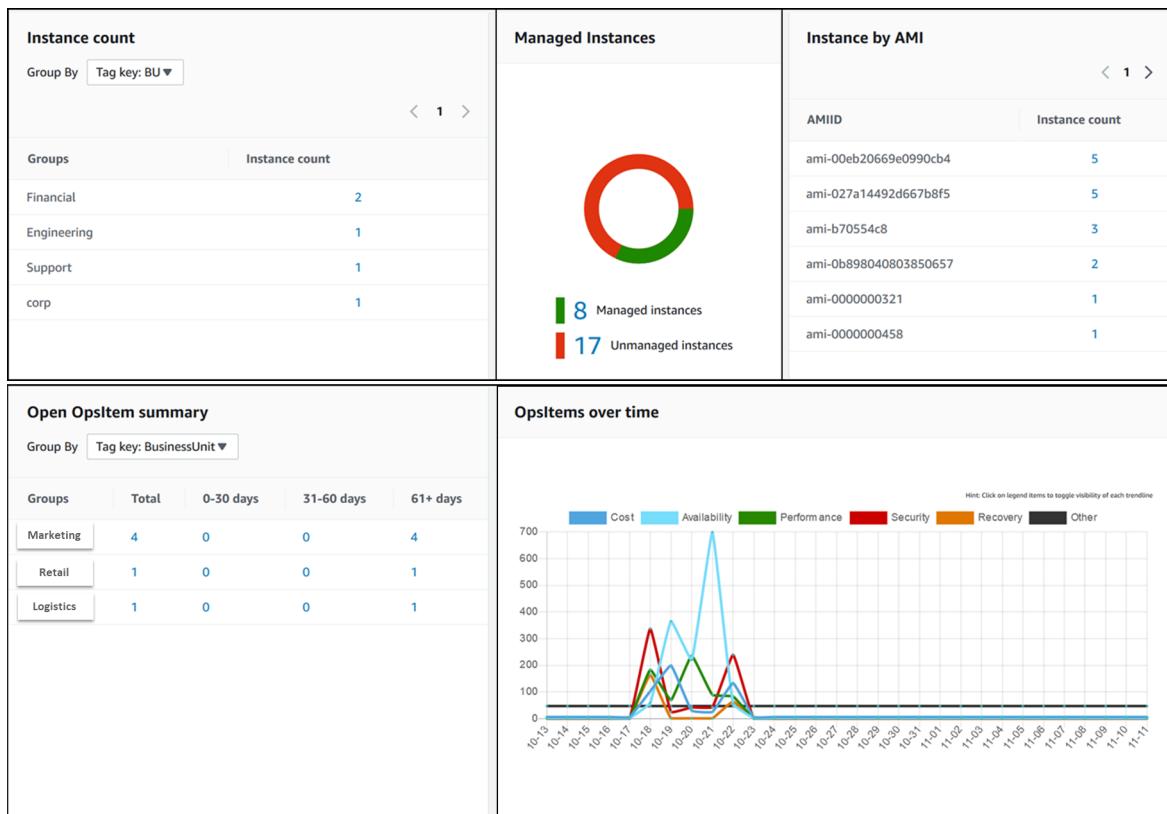
Use Incident Manager, a capability of AWS Systems Manager, to manage incidents occurring in your AWS hosted applications. Incident Manager combines user engagements, escalation, runbooks, response plans, chat channels, and post-incident analysis to help your team triage incidents faster and return your applications to normal. To learn more about Incident Manager, see the [Incident Manager User Guide](#).

## 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 AWS Regions. In Explorer, OpsData includes metadata about your Amazon Elastic Compute Cloud (Amazon EC2) instances, Systems Manager Patch Manager patch compliance details, and Systems Manager State Manager association compliance details. OpsData also includes information from supporting AWS services like AWS Trusted Advisor, AWS Compute Optimizer, and information about your AWS Support cases.

To raise operational awareness, Explorer also displays 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. To get started with Explorer, open the [Systems Manager console](#). In the navigation pane, choose **Explorer**.

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, Patch Manager, State Manager, and supporting AWS services like AWS Trusted Advisor, AWS Compute Optimizer, and AWS Support. 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**, **Instance by AMI**, **Non-compliant instances for patching**, **Non-compliant associations**, and **Support Center cases**.
  - 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 might 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 (Amazon EBS) 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, AWS 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 allow 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 AWS Region display:** Explorer includes the following display modes for OpsData and OpsItems in AWS accounts and AWS Regions:
  - *Single-account/single-Region:* This is the default view. This mode allows users to view data and OpsItems from their own account and the current Region.
  - *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.
  - *Multiple-account/multiple-Region:* This mode requires that your organization or company use [AWS Organizations](#) with **All features** turned on. After you configure AWS Organizations in your computing environment, you can aggregate all account data in a management 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 is complete.

## How does Explorer relate to OpsCenter?

[Systems Manager OpsCenter \(p. 180\)](#) 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 haven't 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 \(p. 160\)](#).

## 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 nodes, total number of managed and unmanaged nodes, and a count of nodes 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 nodes 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.

- **AWS Support Center cases**

Data displayed in Explorer includes: case ID, severity, status, created time, subject, service, and category.

- **AWS Config**

Data displayed in Explorer includes: overall summary of compliant and non-compliant AWS Config rules, the number of compliant and non-compliant resources, and specific details about each (when you drill down into a non-compliant rule or resource).

- **AWS Security Hub**

Data displayed in Explorer includes: overall summary of Security Hub findings, the number of each finding grouped by severity, and specific details about finding.

**Note**

To view AWS Trusted Advisor and AWS Support Center cases in Explorer, you must have either an Enterprise or Business account set up with AWS Support.

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 \(p. 161\)](#).

## Is there a charge to use Explorer?

Yes. When you turn on 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](#).

**Topics**

- [Getting started with Systems Manager Explorer and OpsCenter \(p. 160\)](#)
- [Using Systems Manager Explorer \(p. 171\)](#)
- [Exporting OpsData from Systems Manager Explorer \(p. 177\)](#)
- [Troubleshooting Systems Manager Explorer \(p. 179\)](#)

## Getting started with Systems Manager Explorer and OpsCenter

AWS 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 haven't set up OpsCenter, then you can use Integrated Setup to get started with both capabilities.

**Note**

Integrated Setup is only available in the Systems Manager console. You can't set up Explorer or OpsCenter programmatically.

Integrated Setup performs the following tasks:

- [Configures roles and permissions \(p. 162\)](#): Integrated Setup creates an AWS Identity and Access Management (IAM) role that allows Amazon EventBridge 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.
- [Allows default rules for OpsItem creation \(p. 166\)](#): Integrated Setup creates default rules in EventBridge. 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.
- [Allows OpsData sources \(p. 166\)](#): Integrated Setup allows data sources that populate Explorer widgets.
- [Allows you to specify reporting tag keys \(p. 167\)](#): Integrated Setup allows 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 \(p. 167\)](#). Explorer and OpsCenter automatically synchronize OpsData and OpsItems for the AWS account and AWS 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

AWS Systems Manager Explorer and AWS Systems Manager 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 allow Explorer and OpsCenter to collect information from, or interact with, other AWS services and Systems Manager capabilities.

Task	Information
Verify permissions in Systems Manager Automation	Explorer and OpsCenter allow you to remediate issues with AWS resources by using Systems Manager Automation runbooks. To use this remediation capability, you must have permission to run Systems Manager Automation runbooks. For more information, see <a href="#">Setting up Automation (p. 401)</a> .
Set up and configure Systems Manager Patch Manager	Explorer includes a widget that provides information about patch compliance. To view this data in Explorer, you must configure patching. For more information, see <a href="#">AWS Systems Manager Patch Manager (p. 1093)</a> .
Set up and configure Systems Manager State Manager	Explorer includes a widget that provides information about Systems Manager State Manager association compliance. To view this data in Explorer, you must configure State Manager. For more information, see <a href="#">AWS Systems Manager State Manager (p. 1034)</a> .

Task	Information
Turn on AWS Config Configuration Recorder	<p>Explorer uses data provided by AWS Config configuration recorder to populate widgets with information about your EC2 instances. To view this data in Explorer, turn on AWS Config configuration recorder. For more information, see <a href="#">Managing the Configuration Recorder</a>.</p> <p><b>Note</b> After you allow configuration recorder, Systems Manager can take up to six hours to display data in Explorer widgets that display information about your EC2 instances.</p>
Turn on AWS Trusted Advisor	<p>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 <a href="#">AWS Support</a>.</p>
Turn on AWS Compute Optimizer	<p>Explorer uses data provided by Compute Optimizer to display details a count of <b>Under provisioned</b> and <b>Over provisioned</b> EC2 instances, optimization findings, on-demand pricing details, and recommendations for instance type and price. To view this data in Explorer, turn on Compute Optimizer. For more information, see <a href="#">Getting started with AWS Compute Optimizer</a>.</p>
Turn on AWS Security Hub	<p>Explorer uses data provided by Security Hub to populate widgets with information about your security findings. To view this data in Explorer, turn on Security Hub integration. For more information, see <a href="#">What is AWS Security Hub</a>.</p>

## Configuring roles and permissions for Systems Manager Explorer

Integrated Setup automatically creates and configures AWS Identity and Access Management (IAM) roles for AWS Systems Manager Explorer and AWS Systems Manager 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 \(p. 162\)](#)
- [Configuring permissions for Systems Manager OpsCenter \(p. 163\)](#)

### 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`: Allows CloudWatch Events and EventBridge to create OpsItems in response to common events.
- `AWSServiceRoleForAmazonSSM_AccountDiscovery`: Allows 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 \(p. 163\)](#).
- `AmazonSSMExplorerExport`: Allows 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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "organizations:DescribeAccount",
 "organizations:DescribeOrganization",
 "organizations>ListAccounts",
 "organizations>ListAWSAccessForOrganization",
 "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: AWSServiceRoleForAmazonSSM\\_AccountDiscovery \(p. 1412\)](#).

### [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 operations. 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
  - GetOpsItem
  - GetOpsSummary
  - ListResourceDataSync
  - UpdateOpsItem
  - UpdateResourceDataSync

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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetOpsItem",
 "ssm:GetOpsSummary",
 "ssm:DescribeOpsItems",
 "ssm:GetServiceSetting",
 "ssm>ListResourceDataSync"
],
 "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. Choose **Add inline policy**.
  6. Choose the **JSON** tab.
  7. Replace the default content with the following:

```

 "ssm>ListResourceDataSync",
 "ssm>UpdateResourceDataSync"

],
 "Resource": "*"
}
]
}

```

8. Choose **Review policy**.
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. 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 operation to view OpsItems that were previously tagged with Key=Department and Value=Finance. Users can't view any other OpsItems.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetOpsItem"
],
 "Resource": "*"
 },
 {
 "Condition": { "StringEquals": { "ssm:resourceTag/Department": "Finance" } }
 }
]
}
```

Here is an example that specifies API operations for viewing and updating OpsItems. This policy also specifies two sets of tag key-value pairs: Department-Finance and Project-Unity.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetOpsItem",
 "ssm:UpdateOpsItem"
],
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "ssm:resourceTag/Department": "Finance",
 "ssm:resourceTag/Project": "Unity"
 }
 }
 }
]
}
```

For information about adding tags to an OpsItem, see [Creating OpsItems manually \(p. 205\)](#).

## Turning on default rules

Integrated Setup automatically configures the following default rules in Amazon EventBridge. These rules create OpsItems in AWS Systems Manager OpsCenter. If you don't want EventBridge 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 EventBridge events. For more information, see [Configuring EventBridge to automatically create OpsItems for specific events \(p. 203\)](#). You can also turn off the default rules at any time on the [Settings](#) page.

### Important

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.

Rule	Category	Severity
<input type="checkbox"/> <b>CWE rules (11)</b>		
SSMOpsItems-Autoscaling-instance-launch-failure	Availability	2-High
SSMOpsItems-Autoscaling-instance-termination-failure	Availability	2-High
SSMOpsItems-EBS-snapshot-copy-failed	Availability	2-High
SSMOpsItems-EBS-snapshot-creation-failed	Availability	2-High
SSMOpsItems-EBS-volume-performance-issue	Performance	3-Medium
SSMOpsItems-EC2-issue	Availability	2-High
SSMOpsItems-EC2-scheduled-change	Availability	3-Medium
SSMOpsItems-RDS-issue	Availability	2-High
SSMOpsItems-RDS-scheduled-change	Availability	3-Medium
SSMOpsItems-SSM-maintenance-window-execution-failed	Availability	3-Medium
SSMOpsItems-SSM-maintenance-window-execution-timedout	Availability	2-High

## Configuring OpsData sources

Integrated Setup activates the following data sources that populate Explorer widgets.

- AWS Support Center (You must have either a Business or Enterprise Support plan to activate this source.)
- AWS Compute Optimizer (You must have either a Business or Enterprise Support plan to activate this source.)
- Systems Manager State Manager association compliance
- AWS Config Compliance
- Systems Manager OpsCenter
- Systems Manager Patch Manager patch compliance
- Amazon Elastic Compute Cloud (Amazon EC2)
- Systems Manager Inventory
- AWS Trusted Advisor (You must have either a Business or Enterprise Support plan to activate this source.)
- AWS Security Hub

## Specifying tag keys

When you set up AWS Systems Manager Explorer, you can specify up to five reporting tag keys. These tag keys should already exist on your AWS resources. These aren't 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

AWS Systems Manager uses an integrated setup experience to help you get started with AWS Systems Manager Explorer *and* AWS 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 AWS 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 \(p. 160\)](#).

### About resource data sync for Explorer

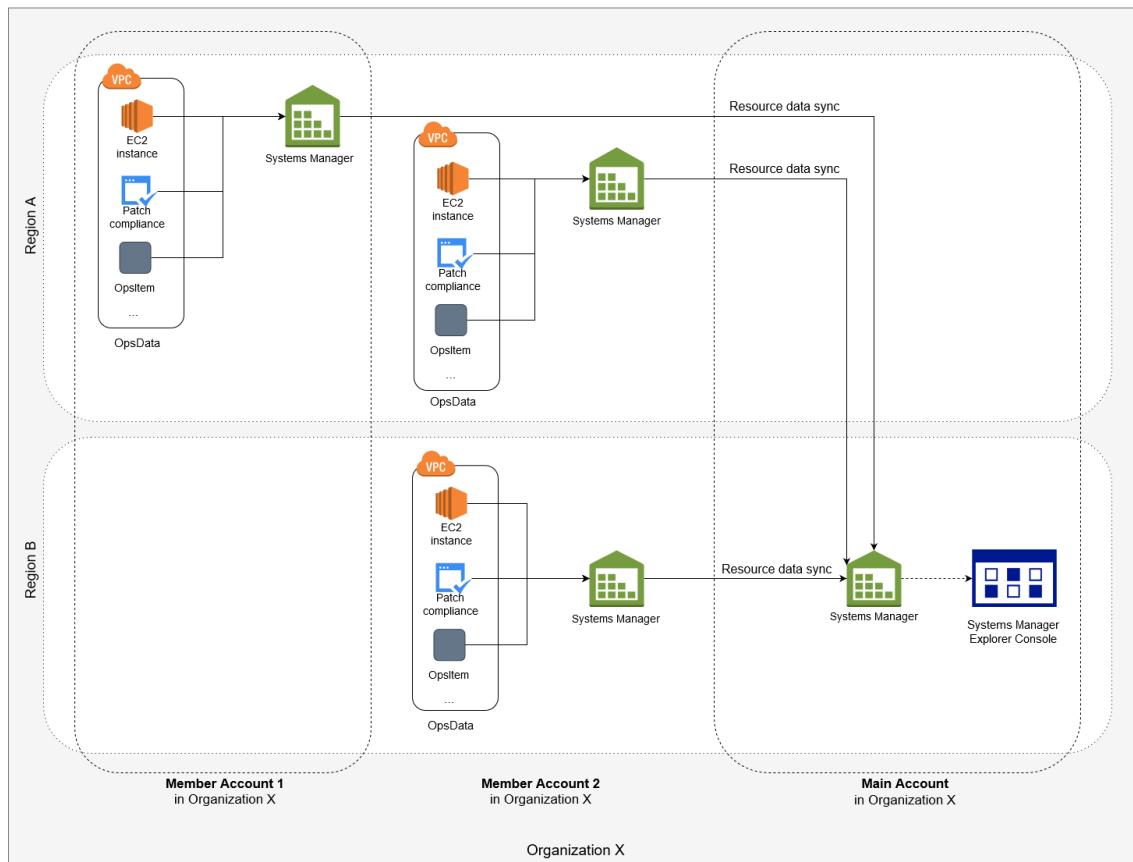
Resource data sync for Explorer offers two aggregation options:

- **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 management account before displaying it in Explorer. For more information, see [What is AWS Organizations?](#) in the [AWS Organizations User Guide](#).

**Warning**

If you configure Explorer to aggregate data from an organization in AWS Organizations, the system enables OpsData in all member accounts in the organization. Enabling OpsData sources in all member accounts increases the number of calls to OpsCenter APIs like [CreateOpsItem](#) and [GetOpsSummary](#). You are charged for calls to these API actions.

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 management account where it's then displayed in Explorer.



## About multiple account and Region resource data syncs

This section describes important details about multiple account and multiple Region resource data syncs that use AWS Organizations. Specifically, the information in this section applies if you choose one of the following options in the **Create resource data sync** page:

- Include all accounts from my AWS Organizations configuration
- Select organization units in AWS Organizations

If you don't plan to use one of these options, you can skip this section.

When you create a resource data sync, if you choose one of the AWS Organizations options, then Systems Manager automatically allows all OpsData sources in the selected Regions for all AWS accounts in your organization (or in the selected organizational units). For example, even if you haven't turned Explorer on in a Region, if you select an AWS Organizations option for your resource data sync, then Systems Manager automatically collects OpsData from that Region.

If you don't choose one of the AWS Organizations options for a resource data sync, then 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.

If you add a child account to your organization, Explorer automatically allows all OpsData sources for the account. If, at a later time, you remove the child account from your organization, Explorer continues to collect OpsData from the account.

If you update an existing resource data sync that uses one of the AWS Organizations options, the system prompts you to approve collection of all OpsData sources for all accounts and Regions affected by the change.

If you add a new service to your AWS account, and if Explorer collects OpsData for that service, Systems Manager automatically configures Explorer to collect that OpsData. For example, if your organization didn't use AWS Trusted Advisor when you previously created a resource data sync, but your organization signs up for this service, Explorer automatically updates your resource data syncs to collect this OpsData.

**Important**

Note the following important information about multiple account and Region resource data syncs:

- Deleting a resource data sync doesn't turn off an OpsData source in Explorer.
- To view OpsData and OpsItems from multiple accounts, you must have the AWS Organizations **All features** mode turned on and you must be signed into the AWS Organizations management account.

## Creating a resource data sync

Before you configure resource data sync for Explorer, note the following details.

- Explorer supports a maximum of five resource data syncs.
- 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 \(p. 175\)](#)
- 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 management 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 \(p. 170\)](#).

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 AWS Systems Manager 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 management account to administer resource data syncs in Explorer.

For more information about resource data sync, see [Setting up Systems Manager Explorer to display data from multiple accounts and Regions \(p. 167\)](#). For more information about AWS Organizations, see [What is AWS Organizations?](#) in the *AWS Organizations User Guide*.

### Topics

- [Before you begin \(p. 170\)](#)
- [Configure an Explorer delegated administrator \(p. 170\)](#)
- [Deregister an Explorer delegated administrator \(p. 171\)](#)

## 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](#) in the *AWS Organizations User Guide*.
- A delegated administrator can use all Explorer resource data sync API operations in the console or by using programmatic tools such as the SDK, the AWS Command Line Interface (AWS CLI), or AWS Tools for Windows PowerShell. Resource data sync API operations include the following: [CreateResourceDataSync](#), [DeleteResourceDataSync](#), [ListResourceDataSync](#), and [UpdateResourceDataSync](#).
- 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 operation.
- 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 management 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 management account.

2. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Create role** and **Enable access** 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 **Include all accounts from my AWS Organizations configuration** and **Select organization units in 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 management 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 management account.
2. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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. Enter the account ID and choose **Remove**.

The account no longer has access to the AWS Organizations resource data sync API operations. 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 AWS Systems Manager Explorer by changing the widget layout and by changing the data that displays in the dashboard.

## Contents

- [Editing default rules for OpsItems \(p. 171\)](#)
- [Editing Systems Manager Explorer data sources \(p. 172\)](#)
- [Customizing the display and using filters \(p. 173\)](#)
- [Deleting a Systems Manager Explorer resource data sync \(p. 175\)](#)
- [Receiving findings from AWS Security Hub in Explorer \(p. 175\)](#)

## Editing default rules for OpsItems

When you complete Integrated Setup, the system allows more than a dozen rules in Amazon EventBridge. These rules automatically create OpsItems in AWS Systems Manager OpsCenter. AWS 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**

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.

Rule	Category	Severity
SSMOpsItems-Autoscaling-instance-launch-failure	Availability	2-High
SSMOpsItems-Autoscaling-instance-termination-failure	Availability	2-High
SSMOpsItems-EBS-snapshot-copy-failed	Availability	2-High
SSMOpsItems-EBS-snapshot-creation-failed	Availability	2-High
SSMOpsItems-EBS-volume-performance-issue	Performance	3-Medium
SSMOpsItems-EC2-issue	Availability	2-High
SSMOpsItems-EC2-scheduled-change	Availability	3-Medium
SSMOpsItems-RDS-issue	Availability	2-High
SSMOpsItems-RDS-scheduled-change	Availability	3-Medium
SSMOpsItems-SSM-maintenance-window-execution-failed	Availability	3-Medium
SSMOpsItems-SSM-maintenance-window-execution-timedout	Availability	2-High

### To edit default rules for creating OpsItems

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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

AWS Systems Manager Explorer displays data from the following sources. You can edit Explorer settings to add or remove data sources:

- Amazon Elastic Compute Cloud (Amazon EC2)
- AWS Systems Manager OpsCenter
- AWS Systems Manager Patch Manager patch compliance
- AWS Systems Manager State Manager association compliance
- AWS Trusted Advisor
- AWS Compute Optimizer
- AWS Support Center cases
- AWS Config rule and resource compliance

- AWS Security Hub findings

**Note**

- To view AWS Support Center cases in Explorer, you must have either an Enterprise or Business account set up with AWS Support.
- You can't configure Explorer to stop displaying OpsCenter OpsItem data.

**Before you begin**

Verify that you set up and configured services that populate Explorer widgets with data. For more information, see [Setting up related services \(p. 161\)](#).

**To edit data sources**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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

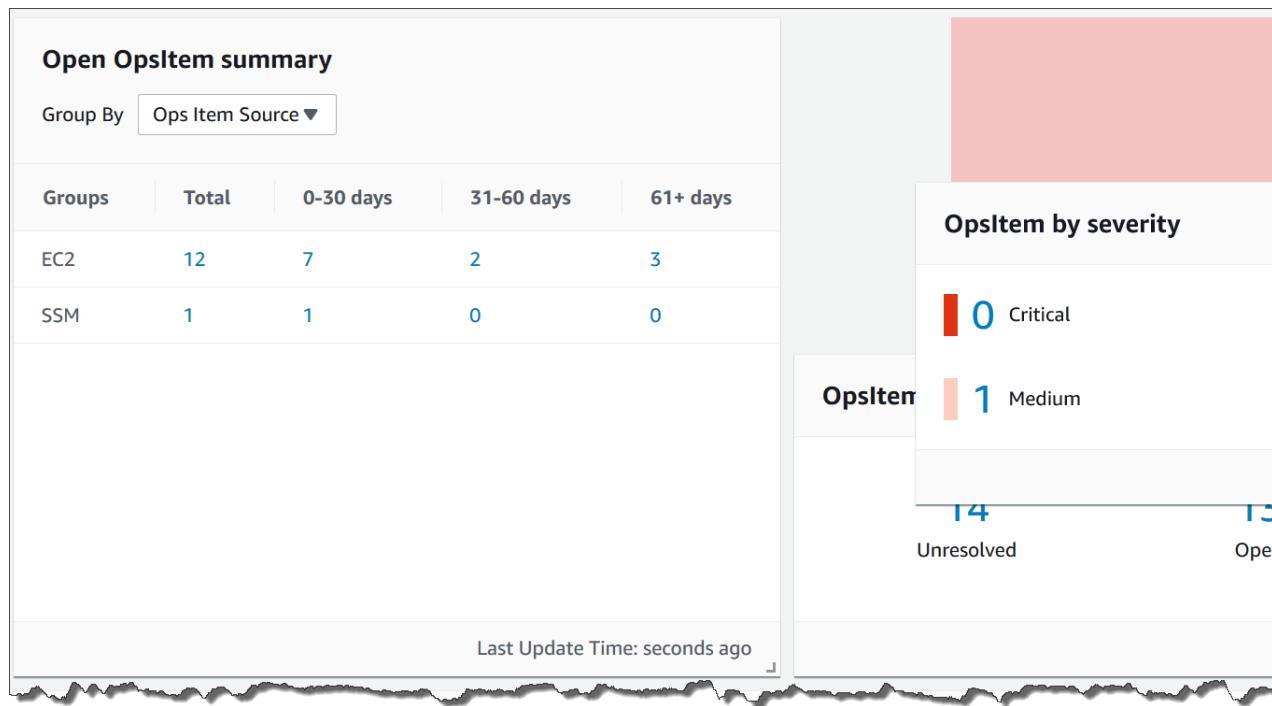
You can customize widget layout in AWS Systems Manager Explorer by using a drag-and-drop capability. You can also customize the OpsData and OpsItems displayed in Explorer by using filters, as described in this topic.

### Customizing widget layout

Use the following procedure to customize widget layout in Explorer.

**To customize widget layout**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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.



- Repeat this process for each widget that you want to reposition.

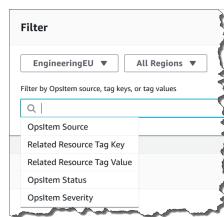
If you decide that you don't like the new layout, choose **Reset layout** to move all widgets back to their original location.

## Using filters to 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

- Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
- In the navigation pane, choose **Explorer**.
- In the **Filter** section, use the **Select a resource data sync** list to choose a sync.
- Use the **Regions** list to choose either a specific AWS Region or choose **All Regions**.
- Choose the Search bar, and then choose the criteria on which to filter the data.



- 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 AWS 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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**.

## Receiving findings from AWS Security Hub in Explorer

[AWS Security Hub](#) provides you with a comprehensive view of your security state in AWS and helps you to check your environment against security industry standards and best practices. Security Hub collects security data from across AWS accounts, services, and supported third-party partner products and helps you to analyze your security trends and identify the highest priority security issues.

Explorer, a capability of AWS Systems Manager, integration with Security Hub allows you to receive findings from Security Hub in Explorer. Security Hub findings provide security information that you can use in Explorer to aggregate and take action on your security, performance, and operational issues in Systems Manager. You can view a widget that provides summary of all Security Hub findings based on severity.

Turning on Security Hub integration accrues a cost. Explorer integrates with Systems Manager OpsCenter (OpsCenter) to provide Security Hub findings. There is a cost to use Explorer with OpsCenter. For more information, see [Receiving findings from AWS Security Hub in OpsCenter \(p. 227\)](#) and [AWS Systems Manager Pricing](#).

### How Explorer receives findings from Security Hub

In Security Hub, security issues are tracked as findings. Some findings come from issues that are detected by other AWS services or by third-party partners. Security Hub also has a set of rules that it uses to detect security issues and generate findings.

AWS Systems Manager Explorer is one of the AWS services that receives findings from Security Hub.

### Mechanism for receiving the findings from Security Hub

To receive the findings from Security Hub, Explorer takes advantage of the Security Hub integration with Amazon EventBridge. Security Hub sends findings to EventBridge, which uses an event rule to send the findings to Explorer.

### Types of findings that Explorer receives

Explorer receives [all findings](#) from Security Hub. You can see all findings based on severity in the Explorer widget when you turn on the Security Hub default settings. Explorer creates OpsItems for Critical and High security findings by default. You can configure Explorer to create OpsItems for Medium and Low

severity findings. Explorer doesn't create OpsItems for Informational severity findings. However, you can view Informational OpsData in the Explorer dashboard from the Security Hub findings summary widget. Explorer creates OpsData for all findings regardless of severity. For more information about the severity of Security Hub findings, see [Severity](#) in the *AWS Security Hub User Guide*.

### How long does it take to receive findings from Security Hub?

When Security Hub creates a new finding, it's usually visible in Explorer within seconds.

### Retrying if there is a system outage

Data from Security Hub is retained in Explorer for up to 7 days to retry if there is a system outage. Security Hub also refreshes every 12 hours on all Security Hub standard rules.

## Turning on and configuring the integration

This topic describes how to configure Explorer start receiving Security Hub findings.

### Before you begin

Complete the following tasks before you configure Explorer to start receiving Security Hub findings.

- Turn on and configure Security Hub. For more information, see [Setting up Security Hub](#) in the *AWS Security Hub User Guide*.
- Log into the AWS Organizations management account. Systems Manager requires access to AWS Organizations to create OpsItems from Security Hub findings. After you log in to the management account, you're prompted to select the **Enable access** button on the Explorer **Configure dashboard** tab, as described in the following procedure. If you don't log in to the AWS Organizations management account, you can't allow access and Explorer can't create OpsItems from Security Hub findings.

### To start receiving Security Hub findings

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Explorer**.
3. Select **Settings**.
4. Select the **Configure dashboard** tab.
5. Select **AWS Security Hub**.
6. Select the **Disabled** slider to turn on **AWS Security Hub**.  
Critical and High security findings are displayed by default. To also display Medium and Low security findings, select the **Disabled** slider next to **Medium,Low**.
7. In the **OpsItems created by Security Hub findings** section, choose **Enable access**. If you don't see this button, log in to the AWS Organizations management account and return to this page to select the button.

## How to view findings from Security Hub

The following procedure describes how to view Security Hub findings.

### To view Security Hub findings

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Explorer**.
3. Find the **AWS Security Hub findings summary** widget. This displays your Security Hub findings. You can select a severity level to view a detailed description of the corresponding OpsItem.

## How to stop receiving findings

The following procedure describes how to stop receiving Security Hub findings.

## To stop receiving Security Hub findings

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
  2. In the navigation pane, choose **Explorer**.
  3. Select **Settings**.
  4. Select the **Configure dashboard** tab.
  5. Select the **Enabled** slider to turn off **AWS Security Hub**.

## Exporting OpsData from Systems Manager Explorer

When you select a link in AWS Systems Manager Explorer, some pages display OpsData in a list. These pages include an **Export** button that allows 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.

Data				
Id	Title	Severity	Status	Timestamp
oi-3ea5	SSM Maintenance Window execution failed	3	Open	2023-09-01T12:00:00Z
oi-0ca23	EC2 instance terminated	4	Open	2023-09-01T12:00:00Z
oi-f497	EC2 instance terminated	4	Open	2023-09-01T12:00:00Z

## 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 is complete. 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.

```
{
 "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}}"
]
},
{
 "Effect": "Allow",
 "Action": [
 "logs:DescribeLogGroups",
 "logs:DescribeLogStreams"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "logs>CreateLogGroup",
 "logs:PutLogEvents",
 "logs>CreateLogStream"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "ssm:GetOpsSummary"
],
 "Resource": [
 "*"
]
}
]
}

```

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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 is complete.
7. Choose **Create**.

You can now export OpsData from Explorer pages to the specified S3 bucket.

If you can't export data by using this procedure, verify that your IAM user account, group, or role includes the `iam:CreatePolicyVersion` and `iam:DeletePolicyVersion` actions. For information about adding these actions to your user account, group, or role, see [Editing IAM policies](#) in the *IAM User Guide*.

## Troubleshooting Systems Manager Explorer

This topic includes information about how to troubleshoot common problems with AWS Systems Manager Explorer.

### Not able to filter AWS resources in Explorer after updating tags on the Settings page

If you update tags keys or other data settings in Explorer, the system can take up to six hours to synchronize data based on your changes.

### The AWS Organizations options on the *Create resource data sync* page are greyed out

The **Include all accounts from my AWS Organizations configuration** and **Select organization units in AWS Organizations** 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 management 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 \(p. 167\)](#) and [Configuring a Delegated Administrator \(p. 170\)](#).

### 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 \(p. 160\)](#).
- When using Explorer to view data from multiple accounts and Regions, verify that you're logged into the AWS Organizations management account. To view OpsData and OpsItems from multiple accounts and Regions, you must be signed in to this account.

### Widgets about Amazon EC2 instances don't display data

If widgets about Amazon Elastic Compute Cloud (Amazon 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 active on the **Settings** page. Also, verify that more than 6 hours have passed since you activated 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 activated 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're in the correct AWS Region where you configured your Amazon 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're signed in to the Organizations management account.

### Patch widget doesn't display data

The **Non-compliant instances for patching** widget only displays data about patch instances that aren't 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 AWS Systems Manager Patch Manager to check your patch compliance. For more information, see [AWS Systems Manager Patch Manager \(p. 1093\)](#).

### 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, a capability of AWS Systems Manager, 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 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. To get started with OpsCenter, open the [Systems Manager console](#). In the navigation pane, choose **OpsCenter**.

OpsCenter is integrated with Amazon EventBridge and Amazon CloudWatch. This means you can configure these services to automatically create an OpsItem in OpsCenter when a CloudWatch alarm enters the ALARM state or when EventBridge processes an event from any AWS service that publishes events. Configuring CloudWatch alarms and EventBridge events to automatically create OpsItems allows you to quickly diagnose and remediate issues with AWS resources from a single console.

To help you diagnose issues, each OpsItem includes contextually relevant information such as the name and ID of the AWS resource that generated the OpsItem, alarm or event details, alarm history, and an alarm timeline graph.

For the AWS resource, OpsCenter aggregates information from AWS Config, AWS CloudTrail logs, and Amazon CloudWatch Events, so you don't have to navigate across multiple console pages during your investigation.

The following list includes types of AWS resources and metrics for which customers configure CloudWatch alarms that create OpsItems.

- Amazon DynamoDB: database read and write actions reach a threshold
- Amazon EC2: CPU utilization reaches a threshold
- AWS billing: estimated charges reach a threshold
- Amazon EC2: an instance fails a status check
- Amazon Elastic Block Store (EBS): disk space utilization reaches a threshold

The following list includes types of EventBridge rules customer configure to create OpsItems.

- AWS Security Hub: security alert issued
- DynamoDB: a throttling event
- Amazon EC2 Auto Scaling: failure to launch an instance
- Systems Manager: failure to run an automation
- AWS Health: an alert for scheduled maintenance
- 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. You can also integrate OpsCenter with AWS Security Hub to aggregate and take action on your security, performance, and operational issues in Systems Manager.

Operations engineers and IT professionals can create, view, and edit OpsItems by using the OpsCenter page in the AWS Systems Manager console, public API operations, the AWS Command Line Interface (AWS CLI), AWS Tools for Windows PowerShell, or the AWS SDKs. OpsCenter public API operations also allows you to integrate OpsCenter with your case management systems and health dashboards.

## OpsCenter integration

The following table describes how OpsCenter integrates with other AWS services and Systems Manager capabilities. When it's integrated with these services and capabilities, OpsCenter helps you to quickly diagnose and remediate issues with AWS resources from a single console.

Service or capability	Details	For more information
EventBridge	<p>You can configure Amazon EventBridge to automatically create an OpsItem in OpsCenter when the system processes an event from any AWS service that publishes events. The following list includes types of EventBridge rules that you can configure to create OpsItems:</p> <ul style="list-style-type: none"><li>• AWS Security Hub: security alert issued</li></ul>	<a href="#">Configuring EventBridge to automatically create OpsItems for specific events (p. 203)</a>

Service or capability	Details	For more information
	<ul style="list-style-type: none"><li>Amazon DynamoDB: a throttling event</li><li>Amazon EC2 Auto Scaling: failure to launch an instance</li><li>Systems Manager: failure to run an automation</li><li>AWS Health: an alert for scheduled maintenance</li><li>Amazon Elastic Compute Cloud (Amazon EC2): instance state change from <b>Running</b> to <b>Stopped</b></li></ul> <p>To help you diagnose issues, each OpsItem includes contextually relevant information about the event, such as the name and ID of the AWS resource that generated the OpsItem and details about the event.</p>	

Service or capability	Details	For more information
CloudWatch	<p>You can configure Amazon CloudWatch to automatically create an OpsItem in OpsCenter when a CloudWatch alarm enters the ALARM state. The following list includes types of AWS resources and metrics for which you can configure CloudWatch alarms to create OpsItems:</p> <ul style="list-style-type: none"> <li>• DynamoDB: database read and write actions reach a threshold</li> <li>• Amazon EC2: CPU utilization reaches a threshold</li> <li>• AWS Billing and Cost Management: estimated charges reach a threshold</li> <li>• Amazon EC2: an instance fails a status check</li> <li>• Amazon Elastic Block Store (Amazon EBS): disk space utilization reaches a threshold</li> </ul> <p>To help you diagnose issues, each OpsItem includes contextually relevant information about the alarm, such as the name and ID of the AWS resource that generated the OpsItem, alarm details, alarm history, and an alarm timeline graph.</p>	<a href="#">Configuring CloudWatch to create OpsItems from alarms (p. 196)</a>

Service or capability	Details	For more information
Incident Manager	<p>AWS Incident Manager, a capability of Systems Manager, provides an incident management console that helps you mitigate and recover from incidents affecting your AWS hosted applications. An <i>incident</i> is any unplanned interruption or reduction in quality of services. After you set up and configure Incident Manager, the system automatically creates OpsItems in OpsCenter when incidents are created in Incident Manager. You can also manually add incidents to an OpsItem.</p> <p>After an incident is resolved, post incident analysis guides you through identifying improvements to your incident response and recommending action items to address the findings. With high severity operational issues such as incidents, creating an OpsItem in OpsCenter provides operators with a complete view of the incident, analysis, and action items. This comprehensive view improves time-to-resolution and helps mitigate similar issues in the future.</p>	<p><a href="#">Working with Incident Manager incidents in OpsCenter (p. 219)</a></p> <p><a href="#">AWS Systems Manager Incident Manager User Guide</a></p>

Service or capability	Details	For more information
CloudWatch Application Insights for .NET and SQL Server.	OpsCenter is also integrated with CloudWatch Application Insights for .NET and SQL Server. CloudWatch Application Insights helps you monitor your applications that use Amazon EC2 instances along with other application resources. This capability identifies and sets up key metrics, logs, and alarms across your application resources and technology stack. This capability also creates automated dashboards for detected problems. Dashboards include correlated metric anomalies, log errors, and other information to help you determine the root cause of the errors. When you configure application resources in CloudWatch Application Insights, you can choose to have the system create OpsItems in OpsCenter when problems are detected.	<a href="#">Setting Up Your Application</a> in the <i>Amazon CloudWatch User Guide</i>

For each AWS resource that automatically generates an OpsItem, OpsCenter aggregates information from AWS Config, AWS CloudTrail logs, and EventBridge. As a result, you don't have to navigate across multiple console pages during your investigation.

## How can OpsCenter benefit my organization?

OpsCenter provides 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 CloudWatch alarms, EventBridge 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 run Systems Manager Automation runbooks to resolve issues.

## What are the features of OpsCenter?

- **Automated and manual OpsItem creation**

OpsCenter is integrated with Amazon CloudWatch. This means you can configure CloudWatch to automatically create an OpsItem in OpsCenter when an alarm enters the ALARM state or when Amazon EventBridge processes an event from any AWS service that publishes 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 initiated the EventBridge 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 \(p. 209\)](#).
- **Related and Similar OpsItems:** With the **Related OpsItems** feature, you can 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 might 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 operation) 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 operation).

- **Deduplication:** By specifying related resources, OpsCenter uses built-in logic to help you avoid creating duplicate OpsItems. OpsCenter also includes a feature called **Operational insights**, which displays information about duplicate OpsItems. To further limit the number of duplicate OpsItems in your account, you can manually specify a deduplication string for an EventBridge event rule. For more information, see [Reducing duplicate OpsItems \(p. 214\)](#).
- **Bulk edit OpsItems:** You can select multiple OpsItems in OpsCenter and edit one of the following fields: **Status**, **Priority**, **Severity**, **Category**.
- **Easy remediation using runbooks**

Each OpsItem includes a **Runbooks** section with a list of Systems Manager Automation runbooks that you can use to automatically remediate common issues with AWS resources. If you open an OpsItem, choose an AWS resource for that OpsItem, and then choose the **Run automation** button in the console, then OpsCenter provides a list of Automation runbooks that you can run on the AWS resource that generated the OpsItem. After you run an Automation runbook from an OpsItem, the runbook is automatically associated with the related resource of the OpsItem for future reference.

Additionally, if you automatically set up OpsItem rules in EventBridge by using OpsCenter, then EventBridge automatically associates runbooks for common events. OpsCenter keeps a 30-day record of Automation runbooks run for a specific OpsItem. For more information, see [Remediating OpsItem issues using Systems Manager Automation \(p. 220\)](#).

- **Change notification:** You can specify the ARN of an Amazon Simple Notification Service (SNS) topic and publish notifications anytime an OpsItem is changed or edited. The SNS topic must exist in the same AWS Region as the OpsItem.
- **Comprehensive OpsItem search capabilities:** OpsCenter provides multiple search options to help you quickly locate OpsItems. Here are several examples of how you can search: OpsItem ID, Title, Last modified time, Operational data value, Source, and Automation ID of a runbook execution, to name a few. You can further limit search results by using status filters.
- **OpsItem summary reports**

OpsCenter includes a summary report page that automatically displays the following sections:

- **Status summary:** a summary of OpsItems by status (Open, In progress, Resolved, Open and In progress).
- **Sources with most open OpsItems:** a breakdown of the top AWS services with open OpsItems.
- **OpsItems by source and age:** a count of OpsItems grouped by source and days since creation.

For more information about viewing OpsCenter summary reports, see [Viewing OpsCenter summary reports \(p. 224\)](#).

- **IAM access control**

By using AWS Identity and Access Management (IAM) policies, you can control which members of your organization can create, view, list, and update OpsItems. You can also assign tags to OpsItems and then create IAM policies that give access to users and groups based on tags. For more information, see [Getting started with OpsCenter \(p. 189\)](#).

- **Logging and auditing capability support**

You can audit and log OpsCenter user actions in your AWS account through integration with other AWS services. For more information, see [Auditing and logging OpsCenter activity \(p. 229\)](#).

- **Console, CLI, PowerShell, and SDK access to OpsCenter capabilities**

You can work with OpsCenter by using the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), AWS Tools for PowerShell, or the AWS SDK of your choice.

## How does OpsCenter work with Amazon EventBridge? Which service should I use?

Amazon EventBridge 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, EventBridge informs you that there is a problem with your resources.

OpsCenter helps you investigate and remediate the problem. OpsCenter brings together data from EventBridge or data entered manually by engineers so that your engineers can perform a thorough investigation. OpsCenter also provides Automation runbooks for quickly remediating those issues. OpsCenter integrates with EventBridge by allowing 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 operations. 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 operations, see the following API operations 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 nodes?

Yes. You can use OpsCenter to investigate and remediate issues with your on-premises managed nodes 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 \(p. 34\)](#).

## What are the quotas for OpsCenter?

You can view quotas for all Systems Manager capabilities in the [Systems Manager service quotas](#) in the *Amazon Web Services General Reference*. Unless otherwise noted, each quota is Region-specific.

### Topics

- [Getting started with OpsCenter \(p. 189\)](#)
- [Creating OpsItems \(p. 195\)](#)
- [Working with OpsItems \(p. 209\)](#)
- [Reducing duplicate OpsItems \(p. 214\)](#)
- [Working with Incident Manager incidents in OpsCenter \(p. 219\)](#)
- [Remediating OpsItem issues using Systems Manager Automation \(p. 220\)](#)
- [Viewing OpsCenter summary reports \(p. 224\)](#)
- [Supported resources reference \(p. 224\)](#)
- [Receiving findings from AWS Security Hub in OpsCenter \(p. 227\)](#)
- [Auditing and logging OpsCenter activity \(p. 229\)](#)

## Getting started with OpsCenter

Set up for AWS Systems Manager OpsCenter is integrated with set up for 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 AWS 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 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 haven't 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 \(p. 160\)](#).

**Note**

Integrated Setup is only available in the AWS Systems Manager console. You can't set up Explorer and OpsCenter programmatically.

## (Optional) Receive OpsItem notifications

You can configure OpsCenter to send notifications to an Amazon Simple Notification Service (Amazon SNS) topic when the system 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 \(p. 189\)](#)
- [Task 2: Update the Amazon SNS access policy \(p. 189\)](#)
- [Task 3: Update the AWS KMS access policy \(optional\) \(p. 190\)](#)
- [Task 4: Turn on default OpsItems rules to send notifications for new OpsItems \(p. 191\)](#)

### 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 AWS account as the OpsItem. If you're 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.

1. Sign in to the AWS Management Console and open the Amazon SNS console at <https://console.aws.amazon.com/sns/v3/home>.
2. In the navigation pane, choose **Topics**.
3. Choose the topic you created in task 1, and then choose **Edit**.
4. Expand **Access policy**.
5. Add the following Sid block to the existing policy.

```
{
 "Sid": "Allow OpsCenter to publish to this topic",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "SNS:Publish",
 "Resource": "arn:aws:sns:Region:account_ID:topic_name"
}
```

Enter this block after the existing Sid block.

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 turned on AWS KMS server-side encryption for your Amazon SNS topic, then you must also update the access policy of the AWS KMS 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 doesn't support publishing OpsItems to an Amazon SNS topic configured with an AWS managed key.

1. Open the AWS KMS console at <https://console.aws.amazon.com/kms>.
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 KMS 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.

```
{
 "Sid": "Allow OpsItems to decrypt the key",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": ["kms:Decrypt", "kms:GenerateDataKey*"],
 "Resource": "arn:aws:kms: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.

## Edit key policy



```
8 "Principal": {
9 "AWS": "arn:aws:iam::1234567890:root"
10 },
11 "Action": "kms:*",
12 "Resource": "*"
13 },
14 {
15 "Sid": "Allow OpsItems to decrypt the key",
16 "Effect": "Allow",
17 "Principal": {
18 "Service": "ssm.amazonaws.com"
19 },
20 "Action": ["kms:Decrypt", "kms:GenerateDataKey*"],
21 "Resource": "arn:aws:kms:us-west-1:1234567890:key/e888567c-a2b8-43da-9f3e-18b87d987ee8"
22 },
```

8. Choose **Save changes**.

### Task 4: Turn on default OpsItems rules to send notifications for new OpsItems

Default OpsItems rules in Amazon EventBridge aren't configured with an ARN for Amazon SNS notifications. Use the following procedure to edit a rule in EventBridge and enter a notifications block.

#### To add a notifications block to a default OpsItem rule

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.
3. Choose the **OpsItems** tab, and then choose **Configure sources**.
4. Choose the name of the source rule that you want to configure with a notification block, as shown in the following example:

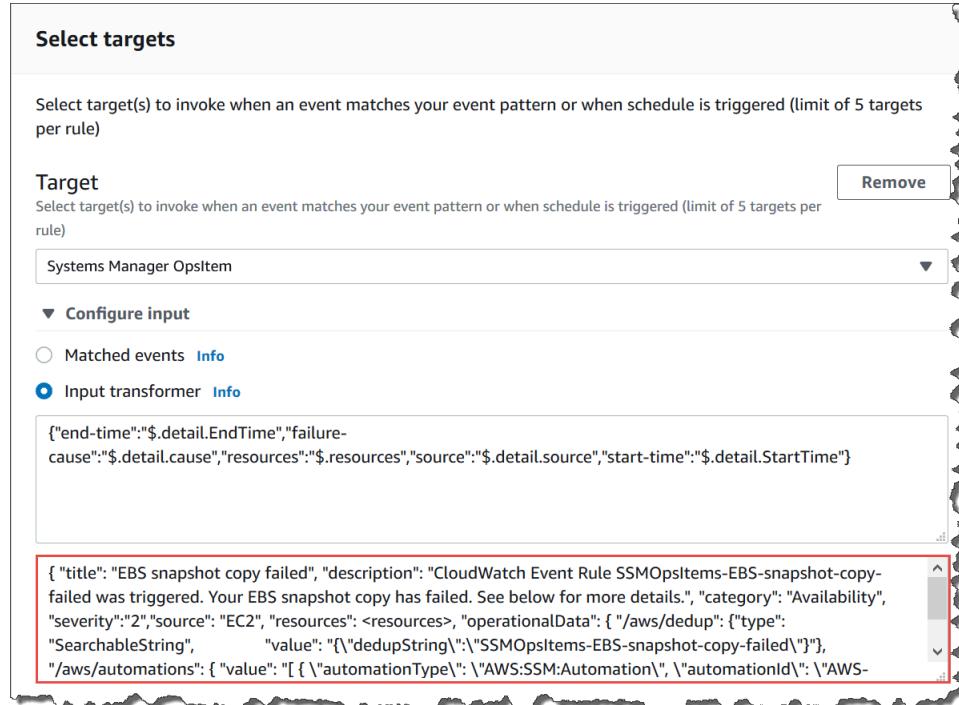
OpsItem rules		Category
Rule		Category
<a href="#">SSMOpsItems-Autoscaling-instance-launch-failure</a>		Availability
<a href="#">SSMOpsItems-Autoscaling-instance-termination-failure</a>		Availability
<a href="#">SSMOpsItems-EBS-snapshot-copy-failed</a>		Availability
<a href="#">SSMOpsItems-EBS-snapshot-creation-failed</a>		Availability
<a href="#">SSMOpsItems-EBS-volume-performance-issue</a>		Performance
<a href="#">SSMOpsItems-EC2-issue</a>		Availability

The rule opens in Amazon EventBridge.

5. On the rule details page, choose **Edit**.

SSMOpsItems-EBS-snapshot-copy-failed	
Rule details	
Rule name	Status
SSMOpsItems-EBS-snapshot-copy-failed	Enabled
Description	Event bus name
Rule for SSM OpsCenter to create OpsItems when EBS snapshot copy failed	default

6. Scroll to the **Select targets** section.



- Enter a notifications block in the following format:

```
"notifications": [{"arn": "arn:aws:sns:region:account_ID:topic_name"}],
```

Here's an example.

```
"notifications": [{"arn": "arn:aws:sns:us-west-2:1234567890:MySNSTopic"}],
```

Enter the notifications block before the resources block, as shown here.

```
{
 "title": "EBS snapshot copy failed",
 "description": "CloudWatch Event Rule SSMOpsItems-EBS-snapshot-copy-failed was triggered. Your EBS snapshot copy has failed. See below for more details.",
 "category": "Availability",
 "severity": "2",
 "source": "EC2",
 "notifications": [{"arn": "arn:aws:sns:us-west-2:1234567890:MySNSTopic"}],
 "resources": "<resources>",
 "operationalData": {
 "/aws/dedup": {"type": "SearchableString", "value": "\"dedupString\"\\\"SSMOpsItems-EBS-snapshot-copy-failed\\\""}, "/aws/automations": {"value": "[{ \"automationType\": \"AWS:SSM:Automation\", \"automationId\": \"AWS-CopySnapshot\\\" }]"}
 }
}
```

- 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 EventBridge 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.

Field	Description
<b>Title</b>	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 environment, and the name or the alias of the engineer actively working the issue, if applicable. All OpsItems created by EventBridge 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 <i>Title:contains</i> . If your naming guidelines encourage consistent use of keywords, you improve your search results.
<b>Source</b>	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 <b>Source</b> field after the OpsItem is created.
<b>Priority</b>	(Optional) Guidelines should 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.
<b>Severity</b>	(Optional) Guidelines should include determining the highest and lowest severity for your organization, and any service-level agreements based on severity. You can specify severity from 1 to 4.
<b>Category</b>	(Optional) Guidelines should include a list of categories to specify when creating or editing OpsItems.
<b>Deduplication strings</b>	(Optional) Guidelines should specify the length and standards for creating effective deduplication strings.
<b>Description</b>	Guidelines should suggest how much detail about the issue to include and any steps (if applicable) for reproducing the issue.

Field	Description
<b>Notifications</b>	Guidelines should suggest which Amazon Simple Notification Service (Amazon SNS) topic Amazon Resource Name (ARN) to specify when creating or editing OpsItems. Be aware that Amazon SNS notifications are region-specific. This means you must specify an ARN that is in the same AWS Region as the OpsItem.
<b>Related Resources</b>	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.
<b>Operational data</b>	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.

## Creating OpsItems

AWS Systems Manager automatically creates operational work items (OpsItems) in OpsCenter for some AWS services and Systems Manager capabilities, if those services and capabilities are enabled. You can also configure the system to create OpsItems automatically based on Amazon CloudWatch alarms and Amazon EventBridge events. When an alarm reaches the alarm state or an event is processed, the system creates an OpsItem in OpsCenter and provides detailed information about the alarm or the event in the OpsItem. You can also create OpsItems manually.

### Note

When you initially configure OpsCenter by using Integrated Setup, you allow EventBridge to automatically create OpsItems based on common rules. The procedures in this section describe how to configure a specific alarm or event to create OpsItems. For information about allowing default EventBridge rules for creating OpsItems by using Integrated Setup, see [Getting started with Systems Manager Explorer and OpsCenter \(p. 160\)](#).

### Contents

- [Services and capabilities that automatically create OpsItems \(p. 196\)](#)
- [Configuring CloudWatch to create OpsItems from alarms \(p. 196\)](#)
- [Configuring EventBridge to automatically create OpsItems for specific events \(p. 203\)](#)
- [Configuring CloudWatch Application Insights for .NET and SQL Server to automatically create OpsItems \(p. 205\)](#)
- [Creating OpsItems manually \(p. 205\)](#)

## Services and capabilities that automatically create OpsItems

OpsCenter, a capability of AWS Systems Manager, integrates with the following AWS services and Systems Manager capabilities. When enabled, these services and capabilities can automatically create OpsItems in OpsCenter on your behalf.

### Amazon DevOps Guru

DevOps Guru applies machine learning to analyze your operational data, application metrics, and application events to identify behaviors that deviate from normal operating patterns. The system notifies you when DevOps Guru detects an operational issue or risk.

If you enabled DevOps Guru, the system creates OpsItems on your behalf by using the following AWS Identity and Access Management (IAM) service-linked role: [AWSServiceRoleForDevOpsGuru](#).

### AWS Security Hub

Security Hub provides you with a comprehensive view of the security state of your AWS resources. Security Hub collects security data from across AWS accounts and services, and helps you analyze your security trends to identify and prioritize the security issues across your AWS environment.

If you enabled Security Hub, and if you enabled the Security Hub data source in Explorer, then the system creates OpsItems on your behalf by using the following IAM service-linked role: [AWSServiceRoleForAmazonSSM\\_OpsInsights](#).

### OpsCenter operational insights

You can configure OpsCenter to automatically analyze OpsItems in your account and generate two types of insights. In the Systems Manager console, the **Duplicate OpsItems** field displays a count of insights when eight or more OpsItems have the same title for the same resource. The **Sources generating most OpsItems** field displays a count of insights when more than 50 OpsItems have the same title. If you choose an insight, OpsCenter displays information about the affected OpsItems and resources.

If you enabled operational insights, then the system creates OpsItems on your behalf by using the following IAM service-linked role: [AWSSMOpsInsightsServiceRolePolicy](#).

## Configuring CloudWatch to create OpsItems from alarms

You can configure Amazon CloudWatch to automatically create an OpsItem in OpsCenter, a capability of AWS Systems Manager, when an alarm enters the `ALARM` state. Doing so allows you to quickly diagnose and remediate issues with AWS resources from a single console.

For example, you can configure an alarm to automatically create an OpsItem if there is a spike in HTTP errors generated by your Application Load Balancer. To help you diagnose the issue, the OpsItem includes contextually relevant information such as the name and ID of the monitored AWS resource, alarm details, alarm history, and alarm timeline graph. For the monitored AWS resource, OpsCenter aggregates information from AWS Config, AWS CloudTrail logs, and EventBridge, so you don't have to navigate across multiple console pages during your investigation. You can run Systems Manager Automation runbooks in OpsCenter for easy remediation.

This feature is available in all AWS Regions where Systems Manager is available. Note the following important details about this feature:

- CloudWatch can create OpsItems in OpsCenter for metric and composite alarms.
- Alarms must use the default aws namespace, such as `AWS/EC2` (metric alarms only).
- CloudWatch automatically creates a new service-linked role in AWS Identity and Access Management (IAM) when you configure an alarm to create OpsItems. The new role is named

`AWSServiceRoleForCloudWatchAlarms_ActionSSM`. For more information about CloudWatch service-linked roles, see [Using Service-Linked Roles for CloudWatch](#) in the *Amazon CloudWatch User Guide*.

- OpsCenter uses a deduplication feature to prohibit a single alarm from creating multiple OpsItems. For more information, see [Reducing duplicate OpsItems \(p. 214\)](#).

For information about how to create a new alarm that automatically creates OpsItems in OpsCenter, see [Create a CloudWatch alarm based on a static threshold](#) in the *Amazon CloudWatch User Guide*. In Step 8 of that procedure, choose **Systems Manager OpsCenter action** and then complete the procedure.

#### Topics

- [Manually configure an existing alarm to create OpsItems \(console\) \(p. 197\)](#)
- [Programmatically configure CloudWatch alarms to create OpsItems \(p. 197\)](#)

## Manually configure an existing alarm to create OpsItems (console)

Use the following procedure to edit an existing alarm and configure **Systems Manager** as the target of that alarm. When the alarm enters the **ALARM** state, CloudWatch creates a new OpsItem in OpsCenter.

### To edit an existing alarm and configure Systems Manager as a target of that alarm

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**.
3. Select the alarm, and then choose **Actions, Edit**.
4. (Optional) Change settings in the **Metrics** and **Conditions** sections, and then choose **Next**.
5. In the **Systems Manager** section, choose **Add Systems Manager OpsCenter action**.
6. For **Severity**, choose a number.

#### Note

Severity is a user-defined value. You or your organization determine what each severity value means and any service-level agreement associated with each severity.

7. (Optional) For **Category**, choose an option.
8. Choose **Next** and complete the wizard.

An OpsItem created from an alarm shows **CloudWatch alarm - '*alarm\_name*' is in ALARM state**. To view details about a specific OpsItem created from an alarm, choose the OpsItem and then choose the **Related resource details** tab.

#### Note

If an alarm creates an OpsItem and if you specified a deduplication string, the alarm won't create additional OpsItems even if you edit the alarm in CloudWatch. (If the OpsItem is resolved in OpsCenter, CloudWatch will create a new OpsItem.)

If you edit an alarm and change the severity or the category for any new OpsItems created from it, Systems Manager won't change the severity or category of OpsItems already created from that alarm. You can manually edit OpsItems to change details such as the severity or category.

## Programmatically configure CloudWatch alarms to create OpsItems

You can programmatically configure Amazon CloudWatch alarms to create OpsItems by using the AWS Command Line Interface (AWS CLI), AWS CloudFormation templates, or Java code snippets.

#### Topics

- [Before you begin \(p. 198\)](#)
- [Manually configure an existing alarm to create OpsItems \(AWS CLI\) \(p. 199\)](#)

- [Use AWS CloudFormation templates to configure Amazon CloudWatch alarms to automatically create OpsItems \(p. 200\)](#)
- [Use Java code snippets to configure CloudWatch alarms to automatically create OpsItems \(p. 202\)](#)

## Before you begin

If you programmatically edit an existing alarm or create a new alarm that creates OpsItems, you must specify an Amazon Resource Name (ARN). This ARN identifies Systems Manager OpsCenter as the target for OpsItems created from the alarm. You can customize the ARN so that OpsItems created from the alarm include specific information such as severity or category. Each ARN includes the information described in the following table.

Parameter	Details
Region (required)	The AWS Region where the alarm exists. For example: us-west-2. For information about AWS Regions where you can use OpsCenter, see <a href="#">AWS Systems Manager endpoints and quotas</a> .
AWS account ID (required)	The same AWS account ID used to create the alarm. For example: 123456789012. The account ID must be followed by a colon (:) and the parameter opsitem as shown in the following examples.
Severity (required)	A user-defined severity level for OpsItems created from the alarm. Valid values: 1,2,3,4  Because this is a user-defined value, you or your organization determine what each severity value means and any service-level agreement associated with each severity.
Category (optional)	A category for OpsItems created from the alarm. Valid values: Availability, Cost, Performance, Recovery, Security.

Create the ARN by using the following syntax. This ARN doesn't include the optional Category parameter.

```
arn:aws:ssm:Region:account_ID:opsitem:severity
```

Following is an example.

```
arn:aws:ssm:us-west-2:123456789012:opsitem:3
```

To create an ARN that uses the optional Category parameter, use the following syntax.

```
arn:aws:ssm:Region:account_ID:opsitem:severity#CATEGORY=category_name
```

Following is an example.

```
arn:aws:ssm:us-west-2:123456789012:opsitem:3#CATEGORY=Security
```

## Manually configure an existing alarm to create OpsItems (AWS CLI)

Use the following command to configure an existing alarm to create OpsItems by using the AWS CLI. This command requires that you specify an Amazon Resource Name (ARN) for the `alarm-actions` parameter. For information about how to create the ARN, see [Before you begin \(p. 198\)](#).

### To configure an existing alarm to create OpsItems

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to collect information about the alarm that you want to configure.

```
aws cloudwatch describe-alarms --alarm-names "alarm_name"
```

3. Run the following command to update an alarm.

```
aws cloudwatch put-metric-alarm --alarm-name name \
--alarm-description "description" \
--metric-name name --namespace namespace \
--statistic statistic --period value --threshold value \
--comparison-operator value \
--dimensions dimensions --evaluation-periods value \
--alarm-actions arn:aws:ssm:Region:account_ID:opsitem:severity#CATEGORY=category_name \
--unit unit
```

Here's an example.

Linux & macOS

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon \
--alarm-description "Alarm when CPU exceeds 70 percent" \
--metric-name CPUUtilization --namespace AWS/EC2 \
--statistic Average --period 300 --threshold 70 \
--comparison-operator GreaterThanThreshold \
--dimensions "Name=InstanceId,Value=i-12345678" --evaluation-periods 2 \
--alarm-actions arn:aws:ssm:us-east-1:123456789012:opsitem:3#CATEGORY=Security \
--unit Percent
```

Windows

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon ^
--alarm-description "Alarm when CPU exceeds 70 percent" ^
--metric-name CPUUtilization --namespace AWS/EC2 ^
--statistic Average --period 300 --threshold 70 ^
--comparison-operator GreaterThanThreshold ^
--dimensions "Name=InstanceId,Value=i-12345678" --evaluation-periods 2 ^
--alarm-actions arn:aws:ssm:us-east-1:123456789012:opsitem:3#CATEGORY=Security ^
--unit Percent
```

An OpsItem created from an alarm shows **CloudWatch alarm - 'alarm\_name' is in ALARM state**. To view details about a specific OpsItem created from an alarm, choose the OpsItem and then choose the **Related resource details** tab.

**Note**

If an alarm creates an OpsItem and if you specified a deduplication string, the alarm won't create additional OpsItems even if you edit the alarm in CloudWatch. (If the OpsItem is resolved in OpsCenter, CloudWatch will create a new OpsItem.)

If you edit an alarm and change the severity or the category for any new OpsItems created from it, Systems Manager won't change the severity or category of OpsItems already created from that alarm. You can manually edit OpsItems to change details such as the severity or category.

## Use AWS CloudFormation templates to configure Amazon CloudWatch alarms to automatically create OpsItems

This section includes AWS CloudFormation templates that you can use to configure CloudWatch alarms to automatically create OpsItems. Each template requires that you specify an Amazon Resource Name (ARN) for the `AlarmActions` parameter. For information about how to create the ARN, see [Before you begin \(p. 198\)](#).

### Metric alarm

Use the following AWS CloudFormation template to create or update an Amazon CloudWatch metric alarm. The alarm specified in this template monitors Amazon EC2 instance status checks. If the alarm enters the `ALARM` state, it creates an OpsItem in OpsCenter.

```
{
 "AWSTemplateFormatVersion": "2010-09-09",
 "Parameters": {
 "RecoveryInstance": {
 "Description": "The EC2 instance ID to associate this alarm with.",
 "Type": "AWS::EC2::InstanceId"
 }
 },
 "Resources": {
 "RecoveryTestAlarm": {
 "Type": "AWS::CloudWatch::Alarm",
 "Properties": {
 "AlarmDescription": "Run a recovery action when instance status check fails for 15 consecutive minutes.",
 "Namespace": "AWS/EC2",
 "MetricName": "StatusCheckFailed_System",
 "Statistic": "Minimum",
 "Period": "60",
 "EvaluationPeriods": "15",
 "ComparisonOperator": "GreaterThanOrEqualToThreshold",
 "Threshold": "0",
 "AlarmActions": [{"Fn::Join": ["",
 ["arn:aws:ssm:Region:account_ID:opsitem:severity#CATEGORY=category_name", { "Ref" :
 "AWS::Partition" }, ":ssm:", { "Ref" : "AWS::Region" }, { "Ref" : "AWS::AccountId" },
 ":opsitem:3"]]}],
 "Dimensions": [{"Name": "InstanceId", "Value": { "Ref": "RecoveryInstance" }}]
 }
 }
 }
}
```

### Composite alarm

Use the following AWS CloudFormation template to create or update a composite alarm. A composite alarm consists of multiple metric alarms. If the alarm enters the `ALARM` state, it creates an OpsItem in OpsCenter.

```
"Resources": {
```

```

 "HighResourceUsage": {
 "Type": "AWS::CloudWatch::CompositeAlarm",
 "Properties": {
 "AlarmName": "HighResourceUsage",
 "AlarmRule": "(ALARM(HighCPUUsage) OR ALARM(HighMemoryUsage)) AND NOT ALARM(DeploymentInProgress)",

 "AlarmActions": "arn:aws:ssm:Region:account_ID:opsitem:severity#CATEGORY=category_name",
 "AlarmDescription": "Indicates that the system resource usage is high while no known deployment is in progress"
 },
 "DependsOn": [
 "DeploymentInProgress",
 "HighCPUUsage",
 "HighMemoryUsage"
]
 },
 "DeploymentInProgress": {
 "Type": "AWS::CloudWatch::CompositeAlarm",
 "Properties": {
 "AlarmName": "DeploymentInProgress",
 "AlarmRule": "FALSE",
 "AlarmDescription": "Manually updated to TRUE/FALSE to disable other alarms"
 }
 },
 "HighCPUUsage": {
 "Type": "AWS::CloudWatch::Alarm",
 "Properties": {
 "AlarmDescription": "CPU usage is high",
 "AlarmName": "HighCPUUsage",
 "ComparisonOperator": "GreaterThanThreshold",
 "EvaluationPeriods": 1,
 "MetricName": "CPUUsage",
 "Namespace": "CustomNamespace",
 "Period": 60,
 "Statistic": "Average",
 "Threshold": 70,
 "TreatMissingData": "notBreaching"
 }
 },
 "HighMemoryUsage": {
 "Type": "AWS::CloudWatch::Alarm",
 "Properties": {
 "AlarmDescription": "Memory usage is high",
 "AlarmName": "HighMemoryUsage",
 "ComparisonOperator": "GreaterThanThreshold",
 "EvaluationPeriods": 1,
 "MetricName": "MemoryUsage",
 "Namespace": "CustomNamespace",
 "Period": 60,
 "Statistic": "Average",
 "Threshold": 65,
 "TreatMissingData": "breaching"
 }
 }
 }
}

```

An OpsItem created from an alarm shows **CloudWatch alarm - '*alarm\_name*' is in ALARM state**. To view details about a specific OpsItem created from an alarm, choose the OpsItem and then choose the **Related resource details** tab.

#### Note

If an alarm creates an OpsItem and if you specified a deduplication string, the alarm won't create additional OpsItems even if you edit the alarm in CloudWatch. (If the OpsItem is resolved in OpsCenter, CloudWatch will create a new OpsItem.)

If you edit an alarm and change the severity or the category for any new OpsItems created from it, Systems Manager won't change the severity or category of OpsItems already created from that alarm. You can manually edit OpsItems to change details such as the severity or category.

### Use Java code snippets to configure CloudWatch alarms to automatically create OpsItems

This section includes Java code snippets that you can use to configure CloudWatch alarms to automatically create OpsItems. Each snippet requires that you specify an Amazon Resource Name (ARN) for the `validSsmActionStr` parameter. For information about how to create the ARN, see [Before you begin \(p. 198\)](#).

#### A specific alarm

Use the following Java code snippet to create or update a CloudWatch alarm. The alarm specified in this template monitors Amazon EC2 instance status checks. If the alarm enters the `ALARM` state, it creates an OpsItem in OpsCenter.

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.ComparisonOperator;
import com.amazonaws.services.cloudwatch.model.Dimension;
import com.amazonaws.services.cloudwatch.model.PutMetricAlarmRequest;
import com.amazonaws.services.cloudwatch.model.PutMetricAlarmResult;
import com.amazonaws.services.cloudwatch.model.StandardUnit;
import com.amazonaws.services.cloudwatch.model.Statistic;

private void putMetricAlarmWithSsmAction() {
 final AmazonCloudWatch cw =
 AmazonCloudWatchClientBuilder.defaultClient();

 Dimension dimension = new Dimension()
 .withName("InstanceId")
 .WithValue(instanceId);

 String validSsmActionStr =
"arn:aws:ssm:Region:account_ID:opsitem:severity#CATEGORY=category_name";

 PutMetricAlarmRequest request = new PutMetricAlarmRequest()
 .withAlarmName(alarmName)
 .withComparisonOperator(
 ComparisonOperator.GreaterThanThreshold)
 .withEvaluationPeriods(1)
 .withMetricName("CPUUtilization")
 .withNamespace("AWS/EC2")
 .withPeriod(60)
 .withStatistic(Statistic.Average)
 .withThreshold(70.0)
 .withActionsEnabled(false)
 .withAlarmDescription(
 "Alarm when server CPU utilization exceeds 70%")
 .withUnit(StandardUnit.Seconds)
 .withDimensions(dimension)
 .withAlarmActions(validSsmActionStr);

 PutMetricAlarmResult response = cw.putMetricAlarm(request);
}
```

#### Update all alarms

Use the following Java code snippet to update all CloudWatch alarms in your AWS account to create OpsItems when an alarm enters the `ALARM` state.

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
```

```
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClientBuilder;
import com.amazonaws.services.cloudwatch.model.DescribeAlarmsRequest;
import com.amazonaws.services.cloudwatch.model.DescribeAlarmsResult;
import com.amazonaws.services.cloudwatch.model.MetricAlarm;

private void listMetricAlarmsAndAddSsmAction() {
 final AmazonCloudWatch cw = AmazonCloudWatchClientBuilder.defaultClient();

 boolean done = false;
 DescribeAlarmsRequest request = new DescribeAlarmsRequest();

 String validSsmActionStr =
"arn:aws:ssm:Region:account_ID:opsitem:severity#CATEGORY=category_name";

 while(!done) {

 DescribeAlarmsResult response = cw.describeAlarms(request);

 for(MetricAlarm alarm : response.getMetricAlarms()) {
 // assuming there are no alarm actions added for the metric alarm
 alarm.setAlarmActions(ImmutableList.of(validSsmActionStr));
 }

 request.setNextToken(response.getNextToken());

 if(response.getNextToken() == null) {
 done = true;
 }
 }
}
```

An OpsItem created from an alarm shows **CloudWatch alarm - '*alarm\_name*' is in ALARM state**. To view details about a specific OpsItem created from an alarm, choose the OpsItem and then choose the **Related resource details** tab.

**Note**

If an alarm creates an OpsItem and if you specified a deduplication string, the alarm won't create additional OpsItems even if you edit the alarm in CloudWatch. (If the OpsItem is resolved in OpsCenter, CloudWatch will create a new OpsItem.)

If you edit an alarm and change the severity or the category for any new OpsItems created from it, Systems Manager won't change the severity or category of OpsItems already created from that alarm. You can manually edit OpsItems to change details such as the severity or category.

## Configuring EventBridge to automatically create OpsItems for specific events

Use the following procedure to configure **Systems Manager OpsItems** as the target of an Amazon EventBridge event. When EventBridge receives the event, it creates a new OpsItem in OpsCenter. This procedure describes how to update an *existing* EventBridge event rule. For information about how to create a new event rule, see [Creating a rule for an AWS service](#) in the *Amazon EventBridge User Guide*.

### To configure OpsCenter as a target of an EventBridge event

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**.
3. On the **Rules** page, either choose an existing rule and then choose **Edit**, or choose **Create rule**.
4. On the **Rules** page, choose an existing rule and then choose **Edit**.
5. In the **Select event bus** section, verify that **AWS default event bus** is selected and **Enable the rule on the selected event bus** option is toggled on.

6. In the **Select targets** section, use the **Target** list to choose **Systems Manager OpsItem**.
7. Expand **Configure input** and choose either **Matched events** or **Input transformer**.

**Note**

We recommend that you choose **Input transformer**. This option allows you to specify a deduplication string and other important information for OpsItems such as a title and a severity.

If you choose **Input transformer**, enter information in the **Input Path** and **Input Template** boxes. Here's an example of how to enter the input path.

```
{
 "end-time": "$.detail.EndTime",
 "failure-cause": "$.detail.cause",
 "resources": "$.resources",
 "source": "$.detail.source",
 "start-time": "$.detail.StartTime"
}
```

Here's an example of how to enter the input template.

```
{
 "title": "EBS snapshot copy failed",
 "description": "CloudWatch Event Rule SSMOpsItems-EBS-snapshot-copy-failed was triggered. Your EBS snapshot copy has failed. See below for more details.",
 "category": "Availability",
 "severity": "2",
 "source": "EC2",
 "resources": "resources",
 "operationalData": {
 "/aws/dedup": {
 "type": "SearchableString",
 "value": "{\"dedupString\":\"SSMOpsItems-EBS-snapshot-copy-failed\"}"
 },
 "/aws/automations": {
 "value": "[{ \\"automationType\\": \\"AWS:SSM:Automation\\\", \\"automationId\\\": \\"AWS-CopySnapshot\\\" }]"
 },
 "failure-cause": {
 "value": "failure-cause"
 },
 "source": {
 "value": "source"
 },
 "start-time": {
 "value": "start-time"
 },
 "end-time": {
 "value": "end-time"
 }
 }
}
```

For more information about these fields, see [Transforming target input](#) in the *Amazon EventBridge User Guide*.

8. 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 EventBridge permission to create OpsItems in OpsCenter. For more information about the required role and permissions, see [Getting started with OpsCenter \(p. 189\)](#).
9. Choose **Update**.

10. In the navigation pane, choose **Rules**. If you created a new rule, verify that the list includes the new rule.

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 \(p. 209\)](#).

## Configuring CloudWatch Application Insights for .NET and SQL Server to automatically create OpsItems

OpsCenter is integrated with CloudWatch Application Insights for .NET and SQL Server. CloudWatch Application Insights helps you monitor your applications that use Amazon Elastic Compute Cloud (Amazon EC2) instances along with other application resources. This capability identifies and sets up key metrics, logs, and alarms across your application resources and technology stack. This capability also creates automated dashboards for detected problems. Dashboards include correlated metric anomalies, log errors, and other information to help you determine the root cause of the errors. When you configure application resources in CloudWatch Application Insights, you can choose to have the system create OpsItems in OpsCenter when problems are detected. For information, see [Setting Up Your Application](#) in the *Amazon CloudWatch User Guide*.

## Creating OpsItems manually

This section includes information about how to manually create OpsItems in AWS Systems Manager OpsCenter.

### Before You Begin

When you manually create an OpsItem, you can specify an Amazon Resource Name (ARN) for an impacted resource. If you specify an ARN, then OpsCenter automatically creates a deep link to detailed information about the resource. For example, if you specify the ARN of an impacted Amazon 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 doesn't 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 \(p. 224\)](#).

### Topics

- [Creating OpsItems by using the console \(p. 205\)](#)
- [Creating OpsItems by using the AWS CLI \(p. 206\)](#)
- [Creating OpsItems by using AWS Tools for Windows PowerShell \(p. 209\)](#)

## Creating OpsItems by using the console

The following procedure describes how to create an OpsItem by using the Systems Manager console.

### To manually create an OpsItem by using the console

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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. (Optional) For **Priority**, choose the priority level.
7. (Optional) For **Severity**, choose the severity level.
8. (Optional) For **Category**, choose a category.
9. For **Description**, enter information about this OpsItem including (if applicable) steps for reproducing the issue.
10. For **Deduplication string**, enter words the system should use to check for duplicate OpsItems. For more information about deduplication strings, see [Reducing duplicate OpsItems \(p. 214\)](#).
11. (Optional) For **Notifications**, specify the Amazon 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.
12. (Optional) Under **Related resources**, choose **Add** to specify the ID or an Amazon Resource Name (ARN) of the impacted resource and any related resources.
13. Choose **Create OpsItem**.

If successful, the OpsItem opens. For information about how to configure the options in an OpsItem, see [Working with OpsItems \(p. 209\)](#).

## Creating OpsItems by using the AWS CLI

The following procedure describes how to create an OpsItem by using the AWS Command Line Interface (AWS CLI).

### To manually create an OpsItem by using the AWS CLI

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Open the AWS Command Line Interface (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="SNS_ARN_in_same_Region" --tags "Key=key_name,Value=a_value"
```

Here are some examples.

#### Linux management portal macOS

```
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:TestUser1" --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":"[{\\"arn\": \\"arn:aws:dynamodb:us-west-2:12345678:table/OpsItems\"}]","Type":"SearchableString"}}' --notifications Arn="arn:aws:sns:us-west-2:12345678:TestUser"
```

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:useast-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.

```
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:

```
{
 "key_name": {
 "Type": "SearchableString",
 "Value": "Up to 20 KB of data"
 }
}
```

Here is an example.

```
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 in the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

The system returns information like the following.

```
{
 "OpsItemId": "oi-1a2b3c4d5e6f"
}
```

- Run the following command to view details about the OpsItem you created.

```
aws ssm get-ops-item --ops-item-id ID
```

The system returns information like the following.

```
{
 "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,
 "RelatedOpsItems": [],
 "Status": "Open",
 "OpsItemId": "oi-1a2b3c4d5e6f",
 "Title": "EC2 instance disk full",
 "Source": "ec2",
 "OperationalData": {
 "EC2": {
 "Value": "12345",
 "Type": "SearchableString"
 }
 }
 }
}
```

- Run the following command to update the OpsItem. This command changes the status from `Open` (the default) to `InProgress`.

```
aws ssm update-ops-item --ops-item-id ID --status InProgress
```

The command has no output.

- Run the following command again to verify that the status changed to `InProgress`

```
aws ssm get-ops-item --ops-item-id ID
```

## Creating OpsItems by using AWS Tools for Windows PowerShell

1. Open AWS Tools for Windows PowerShell and run the following command to specify your credentials.

```
Set-AWSCredentials -AccessKey key-name -SecretKey key-name
```

2. Run the following command to set the region for your PowerShell session.

```
Set-DefaultAWSRegion -Region Region
```

3. Run the following command to create a new OpsItem. This command specifies a Systems Manager Automation runbook for remediating this OpsItem.

```
$opsItem = New-Object Amazon.SimpleSystemsManagement.Model.OpsItemDataValue
$opsItem.Type = [Amazon.SimpleSystemsManagement.OpsItemDataType]::SearchableString
$opsItem.Value = '[{"automationId": "'runbook_name'", "automationType": "AWS::SSM::Automation"}]'
$newHash = @(" /aws/
automations"=[Amazon.SimpleSystemsManagement.Model.OpsItemDataValue]$opsItem}
New-SMOPsItem -Title "title" -Description "description" -Priority priority_number -
Source AWS_service -OperationalData $newHash
```

If successful, the command outputs the ID of the new OpsItem.

The following example specifies the ARN of an impaired Amazon EC2 instance.

```
$opsItem = New-Object Amazon.SimpleSystemsManagement.Model.OpsItemDataValue
$opsItem.Type = [Amazon.SimpleSystemsManagement.OpsItemDataType]::SearchableString
$opsItem.Value = '[{"arn": "'arn:aws:ec2:us-east-1:123456789012:instance/i-1234567890abcdef0"}]'
$newHash = @(" /aws/
resources"=[Amazon.SimpleSystemsManagement.Model.OpsItemDataValue]$opsItem}
New-SMOPsItem -Title "EC2 instance disk full still" -Description "Log clean up may have failed which caused the disk to be full" -Priority 2 -Source ec2 -OperationalData $newHash
```

## Working with OpsItems

This section describes how to configure the options available in an AWS Systems Manager OpsItem. For information about creating OpsItems, see [Creating OpsItems \(p. 195\)](#).

### Topics

- [Working with related resources \(p. 209\)](#)
- [Viewing other OpsItems for a specific resource \(p. 211\)](#)
- [Editing OpsItems \(p. 211\)](#)
- [Working with similar and related OpsItems \(p. 212\)](#)
- [Working with operational data \(p. 213\)](#)

## Working with related resources

A *related resource* is the impacted AWS resource that needs to be investigated or that initiated an Amazon EventBridge event. Each OpsItem has a **Related resources** section. If EventBridge 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, AWS Systems Manager 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 Amazon 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 \(p. 224\)](#).

**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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.
3. Choose the **OpsItems** tab.
4. Choose an OpsItem ID.

ID	Title	Status	Source
oi-a80f1dbb4464	EC2 instance stopped	Open	EC2
oi-0cdb512b47ed	EC2 instance terminated	Open	EC2
oi-06f350858b55	EC2 instance terminated	Open	EC2

5. To view information about the impacted resource, choose the **Related resources details** tab.

The screenshot shows the 'Related resource details' tab for an EC2 instance terminated. The tab title is 'EC2 instance terminated' with an 'Open' button. Below the title, there are tabs for 'Overview' and 'Related resource details', with 'Related resource details' being the active tab. A dropdown menu shows 'i-05d918a'. Below the dropdown are buttons for 'Expand all', 'Open session', 'Run automation', and 'View resource in original console'. A section titled 'CloudWatch Metrics' is expanded, showing three metrics: 'CPU Utilization (Percent)', 'Network In (Bytes)', and 'Network Out (Bytes)'. Each metric has a small graph below it.

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 type**, choose a resource from the list.
9. For **Resource ID**, enter either the ID or the Amazon Resource Name (ARN). The type of information you choose depends on the resource you chose in the previous step.

## Viewing other OpsItems for a specific resource

To help you investigate issues and provide context for a problem, you can view a list of OpsItems for a specific AWS resource. The list displays the status, severity, and title of each OpsItem. The list also includes deep links to each OpsItem so you can quickly open them to view more information.

### To view a list of OpsItems for a specific resource

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.
3. Choose an OpsItem ID to open the details page.
4. Choose the **Related resource details** tab.
5. Expand **Other OpsItems for this resource**.

## Editing OpsItems

The **OpsItem details** section includes information about an OpsItem, including the description, title, source, OpsItem ID, and the status, to name a few. You can edit OpsItems individually or you can select multiple OpsItems and edit one of the following fields: **Status**, **Priority**, **Severity**, **Category**.

For OpsItems that were created automatically, Amazon EventBridge 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:

Status	Details
<b>Open</b>	Active in the system, but <i>not</i> being worked on by an engineer.
<b>In progress</b>	Active in the system and being worked on by an engineer.
<b>Resolved</b>	Not active in the system, but available in Search and when using the <b>Resolved</b> filter on the OpsItem <b>Overview</b> page. You can edit a resolved OpsItem to change the status to <b>Open</b> or <b>In progress</b> .

You can view reports about OpsItem statuses on the **Summary** tab. For more information, see [Viewing OpsCenter summary reports \(p. 224\)](#).

### 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 Amazon 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 might find it helpful to create different ARNs for notifications about different types of AWS resources or different environments. The Amazon SNS topic must exist in the

same AWS Region as the OpsItems. If they're in different Regions, the system returns an error. For more information, see the [Amazon Simple Notification Service Developer Guide](#).

**Important**

The Amazon SNS topic must exist in the same AWS Region as the OpsItem. If they're in different Regions, the system returns an error.

**To edit OpsItem details**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.
3. Choose an OpsItem ID to open the details page or choose multiple OpsItems. If you choose multiple OpsItems, you can only edit the status, priority, severity, or category. If you edit multiple OpsItems, OpsCenter updates and saves your changes as soon as you choose the new status, priority, severity, or category.
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're finished, choose **Save**.

## Working with similar and related OpsItems

The **Similar OpsItems** and **Related OpsItems** features are designed to help you investigate operations issues while providing context about the scope of an issue.

The **Similar OpsItems** feature is a system-generated list of OpsItems that might 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.

ID	Status	Title	Source
oi-a80f1dbb4464	Open	EC2 instance stopped	EC2
oi-0cdb512b47ed	Open	EC2 instance terminated	EC2
oi-06f350858b55	Open	EC2 instance terminated	EC2

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.

ID	Status	Title	Source
oi-0cdb512b47ed	Open	EC2 instance terminated	EC2
oi-06f350858b55	Open	EC2 instance terminated	EC2

To help you organize OpsItems, you can associate one OpsItem with another so that it is displayed in the **Related OpsItems** section.

**To add a related OpsItem**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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`.

### Operational data

Enter one or more key names and values. Ops Center supports searching and filtering OpsItems by using key names and values.

Key	Value	Searchable
event-time	2019-06-04T00:33:35Z	<input checked="" type="checkbox"/> Searchable
instance-state	stopped	<input checked="" type="checkbox"/> Searchable
Log data	<pre>6093] ata1: PATA max MWDMA2 cmd 0x1f0 ctl 0x3f6 bmdma 0xc100 irq 14 [ 1.981012] ata2: PATA max MWDMA2</pre>	<input checked="" type="checkbox"/> Searchable

**Add item**

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 operation) can view and search on the specified data. Operational data that isn't searchable is only viewable by users who have access to the OpsItem (as provided by the [GetOpsItem](#) API operation).

### To add operational data to an OpsItem

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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

You can enable Amazon CloudWatch or Amazon EventBridge to automatically create OpsItems for alarms and events. As a result, OpsCenter, a capability of AWS Systems Manager, can potentially receive dozens of duplicate OpsItems for a single source. This section includes the following topics to help you quickly view and reduce the number of duplicate OpsItems created by the system:

**Topics**

- [Working with operational insights \(p. 214\)](#)
- [Working with deduplication strings \(p. 216\)](#)

## Working with operational insights

OpsCenter, a capability of AWS Systems Manager, automatically analyzes OpsItems in your account and generates two types of *insights* in the **Operational insights** section. The **Duplicate OpsItems** field displays a count of insights when eight or more OpsItems have the same title for the same resource. The **Sources generating most OpsItems** field displays a count of insights when more than 50 OpsItems have the same title. If you choose an insight, OpsCenter displays information about the affected OpsItems and resources. The following screenshot shows an example with the details of a duplicate OpsItem insight.

## Duplicate OpsItems: 1122334455

### Insight details

Insight type	Status
Duplicate OpsItems	Open
Affected OpsItems	Date created
100 <a href="#">View</a>	14 Aug 2020 20:00:00 GMT
Affected resources	Last updated
i-06bd38270	5 Sep 2020 20:00:00 GMT
Description	
Multiple unresolved OpsItems have the same title 'EC2 Instance Launch Unsuccessful' and involve the same resource 'i-06bd38270'	

### Recommended runbooks (1)

Document name	Description	Execution ID	Start time
	Bulk resolve all unresolved OpsItems with the title 'EC2 Instance Launch Unsuccessful'		

#### Note

Operational insights are disabled by default. You must enable this feature, as described in this topic.

## Viewing insights and resolving duplicate OpsItems

This section describes how to enable and view operational insights. This section also describes how to resolve duplicate OpsItems and reduce the number of OpsItems created by a source. Before you begin, be aware of the following important details.

- If you enable operational insights, OpsCenter creates OpsItems of type insight. OpsCenter limits the number of insights the system creates, but your AWS account is charged for insight OpsItems created by the system. For more information, see [AWS Systems Manager Pricing](#).
- If you enable operational insights, Systems Manager creates an AWS Identity and Access Management (IAM) service-linked role called `AWSServiceRoleForAmazonSSM_OpsInsights`. A service-linked role is a unique type of IAM role that is linked directly to Systems Manager. Service-linked roles are predefined and include all the permissions the service requires to call other AWS services on your behalf. For more information about the `AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role, [Using roles to create operational insight OpsItems in Systems Manager OpsCenter: AWSServiceRoleForAmazonSSM\\_OpsInsights \(p. 1418\)](#).
- OpsCenter periodically refreshes insights using a batch process. This means the list of insights displayed in OpsCenter can be out of sync.
- OpsCenter has a limit of 25 insights. The system stops creating new insights when it reaches this limit.
- To resolve insights, you must first resolve all OpsItems associated with an insight. You can use the `AWS-BulkResolveOpsItemsForInsight` runbook to resolve OpsItems associated with an insight. After the runbook finishes processing and all associated OpsItems are resolved, the system immediately resolves the associated insight.

## Viewing OpsCenter operational insights

Use the following procedure to view operational insights about duplicate OpsItems in Systems Manager OpsCenter.

### To view operational insights

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.
3. On the **Overview** tab, scroll down to **Operational insights**.
4. Choose **Enable**.
5. To view a filtered list of insights, choose the link beside **Duplicate OpsItems** or the link beside **Sources generating most OpsItems**. To view all insights, choose **View all operational insights**.
6. Choose an insight ID to view more information.

## Resolving duplicate OpsItems based on insights

Systems Manager provides the following automation runbooks to help you resolve duplicate OpsItems and reduce the number of OpsItems created by a source.

- The `AWS-BulkResolveOpsItems` runbook resolves OpsItems that match a specified filter.
- The `AWS-AddOpsItemDedupStringToEventBridgeRule` runbook adds a deduplication (dedup) string for all OpsItem targets associated with a given Amazon EventBridge rule. This document doesn't add a dedup string if a rule already has one.
- The `AWS-DisableEventBridgeRule` disables a rule in EventBridge. If a rule is generating dozens or hundreds of OpsItems, you might find it useful to simply disable the rule.

To run one of these runbooks, open an insight, choose the runbook, and then choose **Execute**.

## Disabling operational insights

You can disable operational insights on the OpsCenter **Configure sources** page. When you disable this feature, the system stops creating new insights and stops displaying insights in the console. Any active insights remain unchanged in the system, though you won't see them displayed in the console. If you enable this feature again, the system displays any previously unresolved insights and starts creating new insights. Use the following procedure to disable operational insights.

### To disable operational insights

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.
3. Choose the **OpsItems** tab.
4. Choose **Configure sources**.
5. In the **Operational insights** section, choose **Edit** and then toggle the **Disable** option.
6. Choose **Save**.

## Working with deduplication strings

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

operation is called. When creating the OpsItem, Systems Manager creates and stores a hash based on the deduplication string and the resource that initiated 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 aren't case sensitive. If the system finds a matching hash based on a deduplication string in an incoming OpsItem, regardless of the deduplication string 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/InProgress, 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.
- If no deduplication string is specified for an incoming OpsItem, then the OpsItem is always created.

## Configuring deduplication strings

OpsCenter includes the following options for configuring deduplication strings.

- **Edit preconfigured deduplication strings:** Each of the OpsItem default EventBridge rules includes a preconfigured deduplication string. You can edit these deduplication strings in EventBridge.
- **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 Command Line Interface (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.

The screenshot shows the 'OpsItem details' page for an item with ID 'oi-a80f1dbb4464'. The page displays various metadata fields: Description (CloudWatch Event Rule SSMOpsCenter-EC2-instance-stopped was triggered. Your EC2 instance has stopped. See below for more details.), OpsItem ID (oi-a80f1dbb4464), Status (Open), Title (EC2 instance stopped), Source (EC2), Created (2019-06-04T00:33:36Z), Last updated (2019-06-04T16:49:48Z), and Notifications. A red box highlights the 'Deduplication string' field, which contains the value 'SSMOpsCenter-EC2-instance-stopped'.

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 EventBridge rule \(p. 218\)](#)
- [Specifying a deduplication string by using the AWS CLI \(p. 218\)](#)

**Note**

For information about entering deduplication strings when you manually create an OpsItem in the console, see [Creating OpsItems manually \(p. 205\)](#).

### Editing a deduplication string in an OpsCenter default EventBridge rule

Use the following procedure to specify a deduplication string for an EventBridge rule that targets OpsCenter.

#### To edit a deduplication string in an OpsItem default EventBridge rule

1. Sign in to the AWS Management Console and open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**.
3. Choose a rule, and then choose **Edit**.
4. In the **Select targets** section, expand **Configure input**. In the lower **Input transformer** field, locate the "operationalData": { "/aws/dedup" JSON entry and the deduplication strings that you want to edit.

The deduplication string entry in EventBridge 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 **Update** 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 & macOS

```
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"
```

#### Windows

```
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"
```

## Working with Incident Manager incidents in OpsCenter

This topic describes how to create an AWS Incident Manager incident from an existing OpsItem. An *incident* is any unplanned interruption or reduction in quality of services. Incident Manager, a capability of AWS Systems Manager, provides an incident management console that helps you mitigate and recover from incidents affecting your AWS hosted applications.

Incident Manager increases incident resolution by notifying responders of how AWS resources have been impacted, highlighting relevant troubleshooting data, and providing collaboration tools to get services back up and running. To achieve the primary goal of reducing the time-to-resolution of critical incidents, Incident Manager automates response plans and allows responder team escalation. For more information, see the [AWS Systems Manager Incident Manager User Guide](#).

After an incident is resolved, post incident analysis guides you through identifying improvements to your incident response and recommending action items to address the findings. With high severity operational issues such as incidents, creating an OpsItem in OpsCenter provides operators with a complete view of the incident, including analysis and action items. OpsCenter is a capability of Systems Manager. This comprehensive view improves time-to-resolution and helps mitigate similar issues in the future.

### How it works

After you set up and configure Incident Manager, the system integrates with OpsCenter in the following ways:

1. After an incident is created in Incident Manager, the system creates an OpsItem in OpsCenter (if an OpsItem doesn't already exist). The incident is added as a related item to the OpsItem. This first OpsItem is known as the *parent* OpsItem. You can also manually create an incident from an OpsItem. After you create an incident from an OpsItem, the OpsItem is promoted to a parent OpsItem.

OpsItems that include incidents have two additional tabs beyond the **Overview** and **Related resource details** tabs displayed for standard OpsItems. OpsItems that include incidents display related incidents, OpsItems, analyses, and action items on the **Associated items** tab. The **Timeline** tab displays a chronological history of related incidents and analyses for the parent OpsItem.

2. If an incident grows in scale and scope, you can add additional incidents to it.
3. After an incident is closed, you can create an analysis from the incident in Incident Manager. An analysis can help you define improvement processes for mitigating similar issues in the future. The system automatically updates the incident in OpsCenter with an analysis. If an analysis includes action items, the system creates additional OpsItems beneath the analysis. These additional OpsItems are of type **Action Item**.

### Before you begin

You must set up and configure response plans in Incident Manager. A *response plan* defines how to escalate an incident to first responders and what actions those responders should take. For more information, see [Response plans](#).

## Create an incident for an OpsItem

Use the following procedure to manually create an incident for an OpsItem.

### To manually create an incident for an OpsItem

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.
3. If Incident Manager created an OpsItem for you, choose it and go to step 5. If not, choose **Create OpsItem** and complete the form. If you don't see this button, choose the **OpsItems** tab, and then choose **Create OpsItem**.
4. If you created a new OpsItem, open it.
5. Choose **Start Incident**.
6. For **Response plan**, choose the Incident Manager response plan you want to assign to this incident.
7. (Optional) For **Title**, enter a descriptive name to help other team members understand the nature of the incident. If you don't enter a new title, OpsCenter creates the OpsItem and the corresponding incident in Incident Manager using the title in the response plan.
8. (Optional) For **Incident impact**, choose an impact level for this incident. If you don't choose an impact level, OpsCenter creates the OpsItem and the corresponding incident in Incident Manager using the impact level in the response plan.
9. Choose **Start**.

## 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 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 (Amazon 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 EventBridge, 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 Amazon EC2 instances.
- Reset access for an Amazon EC2 instance.

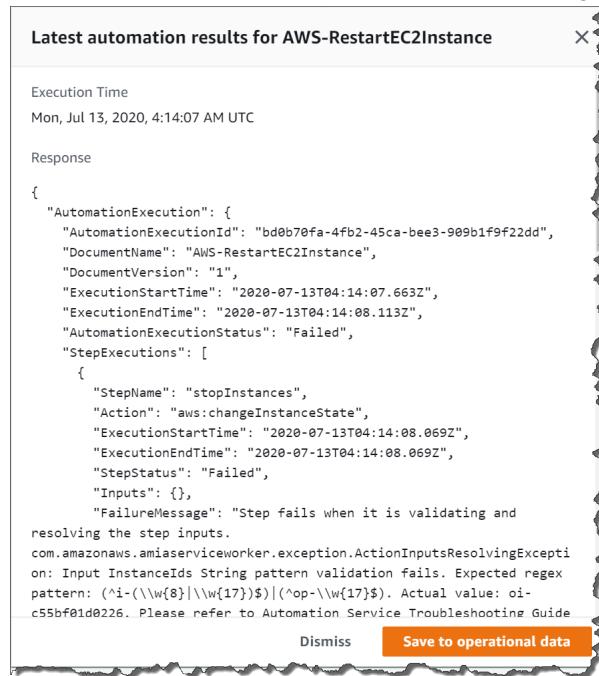
Each OpsItem in the AWS Management Console includes a **Runbooks** section.

## Automation runbook features in OpsCenter

The following list describes some of the features available to help you run Automation runbooks and remediate issues.

- When you choose an AWS resource that generated an OpsItem, OpsCenter displays a list of Automation runbooks that you can run on that resource.

- When you choose an Automation runbook from the list, OpsCenter prepopulates some of the fields required to run the document.
- OpsCenter keeps a 30-day record of Automation runbooks run for a specific OpsItem.
- In the **Status and results** column, you can choose a status to view important details about that run, such as the reason why an Automation failed and which step of the Automation runbook was running when the failure occurred, as shown in the following example.



The screenshot shows a modal window titled "Latest automation results for AWS-RestartEC2Instance". It displays the following information:

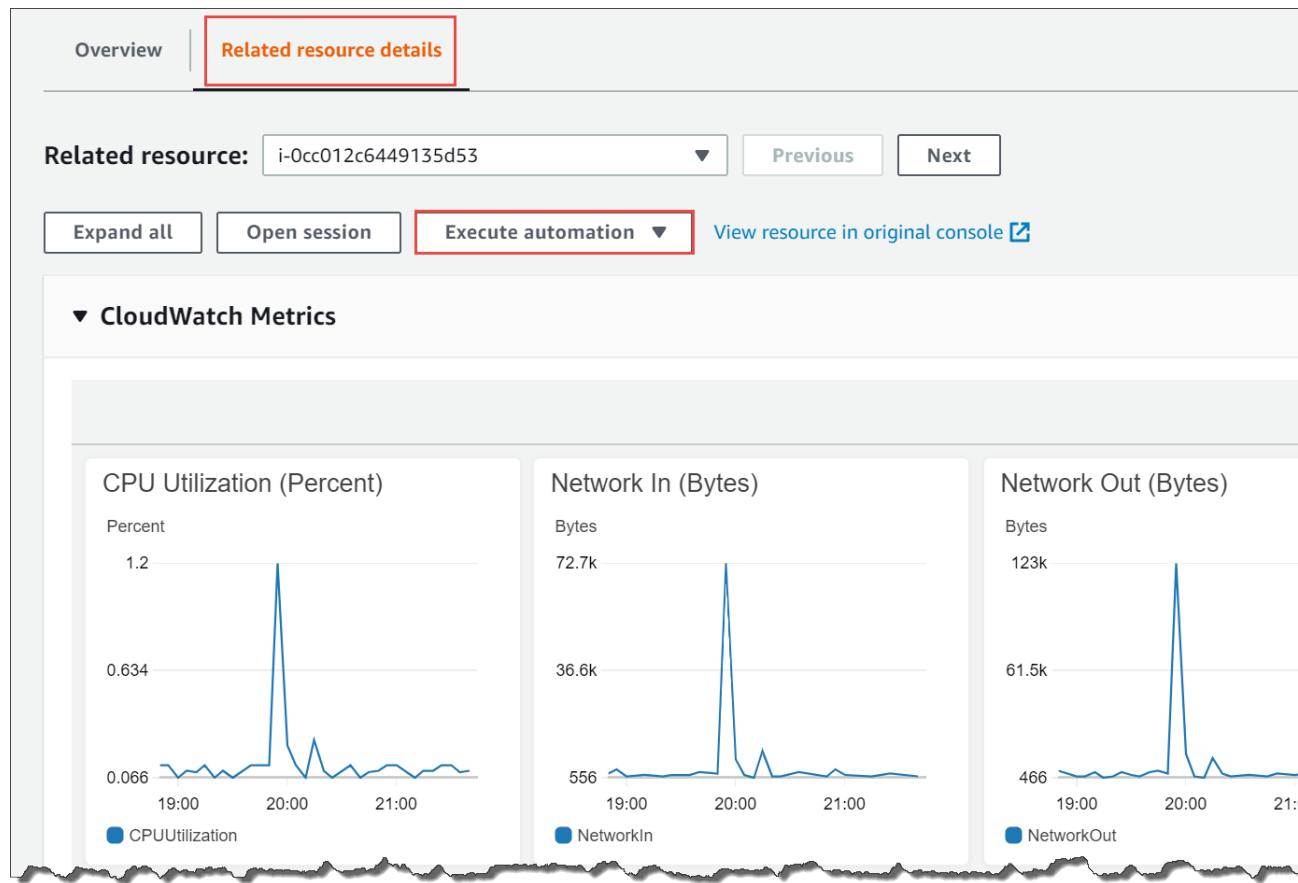
**Execution Time**  
Mon, Jul 13, 2020, 4:14:07 AM UTC

**Response**

```
{
 "AutomationExecution": {
 "AutomationExecutionId": "bd0b70fa-4fb2-45ca-bee3-909b1f9f22dd",
 "DocumentName": "AWS-RestartEC2Instance",
 "DocumentVersion": "1",
 "ExecutionStartTime": "2020-07-13T04:14:07.663Z",
 "ExecutionEndTime": "2020-07-13T04:14:08.113Z",
 "AutomationExecutionStatus": "Failed",
 "StepExecutions": [
 {
 "StepName": "stopInstances",
 "Action": "aws:changeInstanceState",
 "ExecutionStartTime": "2020-07-13T04:14:08.069Z",
 "ExecutionEndTime": "2020-07-13T04:14:08.069Z",
 "StepStatus": "Failed",
 "Inputs": {},
 "FailureMessage": "Step fails when it is validating and
resolving the step inputs.
com.amazonaws.amibserviceworker.exception.ActionInputsResolvingException: Input InstanceId String pattern validation fails. Expected regex
pattern: (^i-(\\w{8}\\\\\\w{17})$)|(^op-\\w{17}$). Actual value: oi-
c55bf01d0226. Please refer to Automation Service Troubleshooting Guide"
 }
]
 }
}
```

**Buttons**: Dismiss, Save to operational data

- The **Related resource details** page for a selected OpsItem includes the **Run automation** list. The list allows you to choose recent or resource-specific Automation runbooks that you can run to remediate issues. This page also includes helpful data providers, including Amazon CloudWatch metrics and alarms, AWS CloudTrail logs, and details from AWS Config, to name a few.



- You can view information about an Automation runbook by either choosing its name in the console or by using the [Systems Manager Automation runbook reference \(p. 604\)](#).

## Using a runbook to remediate an OpsItem issue

When you run a Systems Manager Automation 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 running the runbook.

### Execute automation document

Simple execution  
Execute on targets.

Rate control  
Execute safely on multiple targets by defining concurrency and error thresholds.

Multi-account and Regions  
Execute in multiple Regions.

The following procedure describes how to run a simple version of a runbook. For information about running an **Advanced configuration** runbook, see [Working with automations \(p. 407\)](#).

#### Before You Begin

Before you run an automation document (runbook) to remediate an OpsItem issue, do the following:

- Verify that you have permission to run Systems Manager Automation runbooks. For more information, see [Setting up Automation \(p. 401\)](#).
- Collect resource-specific ID information for the automation that you want to run. For example, if you want to run an automation that restarts an EC2 instance, then you must specify the ID of the instance to restart.

### To run an Automation runbook to remediate an OpsItem issue

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.
3. Choose the OpsItem ID to open the details page.

ID	Title	Status	Source
oi-a80f1dbb4464	EC2 instance stopped	⌚ Open	EC2
oi-0cdb512b47ed	EC2 instance terminated	⌚ Open	EC2
oi-06f350858b55	EC2 instance terminated	⌚ Open	EC2

4. Scroll to the **Runbooks** section.
5. Use the Runbooks Search bar or the numbers in the upper right to find the Automation runbook that you want to run.
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.

## Working with associated runbooks

After you run an Automation runbook from an OpsItem, the runbook is automatically associated with the related resource of that OpsItem for future reference. Associated runbooks are ranked higher than others in the **Runbooks** list.

Use the following procedure to run an Automation runbook that has already been associated with a related resource in an OpsItem. For information about adding related resources, see [Working with OpsItems \(p. 209\)](#).

### To run a resource-associated runbook to remediate an OpsItem issue

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.
3. Open the OpsItem.
4. In the **Related resources** section, choose the resource on which you want to run the Automation runbook.
5. Choose **Run automation**, and then choose the associated Automation runbook that you want to run.
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

AWS Systems Manager 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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

AWS Systems Manager 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 allows 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 \(p. 209\)](#).

### Supported resources

Resource name	ARN format
AWS Certificate Manager certificate	arn:aws:acm: <i>region:account-id</i> :certificate/ <i>certificate-id</i>
Amazon EC2 Auto Scaling group	arn:aws:autoscaling: <i>region:account-id</i> :autoScalingGroup: <i>groupid</i> :autoScalingGroupName/ <i>groupfriendlyname</i>
Amazon CloudFront distribution	arn:aws:cloudfront:: <i>account-id</i> :*
AWS CloudFormation stack	arn:aws:cloudformation: <i>region:account-id</i> :stack/ <i>stackname</i> / <i>additionalidentifier</i>
Amazon CloudWatch alarm	arn:aws:cloudwatch: <i>region:account-id</i> :alarm: <i>alarm-name</i>
AWS CloudTrail trail	arn:aws:cloudtrail: <i>region:account-id</i> :trail/ <i>trailname</i>

Resource name	ARN format
AWS CodeBuild project	<code>arn:aws:codebuild:<i>region:account-id:resourcetype/resource</i></code>
AWS CodePipeline	<code>arn:aws:codepipeline:<i>region:account-id:resource-specifier</i></code>
DevOps Guru insight	<code>arn:aws:devops-guru:<i>region:account-id:insight/proactive or reactive/resource-id</i></code>
Amazon DynamoDB table	<code>arn:aws:dynamodb:<i>region:account-id:table tablename</i></code>
Amazon Elastic Compute Cloud (Amazon EC2) customer gateway	<code>arn:aws:ec2:<i>region:account-id:customer-gateway/cgw-id</i></code>
Amazon EC2 elastic IP	<code>arn:aws:ec2:<i>region:account-id:eip/eipalloc-id</i></code>
Amazon EC2 Dedicated Host	<code>arn:aws:ec2:<i>region:account-id:dedicated-host/host-id</i></code>
Amazon EC2 instance	<code>arn:aws:ec2:<i>region:account-id:instance/instance-id</i></code>
Amazon EC2 internet gateway	<code>arn:aws:ec2:<i>region:account-id:internet-gateway/igw-id</i></code>
Amazon EC2 network access control list (ACL)	<code>arn:aws:ec2:<i>region:account-id:network-acl/nacl-id</i></code>
Amazon EC2 network interface	<code>arn:aws:ec2:<i>region:account-id:network-interface/eni-id</i></code>
Amazon EC2 route table	<code>arn:aws:ec2:<i>region:account-id:route-table/route-table-id</i></code>
Amazon EC2 security group	<code>arn:aws:ec2:<i>region:account-id:security-group/security-group-id</i></code>
Amazon EC2 subnet	<code>arn:aws:ec2:<i>region:account-id:subnet/subnet-id</i></code>

Resource name	ARN format
Amazon EC2 volume	arn:aws:ec2: <i>region:account-id</i> :volume/ <i>volume-id</i>
Amazon EC2 VPC	arn:aws:ec2: <i>region:account-id</i> :vpc/ <i>vpc-id</i>
Amazon EC2 VPN connection	arn:aws:ec2: <i>region:account-id</i> :vpn-connection/ <i>vpn-id</i>
Amazon EC2 VPN gateway	arn:aws:ec2: <i>region:account-id</i> :vpn-gateway/ <i>vgw-id</i>
AWS Elastic Beanstalk application	arn:aws:elasticbeanstalk: <i>region:account-id</i> :application/ <i>applicationname</i>
Elastic Load Balancing (Classic Load Balancer)	arn:aws:elasticloadbalancing: <i>region:account-id</i> :loadbalancer/ <i>name</i>
Elastic Load Balancing (Application Load Balancer)	arn:aws:elasticloadbalancing: <i>region:account-id</i> :loadbalancer/app/ <i>load-balancer-name</i> / <i>load-balancer-id</i>
Elastic Load Balancing (Network Load Balancer)	arn:aws:elasticloadbalancing: <i>region:account-id</i> :loadbalancer/net/ <i>load-balancer-name</i> / <i>load-balancer-id</i>
AWS Identity and Access Management (IAM) group	arn:aws:iam:: <i>account-id</i> :group/ <i>group-name</i>
IAM policy	arn:aws:iam:: <i>account-id</i> :policy/ <i>policy-name</i>
IAM role	arn:aws:iam:: <i>account-id</i> :role/ <i>role-name</i>
IAM user	arn:aws:iam:: <i>account-id</i> :user/ <i>user-name</i>
AWS Lambda function	arn:aws:lambda: <i>region:account-id</i> :function/ <i>function-name</i>
Amazon Relational Database Service (Amazon RDS) cluster	arn:aws:rds: <i>region:account-id</i> :cluster: <i>db-cluster-name</i>
Amazon RDS database instance	arn:aws:rds: <i>region:account-id</i> :db: <i>db-instance-name</i>

Resource name	ARN format
Amazon RDS subscription	arn:aws:rds: <b>region:account-id:es:subscription-name</b>
Amazon RDS security group	arn:aws:rds: <b>region:account-id:secgrp:security-group-name</b>
Amazon RDS cluster snapshot	arn:aws:rds: <b>region:account-id:cluster-snapshot:cluster-snapshot-name</b>
Amazon RDS subnet group	arn:aws:rds: <b>region:account-id:subgrp:subnet-group-name</b>
Amazon Redshift cluster	arn:aws:redshift: <b>region:account-id:cluster:cluster-name</b>
Amazon Redshift parameter group	arn:aws:redshift: <b>region:account-id:parametergroup:parameter-group-name</b>
Amazon Redshift security group	arn:aws:redshift: <b>region:account-id:securitygroup:security-group-name</b>
Amazon Redshift cluster snapshot	arn:aws:redshift: <b>region:account-id:snapshot:cluster-name/snapshot-name</b>
Amazon Redshift subnet group	arn:aws:redshift: <b>region:account-id:subnetgroup:subnet-group-name</b>
Amazon Simple Storage Service (Amazon S3) bucket	arn:aws:s3::: <b>bucket_name</b>
AWS Config recording of AWS Systems Manager managed node inventory	arn:aws:ssm: <b>region:account-id:managed-instance-inventory/node_id</b>
Systems Manager State Manager association	arn:aws:ssm: <b>region:account-id:association/association_ID</b>

## Receiving findings from AWS Security Hub in OpsCenter

[AWS Security Hub](#) provides you with a comprehensive view of your security state in AWS and helps you to check your environment against security industry standards and best practices. Security Hub collects security data from across AWS accounts, services, and supported third-party partner products and helps you to analyze your security trends and identify the highest priority security issues.

The AWS Systems Manager OpsCenter integration with Security Hub allows you to receive findings from Security Hub in OpsCenter. Security Hub findings provide security information that you can use in OpsCenter to aggregate and take action on your security, performance, and operational issues in Systems Manager.

You can automatically create operational issues (OpsItems) in OpsCenter for diagnosis and remediation of critical and high severity findings. To help you with the diagnosis, the OpsItem includes relevant information, such as AWS resource ID and type, and findings details. You can also use Systems Manager Automation runbooks within OpsCenter to run predefined workflows to help remediate common security issues with AWS resources.

OpsCenter has bidirectional integration with Security Hub. When you make updates to OpsItem status and severity fields related to a security finding, those changes are automatically sent to Security Hub to ensure you always see the latest and correct information. OpsCenter is also integrated with AWS Systems Manager Explorer (Explorer). In Explorer, you can view a widget that provides summary of all Security Hub findings based on severity.

## How OpsCenter receives findings from Security Hub

In Security Hub, security issues are tracked as findings. Some findings come from issues that are detected by other AWS services or by third-party partners. Security Hub also has a set of rules that it uses to detect security issues and generate findings.

AWS Systems Manager OpsCenter is one of the AWS services that receives findings from Security Hub.

### Mechanism for receiving the findings from Security Hub

To receive the findings from Security Hub, OpsCenter takes advantage of the Security Hub integration with Amazon EventBridge. Security Hub sends findings to EventBridge, which uses an event rule to send the findings to OpsCenter.

### Types of findings that OpsCenter receives

OpsItems are automatically created for Critical and High severity findings. You can configure OpsCenter to display Medium and Low severity findings as described later in [Turning on and configuring the integration \(p. 228\)](#). You can configure OpsCenter and Explorer to create OpsItems for all findings except Informational severity findings. For more information about the severity of Security Hub findings, see [Severity](#) in the *AWS Security Hub User Guide*.

### How long does it take to receive findings from Security Hub?

When Security Hub creates a new finding, it's usually visible in OpsCenter within seconds.

### Retrying if there is a system outage

Data from Security Hub is retained in OpsCenter for up to 7 days to retry if there is a system outage. Security Hub also refreshes every 12 hours on all Security Hub standard rules.

## Turning on and configuring the integration

To use the integration with Security Hub, you must activate Security Hub. For information on how to turn on Security Hub, see [Setting up Security Hub](#) in the *AWS Security Hub User Guide*.

The following procedure describes how to start receiving and configure Security Hub findings.

### To start receiving Security Hub findings

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.

3. Select **OpsItems**. Then select **Configure sources**.
4. In the **Security Hub findings** section, select **Edit**.
5. Select the **Disabled** slider to allow Security Hub findings to automatically create OpsItems.

Critical and High security findings create OpsItems by default. To also create OpsItems for Medium and Low security findings, select the **Disabled** slider next to **Medium,Low**.

When you have configured Security Hub integration as you want, select **Save**.

## How to view findings from Security Hub

The following procedure describes how to view Security Hub findings.

### To view Security Hub findings through Explorer

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Explorer**.
3. Navigate to the Security Hub findings widget. Select the severity you want to view a detailed list view. The corresponding OpsItem will be displayed.

## How to stop receiving findings

The following procedure describes how to stop receiving Security Hub findings.

### To stop receiving Security Hub findings

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **OpsCenter**.
3. Select **OpsItems**. Then select **Configure sources**.
4. In the **Security Hub findings** section, select **Edit**.
5. Select the **Enable** slider to turn off Security Hub findings from automatically creating OpsItems. Then select **Save**.

## Auditing and logging OpsCenter activity

AWS CloudTrail captures AWS Systems Manager OpsCenter API calls made in the console, the AWS Command Line Interface (AWS CLI), and the 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 \(p. 1442\)](#).

## 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 AWS

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 AWS Regions. You can select the color used for each metric on each graph, so that you can 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 Command Line Interface (AWS CLI), or by using the CloudWatch PutDashboard API. For more information, see [Using Amazon CloudWatch Dashboards](#) in the *Amazon CloudWatch User Guide*.

# AWS Systems Manager Application Management

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

## Topics

- [AWS Systems Manager Application Manager \(p. 231\)](#)
- [AWS AppConfig \(p. 256\)](#)
- [AWS Systems Manager Parameter Store \(p. 256\)](#)

## AWS Systems Manager Application Manager

Application Manager, a capability of AWS Systems Manager, helps DevOps engineers investigate and remediate issues with their AWS resources in the context of their applications and clusters. Application Manager aggregates operations information from multiple AWS services and Systems Manager capabilities to a single AWS Management Console.

In Application Manager, an *application* is a logical group of AWS resources that you want to operate as a unit. This logical group can represent different versions of an application, ownership boundaries for operators, or developer environments, to name a few. Application Manager support for container clusters includes both Amazon Elastic Kubernetes Service (Amazon EKS) and Amazon Elastic Container Service (Amazon ECS) clusters.

When you choose **Get started** on the Application Manager home page, Application Manager automatically imports metadata about your resources that were created in other AWS services or Systems Manager capabilities. For applications, Application Manager imports metadata about all of your AWS resources organized into resource groups. Each resource group is listed in the **Custom applications** category as a unique application. Application Manager also automatically imports metadata about resources that were created by AWS CloudFormation, AWS Launch Wizard, Amazon ECS, and Amazon EKS. Application Manager then displays those resources in predefined categories.

For **Applications**, the list includes the following:

- Custom applications
- Launch Wizard
- CloudFormation stacks
- AppRegistry applications

For **Container clusters**, the list includes the following:

- Amazon ECS clusters
- Amazon EKS clusters

After import is complete, you can view operations information about your resources in these predefined categories. Or, if you want to provide more context about a collection of resources, you can manually create an application in Application Manager and move resources or groups of resources into that application. This allows you to view operations information in the context of an application.

After you [set up](#) and configure AWS services and Systems Manager capabilities, Application Manager displays the following types of information about your resources:

- Alarms provided by Amazon CloudWatch
- Compliance information provided by AWS Config and State Manager (a component of Systems Manager)
- Kubernetes cluster information provided by Amazon EKS
- Log data provided by AWS CloudTrail and Amazon CloudWatch Logs
- OpsItems provided by Systems Manager OpsCenter
- Resource details provided by the AWS services that host them.
- Container cluster information provided by Amazon ECS.

To help you remediate issues with components or resources, Application Manager also provides runbooks that you can associate with your applications. To get started with Application Manager, open the [Systems Manager console](#). In the navigation pane, choose **Application Manager**.

## What are the benefits of using Application Manager?

Application Manager reduces the time it takes for DevOps engineers to detect and investigate issues with AWS resources. To do this, Application Manager displays many types of operations information in the context of an application in one console. Application Manager also reduces the time it takes to remediate issues by providing runbooks that perform common remediation tasks on AWS resources.

## What are the features of Application Manager?

Application Manager includes the following features:

- **Import your AWS resources automatically**

During initial setup, you can choose to have Application Manager automatically import and display resources in your AWS account that are based on CloudFormation stacks, AWS Resource Groups, Launch Wizard deployments, AppRegistry applications, and Amazon ECS and Amazon EKS clusters. The system displays these resources in predefined application or cluster categories. Thereafter, whenever new resources of these types are added to your AWS account, Application Manager automatically displays the new resources in the predefined application and cluster categories.

- **Create or edit CloudFormation stacks and templates**

Application Manager helps you provision and manage resources for your applications by integrating with [CloudFormation](#). You can create, edit, and delete AWS CloudFormation templates and stacks in Application Manager. Application Manager also includes a template library where you can clone, create, and store templates. Application Manager and CloudFormation display the same information about the current status of a stack. Templates and template updates are stored in Systems Manager until you provision the stack, at which time the changes are also displayed in CloudFormation.

- **View operational metrics and alarms for an application or cluster**

Application Manager integrates with [Amazon CloudWatch](#) to provide real-time operational metrics and alarms for an application or cluster. You can drill down into your application tree to view alarms at each component level, or view alarms for an individual cluster.

- **View log data for an application**

Application Manager integrates with [Amazon CloudWatch Logs](#) to provide log data in the context of your application without having to leave Systems Manager.

- **View and manage OpsItems for an application or cluster**

Application Manager integrates with [AWS Systems Manager OpsCenter \(p. 180\)](#) to provide a list of operational work items (OpsItems) for your applications and clusters. The list reflects automatically generated and manually created OpsItems. You can view details about the resource that created an OpsItem and the OpsItem status, source, and severity.

- **View resource compliance data for an application or cluster**

Application Manager integrates with [AWS Config](#) to provide compliance and history details about your AWS resources according to rules you specify. Application Manager also integrates with [AWS Systems Manager State Manager \(p. 1034\)](#) to provide compliance information about the state you want to maintain for your Amazon Elastic Compute Cloud (Amazon EC2) instances.

- **View Amazon ECS and Amazon EKS cluster infrastructure information**

Application Manager integrates with [Amazon ECS](#) and [Amazon EKS](#) to provide information about the health of your cluster infrastructures and a component runtime view of the compute, networking, and storage resources in a cluster.

However, you can't manage or view operations information about your Amazon EKS pods or containers in Application Manager. You can only manage and view operations information about the infrastructure that is hosting your Amazon EKS resources.

- **View resource cost details for an application**

Application Manager is integrated with AWS Billing and Cost Management through the [Cost Explorer](#) widget. After you enable Cost Explorer in the Billing and Cost Management console, the Cost Explorer widget in Application Manager shows cost data for a specific non-container application or application component. You can use filters in the widget to view cost data according to different time periods, granularities, and cost types in either a bar or line chart.

- **View detailed resource information in one console**

Choose a resource name listed in Application Manager and view contextual information and operations information about that resource without having to leave Systems Manager.

- **Receive automatic resource updates for applications**

If you make changes to a resource in a service console, and that resource is part of an application in Application Manager, then Systems Manager automatically displays those changes. For example, if you update a stack in the AWS CloudFormation console, and if that stack is part of an Application Manager application, then the stack updates are automatically reflected in Application Manager.

- **Discover Launch Wizard applications automatically**

Application Manager is integrated with [AWS Launch Wizard](#). If you used Launch Wizard to deploy resources for an application, Application Manager can automatically import and display them in a Launch Wizard section.

- **Monitor application resources in Application Manager by using CloudWatch Application Insights**

Application Manager integrates with Amazon CloudWatch Application Insights. Application Insights identifies and sets up key metrics, logs, and alarms across your application resources and technology stack. Application Insights continuously monitors metrics and logs to detect and correlate anomalies and errors. When the system detects errors or anomalies, Application Insights generates CloudWatch Events that you can use to set up notifications or take actions. You can enable and view Application Insights on the [Overview](#) and [Monitoring](#) tabs in Application Manager. For more information about Application Insights, see [What is Amazon CloudWatch Application Insights](#) in the [Amazon CloudWatch User Guide](#).

- **Remediate issues with runbooks**

Application Manager includes predefined Systems Manager runbooks for remediating common issues with AWS resources. You can choose a resource in Application Manager and then choose a runbook that performs a remediation task.

## Is there a charge to use Application Manager?

Application Manager is available at no additional charge.

## What are the resource quotas for Application Manager?

You can view quotas for all Systems Manager capabilities in the [Systems Manager service quotas](#) in the *Amazon Web Services General Reference*. Unless otherwise noted, each quota is Region specific.

### Topics

- [Getting started with Systems Manager Application Manager \(p. 234\)](#)
- [Working with Application Manager \(p. 241\)](#)

## Getting started with Systems Manager Application Manager

Use the information in this section to help you set up and configure Application Manager, a capability of AWS Systems Manager, to display operations information from different AWS services and Systems Manager capabilities. This section also includes information about adding applications and clusters to Application Manager.

### Topics

- [Setting up related services \(p. 234\)](#)
- [Configuring permissions for Systems Manager Application Manager \(p. 236\)](#)
- [Adding applications and clusters to Application Manager \(p. 240\)](#)

## Setting up related services

Application Manager, a capability of AWS Systems Manager, displays resources and information from other AWS services and Systems Manager capabilities. To maximize the amount of operations information displayed in Application Manager, we recommend that you set up and configure these other services or capabilities *before* you use Application Manager.

### Topics

- [Set up tasks for importing resources \(p. 234\)](#)
- [Set up tasks for viewing operations information about resources \(p. 235\)](#)

## Set up tasks for importing resources

The following setup tasks help you view AWS resources in Application Manager. After each of these tasks is completed, Systems Manager can automatically import resources into Application Manager. After your resources are imported, you can create applications in Application Manager and move your imported resources into them. This helps you view operations information in the context of an application.

### (Optional) Organize your AWS resources by using tags

You can assign metadata to your AWS resources in the form of tags. Each tag is a label consisting of a user-defined key and value. Tags can help you manage, identify, organize, search for, and filter

resources. You can create tags to categorize resources by purpose, owner, environment, or other criteria.

**(Optional) Organizes your AWS resources by using AWS Resource Groups**

You can use resource groups to organize your AWS resources. Resource groups make it easier to manage, monitor, and automate tasks on many resources at one time.

Application Manager automatically imports all of your resource groups and lists them in the **Custom applications** category.

**(Optional) Set up and deploy your AWS resources by using AWS CloudFormation**

AWS CloudFormation allows you to create and provision AWS infrastructure deployments predictably and repeatedly. It helps you use AWS services such as Amazon EC2, Amazon Elastic Block Store (Amazon EBS), Amazon Simple Notification Service (Amazon SNS), Elastic Load Balancing, and AWS Auto Scaling. With CloudFormation, you can build reliable, scalable, cost-effective applications in the cloud without worrying about creating and configuring the underlying AWS infrastructure.

Application Manager automatically imports all of your AWS CloudFormation resources and lists them in the **AWS CloudFormation stacks** category. You can create CloudFormation stacks and templates in Application Manager. Stack and template changes are automatically synchronized between Application Manager and CloudFormation. You can also create applications in Application Manager and move stacks into them. This helps you view operations information for resources in your stacks in the context of an application. For pricing information, see [AWS CloudFormation Pricing](#).

**(Optional) Set up and deploy your applications by using AWS Launch Wizard**

Launch Wizard guides you through the process of sizing, configuring, and deploying AWS resources for third-party applications without the need to manually identify and provision individual AWS resources.

Application Manager automatically imports all of your Launch Wizard resources and lists them in the **Launch Wizard** category. For more information about AWS Launch Wizard, see [Getting started with AWS Launch Wizard for SQL Server](#). Launch Wizard is available at no additional charge. You only pay for the AWS resources that you provision to run your solution.

**(Optional) Set up and deploy your containerized applications by using Amazon ECS and Amazon EKS**

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that makes it easy to run, stop, and manage containers on a cluster. Your containers are defined in a task definition that you use to run individual tasks or tasks within a service.

Amazon EKS is a managed service that helps you to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.

Application Manager automatically imports all of your Amazon ECS and Amazon EKS infrastructure resources and lists them on the **Container clusters** tab. However, you can't manage or view operations information about your Amazon EKS pods or containers in Application Manager. You can only manage and view operations information about the infrastructure that is hosting your Amazon EKS resources. For pricing information, see [Amazon ECS Pricing](#) and [Amazon EKS Pricing](#).

## Set up tasks for viewing operations information about resources

The following setup tasks help you view operations information about your AWS resources in Application Manager.

**(Optional) Set up and configure Amazon CloudWatch logs and alarms**

CloudWatch is a monitoring and management service that provides data and actionable insights for AWS, hybrid, and on-premises applications and infrastructure resources. With CloudWatch, you can

collect and access all your performance and operational data in the form of logs and metrics from a single platform. To view CloudWatch logs and alarms for your resources in Application Manager, you must set up and configure CloudWatch. For pricing information, see [CloudWatch Pricing](#).

**Note**

CloudWatch Logs support applies to applications only, not to clusters.

**(Optional) Set up and configure AWS Config**

AWS Config provides a detailed view of the resources associated with your AWS account, including how they're configured, how they're related to one another, and how the configurations and their relationships have changed over time. You can use AWS Config to evaluate the configuration settings of your AWS resources. You do this by creating AWS Config rules, which represent your ideal configuration settings. While AWS Config continually tracks the configuration changes that occur among your resources, it checks whether these changes violate any of the conditions in your rules. If a resource violates a rule, AWS Config flags the resource and the rule as *noncompliant*. Application Manager displays compliance information about AWS Config rules. To view this data in Application Manager, you must set up and configure AWS Config. For pricing information, see [AWS Config Pricing](#).

**(Optional) Create State Manager associations**

You can use Systems Manager State Manager to create a configuration that you assign to your managed nodes. The configuration, called an *association*, defines the state that you want to maintain on your nodes. To view association compliance data in Application Manager, you must configure one or more State Manager associations. State Manager is offered at no additional charge.

**(Optional) Set up and configure OpsCenter**

You can view operational work items (OpsItems) about your resources in Application Manager by using OpsCenter. You can configure Amazon CloudWatch and Amazon EventBridge to automatically send OpsItems to OpsCenter based on alarms and events. You can also enter OpsItems manually. For pricing information, see [AWS Systems Manager Pricing](#).

**(Recommended) Verify runbook permissions**

You can remediate issues with AWS resources from Application Manager by using Systems Manager Automation runbooks. To use this remediation capability, you must configure or verify permissions. For pricing information, see [AWS Systems Manager Pricing](#).

## Configuring permissions for Systems Manager Application Manager

You can use all features of Application Manager, a capability of AWS Systems Manager, if your AWS Identity and Access Management (IAM) user, group, or role has access to the API operations listed in this topic. The API operations are separated into two tables to help you understand the different functions they perform.

The following table lists the API operations that Systems Manager calls if you choose a resource in Application Manager because you want to view the resource details. For example, if Application Manager lists an Amazon EC2 Auto Scaling group, and if you choose that group to view its details, then Systems Manager calls the `autoscaling:DescribeAutoScalingGroups` API operations. If you don't have any Auto Scaling groups in your account, this API operation isn't called from Application Manager.

Resource details only
acm:DescribeCertificate acm>ListTagsForCertificate autoscaling:DescribeAutoScalingGroups

#### Resource details only

```
cloudfront:GetDistribution
cloudfront>ListTagsForResource
cloudtrail:DescribeTrails
cloudtrail>ListTags
cloudtrail:LookupEvents
codebuild:BatchGetProjects
codepipeline:GetPipeline
codepipeline>ListTagsForResource
dynamodb:DescribeTable
dynamodb>ListTagsOfResource
ec2:DescribeAddresses
ec2:DescribeCustomerGateways
ec2:DescribeHosts
ec2:DescribeInternetGateways
ec2:DescribeNetworkAcls
ec2:DescribeNetworkInterfaces
ec2:DescribeRouteTables
ec2:DescribeSecurityGroups
ec2:DescribeSubnets
ec2:DescribeVolumes
ec2:DescribeVpcs
ec2:DescribeVpnConnections
ec2:DescribeVpnGateways
elasticbeanstalk:DescribeApplications
elasticbeanstalk>ListTagsForResource
elasticloadbalancing:DescribeInstanceHealth
elasticloadbalancing:DescribeListeners
elasticloadbalancing:DescribeLoadBalancers
elasticloadbalancing:DescribeTags
iam:GetGroup
iam:GetPolicy
iam:GetRole
iam: GetUser
lambda:GetFunction
rds:DescribeDBClusters
rds:DescribeDBInstances
rds:DescribeDBSecurityGroups
rds:DescribeDBSnapshots
rds:DescribeDBSubnetGroups
rds:DescribeEventSubscriptions
rds>ListTagsForResource
redshift:DescribeClusterParameters
redshift:DescribeClusterSecurityGroups
redshift:DescribeClusterSnapshots
redshift:DescribeClusterSubnetGroups
redshift:DescribeClusters
s3:GetBucketTagging
```

The following table lists the API operations that Systems Manager uses to make changes to applications and resources listed in Application Manager or to view operations information for a selected application or resource.

#### Application actions and details

```
cloudformation:DescribeStacks
cloudwatch:DescribeAlarms
cloudwatch:DescribeInsightRules
cloudwatch>ListMetrics
cloudwatch>ListTagsForResource
config:DescribeComplianceByResource
```

#### Application actions and details

```
config:DescribeRemediationConfigurations
config:GetComplianceDetailsByResource
config:GetResourceConfigHistory
config:StartConfigRulesEvaluation
ec2:DescribeInstances
eks:DescribeCluster
eks>ListClusters
eks>ListFargateProfiles
eks>ListNodegroups
eks:TagResource
ecs:ListClusters
ecs:DescribeClusters
ecs>ListContainerInstances
ecs:DescribeContainerInstances
ecs:DescribeCapacityProviders
ecs:TagResource
resource-groups>CreateGroup
resource-groups>DeleteGroup
resource-groups:ListGroupResources
resource-groups:ListGroups
resource-groups:Tag
resource-groups:Untag
ssm>CreateOpsMetadata
ssm>DeleteOpsMetadata
ssm:GetOpsSummary
ssm:GetOpsMetadata
ssm:UpdateServiceSetting
ssm:GetServiceSetting
ssm>ListOpsMetadata
ssm:UpdateOpsItem
tag:GetTagKeys
tag:GetTagValues
```

## Configuring permissions

To configure Application Manager permissions for an IAM user, group, or role, create an IAM policy using the following example. This policy example includes all API operations used by Application Manager.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "acm:DescribeCertificate",
 "acm>ListTagsForCertificate",
 "autoscaling:DescribeAutoScalingGroups",
 "cloudfront:GetDistribution",
 "cloudfront>ListTagsForResource",
 "cloudtrail:DescribeTrails",
 "cloudtrail>ListTags",
 "cloudtrail:LookupEvents",
 "codebuild:BatchGetProjects",
 "codepipeline:GetPipeline",
 "codepipeline>ListTagsForResource",
 "dynamodb:DescribeTable",
 "dynamodb>ListTagsOfResource",
```

```
"ec2:DescribeAddresses",
"ec2:DescribeCustomerGateways",
"ec2:DescribeHosts",
"ec2:DescribeInternetGateways",
"ec2:DescribeNetworkAcls",
"ec2:DescribeNetworkInterfaces",
"ec2:DescribeRouteTables",
"ec2:DescribeSecurityGroups",
"ec2:DescribeSubnets",
"ec2:DescribeVolumes",
"ec2:DescribeVpcs",
"ec2:DescribeVpnConnections",
"ec2:DescribeVpnGateways",
"ecs>ListClusters",
"ecs:DescribeClusters",
"ecs>ListContainerInstances",
"ecs:DescribeContainerInstances",
"ecs:DescribeCapacityProviders",
"ecs:TagResource",
"elasticbeanstalk:DescribeApplications",
"elasticbeanstalk>ListTagsForResource",
"elasticloadbalancing:DescribeInstanceHealth",
"elasticloadbalancing:DescribeListeners",
"elasticloadbalancing:DescribeLoadBalancers",
"elasticloadbalancing:DescribeTags",
"iam:GetGroup",
"iam:GetPolicy",
"iam:GetRole",
"iam:GetUser",
"lambda:GetFunction",
"rds:DescribeDBClusters",
"rds:DescribeDBInstances",
"rds:DescribeDBSecurityGroups",
"rds:DescribeDBSnapshots",
"rds:DescribeDBSubnetGroups",
"rds:DescribeEventSubscriptions",
"rds>ListTagsForResource",
"redshift:DescribeClusterParameters",
"redshift:DescribeClusterSecurityGroups",
"redshift:DescribeClusterSnapshots",
"redshift:DescribeClusterSubnetGroups",
"redshift:DescribeClusters",
"s3:GetBucketTagging",
"cloudformation:DescribeStacks",
"cloudwatch:DescribeAlarms",
"cloudwatch:DescribeInsightRules",
"cloudwatch>ListMetrics",
"cloudwatch:ListTagsForResource",
"config:DescribeComplianceByResource",
"config:DescribeRemediationConfigurations",
"config:GetComplianceDetailsByResource",
"config:GetResourceConfigHistory",
"config:StartConfigRulesEvaluation",
"ec2:DescribeInstances",
"eks:DescribeCluster",
"eks>ListClusters",
"eks:ListFargateProfiles",
"eks:ListNodegroups",
"eks:TagResource",
"resource-groups>CreateGroup",
"resource-groups>DeleteGroup",
"resource-groups:ListGroupQuery",
"resource-groups:GetTags",
"resource-groups:ListGroupResources",
"resource-groups>ListGroups",
```

```
 "resource-groups:Tag",
 "resource-groups:Untag",
 "ssm:CreateOpsMetadata",
 "ssm:DeleteOpsMetadata",
 "ssm:GetOpsSummary",
 "ssm:GetOpsMetadata",
 "ssm:UpdateServiceSetting",
 "ssm:GetServiceSetting",
 "ssm>ListOpsMetadata",
 "ssm:UpdateOpsItem",
 "tag:GetTagKeys",
 "tag:GetTagValues"
],
"Resource": "*"
}
]
```

**Note**

You can restrict a user's ability to make changes to applications and resources in Application Manager by removing the following API operations from the IAM permissions policy attached to their user, group, or role. Removing these actions creates a read-only experience in Application Manager.

```
eks:TagResource
resource-groups>CreateGroup
resource-groups>DeleteGroup
resource-groups:Tag
resource-groups:Untag
ssm:CreateOpsMetadata
ssm:DeleteOpsMetadata
ssm:UpdateOpsItem
```

For 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 user, group, or role, see [Adding and removing IAM identity permissions](#).

## Adding applications and clusters to Application Manager

Application Manager is a component of AWS Systems Manager. In Application Manager, an *application* is a logical group of AWS resources that you want to operate as a unit. This logical group can represent different versions of an application, ownership boundaries for operators, or developer environments, to name a few.

When you choose **Get started** on the Application Manager home page, Application Manager automatically imports metadata about your resources that were created in other AWS services or Systems Manager capabilities. For applications, Application Manager imports metadata about all of your AWS resources organized into resource groups. Each resource group is listed in the **Custom applications** category as a unique application. Application Manager also automatically imports metadata about resources that were created by AWS CloudFormation, AWS Launch Wizard, Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (Amazon EKS). Application Manager then displays those resources in predefined categories.

For **Applications**, the list includes the following:

- Custom applications
- Launch Wizard
- CloudFormation stacks

- AppRegistry applications

For **Container clusters**, the list includes the following:

- Amazon ECS clusters
- Amazon EKS clusters

After import is complete, you can view operations information for an application or a specific resource in these predefined categories. Or, if you want to provide more context about a collection of resources, you can manually create an application in Application Manager. You can then add resources or groups of resources into that application. After you create an application in Application Manager, you can view operations information about your resource in the context of an application.

## Creating an application in Application Manager

Use the following procedure to create an application in Application Manager and add resources to that application.

### To create an application in Application Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. Choose the **Applications** tab, and then choose **Create application**.
4. For **Application name**, enter a name to help you understand the purpose of the resources that will be added to this application.
5. For **Application description**, enter information about this application.
6. In the **Choose application components** section, use the options provided to choose resources for this application. You can add a combination of tagged resources, resource groups, and stacks to an application. You must choose a minimum of two components and a maximum of 15. If you choose resources by using tags, then all resources assigned those tags will be listed on the **Resources** tab after you add the new application. This is also true for resources included in a resource group or in a stack.

If you don't see the resources you want to add to the application, then verify the resources have been tagged properly, added to an AWS Resource Groups group, or added to an AWS CloudFormation stack.

7. For **Application tags - optional**, specify tags for this application.
8. Choose **Create**.

Application Manager creates the application and opens it. The **Components** tree lists the new application as the top-level component and the resources, groups, or stacks you selected as subcomponents. The next time you open Application Manager, you can find the new application in the **Custom applications** category.

## Working with Application Manager

Application Manager is a component of AWS Systems Manager. This section includes topics to help you work with Application Manager applications and clusters and view operations information about your AWS resources.

### Contents

- [Working with applications \(p. 242\)](#)

- [Working with AWS CloudFormation templates and stacks in Application Manager \(p. 248\)](#)
- [Working with clusters in Application Manager \(p. 254\)](#)

## Working with applications

Application Manager is a component of AWS Systems Manager. This section includes topics to help you work with Application Manager applications and view operations information about your AWS resources.

### Contents

- [Viewing overview information about an application \(p. 242\)](#)
- [Viewing application resources \(p. 243\)](#)
- [Viewing compliance information \(p. 243\)](#)
- [Viewing monitoring information \(p. 244\)](#)
- [Viewing OpsItems for an application \(p. 245\)](#)
- [Viewing log groups and log data \(p. 246\)](#)
- [Working with runbooks in Application Manager \(p. 246\)](#)
- [Working with tags in Application Manager \(p. 247\)](#)

### Viewing overview information about an application

In Application Manager, a component of AWS Systems Manager, the **Overview** tab displays a summary of Amazon CloudWatch alarms, operational work items (OpsItems), CloudWatch Application Insights, and runbook history. Choose **View all** for any card to open the corresponding tab where you can view all application insights, alarms, OpsItems, or runbook history.

#### About Application Insights

CloudWatch Application Insights identifies and sets up key metrics, logs, and alarms across your application resources and technology stack. Application Insights continuously monitors metrics and logs to detect and correlate anomalies and errors. When the system detects errors or anomalies, Application Insights generates CloudWatch Events that you can use to set up notifications or take actions. If you choose the **Edit configuration** button on the **Monitoring** tab, the system opens the CloudWatch Application Insights console. For more information about Application Insights, see [What is Amazon CloudWatch Application Insights](#) in the *Amazon CloudWatch User Guide*.

#### Actions you can perform on this page

You can perform the following actions on this page:

- In the **Alarms** section, choose a number to open the **Monitoring** tab where you can view more details about alarms of the chosen severity.
- In the **OpsItems** section, choose a severity to open the **OpsItems** tab where you can view all OpsItems of the chosen severity.
- In the **Runbooks** section, choose a runbook to open it in the Systems Manager **Documents** page where you can view more details about the document.
- If you enabled AWS Cost Explorer, the **Cost Explorer** section shows cost data for a specific application or application component. You can enable this feature by choosing the **Go to Billing console** button and then choosing **Cost Explorer** in the Billing and Cost Management console. By default, the data is filtered to the past three months. For a non-container application, if you choose the **View all** button in this section, Application Manager opens the **Resources** tab. For container applications, the **View all** button opens the AWS Cost Explorer console.

- Choose a **View all** button to open the corresponding tab. You can view all alarms, OpsItems, or runbook history entries for the application.

### To open the Overview tab

- Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
- In the navigation pane, choose **Application Manager**.
- In the **Applications** section, choose a category. If you want to open an application you created manually in Application Manager, choose **Custom applications**.
- Choose the application in the list. Application Manager opens the **Overview** tab.

## Viewing application resources

In Application Manager, a component of AWS Systems Manager, the **Resources** tab displays the AWS resources in your application. If you choose a top-level component, this page displays all resources for that component and any subcomponents. If you choose a subcomponent, this page shows only the resources assigned to that subcomponent.

### Actions you can perform on this page

You can perform the following actions on this page:

- Choose a resource name to view information about it, including details provided by the console where it was created, tags, Amazon CloudWatch alarms, AWS Config details, and AWS CloudTrail log information.
- Choose the option button beside a resource name. Then, choose the **Resource timeline** button to open the AWS Config console where you can view compliance information about a selected resource.
- If you enabled AWS Cost Explorer, the **Cost Explorer** section shows cost data for a specific non-container application or application component. You can enable this feature by choosing the **Go to Billing console** button and then choosing **Cost Explorer** in the Billing and Cost Management console. Use the filters in this section to view cost information about your application.

### To open the Resources tab

- Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
- In the navigation pane, choose **Application Manager**.
- In the **Applications** section, choose a category. If you want to open an application you created manually in Application Manager, choose **Custom applications**.
- Choose the application in the list. Application Manager opens the **Overview** tab.
- Choose the **Resources** tab.

## Viewing compliance information

In Application Manager, a component of AWS Systems Manager, the **Configurations** page displays [AWS Config](#) resource and configuration rule compliance information. This page also displays AWS Systems Manager [State Manager](#) association compliance information. You can choose a resource, a rule, or an association to open the corresponding console for more information. This page displays compliance information from the last 90 days.

### Actions you can perform on this page

You can perform the following actions on this page:

- Choose a resource name to open the AWS Config console where you can view compliance information about a selected resource.
- Choose the option button beside a resource name. Then, choose the **Resource timeline** button to open the AWS Config console where you can view compliance information about a selected resource.
- In the **Config rules compliance** section, you can do the following:
  - Choose a rule name to open the AWS Config console where you can view information about that rule.
  - Choose **Add rules** to open the AWS Config console where you can create a rule.
  - Choose the option button beside a rule name, choose **Actions**, and then choose **Manage remediation** to change the remediation action for a rule.
  - Choose the option button beside a rule name, choose **Actions**, and then choose **Re-evaluate** to have AWS Config run a compliance check on the selected rule.
- In the **Association compliance** section, you can do the following:
  - Choose an association name to open the **Associations** page where you can view information about that association.
  - Choose **Create association** to open Systems Manager State Manager where you can create an association.
  - Choose the option button beside an association name and choose **Apply association** to immediately start all actions specified in the association.

### To open the Compliance tab

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose a category. If you want to open an application you created manually in Application Manager, choose **Custom applications**.
4. Choose the application in the list. Application Manager opens the **Overview** tab.
5. Choose the **Compliance** tab.

## Viewing monitoring information

In Application Manager, a component of AWS Systems Manager, the **Monitoring** tab displays Amazon CloudWatch Application Insights and alarm details for resources in an application.

### About Application Insights

CloudWatch Application Insights identifies and sets up key metrics, logs, and alarms across your application resources and technology stack. Application Insights continuously monitors metrics and logs to detect and correlate anomalies and errors. When the system detects errors or anomalies, Application Insights generates CloudWatch Events that you can use to set up notifications or take actions. If you choose the **Edit configuration** button on the **Monitoring** tab, the system opens the CloudWatch Application Insights console. For more information about Application Insights, see [What is Amazon CloudWatch Application Insights](#) in the *Amazon CloudWatch User Guide*.

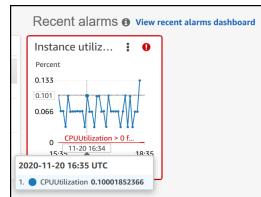
### Actions you can perform on this page

You can perform the following actions on this page:

- Choose a service name in the **Alarms by AWS service** section to open CloudWatch to the selected service and alarm.
- Adjust the time period for data displayed in widgets in the **Recent alarms** section by selecting one of the predefined time period values. You can choose **custom** to define your own time period.

1h 3h 12h 1d 3d 1w custom ▾

- Hover your cursor over a widget in the **Recent alarms** section to view a data pop-up for a specific time.



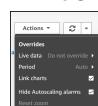
- Choose the options menu in a widget to view display options. Choose **Enlarge** to expand a widget. Choose **Refresh** to update the data in a widget. Click and drag your cursor in a widget data display to select a specific range. You can then choose **Apply time range**.



- Choose the **Actions** menu to view alarm data **Override** options, which include the following:
  - Choose whether your widget displays live data. Live data is data published within the last minute that hasn't been fully aggregated. If live data is turned off, only data points with an aggregation period of at least one minute in the past are shown. For example, when using 5-minute periods, the data point for 12:35 would be aggregated from 12:35 to 12:40, and displayed at 12:41.

If live data is turned on, the most recent data point is shown as soon as any data is published in the corresponding aggregation interval. Each time you refresh the display, the most recent data point might change as new data within that aggregation period is published.

- Specify a time period for live data.
- Link the charts in the **Recent alarms** section, so that when you zoom in or zoom out on one chart, the other chart zooms in or zooms out at the same time. You can unlink charts to limit zoom to one chart.
- Hide Auto Scaling alarms.



## To open the Monitoring tab

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose a category. If you want to open an application you created manually in Application Manager, choose **Custom applications**.
4. Choose the application in the list. Application Manager opens the **Overview** tab.
5. Choose the **Monitoring** tab.

## Viewing OpsItems for an application

In Application Manager, a component of AWS Systems Manager, the **OpsItems** tab displays operational work items (OpsItems) for resources in the selected application. You can configure Systems Manager OpsCenter to automatically create OpsItems from Amazon CloudWatch alarms and Amazon EventBridge events. You can also manually create OpsItems.

### Actions you can perform on this tab

You can perform the following actions on this page:

- Filter the list of OpsItems by using the search field. You can filter by OpsItem name, ID, source ID, or severity. You can also filter the list based on status. OpsItems support the following statuses: Open, In progress, Open and In progress, Resolved, or All.
- Change the status of an OpsItem by choosing the option button beside it and then choosing an option in the **Set status** menu.
- Open Systems Manager OpsCenter to create an OpsItem by choosing **Create OpsItem**.

### To open the OpsItems tab

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose a category. If you want to open an application you created manually in Application Manager, choose **Custom applications**.
4. Choose the application in the list. Application Manager opens the **Overview** tab.
5. Choose the **OpsItems** tab.

## Viewing log groups and log data

In Application Manager, a component of AWS Systems Manager, the **Logs** tab displays a list of log groups from Amazon CloudWatch Logs.

### Actions you can perform on this tab

You can perform the following actions on this page:

- Choose a log group name to open it in CloudWatch Logs. You can then choose a log stream to view logs for a resource in the context of an application.
- Choose **Create log groups** to create a log group in CloudWatch Logs.

### To open the Logs tab

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose a category. If you want to open an application you created manually in Application Manager, choose **Custom applications**.
4. Choose the application in the list. Application Manager opens the **Overview** tab.
5. Choose the **Logs** tab.

## Working with runbooks in Application Manager

You can remediate issues with AWS resources from Application Manager, a capability of AWS Systems Manager, by using Systems Manager Automation runbooks. When you choose **Start runbook** from an Application Manager application or cluster, the system displays a filtered list of runbooks based on the type of resources in your application or cluster. When you choose the runbook you want to start, Systems Manager opens the **Execute automation document** page.

Application Manager includes the following enhancements for working with runbooks.

- If you choose the name of a resource in Application Manager and then choose **Execute runbook**, the system displays a filtered list of runbooks for that resource type.
- You can initiate an automation on all resources of the same type by choosing a runbook in the list and then choosing **Run for resources of same type**.

## Before you begin

Before you start a runbook from Application Manager, do the following:

- Verify that you have the correct permissions for starting runbooks. For more information, see [Setting up Automation \(p. 401\)](#).
- Review the Automation procedure documentation about starting runbooks. For more information, see [Working with automations \(p. 407\)](#).

## To start a runbook from Application Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose a category. If you want to open an application you created manually in Application Manager, choose **Custom applications**.
4. Choose the application in the list. Application Manager opens the **Overview** tab.
5. Choose **Start runbook**. Application Manager opens the **Execute automation document** page in a new tab. For information about the options in the **Execute automation document** page, see [Working with automations \(p. 407\)](#).

## Working with tags in Application Manager

You can quickly add or delete tags on applications and AWS resources in Application Manager. For more information about tags, see [Tagging Systems Manager resources \(p. 1505\)](#).

Use the following procedure to add a tag to or delete a tag from an application and all AWS resources in that application.

### To add a tag to or delete a tag from an application and all resources in the application

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose a category. If you want to open an application you created manually in Application Manager, choose **Custom applications**.
4. Choose the application in the list. Application Manager opens the **Overview** tab.
5. In the **Application information** section, choose the number beneath **Application tags**. If no tags are assigned to the application, the number is zero.
6. To add a tag, choose **Add new tag**. Specify a key and an optional value. To delete a tag, choose **Remove**.
7. Choose **Save**.

Use the following procedure to add a tag to or delete a tag from a specific resource in Application Manager.

### To add a tag to or delete a tag from a resource

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose a category. If you want to open an application you created manually in Application Manager, choose **Custom applications**.
4. Choose the application in the list. Application Manager opens the **Overview** tab.
5. Choose the **Resources** tab.

6. Choose a resource name.
7. In the **Tags** section choose **Edit**.
8. To add a tag, choose **Add new tag**. Specify a key and an optional value. To delete a tag, choose **Remove**.
9. Choose **Save**.

## Working with AWS CloudFormation templates and stacks in Application Manager

Application Manager, a capability of AWS Systems Manager, helps you provision and manage resources for your applications by integrating with AWS CloudFormation. You can create, edit, and delete AWS CloudFormation templates and stacks in Application Manager. A *stack* is a collection of AWS resources that you can manage as a single unit. This means you can create, update, or delete a collection of AWS resources by using CloudFormation stacks. A *template* is a formatted text file in JSON or YAML that specifies the resources you want to provision in your stacks.

Application Manager also includes a template library where you can clone, create, and store templates. Application Manager and CloudFormation display the same information about the current status of a stack. Templates and template updates are stored in Systems Manager until you provision the stack, at which time the changes are also displayed in CloudFormation.

After you create a stack in Application Manager, the **CloudFormation stacks** page displays helpful information about it. This includes the template used to create it, a count of **OpsItems** for resources in your stack, the **stack status**, and **drift status**.

### Before you begin

Use the following links to learn about CloudFormation concepts before you create, edit, or delete CloudFormation templates and stacks by using Application Manager.

- [What is AWS CloudFormation?](#)
- [AWS CloudFormation best practices](#)
- [Learn template basics](#)
- [Working with AWS CloudFormation stacks](#)
- [Working with AWS CloudFormation templates](#)
- [Sample templates](#)

### Topics

- [Working with CloudFormation templates \(p. 248\)](#)
- [Working with CloudFormation stacks \(p. 251\)](#)

## Working with CloudFormation templates

Application Manager, a capability of AWS Systems Manager, includes a template library and other tools to help you manage AWS CloudFormation templates. This section includes the following information.

### Topics

- [Working with the template library \(p. 249\)](#)
- [Creating a template \(p. 249\)](#)
- [Editing a template \(p. 251\)](#)

## Working with the template library

The Application Manager template library provides tools to help you view, create, edit, delete, and clone templates. You can also provision stacks directly from the template library. Templates are stored as Systems Manager (SSM) documents of type CloudFormation. By storing templates as SSM documents, you can use version controls to work with different versions of a template. You can also set permissions and share templates. After you successfully provision a stack, the stack and template are available in Application Manager and CloudFormation.

### Before you begin

We recommend that you read the following topics to learn more about SSM documents before you start working with CloudFormation templates in Application Manager.

- [AWS Systems Manager documents \(p. 1287\)](#)
- [Sharing SSM documents \(p. 1361\)](#)
- [Best practices for shared SSM documents \(p. 1362\)](#)

### To view the template library in Application Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose **CloudFormation stacks**.
4. Choose **Template library**.

### Creating a template

The following procedure describes how to create a CloudFormation template in Application Manager. When you create a template, you enter the stack details of the template in either JSON or YAML. If you're unfamiliar with JSON or YAML, you can use AWS CloudFormation Designer, a tool for visually creating and modifying templates. For more information, see [What is AWS CloudFormation Designer?](#) in the *AWS CloudFormation User Guide*. For information about the structure and syntax of a template, see [Template anatomy](#).

You can also construct a template from multiple template snippets. Template snippets are examples that demonstrate how to write templates for a particular resource. For example, you can view snippets for Amazon Elastic Compute Cloud (Amazon EC2) instances, Amazon Simple Storage Service (Amazon S3) domains, AWS CloudFormation mappings, and more. Snippets are grouped by resource. You can find general-purpose AWS CloudFormation snippets in the [General template snippets](#) section of the *AWS CloudFormation User Guide*.

### [Creating a CloudFormation template in Application Manager \(console\)](#)

Use the following procedure to create a CloudFormation template in Application Manager by using the AWS Management Console.

### To create a CloudFormation template in Application Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose **CloudFormation stacks**.
4. Choose **Template library**, and then either choose **Create template** or choose an existing template and then choose **Actions, Clone**.
5. For **Name**, enter a name for the template that helps you identify the resources it creates or the purpose of the stack.

6. (Optional) For **Version name**, enter a name or a number to identify the template version.
7. (Optional) For **Description**, enter information about this template.
8. In the **Code editor** section, choose either **YAML** or **JSON** and then either enter or copy and paste your template code.
9. (Optional) In the **Tags** section, apply one or more tag key name/value pairs to the template.

Tags are optional metadata that you assign to a resource. By using tags, you can categorize a resource in different ways, such as by purpose, owner, or environment. For more information about tagging Systems Manager resources, see [Tagging Systems Manager resources \(p. 1505\)](#).

10. (Optional) In the **Permissions** section, enter an AWS account ID and choose **Add account**. This action provides read permission to the template. The account owner can provision and clone the template, but they can't edit or delete it.
11. Choose **Create**. The template is saved in the Systems Manager (SSM) Document service.

### [Creating a CloudFormation template in Application Manager \(command line\)](#)

After you create the content of your CloudFormation template in JSON or YAML, you can use the AWS Command Line Interface (AWS CLI) or AWS Tools for PowerShell to save the template as an SSM document.

#### 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 \(p. 60\)](#).

#### Linux & macOS

```
aws ssm create-document \
--content file:///path/to/template_in_json_or_yaml \
--name "a_name_for_the_template" \
--document-type "CloudFormation" \
--document-format "JSON or YAML" \
--tags "Key=tag-key,Value=tag-value"
```

#### Windows

```
aws ssm create-document ^
--content file://C:/path/to/template_in_json_or_yaml ^
--name "a_name_for_the_template" ^
--document-type "CloudFormation" ^
--document-format "JSON or YAML" ^
--tags "Key=tag-key,Value=tag-value"
```

#### PowerShell

```
$json = Get-Content -Path "C:/path/to/template_in_json_or_yaml | Out-String
New-SSMDocument `-
-Content $json `-
-Name "a_name_for_the_template" `-
-DocumentType "CloudFormation" `-
-DocumentFormat "JSON or YAML" `-
-Tags "Key=tag-key,Value=tag-value"
```

If successful, the command returns a response similar to the following.

```
{
```

```
"DocumentDescription": {
 "Hash": "c1d9640f15fbdba6deb41af6471d6ace0acc22f213bdd1449f03980358c2d4fb",
 "HashType": "Sha256",
 "Name": "MyTestCFTemplate",
 "Owner": "428427166869",
 "CreatedDate": "2021-06-04T09:44:18.931000-07:00",
 "Status": "Creating",
 "DocumentVersion": "1",
 "Description": "My test template",
 "PlatformTypes": [],
 "DocumentType": "CloudFormation",
 "SchemaVersion": "1.0",
 "LatestVersion": "1",
 "DefaultVersion": "1",
 "DocumentFormat": "YAML",
 "Tags": [
 {
 "Key": "Templates",
 "Value": "Test"
 }
]
}
```

## Editing a template

Use the following procedure to edit a CloudFormation template in Application Manager. Template changes are available in CloudFormation after you provision a stack that uses the updated template.

### To edit a CloudFormation template in Application Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose **CloudFormation stacks**.
4. Choose **Template library**.
5. Choose a template, and then choose **Actions**, **Edit**. You can't change the name of a template, but you can change all other details.
6. Choose **Save**. The template is saved in the Systems Manager Document service.

## Working with CloudFormation stacks

Application Manager, a capability of AWS Systems Manager, helps you provision and manage resources for your applications by integrating with AWS CloudFormation. You can create, edit, and delete CloudFormation templates and stacks in Application Manager. A *stack* is a collection of AWS resources that you can manage as a single unit. This means you can create, update, or delete a collection of AWS resources by using CloudFormation stacks. A *template* is a formatted text file in JSON or YAML that specifies the resources you want to provision in your stacks. This section includes the following information.

### Topics

- [Creating a stack \(p. 251\)](#)
- [Updating a stack \(p. 253\)](#)

### Creating a stack

The following procedures describe how to create a CloudFormation stack by using Application Manager. A stack is based on a template. When you create a stack, you can either choose an existing template or create a new one. After you create the stack, the system immediately attempts to create the resources

identified in the stack. After the system successfully provisions the resources, the template and the stack are available to view and edit in Application Manager and CloudFormation.

**Note**

There is no charge to use Application Manager to create a stack, but you are charged for AWS resources created in the stack.

### [Creating a CloudFormation stack by using Application Manager \(console\)](#)

Use the following procedure to create a stack by using Application Manager in the AWS Management Console.

#### **To create a CloudFormation stack**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose **CloudFormation stacks**.
4. In the **Prepare a template** section, choose an option. If you choose **Use an existing template**, you can use the tabs in the **Choose a template** section to locate the template you want. If you choose one of the other options, complete the wizard to prepare a template.
5. On the **Specify template details** page, verify the details of the template to ensure the process creates the resources you want.
  - (Optional) In the **Tags** section, apply one or more tag key name/value pairs to the template.
  - Tags are optional metadata that you assign to a resource. By using tags, you can categorize a resource in different ways, such as by purpose, owner, or environment. For more information about tagging Systems Manager resources, see [Tagging Systems Manager resources \(p. 1505\)](#).
  - Choose **Next**.
6. On the **Edit stack details** page, for **Stack name**, enter a name that helps you identify the resources created by the stack or its purpose.
  - The **Parameters** section includes all optional and required parameters specified in the template. Enter one or more parameters in each field.
  - (Optional) In the **Tags** section, apply one or more tag key name/value pairs to the stack.
  - (Optional) In the **Permissions** section, specify an AWS Identity and Access Management (IAM) role name or an IAM Amazon Resource Name (ARN). The system uses the specified service role to create all resources specified in your stack. If you don't specify an IAM role, then AWS CloudFormation uses a temporary session that the system generates from your user credentials. For more information about this IAM role, see [AWS CloudFormation service role](#) in the [AWS CloudFormation User Guide](#).
  - Choose **Next**.
7. On the **Review and provision** page, review all the details of the stack. Choose an **Edit** button on this page to make change.
8. Choose **Provision stack**.

Application Manager displays the **CloudFormation stacks** page and the status of the stack creation and deployment. If CloudFormation fails to create and provision the stack, see the following topics in the [AWS CloudFormation User Guide](#).

- [Stack status codes](#)
- [Troubleshooting AWS CloudFormation](#)

After your stack resources are provisioned and running, users can edit resources directly by using the underlying service that created the resource. For example, a user can use the Amazon Elastic Compute

Cloud (Amazon EC2) console to update a server instance that was created as part of a CloudFormation stack. Some changes may be accidental, and some may be made intentionally to respond to time-sensitive operational events. Regardless, changes made outside of CloudFormation can complicate stack update or deletion operations. You can use drift detection or *drift status* to identify stack resources to which configuration changes have been made outside of CloudFormation management. For information about drift status, see [Detecting unmanaged configuration changes to stacks and resources](#).

### Creating a CloudFormation stack by using Application Manager (command line)

Use the following AWS Command Line Interface (AWS CLI) procedure to provision a stack by using a CloudFormation template that is stored as an SSM document in Systems Manager. For information about other AWS CLI procedures for creating stacks, see [Creating a stack](#) in the *AWS CloudFormation User Guide*.

#### 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 \(p. 60\)](#).

#### Linux & macOS

```
aws cloudformation create-stack \
--stack-name a_name_for_the_stack \
--template-url "ssm-doc://arn:aws:ssm:Region:account_ID:document/template_name" \
```

#### Windows

```
aws cloudformation create-stack ^
--stack-name a_name_for_the_stack ^
--template-url "ssm-doc://arn:aws:ssm:Region:account_ID:document/template_name" ^
```

#### PowerShell

```
New-CFNSStack `
-StackName "a_name_for_the_stack" `
-TemplateURL "ssm-doc://arn:aws:ssm:Region:account_ID:document/template_name" `
```

### Updating a stack

You can deploy updates to a CloudFormation stack by directly editing the stack in Application Manager. With a direct update, you specify updates to a template or input parameters. After you save and deploy the changes, CloudFormation updates the AWS resources according to the changes you specified.

You can preview the changes that CloudFormation will make to your stack before you update it by using change sets. For more information, see [Updating stacks using change sets](#) in the *AWS CloudFormation User Guide*.

### To update a CloudFormation stack in Application Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Applications** section, choose **CloudFormation stacks**.
4. Choose a stack in the list and then choose **Actions, Update stack**.
5. On the **Specify template source** page, choose one of the following options, and then choose **Next**.
  - Choose **Use the template code currently provisioned in the stack** to view a template. Choose a template version in the **Versions** list, and then choose **Next**.

- Choose **Switch to a different template** to choose or create a new template for the stack.
6. After you finish making changes to the template, choose **Next**.
  7. On the **Edit stack details** page, you can edit parameters, tags, and permissions. You can't change the stack name. Make your changes and choose **Next**.
  8. On the **Review and provision** page, review all the details of the stack, and then choose **Provision stack**.

## Working with clusters in Application Manager

This section includes topics to help you work with Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS) container clusters in Application Manager, a component of AWS Systems Manager.

### Contents

- [Working with Amazon ECS in Application Manager \(p. 254\)](#)
- [Working with Amazon EKS in Application Manager \(p. 254\)](#)
- [Working with runbooks for clusters \(p. 255\)](#)

## Working with Amazon ECS in Application Manager

With Application Manager, a capability of AWS Systems Manager, you can view and manage your Amazon Elastic Container Service (Amazon ECS) cluster infrastructure and the component runtime view of networking in the cluster and storage resources of the cluster.

### Actions you can perform on this page

You can perform the following actions on this page:

- Choose **Manage cluster** to open the cluster in Amazon ECS.
- Choose **View all** to view a list of resources in your cluster.
- Choose **View in CloudWatch** to view resource alarms in Amazon CloudWatch.
- Choose **Manage nodes** or **Manager Fargate profiles** to view these resources in Amazon ECS.
- Choose a resource ID to view detailed information about it in the console where it was created.
- View a list of OpsItems related to your clusters.
- View a history of runbooks that have been run on your clusters.

### To open an ECS cluster

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Container clusters** section, choose **ECS clusters**.
4. Choose a cluster in the list. Application Manager opens the **Overview** tab.

## Working with Amazon EKS in Application Manager

Application Manager, a capability of AWS Systems Manager, integrates with [Amazon Elastic Kubernetes Service](#) (Amazon EKS) to provide information about the health of your Amazon EKS cluster infrastructure and a component runtime view of the compute, networking, and storage resources in a cluster.

### Note

You can't manage or view operations information about your Amazon EKS pods or containers in Application Manager. You can only manage and view operations information about the infrastructure hosting your Amazon EKS resources.

### Actions you can perform on this page

You can perform the following actions on this page:

- Choose **Manage cluster** to open the cluster in Amazon EKS.
- Choose **View all** to view a list of resources in your cluster.
- Choose **View in CloudWatch** to view resource alarms in Amazon CloudWatch.
- Choose **Manage nodes** or **Manager Fargate profiles** to view these resources in Amazon EKS.
- Choose a resource ID to view detailed information about it in the console where it was created.
- View a list of OpsItems related to your clusters.
- View a history of runbooks that have been run on your clusters.

### To open an EKS clusters application

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Container clusters** section, choose **EKS clusters**.
4. Choose a cluster in the list. Application Manager opens the **Overview** tab.

## Working with runbooks for clusters

You can remediate issues with AWS resources from Application Manager, a capability of AWS Systems Manager, by using Systems Manager Automation runbooks. When you choose **Start runbook** from an Application Manager cluster, the system displays a filtered list of runbooks based on the type of resources in your cluster. When you choose the runbook you want to start, Systems Manager opens the **Execute automation document** page.

### Before you begin

Before you start a runbook from Application Manager, do the following:

- Verify that you have the correct permissions for starting runbooks. For more information, see [Setting up Automation \(p. 401\)](#).
- Review the Automation procedure documentation about starting runbooks. For more information, see [Working with automations \(p. 407\)](#).
- If you intend to start runbooks on multiple resources at one time, review the documentation about using targets and rate controls. For more information, see [Running automations that use targets and rate controls \(p. 421\)](#).

### To start a runbook for clusters from Application Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Application Manager**.
3. In the **Container clusters** section, choose a container type.
4. Choose the cluster in the list. Application Manager opens the **Overview** tab.

5. On the **Runbooks** tab, choose **Start runbook**. Application Manager opens the **Execute automation document** page in a new tab. For information about the options in the **Execute automation document** page, see [Working with automations \(p. 407\)](#).

## AWS AppConfig

Use AWS AppConfig, a capability of AWS Systems Manager, to create, manage, and quickly deploy application configurations. AWS AppConfig supports controlled deployments to applications of any size and includes built-in validation checks and monitoring. You can use AWS AppConfig with applications hosted on Amazon Elastic Compute Cloud (Amazon EC2) instances, AWS Lambda, containers, mobile applications, or IoT devices.

Information about AWS AppConfig has been moved into a separate user guide. For more information, see [What Is AWS AppConfig?](#)

## AWS Systems Manager Parameter Store

Parameter Store, a capability of AWS Systems Manager, 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. To get started with Parameter Store, open the [Systems Manager console](#). In the navigation pane, choose **Parameter Store**.

Parameter Store is also integrated with 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 \(p. 1488\)](#).

### Note

To implement password rotation lifecycles, use AWS Secrets Manager. You can rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle using Secrets Manager. For more information, see [What is AWS Secrets Manager?](#) in the *AWS Secrets Manager User Guide*.

## How can Parameter Store benefit my organization?

Parameter Store offers these benefits:

- 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.
- Store parameters reliably because Parameter Store is hosted in multiple Availability Zones in an AWS Region.

## Who should use Parameter Store?

- Any AWS customer who wants to have a centralized way to manage configuration data.
- Software developers who want to store different logins and reference streams.

- Administrators who want to receive notifications when their secrets and passwords are or aren't changed.

## What are the features of Parameter Store?

- **Change notification**

You can configure change notifications and invoke automated actions for both parameters and parameter policies. For more information, see [Setting up notifications or trigger actions based on Parameter Store events \(p. 275\)](#).

- **Organize and control access**

You can tag your parameters individually to help you 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 AWS Identity and Access Management (IAM) policy that specifies the tags that a user or group can access. For more information, see [Tagging Systems Manager parameters \(p. 1525\)](#).

- **Label versions**

You can associate an alias for versions of your parameter by creating labels. Labels can help you remember the purpose of a parameter version when there are multiple versions.

- **Data validation**

You can create parameters that point to an Amazon Elastic Compute Cloud (Amazon EC2) instance and Parameter Store validates these parameters to make sure that it references expected resource type, that the resource exists, and that the customer has permission to use the resource. For example, you can create a parameter with Amazon Machine Image (AMI) ID as a value with `aws:ec2:image` data type, and Parameter Store performs an asynchronous validation operation to make sure that the parameter value meets the formatting requirements for an AMI ID, and that the specified AMI is available in your AWS account.

- **Reference secrets**

Parameter Store is 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.

- **Accessible from other AWS services**

You can 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, Automation, and State Manager, capabilities of AWS Systems Manager. 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

- **Integrate with other AWS services**

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: For more information, see [Configuring EventBridge for parameters \(p. 276\)](#).
- Amazon EventBridge: For more information, see [Monitoring Systems Manager status changes using Amazon SNS notifications \(p. 1462\)](#) and [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#).
- AWS CloudTrail: For more information, see [Logging AWS Systems Manager API calls with AWS CloudTrail \(p. 1442\)](#).

## What is a parameter?

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 AMI ID, a license key, and so on. You can centrally and securely reference this data in your scripts, commands, and SSM documents.

When you reference a parameter, you specify the parameter name by using the following convention.

`{{ssm:parameter-name`

### Note

Parameters can't be referenced or nested in the values of other parameters. You can't include `{}` or `{{ssm:parameter-name in a parameter value.`

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 plaintext, 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-12345abcdeEXAMPLE`.

### String

By default, `String` parameters consist of any block of text you enter. For example:

- abc123
- Example Corp
- 

### 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

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` data type.

### Important

Don't 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 [Create a SecureString parameter \(AWS CLI\) \(p. 287\)](#).

We recommend using SecureString parameters for the following scenarios:

- You want to use data/parameters across AWS services without exposing the values as plaintext in commands, functions, agent logs, or CloudTrail logs.
- You want to control who has access to sensitive data.
- You want to be able to audit when sensitive data is accessed (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 aren't encrypted.

You can use the SecureString parameter type for textual data that you want to encrypt, such as passwords, application secrets, confidential configuration data, or any other types of data that you want to protect. SecureString data is encrypted and decrypted using an AWS KMS key. You can use either a default KMS key provided by AWS or create and use your own AWS KMS key. (Use your own AWS KMS key if you want to restrict user access to SecureString parameters. For more information, see [IAM permissions for using AWS default keys and customer managed keys \(p. 263\)](#).)

You can also use SecureString parameters with other AWS services. In the following example, the Lambda function retrieves a SecureString parameter by using the [GetParameters API](#).

```
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
```

### AWS KMS encryption and pricing

If you choose the SecureString parameter type when you create your parameter, Systems Manager uses AWS KMS to encrypt the parameter value.

**Important**

Parameter Store only supports [symmetric encryption KMS keys](#). You can't use an [asymmetric encryption KMS key](#) to encrypt your parameters. For help determining whether a KMS key is symmetric or asymmetric, see [Identifying symmetric and asymmetric KMS keys in the AWS Key Management Service Developer Guide](#)

There is no charge from Parameter Store to create a SecureString parameter, but charges for use of AWS KMS encryption do apply. For information, see [AWS Key Management Service pricing](#).

For more information about AWS managed and customer managed keys, 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 an AWS managed key, use the AWS KMS `DescribeKey` operation. This AWS Command Line Interface (AWS CLI) example uses `DescribeKey` to view an AWS managed key.

```
aws kms describe-key --key-id alias/aws/ssm
```

**Related topics**

For an example of how to create and use a `SecureString` parameter, see [Create a SecureString parameter and join a node to a Domain \(PowerShell\) \(p. 342\)](#). For more information about using Systems Manager parameters with other AWS services, see the following blog posts:

- [Use Parameter Store to Securely Access Secrets and Config Data in CodeDeploy](#)
- [Interesting Articles on Amazon EC2 Systems Manager Parameter Store](#)

## Setting up Parameter Store

Before setting up parameters in Parameter Store, a capability of AWS Systems Manager, 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 a managed node. This section also includes information about how to create Amazon EventBridge rules that let you receive notifications about changes to Systems Manager parameters. You can also use EventBridge rules to invoke other actions in AWS based on changes in Parameter Store.

**Contents**

- [Restricting access to Systems Manager parameters using IAM policies \(p. 260\)](#)
- [Managing parameter tiers \(p. 265\)](#)
- [Increasing Parameter Store throughput \(p. 271\)](#)
- [Setting up notifications or trigger actions based on Parameter Store events \(p. 275\)](#)

## Restricting access to Systems Manager parameters using IAM policies

You restrict access to AWS 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:

- `DeleteParameter`
- `DeleteParameters`
- `DescribeParameters`
- `GetParameter`
- `GetParameters`
- `GetParameterHistory`
- `GetParametersByPath`
- `PutParameter`

When using IAM policies to restrict access to Systems Manager parameters, we recommend that you 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"
],
 "Resource": "arn:aws:ssm:us-east-2:123456789012:parameter/prod-*"
 }
]
}
```

### 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 operation recursively for `/a` and view `/a/b`.

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": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:PutParameter",
 "ssm>DeleteParameter",
 "ssm:GetParameterHistory",
 "ssm:GetParametersByPath",
 "ssm:GetParameters",
 "ssm:GetParameter",
 "ssm:DeleteParameters"
],
 "Resource": "arn:aws:ssm:us-east-2:123456789012:parameter/dbserver-prod-*"
 },
 {
 "Effect": "Allow",
 "Action": "ssm:DescribeParameters",
 "Resource": "*"
 }
]
}
```

### Deny permissions

Each API is unique and has distinct operations and permissions that you can allow or deny individually. An explicit deny in any policy overrides the allow.

**Note**

The default AWS Key Management Service (AWS KMS) key has Decrypt permission for all IAM principals within the AWS account. If you want to have different access levels to SecureString parameters in your account, we don't recommend that you use the default key.

If you want all API operations retrieving parameter values to have the same behavior, then you can use a pattern like GetParameter\* in a policy. The following example shows how to deny GetParameter, GetParameters, GetParameterHistory, and GetParametersByPath for all parameters beginning with prod-\*.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Deny",
 "Action": [
 "ssm:GetParameter*",
 "ssm:PutParameter",
 "ssm:DeleteParameter",
 "ssm:DeleteParameters",
 "ssm:DescribeParameters"
],
 "Resource": "arn:aws:ssm:us-east-2:123456789012:parameter/prod-*"
 }
]
}
```

The following example shows how to deny some commands while allowing the user to perform other commands on all parameters that begin with prod-\*.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Deny",
 "Action": [
 "ssm:PutParameter",
 "ssm:DeleteParameter",
 "ssm:DeleteParameters",
 "ssm:DescribeParameters"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetParametersByPath",
 "ssm:GetParameters",
 "ssm:GetParameter",
 "ssm:GetParameterHistory"
],
 "Resource": "arn:aws:ssm:us-east-2:123456789012:parameter/prod-*"
 }
]
}
```

**Note**

The parameter history includes all parameter versions, including the current one. Therefore, if a user is denied permission for GetParameter, GetParameters, and GetParameterByPath but is allowed permission for GetParameterHistory, they can see the current parameter, including SecureString parameters, using GetParameterHistory.

## Allowing only specific parameters to run on nodes

You can control access so that managed nodes can run only parameters that you specify.

If you choose the `SecureString` parameter type when you create your parameter, Systems Manager uses AWS KMS to encrypt the parameter value. AWS KMS encrypts the value by using either an AWS managed key or a customer managed key. For more information about AWS KMS and AWS KMS key, see the [AWS Key Management Service Developer Guide](#).

You can view the AWS managed key by running the following command from the AWS CLI.

```
aws kms describe-key --key-id alias/aws/ssm
```

The following example allows nodes to get a parameter value only for parameters that begin with `prod-`. If the parameter is a `SecureString` parameter, then the node 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 for EC2 instances \(p. 16\)](#).

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetParameters"
],
 "Resource": [
 "arn:aws:ssm:us-east-2:123456789012:parameter/prod-*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "kms:Decrypt"
],
 "Resource": [
 "arn:aws:kms:us-east-2:123456789012:key/4914ec06-e888-4ea5-a371-5b88eEXAMPLE"
]
 }
]
}
```

## IAM permissions for using AWS default keys and customer managed keys

Parameter Store `SecureString` parameters are encrypted and decrypted using AWS KMS keys. You can choose to encrypt your `SecureString` parameters using either an AWS KMS key 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 AWS Region and AWS account.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:PutParameter",
 "ssm:GetParameter"
],
 "Resource": [
 "arn:aws:ssm:us-east-2:123456789012:parameter/prod-*"
]
 }
]
}
```

```

 "ssm:PutParameter",
 "ssm:GetParameter",
 "ssm:GetParameters"
],
 "Resource": [
 "arn:aws:ssm:us-east-2:111122223333:parameter/prod-*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "kms:Decrypt",
 "kms:Encrypt",
 "kms:GenerateDataKey" ①
],
 "Resource": [
 "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-12345EXAMPLE"
]
}
]
}

```

<sup>1</sup>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 don't 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 Amazon Resource Name (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": [
 "arn:aws:kms:us-east-2:111122223333:key/abcd1234-ab12-cd34-ef56-
abcdeEXAMPLE"
]
 }
]
}
```

If you require fine-grained access control over the SecureString parameters in your account, you should use a customer managed key 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](#)

## Managing parameter tiers

Parameter Store, a capability of AWS Systems Manager, 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. This is because reverting an advanced parameter to a standard parameter would cause the system to truncate the size of the parameter from 8 KB to 4 KB, resulting in data loss. 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, delete it and recreate it as a new standard parameter.

The following table describes the differences between the tiers.

	Standard	Advanced
Total number of parameters allowed (per AWS account and AWS Region)	10,000	100,000
Maximum size of a parameter value	4 KB	8 KB
Parameter policies available	No	Yes  For more information, see <a href="#">Assigning parameter policies (p. 314)</a> .
Cost	No additional charge	Charges apply  For more information, see <a href="#">AWS Systems Manager Pricing</a> .

### Topics

- [Specifying a default parameter tier \(p. 265\)](#)
- [Changing a standard parameter to an advanced parameter \(p. 271\)](#)

## Specifying a default parameter tier

In requests to create or update a parameter (that is, the [PutParameter](#) operation), you can specify the parameter tier to use in the request. The following is an example, using the AWS Command Line Interface (AWS CLI).

Linux & macOS

```
aws ssm put-parameter \
```

```
--name "default-ami" \
--type "String" \
--value "t2.micro" \
--tier "Standard"
```

## Windows

```
aws ssm put-parameter ^
--name "default-ami" ^
--type "String" ^
--value "t2.micro" ^
--tier "Standard"
```

Whenever you specify a tier in the request, Parameter Store creates or updates the parameter according to your request. However, if you don't 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 creates 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 don't need to change your code to handle the upgrade.

Here are some examples of automatic upgrades:

- Your AWS CloudFormation templates provision numerous parameters when they're run. When this process causes you to reach the 10,000 parameter quota in the standard-parameter tier, Intelligent-Tiering automatically upgrades you to the advanced-parameter tier, and your AWS CloudFormation processes aren't interrupted.
- You store a certificate value in a parameter, rotate the certificate value regularly, and the content is less than the 4 KB quota 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 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 AWS 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 AWS Region in an AWS account. The content size of each parameter can equal a maximum of 4 KB. Standard parameters don't 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** – Use the advanced-parameter tier to create a maximum of 100,000 parameters for each AWS 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, explicitly authorize AWS to charge your account for any advanced parameters you create.

- **Intelligent-Tiering** – With the Intelligent-Tiering option, Parameter Store determines 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 AWS Region in your AWS account, and you don't 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 AWS Region in your AWS account, or you specify a parameter policy, an advanced parameter is created.

#### Note

When you choose Intelligent-Tiering, explicitly authorize AWS to charge your account for any advanced parameters you 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:

- Make sure that you attach the `AdministratorAccess` policy to your IAM user, group, or role.
- Make sure that you have permission to change the default tier setting by using the following API operations:
  - [GetServiceSetting](#)
  - [UpdateServiceSetting](#)
  - [ResetServiceSetting](#)

Use the following procedure to add an inline IAM policy to a user account. This policy allows a user to view and change the default tier setting for parameters in a specific AWS 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:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetServiceSetting"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:UpdateServiceSetting"
],
 "Resource": "arn:aws:ssm:region:account-id:servicesetting/ssm/parameter-store/default-parameter-tier"
 }
]
}
```

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.

```
{
 "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*.

## 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 AWS Region.

### Tip

If you haven't created a parameter yet, you can use the AWS Command Line Interface (AWS CLI) or AWS Tools for Windows PowerShell to change the default parameter tier. For information, see [Specifying or changing the Parameter Store default tier \(AWS CLI\) \(p. 269\)](#) and [Specifying or changing the Parameter Store default tier \(PowerShell\) \(p. 270\)](#).

### To specify or change the Parameter Store default tier

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Change default tier**.
5. Choose one of the following options.
  - **Standard**
  - **Advanced**
  - **Intelligent-Tiering**

For information about these options, see [Specifying a default parameter tier \(p. 265\)](#).

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 CLI to change the default parameter tier setting for the current AWS account and AWS 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 AWS Region in an AWS account.

```
aws ssm update-service-setting --setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/parameter-store/default-parameter-tier --setting-value tier-option
```

*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 \(p. 265\)](#).

There is no output if the command succeeds.

- Run the following command to view the current throughput service settings for Parameter Store in the current AWS account and AWS Region.

```
aws ssm get-service-setting --setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/parameter-store/default-parameter-tier
```

The system returns information similar to the following.

```
{
 "ServiceSetting": {
 "SettingId": "/ssm/parameter-store/default-parameter-tier",
 "SettingValue": "Advanced",
 "LastModifiedDate": 1556551683.923,
 "LastModifiedUser": "arn:aws:sts::123456789012:assumed-role/Administrator/
Jasper",
 "ARN": "arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/parameter-store/
default-parameter-tier",
 "Status": "Customized"
 }
}
```

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 Tools for Windows PowerShell to change the default parameter tier setting for a specific AWS Region in an Amazon Web Services account.

#### To specify or change the Parameter Store default tier using PowerShell

- Change the Parameter Store default tier in the current AWS account and AWS Region using the AWS Tools for PowerShell (Tools for PowerShell).

```
Update-SMSServiceSetting -SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/
parameter-store/default-parameter-tier" -SettingValue "tier-option" -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*.

`tier-option` values include `Standard`, `Advanced`, and `Intelligent-Tiering`. For information about these options, see [Specifying a default parameter tier \(p. 265\)](#).

There is no output if the command succeeds.

- Run the following command to view the current throughput service settings for Parameter Store in the current AWS account and AWS Region.

```
Get-SMSServiceSetting -SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/
parameter-store/default-parameter-tier" -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 system returns information similar to the following.

```
ARN : arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/parameter-store/default-parameter-tier
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 \(p. 279\)](#).

### To change a standard parameter to an advanced parameter

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 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, a capability of AWS Systems Manager, can process. Increased throughput allows you to operate Parameter Store at higher volumes to support applications and workloads that need concurrent access to multiple parameters. You can increase the quota up to the max throughput on the **Settings** tab. For more information about max throughput, see [AWS Systems Manager endpoints and quotas](#). Increasing the throughput quota 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 AWS Region. The throughput setting applies to standard and advanced parameters.

### Topics

- [Configuring permissions to increase Parameter Store throughput \(p. 272\)](#)
- [Increasing throughput \(console\) \(p. 273\)](#)
- [Increasing throughput \(AWS CLI\) \(p. 273\)](#)

- [Increasing throughput \(PowerShell\) \(p. 274\)](#)

## Configuring permissions to increase Parameter Store throughput

Verify that you have permission in IAM to increase Parameter Store throughput by doing one of the following:

- Make sure that the `AdministratorAccess` policy is attached to your IAM user, group, or role.
- Make sure that you have permission to change the throughput service setting by using the following API operations:
  - [GetServiceSetting](#)
  - [UpdateServiceSetting](#)
  - [ResetServiceSetting](#)

Use the following procedure to add an inline IAM policy to a user account. This policy allows 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. Choose **Add inline policy**.
6. Choose the **JSON** tab.
7. Replace the default content with the following:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetServiceSetting"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:UpdateServiceSetting"
],
 "Resource": "arn:aws:ssm:region:account-id:servicesetting/ssm/parameter-store/high-throughput-enabled"
 }
]
}
```

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.

```
{
 "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 AWS Region.

### Tip

If you haven't created a parameter yet, you can use the AWS Command Line Interface (AWS CLI) or AWS Tools for Windows PowerShell to increase throughput. For information, see [Increasing throughput \(AWS CLI\) \(p. 273\)](#) and [Increasing throughput \(PowerShell\) \(p. 274\)](#).

### To increase Parameter Store throughput

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 CLI to increase the number of transactions per second that Parameter Store can process for the current AWS account and AWS 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 AWS Region.

```
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.

- Run the following command to view the current throughput service settings for Parameter Store in the current AWS account and AWS Region.

```
aws ssm get-service-setting --setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/parameter-store/high-throughput-enabled
```

The system returns information similar to the following:

```
{
 "ServiceSetting": {
 "SettingId": "/ssm/parameter-store/high-throughput-enabled",
 "SettingValue": "true",
 "LastModifiedDate": 1556551683.923,
 "LastModifiedUser": "arn:aws:sts::123456789012:assumed-role/Administrator/Jasper",
 "ARN": "arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/parameter-store/high-throughput-enabled",
 "Status": "Customized"
 }
}
```

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.

```
aws ssm reset-service-setting --setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/parameter-store/high-throughput-enabled
```

```
{
 "ServiceSetting": {
 "SettingId": "/ssm/parameter-store/high-throughput-enabled",
 "SettingValue": "false",
 "LastModifiedDate": 1555532818.578,
 "LastModifiedUser": "System",
 "ARN": "arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/parameter-store/high-throughput-enabled",
 "Status": "Default"
 }
}
```

## Increasing throughput (PowerShell)

The following procedure shows how to use the Tools for Windows PowerShell to increase the number of transactions per second that Parameter Store can process for the current AWS account and AWS Region.

### To increase Parameter Store throughput using PowerShell

- Increase Parameter Store throughput in the current AWS account and AWS Region using the AWS Tools for PowerShell (Tools for PowerShell).

```
Update-SSMSERVICESETTING -SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/parameter-store/high-throughput-enabled" -SettingValue "true" -Region region
```

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 AWS Region.

```
Get-SSMSERVICESETTING -SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/parameter-store/high-throughput-enabled" -Region region
```

The system returns information similar to the following:

```
ARN : arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/parameter-store/high-throughput-enabled
LastModifiedDate : 4/29/2019 3:35:44 PM
LastModifiedUser : arn:aws:sts::123456789012:assumed-role/Administrator/Jasper
SettingId : /ssm/parameter-store/high-throughput-enabled
SettingValue : true
Status : Customized
```

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
```

The system returns information similar to the following:

```
ARN : arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/parameter-store/high-throughput-enabled
LastModifiedDate : 4/17/2019 8:26:58 PM
LastModifiedUser : System
SettingId : /ssm/parameter-store/high-throughput-enabled
SettingValue : false
Status : Default
```

## Setting up notifications or trigger actions based on Parameter Store events

The topics in this section explain how to use Amazon EventBridge and Amazon Simple Notification Service (Amazon SNS) to notify you about changes to AWS Systems Manager parameters. You can create an EventBridge rule to notify you when a parameter or a parameter label version is created, updated, or deleted. Events are emitted on a best effort basis. 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 \(p. 314\)](#) and [Managing parameter tiers \(p. 265\)](#).

The topics in this section also explain how to initiate 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 invoke 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. EventBridge also supports running Run Command commands and Automation executions, and actions in many other AWS services. Run Command and Automation are both capabilities of AWS Systems Manager. For more information, see the [Amazon EventBridge 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 EventBridge for parameters \(p. 276\)](#)
- [Configuring EventBridge for parameter policies \(p. 277\)](#)

## Configuring EventBridge for parameters

This topic explains how to create an EventBridge rule that invokes a target based on events that happen to one or more parameters in your AWS account.

### To configure EventBridge for Systems Manager parameters

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**, and then choose **Create rule**.

-or-

If the EventBridge home page opens first, choose **Create rule**.

3. Enter a name and description for the rule.

A rule can't have the same name as another rule in the same Region and on the same event bus.

4. For **Define pattern**, choose **Event pattern**.
5. For **Event matching pattern**, choose **Custom pattern**.
6. For **Event pattern**, paste the following content in the box:

```
{
 "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"
],
 "operation": [
 "Create",
 "Update",
 "Delete",
 "LabelParameterVersion"
]
 }
}
```

7. Modify the contents for the parameters and the operations you want to act on.

For example, the following content means an action is taken when either of the parameters named /Oncall and /Project/Teamlead are updated:

```
{
```

```
"source": [
 "aws.ssm"
],
"detail-type": [
 "Parameter Store Change"
],
"detail": {
 "name": [
 "/Oncall",
 "/Project/Teamlead"
],
 "operation": [
 "Update"
]
}
}
```

8. Choose **Save**.
9. For **Select event bus**, choose the event bus that you want to associate with this rule. If you want this rule to initiate on matching events that come from your own AWS account, select **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
10. For **Select targets**, choose a target type and a supported resource. For example, if you choose **SNS topic**, make a selection for **Topic**. If you choose **CodePipeline**, make a selection for **Pipeline ARN**.
11. Expand any collapsed sections to choose additional options. Collapsible sections vary by target type and include such groups as **Configure input**, **Retry policy and dead-letter queue**, and **Compute options**, among others. Then provide any other configuration details required by the target type you selected.
12. (Optional) Enter one or more tags for the rule. For more information, see [Amazon EventBridge tags](#) in the *Amazon EventBridge User Guide*.
13. Choose **Create**.

## Configuring EventBridge for parameter policies

This topic explains how to create EventBridge 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 \(p. 314\)](#) and [Managing parameter tiers \(p. 265\)](#).

### To configure EventBridge for Systems Manager parameter policies

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**, and then choose **Create rule**.

-or-

If the Amazon EventBridge home page opens first, choose **Create rule**.

3. Enter a name and description for the rule.

A rule can't have the same name as another rule in the same Region and on the same event bus.

4. For **Define pattern**, choose **Event pattern**.
5. For **Event matching pattern**, choose **Custom pattern**.
6. For **Event pattern**, paste the following content in the box:

```
{
 "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"
]
 }
}
```

7. Modify the contents for the parameters and the policy types you want to act on. For example, the following content means an action is taken whenever the parameter named /OncallDuties expires and is deleted:

```
{
 "source": [
 "aws.ssm"
],
 "detail-type": [
 "Parameter Store Policy Action"
],
 "detail": {
 "parameter-name": [
 "/OncallDuties"
],
 "policy-type": [
 "Expiration"
]
 }
}
```

8. Choose **Save**.
9. For **Select event bus**, choose the event bus that you want to associate with this rule. If you want this rule to initiate on matching events that come from your own AWS account, select **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
10. For **Select targets**, choose a target type and a supported resource. For example, if you choose **SNS topic**, make a selection for **Topic**. If you choose **CodePipeline**, make a selection for **Pipeline ARN**.
11. Expand any collapsed sections to choose additional options. Collapsible sections vary by target type and include such groups as **Configure input**, **Retry policy and dead-letter queue**, and **Compute options**, among others. Then provide any other configuration details required by the target type you selected.
12. (Optional) Enter one or more tags for the rule. For more information, see [Amazon EventBridge tags](#) in the *Amazon EventBridge User Guide*.
13. Choose **Create**.

#### Related Information

- [\(Blog post\) Use parameter labels for easy configuration update across environments](#)
- [Tutorial: Use EventBridge to relay events to AWS Systems ManagerRun Command in the Amazon EventBridge User Guide](#)
- [Tutorial: Set AWS Systems Manager Automation as an EventBridge target in the Amazon EventBridge User Guide](#)

## Working with Parameter Store

This section describes how to organize and create tag parameters, and how to create different versions of parameters. You can use the AWS Systems Manager console, the Amazon Elastic Compute Cloud (Amazon EC2) console, or the AWS Command Line Interface (AWS CLI) to create and work with parameters. For more information about parameters, see [What is a parameter? \(p. 258\)](#)

### Topics

- [Creating Systems Manager parameters \(p. 279\)](#)
- [Deleting Systems Manager parameters \(p. 295\)](#)
- [Working with public parameters \(p. 295\)](#)
- [Assigning parameter policies \(p. 314\)](#)
- [Searching for Systems Manager parameters \(p. 320\)](#)
- [Working with parameters using Run Command commands \(p. 322\)](#)
- [Working with parameter hierarchies \(p. 326\)](#)
- [Working with parameter labels \(p. 331\)](#)
- [Working with parameter versions \(p. 338\)](#)
- [Native parameter support for Amazon Machine Image IDs \(p. 340\)](#)

## 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 Command Line Interface (AWS CLI), or AWS Tools for Windows PowerShell (Tools for Windows PowerShell).

This section demonstrates how to create, store, and run parameters with Parameter Store in a test environment. It also demonstrates how to use Parameter Store with other Systems Manager capabilities and AWS services. For more information, see [What is a parameter? \(p. 258\)](#)

### 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\_-

In addition, the slash character ( / ) is used to delineate hierarchies in parameter names. For example: /Dev/Production/East/Project-ABC/MyParameter

- **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-12345abcdeEXAMPLE`.
- **Fully qualified:** When you create or reference a parameter in a hierarchy, include a leading forward slash character (/). When you reference a parameter that is part of a hierarchy, 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 that you create is 1011 characters. This includes the characters in the ARN that precede the name you specify, such as `arn:aws:ssm:us-east-2:111122223333:parameter/`.
- **Prefixes:** A parameter name can't be prefixed with "aws" or "ssm" (case-insensitive). For example, attempts to create parameters with the following names fail with an exception:
  - `awsTestParameter`
  - `SSM-testparameter`
  - `/aws/testparam1`

**Note**

When you specify a parameter in an SSM document, command, or script, include `ssm` as part of the syntax. For example, `{{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, aren't 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 `HierarchyLevelLimitExceededException` 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 AWS Identity and Access Management (IAM) for parameter `/a/b`, they can still call the [GetParametersByPath](#) API operation recursively for `/a` and view `/a/b`.

**Topics**

- [Create a Systems Manager parameter \(console\) \(p. 281\)](#)
- [Create a Systems Manager parameter \(AWS CLI\) \(p. 282\)](#)

- [Create a Systems Manager parameter \(Tools for Windows PowerShell\) \(p. 292\)](#)

## Create a Systems Manager parameter (console)

You can use the AWS Systems Manager console to create and run `String`, `StringList`, and `SecureString` parameter types. After deleting a parameter, wait for at least 30 seconds to create a parameter with the same name.

### Note

Parameters are only available in the AWS Region where they were created.

The following procedure walks you through the process of creating a parameter in the Parameter Store console. You can create `String`, `StringList` parameter types from the console.

### To create a parameter

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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. In the **Name** box, enter a hierarchy and a name. For example, enter `/Test/helloWorld`.

For more information about parameter hierarchies, see [Working with parameter hierarchies \(p. 326\)](#).

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 \(p. 265\)](#).
7. For **Type**, choose `String`, `StringList`, or `SecureString`.
  - If you choose `String`, the **Data type** field is displayed. If you're creating a parameter to hold the resource ID for an Amazon Machine Image (AMI), select `aws:ec2:image`. Otherwise, keep the default `text` selected.
  - If you choose `SecureString`, the **KMS Key ID** field is displayed. If you don't provide an AWS Key Management Service AWS KMS key ID, an AWS KMS key Amazon Resource Name (ARN), an alias name, or an alias ARN, then the system uses `alias/aws/ssm`, which is the AWS managed key for Systems Manager. If you don't want to use this key, then you can use a customer managed key. For more information about AWS managed and customer managed keys, 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](#).

### Important

Parameter Store only supports [symmetric encryption KMS keys](#). You can't use an [asymmetric encryption KMS key](#) to encrypt your parameters. For help determining whether a KMS key is symmetric or asymmetric, see [Identifying symmetric and asymmetric KMS keys](#) in the *AWS Key Management Service Developer Guide*

- When creating a `SecureString` parameter in the console by using the `key-id` parameter with either a customer managed key alias name or an alias ARN, specify the prefix `alias/` before the alias. Following is an ARN example.

```
arn:aws:kms:us-east-2:123456789012:alias/abcd1234-ab12-cd34-ef56-abcdeEXAMPLE
```

Following 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**.

**Note**

Parameters can't be referenced or nested in the values of other parameters. You can't include `{}` or `{}ssm:parameter-name` in a parameter value.

If you chose **SecureString**, the value of the parameter is masked by default ("\*\*\*\*\*") when you view it later on the parameter **Overview** tab. Choose **Show** to display the parameter value.



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 allow 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, delete it and recreate it as a new standard parameter.

## Create a Systems Manager parameter (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to create **String**, **StringList**, and **SecureString** parameter types. After deleting a parameter, wait for at least 30 seconds to create a parameter with the same name.

Parameters can't be referenced or nested in the values of other parameters. You can't include `{}` or `{}ssm:parameter-name` in a parameter value.

**Note**

Parameters are only available in the AWS Region where they were created.

### Topics

- [Create a String parameter \(AWS CLI\) \(p. 283\)](#)

- [Create a StringList parameter \(AWS CLI\) \(p. 286\)](#)
- [Create a SecureString parameter \(AWS CLI\) \(p. 287\)](#)
- [Create a multi-line parameter \(AWS CLI\) \(p. 291\)](#)

### Create a String parameter (AWS CLI)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to create a String-type parameter.

Linux & macOS

```
aws ssm put-parameter \
--name "parameter-name" \
--value "parameter-value" \
--type String \
--tags "Key=tag-key,Value=tag-value"
```

Windows

```
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 & macOS

```
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 --name option supports hierarchies. For information about hierarchies, see [Working with parameter hierarchies \(p. 326\)](#).

The --data-type option must be specified only if you are creating a parameter that contains an AMI ID. It validates that the parameter value you enter is a properly formatted Amazon Elastic

Compute Cloud (Amazon EC2) AMI ID. For all other parameters, the default data type is `text` and it's optional to specify a value. For more information, see [Native parameter support for Amazon Machine Image IDs \(p. 340\)](#).

**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 doesn't mean that the parameter value has been validated yet. For more information, see [Native parameter support for Amazon Machine Image IDs \(p. 340\)](#).

The following example adds two key-value pair tags to a parameter.

Linux & macOS

```
aws ssm put-parameter \
--name parameter-name \
--value "parameter-value" \
--type "String" \
--tags '[{"Key": "Region", "Value": "East"}, {"Key": "Environment", "Value": "Production"}]'
```

Windows

```
aws ssm put-parameter ^
--name parameter-name ^
--value "parameter-value" ^
--type "String" ^
--tags [{"Key": "Region1", "Value": "East1"}, {"Key": "Environment1", "Value": "Production1"}]
```

The following example uses a parameter hierarchy in the name to create a plaintext `String` parameter. It returns the version number of the parameter. For more information about parameter hierarchies, see [Working with parameter hierarchies \(p. 326\)](#).

Linux & macOS

**Parameter not in a hierarchy**

```
aws ssm put-parameter \
--name "golden-ami" \
--type "String" \
--value "ami-12345abcdeEXAMPLE"
```

**Parameter in a hierarchy**

```
aws ssm put-parameter \
--name "/amis/linux/golden-ami" \
--type "String" \
--value "ami-12345abcdeEXAMPLE"
```

Windows

**Parameter not in a hierarchy**

```
aws ssm put-parameter ^
--name "golden-ami" ^
--type "String" ^
```

```
--value "ami-12345abcdeEXAMPLE"
```

### Parameter in a hierarchy

```
aws ssm put-parameter ^
--name "/amis/windows/golden-ami" ^
--type "String" ^
--value "ami-12345abcdeEXAMPLE"
```

- Run the following command to view the latest parameter value and verify the details of your new parameter.

```
aws ssm get-parameters --names "/Test/IAD/helloWorld"
```

The system returns information like the following.

```
{
 "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"
 }
]
}
```

Run the following command to change the parameter value. It returns the version number of the parameter.

```
aws ssm put-parameter --name "/Test/IAD/helloWorld" --value "My updated 1st parameter" --
type String --overwrite
```

Run the following command to view the parameter value history.

```
aws ssm get-parameter-history --name "/Test/IAD/helloWorld"
```

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"
```

Run the following command if you only want to retrieve the parameter Value.

```
aws ssm get-parameter --name testDataTypeParameter --query "Parameter.Value"
```

Run the following command if you only want to retrieve the parameter Value using get-parameters.

```
aws ssm get-parameters --names "testDataTypeParameter" --query "Parameters[*].Value"
```

Run the following command to view the parameter metadata.

```
aws ssm describe-parameters --filters "Key=Name,Values=/Test/IAD/helloWorld"
```

**Note**

*Name* must be capitalized.

The system returns information like the following.

```
{
 "Parameters": [
 {
 "Name": "helloworld",
 "Type": "String",
 "LastModifiedUser": "arn:aws:iam::123456789012:user/JohnDoe",
 "LastModifiedDate": 1494529763.156,
 "Version": 1,
 "Tier": "Standard",
 "Policies": []
 }
]
}
```

### Create a `StringList` parameter (AWS CLI)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to create a parameter.

Linux & macOS

```
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 & macOS

```
aws ssm put-parameter \
 --name "ParameterGroup/HelloWorld" \
 --value "Hello,World!" \
 --type StringList \
 --tags "Key=tag-key,Value=tag-value"
```

```
--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 characters to escape items in the list. If you have a parameter value that requires a comma, then use the `String` type.

- Run the `get-parameters` command to verify the details of the parameter. For example:

```
aws ssm get-parameters --name "/IAD/ERP/Oracle/addUsers"
```

## Create a SecureString parameter (AWS CLI)

Use the following procedure to create a `SecureString` parameter.

### Important

Only the *value* of a `SecureString` parameter is encrypted. Parameter names, descriptions, and other properties aren't encrypted.

### Important

Parameter Store only supports [symmetric encryption KMS keys](#). You can't use an [asymmetric encryption KMS key](#) to encrypt your parameters. For help determining whether a KMS key is symmetric or asymmetric, see [Identifying symmetric and asymmetric KMS keys](#) in the *AWS Key Management Service Developer Guide*

- Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
- Run **one** of the following commands to create a parameter that uses the `SecureString` data type.

### Linux & macOS

#### Create a `SecureString` parameter using the default AWS managed key

```
aws ssm put-parameter \
--name "parameter-name" \
--value "parameter-value" \
--type "SecureString"
```

#### Create a `SecureString` parameter that uses a customer managed key

```
aws ssm put-parameter \
--name "parameter-name" \
--value "a-parameter-value, for example P@ssW%rd#1" \
--type "SecureString" \
--tags "Key=tag-key,Value=tag-value"
```

#### 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-account-ID/the-custom-AWS KMS-key" \
--tags "Key=tag-key,Value=tag-value"
```

## Windows

### Create a SecureString parameter using the default AWS managed key

```
aws ssm put-parameter ^
--name "parameter-name" ^
--value "parameter-value" ^
--type "SecureString"
```

### Create a SecureString parameter that uses a customer managed key

```
aws ssm put-parameter ^
--name "parameter-name" ^
--value "a-parameter-value, for example P@ssW%rd#1" ^
--type "SecureString" ^
--tags "Key=tag-key,Value=tag-value"
```

### 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 " "
--tags "Key=tag-key,Value=tag-value"account-ID/the-custom-AWS KMS-key"
```

If you create a `SecureString` parameter by using the AWS-managed AWS Key Management Service (AWS KMS) key in your account and Region, then you *don't* have to provide a value for the `--key-id` parameter.

#### Note

To use the AWS KMS key assigned to your AWS account and AWS Region, remove the `key-id` parameter from the command. For more information about AWS KMS keys, see [AWS Key Management Service Concepts](#) in the [AWS Key Management Service Developer Guide](#).

To use a customer managed key instead of the AWS managed key assigned to your account, specify the key by using the `--key-id` parameter. The parameter supports the following KMS parameter formats.

- Key Amazon Resource Name (ARN) example:

`arn:aws:kms:us-east-2:123456789012:key/key-id`

- Alias ARN example:

`arn:aws:kms:us-east-2:123456789012:alias/alias-name`

- Key ID example:

`12345678-1234-1234-1234-123456789012`

- Alias Name example:

```
alias/MyAliasName
```

You can create a customer managed key by using the AWS Management Console or the AWS KMS API. The following AWS CLI commands create a customer managed key in the current AWS 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.

The following example uses an obfuscated name (`3l3vat3131`) for a password parameter and an AWS KMS key.

Linux & macOS

```
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

```
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
```

3. Run the following command to verify the details of the parameter.

If you don't specify the `with-decryption` parameter, or if you specify the `no-with-decryption` parameter, the command returns an encrypted GUID.

Linux & macOS

```
aws ssm get-parameters \
 --name "the-parameter-name-you-specified" \
 --with-decryption
```

Windows

```
aws ssm get-parameters ^
 --name "the-parameter-name-you-specified" ^
 --with-decryption
```

4. Run the following command to view the parameter metadata.

Linux & macOS

```
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"
```

5. Run the following command to change the parameter value if you're **not** using a customer managed AWS KMS key.

Linux & macOS

```
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 of the following commands to change the parameter value if you **are** using a customer managed AWS KMS key.

Linux & macOS

```
aws ssm put-parameter \
--name "the-name-that-you-specified" \
--value "a-new-parameter-value" \
--type "SecureString" \
--key-id "the-KMSkey-ID" \
--overwrite
```

```
aws ssm put-parameter \
--name "the-name-that-you-specified" \
--value "a-new-parameter-value" \
--type "SecureString" \
--key-id "account-alias/the-KMSkey-ID" \
--overwrite
```

Windows

```
aws ssm put-parameter ^
--name "the-name-that-you-specified" ^
--value "a-new-parameter-value" ^
--type "SecureString" ^
--key-id "the-KMSkey-ID" ^
--overwrite
```

```
aws ssm put-parameter ^
--name "the-name-that-you-specified" ^
```

```
--value "a-new-parameter-value" ^
--type "SecureString" ^
--key-id "account-alias/the-KMSkey-ID" ^
--overwrite
```

6. Run the following command to view the latest parameter value.

Linux & macOS

```
aws ssm get-parameters \
--name "the-name-that-you-specified" \
--with-decryption
```

Windows

```
aws ssm get-parameters ^
--name "the-name-that-you-specified" ^
--with-decryption
```

7. Run the following command to view the parameter value history.

Linux & macOS

```
aws ssm get-parameter-history \
--name "the-name-that-you-specified"
```

Windows

```
aws ssm get-parameter-history ^
--name "the-name-that-you-specified"
```

### 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 is doubly encrypted.

By default, all `SecureString` values are displayed as cipher-text. To decrypt a `SecureString` value, a user must have permission to call the AWS KMS [Decrypt](#) API operation. For information about configuring AWS KMS access control, see [Authentication and Access Control for AWS KMS](#) in the [AWS Key Management Service Developer Guide](#).

### Important

If you change the KMS key alias for the KMS key used to encrypt a parameter, then you must also update the key alias the parameter uses to reference AWS KMS. This only applies to the KMS key alias; the key ID that an alias attaches to stays the same unless you delete the whole key.

### Create a multi-line parameter (AWS CLI)

You can use the AWS CLI to create a parameter with line breaks. Use line breaks to break up the text in longer parameter values for better legibility or, for example, update multi-paragraph parameter content for a web page. You can include the content in a JSON file and use the `--cli-input-json` option, using line break characters like `\n`, as shown in the following example.

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to create a multi-line parameter.

## Linux & macOS

```
aws ssm put-parameter \
 --name "MultiLineParameter" \
 --type String \
 --cli-input-json file://MultiLineParameter.json
```

## Windows

```
aws ssm put-parameter ^
 --name "MultiLineParameter" ^
 --type String ^
 --cli-input-json file://MultiLineParameter.json
```

The following example shows the contents of the file `MultiLineParameter.json`.

```
{
 "Value": "<para>Paragraph One</para>\n<para>Paragraph Two</para>\n<para>Paragraph
Three</para>"
}
```

The saved parameter value is stored as follows.

```
<para>Paragraph One</para>
<para>Paragraph Two</para>
<para>Paragraph Three</para>
```

## Create a Systems Manager parameter (Tools for Windows PowerShell)

You can use AWS Tools for Windows PowerShell to create `String`, `StringList`, and `SecureString` parameter types. After deleting a parameter, wait for at least 30 seconds to create a parameter with the same name.

Parameters can't be referenced or nested in the values of other parameters. You can't include `{ { } }` or `{ { ssm:parameter-name } }` in a parameter value.

### Note

Parameters are only available in the AWS Region where they were created.

### Topics

- [Create a String parameter \(Tools for Windows PowerShell\) \(p. 292\)](#)
- [Create a StringList parameter \(Tools for Windows PowerShell\) \(p. 293\)](#)
- [Create a SecureString parameter \(Tools for Windows PowerShell\) \(p. 294\)](#)

## Create a String parameter (Tools for Windows PowerShell)

1. Install and configure the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to create a parameter that contains a plain text value.

```
Write-SMSPParameter ^
 -Name "parameter-name" ^
```

```
-Value "parameter-value" `
-Type "String"
```

-or-

Run the following command to create a parameter that contains an Amazon Machine Image (AMI) ID as the parameter value.

**Note**

To create a parameter with a tag, create the service.model.tag before hand as a variable. Here is an example.

```
$tag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag.Key = "tag-key"
$tag.Value = "tag-value"
```

```
Write-SMSPParameter `
-Name "parameter-name" `
-Value "an-AMI-id" `
-Type "String" `
-DataType "aws:ec2:image" `
-Tags $tag
```

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 \(p. 340\)](#).

Here is an example that uses a parameter hierarchy.

```
Write-SMSPParameter `
-Name "/IAD/Web/SQL/IPaddress" `
-Value "99.99.99.999" `
-Type "String" `
-Tags $tag
```

3. Run the following command to verify the details of the parameter.

```
(Get-SMSPParameterValue -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 haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to create a StringList parameter.

**Note**

To create a parameter with a tag, create the service.model.tag before hand as a variable. Here is an example.

```
$tag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag.Key = "tag-key"
$tag.Value = "tag-value"
```

```
Write-SMSPParameter `
-Name "parameter-name" `
```

```
-Value "a-comma-separated-list-of-values"
-Type "StringList"
-Tags $tag
```

If successful, the command returns the version number of the parameter.

Here is an example.

```
Write-SMSPParameter ^
-Name "stringlist-parameter" ^
-Value "Milana,Mariana,Mark,Miguel" ^
-Type "StringList"
-Tags $tag
```

**Note**

Items in a StringList must be separated by a comma (,). You can't use other punctuation or special characters 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-SMSPParameterValue -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 [Create a SecureString parameter \(AWS CLI\) \(p. 287\)](#).

**Important**

Only the *value* of a SecureString parameter is encrypted. Parameter names, descriptions, and other properties aren't encrypted.

**Important**

Parameter Store only supports [symmetric encryption KMS keys](#). You can't use an [asymmetric encryption KMS key](#) to encrypt your parameters. For help determining whether a KMS key is symmetric or asymmetric, see [Identifying symmetric and asymmetric KMS keys in the AWS Key Management Service Developer Guide](#)

1. Install and configure the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to create a parameter.

**Note**

To create a parameter with a tag, first create the service.model.tag as a variable. Here is an example.

```
$tag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag.Key = "tag-key"
$tag.Value = "tag-value"
```

```
Write-SMSPParameter ^
-Name "parameter-name" ^
-Value "parameter-value" ^
-Type "SecureString"
-KeyId "an AWS KMS key ID, an AWS KMS key ARN, an alias name, or an alias ARN" ^
-Tags $tag
```

If successful, the command returns the version number of the parameter.

**Note**

To use the AWS managed AWS KMS key 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 AWS KMS key.

```
Write-SMSPParameter `~
 -Name "/Finance/Payroll/3l3vat3131" `~
 -Value "P@sSwW)rd" `~
 -Type "SecureString" `~
 -Tags $tag
```

3. Run the following command to verify the details of the parameter.

```
(Get-SSMParameterValue -Name "the-parameter-name-you-specified" -WithDecryption
$true).Parameters
```

By default, all `SecureString` values are displayed as cipher-text. To decrypt a `SecureString` value, a user must have permission to call the AWS KMS `Decrypt` API operation. For information about configuring AWS KMS access control, see [Authentication and Access Control for AWS KMS](#) in the [AWS Key Management Service Developer Guide](#).

**Important**

If you change the KMS key alias for the KMS key used to encrypt a parameter, then you must also update the key alias the parameter uses to reference AWS KMS. This only applies to the KMS key alias; the key ID that an alias attaches to stays the same unless you delete the whole key.

## Deleting Systems Manager parameters

This topic describes how to delete parameters that you have created in Parameter Store, a capability of AWS Systems Manager.

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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. On the **My parameters** tab, select the check box next to each parameter to delete.
4. Choose **Delete**.
5. On the confirmation dialog, choose **Delete parameters**.

## Working with public parameters

Some AWS services publish information about common artifacts as AWS Systems Manager *public* parameters. For example, the Amazon Elastic Compute Cloud (Amazon EC2) service publishes information about Amazon Machine Images (AMIs) as public parameters.

You can call this information from your scripts and code by using the [DescribeParameters](#), [GetParametersByPath](#), [GetParameter](#), and [GetParameters](#) API operations.

## Related blog posts

- [Query for AWS Regions, Endpoints, and More Using AWS Systems ManagerParameter Store](#)
- [Query for the latest Amazon Linux AMI IDs using AWS Systems ManagerParameter Store](#)
- [Query for the Latest Windows AMI Using AWS Systems ManagerParameter Store](#)

## Topics

- [Finding public parameters \(p. 296\)](#)
- [Calling AMI public parameters \(p. 299\)](#)
- [Calling the ECS optimized AMI public parameter \(p. 304\)](#)
- [Calling the EKS optimized AMI public parameter \(p. 304\)](#)
- [Calling public parameters for AWS services, Regions, endpoints, Availability Zones, local zones, and Wavelength Zones \(p. 305\)](#)

## Finding public parameters

You can search for public parameters using the Parameter Store console or the AWS Command Line Interface. A public parameter name begins with `aws/service/list`. The next part of the name corresponds to the service that owns that parameter.

The following is a list of some services which provide public parameters:

- `ami-al-latest`
- `ami-amazon-linux-latest`
- `ami-windows-latest`
- `aws-storage-gateway-latest`
- `bottlerocket`
- `canonical`
- `datasync`
- `debian`
- `ecs`
- `global-infrastructure`
- `redhat`
- `storagegateway`
- `suse`

All public parameters aren't published to all AWS Regions.

### [Finding public parameters using the AWS CLI](#)

Use `describe-parameters` for discovery of public parameters. Use `get-parameters-by-path` to get the actual path for a service listed under `/aws/service/list`. To get the service's path, remove `/list` from the path. For example, `/aws/service/list/ecs` becomes `/aws/service/ecs`.

To retrieve a list of public parameters owned by different services in Parameter Store, run the following command.

```
aws ssm get-parameters-by-path --path /aws/service/list
```

The following example output has been truncated for space.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/list/ami-al-latest",
 "Type": "String",
 "Value": "/aws/service/ami-al-latest/",
 "Version": 1,
 "LastModifiedDate": "2021-01-29T10:25:10.902000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/list/ami-al-latest",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/list/ami-windows-latest",
 "Type": "String",
 "Value": "/aws/service/ami-windows-latest/",
 "Version": 1,
 "LastModifiedDate": "2021-01-29T10:25:12.567000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/list/ami-windows-latest",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/list/aws-storage-gateway-latest",
 "Type": "String",
 "Value": "/aws/service/aws-storage-gateway-latest/",
 "Version": 1,
 "LastModifiedDate": "2021-01-29T10:25:09.903000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/list/aws-storage-gateway-
latest",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/list/global-infrastructure",
 "Type": "String",
 "Value": "/aws/service/global-infrastructure/",
 "Version": 1,
 "LastModifiedDate": "2021-01-29T10:25:11.901000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/list/global-
infrastructure",
 "DataType": "text"
 }
]
}
```

If you want to view parameters owned by a specific service, choose the service from the list that was produced after running the earlier command. Then, make a `get-parameters-by-path` call using the name of your desired service. For example, `/aws/service/global-infrastructure`. The path might be one-level (only calls parameters that match the exact values given) or recursive (contains elements in the path beyond what you have given). If no results are returned for the service you specify, add the `--recursive` flag and run the command again.

```
aws ssm get-parameters-by-path --path /aws/service/global-infrastructure
```

This returns all parameters owned by `global-infrastructure`.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/global-infrastructure/current-region",
 "Type": "String",
 "LastModifiedDate": "2019-06-21T05:15:34.252000-07:00",
 "Version": 1,
 "Tier": "Standard",
 "Value": "us-east-1"
 }
]
}
```

```
 "Policies": [],
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/version",
 "Type": "String",
 "LastModifiedDate": "2019-02-04T06:59:32.875000-08:00",
 "Version": 1,
 "Tier": "Standard",
 "Policies": [],
 "DataType": "text"
 }
]
```

You can also view parameters owned by a specific service by using the `Option=BeginsWith` filter.

```
aws ssm describe-parameters --parameter-filters "Key=Name, Option=BeginsWith, Values=/aws/service/ami-amazon-linux-latest"
```

The command returns information like the following. This example output has been truncated for space.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-ebs",
 "Type": "String",
 "LastModifiedDate": "2021-01-26T13:39:40.686000-08:00",
 "Version": 25,
 "Tier": "Standard",
 "Policies": [],
 "DataType": "text"
 },
 {
 "Name": "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2",
 "Type": "String",
 "LastModifiedDate": "2021-01-26T13:39:40.807000-08:00",
 "Version": 25,
 "Tier": "Standard",
 "Policies": [],
 "DataType": "text"
 },
 {
 "Name": "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-s3",
 "Type": "String",
 "LastModifiedDate": "2021-01-26T13:39:40.920000-08:00",
 "Version": 25,
 "Tier": "Standard",
 "Policies": [],
 "DataType": "text"
 }
]
}
```

#### Note

The returned parameters might be different when you use `Option=BeginsWith` because it uses a different search pattern.

#### [Finding public parameters using the Parameter Store console](#)

You must have at least one parameter in your AWS account and AWS Region before you can search for public parameters using the console.

## To find public parameters using the console

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Public parameters** tab.
4. Choose the **Select a service** dropdown. Choose the service whose parameters you want to use.
5. Filter the parameters owned by the service you selected by entering more information into the search bar.
6. Choose the public parameter you want to use.

## Calling AMI public parameters

Amazon Elastic Compute Cloud (Amazon EC2) Amazon Machine Image (AMI) public parameters are available for Amazon Linux, Amazon Linux 2, and Windows Server from the following paths:

- Amazon Linux and Amazon Linux 2: /aws/service/ami-amazon-linux-latest
- Windows Server: /aws/service/ami-windows-latest

### Calling AMI public parameters for Amazon Linux and Amazon Linux 2

You can view a list of all Amazon Linux and Amazon Linux 2 AMIs in the current AWS Region by using the following command in the AWS Command Line Interface (AWS CLI).

Linux & macOS

```
aws ssm get-parameters-by-path \
 --path /aws/service/ami-amazon-linux-latest \
 --query 'Parameters[].[Name]'
```

Windows

```
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-x86_64-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/amzn-ami-minimal-hvm-x86_64-ebs",
 "/aws/service/ami-amazon-linux-latest/amzn-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 & macOS

```
aws ssm get-parameters-by-path \
--path "/aws/service/ami-amazon-linux-latest" \
--region region
```

#### Windows

```
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.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-ebs",
 "Type": "String",
 "Value": "ami-02f31e7644d23a001",
 "Version": 32,
 "LastModifiedDate": "2021-10-04T14:51:40.313000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-latest/
amzn-ami-hvm-x86_64-ebs",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2",
 "Type": "String",
 "Value": "ami-0a787ac2e0c399e8b",
 "Version": 32,
 "LastModifiedDate": "2021-10-04T14:51:40.424000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-latest/
amzn-ami-hvm-x86_64-gp2",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-s3",
 "Type": "String",
 "Value": "ami-0437136c909273ff3",
 "Version": 32,
 "LastModifiedDate": "2021-10-04T14:51:40.533000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-latest/
amzn-ami-hvm-x86_64-s3",
 "DataType": "text"
 }
]
}
```

```
}
```

You can view details of a specific AMI by using the [GetParameters](#) API operation with the full AMI name, including the path. Here is an example command.

Linux & macOS

```
aws ssm get-parameters \
--names /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2 \
--region us-east-2
```

Windows

```
aws ssm get-parameters ^
--names /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2 ^
--region us-east-2
```

The command returns the following information.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
 "Type": "String",
 "Value": "ami-074cce78125f09d61",
 "Version": 51,
 "LastModifiedDate": "2021-10-06T16:50:43.294000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-latest/
amzn2-ami-hvm-x86_64-gp2",
 "DataType": "text"
 }
],
 "InvalidParameters": []
}
```

### Calling AMI public parameters for Windows Server

You can view a list of all Windows Server AMIs in the current AWS Region by using the following command in the AWS CLI.

Linux & macOS

```
aws ssm get-parameters-by-path \
--path /aws/service/ami-windows-latest \
--query 'Parameters[].Name'
```

Windows

```
aws ssm get-parameters-by-path ^
--path /aws/service/ami-windows-latest ^
--query Parameters[].Name
```

The command returns information like the following. This example output has been truncated for space.

```
[
```

```

"/aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-Chinese_Simplified-64Bit-
Base",
"/aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-Chinese_Traditional-64Bit-
Base",
"/aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-Dutch-64Bit-Base",
"/aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-Hungarian-64Bit-Base",
"/aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-Japanese-64Bit-Base",
"/aws/service/ami-windows-latest/Windows_Server-2016-English-Core-Containers",
"/aws/service/ami-windows-latest/Windows_Server-2016-German-Full-Base",
"/aws/service/ami-windows-latest/Windows_Server-2016-Japanese-Full-SQL_2017_Web",
"/aws/service/ami-windows-latest/Windows_Server-2019-English-Core-EKS_Optimized-1.17",
"/aws/service/ami-windows-latest/amzn2-ami-hvm-2.0.20191217.0-x86_64-gp2-mono",
"/aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-English-64Bit-
SQL_2016_SP2_Standard",
"/aws/service/ami-windows-latest/Windows_Server-2012-RTM-Italian-64Bit-Base",
"/aws/service/ami-windows-latest/Windows_Server-2016-English-Deep-Learning",
"/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-
SQL_2014_SP3_Enterprise",
"/aws/service/ami-windows-latest/Windows_Server-2019-English-Core-EKS_Optimized-1.21",
"/aws/service/ami-windows-latest/Windows_Server-2019-Italian-Full-Base",
"/aws/service/ami-windows-latest/Windows_Server-2019-Japanese-Full-
SQL_2017_Enterprise",
"/aws/service/ami-windows-latest/Windows_Server-2022-Italian-Full-Base",
"/aws/service/ami-windows-latest/Windows_Server-2022-Portuguese_Brazil-Full-Base",
"/aws/service/ami-windows-latest/Windows_Server-20H2-English-Core-ContainersLatest",
"/aws/service/ami-windows-latest/EC2LaunchV2_Preview-Windows_Server-2019-English-Core-
Base",
]

```

You can view details about these AMIs, including the AMI IDs and Amazon Resource Names (ARNs), by using the following command.

#### Linux & macOS

```
aws ssm get-parameters-by-path \
--path "/aws/service/ami-windows-latest" \
--region region
```

#### Windows

```
aws ssm get-parameters-by-path ^
--path "/aws/service/ami-windows-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.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-
Chinese_Simplified-64Bit-Base",
 "Type": "String",
 "Value": "ami-085a8792434781696",
 "Version": 72,
 "LastModifiedDate": "2021-11-11T16:29:44.401000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-windows-latest/
Windows_Server-2012-R2_RTM-Chinese_Simplified-64Bit-Base",
 }
]
}
```

```

 "DataType": "text"
 },
 {
 "Name": "/aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-
Chinese_Traditional-64Bit-Base",
 "Type": "String",
 "Value": "ami-01d65ee0d6d795b37",
 "Version": 72,
 "LastModifiedDate": "2021-11-11T16:30:04.345000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-windows-latest/
Windows_Server-2012-R2_RTM-Chinese_Traditional-64Bit-Base",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-
Dutch-64Bit-Base",
 "Type": "String",
 "Value": "ami-0ddf2b503c5585e35",
 "Version": 72,
 "LastModifiedDate": "2021-11-11T16:30:16.207000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-windows-latest/
Windows_Server-2012-R2_RTM-Dutch-64Bit-Base",
 "DataType": "text"
 }
]
}

```

You can view details of a specific AMI by using the [GetParameters](#) API operation with the full AMI name, including the path. Here is an example command.

#### Linux & macOS

```
aws ssm get-parameters \
--names /aws/service/ami-windows-latest/Windows_Server-2016-English-Core-Containers
\
--region us-east-2
```

#### Windows

```
aws ssm get-parameters ^
--names /aws/service/ami-windows-latest/Windows_Server-2016-English-Core-Containers
^
--region us-east-2
```

The command returns the following information.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/ami-windows-latest/Windows_Server-2016-English-Core-
Containers",
 "Type": "String",
 "Value": "ami-0581eb234ac9bf3ec",
 "Version": 68,
 "LastModifiedDate": "2021-11-11T16:40:51.934000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-windows-latest/
Windows_Server-2016-English-Core-Containers",
 "DataType": "text"
 }
],
 "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 Amazon Machine Images (AMIs) as public parameters. Users are encouraged to use this AMI when creating a new Amazon Elastic Compute Cloud (Amazon EC2) cluster for Amazon ECS because the optimized AMIs include bug fixes and feature updates.

Use the following command to view the name of the latest Amazon ECS optimized AMI for Amazon Linux 2. To see commands for other operating systems, see [Retrieving Amazon ECS-Optimized AMI metadata](#) in the *Amazon Elastic Container Service Developer Guide*.

Linux & macOS

```
aws ssm get-parameters \
--names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended
```

Windows

```
aws ssm get-parameters ^
--names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended
```

The command returns information like the following.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/recommended",
 "Type": "String",
 "Value": "{\"schema_version\":1,\"image_name\":\"amzn2-ami-ecs-hvm-2.0.20210929-x86_64-ebs\",\"image_id\":\"ami-0c38a2329ed4dae9a\",\"os\":\"Amazon Linux 2\",\"ecs_runtime_version\":\"Docker version 20.10.7\",\"ecs_agent_version\":\"1.55.4\"}",
 "Version": 73,
 "LastModifiedDate": "2021-10-06T16:35:10.004000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/amazon-linux-2/recommended",
 "DataType": "text"
 }
],
 "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 Amazon Machine Image (AMI) as a public parameter. We encourage you to use this AMI when adding nodes to an Amazon EKS cluster, as new releases include Kubernetes patches and security updates. Previously, to guarantee 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 for Amazon Linux 2.

Linux & macOS

```
aws ssm get-parameters \
```

```
--names /aws/service/eks/optimized-ami/1.14/amazon-linux-2/recommended
```

## Windows

```
aws ssm get-parameters ^
--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-08984d8491de17ca0\",
\"image_name\":\"amazon-eks-node-1.14-v20201007\",\"release_version\":
\"1.14.9-20201007\"}",
 "Version": 24,
 "LastModifiedDate": "2020-11-17T10:16:09.971000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/eks/optimized-ami/1.14/
amazon-linux-2/recommended",
 "DataType": "text"
 }
],
 "InvalidParameters": []
}
```

## Calling public parameters for AWS services, Regions, endpoints, Availability Zones, local zones, and Wavelength Zones

You can call the AWS Region, service, endpoint, Availability, and Wavelength Zones of public parameters by using the following path.

```
/aws/service/global-infrastructure
```

### View active AWS Regions

You can view a list of all active AWS Regions by using the following command in the AWS Command Line Interface (AWS CLI).

#### Linux & macOS

```
aws ssm get-parameters-by-path \
--path /aws/service/global-infrastructure/regions \
--query 'Parameters[].Name'
```

#### Windows

```
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/sa-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/eu-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 & macOS

```
aws ssm get-parameters-by-path \
--path /aws/service/global-infrastructure/services \
--query 'Parameters[].Name | sort(@)'
```

Windows

```
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/accessanalyzer",
"/aws/service/global-infrastructure/services/account",
"/aws/service/global-infrastructure/services/acm",
"/aws/service/global-infrastructure/services/acm-pca",
"/aws/service/global-infrastructure/services/ahl",
"/aws/service/global-infrastructure/services/aiq",
"/aws/service/global-infrastructure/services/alexaforbusiness",
"/aws/service/global-infrastructure/services/amazonlocationservice",
"/aws/service/global-infrastructure/services/amp",
"/aws/service/global-infrastructure/services/amplify",
"/aws/service/global-infrastructure/services/amplifybackend",
"/aws/service/global-infrastructure/services/apigateway",
"/aws/service/global-infrastructure/services/apigatewaymanagementapi",
"/aws/service/global-infrastructure/services/apigatewayv2",
"/aws/service/global-infrastructure/services/appconfig",
```

```
"/aws/service/global-infrastructure/services/appconfigdata",
"/aws/service/global-infrastructure/services/appflow",
"/aws/service/global-infrastructure/services/appintegrations",
"/aws/service/global-infrastructure/services/application-autoscaling",
"/aws/service/global-infrastructure/services/application-insights",
"/aws/service/global-infrastructure/services/applicationcostprofiler",
"/aws/service/global-infrastructure/services/appmesh",
"/aws/service/global-infrastructure/services/apprunner",
```

### **View supported Regions for an AWS service**

You can view a list of AWS Regions where a service is available. This example uses AWS Systems Manager (ssm).

Linux & macOS

```
aws ssm get-parameters-by-path \
--path /aws/service/global-infrastructure/services/ssm/regions \
--query 'Parameters[].Value'
```

Windows

```
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-northeast-3",
 "ap-south-1",
 "cn-north-1",
 "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-southeast-1",
 "ap-southeast-3",
 "ca-central-1",
 "eu-north-1",
 "eu-south-1",
 "us-gov-east-1",
 "us-west-1",
 "ap-northeast-2",
 "ap-southeast-2",
 "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 & macOS

```
aws ssm get-parameter \
--name /aws/service/global-infrastructure/regions/us-east-2/services/ssm/endpoint \
--query 'Parameter.Value'
```

Windows

```
aws ssm get-parameter ^
--name /aws/service/global-infrastructure/regions/us-east-2/services/ssm/endpoint ^
--query Parameter.Value
```

The command returns information like the following.

```
"ssm.us-east-2.amazonaws.com"
```

**View complete Availability Zone details**

You can view Availability Zones by using the following command.

Linux & macOS

```
aws ssm get-parameters-by-path \
--path /aws/service/global-infrastructure/availability-zones/
```

Windows

```
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.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/global-infrastructure/availability-zones/afsl1-az3",
 "Type": "String",
 "Value": "afsl1-az3",
 "Version": 1,
 "LastModifiedDate": "2020-04-21T09:05:35.375000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
availability-zones/afsl1-az3",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/availability-zones/apne2-az3",
 "Type": "String",
 "Value": "apne2-az3",
 "Version": 1,
 "LastModifiedDate": "2020-04-03T13:19:44.642000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
availability-zones/apne2-az3",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/availability-zones/apsl1-az2",
 "Type": "String",
 "Value": "apsl1-az2",
 "Version": 1,
 "LastModifiedDate": "2020-04-03T13:19:44.642000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
availability-zones/apsl1-az2",
 "DataType": "text"
 }
]
}
```

```
 "Type": "String",
 "Value": "aps1-az2",
 "Version": 1,
 "LastModifiedDate": "2020-04-03T13:13:57.351000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
availability-zones/aps1-az2",
 "dataType": "text"
 }
]
```

### View Availability Zone names only

You can view the names of Availability Zones only by using the following command.

Linux & macOS

```
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/ape1-az1",
 "/aws/service/global-infrastructure/availability-zones/ape1-az2",
 "/aws/service/global-infrastructure/availability-zones/ape1-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",
```

### View names of Availability Zones in a single Region

You can view the names of the Availability Zones in one Region (us-east-2, in this example) using the following command.

Linux & macOS

```
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 Amazon Resource Names (ARNs) of Availability Zones only by using the following command.

Linux & macOS

```
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. This example has been truncated for space.

```
[
 "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/availability-zones/
afs1-az1",
 "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/availability-zones/
afs1-az2",
 "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/availability-zones/
afs1-az3",
 "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/availability-zones/
ape1-az1",
 "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/availability-zones/
ape1-az2",
 "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/availability-zones/
ape1-az3",
 "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/availability-zones/
apnel-az1",
```

### View local zone details

You can view local zones by using the following command.

Linux & macOS

```
aws ssm get-parameters-by-path \
 --path /aws/service/global-infrastructure/local-zones
```

Windows

```
aws ssm get-parameters-by-path ^
```

```
--path /aws/service/global-infrastructure/local-zones
```

The command returns information like the following. This example has been truncated for space.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-bos1-az1",
 "Type": "String",
 "Value": "use1-bos1-az1",
 "Version": 3,
 "LastModifiedDate": "2020-12-15T14:16:17.502000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
local-zones/use1-bos1-az1",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-chi1-az1",
 "Type": "String",
 "Value": "use1-chi1-az1",
 "Version": 1,
 "LastModifiedDate": "2021-09-08T10:05:38.634000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
local-zones/use1-chi1-az1",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-dfw1-az1",
 "Type": "String",
 "Value": "use1-dfw1-az1",
 "Version": 1,
 "LastModifiedDate": "2021-07-07T05:28:16.412000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
local-zones/use1-dfw1-az1",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-mci1-az1",
 "Type": "String",
 "Value": "use1-mci1-az1",
 "Version": 1,
 "LastModifiedDate": "2021-09-08T10:05:37.571000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
local-zones/use1-mci1-az1",
 "DataType": "text"
 }
]
}
```

## View Wavelength Zone details

You can view Wavelength Zones by using the following command.

Linux & macOS

```
aws ssm get-parameters-by-path \
 --path /aws/service/global-infrastructure/wavelength-zones
```

Windows

```
aws ssm get-parameters-by-path ^
```

```
--path /aws/service/global-infrastructure/wavelength-zones
```

The command returns information like the following. This example has been truncated for space.

```
{
 "Parameters": [
 {
 "Name": "/aws/service/global-infrastructure/wavelength-zones/apne1-wl1-nrt-wlz1",
 "Type": "String",
 "Value": "apne1-wl1-nrt-wlz1",
 "Version": 3,
 "LastModifiedDate": "2020-12-15T14:16:04.715000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/wavelength-zones/apne1-wl1-nrt-wlz1",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/wavelength-zones/use1-wl1-atl-wlz1",
 "Type": "String",
 "Value": "use1-wl1-atl-wlz1",
 "Version": 3,
 "LastModifiedDate": "2020-09-22T13:10:11.783000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/wavelength-zones/use1-wl1-atl-wlz1",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/wavelength-zones/use1-wl1-bos-wlz1",
 "Type": "String",
 "Value": "use1-wl1-bos-wlz1",
 "Version": 3,
 "LastModifiedDate": "2020-08-06T07:11:23.225000-07:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/wavelength-zones/use1-wl1-bos-wlz1",
 "DataType": "text"
 }
]
}
```

### 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 & macOS

```
aws ssm get-parameters-by-path \
 --path "/aws/service/global-infrastructure/local-zones/usw2-lax1-az1/"
```

Windows

```
aws ssm get-parameters-by-path ^
 --path "/aws/service/global-infrastructure/local-zones/use1-bos1-az1"
```

The command returns information like the following.

```
{
```

```

"Parameters": [
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/geolocationCountry",
 "Type": "String",
 "Value": "US",
 "Version": 3,
 "LastModifiedDate": "2020-12-15T14:16:17.641000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/local-zones/use1-bos1-az1/geolocationCountry",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/geolocationRegion",
 "Type": "String",
 "Value": "US-MA",
 "Version": 3,
 "LastModifiedDate": "2020-12-15T14:16:17.794000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/local-zones/use1-bos1-az1/geolocationRegion",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/location",
 "Type": "String",
 "Value": "US East (Boston)",
 "Version": 1,
 "LastModifiedDate": "2021-01-11T10:53:24.634000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/local-zones/use1-bos1-az1/location",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/network-border-group",
 "Type": "String",
 "Value": "us-east-1-bos-1",
 "Version": 3,
 "LastModifiedDate": "2020-12-15T14:16:20.641000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/local-zones/use1-bos1-az1/network-border-group",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/parent-availability-zone",
 "Type": "String",
 "Value": "use1-az4",
 "Version": 3,
 "LastModifiedDate": "2020-12-15T14:16:20.834000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/local-zones/use1-bos1-az1/parent-availability-zone",
 "DataType": "text"
 },
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/parent-region",
 "Type": "String",
 "Value": "us-east-1",
 "Version": 3,
 "LastModifiedDate": "2020-12-15T14:16:20.721000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/local-zones/use1-bos1-az1/parent-region",
 "DataType": "text"
 }
]

```

```
 },
 {
 "Name": "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/zone-
group",
 "Type": "String",
 "Value": "us-east-1-bos-1",
 "Version": 3,
 "LastModifiedDate": "2020-12-15T14:16:17.983000-08:00",
 "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
local-zones/use1-bos1-az1/zone-group",
 "DataType": "text"
 }
]
}
```

### View local zone parameter names only

You can view just the names of local zone parameters by using the following command.

Linux & macOS

```
aws ssm get-parameters-by-path \
--path /aws/service/global-infrastructure/local-zones/usw2-lax1-az1 \
--query 'Parameters[].Name | sort(@)'
```

Windows

```
aws ssm get-parameters-by-path ^
--path /aws/service/global-infrastructure/local-zones/use1-bos1-az1 ^
--query "Parameters[].Name | sort(@)"
```

The command returns information like the following.

```
[
 "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/geolocationCountry",
 "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/geolocationRegion",
 "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/location",
 "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/network-border-group",
 "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/parent-availability-
zone",
 "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/parent-region",
 "/aws/service/global-infrastructure/local-zones/use1-bos1-az1/zone-group"
]
```

## Assigning parameter policies

Parameter policies help you manage a growing set of parameters by allowing 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, a capability of AWS Systems Manager. Parameter Store offers the following types of policies: `Expiration`, `ExpirationNotification`, and `NoChangeNotification`.

### Note

To implement password rotation lifecycles, use AWS Secrets Manager. You can rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle using Secrets Manager. For more information, see [What is AWS Secrets Manager?](#) in the *AWS Secrets Manager User Guide*.

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 \(p. 265\)](#).

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.

Policy	Details	Examples
<b>Expiration</b>	<p>This policy deletes the parameter. You can specify a specific date and time by using either the <code>ISO_INSTANT</code> format or the <code>ISO_OFFSET_DATE_TIME</code> format. To change when you want the parameter to be deleted, update the policy. Updating a <i>parameter</i> doesn't 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.</p> <p><b>Note</b>            This example uses the <code>ISO_INSTANT</code> format. You can also specify a date and time by using the <code>ISO_OFFSET_DATE_TIME</code> format. Here is an example:  <code>2019-11-01T22:13:48.87+10:30:00</code>            .</p>	<pre>{   "Type": "Expiration",   "Version": "1.0",   "Attributes": {     "Timestamp":     "2018-12-02T21:34:33.000Z"   } }</pre>
<b>ExpirationNotification</b>	<p>This policy initiates an event in Amazon EventBridge (EventBridge) 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.</p>	<pre>{   "Type": "ExpirationNotification",   "Version": "1.0",   "Attributes": {     "Before": "15",     "Unit": "Days"   } }</pre>
<b>NoChangeNotification</b>	<p>This policy initiates an event in EventBridge if a parameter has <i>not</i> been modified for a specified period of time. This policy type is useful when, for</p>	<pre>{   "Type": "NoChangeNotification",   "Version": "1.0",   "Attributes": {     "After": "20",     "Unit": "Hours"   } }</pre>

Policy	Details	Examples
	<p>example, a password needs to be changed within a period of time.</p> <p>This policy determines when to send a notification by reading the <code>LastModifiedTime</code> attribute of the parameter. If you change or edit a parameter, the system resets the notification time period based on the new value of <code>LastModifiedTime</code>.</p>	<pre style="background-color: #f0f0f0; padding: 10px;">         "Unit": "Days"     } }</pre>

You can assign multiple policies to a parameter. For example, you can assign `Expiration` and `ExpirationNotification` policies so that the system initiates an EventBridge 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 the request syntax for a [PutParameter](#) API request that assigns four policies to a new `SecureString` parameter named `ProdDB3`.

```
{
 "Name": "ProdDB3",
 "Description": "Parameter with policies",
 "Value": "P@ssW*rd21",
 "Type": "SecureString",
 "Overwrite": "True",
 "Policies": [
 {
 "Type": "Expiration",
 "Version": "1.0",
 "Attributes": {
 "Timestamp": "2018-12-02T21:34:33.000Z"
 }
 },
 {
 "Type": "ExpirationNotification",
 "Version": "1.0",
 "Attributes": {
 "Before": "30",
 "Unit": "Days"
 }
 },
 {
 "Type": "ExpirationNotification",
 "Version": "1.0",
 "Attributes": {
 "Before": "15",
 "Unit": "Days"
 }
 },
 {
 "Type": "NoChangeNotification",
 "Version": "1.0",
 "Attributes": {
 "After": "20",
 "Unit": "Days"
 }
 }
]
}
```

```
]
 }
}
```

## 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 Command Line Interface (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 \(p. 279\)](#).

### Topics

- [Add policies to an existing parameter \(console\) \(p. 317\)](#)
- [Add policies to an existing parameter \(AWS CLI\) \(p. 317\)](#)
- [Add policies to an existing parameter \(Tools for Windows PowerShell\) \(p. 319\)](#)

### 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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, 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 Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to add policies to an existing parameter.

Linux & macOS

```
aws ssm put-parameter
--name "parameter-name" \
--value 'parameter-value' \
--type parameter-type \
--overwrite \
--policies "[{policies-enclosed-in-brackets-and-curly-braces}]"
```

Windows

```
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 an EventBridge 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 (313vat3131) for a password and an AWS Key Management Service AWS KMS key. For more information about AWS KMS keys, see [AWS Key Management Service Concepts](#) in the [AWS Key Management Service Developer Guide](#).

Linux & macOS

```
aws ssm put-parameter \
--name "/Finance/Payroll/313vat3131" \
--value "P@sSwW)rd" \
--type "SecureString" \
--overwrite \
--policies "[{\\"Type\\":\\"Expiration\\",\\"Version\\":\\"1.0\\",\\"Attributes\\":{\\"Timestamp\\":\\"2020-05-13T00:00:00.000Z\\"}}, {\\"Type\\":\\"ExpirationNotification\\",\\"Version\\":\\"1.0\\",\\"Attributes\\":{\\"Before\\":\\"5\\",\\"Unit\\":\\"Days\\"}}, {\\"Type\\":\\"NoChangeNotification\\",\\"Version\\":\\"1.0\\",\\"Attributes\\":{\\"After\\":\\"60\\",\\"Unit\\":\\"Days\\\"}}]"
```

Windows

```
aws ssm put-parameter ^
--name "/Finance/Payroll/313vat3131" ^
--value "P@sSwW)rd" ^
--type "SecureString" ^
--overwrite ^
--policies "[{\\"Type\\":\\"Expiration\\",\\"Version\\":\\"1.0\\",\\"Attributes\\":{\\"Timestamp\\":\\"2020-05-13T00:00:00.000Z\\"}}, {\\"Type\\":\\"ExpirationNotification\\",\\"Version\\":\\"1.0\\",\\"Attributes\\":{\\"Before\\":\\"5\\",\\"Unit\\":\\"Days\\"}}, {\\"Type\\":\\"NoChangeNotification\\",\\"Version\\":\\"1.0\\",\\"Attributes\\":{\\"After\\":\\"60\\",\\"Unit\\":\\"Days\\\"}}]"
```

```
\":\"NoChangeNotification\", \"Version\": \"1.0\", \"Attributes\": {\"After\": \"60\",
\\"Unit\": \"Days\"}}]}]
```

- Run the following command to verify the details of the parameter.

Linux & macOS

```
aws ssm describe-parameters \
--parameter-filters "Key=Name,Values=parameter-name"
```

Windows

```
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:

Linux & macOS

```
aws ssm put-parameter \
--name parameter-name \
--type parameter-type \
--value 'parameter-value' \
--policies "[{}]"
```

Windows

```
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, copy and paste the original policies, type the new policy, and then save your changes.

### Add policies to an existing parameter (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

- Open Tools for Windows PowerShell and run the following command to specify your credentials. You must either have administrator permissions in Amazon Elastic Compute Cloud (Amazon EC2), or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM). For more information, see [Systems Manager prerequisites \(p. 13\)](#).

```
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).

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

3. Run the following command to add policies to an existing parameter.

```
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 an EventBridge 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 key.

```
Write-SSMParameter -Name "/Finance/Payroll/3l3vat3131" -Value "P@ssSwW)rd" -Type "SecureString" -Policies "[{"Type": "Expiration", "Version": "1.0", "Attributes": {"Timestamp": "2018-05-13T00:00:00.000Z"}, {"Type": "ExpirationNotification", "Version": "1.0", "Attributes": {"Before": "5", "Unit": "Days"}, {"Type": "NoChangeNotification", "Version": "1.0", "Attributes": {"After": "60", "Unit": "Days"}}]" -Overwrite
```

4. Run the following command to verify the details of the parameter.

```
(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, copy and paste the original policies, type the new policy, and then save your changes.

## Searching for Systems Manager parameters

When you have a lot of parameters in your account, it can be difficult to find information about a single or several 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 AWS Systems Manager console, the AWS Command Line Interface (AWS CLI), the AWS Tools for PowerShell, or the [DescribeParameters API](#) to search for parameters.

### Topics

- [Search for a parameter \(console\) \(p. 321\)](#)
- [Search for a parameter \(AWS CLI\) \(p. 321\)](#)

## Search for a parameter (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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. Select 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're searching by Type, choose from String, StringList, or SecureString.
  - If you're 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.

5. Press **Enter**.

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 Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Replace the sample values in the following commands with values reflecting parameters that have been created in your account.

Linux & macOS

```
aws ssm describe-parameters \
--parameter-filters "Key=Name,Values=MyParameterName"
```

Windows

```
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*".

```
aws ssm describe-parameters --parameter-filters
"Key=Name,Option=Contains,Values=Product"
```

```
aws ssm describe-parameters --parameter-filters "Key=Type,Values=String"
```

```
aws ssm describe-parameters --parameter-filters "Key=Path,Values=/Production/West"
```

```
aws ssm describe-parameters --parameter-filters "Key=Tier,Values=Standard"
```

```
aws ssm describe-parameters --parameter-filters "Key=tag:tag-key,Values=tag-value"
```

```
aws ssm describe-parameters --parameter-filters "Key=KeyId,Values=key-id"
```

**Note**

In the last example, `key-id` represents the ID of an 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 [Create a SecureString parameter \(AWS CLI\) \(p. 287\)](#).

If successful, the command returns output similar to the following.

```
{
 "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",
 "LastModifiedDate": 1572363610.175,
 "LastModifiedUser": "arn:aws:iam::111122223333:user/Mateo.Jackson",
 "Version": 1,
 "Tier": "Standard",
 "Policies": []
 },
 {
 "Name": "/Production/West/HR",
 "Type": "String",
 "LastModifiedDate": 1572363680.503,
 "LastModifiedUser": "arn:aws:iam::111122223333:user/Mateo.Jackson",
 "Version": 1,
 "Tier": "Standard",
 "Policies": []
 }
]
}
```

## Working with parameters using Run Command commands

You can work with parameters in Run Command, a capability of AWS Systems Manager. For more information, see [Sending commands using Systems Manager Run Command \(p. 995\)](#).

## Run a String parameter (console)

The following procedure walks you through the process of running a command that uses a **String** parameter.

### To run a String parameter using Parameter Store

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
  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-RunPowerShellScript** (Windows) or **AWS-RunShellScript** (Linux).
  5. For **Command parameters**, enter **echo {{ssm:parameter-name}}**. For example: **echo {{ssm:/Test/helloWorld}}**.
  6. In the **Targets** section, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Other parameters**:
  - For **Comment**, enter information about this command.
  - For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.
8. For **Rate control**:
  - For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

11. Choose **Run**.
12. In the **Command ID** page, in the **Targets and outputs** area, select the button next to the ID of a node 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**.

## Run a parameter (AWS CLI)

The following example command includes a Systems Manager parameter named **DNS-IP**. The value of this parameter is simply the IP address of a node. This example uses an AWS Command Line Interface (AWS CLI) command to echo the parameter value.

### Linux & macOS

```
aws ssm send-command \
--document-name "AWS-RunShellScript" \
--document-version "1" \
--targets "Key=instanceids,Values=i-02573cafefEXAMPLE" \
--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-02573cafefEXAMPLE" ^
--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 used in the **parameters** field 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.

### Linux

```
aws ssm send-command \
--document-name "AWS-RunShellScript" \
--document-version "1" \
--targets "Key=instanceids,Values=i-02573cafefEXAMPLE" \
--parameters '{"commands":["secure=$(aws ssm get-parameters --names SecurePassword --with-decryption --query Parameters[0].Value --output text --region us-east-2)","echo $secure | passwd myuser --stdin"]}' \
--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-02573cafefEXAMPLE" ^
--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.

```
{
 "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.

```
"runtimeConfig": {
 "aws:runShellScript": {
 "properties": [
 {
 "id": "0.aws:runShellScript",
 "runCommand": "{{ commands }}",
 "workingDirectory": "{{ workingDirectory }}",
 "timeoutSeconds": "{{ executionTimeout }}"
 }
]
 }
}
```

### Note

SSM documents 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 & macOS

```
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

```
aws ssm send-command ^
--name AWS-JoinDomain ^
--parameters password=$value ^
--instance-id instance-id
```

### Powershell

```
$secure = (Get-SSMParameterValue -Names parameter-name -WithDecryption $True).Parameters[0].Value | ConvertTo-SecureString -AsPlainText -Force
```

```
$cred = New-Object System.Management.Automation.PSCredential -argumentlist user-name,$secure
```

## Working with parameter 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 \(p. 326\)](#)
- [Querying parameters in a hierarchy \(p. 327\)](#)
- [Restricting access to Parameter Store API operations \(p. 327\)](#)
- [Manage parameters using hierarchies \(AWS CLI\) \(p. 328\)](#)

### 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

```
/MyApp/.NET/Libraries/my-password
```

- Your business organization

```
/Finance/Accountants/UserList
```

```
/Finance/Analysts/UserList
```

```
/HR/Employees/EU/UserList
```

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.

```
/Dev/Test/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 aren't 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 system treats both of the following parameters as root parameters.

```
/parameter-name
```

```
parameter-name
```

## 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 operation. For example, if you run the following command from the AWS Command Line Interface (AWS CLI), the system returns all parameters in the IIS level.

```
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.

```
aws ssm get-parameters-by-path --path /Prod/ERP/SAP --with-decryption
```

## Restricting access to Parameter Store API operations

Using AWS Identity and Access Management (IAM) policies, you can provide or restrict user access to Parameter Store API operations and content.

In the following sample policy, users are first granted access to run the `PutParameter` API operation 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` operation. 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"
],
 "Resource": "arn:aws:ssm:us-east-2:123456789012:parameter/*"
 },
 {
 "Effect": "Deny",
 "Action": [
 "ssm:PutParameter"
],
 "Condition": {
 "StringEquals": {
 "ssm:Overwrite": [
 "true"
]
 }
 },
 "Resource": "arn:aws:ssm:us-east-2:123456789012:parameter/*"
 }
]
}
```

## Manage parameters using hierarchies (AWS CLI)

This procedure shows how to work with parameters and parameter hierarchies by using the AWS CLI.

### To manage parameters using hierarchies

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

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 & macOS

```
aws ssm put-parameter \
 --name "/MyService/Test/MaxConnections" \
 --value 100 --allowed-pattern "\d{1,4}" \
 --type String
```

Windows

```
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 & macOS

```
aws ssm put-parameter \
--name "/MyService/Test/MaxConnections" \
--value 10,000 \
--type String \
--overwrite
```

Windows

```
aws ssm put-parameter ^
--name "/MyService/Test/MaxConnections" ^
--value 10,000 ^
--type String ^
--overwrite
```

The system returns the following error because the new value doesn't 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 key. The allowed pattern in this example means the user can specify any character, and the value must be between 8 and 20 characters.

Linux & macOS

```
aws ssm put-parameter \
--name "/MyService/Test/my-password" \
--value "p#sW*rd33" \
--allowed-pattern ".{8,20}" \
--type SecureString
```

Windows

```
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 & macOS

```
aws ssm put-parameter \
--name "/MyService/Test/DBname" \
--value "SQLDevDb" \
--type String
```

```
aws ssm put-parameter \
--name "/MyService/Test/user" \
--value "user1" \
--type String
```

```
--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 & macOS

```
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 & macOS

```
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 & macOS

```
aws ssm delete-parameters \
--names "/IADRegion/Dev/user" "/IADRegion/Dev/userType"
```

## Windows

```
aws ssm delete-parameters ^
--names "/IADRegion/Dev/user" "/IADRegion/Dev/userType"
```

## Working with parameter labels

A parameter label is a user-defined alias to help you manage different versions of a parameter. When you modify a parameter, AWS 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 3 of the `/MyApp/DB/ConnectionString` parameter with a connection string for a new production database, then you can move the *Production* label from version 2 of the parameter to version 3 of the parameter.

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 operation, 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 of the parameter 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 parameter. You must attach a label to a specific version of a parameter.
- If you no longer want to use a parameter label, then you can move it to a different version of a parameter or delete it.
- 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 doesn't meet these requirements, then the label isn't attached to the parameter version and the system displays it in the list of `InvalidLabels`.

### Topics

- [Working with parameter labels \(console\) \(p. 332\)](#)

- [Working with parameter labels \(AWS CLI\) \(p. 333\)](#)

## Working with parameter labels (console)

This section describes how to perform the following tasks by using the Systems Manager console.

- [Create a parameter label \(console\) \(p. 332\)](#)
- [View labels attached to a parameter \(console\) \(p. 332\)](#)
- [Move a parameter label \(console\) \(p. 333\)](#)
- [Delete parameter labels \(console\) \(p. 333\)](#)

### Create a parameter label (console)

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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Manage labels**.
7. Choose **Add new label**.
8. In the text box, enter the label name. To add more labels, choose **Add new label**. You can attach a maximum of ten labels.
9. When you're finished, choose **Save changes**.

### View labels attached to a parameter (console)

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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 (console)

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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Manage labels**.
7. Choose **Add new label**.
8. In the text box, enter the label name.
9. When you're finished, choose **Save changes**.

## Delete parameter labels (console)

The following procedure describes how to delete one or multiple parameter labels using the Systems Manager console.

### To delete labels from a parameter

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 delete labels.
6. Choose **Manage labels**.
7. Choose **Remove** next to each label you want to delete.
8. When you're finished, choose **Save changes**.
9. Confirm that your changes are correct, enter **Confirm** in the text box, and choose **Confirm**.

## Working with parameter labels (AWS CLI)

This section describes how to perform the following tasks by using the AWS Command Line Interface (AWS CLI).

- [Create a new parameter label \(AWS CLI\) \(p. 334\)](#)
- [View labels for a parameter \(AWS CLI\) \(p. 335\)](#)

- View a list of parameters that are assigned a label (AWS CLI) (p. 336)
- Move a parameter label (AWS CLI) (p. 337)
- Delete parameter labels (AWS CLI) (p. 337)

### Create a new parameter label (AWS CLI)

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 parameter label

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to view a list of parameters for which you have permission to attach a label.

##### Note

Parameters are only available in the AWS Region where they were created. If you don't see a parameter for which you want to attach a label, then verify your Region.

```
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.

```
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.

```
aws ssm get-parameters --names "parameter-name:version-number"
```

Here is an example.

```
aws ssm get-parameters --names "/Production/SQLConnectionString:3"
```

5. Run one of the following commands to attach a label to a version of a parameter. If you attach multiple labels, 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 doesn't 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 (AWS CLI)

You can use the [GetParameterHistory](#) API operation to view the full history and all labels attached to a specified parameter. Or, you can use the [GetParametersByPath](#) API operation to view a list of all parameters that are assigned a specific label.

### To view labels for a parameter by using the GetParameterHistory API operation

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 the parameter you want to view the labels of.

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 (AWS CLI)

You can use the [GetParametersByPath](#) API operation 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.

```
{
 "Parameters": [
 {
 "Version": 3,
 "Type": "SecureString",
 "Name": "/Config/DBpwd",
 "Value": "MyS@perGr&pass33"
 },
 {
 "Version": 2,
 "Type": "String",
 "Name": "/Config/DBusername",
 "Value": "TestUserDB"
 },
 {
 "Version": 2,
 "Type": "String",
 "Name": "/Config/endpoint",
 "Value": "MyTestService-July-Release.example.com"
 }
]
}
```

## Move a parameter label (AWS CLI)

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 versions you want to move the label to and from.

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.

## Delete parameter labels (AWS CLI)

The following procedure describes how to delete parameter labels by using the AWS CLI.

### To delete a parameter label

1. Run the following command to view all versions of the parameter.

```
aws ssm get-parameter-history --name "parameter-name"
```

The system returns information like the following.

```
{
 "Parameters": [
 {
 "Name": "foo",
 "DataType": "text",
 "LastModifiedDate": 1607380761.11,
 "Labels": [
 "l3",
 "l2"
],
 "Value": "test",
 "Version": 1,
 "LastModifiedUser": "arn:aws:iam::123456789012:user/test",
 "Policies": [],
 "Tier": "Standard",
 "Type": "String"
 },
 {
 "Name": "foo",
 "DataType": "text",
 "LastModifiedDate": 1607380763.11,
 "Labels": [
 "l1"
],
 "Value": "test",
 "Version": 2,
 }
]
}
```

```
 "LastModifiedUser": "arn:aws:iam::123456789012:user/test",
 "Policies": [],
 "Tier": "Standard",
 "Type": "String"
 }
]
```

Note the parameter version for which you want to delete a label or labels.

- Run the following command to delete the labels you choose from that parameter.

```
aws ssm unlabel-parameter-version --name parameterName --parameter-version version --
labels label1, label2, label3
```

The system returns information like the following.

```
{
 "InvalidLabels": ["invalid"],
 "DeletedLabels" : ["Prod"]
}
```

## Working with parameter versions

Each time you edit the value of a parameter, Parameter Store, a capability of AWS Systems Manager 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 the number of 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 and maintain up to 100 versions of a parameter. After you have created 100 versions of a parameter, each time you create a new version, the oldest version of the parameter is removed from history to make room for the new version.

An exception to this is when there are already 100 parameter versions in history, and a parameter label is assigned to the oldest version of a parameter. In this case, that version isn't removed from history, and the request to create a new parameter version fails. This safeguard is to prevent parameter versions with mission critical labels assigned to them from being deleted. To continue creating new parameters, first move the label from the oldest version of the parameter to a newer one for use in your operations. For information about moving parameter labels, see [Move a parameter label \(console\) \(p. 333\)](#) and [Move a parameter label \(AWS CLI\) \(p. 337\)](#).

The following procedures show you how to edit a parameter and then verify that you created a new version. You can use the `get-parameter` and `get-parameters` commands to view parameter versions. For examples on using these commands, see [GetParameter](#) and [GetParameters](#) in the *AWS Systems Manager API Reference*

### Topics

- [Create a new version of a parameter \(console\) \(p. 339\)](#)
- [Reference a parameter version \(p. 339\)](#)

## Create a new version of a parameter (console)

You can use the 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 that you created earlier. For information about creating a new parameter, see [Creating Systems Manager parameters \(p. 279\)](#).
4. Choose **Edit**.
5. In the **Value** box, enter 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, the Amazon Elastic Compute Cloud (Amazon EC2) `run-instances` command uses version 3 of the parameter `golden-ami`.

### Linux & macOS

```
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
```

### Windows

```
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

Using `resolve` and a parameter value only works with the `--image-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 \(p. 340\)](#).

Here is an example for specifying version 2 of a parameter named `MyRunCommandParameter` in an SSM document.

### YAML

```

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}}"
```

### 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}}"
]
 }
 }
]
}
```

## 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 allows you to 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 you want to create new instances from. 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 Elastic Compute Cloud (Amazon EC2) `run-instances` command.

#### Note

The user who runs this command must have AWS Identity and Access Management (IAM) permissions that include the `ssm:GetParameters` API operation in order for the parameter value to be validated. Otherwise, the parameter creation process fails.

## Linux & macOS

```
aws ec2 run-instances \
--image-id resolve:ssm:/golden-ami \
--count 1 \
--instance-type t2.micro \
--key-name my-key-pair \
--security-groups my-security-group
```

## Windows

```
aws ec2 run-instances ^
--image-id resolve:ssm:/golden-ami ^
--count 1 ^
--instance-type t2.micro ^
--key-name my-key-pair ^
--security-groups my-security-group
```

You can also choose your preferred AMI when you create an instance using the Amazon EC2 console. For more information, see [Using a Systems Manager parameter to find an AMI](#) in the *Amazon EC2 User Guide for Windows 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.

## Grant permissions to create a parameter of `aws:ec2:image` data type

Using AWS Identity and Access Management (IAM) policies, you can provide or restrict user access to Parameter Store API operations and content.

The following example policy grants users permission to call the `PutParameter` API operation for `aws:ec2:image`. This means that the user can add a parameter of data type `aws:ec2:image` to the system.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "ssm:PutParameter",
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": "ec2:DescribeImages",
 "Resource": "*"
 }
]
}
```

## How AMI format validation works

When you specify `aws:ec2:image` as the data type for a parameter, Systems Manager doesn't 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.

A parameter version number might be generated before the validation operation is complete. The operation might not be complete even if a parameter version number is generated.

To monitor whether your parameters are created successfully, we recommend using Amazon EventBridge (EventBridge) 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.

```
{
 "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": [
 "arn:aws:ssm:us-east-2:111122223333:parameter/golden-ami"
],
 "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 EventBridge, see [Setting up notifications or trigger actions based on Parameter Store events \(p. 275\)](#).

## Parameter Store walkthroughs

The walkthrough in this section shows you how to create, store, and run parameters with Parameter Store, a capability of AWS Systems Manager, in a test environment. This walkthrough shows 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 [What is a parameter? \(p. 258\)](#).

### Contents

- [Create a SecureString parameter and join a node to a Domain \(PowerShell\) \(p. 342\)](#)
- [Use Parameter Store parameters in Amazon Elastic Kubernetes Service \(p. 345\)](#)

## Create a SecureString parameter and join a node to a Domain (PowerShell)

This walkthrough shows how to join a Windows Server node to a domain using AWS 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 key 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](#) in the *Amazon VPC User Guide*.

#### To create a SecureString parameter and join a node to a domain

1. Enter parameters into the system using AWS Tools for Windows PowerShell (Tools for Windows PowerShell).

```
Write-SMParameter -Name "domainName" -Value "$DOMAIN-NAME" -Type String
Write-SMParameter -Name "domainJoinUserName" -Value "$DOMAIN\USERNAME" -Type String
Write-SMParameter -Name "domainJoinPassword" -Value "$PASSWORD" -Type SecureString
```

### Important

Only the *value* of a `SecureString` parameter is encrypted. Parameter names, descriptions, and other properties aren't encrypted.

2. Attach the following AWS Identity and Access Management (IAM) policies to the IAM role permissions for your node:
  - **AmazonSSMManagedInstanceCore** – Required. This AWS managed policy allows a managed node 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 node.
  - **A custom policy for S3 bucket access** – Required. SSM Agent, which is on your node and performs Systems Manager tasks, requires access to specific Amazon-owned Amazon Simple Storage Service (Amazon 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 \(p. 24\)](#)

### 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 User Guide*.

- **CloudWatchAgentServerPolicy** – Optional. This AWS managed policy allows you to run the CloudWatch agent on managed nodes. This policy makes it possible to read information on a node and write it to Amazon CloudWatch. Your instance profile needs this policy only if you use services such as Amazon EventBridge or CloudWatch Logs.

### Note

Using CloudWatch and EventBridge 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 EventBridge User Guide](#) and the [Amazon CloudWatch Logs User Guide](#).

3. Edit the IAM role attached to the node and add the following policy. This policy gives the node permissions to call the `kms:Decrypt` and the `ssm:CreateDocument` API.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "kms:Decrypt",
 "ssm:CreateDocument"
],
 "Resource": [
 "arn:aws:kms:$region:$account-id:key/$kms-key-id"
]
 }
]
}
```

```

]
 }
}
```

4. Copy and paste the following json text into a 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 join a Windows Server instance to a domain",
 "mainSteps": [
 {
 "action": "aws:runPowerShellScript",
 "name": "runPowerShellWithSecureString",
 "precondition": {
 "StringEquals": [
 "platformType",
 "Windows"
]
 },
 "inputs": {
 "runCommand": [
 "$domain = (Get-SSMParameterValue -Name $domainName).Parameters[0].Value",
 "if ((gwmi Win32_ComputerSystem).domain -eq $domain){write-host \\\"Computer is part of $domain, exiting\\\"; exit 0}",
 "$username = (Get-SSMParameterValue -Name $domainJoinUserName).Parameters[0].Value",
 "$password = (Get-SSMParameterValue -Name $domainJoinPassword -WithDecryption $True).Parameters[0].Value | ConvertTo-SecureString -asPlainText -Force",
 "$credential = New-Object System.Management.Automation.PSCredential($username,$password)",
 "Add-Computer -DomainName $domain -Credential $credential -ErrorAction SilentlyContinue -ErrorVariable domainjoinerror",
 "if($?){Write-Host \\\"Instance joined to domain successfully.\\\"; exit 3010}else{Write-Host \\\"Instance failed to join domain with error:\\$domainjoinerror; exit 1 }"
]
 }
 }
]
}
```

5. Run the following command in Tools for Windows PowerShell to create a new SSM document.

```
$json = Get-Content C:\temp\JoinInstanceToDomain | Out-String
New-SSMDocument -Name JoinInstanceToDomain -Content $json -DocumentType Command
```

6. Run the following command in Tools for Windows PowerShell to join the node to the domain.

```
Send-SSMCommand -InstanceId instance-id -DocumentName JoinInstanceToDomain
```

If the command succeeds, the system returns information similar to the following.

```
WARNING: The changes will take effect after you restart the computer EC2ABCD-EXAMPLE.
Domain join succeeded, restarting
Computer is part of example.local, exiting
```

If the command fails, the system returns information similar to the following.

```
Failed to join domain with error:
Computer 'EC2ABCD-EXAMPLE' failed to join domain 'example.local'
from its current workgroup 'WORKGROUP' with following error message:
The specified domain either does not exist or could not be contacted.
```

## Use Parameter Store parameters in Amazon Elastic Kubernetes Service

To show secrets from Secrets Manager and parameters from Parameter Store as files mounted in Amazon EKS pods, you can use the AWS Secrets and Configuration Provider (ASCP) for the [Kubernetes Secrets Store CSI Driver](#). The ASCP works with Amazon Elastic Kubernetes Service (Amazon EKS) 1.17+. Parameter Store is a capability of AWS Systems Manager.

With the ASCP, you can retrieve parameters that are stored and managed in Parameter Store. Then you can use the parameters in your workloads running on Amazon EKS. If your parameter contains multiple key value pairs in JSON format, you can optionally choose to mount them in Amazon EKS. The ASCP can use JMESPath syntax to query the key value pairs in your parameter.

You can use AWS Identity and Access Management (IAM) roles and policies to limit access to your parameters to specific Amazon EKS pods in a cluster. The ASCP retrieves the pod identity and exchanges the identity for an IAM role. ASCP assumes the IAM role of the pod. Then it can retrieve parameters from Parameter Store that are authorized for that role.

To learn how to integrate Secrets Manager with Amazon EKS, see [Using Secrets Manager secrets in Amazon Elastic Kubernetes Service](#).

### Installing the ASCP

The ASCP is available on GitHub in the [secrets-store-csi-driver-provider-aws](#) repository. The repository also contains example YAML files for creating and mounting a secret. You first install the Kubernetes Secrets Store CSI Driver, and then you install the ASCP.

#### To install the Kubernetes Secrets Store CSI Driver and the ASCP

1. To install the Kubernetes Secrets Store CSI Driver, run the following commands. For full installation instructions, see [Installation](#) in the Kubernetes Secrets Store CSI Driver Book. For information about installing Helm, see [Using Helm with Amazon EKS](#).

```
helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
helm install -n kube-system csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver
```

2. To install the ASCP, use the YAML file in the GitHub repository deployment directory. For information about installing kubectl, see [Installing kubectl](#).

```
kubectl apply -f https://raw.githubusercontent.com/aws/secrets-store-csi-driver-provider-aws/main/deployment/aws-provider-installer.yaml
```

### Step 1: Set up access control

To grant your Amazon EKS pod access to parameters in Parameter Store, you first create a policy that limits access to the parameters that the pod needs to access. Then you create an [IAM role for](#)

service account and attach the policy to it. For more information about restricting access to Systems Manager parameters using IAM policies, see [Restricting access to Systems Manager parameters using IAM policies \(p. 260\)](#).

**Note**

When using Parameter Store parameters, the permission `ssm:GetParameters` is needed in the policy.

The ASCP retrieves the pod identity and exchanges it for the IAM role. ASCP assumes the IAM role of the pod, which gives it access to the parameters you authorized. Other containers can't access the parameters unless you also associate them with the IAM role.

## Step 2: Mount parameters in Amazon EKS

To show parameters in Amazon EKS as though they are files on the filesystem, you create a `SecretProviderClass` YAML file that contains information about your parameters and how to mount them in the Amazon EKS pod.

The `SecretProviderClass` must be in the same namespace as the Amazon EKS pod it references.

### `SecretProviderClass`

The `SecretProviderClass` YAML has the following format.

```
apiVersion: secrets-store.csi.x-k8s.io/v1alpha1
kind: SecretProviderClass
metadata:
 name: <NAME>
spec:
 provider: aws
 parameters:
```

#### **parameters**

Contains the details of the mount request.

#### **objects**

A string containing a YAML declaration of the parameters to be mounted. We recommend using a YAML multi-line string or pipe (|) character.

#### **objectName**

The friendly name of the parameter. This becomes the file name of the parameter in the Amazon EKS pod unless you specify `objectAlias`. Use the Name of the parameter, not a full Amazon Resource Name (ARN).

#### **jmesPath**

(Optional) A map of the keys in the JSON encoded parameter to the files to be mounted in Amazon EKS. The following example shows what a JSON encoded parameter looks like.

```
{
 "username" : "myusername",
 "password" : "mypassword"
}
```

The keys are `username` and `password`. The value associated with `username` is `myusername`, and the value associated with `password` is `mypassword`.

**path**

The key in the parameter.

**objectAlias**

The file name to be mounted in the Amazon EKS pod.

**objectType**

Required if you don't use a Secrets Manager ARN for `objectName`. Can be either `secretsmanager` or `ssmparameter`.

**objectAlias**

(Optional) The file name of the parameter in the Amazon EKS pod. If you don't specify this field, the `objectName` appears as the file name.

**objectVersion**

(Optional) The version number of the parameter. We recommend that you don't use this field because you must update it every time you update the parameter. By default, the most recent version is used. For Parameter Store parameters, you can use `objectVersion` or `objectVersionLabel` but not both.

**objectVersionLabel**

(Optional) The parameter label for the version. The default is the most recent version. For Parameter Store parameters, you can use `objectVersion` or `objectVersionLabel` but not both.

**region**

(Optional) The AWS Region of the parameter. If you don't use this field, the ASCP looks up the Region from the annotation on the node. This lookup adds overhead to mount requests, so we recommend that you provide the Region for clusters that use large numbers of pods.

**pathTranslation**

(Optional) A single substitution character to use if the file name (either `objectName` or `objectAlias`) contains the path separator character, such as slash (/) on Linux. If a parameter name contains the path separator, the ASCP can't create a mounted file with that name. Instead, you can replace the path separator character with a different character by entering it in this field. If you don't use this field, the default is underscore (\_), so for example, `My/Path/Parameter` mounts as `My_Path_Parameter`.

To prevent character substitution, enter the string `False`.

**Example**

The following example configuration shows a `SecretProviderClass` with a Parameter Store parameter resource.

```
apiVersion: secrets-store.csi.x-k8s.io/v1alpha1
kind: SecretProviderClass
metadata:
 name: aws-secrets
spec:
 provider: aws
 parameters:
 objects: |
 - objectName: "MyParameter"
 objectType: "ssmparameter"
```

## Step 3: Update the deployment YAML

Update your deployment YAML to use the `secrets-store.csi.k8s.io` driver and reference the `SecretProviderClass` resource created in the previous step. This ensures your cluster is using the Secrets Store CSI driver.

Below is a sample deployment YAML using a `SecretProviderClass` named `aws-secrets`.

```
volumes:
- name: secrets-store-inline
 csi:
 driver: secrets-store.csi.k8s.io
 readOnly: true
 volumeAttributes:
 secretProviderClass: "my-secret-provider-class"
```

## Tutorial: Create and mount a parameter in an Amazon EKS pod

In this tutorial, you create an example parameter in Parameter Store, and then you mount the parameter in an Amazon EKS pod and deploy it.

Before you begin, install the ASCP. For more information, see [the section called “Installing the ASCP” \(p. 345\)](#).

### To create and mount a secret

1. Set the AWS Region and the name of your cluster as shell variables so you can use them in bash commands. For `region`, enter the AWS Region where your Amazon EKS cluster runs. For `clusternamespace`, enter the name of your cluster.

```
REGION=region
CLUSTERNAME=clusternamespace
```

2. Create a test parameter.

```
aws ssm put-parameter --name "MyParameter" --value "EKS parameter" --type String --region "$Region"
```

3. Create a resource policy for the pod that limits its access to the parameter you created in the previous step. For `<PARAMETERARN>`, use the ARN of the parameter. Save the policy ARN in a shell variable. To retrieve the parameter ARN, use `get-parameter`.

```
POLICY_ARN=$(aws --region "$REGION" --query Policy.Arn --output text iam create-policy --policy-name nginx-parameter-deployment-policy --policy-document '{
 "Version": "2012-10-17",
 "Statement": [{
 "Effect": "Allow",
 "Action": ["ssm:GetParameter", "ssm:GetParameters"],
 "Resource": ["parameter-arn"]
 }]
}')
```

4. Create an IAM OpenID Connect (OIDC) provider for the cluster if you don't already have one. For more information, see [Create an IAM OIDC provider for your cluster](#).

```
eksctl utils associate-iam-oidc-provider --region="$REGION" --cluster="$CLUSTERNAME" --approve # Only run this once
```

5. Create the service account the pod uses, and associate the resource policy you created in step 3 with that service account. For this tutorial, for the service account name, you use *nginx-deployment-sa*. For more information, see [Create an IAM role for a service account](#).

```
eksctl create iamserviceaccount --name nginx-deployment-sa --region="$REGION" --cluster "$CLUSTERNAME" --attach-policy-arn "$POLICY_ARN" --approve --override-existing-serviceaccounts
```

6. Create the `SecretProviderClass` to specify which parameter to mount in the pod. The following command uses the file location of a `SecretProviderClass` file named `ExampleSecretProviderClass.yaml`. For information about creating your own `SecretProviderClass`, see [the section called "SecretProviderClass" \(p. 346\)](#).

```
kubectl apply -f ./ExampleSecretProviderClass.yaml
```

7. Deploy your pod. The following command uses a deployment file named `ExampleDeployment.yaml`. For information about creating your own `SecretProviderClass`, see [the section called "Step 3: Update the deployment YAML" \(p. 348\)](#).

```
kubectl apply -f ./ExampleDeployment.yaml
```

8. To verify the parameter has been mounted properly, use the following command and confirm that your parameter value appears.

```
kubectl exec -it $(kubectl get pods | awk '/nginx-deployment/{print $1}') | head -1
cat /mnt/secrets-store/MyParameter; echo
```

The parameter value appears.

```
"EKS parameter"
```

## Troubleshooting

You can view most errors by describing the pod deployment.

### To see error messages for your container

1. Get a list of pod names with the following command. If you aren't using the default namespace, use `-n <NAMESPACE>`.

```
kubectl get pods
```

2. To describe the pod, in the following command, for `pod-id` use the pod ID from the pods you found in the previous step. If you aren't using the default namespace, use `-n <NAMESPACE>`.

```
kubectl describe pod/pod-id
```

### To see errors for the ASCP

- To find more information in the provider logs, in the following command, for `<PODID>` use the ID of the `csi-secrets-store-provider-aws` pod.

```
kubectl -n kube-system get pods
kubectl -n kube-system logs pod/pod-id
```

## Auditing and logging Parameter Store activity

AWS CloudTrail captures API calls made in the AWS Systems Manager console, the AWS Command Line Interface (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. All CloudTrail logs for your account use one bucket. For more information about viewing and using CloudTrail logs of Systems Manager activity, see [Logging AWS Systems Manager API calls with AWS CloudTrail \(p. 1442\)](#). For more information about auditing and logging options for Systems Manager, see [Monitoring AWS Systems Manager \(p. 1428\)](#).

## Troubleshooting Parameter Store

Use the following information to help you troubleshoot problems with Parameter Store, a capability of AWS Systems Manager.

### Troubleshooting aws:ec2:image parameter creation

Use the following information to help troubleshoot problems with creating aws:ec2:image data type parameters.

#### EventBridge 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 Amazon EventBridge that reports the exception "Unable to Describe Resource".

**Solution:** This message can indicate the following:

- You don't have the AWS Identity and Access Management (IAM) permission for the ec2:DescribeImages API operation, 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 Amazon Machine Image (AMI) ID you entered as a parameter value isn't valid. Make sure you're entering the ID of an AMI that is available in the current AWS Region and account you're 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 EventBridge 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 EventBridge, see [Setting up notifications or trigger actions based on Parameter Store events \(p. 275\)](#).
  - In the Parameter Store section of the Systems Manager console, refresh the list of parameters periodically to search for the new or updated parameter details.
  - Use the **GetParameter** command to check for the new or updated parameter. For example, using the AWS Command Line Interface (AWS CLI):

```
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're 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 is complete, the system reports that validation is still in process. If the parameter isn't created or updated, you can try again after 5 minutes have passed from the original attempt.

# AWS Systems Manager Change Management

AWS Systems Manager provides the following capabilities for making changes to your AWS resources.

## Topics

- [AWS Systems Manager Change Manager \(p. 352\)](#)
- [AWS Systems Manager Automation \(p. 397\)](#)
- [AWS Systems Manager Change Calendar \(p. 695\)](#)
- [AWS Systems Manager Maintenance Windows \(p. 706\)](#)

## AWS Systems Manager Change Manager

Change Manager, a capability of AWS Systems Manager, is an enterprise change management framework for requesting, approving, implementing, and reporting on operational changes to your application configuration and infrastructure. From a single *delegated administrator account*, if you use AWS Organizations, you can manage changes across multiple AWS accounts and across AWS Regions. Alternatively, using a *local account*, you can manage changes for a single AWS account. Use Change Manager for managing changes to both AWS resources and on-premises resources. To get started with Change Manager, open the [Systems Manager console](#). In the navigation pane, choose **Change Manager**.

With Change Manager, you can use pre-approved *change templates* to help automate change processes for your resources and help avoid unintentional results when making operational changes. Each change template specifies the following:

- One or more Automation runbooks for a user to choose from when creating a change request. The changes that are made to your resources are defined in Automation runbooks. You can include custom runbooks or [AWS managed runbooks \(p. 604\)](#) in the change templates you create. When a user creates a change request, they can choose which one of the available runbooks to include in the request. Additionally, you can create change templates that let the user making the request specify any runbook in the change request.
- The users in the account who must review change requests that were made using that change template.
- The Amazon Simple Notification Service (Amazon SNS) topic that is used to notify assigned approvers that a change request is ready for review.
- The Amazon CloudWatch alarm that is used to monitor the runbook workflow.
- The Amazon SNS topic that is used to send notifications about status changes for change requests that are created using the change template.
- The tags to apply to the change template for use in categorizing and filtering your change templates.
- Whether change requests created from the change template can be run without an approval step (auto-approved requests).

Through its integration with Change Calendar, which is another capability of Systems Manager, Change Manager also helps you safely implement changes while avoiding schedule conflicts with important business events. Change Manager integration with AWS Organizations and AWS Single Sign-On helps you manage changes across your organization from a single account using your existing identity

management system. You can monitor change progress from Change Manager and audit operational changes across your organization, providing improved visibility and accountability.

Change Manager complements the safety controls of your [continuous integration](#) (CI) practices and [continuous delivery](#) (CD) methodology. Change Manager isn't intended for changes made as part of an automated release process, such as a CI/CD pipeline, unless there is an exception or approval required.

## How Change Manager works

When the need for a standard or emergency operational change is identified, someone in the organization creates a change request that is based on one of the change templates created for use in your organization or account.

If the requested change requires manual approvals, Change Manager notifies the designated approvers through an Amazon SNS notification that a change request is ready for their review. You can designate approvers for change requests in the change template, or let users designate approvers the change request itself. You can assign different reviewers to different templates. For example, assign one user, user group, or AWS Identity and Access Management (IAM) role who must approve requests for changes to managed nodes, and another user, group, or IAM role for database changes. If the change template allows auto-approvals, and a requester's IAM user policy doesn't prohibit it, the user can also choose to run the Automation runbook for their request without a review step (with the exception of change freeze events).

For each change template, you can add up to five levels of approvers. For example, you might require technical reviewers to approve a change request created from a change template first, and then require a second level of approvals from one or more managers.

Change Manager is integrated with [AWS Systems Manager Change Calendar \(p. 695\)](#). When a requested change is approved, the system first determines whether the request conflicts with other scheduled business activities. If a conflict is detected, Change Manager can block the change or require additional approvals before starting the runbook workflow. For example, you might allow changes only during business hours to ensure that teams are available to manage any unexpected problems. For any changes requested to run outside those hours, you can require higher-level management approval in the form of *change freeze approvers*. For emergency changes, Change Manager can skip the step of checking Change Calendar for conflicts or blocking events after a change request is approved.

When it's time to implement an approved change, Change Manager runs the Automation runbook that is specified in the associated change request. Only the operations defined in approved change requests are permitted when runbook workflows run. This approach helps to avoid unintentional results while changes are being implemented.

In addition to restricting the changes that can be made when a runbook workflow runs, Change Manager also helps you control concurrency and error thresholds. You choose how many resources a runbook workflow can run on at once, how many accounts the change can run in at once, and how many failures to allow before the process is stopped and (if the runbook includes a rollback script) rolled back. You can also monitor the progress of changes being made by using CloudWatch alarms.

After a runbook workflow has completed, you can review details about the changes made. These details include the reason for a change request, which change template was used, who requested and approved the changes, and how the changes were implemented.

### Related content

[Introducing AWS Systems Manager Change Manager](#) on the [AWS News Blog](#)

## How can Change Manager benefit my operations?

Benefits of Change Manager include the following:

- **Reduce risk of service disruption and downtime**

Change Manager can make operational changes safer by ensuring that only approved changes are implemented when a runbook workflow runs. You can block unplanned and unreviewed changes. Change Manager helps you avoid the types of unintentional results caused by human error that require costly hours of research and backtracking.

- **Get detailed auditing and reporting on change histories**

Change Manager provides accountability with a consistent way to report and audit changes made across your organization, the intent of the changes, and details about who approved and implemented them.

- **Avoid schedule conflicts or violations**

Change Manager can detect schedule conflicts such as holiday events or new product launches, based on the active change calendar for your organization. You can allow runbook workflows to run only during business hours, or allow them only with additional approvals.

- **Adapt change requirements to your changing business**

During different business periods, you can implement different change management requirements. For example, during end-of-month reporting, tax season, or other critical business periods, you can block changes or require director-level approval for changes that could introduce unnecessary operational risks.

- **Centrally manage changes across accounts**

Through its integration with Organizations, Change Manager makes it possible for you to manage changes throughout all of your organizational units (OUs) from a single delegated administrator account. You can turn on Change Manager for use with your entire organization or with only some of your OUs.

## Who should use Change Manager?

Change Manager is appropriate for the following AWS customers and organizations:

- Any AWS customer who wants to improve the safety and governance of operational changes made to their cloud or on-premises environments.
- Organizations that want to increase collaboration and visibility across teams, improve application availability by avoiding downtime, and reduce the risk associated with manual and repetitive tasks.
- Organizations that must comply with best practices for change management.
- Customers who need a fully auditable history of changes made to their application configuration and infrastructure.

## What are the main features of Change Manager?

Primary features of Change Manager include the following:

- **Integrated support for change management best practices**

With Change Manager, you can apply select change management best practices to your operations. You can choose to turn on the following options:

- Check Change Calendar to see if events are currently restricted so changes are made only during open calendar periods.
- Allow changes during restricted events with extra approvals from change freeze approvers.
- Require CloudWatch alarms to be specified for all change templates.

- Require all change templates created in your account to be reviewed and approved before they can be used to create change requests.
- **Different approval paths for closed calendar periods and emergency change requests**

You can allow an option to check Change Calendar for restricted events and block approved change requests until the event is complete. However, you can also designate a second group of approvers, change freeze approvers, who can permit the change to be made even if the calendar is closed. You can also create emergency change templates. Change requests created from an emergency change template still require regular approvals but aren't subject to calendar restrictions and don't require change freeze approvals.

- **Control how and when runbook workflows are started**

Runbook workflows can be started according to a schedule, or as soon as approvals are complete (subject to calendar restriction rules).

- **Built-in notification support**

Specify who in your organization should review and approve change templates and change requests. Assign an Amazon SNS topic to a change template to send notifications to the topic's subscribers about status changes for change requests created with that change template.

- **Integration with AWS Systems Manager Change Calendar**

Change Manager allows administrators to restrict scheduling changes during specified time periods. For instance, you can create a policy that allows changes only during business hours to ensure that the team is available to handle any issues. You can also restrict changes during important business events. For example, retail businesses might restrict changes during large sales events. You can also require additional approvals during restricted periods.

- **Integration with AWS Single Sign-On and Active Directory support**

With AWS SSO integration, members of your organization can access AWS accounts and manage their resources using Systems Manager based on a common user identity. Using AWS SSO, you can assign your users access to accounts across AWS.

Integration with Active Directory makes it possible to assign users in your Active Directory account as approvers for change templates created for your Change Manager operations.

- **Integration with Amazon CloudWatch alarms**

Change Manager is integrated with CloudWatch alarms. Change Manager listens for CloudWatch alarms during the runbook workflow and takes any actions, including sending notifications, that are defined for the alarm.

- **Integration with AWS Organizations**

Using the cross-account capabilities provided by Organizations, you can use a delegated administrator account for managing Change Manager operations in OUs in your organization. In your Organizations management account, you can specify which account is to be the delegated administrator account. You can also control which of your OUs Change Manager can be used in.

## Is there a charge to use Change Manager?

Yes. Change Manager is priced on a pay-per-use basis. You pay only for what you use. For more information, see [AWS Systems Manager Pricing](#).

# What are the primary components of Change Manager?

Change Manager components that you use to manage the change process in your organization or account include the following:

## Delegated administrator account

If you use Change Manager across an organization, you use a delegated administrator account. This is the AWS account designated as the account for managing operations activities across Systems Manager, including Change Manager. The delegated administrator account manages change activities across your organization. When you set up your organization for use with Change Manager, you specify which of your accounts serves in this role. The delegated administrator account must be the only member of the organizational unit (OU) to which it's assigned. The delegated administrator account isn't required if you use Change Manager with a single AWS account only.

### Important

If you use Change Manager across an organization, we recommend always making changes from the delegated administrator account. While you can make changes from other accounts in the organization, those changes won't be reported in or viewable from the delegated administrator account.

## Change template

A change template is a collection of configuration settings in Change Manager that define such things as required approvals, available runbooks, and notification options for change requests.

You can require that the change templates created by users in your organization or account go through an approval process before they can be used.

Change Manager supports two types of change templates. For an approved change request that is based on an *emergency change template*, the requested change can be made even if there are blocking events in Change Calendar. For an approved change request that is based on a *standard change template*, the requested change can't be made if there are blocking events in Change Calendar unless additional approvals are received from designated *change freeze event* approvers.

## Change request

A change request is a request in Change Manager to run an Automation runbook that updates one or more resources in your AWS or on-premises environments. A change request is created using a change template.

When you create a change request, one or more approvers in your organization or account must review and approve the request. Without the required approvals, the runbook workflow, which applies the changes you request, isn't permitted to run.

In the system, change requests are a type of OpsItem in AWS Systems Manager OpsCenter. However, OpsItems of the type `/aws/changerequest` aren't displayed in OpsCenter. As OpsItems, change requests are subject to the same enforced quotas as other types of OpsItems. For information about the number of OpsItems that can be created for an AWS account in an AWS Region, see [What are the quotas for OpsCenter? \(p. 188\)](#).

Additionally, to create a change request programmatically, you don't call the `CreateOpsItem` API operation. Instead, you use the `StartChangeRequestExecution` API operation. But rather than running immediately, the change request must be approved, and there must not be any blocking events in Change Calendar to prevent the workflow from running. When approvals have been received and the calendar isn't blocked (or permission has been given to bypass blocking calendar events), the `StartChangeRequestExecution` action is able to complete.

## Runbook workflow

A runbook workflow is the process of requested changes being made to the targeted resources in your cloud or on-premises environment. Each change request designates a single Automation runbook to use to make the requested change. The runbook workflow occurs after all required approvals have been granted and there are no blocking events in Change Calendar. If the change has been scheduled for a specific date and time, the runbook workflow doesn't begin until scheduled, even if all approvals have been received and the calendar isn't blocked.

### Topics

- [Setting up Change Manager \(p. 357\)](#)
- [Working with Change Manager \(p. 371\)](#)
- [Auditing and logging Change Manager activity \(p. 396\)](#)
- [Troubleshooting Change Manager \(p. 397\)](#)

## Setting up Change Manager

You can use Change Manager, a capability of AWS Systems Manager, to manage changes for an entire organization, as configured in AWS Organizations, or for a single AWS account.

If you're using Change Manager with an organization, begin with the topic [Setting up Change Manager for an organization \(management account\) \(p. 357\)](#), and then proceed to [Configuring Change Manager options and best practices \(p. 362\)](#).

If you're using Change Manager with a single account, proceed directly to [Configuring Change Manager options and best practices \(p. 362\)](#).

### Note

If you begin using Change Manager with a single account, but that account is later added to an organizational unit for which Change Manager is allowed, your single account settings are disregarded.

### Topics

- [Setting up Change Manager for an organization \(management account\) \(p. 357\)](#)
- [Configuring Change Manager options and best practices \(p. 362\)](#)
- [Configuring roles and permissions for Change Manager \(p. 366\)](#)

## Setting up Change Manager for an organization (management account)

The tasks in this topic apply if you're using Change Manager, a capability of AWS Systems Manager, with an organization that is set up in AWS Organizations. If you want to use Change Manager only with a single AWS account, skip to the topic [Configuring Change Manager options and best practices \(p. 362\)](#).

Perform the tasks in this section in an AWS account that is serving as the *management account* in Organizations. For information about the management account and other Organizations concepts, see [AWS Organizations terminology and concepts](#).

If you need to turn on Organizations and specify your account as the management account before proceeding, see [Creating and managing an organization](#) in the *AWS Organizations User Guide*.

### Note

This setup process can't be performed in the following AWS Regions:

- Europe (Milan) (eu-south-1)

- Middle East (Bahrain) (me-south-1)
- Africa (Cape Town) (af-south-1)
- Asia Pacific (Hong Kong) (ap-east-1)

Ensure that you're working in a different Region in your management account for this procedure.

During the setup procedure, you perform the following major tasks in Quick Setup, a capability of AWS Systems Manager.

- **Task 1: Register the delegated administrator account for your organization**

The change-related tasks that are performed using Change Manager are managed in one of your member accounts, which you specify to be the *delegated administrator account*. The delegated administrator account you register for Change Manager becomes the delegated administrator account for all your Systems Manager operations. (You might have delegated administrator accounts for other AWS services.) Your delegated administrator account for Change Manager, which isn't the same as your management account, manages change activities across your organization, including change templates, change requests, and approvals for each. In the delegated administrator account, you also specify other configuration options for your Change Manager operations.

**Important**

The delegated administrator account must be the only member of the organizational unit (OU) to which it's assigned in Organizations.

- **Task 2: Define and specify runbook access policies for change requester roles, or custom job functions, that you want to use for your Change Manager operations**

In order to create change requests in Change Manager, users in your member accounts must be granted AWS Identity and Access Management (IAM) permissions that allow them to access only the Automation runbooks and change templates you choose to make available to them.

**Note**

When a user creates a change request, they first select a change template. This change template might make multiple runbooks available, but the user can select only one runbook for each change request. Change templates can also be configured to allow users to include any available runbook in their requests.

To grant the needed permissions, Change Manager uses the concept of *job functions*, which is also used by IAM. However, unlike the [AWS managed policies for job functions](#) in IAM, you specify both the names of your Change Manager job functions and the IAM permissions for those job functions.

When you configure a job function, we recommend creating a custom policy and providing only the permissions needed to perform change management tasks. For instance, you might specify permissions that limit users to that specific set of runbooks based on *job functions* that you define.

For example, you might create a job function with the name `DBAdmin`. For this job function, you might grant only permissions needed for runbooks related to Amazon DynamoDB databases, such as `AWS-CreateDynamoDbBackup` and `AWSConfigRemediation-DeleteDynamoDbTable`.

As another example, you might want to grant some users only the permissions needed to work with runbooks related to Amazon Simple Storage Service (Amazon S3) buckets, such as `AWS-ConfigureS3BucketLogging` and `AWSConfigRemediation-ConfigureS3BucketPublicAccessBlock`.

The configuration process in Quick Setup for Change Manager also makes a set of full Systems Manager administrative permissions available for you to apply to an administrative role you create.

Each Change Manager Quick Setup configuration you deploy creates a job function in your delegated administrator account with permissions to run Change Manager templates and Automation runbooks

in the organizational units you have selected. You can create up to 15 Quick Setup configurations for Change Manager.

- **Task 3: Choose which member accounts in your organization to use with Change Manager**

You can use Change Manager with all the member accounts in all your organizational units that are set up in Organizations, and in all the AWS Regions they operate in. If you prefer, you can instead use Change Manager with only some of your organizational units.

**Important**

We strongly recommend, before you begin this procedure, that you read through its steps to understand the configuration choices you're making and the permissions you're granting. In particular, plan the custom job functions you will create and the permissions you assign to each job function. This ensures that when later you attach the job function policies you create to individual users, user groups, or IAM roles, they're being granted only the permissions you intend for them to have.

As a best practice, begin by setting up the delegated administrator account using the login for an AWS account administrator. Then configure job functions and their permissions after you have created change templates and identified the runbooks that each one uses.

To set up Change Manager for use with an organization, perform the following task in the Quick Setup area of the Systems Manager console.

You repeat this task for each job function you want to create for your organization. Each job function you create can have permissions for a different set of organizational units.

**To set up an organization for Change Manager in the Organizations management account**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Quick Setup**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Quick Setup** in the navigation pane.

3. Choose **Create**.

-or-

If the **Quick Setup** start page opens first, choose **Get started**, and then choose **Create**.

4. Choose **Change Manager**, and then choose **Next**.
5. For **Delegated administrator account**, enter the ID of the AWS account you want to use for managing change templates, change requests, and runbook workflows in Change Manager.

If you have previously specified a delegated administrator account for Systems Manager, its ID is already reported in this field.

**Important**

The delegated administrator account must be the only member of the organizational unit (OU) to which it's assigned in Organizations.

If the delegated administrator account you register is later deregistered from that role, the system removes its permissions for managing Systems Manager operations at the same time. Keep in mind that it will be necessary for you return to Quick Setup, designate a different delegated administrator account, and specify all job functions and permissions again.

If you use Change Manager across an organization, we recommend always making changes from the delegated administrator account. While you can make changes from other

accounts in the organization, those changes won't be reported in or viewable from the delegated administrator account.

6. In the **Permissions to request and make changes** section, do the following.

**Note**

Each deployment configuration you create provides the permissions policy for just one job function. You can return to Quick Setup later to create more job functions when you have created change templates to use in your operations.

**To create an administrative role** – For an administrator job function that has IAM permissions for all AWS actions, do the following.

**Important**

Granting users full administrative permissions should be done sparingly, and only if their roles require full Systems Manager access. For important information about security considerations for Systems Manager access, see [Identity and access management for AWS Systems Manager \(p. 1380\)](#) and [Security best practices for Systems Manager \(p. 1424\)](#).

1. For **Job function**, enter a name to identify this role and its permissions, such as **MyAWSAdmin**.
2. For **Role and permissions option**, choose **Administrator permissions**.

**To create other job functions** – To create a non-administrative role, do the following:

1. For **Job function**, enter a name to identify this role and suggest its permissions. The name you choose should represent scope of the runbooks for which you will provide permissions, such as DBAdmin or S3Admin.
2. For **Role and permissions option**, choose **Custom permissions**.
3. In the **Permissions policy editor**, enter the IAM permissions, in JSON format, to grant to this job function.

**Tip**

We recommend that you use the IAM policy editor to construct your policy and then paste the policy JSON into the **Permissions policy** field.

**Sample policy: DynamoDB database management**

For example, you might begin with policy content that provides permissions for working with the Systems Manager documents (SSM documents) the job function needs access to. Here is a sample policy content that grants access to all the AWS managed Automation runbooks related to DynamoDB databases and two change templates that have been created in the sample AWS account 123456789012, in the US East (Ohio) Region (us-east-2).

The policy also includes permission for the [StartChangeRequestExecution](#) operation, which is required for creating a change request in Change Calendar.

**Note**

This example isn't comprehensive. Additional permissions might be needed for working with other AWS resources, such as databases and nodes.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:CreateDocument",
 "ssm:DescribeDocument",
 "ssm:DescribeDocumentParameters",
 "ssm:PutParameter",
 "ssm:UpdateDocument",
 "ssm:StartChangeRequestExecution"
]
 }
]
}
```

```
 "ssm:DescribeDocumentPermission",
 "ssm:GetDocument",
 "ssm>ListDocumentVersions",
 "ssm:ModifyDocumentPermission",
 "ssm:UpdateDocument",
 "ssm:UpdateDocumentDefaultVersion"
],
 "Resource": [
 "arn:aws:ssm:us-east-2:123456789012:document/AWS-CreateDynamoDbBackup",
 "arn:aws:ssm:us-east-2:123456789012:document/AWS-AWS-
DeleteDynamoDbBackup",
 "arn:aws:ssm:us-east-2:123456789012:document/AWS-
DeleteDynamoDbTableBackups",
 "arn:aws:ssm:us-east-2:123456789012:document/AWS-AWSConfigRemediation-
DeleteDynamoDbTable",
 "arn:aws:ssm:us-east-2:123456789012:document/AWS-AWSConfigRemediation-
EnableEncryptionOnDynamoDbTable",
 "arn:aws:ssm:us-east-2:123456789012:document/AWS-AWSConfigRemediation-
EnablePITRForDynamoDbTable",
 "arn:aws:ssm:us-east-2:123456789012:document/MyFirstDBChangeTemplate",
 "arn:aws:ssm:us-east-2:123456789012:document/MySecondDBChangeTemplate"
]
},
{
 "Effect": "Allow",
 "Action": "ssm>ListDocuments",
 "Resource": "*"
},
{
 "Effect": "Allow",
 "Action": "ssm>StartChangeRequestExecution",
 "Resource": "arn:aws:ssm:us-east-2:123456789012:automation-definition/*:*"
}
]
}
```

For more information about IAM policies, see [Access management for AWS resources](#) and [Creating IAM policies](#) in the *IAM User Guide*.

7. In the **Targets** section, choose whether to grant permissions for the job function you're creating to your entire organization or only some of your organizational units.

If you choose **Entire organization**, continue to step 9.

If you choose **Custom**, continue to step 8.

8. In the **Target OUs** section, select the check boxes of the organizational units to use with Change Manager.
9. Choose **Create**.

After the system finishes setting up Change Manager for your organization, it displays a summary of your deployments. This summary information includes the name of the permissions policy that was created for the job function you configured. For example, **AWS-QuickSetup-SSMChangeMgr-DBAdminInvocationRole**. Make a note of this policy name and attach it to the users, groups, or IAM roles who will perform this job function. For information about attaching IAM policies, see [Changing permissions for an IAM user](#) in the *IAM User Guide*.

#### Note

Quick Setup uses AWS CloudFormation StackSets to deploy your configurations. You can also view information about a completed deployment configuration in the AWS CloudFormation console. For information about StackSets, see [Working with AWS CloudFormation StackSets](#) in the *AWS CloudFormation User Guide*.

Your next step is to configure additional Change Manager options. You can complete this task in either your delegated administrator account or any account in an organization unit that you have allowed for use with Change Manager. You configure options such as choosing a user identity management option, specifying which users can review and approve or reject change templates and change requests, and choosing which best practice options to allow for your organization. For information, see [Configuring Change Manager options and best practices \(p. 362\)](#).

## Configuring Change Manager options and best practices

The tasks in this section must be performed whether you're using Change Manager, a capability of AWS Systems Manager, across an organization or in a single AWS account.

If you're using Change Manager for an organization, you can perform the following tasks in either your delegated administrator account or any account in an organization unit that you have allowed for use with Change Manager.

### Topics

- [Task 1: Configuring Change Manager user identity management and template reviewers \(p. 362\)](#)
- [Task 2: Configuring Change Manager change freeze event approvers and best practices \(p. 363\)](#)
- [Configuring Amazon SNS topics for Change Manager notifications \(p. 365\)](#)

### Task 1: Configuring Change Manager user identity management and template reviewers

Perform the task in this procedure the first time you access Change Manager. You can update these configuration settings later by returning to Change Manager and choosing **Edit** on the **Settings** tab.

#### To configure Change Manager user identity management and template reviewers

1. Sign in to the AWS Management Console.

If you're using Change Manager for an organization, sign in using your credentials for your delegated administrator account. The user account you use must have the necessary AWS Identity and Access Management (IAM) permissions for making updates to your Change Manager settings.

2. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
3. In the navigation pane, choose **Change Manager**.

-or-

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

4. On the service home page, depending on the available options, do one of the following:
  - If you're using Change Manager with AWS Organizations , choose **Set up delegated account**.
  - If you're using Change Manager with a single AWS account, choose **Set up Change Manager**.

-or-

Choose **Create sample change request**, **Skip**, and then choose the **Settings** tab.

5. For **User identity management**, choose one of the following.
  - **AWS Identity and Access Management (IAM)** – Identify the users who make and approve requests and perform other action in Change Manager by using your existing IAM user accounts, groups, and roles.
  - **AWS Single Sign-On (AWS SSO)** – Allow [AWS SSO](#) to create and manage identities, or connect to your existing identity source to identify the users who perform actions in Change Manager.

6. In the **Template reviewer notification** section, specify the Amazon Simple Notification Service (Amazon SNS) topics to use to notify template reviewers that a new change template or change template version is ready for review. Ensure that the Amazon SNS topic you choose is configured to send notifications to your template reviewers.

For information about creating and configuring Amazon SNS topics for change template reviewer notifications, see [Configuring Amazon SNS topics for Change Manager notifications \(p. 365\)](#).

1. To specify the Amazon SNS topic for template reviewer notification, choose one of the following:
  - **Enter an SNS Amazon Resource Name (ARN)** – For **Topic ARN**, enter the ARN of an existing Amazon SNS topic. This topic can be in any of your organization's accounts.
  - **Select an existing SNS topic** – For **Target notification topic**, select the ARN of an existing Amazon SNS topic in your current AWS account. (This option isn't available if you haven't yet created any Amazon SNS topics in your current AWS account and AWS Region.)

**Note**

The Amazon SNS topic you select must be configured to specify the notifications it sends and the subscribers they're sent to. Its access policy must also grant permissions to Systems Manager so Change Manager can send notifications. For information, see [Configuring Amazon SNS topics for Change Manager notifications \(p. 365\)](#).

2. Choose **Add notification**.
7. In the **Change template reviewers** section, select the users in your organization or account to review new change templates or change template versions before they can be used in your operations.

Change template reviewers are responsible for verifying the suitability and security of templates other users have submitted for use in Change Manager runbook workflows.

Select change template reviewers by doing the following:

1. Choose **Add**.
2. Select the check box next to the name of each user, group, or IAM role you want to assign as a change template reviewer.
3. Choose **Add approvers**.
8. Choose **Submit**.

After you complete this initial setup process, configure additional Change Manager settings and best practices by following the steps in [Task 2: Configuring Change Manager change freeze event approvers and best practices \(p. 363\)](#).

## Task 2: Configuring Change Manager change freeze event approvers and best practices

After you complete the steps in [Task 1: Configuring Change Manager user identity management and template reviewers \(p. 362\)](#), you can designate extra reviewers for change requests during *change freeze events* and specify which available best practices you want to allow for your Change Manager operations.

A change freeze event means that restrictions are in place in the current change calendar (the calendar state in AWS Systems Manager Change Calendar is **CLOSED**). In these cases, in addition to regular approvers for change requests, or if the change request is created using a template that allow auto-approvals, change freeze approvers must grant permission for this change request to run. If they don't, the change won't be processed until the calendar state is again **OPEN**.

### To configure Change Manager change freeze event approvers and best practices

1. In the navigation pane, choose **Change Manager**.

-or-

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

2. Choose the **Settings** tab, and then choose **Edit**.
3. In the **Approvers for change freeze events** section, select the users in your organization or account who can approve changes to run even when the calendar in use in Change Calendar is currently CLOSED.

**Note**

To allow change freeze reviews, you must turn on the **Check Change Calendar for restricted change events** option in **Best practices**.

Select approvers for change freeze events by doing the following:

1. Choose **Add**.
2. Select the check box next to the name of each user, group, or IAM role you want to assign as an approver for change freeze events.
3. Choose **Add approvers**.
4. In the **Best practices** section near the bottom of the page, turn on the best practices you want to enforce for each of the following options.
  - Option: **Check Change Calendar for restricted change events**

To specify that Change Manager checks a calendar in Change Calendar to make sure changes aren't blocked by scheduled events, first select the **Enabled** check box, and then select the calendar to check for restricted events from the **Change Calendar** list.

For more information about Change Calendar, see [AWS Systems Manager Change Calendar \(p. 695\)](#).

- Option: **SNS topic for approvers for closed events**

1. Choose one of the following to specify the Amazon Simple Notification Service (Amazon SNS) topic in your account to use for sending notifications to approvers during change freeze events. (Note that you must also specify approvers in the **Approvers for change freeze events** section above **Best practices**).
  - **Enter an SNS Amazon Resource Name (ARN)** – For **Topic ARN**, enter the ARN of an existing Amazon SNS topic. This topic can be in any of your organization's accounts.
  - **Select an existing SNS topic** – For **Target notification topic**, select the ARN of an existing Amazon SNS topic in your current AWS account. (This option isn't available if you haven't yet created any Amazon SNS topics in your current AWS account and AWS Region.)

**Note**

The Amazon SNS topic you select must be configured to specify the notifications it sends and the subscribers they're sent to. Its access policy must also grant permissions to Systems Manager so Change Manager can send notifications. For information, see [Configuring Amazon SNS topics for Change Manager notifications \(p. 365\)](#).

2. Choose **Add notification**.

- Option: **Require monitors for all templates**

If you want to ensure that all templates for your organization or account specify an Amazon CloudWatch alarm to monitor your change operation, select the **Enabled** check box.

- Option: **Require template review and approval before use**

To ensure that no change requests are created, and no runbook workflows run, without being based on a template that has been reviewed and approved, select the **Enabled** check box.

5. Choose **Save**.

## Configuring Amazon SNS topics for Change Manager notifications

You can configure Change Manager, a capability of AWS Systems Manager, to send notifications to an Amazon Simple Notification Service (Amazon SNS) topic for events related to change requests and change templates. Complete the following tasks to receive notifications for the Change Manager events you add a topic to.

### Topics

- [Task 1: Create and subscribe to an Amazon SNS topic \(p. 365\)](#)
- [Task 2: Update the Amazon SNS access policy \(p. 365\)](#)
- [Task 3: \(Optional\) Update the AWS Key Management Service access policy \(p. 366\)](#)

### Task 1: Create and subscribe to an Amazon SNS topic

First, you must create and subscribe to an Amazon SNS topic. For more information, see [Creating a Amazon SNS topic](#) and [Subscribing 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 AWS account as the delegated administrator account.

### 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 Change Manager notifications to the Amazon SNS topic you created in Task 1. Without completing this task, Change Manager doesn't have permission to send notifications for the events you add the topic for.

1. Sign in to the AWS Management Console and open the Amazon SNS console at <https://console.aws.amazon.com/sns/v3/home>.
2. In the navigation pane, choose **Topics**.
3. Choose the topic you created in Task 1, and then choose **Edit**.
4. Expand **Access policy**.
5. Add and update the following `Sid` block to the existing policy and replace each `user input placeholder` with your own information .

```
{
 "Sid": "Allow Change Manager to publish to this topic",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "sns:Publish",
 "Resource": "arn:aws:sns:region:account-id:topic-name",
 "Condition": {
 "StringEquals": {
 "aws:SourceAccount": [
 "account-id"
]
 }
 }
}
```

Enter this block after the existing `Sid` block, and replace `region`, `account-id`, and `topic-name` with the appropriate values for the topic you created.

6. Choose **Save changes**.

The system now sends notifications to the Amazon SNS topic when the event type you add to topic for occurs.

**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.

### Task 3: (Optional) Update the AWS Key Management Service access policy

If you turned on 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 key you chose when you configured the topic. Use the following procedure to update the access policy so that Systems Manager can publish Change Manager approval notifications to the Amazon SNS topic you created in Task 1.

1. Open the AWS KMS console at <https://console.aws.amazon.com/kms>.
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 managed 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 and replace each `user input placeholder` with your own information .

```
{
 "Sid": "Allow Change Manager to decrypt the key",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": [
 "kms:Decrypt",
 "kms:GenerateDataKey"
],
 "Resource": "arn:aws:kms:region:account-id:key/key-id",
 "Condition": {
 "StringEquals": {
 "aws:SourceAccount": [
 "account-id"
]
 }
 }
}
```

Enter this block after one of the existing `Sid` blocks.

8. Choose **Save changes**.

## Configuring roles and permissions for Change Manager

By default, Change Manager doesn't have permission to perform actions on your resources. You must grant access by using an AWS Identity and Access Management (IAM) service role, or *assume role*. This role enables Change Manager to securely run the runbook workflows specified in an approved change

request on your behalf. The role grants AWS Security Token Service (AWS STS) [AssumeRole](#) trust to Change Manager.

By providing these permissions to a role to act on behalf of users in an organization, users don't need to be granted that array of permissions themselves. The actions allowed by the permissions are limited to approved operations only.

When users in your account or organization create a change request, they can select this assume role to perform the change operations.

You can create a new assume role for Change Manager or update an existing role with the needed permissions.

If you need to create a service role for Change Manager, complete the following tasks.

### Tasks

- [Task 1: Creating an assume role policy for Change Manager \(p. 367\)](#)
- [Task 2: Creating an assume role for Change Manager \(p. 368\)](#)
- [Task 3: Attaching the iam:PassRole policy to other roles \(p. 369\)](#)
- [Task 4: Adding inline policies to an assume role to invoke other AWS services \(p. 369\)](#)
- [Task 5: Configuring user access to Change Manager \(p. 371\)](#)

## Task 1: Creating an assume role policy for Change Manager

Use the following procedure to create the policy that you will attach to your Change Manager assume role.

### To create an assume role policy for Change Manager

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 and replace the default content with the following, which you will modify for your own Change Manager operations in following steps.

#### Note

If you're creating a policy to use with a single AWS account, and not an organization with multiple accounts and AWS Regions, you can omit the first statement block. The `iam:PassRole` permission isn't required in the case of a single account using Change Manager.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": "arn:aws:iam::delegated-admin-account-id:role/AWS-
SystemsManager-job-functionAdministrationRole",
 "Condition": {
 "StringEquals": {
 "iam:PassedToService": "ssm.amazonaws.com"
 }
 }
 },
 {
 "Effect": "Allow",
 "Action": [
```

```
 "ssm:DescribeDocument",
 "ssm:GetDocument",
 "ssm:StartChangeRequestExecution"
],
 "Resource": [
 "arn:aws:ssm:region:account-id:automation-definition/template-name":
$DEFAULT",
 "arn:aws:ssm:region::document/template-name"
]
},
{
 "Effect": "Allow",
 "Action": [
 "ssm>ListOpsItemEvents",
 "ssm:GetOpsItem",
 "ssm>ListDocuments",
 "ssm:DescribeOpsItems"
],
 "Resource": "*"
}
]
```

4. For the `iam:PassRole` action, update the `Resource` value to include the ARNs of all job functions defined for your organization that you want to grant permissions to initiate runbook workflows.
5. Replace the `region`, `account-id`, `template-name`, `delegated-admin-account-id`, and `job-function` placeholders with values for your Change Manager operations.
6. For the second `Resource` statement, modify the list to include all change templates that you want to grant permissions for. Alternatively, specify `"Resource": "*"` to grant permissions for all change templates in your organization.
7. Choose **Next: Tags**.
8. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this policy, and then choose **Next: Review**.
9. On the **Review policy** page, enter a name in the **Name** box, such as `MyChangeManagerAssumeRole`, and then enter an optional description.
10. Choose **Create policy**, and continue to [Task 2: Creating an assume role for Change Manager \(p. 368\)](#).

## Task 2: Creating an assume role for Change Manager

Use the following procedure to create a Change Manager assume role, a type of service role, for Change Manager.

### To create an assume role for Change Manager

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**, ensure that **AWS service** is selected by default.
4. In the **Choose a use case** section, choose **Systems Manager**, and then, in the **Select use case** section, choose **Systems Manager**, and then choose **Next: Permissions**.
5. On the **Attached permissions policy** page, search for the assume role policy you created in [Task 1: Creating an assume role policy for Change Manager \(p. 367\)](#), such as `MyChangeManagerAssumeRole`.
6. Select the check box next to the assume role policy name, and then 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 **MyChangeManagerAssumeRole**, and then type an optional 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.

### Task 3: Attaching the `iam:PassRole` policy to other roles

Use the following procedure to attach the `iam:PassRole` policy to an IAM instance profile or IAM service role. (The Systems Manager service uses IAM instance profiles to communicate with EC2 instances. For on-premises instances or virtual machines (VMs), or *hybrid instances*, an IAM service role is used instead.)

By attaching the `iam:PassRole` policy, the Change Manager service can pass assume role permissions to other services or Systems Manager capabilities when running runbook workflows.

#### To attach the `iam:PassRole` policy to an IAM instance profile or service role for hybrid instances

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**.
3. Search for the Change Manager assume role you created, such as **MyChangeManagerAssumeRole**, and choose its name.
4. In the **Summary** page for the assume role, choose the **Permissions** tab.
5. Choose **Add inline policy**.
6. On the **Create policy** page, choose the **Visual editor** tab.
7. Choose **Service**, and then choose **IAM**.
8. In the **Filter actions** text box, enter `PassRole`, and then choose the `PassRole` option.
9. Expand **Resources**. Verify that **Specific** is selected, and then choose **Add ARN**.
10. In the **Specify ARN for role** field, enter the ARN of the IAM instance profile role or IAM service role to which you want to pass assume role permissions. The system populates the **Account** and **Role name with path** fields.
11. Choose **Add**.
12. Choose **Review policy**.
13. On the **Review policy** page, enter a name and then choose **Create policy**.

#### Related content

- [Create an IAM instance profile for Systems Manager \(p. 21\)](#)
- [Create an IAM service role for a hybrid environment \(p. 36\)](#)

### Task 4: Adding inline policies to an assume role to invoke other AWS services

When a change request invokes other AWS services by using the Change Manager assume role, the assume role must be configured with permission to invoke those services. This requirement applies to all AWS Automation runbooks (AWS-\* runbooks) that might be used in a change request, such as the `AWS-ConfigureS3BucketLogging`, `AWS-CreateDynamoDBBackup`, and `AWS-RestartEC2Instance` runbooks. This requirement also applies to any custom runbooks 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.

### To add an inline policy to an assume role to invoke other AWS services (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 assume role that you want to update, such as `MyChangeManagerAssumeRole`.
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

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "s3:PutObject",
 "s3:GetObject"
],
 "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
 }
]
}
```

#### Amazon EC2 CreateSnapshot and DescribeSnapshots example

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "ec2:CreateSnapshot",
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": "ec2:DescribeSnapshots",
 "Resource": "*"
 }
]
}
```

For details about the IAM policy language, see [IAM JSON policy reference](#) in the *IAM User Guide*.

8. When you're 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're 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's automatically embedded in your role.

## Task 5: Configuring user access to Change Manager

If your IAM user account, group, or role is assigned administrator permissions, then you have access to Change Manager. If you don't have administrator permissions, then an administrator must assign 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 Change Manager. The user account you choose will have permission to configure and run Change Manager. 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 or a comparable policy that gives the account permissions to access Systems Manager is listed.
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 Change Manager assume role you copied at the end of [Task 2: Creating an assume role for Change Manager \(p. 368\)](#), 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 Change Manager. You can now use the Change Manager assume role ARN in your Change Manager operations.

## Working with Change Manager

With Change Manager, a capability of AWS Systems Manager, users across your organization or in a single AWS account can perform change-related tasks for which they have been granted the necessary permissions. Change Manager tasks include the following:

- Create, review, and approve or reject change templates.

A change template is a collection of configuration settings in Change Manager that define such things as required approvals, available runbooks, and notification options for change requests.

- Create, review, and approve or reject change requests.

A change request is a request in Change Manager to run an Automation runbook that updates one or more resources in your AWS or on-premises environments. A change request is created using a change template.

- Specify which users in your organization or account can be made reviewers for change templates and change requests.
- Edit configuration settings, such as how user identities are managed in Change Manager and which of the available *best practice* options are enforced in your Change Manager operations. For information about configuring these settings, see [Configuring Change Manager options and best practices \(p. 362\)](#).

### Topics

- [Working with change templates \(p. 372\)](#)
- [Working with change requests \(p. 384\)](#)
- [Reviewing change request details, tasks, and timelines \(console\) \(p. 391\)](#)
- [Viewing aggregated counts of change requests \(command line\) \(p. 392\)](#)

## Working with change templates

A change template is a collection of configuration settings in Change Manager that define such things as required approvals, available runbooks, and notification options for change requests.

### Note

AWS provides a sample [Hello World \(p. 372\)](#) change template you can use to try out Change Manager, a capability of AWS Systems Manager. However, you create your own change templates to define the changes you want to allow to the resources in your organization or account.

The changes that are made when a runbook workflow runs are based on the contents an Automation runbook. In each change template you create, you can include one or more Automation runbooks that the user making a change request can choose from to run during the update. You can also create change templates that allow requesters to choose any available Automation runbook for the change request.

To create a change template, you can use the **Builder** option in the [Create template](#) console page to build a change template. Alternatively, using the **Editor** option, you can manually author JSON or YAML content with the configuration you want for your runbook workflow. You can also use a command line tool to create a change template, with JSON content for the change template stored in an external file.

### Topics

- [Try out the AWS managed Hello World change template \(p. 372\)](#)
- [Creating change templates \(p. 373\)](#)
- [Reviewing and approving or rejecting change templates \(p. 383\)](#)
- [Deleting change templates \(p. 383\)](#)

## Try out the AWS managed Hello World change template

You can use the sample change template `AWS-HelloWorldChangeTemplate`, which uses the sample Automation runbook `AWS-HelloWorld`, to test the review and approval process after you have finished setting up Change Manager, a capability of AWS Systems Manager. This template is designed for testing or verifying your configured permissions, approver assignments, and approval process. Approval to use this change template in your organization or account has already been provided by AWS. Any change request based on this change template, however, must still be approved by reviewers in your organization or account.

Rather than make changes to a resource, the result of the runbook workflow associated with this template is to print a message in the output of an Automation step.

### Before you begin

Before you begin, ensure you have completed the following tasks:

- If you're using AWS Organizations to manage change across an organization, complete the organization setup tasks described in [Setting up Change Manager for an organization \(management account\) \(p. 357\)](#).
- Configure Change Manager for your delegated administrator account or single account, as described in [Configuring Change Manager options and best practices \(p. 362\)](#).

**Note**

If you turned on the best practice option **Require monitors for all templates** in your Change Manager settings, turn it off temporarily while you test the Hello World change template.

### To try out the AWS managed Hello World change template

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Manager**.

-or-

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

3. Choose **Create request**.
4. Choose the change template named `AWS-HelloWorldChangeTemplate`, and then choose **Next**.
5. For **Name**, enter a name for the change request that makes its purpose easy to identify, such as `MyChangeRequestTest`.
6. For the remainder of the steps to create your change request, see [Creating change requests \(p. 384\)](#).

### Next steps

For information about approving change requests, see [Reviewing and approving or rejecting change requests \(p. 389\)](#).

To view the status and results of your change request, choose the name of your change request on the **Requests** tab in Change Manager.

## Creating change templates

A change template is a collection of configuration settings in Change Manager that define such things as required approvals, available runbooks, and notification options for change requests.

You can create change templates for your operations in Change Manager, a capability of AWS Systems Manager, using the console, which includes Builder and Editor options, or command line tools.

### Topics

- [Creating change templates using Builder \(p. 373\)](#)
- [Creating change templates using Editor \(p. 376\)](#)
- [Creating change templates using command line tools \(p. 380\)](#)

### Creating change templates using Builder

Using the Builder for change templates in Change Manager, a capability of AWS Systems Manager, you can configure the runbook workflow defined in your change template without having to use JSON or YAML syntax. After you specify your options, the system converts your input into the YAML format that Systems Manager can use to run runbook workflows.

### To create a change template using Builder

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Manager**.

-or-

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

3. Choose **Create template**.
4. For **Name**, enter a name for the template that makes its purpose easy to identify, such as **UpdateEC2LinuxAMI**.
5. In the **Change template details** section, do the following:
  - For **Description**, provide a brief explanation of how and when the change template you're creating is to be used.

This description helps users who create change requests determine whether they're using the correct change template. It helps those who review change requests understand whether the request should be approved.

- For **Change template type**, specify whether you're creating a standard change template or an emergency change template.

An emergency change template is used for situations when a change must be made even if changes are otherwise blocked by an event in the calendar in use by AWS Systems Manager Change Calendar. Change requests created from an emergency change template must still be approved by its designated approvers, but the requested changes can still run even when the calendar is blocked.

- For **Runbook options**, specify the runbooks that users can choose from when creating a change request. You can add a single runbook or multiple runbooks. Alternatively, you can allow requesters to specify which runbook to use. In any of these cases, only one runbook can be included in the change request.
- For **Runbook**, select the names of the runbooks and the versions of those runbooks that users can choose from for their change requests. No matter how many runbooks you add to the change template, only one can be selected per change request.

You don't specify a runbook if you chose **Any runbook can be used** earlier.

**Tip**

Select a runbook and runbook version, and then choose **View** to examine the contents of the runbook in the Systems Manager Documents interface.

6. In the **Template information** section, use Markdown to enter information for users who create change requests from this change template. We have provided a set of questions that you can include for users who create change requests, or you can add other information and questions instead.

**Note**

Markdown is a markup language that allows you to add wiki-style descriptions to documents and individual steps within the document. For more information about using Markdown, see [Using Markdown in AWS](#).

We recommend providing questions for users to answer about their change requests to help approvers decide whether or not to grant each change request, such as listing any manual steps required to run as part of the change and a rollback plan.

**Tip**

Toggle between **Hide preview** and **Show preview** to see what your content looks like as you compose.

7. In the **Change request approvals** section, do the following:

- (Optional) If you want to allow change requests that are created from this change template to run automatically, without review by any approvers (with the exception of change freeze events), select **Enable auto-approval**.

**Note**

Enabling auto-approvals in a change template provides users with the *option of* bypassing reviewers. They can still choose to specify reviewers when creating a change request. Therefore, you must still specify reviewer options in the change template.

**Important**

If you enable auto-approval for a change template, users can submit change requests using that template that do not require review by reviewers before they run (with the exception of change freeze event approvers).

- To add mandatory first-level approvers, choose **Add approver**, and then choose from the following:
  - **Template specified approvers** – Choose one or more users, groups, or AWS Identity and Access Management (IAM) roles from your account to approve change requests created from this change template. Any change requests that are created using this template must be reviewed and approved by each approver you specify.
  - **Request specified approvers** – The user who makes the change request specifies reviewers at the time they make the request and can choose from a list of users in your account.

The number you enter in the **Required** column determines how many reviewers must be specified by a change request that uses this change template.

- For **SNS topic to notify approvers**, do the following:
  1. Choose one of the following to specify the Amazon Simple Notification Service (Amazon SNS) topic in your account to use for sending notifications to approvers that a change request is ready for their review:
    - **Enter an SNS Amazon Resource Name (ARN)** – For **Topic ARN**, enter the ARN of an existing Amazon SNS topic. This topic can be in any of your organization's accounts.
    - **Select an existing SNS topic** – For **Target notification topic**, select the ARN of an existing Amazon SNS topic in your current AWS account. (This option isn't available if you haven't yet created any Amazon SNS topics in your current AWS account and AWS Region.)
    - **Specify SNS topic when the change request is created** – The user who creates a change request can specify the Amazon SNS topic to use for notifications.

**Note**

The Amazon SNS topic you select must be configured to specify the notifications it sends and the subscribers they're sent to. Its access policy must also grant permissions to Systems Manager so Change Manager can send notifications. For information, see [Configuring Amazon SNS topics for Change Manager notifications \(p. 365\)](#).

2. Choose **Add notification**.

8. (Optional) To add an additional level of approvers, choose **Add approval level** and choose between template-specified approvers and request-specified approvers for this level. Then choose an SNS topic to notify this level of approvers.

After all approvals have been received by first-level approvers, second-level approvers are notified, and so on.

You can add a maximum of five levels of approvers in each template. You might, for example, require approvals from users in technical roles for the first level, then managerial approval for the second level.

9. In the **Monitoring** section, for **CloudWatch alarm to monitor**, enter the name of an Amazon CloudWatch alarm in the current account to monitor the progress of runbook workflows that are based on this template.

**Tip**

To create a new alarm, or to review the settings of an alarm you want to specify, choose **Open the Amazon CloudWatch console**. For information about working with CloudWatch alarms, see [Using CloudWatch Alarms](#) in the *Amazon CloudWatch User Guide*.

10. In the **Notifications** section, do the following:

1. Choose one of the following to specify the Amazon SNS topic in your account to use for sending notifications about change requests that are created using this change template:
  - **Enter an SNS Amazon Resource Name (ARN)** – For **Topic ARN**, enter the ARN of an existing Amazon SNS topic. This topic can be in any of your organization's accounts.
  - **Select an existing SNS topic** – For **Target notification topic**, select the ARN of an existing Amazon SNS topic in your current AWS account. (This option isn't available if you haven't yet created any Amazon SNS topics in your current AWS account and AWS Region.)

**Note**

The Amazon SNS topic you select must be configured to specify the notifications it sends and the subscribers they're sent to. Its access policy must also grant permissions to Systems Manager so Change Manager can send notifications. For information, see [Configuring Amazon SNS topics for Change Manager notifications \(p. 365\)](#).

2. Choose **Add notification**.

11. (Optional) In the **Tags** section, apply one or more tag key name/value pairs to the change template.

Tags are optional metadata that you assign to a resource. By using tags, you can categorize a resource in different ways, such as by purpose, owner, or environment. For example, you might want to tag a change template to identify the type of change it makes and the environment it runs in. In this case, you could specify the following key name/value pairs:

- Key=TaskType, Value=InstanceRepair
- Key=Environment, Value=Production

For more information about tagging Systems Manager resources, see [Tagging Systems Manager resources \(p. 1505\)](#).

12. Choose **Save and preview**.

13. Review the details of the change template you're creating.

If you want to make changes to the change template before submitting it for review, choose **Actions**, **Edit**.

If you're satisfied with the contents of the change template, choose **Submit for review**. The users in your organization or account who have been specified as template reviewers on the **Settings** tab in Change Manager are notified that a new change template is pending their review.

If an Amazon SNS topic has been specified for change templates, notifications are sent when the change template is rejected or approved. If you don't receive notifications related to this change template, you can return to Change Manager later to check on its status.

## [Creating change templates using Editor](#)

Use the steps in this topic to configure a change template in Change Manager, a capability of AWS Systems Manager, by entering JSON or YAML instead of using the console controls.

### **To create a change template using Editor**

1. In the navigation pane, choose **Change Manager**.

-or-

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

2. Choose **Create template**.

3. For **Name**, enter a name for the template that makes its purpose easy to identify, such as **RestartEC2LinuxInstance**.
4. Above **Change template details**, choose **Editor**.
5. In the **Document editor** section, choose **Edit**, and then enter the JSON or YAML content for your change template.

The following is an example.

**Note**

The parameter `minRequiredApprovals` is used to specify how many reviewers at a specified level must approve a change request that is created using this template.

This example demonstrates two levels of approvals. You can specify up to five levels of approvals, but only one level is required.

In the first level, the specific user "John-Doe" must approve each change request. After that, any three members of the IAM role Admin must approve the change request.

**YAML**

```
description: >-
 This change template demonstrates the feature set available for creating
 change templates for Change Manager. This template starts a Runbook workflow
 for the Automation runbook called AWS-HelloWorld.
templateInformation: >
 ### Document Name: HelloWorldChangeTemplate

 ## What does this document do?

 This change template demonstrates the feature set available for creating
 change templates for Change Manager. This template starts a Runbook workflow
 for the Automation runbook called AWS-HelloWorld.

 ## Input Parameters

 * ApproverSnsTopicArn: (Required) Amazon Simple Notification Service ARN for
 approvers.

 * Approver: (Required) The name of the approver to send this request to.

 * ApproverType: (Required) The type of reviewer.
 * Allowed Values: IamUser, IamGroup, IamRole, SSOGroup, SSOUser

 ## Output Parameters

 This document has no outputs
schemaVersion: '0.3'
parameters:
 ApproverSnsTopicArn:
 type: String
 description: Amazon Simple Notification Service ARN for approvers.
 Approver:
 type: String
 description: IAM approver
 ApproverType:
 type: String
 description: >-
 Approver types for the request. Allowed values include IamUser, IamGroup,
 IamRole, SSOGroup, and SSOUser.
 executableRunBooks:
 - name: AWS-HelloWorld
 version: '1'
 emergencyChange: false
 autoAppovable: false
 mainSteps:
```

```

- name: ApproveAction1
 action: 'aws:approve'
 timeoutSeconds: 3600
 inputs:
 Message: >-
 A sample change request has been submitted for your review in Change
 Manager. You can approve or reject this request.
 EnhancedApprovals:
 NotificationArn: '{{ ApproverSnsTopicArn }}'
 Approvers:
 - approver: John-Doe
 type: IamUser
 minRequiredApprovals: 1
- name: ApproveAction2
 action: 'aws:approve'
 timeoutSeconds: 3600
 inputs:
 Message: >-
 A sample change request has been submitted for your review in Change
 Manager. You can approve or reject this request.
 EnhancedApprovals:
 NotificationArn: '{{ ApproverSnsTopicArn }}'
 Approvers:
 - approver: Admin
 type: IamRole
 minRequiredApprovals: 3

```

#### JSON

```
{
 "description": "This change template demonstrates the feature set available for
 creating
 change templates for Change Manager. This template starts a Runbook workflow
 for the Automation runbook called AWS-HelloWorld",
 "templateInformation": "### Document Name: HelloWorldChangeTemplate\n\n
 ## What does this document do?\n
 This change template demonstrates the feature set available for creating change
 templates for Change Manager.
 This template starts a Runbook workflow for the Automation runbook called AWS-
 HelloWorld.\n\n
 ## Input Parameters\n* ApproverSnsTopicArn: (Required) Amazon Simple
 Notification Service ARN for approvers.\n
 * Approver: (Required) The name of the approver to send this request to.\n
 * ApproverType: (Required) The type of reviewer. * Allowed Values: IamUser,
 IamGroup, IamRole, SSOGroup, SSOUUser\n\n
 ## Output Parameters\nThis document has no outputs\n",
 "schemaVersion": "0.3",
 "parameters": {
 "ApproverSnsTopicArn": {
 "type": "String",
 "description": "Amazon Simple Notification Service ARN for approvers."
 },
 "Approver": {
 "type": "String",
 "description": "IAM approver"
 },
 "ApproverType": {
 "type": "String",
 "description": "Approver types for the request. Allowed values include
 IamUser, IamGroup, IamRole, SSOGroup, and SSOUUser."
 }
 },
 "executableRunBooks": [
]
}
```

```

 "name": "AWS-Helloworld",
 "version": "1"
 }
],
"emergencyChange": false,
"autoAppovable": false,
"mainSteps": [
{
 "name": "ApproveAction1",
 "action": "aws:approve",
 "timeoutSeconds": 3600,
 "inputs": {
 "Message": "A sample change request has been submitted for your review in Change Manager. You can approve or reject this request.",
 "EnhancedApprovals": {
 "NotificationArn": "{{ ApproverSnsTopicArn }}",
 "Approvers": [
 {
 "approver": "John-Doe",
 "type": "IamUser",
 "minRequiredApprovals": 1
 }
]
 }
 }
},
{
 "name": "ApproveAction2",
 "action": "aws:approve",
 "timeoutSeconds": 3600,
 "inputs": {
 "Message": "A sample change request has been submitted for your review in Change Manager. You can approve or reject this request.",
 "EnhancedApprovals": {
 "NotificationArn": "{{ ApproverSnsTopicArn }}",
 "Approvers": [
 {
 "approver": "Admin",
 "type": "IamRole",
 "minRequiredApprovals": 3
 }
]
 }
 }
}
]
}

```

6. Choose **Save and preview**.
7. Review the details of the change template you're creating.

If you want to make changes to the change template before submitting it for review, choose **Actions**, **Edit**.

If you're satisfied with the contents of the change template, choose **Submit for review**. The users in your organization or account who have been specified as template reviewers on the **Settings** tab in Change Manager are notified that a new change template is pending their review.

If an Amazon Simple Notification Service (Amazon SNS) topic has been specified for change templates, notifications are sent when the change template is rejected or approved. If you don't receive notifications related to this change template, you can return to Change Manager later to check on its status.

## Creating change templates using command line tools

The following procedures describe how to use the AWS Command Line Interface (AWS CLI) (on Linux, macOS, or Windows) or AWS Tools for Windows PowerShell to create a change request in Change Manager, a capability of AWS Systems Manager.

### To create a change template

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Create a JSON file on your local machine with a name such as `MyChangeTemplate.json`, and then paste the content for your change template into it.

#### Note

Change templates use a version of schema 0.3 that doesn't include all the same support as for Automation runbooks.

The following is an example.

#### Note

The parameter `minRequiredApprovals` is used to specify how many reviewers at a specified level must approve a change request that is created using this template.

This example demonstrates two levels of approvals. You can specify up to five levels of approvals, but only one level is required.

In the first level, the specific user "John-Doe" must approve each change request. After that, any three members of the IAM role Admin must approve the change request.

```
{
 "description": "This change template demonstrates the feature set available for creating change templates for Change Manager. This template starts a Runbook workflow for the Automation runbook called AWS-HelloWorld",
 "templateInformation": "### Document Name: HelloWorldChangeTemplate\n\n## What does this document do?\n\nThis change template demonstrates the feature set available for creating change templates for Change Manager.
This template starts a Runbook workflow for the Automation runbook called AWS-HelloWorld.\n## Input Parameters\n* ApproverSnsTopicArn: (Required) Amazon Simple Notification Service ARN for approvers.\n* Approver: (Required) The name of the approver to send this request to.\n* ApproverType: (Required) The type of reviewer. * Allowed Values: IamUser, IamGroup, IamRole, SSOGroup, SSOUser\n## Output Parameters\nThis document has no outputs.",
 "schemaVersion": "0.3",
 "parameters": {
 "ApproverSnsTopicArn": {
 "type": "String",
 "description": "Amazon Simple Notification Service ARN for approvers."
 },
 "Approver": {
 "type": "String",
 "description": "IAM approver"
 },
 "ApproverType": {
 "type": "String",
 "description": "Approver types for the request. Allowed values include IamUser, IamGroup, IamRole, SSOGroup, and SSOUser."
 }
 },
 "executableRunBooks": [
]
}
```

```

 "name": "AWS-HelloWorld",
 "version": "1"
 }
],
"emergencyChange": false,
"autoAppovable": false,
"mainSteps": [
{
 "name": "ApproveAction1",
 "action": "aws:approve",
 "timeoutSeconds": 3600,
 "inputs": {
 "Message": "A sample change request has been submitted for your review in Change Manager. You can approve or reject this request.",
 "EnhancedApprovals": {
 "NotificationArn": "{{ ApproverSnsTopicArn }}",
 "Approvers": [
 {
 "approver": "John-Doe",
 "type": "IamUser",
 "minRequiredApprovals": 1
 }
]
 }
 }
},
{
 "name": "ApproveAction2",
 "action": "aws:approve",
 "timeoutSeconds": 3600,
 "inputs": {
 "Message": "A sample change request has been submitted for your review in Change Manager. You can approve or reject this request.",
 "EnhancedApprovals": {
 "NotificationArn": "{{ ApproverSnsTopicArn }}",
 "Approvers": [
 {
 "approver": "Admin",
 "type": "IamRole",
 "minRequiredApprovals": 3
 }
]
 }
 }
}
]
}

```

- Run the following command to create the change template.

Linux & macOS

```

aws ssm create-document \
--name MyChangeTemplate \
--document-format JSON \
--document-type Automation.ChangeTemplate \
--content file://MyChangeTemplate.json \
--tags Key=tag-key,Value=tag-value

```

Windows

```

aws ssm create-document ^
--name MyChangeTemplate ^
--document-format JSON ^

```

```
--document-type Automation.ChangeTemplate ^
--content file://MyChangeTemplate.json ^
--tags Key=tag-key,Value=tag-value
```

## PowerShell

```
$json = Get-Content -Path "C:\path\to\file\MyChangeTemplate.json" | Out-String
New-SSMDocument `

-Content $json `

-Name "MyChangeTemplate" `

-DocumentType "Automation.ChangeTemplate" `

-Tags "Key=tag-key,Value=tag-value"
```

For information about other options you can specify, see [create-document](#).

The system returns information like the following.

```
{
 "DocumentDescription": {
 "CreatedDate": 1.585061751738E9,
 "DefaultVersion": "1",
 "Description": "Use this template to update an EC2 Linux AMI. Requires one approver specified in the template and an approver specified in the request.",
 "DocumentFormat": "JSON",
 "DocumentType": "Automation",
 "DocumentVersion": "1",
 "Hash": "0d3d879b3ca072e03c12638d0255ebd004d2c65bd318f8354fcde820dEXAMPLE",
 "HashType": "Sha256",
 "LatestVersion": "1",
 "Name": "MyChangeTemplate",
 "Owner": "123456789012",
 "Parameters": [
 {
 "DefaultValue": "",
 "Description": "Level one approvers",
 "Name": "LevelOneApprovers",
 "Type": "String"
 },
 {
 "DefaultValue": "",
 "Description": "Level one approver type",
 "Name": "LevelOneApproverType",
 "Type": "String"
 }
],
 "cloudWatchMonitors": {
 "monitors": [
 "my-cloudwatch-alarm"
]
 }
 },
 "PlatformTypes": [
 "Windows",
 "Linux"
],
 "SchemaVersion": "0.3",
 "Status": "Creating",
 "Tags": [
]
}
```

The users in your organization or account who have been specified as template reviewers on the **Settings** tab in Change Manager are notified that a new change template is pending their review.

If an Amazon Simple Notification Service (Amazon SNS) topic has been specified for change templates, notifications are sent when the change template is rejected or approved. If you don't receive notifications related to this change template, you can return to Change Manager later to check on its status.

## Reviewing and approving or rejecting change templates

If you're specified as a reviewer for change templates in Change Manager, a capability of AWS Systems Manager, you're notified when a new change template, or new version of a change template, is awaiting your review. An Amazon Simple Notification Service (Amazon SNS) topic sends the notifications.

### Note

This functionality depends on whether your account has been configured to use an Amazon SNS topic to send change template review notifications. For information about specifying a template reviewer notification topic, see [Task 1: Configuring Change Manager user identity management and template reviewers \(p. 362\)](#).

To review the change template, follow the link in your notification, sign in to the AWS Management Console, and follow the steps in this procedure.

### To review and approve or reject a change template

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Manager**.

-or-

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

3. In the **Change templates** section at the bottom of the **Overview** tab, choose the number in **Pending review**.
4. In the **Change templates** list, locate and choose the name of change template to review.
5. In the summary page, review the proposed content of the change template and do one of the following:
  - To approve the change template, which allows it to be used in change requests, choose **Approve**.
  - To reject the change template, which prevents it from being used in change requests, choose **Reject**.

## Deleting change templates

This topic describes how to delete templates that you have created in Change Manager, a capability of Systems Manager. If you are using Change Manager for an organization, this procedure is performed in your delegated administrator account.

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Manager**.

-or-

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

3. Choose the **Templates** tab.
4. Choose the name of the template to delete.

5. Choose **Actions, Delete template**.
6. In the confirmation dialog, enter the word **DELETE**, and then choose **Delete**.

## Working with change requests

A change request is a request in Change Manager to run an Automation runbook that updates one or more resources in your AWS or on-premises environments. A change request is created using a change template.

When you create a change request in Change Manager, a capability of AWS Systems Manager, one or more approvers in your organization or account must review and approve the request. Without the required approvals, the runbook workflow, which makes the changes you request, isn't permitted to run.

### Topics

- [Creating change requests \(p. 384\)](#)
- [Reviewing and approving or rejecting change requests \(p. 389\)](#)

## Creating change requests

When you create a change request in Change Manager, a capability of AWS Systems Manager, the change template you select typically does the following:

- Designates approvers for the change request or specifies how many approvals are required
- Specifies the Amazon Simple Notification Service (Amazon SNS) topic to use to notify approvers about your change request
- Specifies an Amazon CloudWatch alarm to monitor the runbook workflow for the change request
- Identifies which Automation runbooks you can choose from to make the requested change

In some cases, a change template might be configured so you specify your own Automation runbook to use, and to specify who should review and approve the request.

### Important

If you use Change Manager across an organization, we recommend always making changes from the delegated administrator account. While you can make changes from other accounts in the organization, those changes won't be reported in or viewable from the delegated administrator account.

### Creating change requests (console)

The following procedure describes how to create a change request by using the Systems Manager console.

#### To create a change request (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Manager**.  
-or-  
If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Change Manager**.
3. Choose **Create request**.
4. Search for and select a change template that you want to use for this change request.
5. Choose **Next**.

6. For **Name**, enter a name for the change request that makes its purpose easy to identify, such as `UpdateEC2LinuxAMI-us-east-2`.
7. For **Runbook**, select the runbook you want to use to make your requested change.

**Note**

If the option to select a runbook isn't available, the change template author has specified which runbook must be used.

8. For **Change request information**, use Markdown to provide additional information about the change request to help reviewers decide whether to approve or reject the change request. The author of the template you're using might have provided instructions or questions for you to answer.

**Note**

Markdown is a markup language that allows you to add wiki-style descriptions to documents and individual steps within the document. For more information about using Markdown, see [Using Markdown in AWS](#).

9. In the **Workflow start time** section, choose one of the following:

- **Run the operation at a scheduled time** – For **Requested start time**, enter the date and time you propose for running the runbook workflow for this request. For **Estimated end time**, enter the date and time that you expect the runbook workflow to complete. (This time is an estimate only that you're providing for reviewers.)

**Tip**

Choose [View Change Calendar](#) to check for any blocking events for the time you specify.

- **Run the operation as soon as possible after approval** – If the change request is approved, the runbook workflow runs as soon as there is a non-restricted period when changes can be made.

10. In the **Change request approvals** section, do the following:

1. If **Approval type** options are presented, choose one of the following:

- **Automatic approval** – The change template you selected is configured to allow change requests to run automatically without review by any approvers. Continue to Step 11.

**Note**

The permissions specified in the IAM policies that govern your use of Systems Manager must not restrict you from submitting auto-approval change requests in order for them to run automatically.

- **Specify approvers** – You must add one or more users, groups, or IAM roles to review and approve this change request.

**Note**

You can choose to specify reviewers even if the permissions specified in the IAM policies that govern your use of Systems Manager allow you to run auto-approval change requests.

2. Choose **Add approver**, and then select one or more users, groups, or AWS Identity and Access Management (IAM) roles from the lists of available reviewers.

**Note**

One or more approvers might already be specified. This means that mandatory approvers are already specified in the change template you have selected. These approvers can't be removed from the request. If the **Add approver** button isn't turned on, the template you have chosen doesn't allow additional reviewers to be added to requests.

3. Under **SNS topic to notify approvers**, choose one of the following to specify the Amazon SNS topic in your account to use for sending notifications to the approvers you are adding to this change request.

**Note**

If the option to specify an Amazon SNS topic isn't available, the change template you selected already specifies the Amazon SNS topic to use.

- **Enter an SNS Amazon Resource Name (ARN)** – For **Topic ARN**, enter the ARN of an existing Amazon SNS topic. This topic can be in any of your organization's accounts.
- **Select an existing SNS topic** – For **Target notification topic**, select the ARN of an existing Amazon SNS topic in your current account. (This option isn't available if you haven't yet created any Amazon SNS topics in your current AWS account and AWS Region.)

**Note**

The Amazon SNS topic you select must be configured to specify the notifications it sends and the subscribers they're sent to. Its access policy must also grant permissions to Systems Manager so Change Manager can send notifications. For information, see [Configuring Amazon SNS topics for Change Manager notifications \(p. 365\)](#).

4. Choose **Add notification**.
11. Choose **Next**.
12. For **IAM role**, select an IAM role *in your current account* that has the permissions needed to run the runbooks that are specified for this change request.

This role is also referred to as the service role, or assume role, for Automation. For more information about this role, see [Setting up Automation \(p. 401\)](#).

13. In the **Deployment location** section, choose one of the following:

**Note**

If you're using Change Manager with a single AWS account only and not with an organization set up in AWS Organizations, you don't need to specify a deployment location.

- **Apply change to this account** – The runbook workflow runs in the current account only. For an organization, this means the delegated administrator account.
- **Apply change to multiple organizational units (OUs)** – Do the following:
  1. For **Accounts and organizational units (OUs)**, enter the ID of a member account in your organization, in the format **123456789012**, or the ID of an organizational unit, in the format **o-o96EXAMPLE**.
  2. (Optional) For **Execution role name**, enter the name of the IAM role *in the target account* or OU that has the permissions needed to run the runbooks that are specified for this change request. All accounts in any OU you specify should use the same name for this role.
  3. (Optional) Choose **Add another target location** for each additional account or OU you want to specify and repeat steps a and b.
  4. For **Target AWS Region**, select the Region to make the change in, such as **Ohio (us-east-2)** for the US East (Ohio) Region.

5. Expand **Rate control**.

For **Concurrency**, enter a number, then from the list select whether this represents the number or percentage of accounts the runbook workflow can run in at the same time.

For **Error threshold**, enter a number, then from the list select whether this represents the number or percentage of accounts where runbook workflow can fail before the operation is stopped.

14. In the **Deployment targets** section, do the following:

1. Choose one of the following:
  - **Single resource** – The change is to be made for just one resource. For example, a single node or a single Amazon Machine Image (AMI), depending on the operation defined in the runbooks for this change request.
  - **Multiple resources** – For **Parameter**, select from the available parameters from the runbooks for this change request. This selection reflects the type of resource being updated.

For example, if the runbook for this change request is `AWS-RestartEC2Instance`, you might choose `InstanceId`, and then define which instances are updated by selecting from the following:

- **Specify tags** – Enter a key-value pair that all resources to be updated are tagged with.
- **Choose a resource group** – Choose the name of the resource group that all resources to be updated belong to.
- **Specify parameter values** – Identify the resources to update in the **Runbook parameters** section.
- **Target all instances** – Make the change on all managed nodes in the target locations.

2. If you chose **Multiple resources**, expand **Rate control**.

For **Concurrency**, enter a number, then from the list select whether this represents the number or percentage of targets the runbook workflow can update at the same time.

For **Error threshold**, enter a number, then from the list select whether this represents the number or percentage of targets where the update can fail before the operation is stopped.

15. If you chose **Specify parameter values** to update multiple resources in the previous step: In the **Runbook parameters** section, specify values for the required input parameters. The parameter values you must supply are based on the contents of the Automation runbooks associated with the change template you chose.

For example, if the change template uses the `AWS-RestartEC2Instance` runbook, then you must enter one or more instance IDs for the `InstanceId` parameter. Alternatively, choose **Show interactive instance picker** and select available instances one by one.

16. Choose **Next**.

17. In the **Review and submit** page, double-check the resources and options you have specified for this change request.

Choose the **Edit** button for any section you want to make changes to.

When you're satisfied with the change request details, choose **Submit for approval**.

If an Amazon SNS topic has been specified in the change template you chose for the request, notifications are sent when the request is rejected or approved. If you don't receive notifications for the request, you can return to Change Manager to check the status of your request.

## Creating change requests (AWS CLI)

You can create a change request using the AWS Command Line Interface (AWS CLI) by specifying options and parameters for the change request in a JSON file and using the `--cli-input-json` option to include it in your command.

### To create a change request (AWS CLI)

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Create a JSON file on your local machine with a name such as `MyChangeRequest.json` and paste the following content into it.

Replace `placeholders` with values for your change request.

#### Note

This sample JSON creates a change request using the `AWS-HelloWorldChangeTemplate` change template and `AWS-HelloWorld` runbook. To help you adapt this sample for your

own change requests, see [StartChangeRequestExecution](#) in the *AWS Systems Manager API Reference* for information about all available parameters,

```
{
 "ChangeRequestName": "MyChangeRequest",
 "DocumentName": "AWS-HelloWorldChangeTemplate",
 "DocumentVersion": "$DEFAULT",
 "ScheduledTime": "2021-12-30T03:00:00",
 "ScheduledEndTime": "2021-12-30T03:05:00",
 "Tags": [
 {
 "Key": "Purpose",
 "Value": "Testing"
 }
],
 "Parameters": {
 "Approver": [
 "JohnDoe"
],
 "ApproverType": [
 "IamUser"
],
 "ApproverSnsTopicArn": [
 "arn:aws:sns:us-east-2:123456789012:MyNotificationTopic"
]
 },
 "Runbooks": [
 {
 "DocumentName": "AWS-Helloworld",
 "DocumentVersion": "1",
 "MaxConcurrency": "1",
 "MaxErrors": "1",
 "Parameters": {
 "AutomationAssumeRole": [
 "arn:aws:iam::123456789012:role/MyChangeManagerAssumeRole"
]
 }
 }
],
 "ChangeDetails": "### Document Name: HelloWorldChangeTemplate\n\n## What does this document do?\nThis change template demonstrates the feature set available for creating change templates for Change Manager. This template starts a Runbook workflow for the Automation document called AWS-Helloworld.\n\n## Input Parameters\n* ApproverSnsTopicArn: (Required) Amazon Simple Notification Service ARN for approvers.\n* Approver: (Required) The name of the approver to send this request to.\n* ApproverType: (Required) The type of reviewer.\n * Allowed Values: IamUser, IamGroup, IamRole, SSOGroup, SSOUser\n\n## Output Parameters\nThis document has no outputs\n"}
}
```

3. In the directory where you created the JSON file, run the following command.

```
aws ssm start-change-request-execution --cli-input-json file://MyChangeRequest.json
```

The system returns information like the following.

```
{
 "AutomationExecutionId": "b3c1357a-5756-4839-8617-2d2a4EXAMPLE"
}
```

## Reviewing and approving or rejecting change requests

If you're specified as a reviewer for a change request in Change Manager, a capability of AWS Systems Manager, you're notified through an Amazon Simple Notification Service (Amazon SNS) topic when a new change request is awaiting your review.

**Note**

This functionality depends on whether an Amazon SNS was specified in the change template for sending review notifications. For information, see [Configuring Amazon SNS topics for Change Manager notifications \(p. 365\)](#).

To review the change request, you can follow the link in your notification, or sign in to the AWS Management Console directly and follow the steps in this procedure.

**Note**

If an Amazon SNS topic is assigned for reviewers in a change template, notifications are sent to the topic's subscribers when the change request changes status.

### Reviewing and approving or rejecting change requests (console)

The following procedures describe how to use the Systems Manager console to review and approve or reject change requests.

#### To review and approve or reject a single change request

1. Open the link in the email notification you received and sign in to the AWS Management Console, which directs you to the change request for your review.
2. In the summary page, review the proposed content of the change request.

To approve the change request, choose **Approve**. In the dialog box, provide any comments you want to add for this approval, and then choose **Approve**. The runbook workflow represented by this request starts to run either when scheduled, or as soon as changes aren't blocked by any restrictions.

-or-

To reject the change request, choose **Reject**. In the dialog box, provide any comments you want to add for this rejection, and then choose **Reject**.

#### To review and approve or reject change requests in bulk

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Manager**.

-or-

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

3. Choose the **Approvals** tab.
4. (Optional) Review the details of requests pending your approval by choosing the name of each request, and then return to the **Approvals** tab.
5. Select the check box of each change request that you want to approve.

-or-

Select the check box of each change request that you want to reject.

6. In the dialog box, provide any comments you want to add for this approval or rejection.

7. Depending on whether you're approving or rejecting the selected change requests, choose **Approve** or **Reject**.

### Reviewing and approving or rejecting a change request (command line)

The following procedure describes how to use the AWS Command Line Interface (AWS CLI) (on Linux, macOS, or Windows) to review and approve or reject a change request.

#### To review and approve or reject a change request

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Create a JSON file on your local machine that specifies the parameters for your AWS CLI call.

```
{
 "OpsItemFilters":
 [
 {
 "Key": "OpsItemType",
 "Values": ["/aws/changerequest"],
 "Operator": "Equal"
 }
],
 "MaxResults": number
}
```

You can filter the results for a specific approver by specifying the approver's Amazon Resource Name (ARN) in the JSON file. Here is an example.

```
{
 "OpsItemFilters":
 [
 {
 "Key": "OpsItemType",
 "Values": ["/aws/changerequest"],
 "Operator": "Equal"
 },
 {
 "Key": "ChangeRequestByApproverArn",
 "Values": ["arn:aws:iam::account-id:user/user-name"],
 "Operator": "Equal"
 }
],
 "MaxResults": number
}
```

3. Run the following command to view the maximum number of change requests you specified in the JSON file.

Linux & macOS

```
aws ssm describe-ops-items \
--cli-input-json file://filename.json
```

Windows

```
aws ssm describe-ops-items ^
--cli-input-json file://filename.json
```

- Run the following command to approve or reject a change request.

Linux & macOS

```
aws ssm send-automation-signal \
--automation-execution-id ID \
--signal-type Approve_or_Reject \
--payload Comment="message"
```

Windows

```
aws ssm send-automation-signal ^
--automation-execution-id ID ^
--signal-type Approve_or_Reject ^
--payload Comment="message"
```

If an Amazon SNS topic has been specified in the change template you chose for the request, notifications are sent when the request is rejected or approved. If you don't receive notifications for the request, you can return to Change Manager to check the status of your request. For information about other options when using this command, see [send-automation-signal](#) in the AWS Systems Manager section of the *AWS CLI Command Reference*.

## Reviewing change request details, tasks, and timelines (console)

You can view information about a change request, including requests for which changes have already been processed, in the dashboard of Change Manager, a capability of AWS Systems Manager. These details include a link to the Automation operation that runs the runbooks that make the change. An Automation execution ID is generated when the request is created, but the process doesn't run until all approvals have been given and no restrictions are in place to block the change.

### To review change request details, tasks, and timelines

- In the navigation pane, choose **Change Manager**.

-or-

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

- Choose the **Requests** tab.
- In the **Change requests** section, search for the change request you want to review.

You can use the **Create date range** options to limit results to a specific time period.

You can filter requests by the following properties:

- Status
- Request ID
- Approver
- Requester

For example, to view details about all change requests that have completed successfully in the past 24 hours, do the following:

- For **Create date range**, choose **1d**.

2. In the search box, select **Status, CompletedWithSuccess**.
3. In the results, choose the name of the successfully completed change request to review results for.
4. View information about the change request on the following tabs:
  - **Request details** – View basic details about the change request, including the requester, the change template, and the Automation runbooks selected for the change. You can also follow a link to the Automation operation details and view information about any runbook parameters specified in the request, Amazon CloudWatch alarms assigned to the change request, and approvals and comments provided for the request.
  - **Task** – View information about the task in the change, including task status for completed change requests, the targeted resources, the steps in the associated Automation runbooks, and concurrency and error threshold details.
  - **Timeline** – View a summary of all events associated with the change request, listed by date and time. The summary indicates when the change request was created, actions by assigned approvers, a note of when approved change requests are scheduled to run, runbook workflow details, and status changes for the overall change process and each step in the runbook.

## Viewing aggregated counts of change requests (command line)

You can view aggregated counts of change requests in Change Manager, a capability of AWS Systems Manager, by using the [GetOpsSummary](#) API operation. This API operation can return counts for a single AWS account in a single AWS Region or for multiple accounts and multiple Regions.

### Note

If you want to view aggregated counts of change requests for multiple AWS accounts and multiple AWS Regions, you must set up and configure a resource data sync. For more information, see [Configuring resource data sync for Inventory \(p. 858\)](#).

The following procedure describes how to use the AWS Command Line Interface (AWS CLI) (on Linux, macOS, or Windows) to view aggregated counts of change requests.

### To view aggregated counts of change requests

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run one of the following commands.

#### Single account and Region

This command returns a count of all change requests for the AWS account and AWS Region for which your AWS CLI session is configured.

Linux & macOS

```
aws ssm get-ops-summary \
--filters Key=AWS:OpsItem.OpsItemType,Values="/aws/changerequests",Type=Equal \
--aggregators AggregatorType=count,AttributeName=Status,TypeName=AWS:OpsItem
```

Windows

```
aws ssm get-ops-summary ^
--filters Key=AWS:OpsItem.OpsItemType,Values="/aws/changerequests",Type=Equal ^
--aggregators AggregatorType=count,AttributeName=Status,TypeName=AWS:OpsItem
```

The call returns information like the following.

```
{
 "Entities": [
 {
 "Data": {
 "AWS:OpsItem": {
 "Content": [
 {
 "Count": "38",
 "Status": "Open"
 }
]
 }
 }
]
 }
}
```

### Multiple accounts and/or Regions

This command returns a count of all change requests for the AWS accounts and AWS Regions specified in the resource data sync.

Linux & macOS

```
aws ssm get-ops-summary \
--sync-name resource_data_sync_name \
--filters Key=AWS:OpsItem.OpsItemType,Values="/aws/changerequests",Type=Equal \
--aggregators AggregatorType=count,AttributeName=Status,TypeName=AWS:OpsItem
```

Windows

```
aws ssm get-ops-summary ^
--sync-name resource_data_sync_name ^
--filters Key=AWS:OpsItem.OpsItemType,Values="/aws/changerequests",Type=Equal ^
--aggregators AggregatorType=count,AttributeName=Status,TypeName=AWS:OpsItem
```

The call returns information like the following.

```
{
 "Entities": [
 {
 "Data": {
 "AWS:OpsItem": {
 "Content": [
 {
 "Count": "43",
 "Status": "Open"
 },
 {
 "Count": "2",
 "Status": "Resolved"
 }
]
 }
 }
]
 }
}
```

```
]
}
```

### Multiple accounts and a specific Region

This command returns a count of all change requests for the AWS accounts specified in the resource data sync. However, it only returns data from the Region specified in the command.

Linux & macOS

```
aws ssm get-ops-summary \
 --sync-name resource_data_sync_name \
 --filters Key=AWS:OpsItem.SourceRegion,Values='Region',Type=Equal \
 Key=AWS:OpsItem.OpsItemType,Values="/aws/changerequests",Type=Equal \
 --aggregators AggregatorType=count,AttributeName=Status,TypeName=AWS:OpsItem
```

Windows

```
aws ssm get-ops-summary ^
 --sync-name resource_data_sync_name ^
 --filters Key=AWS:OpsItem.SourceRegion,Values='Region',Type=Equal
 Key=AWS:OpsItem.OpsItemType,Values="/aws/changerequests",Type=Equal ^
 --aggregators AggregatorType=count,AttributeName=Status,TypeName=AWS:OpsItem
```

### Multiple accounts and Regions with output grouped by Region

This command returns a count of all change requests for the AWS accounts and AWS Regions specified in the resource data sync. The output displays count information per Region.

Linux & macOS

```
aws ssm get-ops-summary \
 --sync-name resource_data_sync_name \
 --filters Key=AWS:OpsItem.OpsItemType,Values="/aws/changerequests",Type=Equal \
 --aggregators
 '[{"AggregatorType": "count", "TypeName": "AWS:OpsItem", "AttributeName": "Status", "Aggregators": [
 {"AggregatorType": "count", "TypeName": "AWS:OpsItem", "AttributeName": "SourceRegion"}]}]'
```

Windows

```
aws ssm get-ops-summary ^
 --sync-name resource_data_sync_name ^
 --filters Key=AWS:OpsItem.OpsItemType,Values="/aws/changerequests",Type=Equal ^
 --aggregators
 '[{"AggregatorType": "count", "TypeName": "AWS:OpsItem", "AttributeName": "Status", "Aggregators": [
 {"AggregatorType": "count", "TypeName": "AWS:OpsItem", "AttributeName": "SourceRegion"}]}]'
```

The call returns information like the following.

```
{
 "Entities": [
 {
 "Data": {
 "AWS:OpsItem": {
 "Content": [
 {
 "Count": "38",
 "Region": "us-east-1",
 "SyncName": "ResourceDataSync",
 "Type": "OpsItem",
 "Value": "aws/changerequests",
 "Version": 1
 }
]
 }
 }
 }
]
}
```

```
 "SourceRegion": "us-east-1",
 "Status": "Open"
 },
 {
 "Count": "4",
 "SourceRegion": "us-east-2",
 "Status": "Open"
 },
 {
 "Count": "1",
 "SourceRegion": "us-west-1",
 "Status": "Open"
 },
 {
 "Count": "2",
 "SourceRegion": "us-east-2",
 "Status": "Resolved"
 }
]
}
}
]
```

## Multiple accounts and Regions with output grouped by accounts and Regions

This command returns a count of all change requests for the AWS accounts and AWS Regions specified in the resource data sync. The output groups the count information by accounts and Regions.

Linux & macOS

```
aws ssm get-ops-summary \
--sync-name resource_data_sync_name \
--filters Key=AWS:OpsItem.OpsItemType,Values="/aws/changerequests",Type=Equal \
--aggregators
'[{{"AggregatorType": "count", "TypeName": "AWS:OpsItem", "AttributeName": "Status", "Aggregators": [
[{"AggregatorType": "count", "TypeName": "AWS:OpsItem", "AttributeName": "SourceAccountId", "Aggregators": [
[{"AggregatorType": "count", "TypeName": "AWS:OpsItem", "AttributeName": "SourceRegion"}]]}]}]'
```

Windows

```
aws ssm get-ops-summary ^
--sync-name resource_data_sync_name ^
--filters Key=AWS:OpsItem.OpsItemType,Values="/aws/changerequests",Type=Equal ^
--aggregators
'[{ "AggregatorType": "count", "TypeName": "AWS:OpsItem", "AttributeName": "Status", "Aggregators": [
{ "AggregatorType": "count", "TypeName": "AWS:OpsItem", "AttributeName": "SourceAccountId", "Aggregators": [
{ "AggregatorType": "count", "TypeName": "AWS:OpsItem", "AttributeName": "SourceRegion"}] }] }]'
```

The call returns information like the following:

```
{
 "Entities": [
 {
 "Data": {
 "AWS:OpsItem": {
 "Content": [
 {
 "Content": "Value 1"
 },
 {
 "Content": "Value 2"
 }
]
 }
 }
 }
]
}
```

```
 "Count": "38",
 "SourceAccountId": "123456789012",
 "SourceRegion": "us-east-1",
 "Status": "Open"
 },
 {
 "Count": "4",
 "SourceAccountId": "111122223333",
 "SourceRegion": "us-east-2",
 "Status": "Open"
 },
 {
 "Count": "1",
 "SourceAccountId": "111122223333",
 "SourceRegion": "us-west-1",
 "Status": "Open"
 },
 {
 "Count": "2",
 "SourceAccountId": "444455556666",
 "SourceRegion": "us-east-2",
 "Status": "Resolved"
 },
 {
 "Count": "1",
 "SourceAccountId": "222222222222",
 "SourceRegion": "us-east-1",
 "Status": "Open"
 }
]
}
]
```

## Auditing and logging Change Manager activity

You can audit activity in Change Manager, a capability of AWS Systems Manager, by using Amazon CloudWatch and AWS CloudTrail alarms.

For more information about auditing and logging options for Systems Manager, see [Monitoring AWS Systems Manager \(p. 1428\)](#).

### Audit Change Manager activity using CloudWatch alarms

You can configure and assign a CloudWatch alarm to a change template. If any conditions defined in the alarm are met, the actions specified for the alarm are taken. In the alarm configuration, you can specify an Amazon Simple Notification Service (Amazon SNS) topic to notify when an alarm condition is met.

For information about creating a Change Manager template, see [Working with change templates \(p. 372\)](#).

For information about creating CloudWatch alarms, see [Using CloudWatch Alarms in the Amazon CloudWatch User Guide](#).

### Audit Change Manager activity using CloudTrail

CloudTrail captures API calls made in the Systems Manager console, the AWS Command Line Interface (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, where it's stored. One bucket is used for all CloudTrail logs for your account.

Logs of Change Manager actions show change template document creation, change template and change request approvals and rejections, activity generated by Automation runbooks, and more. For more information about viewing and using CloudTrail logs of Systems Manager activity, see [Logging AWS Systems Manager API calls with AWS CloudTrail \(p. 1442\)](#).

## Troubleshooting Change Manager

Use the following information to help you troubleshoot problems with Change Manager, a capability of AWS Systems Manager.

### Topics

- “Group *{GUID}* not found” error during change request approvals when using Active Directory (groups) (p. 397)

### “Group *{GUID}* not found” error during change request approvals when using Active Directory (groups)

**Problem:** When AWS Single Sign-On (AWS SSO) is used for user identity management, a member of an Active Directory group who is granted approval permissions in Change Manager receives a “not authorized” or “group not found” error.

- **Solution:** When you select Active Directory groups in AWS SSO for access to the AWS Management Console, the system schedules a periodic synchronization that copies information from those Active Directory groups into AWS SSO. This process must complete before users authorized through Active Directory group membership can successfully approve a request. For more information, see [Connect to your Microsoft AD directory](#) in the [AWS Single Sign-On User Guide](#).

## AWS Systems Manager Automation

Automation, a capability of AWS Systems Manager, simplifies common maintenance, deployment, and remediation tasks for AWS services like Amazon Elastic Compute Cloud (Amazon EC2), Amazon Relational Database Service (Amazon RDS), Amazon Redshift, Amazon Simple Storage Service (Amazon S3), and many more. To get started with Automation, open the [Systems Manager console](#). In the navigation pane, choose **Automation**.

Automation helps you to build automated solutions to deploy, configure, and manage AWS resources at scale. With Automation, you have granular control over the concurrency of your automations. This means you can specify how many resources to target concurrently, and how many errors can occur before an automation is stopped.

To help you get started with Automation, AWS develops and maintains several pre-defined runbooks. Depending on your use case, you can use these pre-defined runbooks that perform a variety of tasks, or create your own custom runbooks that might better suit your needs. To monitor the progress and status of your automations, you can use the Systems Manager Automation console, or your preferred command line tool. Automation also integrates with Amazon EventBridge to help you build event-driven architecture at scale.

## How can Automation benefit my organization?

Automation offers these benefits:

- **Scripting support in runbook content**

Using the `aws:executeScript` action, you can run custom Python and PowerShell functions directly from your runbooks. This provides you greater flexibility in creating your custom runbooks because you can complete various tasks that other Automation actions don't support. You also have greater control over the logic of the runbook. For an example of how this action can be used and how it can help to improve an existing automated solution, see [Authoring Automation runbooks \(p. 604\)](#).

- **Run automations across multiple AWS accounts and AWS Regions from a centralized location**

Administrators can run automations on resources across multiple accounts and Regions from the Systems Manager console.

- **Enhanced operations security**

Administrators have a centralized place to grant and revoke access to runbooks. Using only AWS Identity and Access Management (IAM) policies, you can control which individual users or groups in your organization can use Automation and which runbooks they can access. For an example of how to delegate access to an automation, see [Running an automation by using delegated administration \(p. 464\)](#).

- **Automate common IT tasks**

Automating common tasks can help improve operational efficiency, enforce organizational standards, and reduce operator errors. For example, you can use the `AWS-UpdateCloudFormationStackWithApproval` runbook to update resources that were deployed by using an AWS 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.

- **Safely perform disruptive tasks in bulk**

Automation includes features, like rate controls, that allow you to control the deployment of an automation across your fleet by specifying a concurrency value and an error threshold. For more information about working with rate controls, see [Running automations that use targets and rate controls \(p. 421\)](#).

- **Streamline complex tasks**

Automation provides pre-defined runbooks that streamline complex and time-consuming tasks such as creating golden Amazon Machine Images (AMIs). For example, you can use the `AWS-UpdateLinuxAmi` and `AWS-UpdateWindowsAmi` runbooks to create golden AMIs from a source AMI. Using these runbooks, you can run custom scripts before and after updates are applied. You can also include or exclude specific software packages from being installed. For examples of how to use these runbooks, see [Automation walkthroughs \(p. 604\)](#).

- **Define constraints for inputs**

You can define constraints in custom runbooks to limit the values that Automation will accept for a particular input parameter. For example, `allowedPattern` will only accept values for an input parameter that match the regular expression you define. If you specify `allowedValues` for an input parameter, only the values you've specified in the runbook are accepted.

- **Log automation action output to Amazon CloudWatch Logs**

To meet operational or security requirements in your organization, you might need to provide a record of the scripts run during a runbook. With CloudWatch Logs, you can monitor, store, and access log files from various AWS services. You can send output from the `aws:executeScript` action to a CloudWatch Logs log group for debugging and troubleshooting purposes. Log data can be sent to your log group with or without AWS KMS encryption using your KMS key. For more information, see [Logging Automation action output with CloudWatch Logs \(p. 1445\)](#).

- **Amazon EventBridge integration**

Automation is supported as a *target* type in Amazon EventBridge rules. This means you can trigger runbooks by using events. For more information, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#) and [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#).

- **Share organizational best practices**

You can define best practices for resource management, operations tasks, and more in runbooks that you share across accounts and Regions.

## Who should use Automation?

- Any AWS customer who wants to improve their operational efficiency at scale, reduce errors associated with manual intervention, and reduce time to resolution of common issues.
- Infrastructure experts who want to automate deployment and configuration tasks.
- Administrators who want to reliably resolve common issues, improve troubleshooting efficiency, and reduce repetitive operations.
- Users who want to automate a task they normally perform manually.

## What is an automation?

An *automation* consists of all of the tasks that are defined in a runbook, and are performed by the Automation service. Automation uses the following components to run automations.

Concept	Details
Automation runbook	<p>A Systems Manager Automation runbook defines the automation (the actions that Systems Manager performs on your managed nodes and AWS resources). Automation includes several pre-defined runbooks that you can use to perform common tasks like restarting one or more Amazon EC2 instances or creating an Amazon Machine Image (AMI). You can create your own runbooks as well. Runbooks use YAML or JSON, and they include steps and parameters that you specify. Steps run in sequential order. For more information, see <a href="#">Working with runbooks (p. 539)</a>.</p> <p>Runbooks are Systems Manager documents of type Automation, as opposed to Command, Policy, Session documents. Runbooks support schema version 0.3. Command documents use schema version 1.2, 2.0, or 2.2. Policy documents use schema version 2.0 or later.</p>
Automation action	<p>The automation defined in a runbook includes one or more steps. Each step is associated with a particular action. The action determines the inputs, behavior, and outputs of the step. Steps are defined in the <code>mainSteps</code> section of your runbook. Automation supports 20 distinct action</p>

Concept	Details
	types. For more information, see the <a href="#">Systems Manager Automation actions reference (p. 477)</a> .
Automation quota	Each AWS account can run 100 automations simultaneously. This includes child automations (automations that are started by another automation), and rate control automations. If you attempt to run more automations than this, Systems Manager adds the additional automations to a queue and displays a status of Pending. For more information about running automations, see <a href="#">Running a simple automation (p. 407)</a> .
Automation queue quota	If you attempt to run more automations than the concurrent automation limit, subsequent automations are added to a queue. Each AWS account can queue 1,000 automations. When an automation is complete (or reaches a terminal state), the first automation in the queue is started.
Rate control automation quota	Each AWS account can run 25 rate control automations simultaneously. If you attempt to run more rate control automations than the concurrent rate control automation limit, Systems Manager adds the subsequent rate control automations to a queue and displays a status of Pending. For more information about running rate control automations, see <a href="#">Running automations that use targets and rate controls (p. 421)</a> .
Rate control automation queue quota	If you attempt to run more automations than the concurrent rate control automation limit, subsequent automations are added to a queue. Each AWS account can queue 1,000 rate control automations. When an automation is complete (or reaches a terminal state), the first automation in the queue is started.

## Topics

- [Setting up Automation \(p. 401\)](#)
- [Working with automations \(p. 407\)](#)
- [Systems Manager Automation actions reference \(p. 477\)](#)
- [Working with runbooks \(p. 539\)](#)
- [Systems Manager Automation runbook reference \(p. 604\)](#)
- [Automation walkthroughs \(p. 604\)](#)
- [Understanding automation statuses \(p. 690\)](#)
- [Troubleshooting Systems Manager Automation \(p. 691\)](#)

## Setting up Automation

To set up Automation, a capability of AWS Systems Manager, you must verify user access to the Automation service and situationally configure roles so that the service can perform actions on your resources. We also recommend that you opt in to the adaptive concurrency mode in your Automation preferences. Adaptive concurrency automatically scales your automation quota to meet your needs. For more information, see [Allowing Automation to adapt to your concurrency needs \(p. 420\)](#).

To ensure proper access to AWS Systems Manager Automation, review the following user and service role requirements.

### Verifying user access for runbooks

Verify that you have permission to use runbooks. 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 runbooks require permissions to other services, such as the runbook `AWS-ReleaseElasticIP`, which requires IAM permissions for `ec2:ReleaseAddress`. Therefore, you must review the actions taken in a runbook to ensure your IAM user account, group, or role is assigned the necessary permissions to perform the actions included in the runbook.

### Configuring a service role (assume role) access for automations

Automations 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 don't specify an assume role, Automation uses the context of the user who invoked the automation.

However, the following situations require that you specify a service role for Automation:

- When you want to restrict a user's permissions on a resource, but you want the user to run an automation that requires elevated permissions. In this scenario, you can create a service role with elevated permissions and allow the user to run the automation.
- When you create a Systems Manager State Manager association that runs a runbook.
- When you have operations that you expect to run longer than 12 hours.
- When you're running a runbook 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 using runbooks \(p. 545\)](#).

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 \(p. 401\)](#)
- [Method 2: Use IAM to configure roles for Automation \(p. 403\)](#)

### Method 1: Use AWS CloudFormation to configure a service role for Automation

You can create a service role for Automation, a capability of AWS Systems Manager, from an AWS CloudFormation template. After you create the service role, you can specify the service role in runbooks

using the parameter `AutomationAssumeRole`. For information about how to run an automation using the Automation service role, see [Running an automation by using an IAM service role \(p. 460\)](#).

## Create the service role using AWS CloudFormation

Use the following procedure to create the required AWS Identity and Access Management (IAM) role for Systems Manager Automation by using AWS CloudFormation.

### To create the required IAM role

1. Download and unzip the [AWS-SystemsManager-AutomationServiceRole.zip](#) file. This file includes the `AWS-SystemsManager-AutomationServiceRole.yaml` AWS CloudFormation template file.
2. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
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**.

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 runbooks (`AWS-*` runbooks) such as the `AWS-ConfigureS3BucketLogging`, `AWS-CreateDynamoDBBackup`, and `AWS-RestartEC2Instance` runbooks, to name a few. This requirement also applies to any custom Automation runbooks 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, configure the service role with permission to invoke those services. You can give 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 \(p. 405\)](#).

## 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 use a runbook.

#### Note

You don't need to copy role information using this procedure if you run the `AWS-UpdateLinuxAmi` or `AWS-UpdateWindowsAmi` runbooks. These runbooks already have the required roles specified as default values. The roles specified in these runbooks use IAM managed policies.

### To copy the role names

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
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 runbooks.

## Method 2: Use IAM to configure roles for Automation

If you need to create a service role for Automation, a capability of AWS Systems Manager, complete the following tasks. For more information about when a service role is required for Automation, see [Setting up Automation \(p. 401\)](#).

### Tasks

- [Task 1: Create a service role for Automation \(p. 403\)](#)
- [Task 2: Attach the iam:PassRole policy to your Automation role \(p. 406\)](#)
- [Task 3: Configure user access to Automation \(p. 406\)](#)

### 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 runbooks, such as the `AWS-CreateManagedLinuxInstance` runbook. Using this role, or the Amazon Resource Name (ARN) of an AWS Identity and Access Management (IAM) role, in runbooks allows 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

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 **Attached permissions policy** page, search for the **AmazonSSMAutomationRole** policy, choose it, and then choose **Next: Review**.
6. On the **Review** page, enter a name in the **Role name** box, and then enter 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 runbooks.

#### 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 don't 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.

After creating your service role, we recommend editing the trust policy to help prevent the cross-service confused deputy problem. The *confused deputy problem* is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that Automation gives another service to the resource. If the `aws:SourceArn` value doesn't contain the account ID, such as an Amazon S3 bucket ARN, you must use both global condition context keys to limit permissions. If you use both global condition context keys and the `aws:SourceArn` value contains the account ID, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement. Use `aws:SourceArn` if you want only one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use. The value of `aws:SourceArn` must be the ARN for automation executions. If you don't know the full ARN of the resource or if you're specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (\*) for the unknown portions of the ARN. For example, `arn:aws:ssm:*:123456789012:automation-execution/*`.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys for Automation to prevent the confused deputy problem.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "Service": [
 "ssm.amazonaws.com"
]
 },
 "Action": "sts:AssumeRole",
 "Condition": {
 "StringEquals": {
 "aws:SourceAccount": "123456789012"
 },
 "ArnLike": {
 "aws:SourceArn": "arn:aws:ssm:*:123456789012:automation-execution/*"
 }
 }
 }
]
}
```

### To modify the role's trust policy

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**.
3. In the list of roles in your account, choose the name of your Automation service role.
4. Choose the **Trust relationships** tab, and then choose **Edit trust relationship**.
5. Edit the trust policy using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys for Automation to prevent the confused deputy problem.

6. Choose **Update Trust Policy** to save your changes.

#### (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 runbooks (AWS-\* runbooks) such as the AWS-ConfigureS3BucketLogging, AWS-CreateDynamoDBBackup, and AWS-RestartEC2Instance runbooks, to name a few. This requirement also applies to any custom runbooks 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 give 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

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

#### Amazon EC2 CreateSnapshot and DescribeSnapshots Example

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "ec2:CreateSnapshot",
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": "ec2:DescribeSnapshots",
 "Resource": "*"
 }
]
}
```

```
]
}
```

For details about the IAM policy language, see [IAM JSON Policy Reference](#) in the *IAM User Guide*.

8. When you're 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're 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's 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 allows the Automation service to pass the role to other services or Systems Manager capabilities when running automations.

### 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.

9. Choose **Add**.
10. Choose **Review policy**.
11. On the **Review Policy** page, enter 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 the **Visual editor** tab, and then choose **Choose a service**.
5. In the search box, enter **IAM**, or scroll down to find **IAM** lower in the page, and choose **IAM**.
6. For **Actions**, enter **PassRole** in the search box, and choose **PassRole**.
7. Expand 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 runbooks.

## Working with automations

This section includes information about how to run Automation runbooks. Automation is a capability of AWS Systems Manager. For more examples of how to run automations, see [Automation walkthroughs \(p. 604\)](#).

### Contents

- [Running a simple automation \(p. 407\)](#)
- [Running an automation manually \(p. 411\)](#)
- [Running an automation with approvers \(p. 417\)](#)
- [Allowing Automation to adapt to your concurrency needs \(p. 420\)](#)
- [Running automations that use targets and rate controls \(p. 421\)](#)
- [Running automations based on triggers \(p. 435\)](#)
- [Running automations by using different security models \(p. 456\)](#)
- [Running automations in multiple AWS Regions and accounts \(p. 467\)](#)

## Running a simple automation

The following procedures describe how to run a simple AWS Systems Manager automation using the Systems Manager console and AWS Command Line Interface (AWS CLI). The automation 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 runbook, and any actions called by the runbook. If you have administrator permissions in IAM, then you already have permission to run this automation.

#### Note

For information about how to run an automation that uses an IAM service role or more advanced forms of delegated administration, see [Running automations by using different security models \(p. 456\)](#).

### Running a simple automation (console)

The following procedure describes how to use the Systems Manager console to run a simple automation.

#### To run a simple automation

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the **Automation document** list, choose a runbook. Choose one or more options in the **Document categories** pane to filter SSM documents according to their purpose. To view a runbook that you own, choose the **Owned by me** tab. To view a runbook that is shared with your account, choose the **Shared with me** tab. To view all runbooks, choose the **All documents** tab.

**Note**

You can view information about a runbook by choosing the runbook 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 runbook is updated periodically and a new default version is assigned.
  - **Latest version at runtime:** Choose this option if the Automation runbook 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. If the automation fails to run, see [Troubleshooting Systems Manager Automation \(p. 691\)](#).

## Running a simple automation (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.

### To run a simple automation

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to start a simple automation. Replace each *example resource placeholder* with your own information.

Linux & macOS

```
aws ssm start-automation-execution \
--document-name runbook name \
--parameters runbook parameters
```

Windows

```
aws ssm start-automation-execution ^
--document-name runbook name ^
--parameters runbook parameters
```

PowerShell

```
Start-SMAutomationExecution ^
-DocumentName runbook name ^
-Parameter runbook parameters
```

Here is an example using the runbook `AWS-RestartEC2Instance` to restart the specified EC2 instance.

Linux & macOS

```
aws ssm start-automation-execution \
--document-name "AWS-RestartEC2Instance" \
--parameters "InstanceId=i-02573cafefEXAMPLE"
```

Windows

```
aws ssm start-automation-execution ^
--document-name "AWS-RestartEC2Instance" ^
--parameters "InstanceId=i-02573cafefEXAMPLE"
```

PowerShell

```
Start-SMAutomationExecution `-
-DocumentName AWS-RestartEC2Instance `-
-Parameter @{"InstanceId"="i-02573cafefEXAMPLE"}
```

The system returns information like the following.

Linux & macOS

```
{
 "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab"
}
```

Windows

```
{
 "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab"
}
```

PowerShell

```
4105a4fc-f944-11e6-9d32-0123456789ab
```

3. Run the following command to retrieve the status of the automation.

Linux & macOS

```
aws ssm describe-automation-executions \
--filter "Key=ExecutionId,Values=4105a4fc-f944-11e6-9d32-0123456789ab"
```

Windows

```
aws ssm describe-automation-executions ^
--filter "Key=ExecutionId,Values=4105a4fc-f944-11e6-9d32-0123456789ab"
```

PowerShell

```
Get-SMAutomationExecutionList | `
Where {$_.AutomationExecutionId -eq "4105a4fc-f944-11e6-9d32-0123456789ab"}
```

The system returns information like the following.

Linux & macOS

```
{
 "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

```
{
 "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

```
AutomationExecutionId : 4105a4fc-f944-11e6-9d32-0123456789ab
AutomationExecutionStatus : InProgress
```

```

AutomationType : Local
CurrentAction : aws:changeInstanceState
CurrentStepName : startInstances
DocumentName : AWS-RestartEC2Instance
DocumentVersion : 1
ExecutedBy : arn:aws:sts::123456789012:assumed-role/Administrator/
Admin
ExecutionEndTime : 1/1/0001 12:00:00 AM
ExecutionStartTime : 7/31/2019 7:17:28 PM
FailureMessage :
LogFile :
MaxConcurrency :
MaxErrors :
Mode : Auto
Outputs : {}
ParentAutomationExecutionId :
ResolvedTargets : Amazon.SimpleSystemsManagement.Model.ResolvedTargets
Target :
TargetMaps : {}
TargetParameterName :
Targets : {}

```

## Running an automation manually

The following procedures describe how to use the AWS Systems Manager console and AWS Command Line Interface (AWS CLI) to run an automation using the manual execution mode. By using the manual execution mode, the automation starts in a *Waiting* status and pauses in the *Waiting* status between each step. This allows you to control when the automation proceeds, which is useful if you need to review the result of a step before continuing.

The automation 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 use the runbook, and any actions called by the runbook. If you have administrator permissions in IAM, then you already have permission to run this automation.

**Note**

For information about how to run an automation that uses an IAM service role or more advanced forms of delegated administration, see [Running automations by using different security models \(p. 456\)](#).

### Running an automation step by step (console)

The following procedure shows how to use the Systems Manager console to manually run an automation step by step.

#### To run an automation step by step

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the **Automation document** list, choose a runbook. Choose one or more options in the **Document categories** pane to filter SSM documents according to their purpose. To view a runbook that you own, choose the **Owned by me** tab. To view a runbook that is shared with your account, choose the **Shared with me** tab. To view all runbooks, choose the **All documents** tab.

**Note**

You can view information about a runbook by choosing the runbook 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 runbook is updated periodically and a new default version is assigned.
  - **Latest version at runtime:** Choose this option if the Automation runbook 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're ready to start the first step of the automation. The automation proceeds with step one and pauses before running any subsequent steps specified in the runbook you chose in step 3 of this procedure. If the runbook has multiple steps, you must select **Execute this step** for each step for the automation to proceed. Each time you choose **Execute this step** the action runs.

**Note**

The console displays the status of the automation. If the automation fails to run a step, see [Troubleshooting Systems Manager Automation \(p. 691\)](#).

10. After you complete all steps specified in the runbook, choose **Complete and view results** to finish the automation and view the results.

## Running an automation step by step (command line)

The following procedure describes how to use the AWS CLI (on Linux, macOS, or Windows) or AWS Tools for PowerShell to manually run an automation step by step.

### To run an automation step by step

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to start a manual automation. Replace each *example resource placeholder* with your own information.

#### Linux & macOS

```
aws ssm start-automation-execution \
--document-name runbook name \
--mode Interactive \
--parameters runbook parameters
```

#### Windows

```
aws ssm start-automation-execution ^
--document-name runbook name ^
--mode Interactive ^
--parameters runbook parameters
```

#### PowerShell

```
Start-SMAutomationExecution `-
-DocumentName runbook name `-
-Mode Interactive `-
```

-Parameter *runbook parameters*

Here is an example using the runbook AWS-RestartEC2Instance to restart the specified EC2 instance.

Linux & macOS

```
aws ssm start-automation-execution \
--document-name "AWS-RestartEC2Instance" \
--mode Interactive \
--parameters "InstanceId=i-02573cafefEXAMPLE"
```

Windows

```
aws ssm start-automation-execution ^
--document-name "AWS-RestartEC2Instance" ^
--mode Interactive ^
--parameters "InstanceId=i-02573cafefEXAMPLE"
```

PowerShell

```
Start-SMAutomationExecution `-
-DocumentName AWS-RestartEC2Instance `-
-Mode Interactive `-
-Parameter @{"InstanceId"="i-02573cafefEXAMPLE"}
```

The system returns information like the following.

Linux & macOS

```
{ "AutomationExecutionId": "ba9cd881-1b36-4d31-a698-0123456789ab" }
```

Windows

```
{ "AutomationExecutionId": "ba9cd881-1b36-4d31-a698-0123456789ab" }
```

PowerShell

```
ba9cd881-1b36-4d31-a698-0123456789ab
```

3. Run the following command when you're ready to start the first step of the automation. Replace each *example resource placeholder* with your own information. The automation proceeds with step one and pauses before running any subsequent steps specified in the runbook you chose in step 1 of this procedure. If the runbook has multiple steps, you must run the following command for each step for the automation to proceed.

Linux & macOS

```
aws ssm send-automation-signal \
--automation-execution-id ba9cd881-1b36-4d31-a698-0123456789ab \
--signal-type StartStep \
```

```
--payload StepName="stopInstances"
```

### Windows

```
aws ssm send-automation-signal ^
--automation-execution-id ba9cd881-1b36-4d31-a698-0123456789ab ^
--signal-type StartStep ^
--payload StepName="stopInstances"
```

### PowerShell

```
Send-SMMAutomationSignal `^
-AutomationExecutionId ba9cd881-1b36-4d31-a698-0123456789ab `^
-SignalType StartStep `^
-Payload @{"StepName"="stopInstances"}^
```

There is no output if the command succeeds.

- Run the following command to retrieve the status of each step execution in the automation.

### Linux & macOS

```
aws ssm describe-automation-step-executions \
--automation-execution-id ba9cd881-1b36-4d31-a698-0123456789ab
```

### Windows

```
aws ssm describe-automation-step-executions ^
--automation-execution-id ba9cd881-1b36-4d31-a698-0123456789ab
```

### PowerShell

```
Get-SMMAutomationStepExecution `^
-AutomationExecutionId ba9cd881-1b36-4d31-a698-0123456789ab
```

The system returns information like the following.

### Linux & macOS

```
{
 "StepExecutions": [
 {
 "StepName": "stopInstances",
 "Action": "aws:changeInstanceState",
 "ExecutionStartTime": 1557167178.42,
 "ExecutionEndTime": 1557167220.617,
 "StepStatus": "Success",
 "Inputs": {
 "DesiredState": "\"stopped\"",
 "InstanceIds": "[\"i-02573cafefEXAMPLE\"]"
 },
 "Outputs": {
 "InstanceStates": [
 "stopped"
]
 },
 "StepExecutionId": "654243ba-71e3-4771-b04f-0123456789ab",
 "AutomationExecutionId": "ba9cd881-1b36-4d31-a698-0123456789ab"
 }
]
}
```

```

 "OverriddenParameters": {},
 "ValidNextSteps": [
 "startInstances"
],
},
{
 "StepName": "startInstances",
 "Action": "aws:changeInstanceState",
 "ExecutionStartTime": 1557167273.754,
 "ExecutionEndTime": 1557167480.73,
 "StepStatus": "Success",
 "Inputs": {
 "DesiredState": "\"running\"",
 "InstanceIds": "[\"i-02573cafefEXAMPLE\"]"
 },
 "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-02573cafefEXAMPLE\"]"
 },
 "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-02573cafefEXAMPLE\"]"
 },
 "Outputs": {
 "InstanceStates": [
 "running"
]
 }
 }
]
}

```

```
]
 },
 "StepExecutionId": "8a4a1e0d-dc3e-4039-a599-0123456789ab",
 "OverriddenParameters": {}
}
]
```

### 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-02573cafccEXAMPLE"]]}
IsCritical : False
IsEnd : False
MaxAttempts : 0
NextStep :
OnFailure :
Outputs : {[InstanceStates,
Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
OverriddenParameters : {}
Response :
ResponseCode :
StepExecutionId : 8fcc9641-24b7-40b3-a9be-0123456789ab
StepName : stopInstances
StepStatus : Success
TimeoutSeconds : 0
ValidNextSteps : {startInstances}
```

- Run the following command to complete the automation after all steps specified within the chosen runbook have finished. Replace each *example resource placeholder* with your own information.

### Linux & macOS

```
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-SMAutomationExecution `-
-AutomationExecutionId ba9cd881-1b36-4d31-a698-0123456789ab `-
-Type Complete
```

There is no output if the command succeeds.

## Running an automation with approvers

The following procedures describe how to use the AWS Systems Manager console and AWS Command Line Interface (AWS CLI) to run an automation with approvals using simple execution. The automation uses the automation action `aws:approve`, which temporarily pauses the automation until the designated principals either approve or deny the action. The automation 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 use the runbook, and any actions called by the runbook. If you have administrator permissions in IAM, then you already have permission to use this runbook.

### Note

For information about how to run an automation that uses an IAM service role or more advanced forms of delegated administration, see [Running automations by using different security models \(p. 456\)](#).

### Before you begin

In addition to the standard inputs required by the runbook, 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 runbook.
- 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 with approvers (console)

### To run an automation with approvers

The following procedure describes how to use the Systems Manager console to run an automation with approvers.

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the **Automation document** list, choose a runbook. Choose one or more options in the **Document categories** pane to filter SSM documents according to their purpose. To view a runbook that you own, choose the **Owned by me** tab. To view a runbook that is shared with your account, choose the **Shared with me** tab. To view all runbooks, choose the **All documents** tab.

### Note

You can view information about a runbook by choosing the runbook 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 runbook is updated periodically and a new default version is assigned.
  - **Latest version at runtime:** Choose this option if the Automation runbook 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 **Simple execution**.
7. In the **Input parameters** section, specify the required input parameters.

For example, if you chose the **AWS-StartEC2InstanceWithApproval** runbook, 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. 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, the automation continues to run the steps included in the specified runbook. The console displays the status of the automation. If the automation fails to run, see [Troubleshooting Systems Manager Automation \(p. 691\)](#).

#### To approve or deny an automation

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**, and then select the automation 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 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 with approvers.

#### To run an automation with approvers

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.  
  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to run an automation with approvers. Replace each *example resource placeholder* with your own information. In the document name section, specify a runbook 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 Amazon 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 runbook you choose.

Linux & macOS

```
aws ssm start-automation-execution \
--document-name "AWS-StartEC2InstanceWithApproval" \
```

```
--parameters
"InstanceId=i-02573cafefEXAMPLE,Approvers=arn:aws:iam::123456789012:role/
Administrator,SNSTopicArn=arn:aws:sns:region:123456789012:AutomationApproval"
```

### Windows

```
aws ssm start-automation-execution ^
--document-name "AWS-StartEC2InstanceWithApproval" ^
--parameters
"InstanceId=i-02573cafefEXAMPLE,Approvers=arn:aws:iam::123456789012:role/
Administrator,SNSTopicArn=arn:aws:sns:region:123456789012:AutomationApproval"
```

### PowerShell

```
Start-SMAutomationExecution `-
-DocumentName AWS-StartEC2InstanceWithApproval `-
-Parameters @{
 "InstanceId"="i-02573cafefEXAMPLE"
 "Approvers"="arn:aws:iam::123456789012:role/Administrator"
 "SNSTopicArn"="arn:aws:sns:region:123456789012:AutomationApproval"
}
```

The system returns information like the following.

### Linux & macOS

```
{ "AutomationExecutionId": "df325c6d-b1b1-4aa0-8003-6cb7338213c6" }
```

### Windows

```
{ "AutomationExecutionId": "df325c6d-b1b1-4aa0-8003-6cb7338213c6" }
```

### PowerShell

```
df325c6d-b1b1-4aa0-8003-6cb7338213c6
```

## To approve an automation

- Run the following command to approve an automation. Replace each *example resource placeholder* with your own information.

### Linux & macOS

```
aws ssm send-automation-signal \
--automation-execution-id "df325c6d-b1b1-4aa0-8003-6cb7338213c6" \
--signal-type "Approve" \
--payload "Comment=your comments"
```

### Windows

```
aws ssm send-automation-signal ^
```

```
--automation-execution-id "df325c6d-b1b1-4aa0-8003-6cb7338213c6" ^
--signal-type "Approve" ^
--payload "Comment=your comments"
```

### PowerShell

```
Send-SSMAutomationSignal ^
-AutomationExecutionId df325c6d-b1b1-4aa0-8003-6cb7338213c6 ^
-SignalType Approve ^
-Payload @{"Comment"="your comments"}
```

There is no output if the command succeeds.

### To deny an automation

- Run the following command to deny an automation. Replace each *example resource placeholder* with your own information.

#### Linux & macOS

```
aws ssm send-automation-signal \
--automation-execution-id "df325c6d-b1b1-4aa0-8003-6cb7338213c6" \
--signal-type "Deny" \
--payload "Comment=your comments"
```

#### Windows

```
aws ssm send-automation-signal ^
--automation-execution-id "df325c6d-b1b1-4aa0-8003-6cb7338213c6" ^
--signal-type "Deny" ^
--payload "Comment=your comments"
```

### PowerShell

```
Send-SSMAutomationSignal ^
-AutomationExecutionId df325c6d-b1b1-4aa0-8003-6cb7338213c6 ^
-SignalType Deny ^
-Payload @{"Comment"="your comments"}
```

There is no output if the command succeeds.

## Allowing Automation to adapt to your concurrency needs

By default, Automation allows you to run up to 100 concurrent automations at a time. Automation also provides an optional setting that you can use to adjust your concurrency automation quota automatically. With this setting, your concurrency automation quota can accommodate up to 500 concurrent automations, depending on available resources.

### Note

If your automation calls API operations, adaptively scaling to your targets can result in throttling exceptions. If recurring throttling exceptions occur when running automations with adaptive concurrency turned on, you might have to request quota increases for the API operation if available.

### To turn on adaptive concurrency (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**.
3. Choose the **Preferences** tab, and then choose **Edit**.
4. Select the check box next to **Enable adaptive concurrency**.
5. Choose **Save**.

## Running automations that use targets and rate controls

With AWS Systems Manager Automation, you can run automations on a fleet of AWS resources by using targets. Additionally, you can control the deployment 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. Automation also provides an adaptive concurrency mode you can opt in to. Adaptive concurrency automatically scales your automation quota from 100 concurrently running automations up to 500. An error threshold determines how many automations are allowed to fail before Systems Manager stops sending the automation 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 \(p. 435\)](#). For more information about targets, see [About targets \(p. 429\)](#).

The following procedures show you how to turn on adaptive concurrency, and how to run an automation with targets and rate controls by using the Systems Manager console and AWS Command Line Interface (AWS CLI).

### Running an automation with targets and rate controls (console)

The following procedure describes how to use the Systems Manager console to run an automation with targets and rate controls.

#### To run an automation with targets and rate controls

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the **Automation document** list, choose a runbook. Choose one or more options in the **Document categories** pane to filter SSM documents according to their purpose. To view a runbook that you own, choose the **Owned by me** tab. To view a runbook that is shared with your account, choose the **Shared with me** tab. To view all runbooks, choose the **All documents** tab.

#### Note

You can view information about a runbook by choosing the runbook 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 runbook is updated periodically and a new default version is assigned.
  - **Latest version at runtime:** Choose this option if the Automation runbook 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 runbook 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 runbook on all instances in the current AWS account and AWS 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 might 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 runbook, 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**.

To view automations started by your rate control automation, in the navigation pane, choose Automation, and then select **Show child automations**.

## Running an automation 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 with targets and rate controls.

### To run an automation with targets and rate controls

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to view a list of documents.

Linux & macOS

```
aws ssm list-documents
```

Windows

```
aws ssm list-documents
```

PowerShell

```
Get-SSMDocumentList
```

Note the name of the runbook that you want to use.

3. Run the following command to view details about the runbook. Replace the *runbook name* with the name of the runbook whose details you want to view. Also, 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 & macOS

```
aws ssm describe-document \
--name runbook name
```

Windows

```
aws ssm describe-document ^
--name runbook name
```

PowerShell

```
Get-SSMDocumentDescription ^
-Name runbook name
```

4. Create a command that uses the targets and rate control options you want to run. Replace each *example resource placeholder* with your own information..

*Targeting using tags*

Linux & macOS

```
aws ssm start-automation-execution \
--document-name runbook name \
--targets Key=tag:key name,Values=value \
--target-parameter-name parameter name \
--parameters "input parameter name=input parameter value,input parameter 2 \
name=input parameter 2 value" \
--max-concurrency 10 \
--max-errors 25%
```

Windows

```
aws ssm start-automation-execution ^
--document-name runbook name ^
--targets Key=tag:key name,Values=value ^
```

```
--target-parameter-name parameter name ^
--parameters "input parameter name=input parameter value,input parameter 2
name=input parameter 2 value" ^
--max-concurrency 10 ^
--max-errors 25%
```

### PowerShell

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "tag:key name"
$Targets.Values = "value"

Start-SMAutomationExecution `-
 DocumentName "runbook name" `-
 -Targets $Targets `-
 -TargetParameterName "parameter name" `-
 -Parameter @{"input parameter name"="input parameter value";"input parameter 2
name"="input parameter 2 value"} `-
 -MaxConcurrency "10" `-
 -MaxError "25%"
```

### Targeting using parameter values

#### Linux & macOS

```
aws ssm start-automation-execution \
--document-name runbook name \
--targets Key=ParameterValues,Values=value,value 2,value 3 \
--target-parameter-name parameter name \
--parameters "input parameter name=input parameter value" \
--max-concurrency 10 \
--max-errors 25%
```

#### Windows

```
aws ssm start-automation-execution ^
--document-name runbook name ^
--targets Key=ParameterValues,Values=value,value 2,value 3 ^
--target-parameter-name parameter name ^
--parameters "input parameter name=input parameter value" ^
--max-concurrency 10 ^
--max-errors 25%
```

### PowerShell

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ParameterValues"
$Targets.Values = "value","value 2","value 3"

Start-SMAutomationExecution `-
 -DocumentName "runbook name" `-
 -Targets $Targets `-
 -TargetParameterName "parameter name" `-
 -Parameter @{"input parameter name"="input parameter value"} `-
 -MaxConcurrency "10" `-
 -MaxError "25%"
```

### Targeting using AWS Resource Groups

### Linux & macOS

```
aws ssm start-automation-execution \
--document-name runbook name \
--targets Key=ResourceGroup,Values=Resource group name \
--target-parameter-name parameter name \
--parameters "input parameter name=input parameter value" \
--max-concurrency 10 \
--max-errors 25%
```

### Windows

```
aws ssm start-automation-execution ^
--document-name runbook name ^
--targets Key=ResourceGroup,Values=Resource group name ^
--target-parameter-name parameter name ^
--parameters "input parameter name=input parameter value" ^
--max-concurrency 10 ^
--max-errors 25%
```

### PowerShell

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ResourceGroup"
$Targets.Values = "Resource group name"

Start-SMAutomationExecution `-
-DocumentName "runbook name" `-
-Targets $Targets `-
-TargetParameterName "parameter name" `-
-Parameter @{"input parameter name"="input parameter value"} `-
-MaxConcurrency "10" `-
-MaxError "25%"
```

*Targeting all instances in the current AWS account and AWS Region*

### Linux & macOS

```
aws ssm start-automation-execution \
--document-name runbook name \
--targets "Key=AWS::EC2::Instance,Values=*" \
--target-parameter-name instanceId \
--parameters "input parameter name=input parameter value" \
--max-concurrency 10 \
--max-errors 25%
```

### Windows

```
aws ssm start-automation-execution ^
--document-name runbook name ^
--targets Key=AWS::EC2::Instance,Values=* ^
--target-parameter-name instanceId ^
--parameters "input parameter name=input parameter value" ^
--max-concurrency 10 ^
--max-errors 25%
```

## PowerShell

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "AWS::EC2::Instance"
$Targets.Values = "*"

Start-SMAutomationExecution ^
 -DocumentName "runbook name" ^
 -Targets $Targets ^
 -TargetParameterName "instanceId" ^
 -Parameter @{"input parameter name"="input parameter value"} ^
 -MaxConcurrency "10" ^
 -MaxError "25%"
```

The command returns an execution ID. Copy this ID to the clipboard. You can use this ID to view the status of the automation.

## Linux & macOS

```
{
 "AutomationExecutionId": "a4a3c0e9-7efd-462a-8594-01234EXAMPLE"
}
```

## Windows

```
{
 "AutomationExecutionId": "a4a3c0e9-7efd-462a-8594-01234EXAMPLE"
}
```

## PowerShell

```
a4a3c0e9-7efd-462a-8594-01234EXAMPLE
```

5. Run the following command to view the automation. Replace each *automation execution ID* with your own information.

## Linux & macOS

```
aws ssm describe-automation-executions \
 --filter Key=ExecutionId,Values=automation execution ID
```

## Windows

```
aws ssm describe-automation-executions ^
 --filter Key=ExecutionId,Values=automation execution ID
```

## PowerShell

```
Get-SMAutomationExecutionList | ^
 Where {$_.AutomationExecutionId -eq "automation execution ID"}
```

6. To view details about the automation progress, run the following command. Replace each *automation execution ID* with your own information.

Linux & macOS

```
aws ssm get-automation-execution \
--automation-execution-id automation execution ID
```

Windows

```
aws ssm get-automation-execution ^
--automation-execution-id automation execution ID
```

PowerShell

```
Get-SMAutomationExecution ^
-AutomationExecutionId automation execution ID
```

The system returns information like the following.

Linux & macOS

```
{
 "AutomationExecution": {
 "StepExecutionsTruncated": false,
 "AutomationExecutionStatus": "Success",
 "MaxConcurrency": "1",
 "Parameters": {},
 "MaxErrors": "1",
 "Outputs": {},
 "DocumentName": "AWS-StopEC2Instance",
 "AutomationExecutionId": "a4a3c0e9-7efd-462a-8594-01234EXAMPLE",
 "ResolvedTargets": {
 "ParameterValues": [
 "i-02573cafefEXAMPLE"
],
 "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-02573cafefEXAMPLE"
 },
 "Outputs": {},
 "StepName": "i-02573cafefEXAMPLE",
 "ExecutionEndTime": 1564681619.093,
 "StepExecutionId": "86c7b811-3896-4b78-b897-01234EXAMPLE",
 "ExecutionStartTime": 1564681576.836,
 "Action": "aws:executeAutomation",
 "StepStatus": "Success"
 }
]
 }
}
```

```
 }
],
 "TargetParameterName": "InstanceId",
 "Mode": "Auto"
}
}
```

### Windows

```
{
 "AutomationExecution": {
 "StepExecutionsTruncated": false,
 "AutomationExecutionStatus": "Success",
 "MaxConcurrency": "1",
 "Parameters": {},
 "MaxErrors": "1",
 "Outputs": {},
 "DocumentName": "AWS-StopEC2Instance",
 "AutomationExecutionId": "a4a3c0e9-7efd-462a-8594-01234EXAMPLE",
 "ResolvedTargets": {
 "ParameterValues": [
 "i-02573cafccEXAMPLE"
],
 "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-02573cafccEXAMPLE"
 },
 "Outputs": {},
 "StepName": "i-02573cafccEXAMPLE",
 "ExecutionEndTime": 1564681619.093,
 "StepExecutionId": "86c7b811-3896-4b78-b897-01234EXAMPLE",
 "ExecutionStartTime": 1564681576.836,
 "Action": "aws:executeAutomation",
 "StepStatus": "Success"
 }
],
 "TargetParameterName": "InstanceId",
 "Mode": "Auto"
 }
}
```

### PowerShell

```
AutomationExecutionId : a4a3c0e9-7efd-462a-8594-01234EXAMPLE
AutomationExecutionStatus : Success
CurrentAction :
CurrentStepName :
DocumentName : AWS-StopEC2Instance
```

```

DocumentVersion : 1
ExecutedBy : arn:aws:sts::123456789012:assumed-role/Administrator/
Admin :
ExecutionEndTime : 8/1/2019 5:46:59 PM
ExecutionStartTime : 8/1/2019 5:46:16 PM
FailureMessage :
MaxConcurrency : 1
MaxErrors : 1
Mode : Auto
Outputs : {}
Parameters : {}
ParentAutomationExecutionId :
ProgressCounters :
ResolvedTargets : Amazon.SimpleSystemsManagement.Model.ResolvedTargets
StepExecutions : {i-02573cafcfEXAMPLE}
StepExecutionsTruncated : False
Target :
TargetLocations : {}
TargetMaps : {}
TargetParameterName : InstanceId
Targets : {tag:Name}

```

#### **Note**

You can also monitor the status of the automation in the console. In the **Automation executions** list, choose the automation you just ran and then choose the **Execution steps** tab. This tab shows the status of the automation actions.

## About targets

Use the **Targets** parameter to quickly define which resources in your fleet can run an automation. 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 Amazon Elastic Compute Cloud (Amazon EC2) tags with the **Targets** parameter.

When you run an automation that uses a target, AWS 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 automations. 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 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 a tag

You can specify a single tag as the target of an automation. Many AWS resources support tags, including Amazon Elastic Compute Cloud (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 on your AWS resources by targeting a tag. 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 runbook. 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. Further, when creating a custom runbook you can specify the **Target type** as `/AWS::EC2::Instance` to ensure only instances are used. When targeting instances with a tag, terminated instances might be included.

The following screenshot uses the `AWS-DetachEBSVolume` runbook. The logical target parameter is `VolumeId`.

The screenshot shows the 'Targets and Rate Control' section of the AWS Systems Manager configuration interface. It includes fields for enabling targets selection and rate control, specifying targets (Tags), parameters (VolumeId), and tags (Finance:TestEnv). A note at the bottom indicates that the tag key and optional values should be separated by a colon and space.

Targets and Rate Control

Enable the targets selection and rate control. Available for Execution mode "Auto" only.

Enable targets and rate control

Targets

Tags

Parameter

VolumeId

Tags

Enter a custom tag key and value.

Finance TestEnv

Type the tag key and optional values shared by the targets, and then click Add.

The `AWS-DetachEBSVolume` runbook 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 Command Line Interface (AWS CLI) or AWS Tools for Windows PowerShell, then you must enter the target parameter name exactly as it's defined in the runbook. If you don't, the system returns an `InvalidAutomationExecutionParametersException` error. You can use the `DescribeDocument` API operation to see information about the available target parameters in a specific runbook. Following 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 a tag.

#### Example 1: Targeting a tag using a key-value pair to restart Amazon EC2 instances

This example restarts all Amazon EC2 instances that are tagged with a key of `Department` and a value of `HumanResources`. The target parameter uses the `InstanceId` parameter from the runbook. 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 a tag using a key-value pair to delete Amazon EBS snapshots

The following example uses the AWS-DeleteSnapshot runbook to delete all snapshots with a key of *Name* and a value of *January2018Backups*. The target parameter uses the *VolumeId* 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 runbook 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 runbook defaults to the t2.medium instance type, so this command would create 25 t2.medium Amazon 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.

## Targets and Rate Control

Enable the targets selection and rate control. Available for Execution mode "Auto" only.

Enable targets and rate control

Targets

Resource Group

Parameter

Amild

Resource group

t2-micro-instances

## 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 to run. If you specify multiple values, Systems Manager runs a child automation on each value specified.

For example, say that your runbook 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 automation fails. You can target a maximum of 50 parameter values.

The following example uses the `AWS-CreateImage` runbook. The target parameter name specified is `InstanceId`. The key uses `ParameterValues`. The values are two Amazon EC2 instance IDs. This command creates an automation for each instance, which produces an AMI from each instance.

```
aws ssm start-automation-execution
--document-name AWS-CreateImage \
--target-parameter-name InstanceId \
--targets Key=ParameterValues,Values=i-02573cafefEXAMPLE,i-0471e04240EXAMPLE
```

### Note

`AutomationAssumeRole` isn't a valid parameter. Don't choose this item when running automation 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

The `TargetMaps` option isn't supported in the console.

Use the following format to specify multiple parameter values by using the `TargetMaps` option in a command. Replace each `example resource placeholder` with your own information.

```
aws ssm start-automation-execution \
--document-name runbook name \
--target-maps "parameter=value, parameter 2=value, parameter 3=value" "parameter
4=value, parameter 5=value, parameter 6=value"
```

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.

```
[
 {"parameter": "value", "parameter 2": "value", "parameter 3": "value"},
 {"parameter 4": "value", "parameter 5": "value", "parameter 6": "value"}
]
```

Save the file with a .json file extension. You can call the file by using the following command. Replace each *example resource placeholder* with your own information.

```
aws ssm start-automation-execution \
--document-name runbook name \
--parameters input parameters \
--target-maps path to file/file name.json
```

You can also download the file from an Amazon Simple Storage Service (Amazon S3) bucket, as long as you have permission to read data from the bucket. Use the following command format. Replace each *example resource placeholder* with your own information.

```
aws ssm start-automation-execution \
--document-name runbook name \
--target-maps http://DOC-EXAMPLE-BUCKET.s3.amazonaws.com/file_name.json
```

Here is an example scenario to help you understand the TargetMaps option. In this scenario, a user wants to create Amazon EC2 instances of different types from different AMIs. To perform this task, the user creates a runbook named AMI\_Testing. This runbook defines two input parameters: instanceType and imageId.

```
{
 "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`.

```

[
 {
 "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 AWS 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 AWS Region, you can choose the **AWS-RestartEC2Instance** runbook 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 **Instanceld** 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 deployment of an automation 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

Use Concurrency 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

Use an error threshold to specify how many automations are allowed to fail before AWS 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 running the automation when the fourth error is received. If you specify 0, then the system stops running the automation on additional targets 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 automations based on triggers

This section includes information about how to run automations using a trigger. Automations can be initiated by several different triggers, such as Amazon EventBridge, State Manager associations, or

maintenance windows. State Manager is a capability of AWS Systems Manager. By using triggers, you can run automations as a result of a specific event or on a scheduled basis.

## Contents

- [Running automations with triggers using EventBridge \(p. 436\)](#)
- [Running automations with triggers using State Manager \(p. 441\)](#)
- [Running automations with triggers using a maintenance window \(p. 449\)](#)

## Running automations with triggers using EventBridge

You can start an automation by specifying a runbook as the target of an Amazon EventBridge event. You can start automations according to a schedule, or when a specific AWS system event occurs. For example, let's say you create a runbook named *BootStrapInstances* that installs software on an instance when an instance starts. To specify the *BootStrapInstances* runbook (and corresponding automation) as a target of an EventBridge event, you first create a new EventBridge 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* runbook as the target of the event using the EventBridge console and AWS Command Line Interface (AWS CLI). When a new instance starts, the system runs the automation and installs software.

For information about creating runbooks, see [Working with runbooks \(p. 539\)](#).

### Creating an EventBridge event that uses a runbook (console)

Use the following procedure to configure a runbook as the target of a EventBridge event.

#### To configure a runbook as a target of a EventBridge event rule

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**, and then choose **Create rule**.

-or-

If the Amazon EventBridge home page opens first, choose **Create rule**.

3. Enter a name and description for the rule.

A rule can't have the same name as another rule in the same Region and on the same event bus.

4. For **Define pattern**, choose either **Event pattern** or **Schedule**. Use **Event pattern** to build a rule that generates events for specific actions in AWS services. Use **Schedule** to build a rule that generates events according to a schedule that you specify by using the cron format.
5. Choose the remaining options for the rule you want to create, and then choose **Add target**.

For information about creating EventBridge rules, see [Getting Started with Amazon EventBridge](#) in the [Amazon EventBridge User Guide](#).

6. For **Select event bus**, choose the event bus that you want to associate with this rule. If you want this rule to initiate on matching events that come from your own AWS account, select **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
7. For **Target**, choose **Systems Manager Automation**.
8. For **Document**, choose a runbook to use when your target is invoked.
9. Expand **Configure document version**, and choose a version. \$DEFAULT was explicitly set as the default runbook version in Systems Manager. You can choose a specific version, or use the latest version.
10. 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.

11. At the bottom of the **Select targets** area, choose a role to grant EventBridge permission to start the automation with the specified runbook and parameters. EventBridge uses the role to start the automation. You can let EventBridge create a new role or use a role that already has the needed permissions.
12. (Optional) Enter one or more tags for the rule. For more information, see [Tagging Your Amazon EventBridge Resources](#) in the *Amazon EventBridge User Guide*.
13. Choose **Create** and complete the wizard.

### Create an EventBridge event that uses a runbook (command line)

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to create an EventBridge event rule and configure a runbook as the target.

#### To configure a runbook as a target of an EventBridge event rule

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Create a command to specify a new EventBridge event rule. Replace each *example resource placeholder* with your own information.

*Triggers based on a schedule*

Linux & macOS

```
aws events put-rule \
--name "rule name" \
--schedule-expression "cron or rate expression"
```

Windows

```
aws events put-rule ^
--name "rule name" ^
--schedule-expression "cron or rate expression"
```

PowerShell

```
Write-CWERule `
-Name "rule name" `
-ScheduleExpression "cron or rate expression"
```

The following example creates an EventBridge event rule that starts every day at 9:00 AM (UTC).

Linux & macOS

```
aws events put-rule \
--name "DailyAutomationRule" \
--schedule-expression "cron(0 9 * * ? *)"
```

## Windows

```
aws events put-rule ^
--name "DailyAutomationRule" ^
--schedule-expression "cron(0 9 * * ? *)"
```

## PowerShell

```
Write-CWERule `^
-Name "DailyAutomationRule" `^
-ScheduleExpression "cron(0 9 * * ? *)"
```

*Triggers based on an event*

## Linux & macOS

```
aws events put-rule \
--name "rule name" \
--event-pattern "{\"source\":[\"aws.service\"],\"detail-type\":[\"service event detail type\"]}"
```

## Windows

```
aws events put-rule ^
--name "rule name" ^
--event-pattern "{\"source\":[\"aws.service\"],\"detail-type\":[\"service event detail type\"]}"
```

## PowerShell

```
Write-CWERule `^
-Name "rule name" `^
-EventPattern '{"source":["aws.service"],"detail-type":["service event detail type"]}'
```

The following example creates an EventBridge event rule that starts when any EC2 instance in the Region changes state.

## Linux & macOS

```
aws events put-rule \
--name "EC2InstanceStateChanges" \
--event-pattern "{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change Notification\"]}"
```

## Windows

```
aws events put-rule ^
--name "EC2InstanceStateChanges" ^
--event-pattern "{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change Notification\"]}"
```

## PowerShell

```
Write-CWERule `
 -Name "EC2InstanceStateChanges" `
 -EventPattern '{"source":["aws.ec2"],"detail-type":["EC2 Instance State-change Notification"]}'
```

The command returns details for the new EventBridge rule similar to the following.

## Linux & macOS

```
{
 "RuleArn": "arn:aws:events:us-east-1:123456789012:rule/automationrule"
}
```

## Windows

```
{
 "RuleArn": "arn:aws:events:us-east-1:123456789012:rule/automationrule"
}
```

## PowerShell

```
arn:aws:events:us-east-1:123456789012:rule/EC2InstanceStateChanges
```

3. Create a command to specify a runbook as a target of the EventBridge event rule you created in step 2. Replace each *example resource placeholder* with your own information.

## Linux & macOS

```
aws events put-targets \
 --rule rule name \
 --targets '{"Arn": "arn:aws:ssm:us-east-1:123456789012:automation-
definition/runbook name","Input": "{\"input parameter\": ["value"]},
 \"AutomationAssumeRole\": [
 \"arn:aws:iam::123456789012:role/AutomationServiceRole\"]}" , "Id": "target
 ID", "RoleArn": "arn:aws:iam::123456789012:role/service-role/EventBridge service
 role"}
```

## Windows

```
aws events put-targets ^
 --rule rule name ^
 --targets '{"Arn": "arn:aws:ssm:us-east-1:123456789012:automation-
definition/runbook name","Input": "{\"input parameter\": ["value"]},
 \"AutomationAssumeRole\": [
 \"arn:aws:iam::123456789012:role/AutomationServiceRole\"]}" , "Id": "target
 ID", "RoleArn": "arn:aws:iam::123456789012:role/service-role/EventBridge service
 role"}
```

## PowerShell

```
$Target = New-Object Amazon.CloudWatchEvents.Model.Target
$Target.Id = "target ID"
$Target.Arn = "arn:aws:ssm:us-east-1:123456789012:automation-definition/runbook
name"
```

```
$Target.RoleArn = "arn:aws:iam::123456789012:role/service-role/EventBridge service role"
$Target.Input = '{"input parameter": ["value"], "AutomationAssumeRole": ["arn:aws:iam::123456789012:role/AutomationServiceRole"]}'

Write-CWETarget `
 -Rule "rule name" `
 -Target $Target
```

The following example creates an EventBridge event target that starts the specified instance ID using the runbook AWS-StartEC2Instance.

#### Linux & macOS

```
aws events put-targets \
 --rule DailyAutomationRule \
 --targets '{"Arn": "arn:aws:ssm:us-east-1:123456789012:automation-definition/AWS-StartEC2Instance", "Input": "{\"InstanceId\": [\"i-02573cafefEXAMPLE\"], \"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

```
aws events put-targets ^
 --rule DailyAutomationRule ^
 --targets '{"Arn": "arn:aws:ssm:us-east-1:123456789012:automation-definition/AWS-StartEC2Instance", "Input": "{\"InstanceId\": [\"i-02573cafefEXAMPLE\"], \"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

```
$Target = New-Object Amazon.CloudWatchEvents.Model.Target
$Target.Id = "Target1"
$Target.Arn = "arn:aws:ssm:us-east-1:123456789012:automation-definition/AWS-StartEC2Instance"
$Target.RoleArn = "arn:aws:iam::123456789012:role/service-role/AWS_Events_Invoke_Start_Automation_Execution_1213609520"
$Target.Input = '{"InstanceId": ["i-02573cafefEXAMPLE"], "AutomationAssumeRole": ["arn:aws:iam::123456789012:role/AutomationServiceRole"]}'

Write-CWETarget `
 -Rule "DailyAutomationRule" `
 -Target $Target
```

The system returns information like the following.

#### Linux & macOS

```
{
 "FailedEntries": [],
 "FailedEntryCount": 0
}
```

#### Windows

```
{
 "FailedEntries": [],
 "FailedEntryCount": 0
}
```

#### PowerShell

There is no output if the command succeeds for PowerShell.

## Running automations with triggers using State Manager

You can start an automation by creating a State Manager association with a runbook. State Manager is a capability of AWS Systems Manager. By creating a State Manager association with a runbook, 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 Amazon Elastic Compute Cloud (Amazon EC2) instances to make them *managed instances*.
- Enforce desired ingress and egress rules for a security group.
- Create or delete Amazon DynamoDB backups.
- Create or delete Amazon Elastic Block Store (Amazon EBS) snapshots.
- Turn off 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.
- Apply patches to Linux, macOS, and Windows AMIs.

Use the following procedures to create a State Manager association that runs an automation using the AWS Systems Manager console and AWS Command Line Interface (AWS CLI).

### Before you begin

Be aware of the following important details before you run an automation by using State Manager:

- Before you can create an association that uses a runbook, verify that you configured permissions for Automation, a capability of AWS Systems Manager. For more information, see [Setting up Automation \(p. 401\)](#).
- State Manager associations that use runbooks contribute to the maximum number of concurrently running automations in your AWS account. You can have a maximum of 100 concurrent automations running. 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 operations. 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.

#### Linux & macOS

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

#### Windows

```
aws iam create-service-linked-role ^
```

```
--aws-service-name ssm.amazonaws.com
```

## PowerShell

```
New-IAMServiceLinkedRole `
 -AWSPropertyName ssm.amazonaws.com
```

For more information about service-linked roles, see [Using service-linked roles for Systems Manager \(p. 1410\)](#).

### Creating an association that runs an automation (console)

The following procedure describes how to use the Systems Manager console to create a State Manager association that runs an automation.

#### To create a State Manager association that runs an automation

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 runbook. Use the Search bar to filter on **Document type : Equal : Automation** runbooks. To view more runbooks, use the numbers to the right of the Search bar.

##### Note

You can view information about a runbook by choosing the runbook 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 operation of the automation across your resources by specifying concurrency and error thresholds.

If you chose **Rate control**, the **Targets** section is displayed.

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 runbook that you selected at the start of this procedure. By choosing a parameter, you define the type of resource on which the automation 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 runbooks, see [Targeting AWS Resource Groups \(p. 431\)](#).
    - **Tags:** Enter the tag key and (optionally) the tag value in the fields provided. Choose **Add**. For more information about targeting tags in runbooks, see [Targeting a tag \(p. 429\)](#).
    - **Parameter Values:** Enter values in the **Input parameters** section. If you specify multiple values, Systems Manager runs a child automation on each value specified.

For example, say that your runbook 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 automation fails. You can target a maximum of 50 parameter values. For more information about targeting parameter values in runbooks, see [Targeting parameter values \(p. 432\)](#).

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 might not need to choose some of the options in the **Input parameters** section. For example, if you chose the `AWS-RestartEC2Instance` runbook, 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 locates the instances to restart by using the tags 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 runbook and run Automation associations on your behalf. For more information, see [Running an automation by using an IAM service role \(p. 460\)](#).

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 \(p. 1544\)](#).

**Note**

Rate expressions are the preferred scheduling mechanism for State Manager associations that use runbooks. 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 deployment 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 simultaneously.
    - Choose **percentage** to enter a percentage of the target set that can run the automation 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 automation to other resources.
    - Choose **percentage** to enter a percentage of errors allowed before Automation stops sending the automation to other resources.

For more information about using targets and rate controls with Automation, see [Running automations that use targets and rate controls \(p. 421\)](#).

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 doesn't run again.

### [Creating an association that runs an automation \(command line\)](#)

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.

#### **Before you begin**

Before you complete the following procedure, make sure you have created an IAM service role that contains the permissions necessary to run the runbook, and configured a trust relationship for Automation, a capability of AWS Systems Manager. For more information, see [Task 1: Create a service role for Automation \(p. 403\)](#).

### To create an association that runs an automation

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to view a list of documents.

Linux & macOS

```
aws ssm list-documents
```

Windows

```
aws ssm list-documents
```

PowerShell

```
Get-SSMDocumentList
```

Note the name of the runbook that you want to use for the association.

3. Run the following command to view details about the runbook. In the following command, replace *runbook name* with your own information.

Linux & macOS

```
aws ssm describe-document \
--name runbook 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

```
aws ssm describe-document ^
--name runbook 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

```
Get-SSMDocumentDescription ^
-Name runbook 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 using a State Manager association. Replace each *example resource placeholder* with your own information.

#### *Targeting using tags*

##### Linux & macOS

```
aws ssm create-association \
--association-name association name \
--targets Key=tag:key name,Values=value \
--name runbook name \
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/RunbookAssumeRole \
--automation-target-parameter-name target parameter \
--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

```
aws ssm create-association ^
--association-name association name ^
--targets Key=tag:key name,Values=value ^
--name runbook name ^
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/RunbookAssumeRole ^
--automation-target-parameter-name target parameter ^
--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

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "tag:key name"
$Targets.Values = "value"

New-SSMAssociation `
-AssociationName "association name" `
-Target $Targets `
-Name "runbook name" `
-Parameters @`
"AutomationAssumeRole="arn:aws:iam::123456789012:role/RunbookAssumeRole" } `
-AutomationTargetParameterName "target parameter" `
-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 & macOS

```
aws ssm create-association \
--association-name association name \
--targets Key=ParameterValues,Values=value,value 2,value 3 \
--name runbook name \
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/
RunbookAssumeRole \
--automation-target-parameter-name target parameter \
--schedule "cron or rate expression"
```

Windows

```
aws ssm create-association ^
--association-name association name ^
--targets Key=ParameterValues,Values=value,value 2,value 3 ^
--name runbook name ^
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/
RunbookAssumeRole ^
--automation-target-parameter-name target parameter ^
--schedule "cron or rate expression"
```

PowerShell

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ParameterValues"
$Targets.Values = "value","value 2","value 3"

New-SSMAssociation `
-AssociationName "association name" `
-Target $Targets `
-Name "runbook name" `
-Parameters @{
 "AutomationAssumeRole"="arn:aws:iam::123456789012:role/RunbookAssumeRole" `
-AutomationTargetParameterName "target parameter" `
-ScheduleExpression "cron or rate expression"`
```

*Targeting using AWS Resource Groups*

Linux & macOS

```
aws ssm create-association \
--association-name association name \
--targets Key=ResourceGroup,Values=resource group name \
--name runbook name \
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/
RunbookAssumeRole \
--automation-target-parameter-name target parameter \
--schedule "cron or rate expression"
```

Windows

```
aws ssm create-association ^
--association-name association name ^
--targets Key=ResourceGroup,Values=resource group name ^
--name runbook name ^
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/
RunbookAssumeRole ^
```

```
--automation-target-parameter-name target parameter ^
--schedule "cron or rate expression"
```

### PowerShell

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ResourceGroup"
$Targets.Values = "resource group name"

New-SMMAssociation ^
 -AssociationName "association name" ^
 -Target $Targets ^
 -Name "runbook name" ^
 -Parameters @{
 "AutomationAssumeRole"="arn:aws:iam::123456789012:role/RunbookAssumeRole" ^
 -AutomationTargetParameterName "target parameter" ^
 -ScheduleExpression "cron or rate expression"
```

### *Targeting multiple accounts and Regions*

#### Linux & macOS

```
aws ssm create-association \
 --association-name association name \
 --targets Key=ResourceGroup,Values=resource group name \
 --name runbook name \
 --parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/RunbookAssumeRole \
 --automation-target-parameter-name target parameter \
 --schedule "cron or rate expression" \
 --target-locations
 Accounts=111122223333,444455556666,444455556666,Regions=region,region
```

#### Windows

```
aws ssm create-association ^
 --association-name association name ^
 --targets Key=ResourceGroup,Values=resource group name ^
 --name runbook name ^
 --parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/RunbookAssumeRole ^
 --automation-target-parameter-name target parameter ^
 --schedule "cron or rate expression" ^
 --target-locations
 Accounts=111122223333,444455556666,444455556666,Regions=region,region
```

### PowerShell

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ResourceGroup"
$Targets.Values = "resource group name"

New-SMMAssociation ^
 -AssociationName "association name" ^
 -Target $Targets ^
 -Name "runbook name" ^
 -Parameters @{
 "AutomationAssumeRole"="arn:aws:iam::123456789012:role/RunbookAssumeRole" ^
 -AutomationTargetParameterName "target parameter" ^
 -ScheduleExpression "cron or rate expression"
```

```
-TargetLocations @{
 "Accounts"=["111122223333,444455556666,444455556666"],
 "Regions"=["region,region"]
```

The command returns details for the new association similar to the following.

Linux & macOS

```
{
 "AssociationDescription": {
 "ScheduleExpression": "cron(0 7 ? * MON *)",
 "Name": "AWS-StartEC2Instance",
 "Parameters": {
 "AutomationAssumeRole": [
 "arn:aws:iam::123456789012:role/RunbookAssumeRole"
]
 },
 "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

```
{
 "AssociationDescription": {
 "ScheduleExpression": "cron(0 7 ? * MON *)",
 "Name": "AWS-StartEC2Instance",
 "Parameters": {
 "AutomationAssumeRole": [
 "arn:aws:iam::123456789012:role/RunbookAssumeRole"
]
 },
 "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

```
Name : AWS-StartEC2Instance
InstanceId :
Date : 8/1/2019 7:31:38 PM
Status.Name :
Status.Date :
Status.Message:
Status.AdditionalInfo :
```

### 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 automations

Systems Manager Automation enforces a limit of 100 concurrent automations, and 1,000 queued automations per account, per Region. If a State Manager association that uses a runbook shows a status of **Failed** and a detailed status of **AutomationExecutionLimitExceeded**, then your automation might have reached the limit. As a result, Systems Manager throttles the automations. 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 automations that have a status of **Pending**. By deleting these automations, you clear the current queue.

### Running automations with triggers using a maintenance window

You can start an automation by configuring a runbook as a registered task for a maintenance window. By registering the runbook as a registered task, the maintenance window runs the automation during the scheduled maintenance period.

For example, let's say you create a runbook named `CreateAMI` that creates an Amazon Machine Image (AMI) of instances registered as targets to the maintenance window. To specify the `CreateAMI` runbook (and corresponding automation) 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 and creates an AMI of the registered targets.

For information about creating Automation runbooks, see [Working with runbooks \(p. 539\)](#). Automation is a capability of AWS Systems Manager.

Use the following procedures to configure an automation 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 task to a maintenance window (console)

The following procedure describes how to use the Systems Manager console to configure an automation 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 \(console\) \(p. 724\)](#).
- [Assign targets to a maintenance window \(console\) \(p. 726\)](#)

### To configure an automation as a registered task for a maintenance window

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 on targets by using a runbook.
4. For **Name**, enter a name for the task.
5. For **Description**, enter a description.
6. For **Document**, choose the runbook that defines the tasks to run.
7. For **Document version**, choose the runbook 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, if the runbook you chose is one that runs tasks on resources, identify the targets on which you want to run this automation by specifying tags or by selecting instances manually.

#### Note

If you want to pass the resources through input parameters instead of targets, you don't need to specify a maintenance window target.

In many cases, you don't need to explicitly specify a target for an automation task. For example, say that you're creating an Automation-type task to update an Amazon Machine Image (AMI) for Linux using the `AWS-UpdateLinuxAmi` runbook. When the task runs, the AMI is updated with the latest available Linux distribution packages and Amazon software. New instances created from the AMI already have these updates installed. Because the ID of the AMI to be updated is specified in the input parameters for the runbook, there is no need to specify a target again in the maintenance window task.

For information about maintenance window tasks that don't require targets, see [the section called "Registering maintenance window tasks without targets" \(p. 794\)](#).

10. (Optional) For **Rate control**:

#### Note

If the task you're running doesn't specify targets, you don't need to specify rate controls.

- For **Concurrency**, specify either a number or a percentage of targets on which to run the automation at the same time.

If you selected targets by choosing tag key-value pairs, and you aren't certain how many targets use the selected tags, then limit the number of automations that can run at the same time by specifying a percentage.

When the maintenance window runs, a new automation is initiated per target. There is a limit of 100 concurrent automations per AWS account. If you specify a concurrency rate greater than 100, concurrent automations greater than 100 are automatically added to the automation 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 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 automations when the fourth error is received. Targets still processing the automation 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:

- **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 \(console\) \(p. 709\)](#)
- [Control access to maintenance windows \(AWS CLI\) \(p. 714\)](#)
- [Control access to maintenance windows \(Tools for Windows PowerShell\) \(p. 720\)](#)

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? \(p. 708\)](#).

12. In the **Input Parameters** section, specify parameters for the runbook. For runbooks, the system auto-populates some of the values. You can keep or replace these values.

**Important**

For runbooks, you can optionally specify an Automation Assume Role. If you don't specify a role for this parameter, then the automation 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 runbook.

For example, the service-linked role for Systems Manager doesn't have the IAM permission `ec2:CreateSnapshot`, which is required to use the runbook `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 [Setting up Automation \(p. 401\)](#).

13. Choose **Register Automation task**.

### Registering an Automation 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 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 \(AWS CLI\) \(p. 734\)](#).
- [Step 2: Register a target node with the maintenance window \(AWS CLI\) \(p. 735\)](#)

## To configure an automation as a registered task for a maintenance window

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Create a command to configure an automation as a registered task for a maintenance window. Replace each *example resource placeholder* with your own information.

Linux & macOS

```
aws ssm register-task-with-maintenance-window \
--window-id window ID \
--name task name \
--task-arn runbook name \
--targets Key=targets,Values=value \
--service-role-arn IAM role arn \
--task-type AUTOMATION \
--task-invocation-parameters task parameters \
--priority task priority \
--max-concurrency 10% \
--max-errors 5
```

### Note

If you configure an automation 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.

For maintenance window tasks without a target specified, you can't supply values for `--max-errors` and `--max-concurrency`. Instead, the system inserts a placeholder value of 1, which might be reported in the response to commands such as [describe-maintenance-window-tasks](#) and [get-maintenance-window-task](#). These values don't affect the running of your task and can be ignored.

For information about maintenance window tasks that don't require targets, see [Registering maintenance window tasks without targets \(p. 794\)](#).

Windows

```
aws ssm register-task-with-maintenance-window ^
--window-id window ID ^
--name task name ^
--task-arn runbook name ^
--targets Key=targets,Values=value ^
--service-role-arn IAM role arn ^
--task-type AUTOMATION ^
--task-invocation-parameters task parameters ^
--priority task priority ^
--max-concurrency 10% ^
--max-errors 5
```

### Note

If you configure an automation 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.

For maintenance window tasks without a target specified, you can't supply values for `--max-errors` and `--max-concurrency`. Instead, the system inserts a placeholder value of 1, which might be reported in the response to commands such as [describe-maintenance-window-tasks](#) and [get-maintenance-window-task](#). These values don't affect the running of your task and can be ignored.

For information about maintenance window tasks that don't require targets, see [Registering maintenance window tasks without targets \(p. 794\)](#).

### PowerShell

```
Register-S SMTTaskWithMaintenanceWindow `
-WindowId window ID `
-Name "task name" `
-TaskArn "runbook name" `
-Target @{ Key="targets";Values="value" } `
-ServiceRoleArn "IAM role arn" `
-TaskType "AUTOMATION" `
-Automation_Parameter @{ "task parameter"="task parameter value" } `
-Priority task priority `
-MaxConcurrency 10% `
-MaxError 5
```

### Note

If you configure an automation 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.

For maintenance window tasks without a target specified, you can't supply values for `-MaxError` and `-MaxConcurrency`. Instead, the system inserts a placeholder value of 1, which might be reported in the response to commands such as `Get-SSMMaintenanceWindowTaskList` and `Get-SSMMaintenanceWindowTask`. These values don't affect the running of your task and can be ignored.

For information about maintenance window tasks that don't require targets, see [Registering maintenance window tasks without targets \(p. 794\)](#).

The following example configures an automation as a registered task to a maintenance window with priority 1. It also demonstrates omitting the `--targets`, `--max-errors`, and `--max-concurrency` options for a targetless maintenance window task. The automation uses the `AWS-StartEC2Instance` runbook and the specified Automation assume role to start EC2 instances registered as targets to the maintenance window. The maintenance window runs the automation simultaneously on 5 instances maximum at any given time. Also, the registered task stops running on more instances for a particular interval if the error count exceeds 1.

### Linux & macOS

```
aws ssm register-task-with-maintenance-window \
--window-id mw-0c50858d01EXAMPLE \
--name StartEC2Instances \
--task-arn AWS-StartEC2Instance \
--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
```

### Windows

```
aws ssm register-task-with-maintenance-window ^
--window-id mw-0c50858d01EXAMPLE ^
--name StartEC2Instances ^
--task-arn AWS-StartEC2Instance ^
--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
```

### PowerShell

```
Register-SMTTaskWithMaintenanceWindow `
-WindowId mw-0c50858d01EXAMPLE `
-Name "StartEC2" `
-TaskArn "AWS-StartEC2Instance" `
-ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowRole" `
-TaskType "AUTOMATION" `
-Automation_Parameter @{
 "InstanceId"="{{TARGET_ID}}";
 "AutomationAssumeRole"="arn:aws:iam::123456789012:role/AutomationAssumeRole"
} `
-Priority 1
```

The command returns details for the new registered task similar to the following.

### Linux & macOS

```
{
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
```

### Windows

```
{
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
```

### PowerShell

```
4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE
```

- To view the registered task, run the following command. Replace `maintenance windows ID` with your own information.

### Linux & macOS

```
aws ssm describe-maintenance-window-tasks \
--window-id maintenance window ID
```

## Windows

```
aws ssm describe-maintenance-window-tasks ^
--window-id maintenance window ID
```

## PowerShell

```
Get-SSMMaintenanceWindowTaskList `^
-WindowId maintenance window ID
```

The system returns information like the following.

## Linux & macOS

```
{
 "Tasks": [
 {
 "ServiceRoleArn": "arn:aws:iam::123456789012:role/MaintenanceWindowRole",
 "MaxErrors": "1",
 "TaskArn": "AWS-StartEC2Instance",
 "MaxConcurrency": "1",
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
 "TaskParameters": {},
 "Priority": 1,
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Type": "AUTOMATION",
 "Targets": [
],
 "Name": "StartEC2"
 }
]
}
```

## Windows

```
{
 "Tasks": [
 {
 "ServiceRoleArn": "arn:aws:iam::123456789012:role/MaintenanceWindowRole",
 "MaxErrors": "1",
 "TaskArn": "AWS-StartEC2Instance",
 "MaxConcurrency": "1",
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
 "TaskParameters": {},
 "Priority": 1,
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Type": "AUTOMATION",
 "Targets": [
],
 "Name": "StartEC2"
 }
]
}
```

## PowerShell

```
Description :
LoggingInfo :
MaxConcurrency : 5
MaxErrors : 1
Name : StartEC2
Priority : 1
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowRole
Targets : {}
TaskArn : AWS-StartEC2Instance
TaskParameters : {}
Type : AUTOMATION
WindowId : mw-0c50858d01EXAMPLE
WindowTaskId : 4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE
```

## Running automations by using different security models

This section includes information about how to run automations by using different security models.

### Topics

- [Running an automation as the current authenticated user \(p. 456\)](#)
- [Running an automation by using an IAM service role \(p. 460\)](#)
- [Running an automation by using delegated administration \(p. 464\)](#)

### Running an automation as the current authenticated user

The following procedures describe how to run an automation that runs in the context of the current AWS Identity and Access Management (IAM) user using the AWS Systems Manager console and AWS Command Line Interface (AWS CLI). Running the automation 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 use the runbook, and any actions called by the runbook. If the IAM user has have administrator permissions in IAM, then you have permission to use this runbook.

#### Running an automation as the current authenticated user (console)

The following procedure describes how to use the Systems Manager console to run an automation as the current authenticated user.

#### To use the runbook as the current authenticated user

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the **Automation document** list, choose a runbook. Choose one or more options in the **Document categories** pane to filter SSM documents according to their purpose. To view a runbook that you own, choose the **Owned by me** tab. To view a runbook that is shared with your account, choose the **Shared with me** tab. To view all runbooks, choose the **All documents** tab.

#### Note

You can view information about a runbook by choosing the runbook 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 runbook is updated periodically and a new default version is assigned.

- **Latest version at runtime:** Choose this option if the Automation runbook 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 as the current authenticated user.

7. In the **Input parameters** section, specify the required inputs. To run the automation as the current authenticated user, don't specify an IAM service role for the value `AutomationAssumeRole`.
8. Choose **Execute**. The console displays the status of the automation.

### Running an automation 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 as the current authenticated user.

#### To run the automation as the current authenticated user

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to start an automation as the current authenticated user. Replace each *example resource placeholder* with your own information.

Linux & macOS

```
aws ssm start-automation-execution \
--document-name runbook name \
--parameters runbook parameters
```

Windows

```
aws ssm start-automation-execution ^
--document-name runbook name ^
--parameters runbook parameters
```

PowerShell

```
Start-SSMAutomationExecution `-
-DocumentName runbook name `-
-Parameter runbook parameters
```

Here is an example using the runbook `AWS-RestartEC2Instance` to restart the specified Amazon Elastic Compute Cloud (Amazon EC2) instance.

Linux & macOS

```
aws ssm start-automation-execution \
--document-name "AWS-RestartEC2Instance" \
--parameters "InstanceId=i-02573cafccEXAMPLE"
```

## Windows

```
aws ssm start-automation-execution ^
--document-name "AWS-RestartEC2Instance" ^
--parameters "InstanceId=i-02573cafefEXAMPLE"
```

## PowerShell

```
Start-SSMAutomationExecution `-
-DocumentName AWS-RestartEC2Instance `-
-Parameter @{"InstanceId"="i-02573cafefEXAMPLE"}
```

The system returns information like the following.

## Linux & macOS

```
{
 "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab"
}
```

## Windows

```
{
 "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab"
}
```

## PowerShell

```
4105a4fc-f944-11e6-9d32-0123456789ab
```

3. Run the following command to retrieve the status of the automation. Replace the *automation execution ID* with your own information.

## Linux & macOS

```
aws ssm describe-automation-executions \
--filter "Key=ExecutionId,Values=automation execution ID"
```

## Windows

```
aws ssm describe-automation-executions ^
--filter "Key=ExecutionId,Values=automation execution ID"
```

## PowerShell

```
Get-SSMAutomationExecutionList | `
Where {$_.AutomationExecutionId -eq "automation execution ID"}
`
```

The system returns information like the following.

## Linux & macOS

```
{
```

```

 "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

```

{
 "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

AutomationExecutionId	:	4105a4fc-f944-11e6-9d32-0123456789ab
AutomationExecutionStatus	:	InProgress
AutomationType	:	Local
CurrentAction	:	aws:changeInstanceState
CurrentStepName	:	startInstances
DocumentName	:	AWS-RestartEC2Instance
DocumentVersion	:	1
ExecutedBy	:	arn:aws:sts::123456789012:assumed-role/Administrator/ Admin
ExecutionEndTime	:	1/1/0001 12:00:00 AM

```
ExecutionStartTime : 7/31/2019 7:17:28 PM
FailureMessage :
LogFile :
MaxConcurrency :
MaxErrors :
Mode : Auto
Outputs : {}
ParentAutomationExecutionId :
ResolvedTargets : Amazon.SimpleSystemsManagement.Model.ResolvedTargets
Target :
TargetMaps :
TargetParameterName :
Targets : {}
```

## Running an automation by using an IAM service role

The following procedures describe how to use the AWS Systems Manager console and AWS Command Line Interface (AWS CLI) to run an automation using an AWS Identity and Access Management (IAM) service role that is known in this case as an *assume role*. The service role gives the automation 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 permissions on a resource, such as an Amazon Elastic Compute Cloud (Amazon EC2) instance, but you want to allow the user to run an automation that performs a specific set of actions. In this scenario, you can create a service role with elevated permissions and allow the user to run the automation.

### Before you begin

Before you complete the following procedures, you must create the IAM service role and configure a trust relationship for Automation, a capability of AWS Systems Manager. For more information, see [Task 1: Create a service role for Automation \(p. 403\)](#).

### Running an automation by using an IAM service role (console)

The following procedure describes how to use the Systems Manager console to run an automation that uses an IAM service role (or *assume role*).

#### To run an automation using a service role

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the **Automation document** list, choose a runbook. Choose one or more options in the **Document categories** pane to filter SSM documents according to their purpose. To view a runbook that you own, choose the **Owned by me** tab. To view a runbook that is shared with your account, choose the **Shared with me** tab. To view all runbooks, choose the **All documents** tab.

#### Note

You can view information about a runbook by choosing the runbook 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 runbook is updated periodically and a new default version is assigned.
  - **Latest version at runtime:** Choose this option if the Automation runbook 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 using a service role.

7. In the **Input parameters** section, specify the required inputs. For **AutomationAssumeRole**, enter the name of the IAM service role that functions as an assume role.

**Tip**

You can select a role from the list, or begin typing the name of a role and select it from the filtered results.

8. Choose **Execute**. The console displays the status of the automation.

### Running an automation by using an IAM service role (command line)

The following procedure describes how to use the AWS CLI (on Linux, macOS, or Windows) or AWS Tools for PowerShell to run an automation that uses an IAM service role that functions in this case as an *assume role*.

#### To run an automation using a service role

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to start an automation that uses an IAM service role. Replace each *example resource placeholder* with your own information.

Linux & macOS

```
aws ssm start-automation-execution \
--document-name runbook name \
--parameters "runbook
parameters","AutomationAssumeRole=arn:aws:iam::123456789012:role/AmazonSSMAutomationRole"
```

Windows

```
aws ssm start-automation-execution ^
--document-name runbook name ^
--parameters "runbook
parameters","AutomationAssumeRole=arn:aws:iam::123456789012:role/AmazonSSMAutomationRole"
```

PowerShell

```
Start-SMAutomationExecution ^
-DocumentName runbook name ^
-Parameter @{
 "runbook parameters"="ParameterValues";
 "AutomationAssumeRole"="arn:aws:iam::123456789012:role/AmazonSSMAutomationRole"}
```

Here is an example using the runbook `AWS-RestartEC2Instance` to restart the specified EC2 instance using the IAM service role `AmazonSSMAutomationRole`.

Linux & macOS

```
aws ssm start-automation-execution \
--document-name "AWS-RestartEC2Instance" \
--parameters
"InstanceId=i-02573cafefEXAMPLE","AutomationAssumeRole=arn:aws:iam::123456789012:role/
AmazonSSMAutomationRole"
```

Windows

```
aws ssm start-automation-execution ^
--document-name "AWS-RestartEC2Instance" ^
--parameters
"InstanceId=i-02573cafefEXAMPLE","AutomationAssumeRole=arn:aws:iam::123456789012:role/
AmazonSSMAutomationRole"
```

PowerShell

```
Start-SSMAutomationExecution `-
-DocumentName "AWS-RestartEC2Instance" `-
-Parameter @{
 "InstanceId"="i-02573cafefEXAMPLE";
 "AutomationAssumeRole"="arn:aws:iam::123456789012:role/
AmazonSSMAutomationRole"}
```

The system returns information like the following.

Linux & macOS

```
{ "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab" }
```

Windows

```
{ "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab" }
```

PowerShell

```
4105a4fc-f944-11e6-9d32-0123456789ab
```

3. Run the following command to retrieve the status of the automation. Replace **automation execution ID** with your own information.

Linux & macOS

```
aws ssm describe-automation-executions \
--filter "Key=ExecutionId,Values=automation execution ID"
```

Windows

```
aws ssm describe-automation-executions ^
```

```
--filter "Key=ExecutionId,Values=automation execution ID"
```

PowerShell

```
Get-SMMAutomationExecutionList | `
Where {$_.AutomationExecutionId -eq "automation execution ID"}
```

The system returns information like the following.

Linux & macOS

```
{
 "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

```
{
 "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

```
AutomationExecutionId : 4105a4fc-f944-11e6-9d32-0123456789ab
AutomationExecutionStatus : InProgress
AutomationType : Local
CurrentAction : aws:changeInstanceState
CurrentStepName : startInstances
DocumentName : AWS-RestartEC2Instance
DocumentVersion : 1
ExecutedBy : arn:aws:sts::123456789012:assumed-role/Administrator/
Admin
ExecutionEndTime : 1/1/0001 12:00:00 AM
ExecutionStartTime : 7/31/2019 7:17:28 PM
FailureMessage :
LogFile :
MaxConcurrency :
MaxErrors :
Mode : Auto
Outputs : {}
ParentAutomationExecutionId :
ResolvedTargets : Amazon.SimpleSystemsManagement.Model.ResolvedTargets
Target :
TargetMaps : {}
TargetParameterName :
Targets : {}
```

For more examples of how to use Systems Manager Automation, see [Automation walkthroughs \(p. 604\)](#). For information about how to get started with Automation, see [Setting up Automation \(p. 401\)](#).

## Running an automation by using delegated administration

When you run an automation, by default, the automation runs in the context of the AWS Identity and Access Management (IAM) user who initiated the automation. 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. As a security best practice, we recommend that you run automation 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 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 Amazon Elastic Compute Cloud (Amazon EC2) instances. Automation is a capability of AWS Systems Manager. The automation restarts the instances, but the service role doesn't give the user permission to access those instances.

You can specify a service role at runtime when you run an automation, or you can create custom runbooks and specify the service role directly in the runbook. If you specify a service role, either at runtime or in a runbook, 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 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 allows 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 runbook (based on a preexisting runbook) that specifies the service role (Runbook 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 allows enhanced security in complex organizations.

**Important**

As a security best practice, we recommend that you always use a service role to run automations, even if you're 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 \(p. 465\)](#)
- [Create an IAM service role for Automation \(p. 466\)](#)
- [Create a custom runbook \(p. 466\)](#)
- [Run the custom runbook \(p. 467\)](#)

[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 automations, but the user doesn't have access to the AWS resources targeted by Automation. The operator can also view the results of the automations. 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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:DescribeAutomationExecutions",
 "ssm:DescribeAutomationStepExecutions",
 "ssm:DescribeDocument",
 "ssm:GetAutomationExecution",
 "ssm:GetDocument",
 "ssm>ListDocuments",
 "ssm:StartAutomationExecution"
]
 }
]
}
```

```
 "ssm>ListDocumentVersions",
 "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 in to 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 allow Automation to trust it

1. Create the Automation service role. For information, see [Task 1: Create a service role for Automation \(p. 403\)](#).
2. Note the service role Amazon Resource Name (ARN). You will specify this ARN in the next procedure.

### Create a custom runbook

This section describes how to create a custom runbook that restarts EC2 instances. AWS provides a runbook for restarting instances called `AWS-RestartEC2Instance`. The following procedure copies the content of that runbook to show you how to enter the service role in a runbook when you create your own. By specifying the service role directly in the runbook, the user doesn't require `iam:PassRole` permissions when using the runbook. Without `iam:PassRole` permissions, the user can't use the service role elsewhere in AWS.

#### To create a custom runbook

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Documents**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon ( $\equiv$ ) to open the navigation pane, and then choose **Documents** in the navigation pane.

3. Choose **Create document**.
4. In the **Name** field, enter a name for the runbook, 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.

```
{
 "description": "Restart EC2 instances(s)",
 "schemaVersion": "0.3",
 "assumeRole": "AssumeRoleARN",
 "parameters": {
 "InstanceId": {
 "type": "StringList",
 "description": "(Required) EC2 Instance to restart"
 }}
```

```
 },
 },
 "mainSteps": [
 {
 "name": "stopInstances",
 "action": "aws:changeInstanceState",
 "inputs": {
 "InstanceIds": "{{ InstanceId }}",
 "DesiredState": "stopped"
 }
 },
 {
 "name": "startInstances",
 "action": "aws:changeInstanceState",
 "inputs": {
 "InstanceIds": "{{ InstanceId }}",
 "DesiredState": "running"
 }
 }
]
}
```

7. Choose **Create document**.

### Run the custom runbook

The following procedure describes how to run the runbook you just created using the restricted operator role you created earlier in this topic. The user can run the runbook you created earlier because their IAM account permissions allow them to see and run the runbook. The user can't, however, log on to the instances that you will restart with this automation.

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.
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.
3. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 runbook 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, enter 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 automations 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. Automation is a capability of

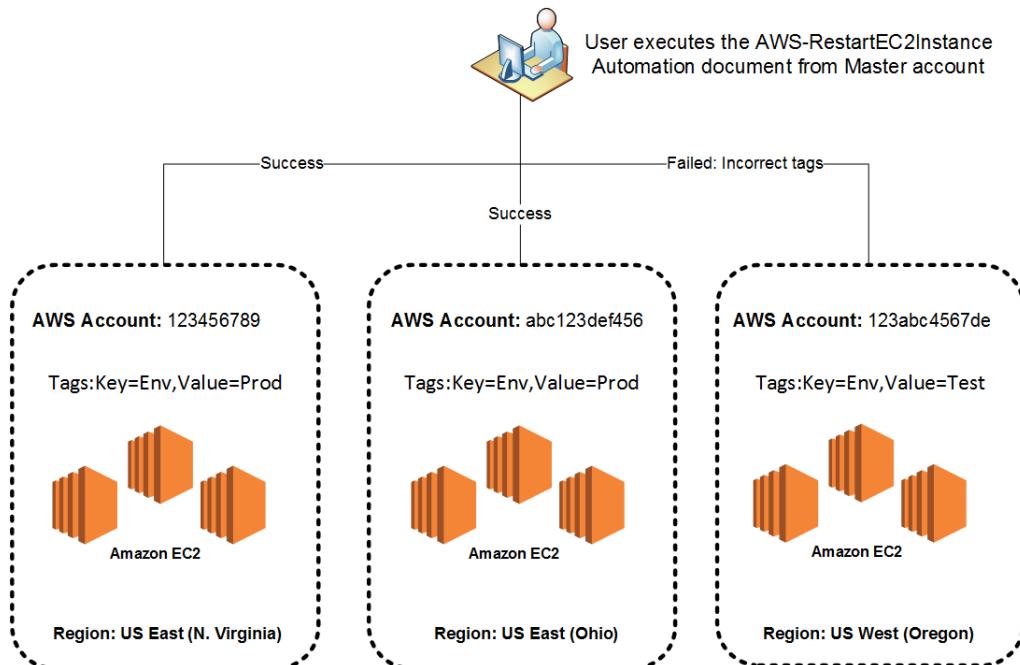
AWS Systems Manager. Running automations in multiple Regions and accounts or OUs reduces the time required to administer your AWS resources while enhancing the security of your computing environment.

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 Amazon Elastic Compute Cloud (Amazon EC2) EC2 instances, at scale. The following graphic shows an example of a user who is running the `AWS-RestartEC2Instances` runbook 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.

**Multi-Region and Account Automation to Restart EC2 Instances Tagged with: Key=Env,Value=Prod**



**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?](#)
2. Sign in to the AWS Identity and Access Management (IAM) account that you want to configure as the Automation Primary account.
3. Use the procedure in this topic to create the IAM automation role named **AWS-SystemsManager-AutomationExecutionRole**. This role gives the user permission to run automations.
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 automations in multiple AWS accounts and OUs.
5. Choose the runbook, Regions, and accounts or OUs where you want to run the automation.

**Note**

Automations don't run recursively through OUs. Be sure the target OU contains the desired accounts. If you choose a custom runbook, the runbook must be shared with all of the target accounts. For information about sharing runbooks, see [Sharing SSM documents \(p. 1361\)](#). For information about using shared runbooks, see [Using shared SSM documents \(p. 1369\)](#).

6. Run the automation. When running automations across multiple Regions, accounts, or OUs, the automation you run from the primary account starts child automations in each of the target accounts. The automation in the primary account will have aws:executeAutomation steps for each of the target accounts.
7. Use the [GetAutomationExecution](#), [DescribeAutomationStepExecutions](#), and [DescribeAutomationExecutions](#) API operations from the AWS Systems Manager console or the AWS CLI to monitor automation progress. The output of the steps for the automation in your primary account will be the AutomationExecutionId of the child automations. To view the output of the child automations created in your target accounts, be sure to specify the appropriate account, Region, and AutomationExecutionId in your request.

## Setting up management account permissions for multi-Region and multi-account automation

Use the following procedure to create the required IAM roles for Systems Manager Automation multi-Region and multi-account automation 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. We recommend using AWS CloudFormation StackSets to create the **AWS-SystemsManager-AutomationExecutionRole** role in the accounts 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 automations by using AWS CloudFormation

1. Download and unzip the [AWS-SystemsManager-AutomationExecutionRole.zip](#) file. This file contains the AWS-SystemsManager-AutomationExecutionRole.json AWS CloudFormation template file.

**Note**

We recommend not changing the role name as specified in the template to something other than AWS-SystemsManager-AutomationExecutionRole. Otherwise, your multi-Region and multi-Account automations might fail.

2. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
3. Choose **Create stack**.
4. In the **Specify template** section, choose **Upload a template**.
5. Choose **Choose file**, and then choose the AWS-SystemsManager-AutomationExecutionRole.json AWS CloudFormation template file.
6. Choose **Next**.
7. On the **Specify stack 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 **Configure stack 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 with custom names** option.
12. Choose **Create stack**.

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-SystemsManager-AutomationAdministrationRole.zip](#) file and repeat this procedure for the AWS-SystemsManager-AutomationAdministrationRole role. You only need to create the AWS-SystemsManager-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-SystemsManager-AutomationAdministrationRole role. We recommend not changing the role name as specified in the template to something besides AWS-SystemsManager-AutomationAdministrationRole. Otherwise, your multi-Region and multi-account automations 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 in multiple Regions and accounts

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the **Automation document** list, choose a runbook. Choose one or more options in the **Document categories** pane to filter SSM documents according to their purpose. To view a runbook that you own, choose the **Owned by me** tab. To view a runbook that is shared with your account, choose the **Shared with me** tab. To view all runbooks, choose the **All documents** tab.

**Note**

You can view information about a runbook by choosing the runbook 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 runbook is updated periodically and a new default version is assigned.
  - **Latest version at runtime:** Choose this option if the Automation runbook 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 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 automations 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 simultaneously.
    - Choose **percent** to enter a percentage of the total number of account-Region pairs that can run the automation 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 automation to other resources.
    - Choose **percent** to enter a percentage of errors allowed before Automation stops sending the automation 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 runbook 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 runbook on all instances in the current AWS account and AWS 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 might 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` runbook, then you don't need to specify or choose instance IDs in the **Input parameters** section. The automation 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 in multiple Regions and accounts

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Use the following format to create a command to run an automation in multiple Regions and accounts. Replace each *example resource placeholder* with your own information.

Linux & macOS

```
aws ssm start-automation-execution \
--document-name runbook name \
--parameters AutomationAssumeRole=arn:aws:iam::management account ID:role/AWS-
SystemsManager-AutomationAdministrationRole \
--target-parameter-name parameter name \
--targets Key=tag key,Values=value \
--target-locations Accounts=account ID,account ID 2,Regions=Region,Region
2,ExecutionRoleName=AWS-SystemsManager-AutomationExecutionRole
```

Windows

```
aws ssm start-automation-execution ^
--document-name runbook name ^
```

```
--parameters AutomationAssumeRole=arn:aws:iam::management account ID:role/AWS-
SystemsManager-AutomationAdministrationRole ^
--target-parameter-name parameter name ^
--targets Key=tag key,Values=value ^
--target-locations Accounts=account ID,account ID 2,Regions=Region,Region
2,ExecutionRoleName=AWS-SystemsManager-AutomationExecutionRole
```

### PowerShell

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "tag key"
$Targets.Values = "value"

Start-SMAutomationExecution ^
-DocumentName "runbook name" ^
-Parameter @{
 "AutomationAssumeRole"="arn:aws:iam::management account ID:role/AWS-
SystemsManager-AutomationAdministrationRole" } ^
-TargetParameterName "parameter name" ^
-Target $Targets ^
-TargetLocation @{
 "Accounts"="account ID","account ID 2";
 "Regions"="Region","Region 2";
 "ExecutionRoleName"="AWS-SystemsManager-AutomationExecutionRole" }
```

Here are several 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 & macOS

```
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

```
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

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "tag:Env"
$Targets.Values = "PROD"
```

```
Start-SSSMAutomationExecution `~
 -DocumentName "AWS-RestartEC2Instance" `~
 -Parameter @{
 "AutomationAssumeRole"="arn:aws:iam::123456789012:role/AWS-SystemsManager-
 AutomationAdministrationRole" } `~
 -TargetParameterName "InstanceId" `~
 -Target $Targets `~
 -TargetLocation @{
 "Accounts"="123456789012","987654321098";
 "Regions"="us-east-2","us-west-1";
 "ExecutionRoleName"="AWS-SystemsManager-AutomationExecutionRole" }
```

**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 & macOS

```
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

```
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

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ResourceGroup"
$Targets.Values = "prod-instances"

Start-SSSMAutomationExecution `~
 -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 & macOS

```
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
```

#### Windows

```
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

```
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ResourceGroup"
$Targets.Values = "WebServices"

Start-SMAutomationExecution `
-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 & macOS

```
{
 "AutomationExecutionId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
```

#### Windows

```
{
 "AutomationExecutionId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
```

```
}
```

### PowerShell

```
4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE
```

3. Run the following command to view details for the automation. Replace *automation execution ID* with your own information.

### Linux & macOS

```
aws ssm describe-automation-executions \
--filters Key=ExecutionId,Values=automation execution ID
```

### Windows

```
aws ssm describe-automation-executions ^
--filters Key=ExecutionId,Values=automation execution ID
```

### PowerShell

```
Get-SSSMAutomationExecutionList | `
Where {$_.AutomationExecutionId -eq "automation execution ID"}
```

4. Run the following command to view details about the automation progress.

### Linux & macOS

```
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-SSSMAutomationExecution `
-AutomationExecutionId a4a3c0e9-7efd-462a-8594-01234EXAMPLE
```

#### Note

You can also monitor the status of the automation in the console. In the **Automation executions** list, choose the automation you just ran and then choose the **Execution steps** tab. This tab shows the status of the automation actions.

## Related content

[Centralized multi-account and multi-Region patching with AWS Systems Manager Automation](#)

# Systems Manager Automation actions reference

## Note

Automation documents are now referred to as runbooks.

This reference describes the Automation actions that you can specify in an Automation runbook. Automation is a capability of AWS Systems Manager. These actions can't be used in other types of Systems Manager (SSM) documents. For information about plugins for other types of SSM documents, see [Systems Manager Command document plugin reference \(p. 1314\)](#).

Systems Manager Automation runs steps defined in Automation runbooks. 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 runbook.

## Note

The following plugins are supported on EC2 instances for macOS:

- `aws:configurePackage`
- `aws:refreshAssociation`
- `aws:runShellScript`
- `aws:softwareInventory`

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 runbooks, 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 runbook.

## 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 runbooks (`AWS-*` runbooks) such as the `AWS-ConfigureS3BucketLogging`, `AWS-CreateDynamoDBBackup`, and `AWS-RestartEC2Instance` runbooks, to name a few. This requirement also applies to any custom Automation runbooks 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, configure the service role with permission to invoke those services. You can give 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 \(p. 405\)](#).

## Topics

- [Properties shared by all actions \(p. 478\)](#)
- [aws:approve – Pause an automation for manual approval \(p. 483\)](#)
- [aws:assertAwsResourceProperty – Assert an AWS resource state or event state \(p. 486\)](#)
- [aws:branch – Run conditional automation steps \(p. 488\)](#)
- [aws:changeInstanceState – Change or assert instance state \(p. 490\)](#)
- [aws:copyImage – Copy or encrypt an Amazon Machine Image \(p. 492\)](#)
- [aws:createImage – Create an Amazon Machine Image \(p. 493\)](#)
- [aws:createStack – Create an AWS CloudFormation stack \(p. 495\)](#)
- [aws:createTags – Create tags for AWS resources \(p. 501\)](#)

- [aws:deleteImage – Delete an Amazon Machine Image \(p. 503\)](#)
- [aws:deleteStack – Delete an AWS CloudFormation stack \(p. 503\)](#)
- [aws:executeAutomation – Run another automation \(p. 505\)](#)
- [aws:executeAwsApi – Call and run AWS API operations \(p. 508\)](#)
- [aws:executeScript – Run a script \(p. 511\)](#)
- [aws:executeStateMachine – Run an AWS Step Functions state machine \(p. 513\)](#)
- [aws:invokeWebhook – Invoke an Automation webhook integration \(p. 514\)](#)
- [aws:invokeLambdaFunction – Invoke an AWS Lambda function \(p. 515\)](#)
- [aws:pause – Pause an automation \(p. 518\)](#)
- [aws:runCommand – Run a command on a managed instance \(p. 518\)](#)
- [aws:runInstances – Launch an Amazon EC2 instance \(p. 523\)](#)
- [aws:sleep – Delay an automation \(p. 527\)](#)
- [aws:waitForAwsResourceProperty – Wait on an AWS resource property \(p. 529\)](#)
- [Automation system variables \(p. 531\)](#)

## Properties shared by all actions

Common properties are parameters or options that are found in all actions. Some options define 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 \(p. 480\)](#)

An identifier that must be unique across all step names in the runbook.

Type: String

Required: Yes

### [action \(p. 480\)](#)

The name of the action the step is to run. [aws:runCommand – Run a command on a managed instance \(p. 518\)](#) is an example of an action you can specify here. This document provides detailed information about all available actions.

Type: String

Required: Yes

### [maxAttempts \(p. 480\)](#)

The number of times the step should be retried in case of failure. If the value is greater than 1, the step isn't considered to have failed until all retry attempts have failed. The default value is 1.

Type: Integer

Required: No

### [timeoutSeconds \(p. 481\)](#)

The timeout value for the step. If the timeout is reached and the value of `maxAttempts` is greater than 1, then the step isn't considered to have timed out until all retries have been attempted.

Type: Integer

Required: No

[onFailure \(p. 481\)](#)

Indicates whether the automation should stop, 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

[onCancel \(p. 481\)](#)

Indicates which step the automation should go to in the event that a user cancels the automation. Automation runs the cancellation workflow for a maximum of two minutes.

Type: String

Valid values: Abort | step:*step\_name*

Required: No

The onCancel property doesn't support moving to the following actions:

- aws:approve
- aws:copyImage
- aws:createImage
- aws:createStack
- aws:createTags
- aws:pause
- aws:runInstances
- aws:sleep

[isEnd \(p. 481\)](#)

This option stops an automation at the end of a specific step. The automation stops if the step failed or succeeded. The default value is false.

Type: Boolean

Valid values: true | false

Required: No

[nextStep \(p. 481\)](#)

Specifies which step in an automation to process next after successfully completing a step.

Type: String

Required: No

[isCritical \(p. 482\)](#)

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. This property is only evaluated if you explicitly define it in your step. If the onFailure property is set to Continue in a step, the value defaults to false. Otherwise, the default value for this option is true.

Type: Boolean

Valid values: true | false

Required: No

[inputs \(p. 483\)](#)

The properties specific to the action.

Type: Map

Required: Yes

## Example

```

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 run this runbook."
 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: $.Payload.noSnapshotFound
 Type: String
nextStep: branchFromResults
- name: branchFromResults
 action: aws:branch
 onFailure: Abort
 onCancel: step:startInstance
inputs:
 Choices:
 - NextStep: createNewRootVolumeFromSnapshot
 Not:
 Variable: "{{ getSnapshotsByStartTime.noSnapshotFound }}"
 StringEquals: "NoSnapshotFound"
 isEnd: true
- name: createNewRootVolumeFromSnapshot
 action: aws:executeAwsApi
 onFailure: Abort
 inputs:
 Service: ec2
 Api: CreateVolume
 AvailabilityZone: "{{ getInstanceDetails.availabilityZone }}"
 SnapshotId: "{{ getSnapshotsByStartTime.latestSnapshotId }}"
outputs:
 - Name: newRootVolumeId
 Selector: "$.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

```

```

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 }}"

```

## aws :approve – Pause an automation for manual approval

Temporarily pauses an automation until designated principals either approve or reject the action. After the required number of approvals is reached, the automation resumes. You can insert the approval step any place in the `mainSteps` section of your runbook.

### 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 until one approver either accepts or rejects the automation. Upon approval, the automation runs a simple PowerShell command.

### 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:
 NotificationArn: arn:aws:sns:us-east-2:12345678901:AutomationApproval
 Message: "{{ message }}"
 MinRequiredApprovals: 1
 Approvers:
 - arn:aws:iam::12345678901:user/AWS-User-1
- name: run
 action: aws:runCommand
 inputs:
 InstanceIds:
 - i-1a2b3c4d5e6f7g
 DocumentName: AWS-RunPowerShellScript

```

```
Parameters:
 commands:
 - date
```

## 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": {
 "NotificationArn": "arn:aws:sns:us-east-2:12345678901:AutomationApproval",
 "Message": "{{ message }}",
 "MinRequiredApprovals": 1,
 "Approvers": [
 "arn:aws:iam::12345678901:user/AWS-User-1"
]
 }
 },
 {
 "name": "run",
 "action": "aws:runCommand",
 "inputs": {
 "InstanceIds": [
 "i-1a2b3c4d5e6f7g"
],
 "DocumentName": "AWS-RunPowerShellScript",
 "Parameters": {
 "commands": [
 "date"
]
 }
 }
 }
]
}
```

You can approve or deny Automations that are waiting for approval in the console.

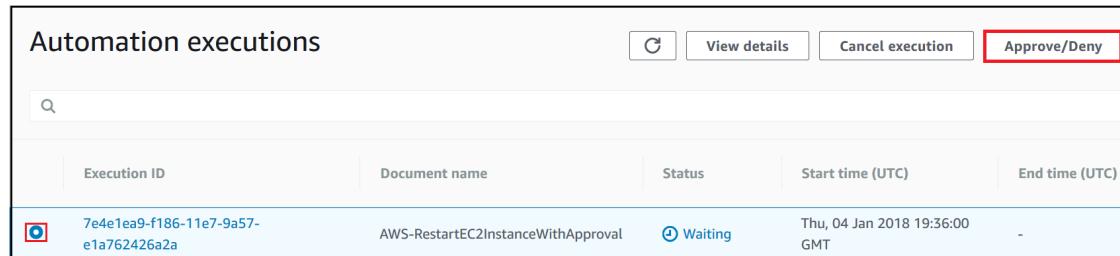
### To approve or deny waiting Automations

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/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 the option next to an Automation with a status of **Waiting**.



Automation executions				
Execution ID	Document name	Status	Start time (UTC)	End time (UTC)
7e4e1ea9-f186-11e7-9a57-e1a762426a2a	AWS-RestartEC2InstanceWithApproval	Waiting	Thu, 04 Jan 2018 19:36:00 GMT	-

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

```
NotificationArn: arn:aws:sns:us-west-1:12345678901:Automation-ApprovalRequest
Message: Please approve this step of the Automation.
MinRequiredApprovals: 3
Approvers:
- IamUser1
- IamUser2
- arn:aws:iam::12345678901:user/IamUser3
- arn:aws:iam::12345678901:role/IamRole
```

### JSON

```
{
 "NotificationArn": "arn:aws:sns:us-west-1:12345678901:Automation-ApprovalRequest",
 "Message": "Please approve this step of the Automation.",
 "MinRequiredApprovals": 3,
 "Approvers": [
 "IamUser1",
 "IamUser2",
 "arn:aws:iam::12345678901:user/IamUser3",
 "arn:aws:iam::12345678901:role/IamRole"
]
}
```

### NotificationArn

The Amazon Resource Name (ARN of an Amazon Simple Notification Service (Amazon SNS) topic for Automation approvals. When you specify an `aws:approve` step in a runbook, 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 Amazon 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. 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 allows 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 Amazon Elastic Compute Cloud (Amazon EC2) instance to start. Then it will call the Amazon EC2 `DescribeInstanceStatus` API operation with the `DesiredValue` property of `running`. This ensures that the automation 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 runbook \(p. 565\)](#).

### Input

Inputs are defined by the API operation that you choose.

#### YAML

```
action: aws:assertAwsResourceProperty
```

```
inputs:
 Service: The official namespace of the service
 Api: The API operation or method name
 API operation 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 operation or method name",
 "API operation inputs or parameters": "A value",
 "PropertySelector": "Response object",
 "DesiredValues": [
 "Desired property values"
]
 }
}
```

#### Service

The AWS service namespace that contains the API operation 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 operation that you want to run. You can view the API operations (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 operations (methods) for Amazon Relational Database Service (Amazon RDS) are listed on the following page: [Amazon RDS methods](#).

Type: String

Required: Yes

#### API operation inputs

One or more API operation 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:
 Service: The official namespace of the service
 Api: The API operation 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 operation name",
 "API input 1":"A value",
 "API Input 2":"A value",
 "API Input 3":"A value"
}
```

Type: Determined by chosen API operation

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: String

Required: Yes

#### DesiredValues

The expected status or state on which to continue the automation. If you specify a Boolean value, you must use a capital letter such as True or False.

Type: StringList

Required: Yes

## aws:branch – Run conditional automation steps

The `aws:branch` action allows you to create a dynamic automation that evaluates different choices in a single step and then jumps to a different step in the runbook based on the results of that evaluation.

When you specify the `aws:branch` action for a step, you specify `Choices` that the automation must evaluate. The `Choices` can be based on either a value that you specified in the `Parameters` section of the runbook, or a dynamic value generated as the output from the previous step. The automation evaluates each choice by using a Boolean expression. If the first choice is true, then the automation jumps to the step designated for that choice. If the first choice is false, the automation evaluates the next choice. The automation continues evaluating each choice until it processes a true choice. The automation then jumps to the designated step for the true choice.

If none of the choices are true, the automation checks to see if the step contains a `default` value. A `default` value defines a step that the automation should jump to if none of the choices are true. If no `default` value is specified for the step, then the automation processes the next step in the runbook.

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 runbooks

and examples that use different operators, see [Creating dynamic automations with conditional branching \(p. 552\)](#).

## 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 runbook, or a dynamic value generated as the output from the previous step. Here is a YAML sample that evaluates a parameter.

```
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.

```
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 runbook 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 runbook. Or specify an output object from a previous step in the runbook. For more information about creating variables for aws:branch, see [About creating the output variable \(p. 556\)](#).
- **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 a runbook, the system validates each operation in the runbook. If an operation isn't supported, the system returns an error when you try to create the runbook.

#### Default

The name of a step the automation 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 automations with conditional branching \(p. 552\)](#).

## aws:changeInstanceState – Change or assert instance state

Changes or asserts the state of the instance.

This action can be used in assert mode (doesn't run the API to change the state but verifies 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 Amazon Machine Image (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

```
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 don't 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 isn't 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 Amazon Machine Image (AMI) from any AWS 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

```
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

```
{
 "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 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 KMS key 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 an Amazon Machine Image (AMI) from an instance that is either running, stopping, or stopped.

## Input

This action supports the following `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 Elastic Compute Cloud (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 don't want the instance to run after you create an AMI from it, first use the [aws:changeInstanceState – Change or assert instance state \(p. 490\)](#) 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.

Type: String

##### 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.

Type: String

## aws:createStack – Create an AWS CloudFormation stack

Creates an AWS CloudFormation stack from a template.

For supplemental information about creating 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/doc-example-bucket/myStackTemplate
 TimeoutInMinutes: 5
parameters:
 - ParameterKey: LambdaRoleArn
 ParameterValue: "{{LambdaAssumeRole}}"
 - ParameterKey: createdResource
```

```
ParameterValue: createdResource-{{automation:EXECUTION_ID}}
```

## JSON

```
{
 "name": "makeStack",
 "action": "aws:createStack",
 "maxAttempts": 1,
 "onFailure": "Abort",
 "inputs": {
 "Capabilities": [
 "CAPABILITY_IAM"
],
 "StackName": "myStack",
 "TemplateURL": "http://s3.amazonaws.com/doc-example-bucket/myStackTemplate",
 "TimeoutInMinutes": 5,
 "Parameters": [
 {
 "ParameterKey": "LambdaRoleArn",
 "ParameterValue": "{{LambdaAssumeRole}}"
 },
 {
 "ParameterKey": "createdResource",
 "ParameterValue": "createdResource-{{automation:EXECUTION_ID}}"
 }
]
 }
}
```

## Capabilities

A list of values that you specify before 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, CloudFormation knows that you aren't attempting to create a new stack with the same name.

Type: String

Required: No

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: [a-zA-Z0-9][-a-zA-Z0-9]\*

DisableRollback

Set to true to turn off 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 Simple Notification Service (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:

`AWS::*`

- For all custom resources:

`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, CloudFormation grants permissions to all resource types. IAM uses this parameter for 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 CloudFormation assumes to create the stack. CloudFormation uses the role's credentials to make calls on your behalf. CloudFormation always uses

this role for all future operations on the stack. As long as users have permission to operate on the stack, 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, CloudFormation uses the role that was previously associated with the stack. If no role is available, CloudFormation uses a temporary session that is generated from your user credentials.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Required: No

#### StackName

The name that is associated with the stack. The name must be unique in the Region in which you're creating the stack.

##### Note

A stack name can contain only alphanumeric characters (case sensitive) and hyphens. It must start with an alphabetic character and can't 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

Length Constraints: Minimum length of 1. Maximum length of 16384.

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

Length Constraints: Minimum length of 1. Maximum length of 1350.

Required: No

#### Tags

Key-value pairs to associate with this stack. 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

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

**TimeoutInMinutes**

The amount of time that can pass before the stack status becomes `CREATE_FAILED`. If `DisableRollback` isn't set or is set to `false`, the stack will be rolled back.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

## Outputs

**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 \(p. 403\)](#).

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "SQS:*",
 "cloudformation>CreateStack",
 "cloudformation>DescribeStacks"
],
 "Resource": "*"
 }
]
}
```

## aws:createTags – Create tags for AWS resources

Creates new tags for Amazon Elastic Compute Cloud (Amazon EC2) instances or AWS Systems Manager managed instances.

### Input

This action supports most Amazon EC2 `CreateTags` and Systems Manager `AddTagsToResource` parameters. For more information, see [CreateTags](#) and [AddTagsToResource](#).

The following example shows how to tag an Amazon Machine Image (AMI) and an instance as production resources for a particular department.

### 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

```
{
 "name": "createTags",
 "action": "aws:createTags",
 "maxAttempts": 3,
 "onFailure": "Abort",
 "inputs": {
 "ResourceType": "EC2",
 "ResourceIds": [
 "ami-9a3768fa",
 "i-02951acd5111a8169"
],
 "Tags": [
 {
 "Key": "production",
 "Value": ""
 },
 {
 "Key": "department",
 "Value": "devops"
 }
]
 }
}
```

#### ResourceIds

The IDs of the resource(s) to be tagged. If resource type isn't "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 Amazon Machine Image (AMI) and all related snapshots.

### Input

This action supports only one parameter. For more information, see the documentation for [DeregisterImage](#) and [DeleteSnapshot](#).

#### YAML

```
name: deleteMyImage
action: aws:deleteImage
maxAttempts: 3
timeoutSeconds: 180
onFailure: Abort
inputs:
 ImageId: ami-12345678
```

#### JSON

```
{
 "name": "deleteMyImage",
 "action": "aws:deleteImage",
 "maxAttempts": 3,
 "timeoutSeconds": 180,
 "onFailure": "Abort",
 "inputs": {
 "ImageId": "ami-12345678"
 }
}
```

#### ImageId

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

```
name: deleteStack
action: aws:deleteStack
maxAttempts: 1
```

```
onFailure: Abort
inputs:
 StackName: "{{stackName}}"
```

#### 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 CloudFormation knows that you aren't attempting to delete a stack with the same name. You can retry `DeleteStack` requests to verify that CloudFormation received them.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

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, CloudFormation deletes the stack, but doesn't 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 AWS Identity and Access Management (IAM) role that CloudFormation assumes to create the stack. CloudFormation uses the role's credentials to make calls on your behalf. CloudFormation always uses this role for all future operations on the stack. As long as users have permission to operate on the stack, 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, CloudFormation uses the role that was previously associated with the stack. If no role is available, CloudFormation uses a temporary session that is generated from your user credentials.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

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 \(p. 403\)](#).

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "sns:*",
 "cloudformation>DeleteStack",
 "cloudformation>DescribeStacks"
],
 "Resource": "*"
 }
]
}
```

### aws:executeAutomation – Run another automation

Runs a secondary automation by calling a secondary runbook. With this action, you can create runbooks for your most common operations, and reference those runbooks during an automation. This action can simplify your runbooks by removing the need to duplicate steps across similar runbooks.

The secondary automation runs in the context of the user who initiated the primary automation. This means that the secondary automation uses the same AWS Identity and Access Management (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 \(p. 403\)](#).

#### Input

##### YAML

```
name: Secondary_Automation
action: aws:executeAutomation
maxAttempts: 3
timeoutSeconds: 3600
onFailure: Abort
inputs:
 DocumentName: secondaryAutomation
 RuntimeParameters:
 instanceIds:
 - i-1234567890abcdef0
```

## JSON

```
{
 "name": "Secondary_Automation",
 "action": "aws:executeAutomation",
 "maxAttempts": 3,
 "timeoutSeconds": 3600,
 "onFailure": "Abort",
 "inputs": {
 "DocumentName": "secondaryAutomation",
 "RuntimeParameters": {
 "instanceIds": [
 "i-1234567890abcdef0"
]
 }
 }
}
```

### DocumentName

The name of the secondary runbook to run during the step. For runbooks in the same AWS account, specify the runbook name. For runbooks shared from a different AWS account, specify the Amazon Resource Name (ARN) of the runbook. For information about using shared runbooks, see [Using shared SSM documents \(p. 1369\)](#).

Type: String

Required: Yes

### DocumentVersion

The version of the secondary runbook to run. If not specified, Automation runs the default runbook version.

Type: String

Required: No

### MaxConcurrency

The maximum number of targets allowed to run this task in parallel. You can specify a number, such as 10, or a percentage, such as 10%.

Type: String

Required: No

### MaxErrors

The number of errors that are allowed before the system stops running the automation on additional targets. 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 running the automation when the fourth error is received. If you specify 0, then the system stops running the automation on additional targets after the first error result is returned. If you run an automation on 50 resources and set `MaxErrors` to 10%, then the system stops running the automation on additional targets when the sixth error is received.

Automations that are already running when the `MaxErrors` threshold is reached are allowed to complete, but some of these automations may fail as well. If you need to ensure that there won't be more failed automations than the specified `MaxErrors`, set `MaxConcurrency` to 1 so the automations proceed one at a time.

Type: String

Required: No

RuntimeParameters

Required parameters for the secondary runbook. The mapping uses the following format:

{"parameter1" : "value1", "parameter2" : "value2" }

Type: Map

Required: No

Tags

Optional metadata that you assign to a resource. You can specify a maximum of five tags for an automation.

Type: MapList

Required: No

TargetLocations

A location is a combination of AWS Regions and/or AWS accounts where you want to run the automation. A minimum number of 1 item must be specified and a maximum number of 100 items can be specified.

Type: MapList

Required: No

TargetMaps

A list of key-value mappings of document parameters to target resources. Both Targets and TargetMaps can't be specified together.

Type: MapList

Required: No

TargetParameterName

The name of the parameter used as the target resource for the rate-controlled automation. Required if you specify Targets.

Type: String

Required: No

Targets

A list of key-value mappings to target resources. Required if you specify TargetParameterName.

Type: MapList

Required: No

## Output

Output

The output generated by the secondary automation. You can reference the output by using the following format: *Secondary\_Automation\_Step\_Name*.Output

Type: StringList

Here is an example:

```
- 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 ID of the secondary automation.

Type: String

#### Status

The status of the secondary automation.

Type: String

## aws : executeAwsApi – Call and run AWS API operations

Calls and runs AWS API operations. Most API operations are supported, although not all API operations have been tested. For example, the following API operations are supported: [CreateImage](#), [DeleteBucket](#), [RebootDBInstance](#), and [CreateGroups](#). Streaming API operations, such as the [GetObject](#) operation, aren't supported. Each aws:executeAwsApi action can run up to a maximum duration of 25 seconds. For more information and examples of how to use this action, see [Invoking other AWS services from a Systems Manager Automation runbook \(p. 565\)](#).

#### Inputs

Inputs are defined by the API operation that you choose.

#### YAML

```
action: aws:executeAwsApi
inputs:
 Service: The official namespace of the service
 Api: The API operation or method name
 API operation 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

```
{
 "action": "aws:executeAwsApi",
 "inputs": {
 "Service": "The official namespace of the service",
 "Api": "The API operation or method name",
 "API operation 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 operation that you want to run. You can view a list of supported AWS service namespaces in [Available services](#) of the AWS SDK for Python (Boto3). The namespace can be found in the **Client** section. For example, the namespace for Systems Manager is `ssm`. The namespace for Amazon Elastic Compute Cloud (Amazon EC2) is `ec2`.

Type: String

Required: Yes

#### Api

The name of the API operation that you want to run. You can view the API operations (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 operations (methods) for Amazon Relational Database Service (Amazon RDS) are listed on the following page: [Amazon RDS methods](#).

Type: String

Required: Yes

#### API operation inputs

One or more API operation 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 operation 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 operation name",
 "API input 1":"A value",
 "API Input 2":"A value",
 "API Input 3":"A value"
}
```

Type: Determined by chosen API operation

Required: Yes

#### Outputs

Outputs are specified by the user based on the response from the chosen API operation.

##### 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. Each aws:executeScript action can run up to a maximum duration of 600 seconds (10 minutes). You can limit the timeout by specifying the `timeoutSeconds` parameter for an aws:executeScript step.

Use return statements in your function to add outputs to your output payload. For examples of defining outputs for your aws:executeScript action, see [Example 2: Scripted runbook \(p. 622\)](#). You can also send the output from aws:executeScript actions in your runbooks to the Amazon CloudWatch Logs log group you specify. For more information, see [Logging Automation action output with CloudWatch Logs \(p. 1445\)](#).

The aws:executeScript action contains the following preinstalled PowerShell Core modules:

- Microsoft.PowerShell.Host
- Microsoft.PowerShell.Management
- Microsoft.PowerShell.Security
- Microsoft.PowerShell.Utility
- PackageManagement
- PowerShellGet

To use PowerShell Core modules that aren't preinstalled, your script must install the module with the `-Force` flag, as shown in the following command. The `AWSPowerShell.NetCore` module isn't supported. Replace `ModuleName` with the module you want to install.

```
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. Replace each `example resource placeholder` with your own information.

- Amazon S3 cmdlets.

```
Install-Module AWS.Tools.S3 -Force
Get-S3Bucket -BucketName bucketname
```

- Amazon EC2 cmdlets.

```
Install-Module AWS.Tools.EC2 -Force
Get-EC2InstanceState -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 = "Tag"
```

```
$tag.Value = "TagValue"

New-EC2Tag -Resource i-02573cafefEXAMPLE -Tag $tag
```

For examples of installing and importing AWS.Tools modules, and using PowerShell Core cmdlets in runbooks, see [Walkthrough: Using Document Builder to create a custom runbook \(p. 648\)](#).

### Input

Provide the information required to run your script. Replace each *example resource placeholder* with your own information.

#### YAML

```
action: "aws:executeScript"
inputs:
 Runtime: runtime
 Handler: "functionName"
 InputPayload:
 scriptInput: '{{parameterValue}}'
 Script: |-
 def functionName(events, context):
 ...
 Attachment: "scriptAttachment.zip"
```

#### JSON

```
{
 "action": "aws:executeScript",
 "inputs": {
 "Runtime": "runtime",
 "Handler": "functionName",
 "InputPayload": {
 "scriptInputparameterValue}}"
 },
 "Attachment": "scriptAttachment.zip"
 }
}
```

#### Runtime

The runtime language to be used for running the provided script. aws:executeScript supports Python 3.6 (python3.6), Python 3.7 (python3.7), Python 3.8 (python3.8), PowerShell Core 6.0 (dotnetcore2.1), and PowerShell 7.0 (dotnetcore3.1) scripts.

Supported values: **python3.6 | python3.7 | python3.8 | PowerShell Core 6.0 | PowerShell 7.0**

Type: String

Required: Yes

#### Handler

The name of your function. You must ensure the function defined in the handler has two parameters, events and context. The PowerShell runtime does not support this parameter.

Type: String

Required: Yes (Python) | Not supported (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. This parameter is not supported for JSON runbooks. JSON runbooks must provide script content using the `Attachment` input parameter.

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. Specify the same value as the `Name` of the document attachment file you specify in the `Attachments` request parameter. For more information, see [Attachments](#) in the AWS Systems Manager API Reference. If you're providing a script using an attachment, you must also define a `files` section in the top-level elements your runbook. For more information, see [Schema version 0.3 \(p. 1305\)](#).

To invoke a file for Python, use the `filename.method_name` format in `Handler`. For PowerShell, invoke the attachment using an inline script. Gzip isn't supported.

When including Python libraries in your attachment, we recommend adding an empty `__init__.py` file in each module directory. This allows you to import the modules from the library in your attachment within your script content. For example: `from library import module`

Type: String

Required: No

### Output

#### Payload

The JSON representation of the object returned by your function. Up to 100KB is returned.

## aws : executeStateMachine – Run an AWS Step Functions state machine

Runs an AWS Step Functions state machine.

### Input

This action supports most parameters for the Step Functions [StartExecution](#) API operation.

#### Required AWS Identity and Access Management (IAM) permissions

- `states:DescribeExecution`
- `states:StartExecution`

- `states:StopExecution`

YAML

```
name: executeTheStateMachine
action: aws:executeStateMachine
inputs:
 stateMachineArn: StateMachine_ARN
 input: '{"parameters": "values"}'
 name: name
```

JSON

```
{
 "name": "executeTheStateMachine",
 "action": "aws:executeStateMachine",
 "inputs": {
 "stateMachineArn": "StateMachine_ARN",
 "input": "{\"parameters\": \"values\"}",
 "name": "name"
 }
}
```

`stateMachineArn`

The Amazon Resource Name (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:invokeWebhook – Invoke an Automation webhook integration

Invokes the specified Automation webhook integration. For information about creating Automation integrations, see [Creating integrations for Automation \(p. 562\)](#).

### Note

To use the `aws:invokeWebhook` action, your AWS Identity and Access Management (IAM) user or service role must allow the following actions:

- `ssm:GetParameter`

- kms:Decrypt

Permission for the AWS Key Management Service (AWS KMS) Decrypt operation is only required if you use a customer managed key to encrypt the parameter for your integration.

## Input

Provide the information for the Automation integration you want to invoke.

### YAML

```
action: "aws:invokeWebhook"
inputs:
 IntegrationName: "exampleIntegration"
 Body: "Request body"
```

### JSON

```
{
 "action": "aws:invokeWebhook",
 "inputs": {
 "IntegrationName": "exampleIntegration",
 "Body": "Request body"
 }
}
```

#### IntegrationName

The name of the Automation integration. For example, `exampleIntegration`. The integration you specify must already exist.

Type: String

Required: Yes

#### Body

The payload you want to send when your webhook integration is invoked.

Type: String

Required: No

## Output

### Response

The text received from the webhook provider response.

### ResponseCode

The HTTP status code received from the webhook provider response.

## aws:invokeLambdaFunction – Invoke an AWS Lambda function

Invokes the specified AWS Lambda function.

**Note**

Each `aws:invokeLambdaFunction` action can run up to a maximum duration of 300 seconds (5 minutes). You can limit the timeout by specifying the `timeoutSeconds` parameter for an `aws:invokeLambdaFunction` step.

**Input**

This action supports most invoked parameters for the Lambda service. For more information, see [Invoke](#).

**YAML**

```
name: invokeMyLambdaFunction
action: aws:invokeLambdaFunction
maxAttempts: 3
timeoutSeconds: 120
onFailure: Abort
inputs:
 FunctionName: MyLambdaFunction
```

**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 value is `RequestResponse`.

Type: String

Valid values: `Event` | `RequestResponse` | `DryRun`

Required: No

**LogType**

If the default value is `Tail`, the invocation type must be `RequestResponse`. 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 HTTP 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 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. Up to 200KB is returned

The following is a portion from the AWS-PatchInstanceWithRollback runbook demonstrating how to reference outputs from the aws:invokeLambdaFunction action.

YAML

```
- name: IdentifyRootVolume
 action: aws:invokeLambdaFunction
 inputs:
 FunctionName: "IdentifyRootVolumeLambda-{{automation:EXECUTION_ID}}"
 Payload: '{"InstanceId": "{{InstanceId}}"}'
- name: PrePatchSnapshot
 action: aws:executeAutomation
 inputs:
 DocumentName: "AWS-CreateSnapshot"
 RuntimeParameters:
 VolumeId: "{{IdentifyRootVolume.Payload}}"
 Description: "ApplyPatchBaseline restoration case contingency"
```

JSON

```
{
```

```
"name": "IdentifyRootVolume",
"action": "aws:invokeLambdaFunction",
"inputs": {
 "FunctionName": "IdentifyRootVolumeLambda-{{automation:EXECUTION_ID}}",
 "Payload": "{\"InstanceId\": \"{{InstanceId}}\"}"
},
{
 "name": "PrePatchSnapshot",
 "action": "aws:executeAutomation",
 "inputs": {
 "DocumentName": "AWS-CreateSnapshot",
 "RuntimeParameters": {
 "VolumeId": "{{IdentifyRootVolume.Payload}}",
 "Description": "ApplyPatchBaseline restoration case contingency"
 }
 }
}
```

## aws : pause – Pause an automation

This action pauses the automation. Once paused, the automation status is *Waiting*. To continue the automation, use the [SendAutomationSignal](#) API operation with the `Resume` signal type. We recommend using the `aws:sleep` or `aws:approve` action for more granular control of your workflows.

### Input

The input is as follows.

#### YAML

```
name: pauseThis
action: aws:pause
inputs: {}
```

#### 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 AWS Systems Manager Run Command action. A runbook 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

```
- name: checkMembership
 action: 'aws:runCommand'
 inputs:
 DocumentName: AWS-RunPowerShellScript
 InstanceIds:
 - '{{InstanceIds}}'
 Parameters:
 commands:
 - (Get-WmiObject -Class Win32_ComputerSystem).PartOfDomain
```

### JSON

```
{
 "name": "checkMembership",
 "action": "aws:runCommand",
 "inputs": {
 "DocumentName": "AWS-RunPowerShellScript",
 "InstanceIds": [
 "{{InstanceIds}}"
],
 "Parameters": {
 "commands": [
 "(Get-WmiObject -Class Win32_ComputerSystem).PartOfDomain"
]
 }
 }
}
```

#### DocumentName

If the Command type document is owned by you or AWS, specify the name of the document. If you're using a document shared with you by a different AWS account, specify the Amazon Resource Name (ARN) of the document. For more information about using shared documents, see [Using shared SSM documents \(p. 1369\)](#).

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 run the command on all instances in the target group. For more information about pseudo parameters, see [About pseudo parameters \(p. 751\)](#).

Another alternative is to send commands to a fleet of instances by using the Targets parameter. The Targets parameter accepts Amazon Elastic Compute Cloud (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 \(p. 1003\)](#).

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 \(p. 1003\)](#).

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.)

Following is an example.

#### 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

```
{
 "name": "checkMembership",
 "action": "aws:runCommand",
 "inputs": {
 "DocumentName": "AWS-RunPowerShellScript",
 "Targets": [
 {
 "Key": "tag:Stage",
 "Values": [
 "Gamma", "Beta"
]
 },
 {
 "Key": "tag:Application",
 "Values": [
 "Suite"
]
 }
],
 "Parameters": {
 "commands": [
 "(Get-WmiObject -Class Win32_ComputerSystem).PartOfDomain"
]
 }
 }
}
```

## 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 \(p. 1447\)](#).

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

Following is an example.

#### 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

```
{
 "name": "checkMembership",
 "action": "aws:runCommand",
 "inputs": {
 "DocumentName": "AWS-RunPowerShellScript",
 "InstanceIds": [
 "{{InstanceIds}}"
],
 "Parameters": {
 "commands": [
 "(Get-WmiObject -Class Win32_ComputerSystem).PartOfDomain"
]
 },
 "CloudWatchOutputConfig" : {
 "CloudWatchLogGroupName": "CloudWatchGroupForSSMAutomationService",
 "CloudWatchOutputEnabled": true
 }
 }
}
```

## 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 output responses.

Type: String

Required: No

**OutputS3KeyPrefix**

The prefix.

Type: String

Required: No

**ServiceRoleArn**

The ARN of the AWS Identity and Access Management (IAM) role.

Type: String

Required: No

**TimeoutSeconds**

The amount of time in seconds to wait for a command to deliver to the AWS Systems Manager SSM Agent on an instance. If the command isn't received by the SSM Agent on the instance before the value specified is reached, then the status of the command changes to `Delivery Timed Out`.

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 Amazon EC2 instance

Launches a new Amazon Elastic Compute Cloud (Amazon EC2) instance.

### Input

The action supports most API parameters. For more information, see the [RunInstances API documentation](#).

### 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

```
{
 "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"
 }
]
 }
]
 }
}
```

```
 "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 isn't 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**

Turns on or turns off instance API termination.

Type: Boolean

Required: No

**EbsOptimized**

Turns on or turns off Amazon Elastic Block Store (Amazon EBS) optimization.

Type: Boolean

Required: No

**IamInstanceProfileArn**

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (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.

Type: String

Required: No

**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**

Turns on or turns off 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 \(p. 501\)](#) action.

Type: MapList (For more information, see [TagSpecification](#).)

Required: No

### UserData

A 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.

### InstanceState

The current state of the instance.

## aws:sleep – Delay an automation

Delays an automation 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 an automation 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 an automation until a specified date and time. If the specified date and time has passed, the action proceeds immediately.

YAML

```
name: sleep
action: aws:sleep
inputs:
 Timestamp: '2020-01-01T01:00:00Z'
```

JSON

```
{
 "name": "sleep",
 "action": "aws:sleep",
 "inputs": {
 "Timestamp": "2020-01-01T01:00:00Z"
 }
}
```

### Note

Automation 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 allows your automation to wait for a specific resource state or event state before continuing the automation. For more information and examples of how to use this action, see [Invoking other AWS services from a Systems Manager Automation runbook \(p. 565\)](#).

### 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. For more information and examples of how to use this action, see [Handling timeouts in runbooks \(p. 564\)](#).

### Input

Inputs are defined by the API operation that you choose.

#### YAML

```
action: aws:waitForAwsResourceProperty
inputs:
 Service: The official namespace of the service
 Api: The API operation or method name
 API operation 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 operation or method name",
 "API operation inputs or parameters": "A value",
 "PropertySelector": "Response object",
 "DesiredValues": [
 "Desired property value"
]
 }
}
```

#### Service

The AWS service namespace that contains the API operation that you want to run. For example, the namespace for AWS Systems Manager is `ssm`. The namespace for Amazon Elastic Compute Cloud (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 operation that you want to run. You can view the API operations (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 operations (methods) for Amazon Relational Database Service (Amazon RDS) are listed on the following page: [Amazon RDS methods](#).

Type: String

Required: Yes

#### API operation inputs

One or more API operation 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 operation 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 operation name",
 "API input 1": "A value",
 "API Input 2": "A value",
 "API Input 3": "A value"
}
```

Type: Determined by chosen API operation

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.

Type: Varies

Required: Yes

## Automation system variables

AWS Systems Manager Automation runbooks use the following variables. For an example of how these variables are used, view the JSON source of the `AWS-UpdateWindowsAmi` runbook.

### To view the JSON source of the `AWS-UpdateWindowsAmi` runbook

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 runbook `AWS-UpdateWindowsAmi`.
4. Choose the **Content** tab.

### System variables

Automation runbooks support the following system variables.

Variable	Details
<code>global:ACCOUNT_ID</code>	The AWS account ID of the AWS Identity and Access Management (IAM) user or role in which Automation runs.
<code>global:DATE</code>	The date (at run time) in the format yyyy-MM-dd.
<code>global:DATE_TIME</code>	The date and time (at run time) in the format yyyy-MM-dd_HH.mm.ss.
<code>global:AWS_PARTITION</code>	The partition that the resource is in. For standard AWS Regions, the partition is aws. For resources in other partitions, the partition is <code>aws-partitionname</code> . For example, the partition for resources in the AWS GovCloud (US-West) Region is <code>aws-us-gov</code> .
<code>global:REGION</code>	The Region that the runbook is run in. For example, us-east-2.

### Automation variables

Automation runbooks support the following automation variables.

Variable	Details
<code>automation:EXECUTION_ID</code>	The unique identifier assigned to the current automation. For example, <code>1a2b3c-1a2b3c-1a2b3c-1a2b3c1a2b3c1a2b3c</code> .

### Topics

- [Terminology \(p. 532\)](#)
- [Supported scenarios \(p. 534\)](#)
- [Unsupported scenarios \(p. 536\)](#)

## Terminology

The following terms describe how variables and parameters are resolved.

Term	Definition	Example
Constant ARN	A valid Amazon Resource Name (ARN) without variables.	arn:aws:iam::123456789012:role/ roleName
Runbook parameter	A parameter defined at the runbook level (for example, <code>instanceId</code> ). The parameter is used in a basic string replace. Its value is supplied at Start Execution time.	<pre>{     "description": "Create Image Demo",     "version": "0.3",     "assumeRole": "Your_Automation_Assume_Role_ARN",     "parameters": {         "instanceId": {             "type": "String",             "description": "Instance to create image from"         }     } }</pre>
System variable	A general variable substituted into the runbook when any part of the runbook is evaluated.	<pre>"activities": [     {         "id": "copyImage",         "activityType": "AWS-CopyImage",         "maxAttempts": 1,         "onFailure": "Continue",         "inputs": {             "ImageName": "{{imageName}}",             "SourceImageId": "{{sourceImageId}}",             "SourceRegion": "{{sourceRegion}}",             "Encrypted": true,             "ImageDescription": "Test CopyImage Description created on {{global:DATE}}"         }     } ]</pre>
Automation variable	A variable relating to the automation substituted into the runbook when any part of the runbook is evaluated.	<pre>{     "name": "runFixedCmds",     "action": "aws:runCommand",     "maxAttempts": 1,     "onFailure": "Continue",     "inputs": {         "DocumentName": "AWS-RunPowerShellScript",         "InstanceIds": [             "{{LaunchInstance.InstanceIds}}"         ],         "Parameters": {             "Parameter": "Value"         }     } }</pre>

Term	Definition	Example
		<pre> "commands": [   "dir",   "date",    "{{{outputFormat}}}" -f   "left","right","{{global:DATE}}","{{auto     ]}   } } } } </pre>
Systems Manager Parameter	<p>A variable defined within AWS Systems Manager Parameter Store. It can't be directly referenced in step input. Permissions might be required to access the parameter.</p>	<pre> description: Launch new Windows test instance schemaVersion: '0.3' assumeRole:   '{{AutomationAssumeRole}}' parameters:   AutomationAssumeRole:     type: String     default: ''     description: &gt;-       (Required) The ARN of the role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager uses your IAM permissions to run this runbook.   LatestAmi:     type: String     default: &gt;-       {{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}}'       ... </pre>

## Supported scenarios

Scenario	Comments	Example
Constant ARN assumeRole at creation.	An authorization check is performed to verify that the calling user is permitted to pass the given assumeRole.	<pre>{   "description": "Test all Automation resolvable parameters",   "schemaVersion": "0.3",    "assumeRole": "<b>arn:aws:iam::123456789012:roleName</b>",   "parameters": {     ...   } }</pre>
Runbook parameter supplied for AssumeRole when the automation is started.	Must be defined in the parameter list of the runbook.	<pre>{   "description": "Test all Automation resolvable parameters",   "schemaVersion": "0.3",    "assumeRole": "{{dynamicARN}}",   "parameters": {     ...   } }</pre>
Value supplied for runbook parameter at start.	Customer supplies the value to use for a parameter. Any inputs supplied at start time need to be defined in the parameter list of the runbook.	<pre>... "parameters": {   "amiId": {     "type": "String",     "default": "<b>ami-12345678</b>",     "description": "list of commands to run as part of first step"   },   ... }</pre> <p>Inputs to Start Automation Execution include : { "amiId" : [ "<b>ami-12345678</b>" ] }</p>
Systems Manager Parameter referenced within runbook content.	The variable exists within the customer's account, or is a publicly accessible parameter, and the AssumeRole for the runbook has access to the variable. A check is performed at create time to confirm the AssumeRole has access. The parameter can't be directly referenced in step input.	<pre>... parameters:   LatestAmi:     type: String     default: &gt;-       {{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'</pre>

Scenario	Comments	Example
		<pre>maxAttempts: 3 timeoutSeconds: 1200 onFailure: Abort inputs:   ImageId:     '{{LatestAmi}}'   ... }</pre>
System variable referenced within step definition	A system variable is substituted into the runbook when the automation is started. The value injected into the runbook 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 don't need to be set in the parameter list of the runbook.	<pre>... "mainSteps": [   {     "name":       "RunSomeCommands",       "action":         "aws:runCommand",         "maxAttempts": 1,         "onFailure":           "Continue",           "inputs": {             "DocumentName":               "AWS:RunPowerShell",               "InstanceIds":                 ["{{LaunchInstance.InstanceIds}}"],                 "Parameters": {                   "commands" : [                     "echo {The time is now {{global:DATE_TIME}}}"                   ]                 }               }             },             ... }</pre>
Automation variable referenced within step definition.	Automation variables don't need to be set in the parameter list of the runbook. The only supported Automation variable is <b>automation:EXECUTION_ID</b> .	<pre>... "mainSteps": [   {     "name":       "invokeLambdaFunction",       "action":         "aws:invokeLambdaFunction",         "maxAttempts": 1,         "onFailure":           "Continue",           "inputs": {             "FunctionName":               "Hello-World- LambdaFunction",              "Payload" :               "{ "executionId" :                 "{{automation:EXECUTION_ID}}"               }             }           ... }</pre>

Scenario	Comments	Example
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 <code>{{stepName.OutputName}}</code> . This syntax can't be used by the customer for runbook parameters. This is resolved when the referring step runs. The parameter isn't listed in the parameters of the runbook.	<pre>... "mainSteps": [   {     "name": "LaunchInstance",     "action": "aws:runInstances",     "maxAttempts": 1,     "onFailure": "Continue",     "inputs": {       "ImageId": "{{amiId}}",       "MinInstanceCount": 1,       "MaxInstanceCount": 2     }   },   {     "name": "changeState",     "action": "aws:changeInstanceState",     "maxAttempts": 1,     "onFailure": "Continue",     "inputs": {       "InstanceIds": ["{{LaunchInstance.InstanceIds}}"],       "DesiredState": "terminated"     }   } ]</pre>

## Unsupported scenarios

Scenario	Comment	Example
Systems Manager Parameter supplied for <code>assumeRole</code> at create	Not supported.	<pre>... {   "description": "Test all Automation resolvable parameters",   "schemaVersion": "0.3",   "assumeRole": "{{ssm:administratorRoleARN}}",   "parameters": {     ...   } }</pre>
Systems Manager Parameter directly referenced in step input.	Returns <code>InvalidDocumentContent</code> exception at create time.	<pre>... mainSteps:   - name: launchInstance</pre>

Scenario	Comment	Example
		<pre> action: 'aws:runInstances' maxAttempts: 3 timeoutSeconds: 1200 onFailure: Abort inputs:   ImageId: '{{ssm:/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-Base}}' ... </pre>
Variable step definition	The definition of a step in the runbook is constructed by variables.	<pre> ... "mainSteps": [   {     "name": "LaunchInstance",     "action": "aws:runInstances",     "{{attemptModel}}": 1,     "onFailure": "Continue",     "inputs": {       "ImageId": "ami-12345678",       "MinInstanceCount": 1,       "MaxInstanceCount": 2     } ... User supplies input : {   "attemptModel" : "minAttempts" } </pre>
Cross referencing runbook parameters	The user supplies an input parameter at start time, which is a reference to another parameter in the runbook.	<pre> ... "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}}" }, ... </pre>

Scenario	Comment	Example
Multi-level expansion	The runbook 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.	<pre>... "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}"</pre>

Scenario	Comment	Example
Referencing output from a runbook step that is a different variable type	The user references the output from a preceding runbook step within a subsequent step. The output is a variable type that doesn't meet the requirements of the action in the subsequent step.	<pre>... 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.</pre>

## Working with runbooks

### Note

Automation documents are now referred to as runbooks.

An Automation runbook defines the *actions* that Systems Manager performs on your managed instances and other AWS resources when an automation runs. Automation is a capability of AWS Systems Manager. A runbook 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*.

Action types supported for runbooks 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 runbook. (When you create a custom runbook, 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, initiating notifications, and many more.

For a list of all 20 supported action types for Automation, see [Systems Manager Automation actions reference \(p. 477\)](#).

AWS Systems Manager Automation provides several runbooks with pre-defined steps that you can use to perform common tasks like restarting one or more Amazon Elastic Compute Cloud (Amazon EC2) instances or creating an Amazon Machine Image (AMI). You can also create your own runbooks and share them with other AWS accounts, or make them public for all Automation users.

Runbooks are written using YAML or JSON. Using the **Document Builder** in the Systems Manager Automation console, however, you can create a runbook 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 runbooks (AWS-\* runbooks) such as the `AWS-ConfigureS3BucketLogging`, `AWS-CreateDynamoDBBackup`, and `AWS-RestartEC2Instance` runbooks, to name a few. This requirement also applies to any custom Automation runbooks 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, configure the service role with permission to invoke those services. You can give 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 \(p. 405\)](#).

For information about the actions that you can specify in a runbook, see [Systems Manager Automation actions reference \(p. 477\)](#).

For information about the AWS managed runbooks that run scripts, see [AWS managed runbooks that run scripts \(p. 552\)](#).

For information about using the AWS Toolkit for Visual Studio Code to create runbooks, see [Working with Systems Manager Automation documents](#) in the *AWS Toolkit for Visual Studio Code User Guide*.

For information about using Document Builder to create a custom runbook, see [Creating runbooks using Document Builder \(p. 542\)](#).

For information about creating custom runbooks that run scripts, see the following topics:

- [Creating runbooks that run scripts \(p. 545\)](#) – Provides information for using Document Builder to create a runbook that includes the `aws:executeScript` action.
- [Creating a runbook that runs scripts \(command line\) \(p. 549\)](#) – Provides information for using a command line tool to create a runbook that runs a script.
- [Walkthrough: Using Document Builder to create a custom runbook \(p. 648\)](#) – Provides step-by-step guidance for creating a runbook that runs scripts to (1) launch an EC2 instance and (2) wait for the instance status to change to `ok`.

### Contents

- [Creating input parameters that populate AWS resources \(p. 541\)](#)
- [Creating runbooks using Document Builder \(p. 542\)](#)
- [Creating a runbook using the Editor \(p. 545\)](#)
- [Creating runbooks that run scripts \(p. 545\)](#)

- [Creating dynamic automations with conditional branching \(p. 552\)](#)
- [Creating integrations for Automation \(p. 562\)](#)
- [Handling timeouts in runbooks \(p. 564\)](#)
- [Invoking other AWS services from a Systems Manager Automation runbook \(p. 565\)](#)
- [Sample scenarios and custom runbook solutions \(p. 574\)](#)

## Creating input parameters that populate AWS resources

Automation, a capability of Systems Manager, populates AWS resources in the AWS Management Console that match the resource type you define for an input parameter. Resources in your AWS account that match the resource type are displayed in a dropdown list for you to choose. You can define input parameter types for Amazon Elastic Compute Cloud (Amazon EC2) instances, Amazon Simple Storage Service (Amazon S3) buckets, and AWS Identity and Access Management (IAM) roles. The supported type definitions and the regular expressions used to locate matching resources are as follows:

- `AWS::EC2::Instance::Id - ^m?i-[a-z0-9]{8,17}$`
- `List<AWS::EC2::Instance::Id> - ^m?i-[a-z0-9]{8,17}$`
- `AWS::S3::Bucket::Name - ^[0-9a-z][a-z0-9\\-\\.]{3,63}$`
- `List<AWS::S3::Bucket::Name> - ^[0-9a-z][a-z0-9\\-\\.]{3,63}$`
- `AWS::IAM::Role::Arn - ^arn:(aws|aws-cn|aws-us-gov|aws-iso|aws-iso-b):iam:[0-9]{12}:role/.*$`
- `List<AWS::IAM::Role::Arn> - ^arn:(aws|aws-cn|aws-us-gov|aws-iso|aws-iso-b):iam:[0-9]{12}:role/.*$`

The following is an example of input parameter types defined in runbook content.

YAML

```
description: Enables encryption on an Amazon S3 bucket
schemaVersion: '0.3'
assumeRole: '{{ AutomationAssumeRole }}'
parameters:
 BucketName:
 type: 'AWS::S3::Bucket::Name'
 description: (Required) The name of the Amazon S3 bucket you want to encrypt.
 SSEAlgorithm:
 type: String
 description: (Optional) The server-side encryption algorithm to use for the default encryption.
 default: AES256
 AutomationAssumeRole:
 type: 'AWS::IAM::Role::Arn'
 description: (Optional) The Amazon Resource Name (ARN) of the role that allows Automation to perform the actions on your behalf.
 default: ''
mainSteps:
 - name: enableBucketEncryption
 action: 'aws:executeAwsApi'
 inputs:
 Service: s3
 Api: PutBucketEncryption
 Bucket: '{{BucketName}}'
 ServerSideEncryptionConfiguration:
 Rules:
 - ApplyServerSideEncryptionByDefault:
 SSEAlgorithm: '{{SSEAlgorithm}}'
```

```
 isEnd: true
```

JSON

```
{
 "description": "Enables encryption on an Amazon S3 bucket",
 "schemaVersion": "0.3",
 "assumeRole": "{{ AutomationAssumeRole }}",
 "parameters": {
 "BucketName": {
 "type": "AWS::S3::Bucket::Name",
 "description": "(Required) The name of the Amazon S3 bucket you want to encrypt."
 },
 "SSEAlgorithm": {
 "type": "String",
 "description": "(Optional) The server-side encryption algorithm to use for the default encryption.",
 "default": "AES256"
 },
 "AutomationAssumeRole": {
 "type": "AWS::IAM::Role::Arn",
 "description": "(Optional) The Amazon Resource Name (ARN) of the role that allows Automation to perform the actions on your behalf.",
 "default": ""
 }
 },
 "mainSteps": [
 {
 "name": "enableBucketEncryption",
 "action": "aws:executeAwsApi",
 "inputs": {
 "Service": "s3",
 "Api": "PutBucketEncryption",
 "Bucket": "{{BucketName}}",
 "ServerSideEncryptionConfiguration": {
 "Rules": [
 {
 "ApplyServerSideEncryptionByDefault": {
 "SSEAlgorithm": "{{SSEAlgorithm}}"
 }
 }
]
 }
 },
 "isEnd": true
 }
]
}
```

## Creating runbooks using Document Builder

If the AWS Systems Manager public runbooks don't support all the actions you want to perform on your AWS resources, you can create your own runbooks. To create a custom runbook, you can manually create a local YAML or JSON format file with the appropriate automation actions. Alternatively, you can use the Document Builder in the Systems Manager console to build a custom runbook.

Using the Document Builder, you can add automation actions to your custom runbook and provide the required parameters without having to use JSON or YAML syntax. After you add steps and create the runbook, the system converts the actions you've added into the YAML format that Systems Manager can use to run automation.

Runbooks support the use of Markdown, a markup language, which allows you to add wiki-style descriptions to runbooks and individual steps within the runbook. For more information about 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 runbooks that run scripts, see the following topics:

- [Creating runbooks that run scripts \(p. 545\)](#) – Provides information for using Document Builder to create a runbook that includes the aws:executeScript action.
- [Creating a runbook that runs scripts \(command line\) \(p. 549\)](#) – Provides information for using a command line tool to create a runbook that runs a script.
- [Walkthrough: Using Document Builder to create a custom runbook \(p. 648\)](#) – Provides step-by-step guidance for creating a runbook 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 runbook using Document Builder, we recommend that you read about the different actions that you can use within a runbook. For more information, see [Systems Manager Automation actions reference \(p. 477\)](#).

### To create a runbook using Document Builder

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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**, enter a descriptive name for the runbook.
5. For **Document description**, provide the markdown style description for the runbook. You can provide instructions for using the runbook, numbered steps, or any other type of information to describe the runbook. 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 runs the automation.

**Important**

For runbooks not owned by Amazon that use the aws:executeScript action, a role must be specified. For information, see [Permissions for using runbooks \(p. 545\)](#).

7. (Optional) For **Outputs**, enter any outputs for the automation of this runbook to make available for other processes.

For example, if your runbook creates a new AMI, you might specify ["CreateImage.ImageId"], and then use this output to create new instances in a subsequent automation.

8. (Optional) Expand the **Input parameters** section and do the following.
  1. For **Parameter name**, enter a descriptive name for the runbook parameter you're 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 runbook parameter must be supplied at runtime.
  - Choose **No** if the parameter isn't required, and (optional) enter a default parameter value in **Default value**.
4. For **Description**, enter a description for the runbook parameter.

**Note**

To add more runbook parameters, choose **Add a parameter**. To remove a runbook 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 automation can run on. For example, to use a runbook on EC2 instances, choose / AWS::EC2::Instance.

**Note**

If you specify a value of '/', the runbook 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 runbook. Tags make it easier to identify, organize, and search for resources. For more information, see [Tagging Systems Manager documents \(p. 1506\)](#).

11. In the **Step 1** section, provide the following information.

- For **Step name**, enter a descriptive name for the first step of the automation.
- 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 \(p. 477\)](#).

- 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 \(p. 477\)](#) for the action type you selected. For example: [aws:executeStateMachine – Run an AWS Step Functions state machine \(p. 513\)](#).

- (Optional) For **Additional inputs**, provide any additional input values needed for your runbook. 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 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's stopped.

For more information, see [Properties shared by all actions \(p. 478\)](#).

**Note**

To add more steps, select **Add step** and repeat the procedure for creating a step. To remove a step, choose **Remove step**.

12. Choose **Create automation** to save the runbook.

## Creating a runbook using the Editor

If the AWS Systems Manager public runbooks don't perform all the actions you want to perform on your AWS resources, you can create your own runbooks. For example, you can use the editor to modify parameters, add additional steps to an existing runbook, or combine multiple runbooks into a single runbook. If you're familiar with writing your own runbooks in JSON or YAML, you can use the editor to enter the JSON or YAML runbook content.

For examples of custom runbooks, see [Sample scenarios and custom runbook solutions \(p. 574\)](#).

**Note**

If your runbook uses the `aws:executeScript` automation action with the `Attachment` input parameter, you must use the AWS CLI or Document Builder to successfully create the runbook.

The following procedure describes how to use the editor to create a runbook.

### To create a runbook using the editor

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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**, enter a descriptive name for the runbook.
5. Choose the **Editor** tab, and choose **Edit**.
6. Enter the runbook content using JSON or YAML.
7. Choose **Create automation**.

## Creating runbooks that run scripts

Automation runbooks support running scripts as part of the automation. Automation is a capability of AWS Systems Manager. By using runbooks, you can run scripts directly in AWS without creating a separate compute environment to run your scripts. Because runbooks can run script steps along with other automation step types, such as approvals, you can manually intervene in critical or ambiguous situations. You can send the output from `aws:executeScript` actions in your runbooks to Amazon CloudWatch Logs. For more information, see [Logging Automation action output with CloudWatch Logs \(p. 1445\)](#).

### Permissions for using runbooks

To use a runbook, 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 runbooks that don't 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 runbook or passed in as a parameter.

- If no IAM service role is specified, the permissions of the IAM user who started the automation.

When a step in a runbook 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 operations. Automation checks for this role in the following order:

- The permissions of an IAM service role, or Assume role, that is specified in the runbook or passed in as a parameter.
- 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 use a runbook that uses an IAM service role or more advanced forms of delegated administration instead, see [Running an automation by using an IAM service role \(p. 460\)](#).

## Adding scripts to runbooks

You can add scripts to your runbooks by including the script inline as part of a step in the runbook. You can also attach scripts to the runbook 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 is complete, the output of the script is available as a JSON object, which you can then use as input for subsequent steps in your runbook.

## Script constraints for runbooks

The automation action `aws : executeScript` supports running Python 3.6, Python 3.7, and PowerShell Core 6.0 scripts.

Runbooks 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.

The following topics describe how to create runbooks that run scripts.

### Topics

- [Creating a runbook that runs a script \(console\) \(p. 546\)](#)
- [Creating a runbook that runs scripts \(command line\) \(p. 549\)](#)
- [AWS managed runbooks that run scripts \(p. 552\)](#)

## Creating a runbook that runs a script (console)

### To create a runbook that runs a script

The following procedure describes how to use Document Builder in the AWS Systems Manager console to create a custom runbook that runs a script that you provide.

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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**, enter a descriptive name for the runbook.
5. For **Document description**, provide the markdown style description for the runbook. You can provide instructions for using the runbook, numbered steps, or any other type of information to describe the runbook. 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 Amazon Resource Name (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.

**Important**

For runbooks not owned by Amazon that use the `aws:executeScript` action, a role must be specified. For information, see [Permissions for using runbooks \(p. 545\)](#).

7. (Optional) For **Outputs**, enter any outputs for the automation to make available for other processes.

For example, if your runbook creates a new AMI, you might specify `["CreateImage.ImageId"]`, and then use this output to create new instances in a subsequent automation.

8. (Optional) Expand the **Input parameters** section and do the following.

1. For **Parameter name**, enter a descriptive name for the runbook parameter you're 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 runbook parameter must be supplied at runtime.
- Choose **No** if the parameter isn't required, and (optional) enter a default parameter value in **Default value**.

4. For **Description**, enter a description for the runbook parameter.

**Note**

To add more runbook parameters, choose **Add a parameter**. To remove a runbook 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 runbook can run on. For example, to use a runbook on EC2 instances, choose `/AWS::EC2::Instance`.

**Note**

If you specify a value of `'/'`, the runbook 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.
- 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're adding. Automation supports Python 3.6, Python 3.7, and PowerShell Core 6.0.
- For **Handler**, enter 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.

```
def launch_instance(events, context):
 import boto3
 ec2 = boto3.client('ec2')

[...truncated...]
```

- For **Script**, choose a method to provide a script to the runbook.
  - To embed a script within the runbook, 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 navigate 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. 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 runbook. 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 \(p. 567\)](#).

**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's stopped.

For more information, see [Properties shared by all actions \(p. 478\)](#).

**Note**

To add more steps, select **Add step** and repeat the procedure for creating a step. To remove a step, choose **Remove step**.

15. Choose **Create document** to save the runbook.

## Creating a runbook that runs scripts (command line)

The following examples show how to use the AWS CLI or AWS Tools for PowerShell to create an runbook 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 runbook in a YAML-formatted file with a `.yaml` extension, such as `MyAutomationDocument.yaml`.
- The files to attach to the runbook. 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 haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

### Attach a single file from an S3 bucket

Run the following command to create a runbook using a script file stored in an S3 bucket. Replace each `example resource placeholder` with your own information.

Linux & macOS

```
aws ssm create-document \
 --name runbook name \
 --content file://content.yaml \
 --document-format YAML \
 --attachments "Key=S3FileUrl,Name=script.py,Values=https://doc-example-bucket.s3-aws-region.amazonaws.com/filePath" \
 --document-type Automation
```

Windows

```
aws ssm create-document ^
 --name runbook name ^
 --content file://content.yaml ^
 --document-format YAML ^
 --attachments "Key=S3FileUrl,Name=script.py,Values=https://doc-example-bucket.s3-aws-region.amazonaws.com/filePath" ^
 --document-type Automation
```

PowerShell

```
New-SSMDocument `
 -Name runbook name `
 -Content file://content.yaml `
 -DocumentFormat YAML `
 -Attachments @{
 "Key"="S3FileUrl";
 "Name"="script.py";
 "Values"="https://doc-example-bucket.s3-aws-region.amazonaws.com/filePath"
 } `
 -DocumentType Automation
```

### Attach files from an S3 bucket

Run the following command to create a runbook using a script or multiple script files stored in an S3 bucket. A `Name` key for files isn't specified. The command attaches all supported files from the S3 bucket location.

#### Linux & macOS

```
aws ssm create-document \
--name runbook name \
--content file://content.yaml \
--document-format YAML \
--attachments Key=SourceUrl,Values="https://doc-example-bucket.s3-
aws-region.amazonaws.com/filePath" \
--document-type Automation
```

#### Windows

```
aws ssm create-document ^
--name runbook name ^
--content file://content.yaml ^
--document-format YAML ^
--attachments Key=SourceUrl,Values="https://doc-example-bucket.s3-
aws-region.amazonaws.com/filePath" ^
--document-type Automation
```

#### PowerShell

```
New-SSMDocument `
-Name runbook name `
-Content file://content.yaml `
-DocumentFormat YAML `
-Attachments @{
 "Key"="SourceUrl";
 "Values"="https://doc-example-bucket.s3-aws-region.amazonaws.com/filePath"
} `
-DocumentType Automation
```

### Attach files from another runbook in your AWS account

Run the following command to create a runbook using a script that is already attached to another runbook in your account. Replace each `example resource placeholder` with your own information.

The format of the key value in this command is `runbook name/runbook version number/file name`. See the following example.

```
MyRunbook/2/script.py
```

#### Linux & macOS

```
aws ssm create-document \
--name runbook name \
--content file://content.yaml \
--document-format YAML \
--attachments Key=AttachmentReference,Values="runbook name/runbook version number/file name" \
--document-type Automation
```

#### Windows

```
aws ssm create-document ^
```

```
--name runbook name ^
--content file://content.yaml ^
--document-format YAML ^
--attachments Key=AttachmentReference,Values="runbook name/runbook version number/
file name" ^
--document-type Automation
```

#### PowerShell

```
New-SSMDocument `
-Name runbook name `
-Content file://content.yaml `
-DocumentFormat YAML `
-Attachments @{
 "Key"="AttachmentReference";
 "Values"="runbook name/runbook version number/file name"
} `
-DocumentType Automation
```

#### Attach files from an runbook in another AWS account

Run the following command to create a runbook using a script that is already attached to a runbook that has been shared with you from another AWS account. Replace each *example resource placeholder* with your own information.

The format of the key value in this command is *runbook arn/runbook version number/file name*. See the following example.

```
arn:aws:ssm:us-east-2:123456789012:document/OtherAccountDocument/2/script.py
```

#### Linux & macOS

```
aws ssm create-document \
--name runbook name \
--content file://content.yaml \
--document-format YAML \
--attachments Key=AttachmentReference,Values="runbook arn/runbook version number/file
name" \
--document-type Automation
```

#### Windows

```
aws ssm create-document ^
--name runbook name ^
--content file://content.yaml ^
--document-format YAML ^
--attachments Key=AttachmentReference,Values="runbook arn/runbook version number/file
name" ^
--document-type Automation
```

#### PowerShell

```
New-SSMDocument `
-Name runbook name `
-Content file://content.yaml `
-DocumentFormat YAML `
-Attachments @{
```

```
"Key"="AttachmentReference";
"Values"="runbook arn/runbook version number/file name"
}
-DocumentType Automation
```

## AWS managed runbooks that run scripts

Automation runbooks support running scripts as part of the automation. Automation is a capability of AWS Systems Manager.

The following are AWS managed runbooks that include support for running scripts.

### Note

You can also create your own custom runbooks that can run scripts. For information, see [Creating runbooks that run scripts \(p. 545\)](#).

- [AWS-CreateRdsSnapshot](#) – Creates an Amazon Relational Database Service (Amazon RDS) snapshot for an Amazon RDS instance.
- [AWS-CreateServiceNowIncident](#) – Creates an incident in the ServiceNow incident table.
- [AWS-ExportOpsDataToS3](#) – Retrieves a list of OpsData summaries in AWS Systems Manager Explorer and exports them to an object in a specified S3 bucket.
- [AWS-RunCfnLint](#) – Uses an [AWS CloudFormation Linter](#) (cfn-python-lint) to validate YAML and JSON templates against the AWS CloudFormation resource specification. This runbook is using cfn-lint v0.24.4.
- [AWS-RunPacker](#) – Uses the HashiCorp [Packer](#) tool to validate, fix, or build packer templates that are used to create machine images. This runbook is using Packer v1.4.4.

## Creating dynamic automations with conditional branching

By default, the steps that you define in the `mainSteps` section of a runbook 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 fails (by default). You can use the `aws:branch` automation action and the runbook options described in this section to create automations that perform *conditional branching*. This means that you can create automations that jump to a different step after evaluating different choices or that dynamically respond to changes when a step is complete. Here is a list of options that you can use to create dynamic automations:

- **aws:branch**: This automation action allows you to create a dynamic automation that evaluates multiple choices in a single step and then jumps to a different step in the runbook based on the results of that evaluation.
- **nextStep**: This option specifies which step in an automation to process next after successfully completing a step.
- **isEnd**: This option stops an automation 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 automation should stop, 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` options, see [Examples of how to use dynamic options \(p. 560\)](#).

## Working with the aws:branch action

The `aws:branch` action offers the most dynamic conditional branching options for automations. As noted earlier, this action allows your automation 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.

```
- name: ChooseOSforCommands
 action: aws:branch
 inputs:
 Choices:
 - NextStep: runPowerShellCommand
 Variable: "{{GetInstance.platform}}"
 StringEquals: Windows
 - NextStep: runShellCommand
 Variable: "{{GetInstance.platform}}"
 StringEquals: Linux
 Default:
 PostProcessing
```

When you specify the `aws:branch` action for a step, you specify `Choices` that the automation must evaluate. The automation can evaluate `Choices` based on the value of a parameter that you specified in the `Parameters` section of the runbook. The automation can also evaluate `Choices` based on output from a previous step.

The automation evaluates each choice by using a Boolean expression. If the evaluation determines that the first choice is `true`, then the automation jumps to the step designated for that choice. If the evaluation determines that the first choice is `false`, then the automation evaluates the next choice. If your step includes three or more `Choices`, then the automation evaluates each choice in sequential order until it evaluates a choice that is `true`. The automation then jumps to the designated step for the `true` choice.

If none of the `Choices` are `true`, the automation checks to see if the step contains a `Default` value. A `Default` value defines a step that the automation should jump to if none of the choices are `true`. If no `Default` value is specified for the step, then the automation processes the next step in the runbook.

Here is an `aws:branch` step in YAML named `chooseOSfromParameter`. The step includes two `Choices`: (`NextStep: runWindowsCommand`) and (`NextStep: runLinuxCommand`). The automation 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 runbook author defined in the `Parameters` section of the runbook.

```
mainSteps:
- name: chooseOSfromParameter
 action: aws:branch
 inputs:
 Choices:
 - NextStep: runWindowsCommand
 Variable: "{{OSName}}"
 StringEquals: Windows
 - NextStep: runLinuxCommand
 Variable: "{{OSName}}"
 StringEquals: Linux
```

Here is an `aws:branch` step in YAML named `chooseOSfromOutput`. The step includes two `Choices`: (`NextStep: runPowerShellCommand`) and (`NextStep: runShellCommand`). The automation 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 runbook. This example also includes an option called `Default`. If the automation evaluates both `Choices`, and neither choice is `true`, then the automation jumps to a step called `PostProcessing`.

```
mainSteps:
- name: chooseOSfromOutput
 action: aws:branch
 inputs:
 Choices:
 - NextStep: runPowerShellCommand
 Variable: "{{GetInstance.platform}}"
 StringEquals: Windows
 - NextStep: runShellCommand
 Variable: "{{GetInstance.platform}}"
 StringEquals: Linux
 Default:
 PostProcessing
```

### Creating an `aws:branch` step in a runbook

When you create an `aws:branch` step in a runbook, you define the `Choices` the automation should evaluate to determine which step the automation 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 runbook 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 runbook, or specify an output object from a previous step in the runbook.

Specify parameter variables by using the following form.

`Variable: "{{parameter name}}"`

Specify output object variables by using the following form.

`Variable: "{{previousStepName.outputName}}"`

#### Note

Creating the output variable is described in more detail in the next section, [About creating the output variable \(p. 556\)](#).

- **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 a runbook, the system validates each operation in the runbook. If an operation isn't supported, the system returns an error when you try to create the runbook.

- **Default:** Specify a fallback step that the automation 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` option. If none of the Choices are true and no Default value is specified, then the automation stops at the end of the step.

Use the following templates to help you construct the `aws:branch` step in your runbook. Replace each *example resource placeholder* with your own information.

### YAML

```
mainSteps:
- name: step name
 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 for this choice is true
 Variable: "{{parameter name or output from previous step}}"
 Operation type: Operation value
 Default:
 step to jump to if all choices are false
```

### 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 for this choice is 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.outputName}{}{}`"

For example, the first step in the following example 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}{}{}`".

```
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 an example that shows how "`Variable": "{}{ describeInstance.Platform }{}{}`" is created from the previous step and the output.

```
- 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
 Default: runEC2RescueForLinux
```

### Example aws:branch runbooks

Here are some example runbooks 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 example (GetInstance), the runbook author uses the aws:executeAwsApi action to call the ssm DescribeInstanceInformation API operation. 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 evaluates the operating system of the instance by using the output from the previous step (Variable: "{{GetInstance.platform}}"). The automation jumps to a step for the designated operating system.

```

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:
 commands:
 - ls
```

```

 isEnd: true
- name: runPowerShellCommand
 action: aws:runCommand
 inputs:
 DocumentName: AWS-RunPowerShellScript
 InstanceIds:
 - "{{GetInstance.myInstance}}"
 Parameters:
 commands:
 - ls
 isEnd: true
- name: Sleep
 action: aws:sleep
 inputs:
 Duration: PT3S

```

**Example 2: Using aws:branch with a parameter variable to run commands based on the operating system type**

The runbook author defines several parameter options at the beginning of the runbook 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 runbook, they specify a value at runtime for OperatingSystemName. The automation uses the runtime parameter during the Choices evaluation. The automation jumps to a step for the designated operating system based on the runtime parameter specified for OperatingSystemName.

```

schemaVersion: '0.3'
assumeRole: "{{AutomationAssumeRole}}"
parameters:
 AutomationAssumeRole:
 default: ""
 type: String
 OperatingSystemName:
 type: String
 LinuxInstanceId:
 type: String
 WindowsInstanceId:
 type: String
mainSteps:
- name: ChooseOS
 action: aws:branch
 inputs:
 Choices:
 - NextStep: runWindowsCommand
 Variable: "{{OperatingSystemName}}"
 StringEquals: windows
 - NextStep: runLinuxCommand
 Variable: "{{OperatingSystemName}}"
 StringEquals: linux
 Default:
 Sleep
- name: runLinuxCommand
 action: aws:runCommand
 inputs:
 DocumentName: "AWS-RunShellScript"
 InstanceIds:
 - "{{LinuxInstanceId}}"
 Parameters:
 commands:
 - ls

```

```

 isEnd: true
- name: runWindowsCommand
 action: aws:runCommand
 inputs:
 DocumentName: "AWS-RunPowerShellScript"
 InstanceIds:
 - "{{WindowsInstanceId}}"
 Parameters:
 commands:
 - date
 isEnd: true
- name: Sleep
 action: aws:sleep
 inputs:
 Duration: PT3S

```

### Creating complex branching automations with operators

You can create complex branching automations 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 jumps to the `runPowerShellCommand` step. If one or more of the variables is `false`, then the automation evaluates the variables for the second choice.

```

mainSteps:
- name: switch2
 action: aws:branch
 inputs:
 Choices:
 - And:
 - Variable: "{{GetInstance.pingStatus}}"
 StringEquals: running
 - Variable: "{{GetInstance.platform}}"
 StringEquals: Windows
 NextStep: runPowerShellCommand

 - And:
 - Variable: "{{GetInstance.pingStatus}}"
 StringEquals: running
 - Variable: "{{GetInstance.platform}}"
 StringEquals: Linux
 NextStep: runShellCommand
 Default:
 sleep3

```

#### 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 jumps to the `RunPowerShellCommand` step. If both variables are `false`, then the automation evaluates the variables for the second choice.

```

- Or:
 - Variable: "{{parameter1}}"
 StringEquals: Windows
 - Variable: "{{BooleanParam1}}"
 BooleanEquals: true

```

```

 NextStep: RunPowershellCommand
- Or:
 - Variable: "{{parameter2}}"
 StringEquals: Linux
 - Variable: "{{BooleanParam2}}"
 BooleanEquals: true
 NextStep: RunShellScript

```

### 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 isn't Linux, then the automation jumps to the `sleep2` step. If the evaluation of the first choice determines that it *is* Linux, then the automation evaluates the next choice.

```

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 options

This section includes different examples of how to use dynamic options in a runbook. Each example in this section extends the following runbook. This runbook 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 didn't install successfully, the automation stops before step two.

### Example 1: Runbook with two linear actions

```

schemaVersion: '0.3'
description: Install MSI package and run validation.
assumeRole: "{{automationAssumeRole}}"
parameters:
 automationAssumeRole:
 type: String
 description: "(Required) Assume role."
 packageName:
 type: String
 description: "(Required) MSI package to be installed."
 instanceIds:
 type: String
 description: "(Required) Comma separated list of instances."
mainSteps:
- name: InstallMsiPackage
 action: aws:runCommand
 maxAttempts: 2
 onFailure: Abort
 inputs:

```

```

InstanceIds:
- "{{instanceIds}}"
DocumentName: AWS-RunPowerShellScript
Parameters:
 commands:
 - msieexec /i {{packageName}}
- name: TestInstall
 action: aws:invokeLambdaFunction
 maxAttempts: 1
 timeoutSeconds: 500
 inputs:
 FunctionName: TestLambdaFunction
...

```

### Creating a dynamic automation 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. With this example, if the `InstallMsiPackage` action fails, then the automation 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 automation jumps to the `TestInstall` action (`nextStep: TestInstall`). Both the `TestInstall` and the `PostFailure` steps use the `isEnd` option (`isEnd: true`) so that the automation finishes 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 doesn't jump to other steps, then the automation stops after running the action in the last step.

### Example 2: A dynamic automation that jumps to different steps

```

mainSteps
- name: InstallMsiPackage
 action: aws:runCommand
 onFailure: step:PostFailure
 maxAttempts: 2
 inputs:
 InstanceIds:
 - "{{instanceIds}}"
 DocumentName: AWS-RunPowerShellScript
 Parameters:
 commands:
 - msieexec /i {{packageName}}
 nextStep: TestInstall
- name: TestInstall
 action: aws:invokeLambdaFunction
 maxAttempts: 1
 timeoutSeconds: 500
 inputs:
 FunctionName: TestLambdaFunction
 isEnd: true
- name: PostFailure
 action: aws:invokeLambdaFunction
 maxAttempts: 1
 timeoutSeconds: 500
 inputs:
 FunctionName: PostFailureRecoveryLambdaFunction
 isEnd: true
...

```

#### Note

Before processing a runbook, the system verifies that the runbook doesn't 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 automation that defines critical steps

You can specify that a step is critical for the overall success of the automation. If a critical step fails, then Automation reports the status of the automation 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 isn't 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

```

name: InstallMsiPackage
action: aws:runCommand
onFailure: step:VerifyDependencies
isCritical: false
maxAttempts: 2
inputs:
 InstanceIds:
 - "{{instanceIds}}"
 DocumentName: AWS-RunPowerShellScript
 Parameters:
 commands:
 - msieexec /i {{packageName}}
nextStep: TestPackage
...
```

## Creating integrations for Automation

To send messages using webhooks during an automation, create an integration. Integrations can be invoked during an automation by using the `aws:invokeWebhook` action in your runbook. If you haven't already created a webhook, see [Creating webhooks for integrations \(p. 563\)](#). To learn more about the `aws:invokeWebhook` action, see [aws:invokeWebhook – Invoke an Automation webhook integration \(p. 514\)](#).

As shown in the following procedures, you can create an integration by using either the Systems Manager Automation console or your preferred command line tool.

### Creating integrations (console)

#### To create an integration for Automation (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**.
3. Choose the **Integrations** tab.
4. Select **Add integration**, and choose **Webhook**.
5. Enter the required values and any optional values you want to include for the integration.
6. Choose **Add** to create the integration.

### Creating integrations (command line)

To create an integration using command line tools, you must create the required `SecureString` parameter for an integration. Automation uses a reserved namespace in Parameter Store, a capability of Systems Manager, to store information about your integration. If you create an integration using the AWS Management Console, Automation handles this process for you. Following the namespace, you must specify the type of integration you want to create and then the name of your integration. Currently, Automation supports webhook type integrations.

The supported fields for webhook type integrations are as follows:

- Description
- headers
- payload
- URL

### Before you begin

If you haven't already, install and configure the AWS Command Line Interface (AWS CLI) or the AWS Tools for PowerShell. For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

### To create an integration for Automation (command line)

- Run the following commands to create the required SecureString parameter for an integration. Replace each *example resource placeholder* with your own information. The /d9d01087-4a3f-49e0-b0b4-d568d7826553/ssm/integrations/webhook/ namespace is reserved in Parameter Store for integrations. The name of your parameter must use this namespace followed by the name of your integration. For example /d9d01087-4a3f-49e0-b0b4-d568d7826553/ssm/integrations/webhook/*myWebhookIntegration*.

Linux & macOS

```
aws ssm put-parameter \
--name "/d9d01087-4a3f-49e0-b0b4-d568d7826553/ssm/integrations/
webhook/myWebhookIntegration" \
--type "SecureString" \
--data-type "aws:ssm:integration" \
--value "{"description": 'My first webhook integration for Automation.', "url": 'myWebHookURL'}"
```

Windows

```
aws ssm put-parameter ^
--name "/d9d01087-4a3f-49e0-b0b4-d568d7826553/ssm/integrations/
webhook/myWebhookIntegration" ^
--type "SecureString" ^
--data-type "aws:ssm:integration" ^
--value "{"description": 'My first webhook integration for Automation.', "url": 'myWebHookURL'}"
```

PowerShell

```
Write-SMSPParameter `
-Name "/d9d01087-4a3f-49e0-b0b4-d568d7826553/ssm/integrations/
webhook/myWebhookIntegration" `
-Type "SecureString"
-DataType "aws:ssm:integration"
-Value "{"description": 'My first webhook integration for Automation.', "url": 'myWebHookURL'}"
```

## Creating webhooks for integrations

When creating webhooks with your provider, note the following:

- Protocol must be HTTPS.

- Custom request headers are supported.
- A default request body can be specified.
- The default request body can be overridden when an integration is invoked by using the `aws:invokeWebhook` action.

## Handling timeouts in runbooks

The `timeoutSeconds` property is shared by all automation actions. You can use this property to specify the execution timeout value for an action. Further, you can change how an action timing out affects the automation and overall execution status. You can accomplish this by also defining the `onFailure` and `isCritical` shared properties for an action.

For example, depending on your use case, you might want your automation to continue to a different action and not affect the overall status of the automation if an action times out. In this example, you specify the length of time to wait before the action times out using the `timeoutSeconds` property. Then you specify the action, or step, the automation should go to if there is a timeout. Specify a value using the format `step:step name` for the `onFailure` property rather than the default value of `Abort`. By default, if an action times out, the automation execution status will be `Timed Out`. To prevent a timeout from affecting the automation execution status, specify `false` for the `isCritical` property.

The following example shows how to define the shared properties for an action described in this scenario.

YAML

```
- 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
```

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 \(p. 478\)](#).

## Invoking other AWS services from a Systems Manager Automation runbook

You can invoke other AWS services and other Systems Manager capabilities by using the following automation actions in your runbooks:

- **aws:executeAwsApi**: This automation action calls and runs AWS API operations. Most API operations are supported, although not all API operations have been tested. For example, the following API operations are supported: [CreateImage](#), [Delete bucket](#), [RebootDBInstance](#), and [CreateGroups](#). Streaming API operations, such as the [Get Object](#) action, aren't supported.
- **aws:waitForAwsResourceProperty**: This automation action allows your automation to wait for a specific resource state or event state before continuing the automation. For example, you can use this action with the Amazon Relational Database Service (Amazon RDS) [DescribeDBInstances](#) API operation to pause an automation so that a database instance has time to start.
- **aws:assertAwsResourceProperty**: This automation action allows you to assert a specific resource state or event state for a specific step. For example, you can specify that a step must wait for an EC2 instance to start. Then it will call the Amazon Elastic Compute Cloud (Amazon EC2) [DescribeInstanceStatus](#) API operation with the DesiredValue property of `running`. This ensures that the automation waits for a running instance and then continues when the instance is, in fact, running.

Here is a sample runbook in YAML that uses the `aws:executeAwsApi` action to turn off read and write permissions on an S3 bucket.

```

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 runbook in YAML that uses all three actions. The automation does the following:

- Uses the `aws:executeAwsApi` action to call the Amazon EC2 [DescribeImages](#) API operation 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 operation to launch one instance that uses the `ImageId` from the previous step. It outputs the instance ID as `InstanceId`.
- Uses the `aws:waitForAwsResourceProperty` action to poll the Amazon EC2 `DescribeInstanceStatus` API operation 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 operation to assert that the instance is in the `running` state. The step fails if the instance state isn't `running`.

```

description: Sample runbook using AWS API operations
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-2022-English-Full-Base"
mainSteps:
- name: getImageId
 action: aws:executeAwsApi
 inputs:
 Service: ec2
 Api: DescribeImages
 Filters:
 - Name: "name"
 Values:
 - "{{ ImageName }}"
 outputs:
 - Name: ImageId
 Selector: "$.Images[0].ImageId"
 Type: "String"
- name: launchOneInstance
 action: aws:executeAwsApi
 inputs:
 Service: ec2
 Api: RunInstances
 ImageId: "{{ getImageId.ImageId }}"
 MaxCount: 1
 MinCount: 1
 outputs:
 - Name: InstanceId
 Selector: "$.Instances[0].InstanceId"
 Type: "String"
- name: waitUntilInstanceStateRunning
 action: aws:waitForAwsResourceProperty
 # timeout is strongly encouraged for action - aws:waitForAwsResourceProperty
 timeoutSeconds: 60
 inputs:
 Service: ec2
 Api: DescribeInstanceStatus
 InstanceIds:
 - "{{ launchOneInstance.InstanceId }}"
 PropertySelector: "$.InstanceStatuses[0].InstanceState.Name"
 DesiredValues:
 - running
```

```

- name: assertInstanceStateRunning
 action: aws:assertAwsResourceProperty
 inputs:
 Service: ec2
 Api: DescribeInstanceStatus
 InstanceIds:
 - "{{ launchOneInstance.InstanceId }}"
 PropertySelector: "$.InstanceStatuses[0].InstanceState.Name"
 DesiredValues:
 - running
 outputs:
 - "launchOneInstance.InstanceId"
 ...

```

## Working with inputs and outputs

Each of the previously described automation actions allows you to call a specific API operation by specifying the service namespace, the API operation name, the input parameters, and the output parameters. Inputs are defined by the API operation that you choose. You can view the API operations (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 operations (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 \(p. 486\)](#)
- [aws:executeAwsApi – Call and run AWS API operations \(p. 508\)](#)
- [aws:waitForAwsResourceProperty – Wait on an AWS resource property \(p. 529\)](#)

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 operation. These fields use the JSONPath syntax.

Here is an example to help illustrate this concept for the `aws:executeAwsApi` action.

```

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 automation invokes the `DescribeImages` API operation and receives a response from `ec2`. The automation 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 can use the value of `ImageId` by specifying `"{{ getImageId.ImageId }}"`.

Here is an example to help illustrate this concept for the `aws:waitForAwsResourceProperty` action.

```

- name: waitUntilInstanceStateRunning
 action: aws:waitForAwsResourceProperty
 # timeout is strongly encouraged for 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 automation invokes the `DescribeInstanceStatus` API operation and receives a response from `ec2`. The automation 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 a runbook

A JSONPath expression is a string beginning with `".` that is used to select one or 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. The return type of this operator is always a JSON array. In the context of an automation action 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 operation. Following this response are several examples that show different results by applying different JSONPath expressions to the response from the `DescribeInstances` API operation.

```
{
 "NextToken": "abcdefg",
 "Reservations": [
 {
 "OwnerId": "123456789012",
 "ReservationId": "r-abcd12345678910",
 "Instances": [
 {
 "ImageId": "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": [
 {
 "ImageId": "ami-12345678",
 "BlockDeviceMappings": [
 {
 "Ebs": {
 "DeleteOnTermination": true,
 "Status": "attached",
 "VolumeId": "vol-111111111111"
 },
 "DeviceName": "/dev/xvda"
 }
],
 "State": {
 "Code": 80,
 "Name": "stopped"
 }
 }
],
 "Groups": []
}
]
```

#### JSONPath Example 1: Get a specific String from a JSON response

```
JSONPath:
$.Reservations[0].Instances[0].ImageId

Returns:
"ami-12345678"

Type: String
```

#### JSONPath Example 2: Get a specific Boolean from a JSON response

```
JSONPath:
$.Reservations[0].Instances[0].BlockDeviceMappings[0].Ebs.DeleteOnTermination

Returns:
true

Type: Boolean
```

#### JSONPath Example 3: Get a specific Integer from a JSON response

```
JSONPath:
```

```
$.Reservations[0].Instances[0].State.Code

>Returns:
16

>Type: Integer
```

**JSONPath Example 4: Deep scan a JSON response, then get all of the values for VolumeId as a StringList**

```
JSONPath:
$.Reservations..BlockDeviceMappings..VolumeId

>Returns:
[
 "vol-000000000000",
 "vol-111111111111"
]

>Type: StringList
```

**JSONPath Example 5: Get a specific BlockDeviceMappings object as a StringMap**

```
JSONPath:
$.Reservations[0].Instances[0].BlockDeviceMappings[0]

>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

runbooks (AWS-\* runbooks) such as the AWS-ConfigureS3BucketLogging, AWS-CreateDynamoDBBackup, and AWS-RestartEC2Instance runbooks, to name a few. This requirement also applies to any custom Automation runbooks 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, configure the service role with permission to invoke those services. You can give 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 \(p. 405\)](#).

## Sample walkthrough: Start an Amazon RDS instance from a Systems Manager Automation runbook

This sample walkthrough shows you how to create and run a Systems Manager Automation runbook in YAML that uses all three API operations to see if an Amazon Relational Database Service (Amazon RDS) database instance is running. If the DB instance isn't running, the automation starts it.

### To invoke an Amazon RDS API operation from a runbook

1. Open a text editor and paste the following runbook content into the file. Replace each *example resource placeholder* with your own information. You will add the mainSteps actions later.

```

description: Start RDS instance
schemaVersion: "0.3"
assumeRole: "IAM service role"
parameters:
 InstanceId: db instance ID
 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.
 default: ""
mainSteps:
```

2. For the first step of the automation, 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 runbook, 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 runbook following the list.
  - a. View the schema to see all available inputs for the [aws:assertAwsResourceProperty – Assert an AWS resource state or event state \(p. 486\)](#) 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 operation allows you to view the status of a database instance. You can view the API operations (also called methods) on the [Amazon RDS methods](#) page.
  - d. Specify one or more request parameters for the `DescribeDBInstances` API operation. 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 operation. 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 `DBInstanceState` as the PropertySelectors. Remember that

PropertySelectors are entered by using JSONPath. This means that you format the information in the runbook like the following.

- 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 operation to determine possible values. For this walkthrough, specify *available* and *starting*.
- g. Enter the information you collected into the runbook as shown in the following example.

```

description: Start RDS instance
schemaVersion: "0.3"
assumeRole: "IAM service role"
parameters:
 InstanceId: db instance ID
 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"]
```

- 3. Specify an `aws:executeAwsApi` action in the `mainSteps` section to start the instance if the previous action determined that it isn't started.
  - a. View the schema to see all available inputs for [aws:executeAwsApi – Call and run AWS API operations \(p. 508\)](#).
  - b. Specify the Amazon RDS [StartDBInstance](#) API operation to start the instance.
  - c. Enter the information you collected into the runbook as shown in the following example.

```

description: Start RDS instance
schemaVersion: "0.3"
assumeRole: "{{ IAM service role }}"
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].DBInstanceState"
 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.
  - a. View the schema to see all available inputs for the [aws:waitForAwsResourceProperty – Wait on an AWS resource property \(p. 529\)](#).
  - b. Specify the Amazon RDS [DescribeDBInstances](#) API operation to determine the instance status.
  - c. Specify `$.DBInstances[0].DBInstanceState` as the `PropertySelector`
  - d. Specify *available* as the `DesiredValue`.
  - e. Enter the information you collected into the runbook as shown in the following example.

```

description: Start RDS instance
schemaVersion: "0.3"
assumeRole: "{{ IAM service role }}"
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].DBInstanceState"
 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].DBInstanceState"
 DesiredValues: ["available"]
 isEnd: true
...
-
```

5. Save the file as sample.yaml.
6. Run the following command in the AWS CLI to add the runbook to your AWS account. Replace each *example resource placeholder* with your own information.

```
aws ssm create-document \
 --name runbook name --document-type Automation --document-format YAML --content
 file://sample.yaml
```

7. Run the following command to start the automation by using the runbook you just created. Make a note of the execution ID returned by Systems Manager after you start the automation.

```
aws ssm start-automation-execution \
 --document-name runbook name
```

8. Run the following command to view the execution status.

```
aws ssm get-automation-execution \
 --automation-execution-id automation execution ID
```

## Sample scenarios and custom runbook solutions

The following sample runbook demonstrate how you can use AWS Systems Manager automation actions to automate common deployment, troubleshooting, and maintenance tasks.

### Note

The runbook samples in this section are provided to demonstrate how you can create custom runbooks to support your specific operational needs. These runbooks aren't 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 \(p. 574\)](#)
- [Restore a root volume from the latest snapshot \(p. 593\)](#)
- [Create an AMI and cross-Region copy \(p. 601\)](#)

## 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 AWS 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. Automation is a capability of AWS Systems Manager.

The following sample AWS Systems Manager runbook 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. Parameter Store is a capability of AWS Systems Manager.
- Uses the `aws:executeAwsApi` automation action to call several AWS API operations to create the VPC architecture. The domain controller instances are launched in private subnets, and connect to the internet using a NAT gateway. This allows 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

```

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 run this runbook.
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
- 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'
 Type: String
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: '$.Instances[0].InstanceId'
 Type: String
 nextStep: verifyDCInstanceState
 - name: verifyDCInstanceState
 action: aws:waitForAwsResourceProperty
 inputs:
 Service: ec2
 Api: DescribeInstanceStatus
 IncludeAllInstances: true
 InstanceIds:
 - '{{ launchDC1.pdcInstanceId }}'
 - '{{ launchDC2.adcInstanceId }}'
 PropertySelector: '$.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
 }
 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 }}'
 Parameters:
 commands: |-
 ipconfig /renew
 $dsrmPass = 'sample123!' | ConvertTo-SecureString -asPlainText -Force
 $domAdminUser = "sample\Administrator"
 $domAdminPass = "sampleAdminPass123!" | ConvertTo-SecureString -asPlainText -
 Force
 $domAdminCred = New-Object
 System.Management.Automation.PSCredential($domAdminUser,$domAdminPass)

 try {
 Install-ADDSDomainController -DomainName "sample.com" -InstallDNS -
 DatabasePath "D:\NTDS" -SysvolPath "D:\SYSVOL" -SafeModeAdministratorPassword $dsrmPass
 -Credential $domAdminCred -Force
 }
 catch {
 Write-Error $_
 }
}

```

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 run this runbook.",
 "default": ""
 }
},
"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"
},
{
 "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 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"
 }
]
 },
 "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",
 "Type": "String"
 }
],
 "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": "$.Instances[0].InstanceId",
 "Type": "String"
}
],
"nextStep": "verifyDCInstanceState"
},
{
 "name": "verifyDCInstanceState",
 "action": "aws:waitForAwsResourceProperty",
 "inputs": {
 "Service": "ec2",
 "Api": "DescribeInstanceStatus",
 "IncludeAllInstances": true,
 "InstanceIds": [
 "{{ launchDC1.pdcInstanceId }}",
 "{{ launchDC2.adcInstanceId }}"
],
 "PropertySelector": "$.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",
 "SessionType": "Powershell"
 }
},
"nextStep": "installADRoles"
}
```

```

 "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 }}"
],
 "Parameters": {
 "commands": [
 "ipconfig /renew",
 "$dsrmPass = 'sample123!' | ConvertTo-SecureString -asPlainText -Force",
 "$domAdminUser = \"sample\\Administrator\"",
 "$domAdminPass = \"sampleAdminPass123!\" | ConvertTo-SecureString -asPlainText -Force",
 "$domAdminCred = New-Object System.Management.Automation.PSCredential($domAdminUser,$domAdminPass)",
 "try {",
 "Install-ADDSDomainController -DomainName \"sample.com\" -InstallDNS -DatabasePath \"D:\\NTDS\" -SysvolPath \"D:\\SYSVOL\" -SafeModeAdministratorPassword $dsrmPass -Credential $domAdminCred -Force",
 "}",
 "catch {",
 "Write-Error $_",
 }"
]
 }
 }
 }
}

```

## 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 might 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. Automation is a capability of AWS Systems Manager.

The following sample AWS Systems Manager runbook 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 automation 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 use this runbook."
 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: 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
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 }}"

```

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 run this runbook.",
 "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 }}"
]
 }
 }
],
"files": {
 "getSnapshotsByStartTime.py": {
 "checksums": {
 "sha256": "sampleETagValue"
 }
 }
}
}

```

## 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 AWS 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. Automation is a capability of AWS Systems Manager.

The following sample AWS Systems Manager runbook 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

```

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
 the actions on your behalf. If no role is specified, Systems Manager Automation
 uses your IAM permissions to use this runbook."
 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 run this runbook.",
 "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"
 }
 }
 }
}

```

# Systems Manager Automation runbook reference

To help you get started quickly, AWS Systems Manager provides predefined runbooks. These runbooks are maintained by Amazon Web Services, AWS Support, and AWS Config. The runbook reference describes each of the predefined runbooks provided by Systems Manager, AWS Support, and AWS Config. For more information, see [Systems Manager Automation runbook reference](#).

## Automation walkthroughs

The following walkthroughs help you get started with AWS Systems Manager Automation by authoring and using your own runbooks, and using predefined runbooks. Automation is a capability of AWS Systems Manager.

Before you begin, you must configure Automation roles and permissions. For more information, see [Setting up Automation \(p. 401\)](#). For information about creating a custom runbook, see [Authoring Automation runbooks \(p. 604\)](#) and [Walkthrough: Patch a Windows Server AMI \(p. 661\)](#).

### Walkthroughs

- [Authoring Automation runbooks \(p. 604\)](#)
- [Walkthrough: Using Document Builder to create a custom runbook \(p. 648\)](#)
- [Patching Amazon Machine Images \(p. 654\)](#)
- [Using AWS Support self-service automations \(p. 676\)](#)
- [Walkthrough: Using input transformers with Automation \(p. 686\)](#)
- [Walkthrough: Using Automation with Jenkins \(p. 687\)](#)

## Authoring Automation runbooks

Each runbook in Automation, a capability of AWS Systems Manager, defines an automation. Automation runbooks define the actions that are performed during an automation. In the runbook content, you define the input parameters, outputs, and actions that Systems Manager performs on your managed instances and AWS resources.

Automation includes several pre-defined runbooks that you can use to perform common tasks like restarting one or more Amazon Elastic Compute Cloud (Amazon EC2) instances or creating an Amazon Machine Image (AMI). However, your use cases might extend beyond the capabilities of the pre-defined runbooks. If this is the case, you can create your own runbooks and modify them to your needs.

A runbook consists of automation actions, parameters for those actions, and input parameters that you specify. A runbook's content is written in either YAML or JSON. If you're not familiar with either YAML or JSON, we recommend using Document Builder, or learning more about either markup language before attempting to author your own runbook. For more information about Document Builder, see [Walkthrough: Using Document Builder to create a custom runbook \(p. 648\)](#).

The following sections will help you author your first runbook.

### Identify your use case

The first step in authoring a runbook is identifying your use case. For example, you scheduled the `AWS-CreateImage` runbook to run daily on all of your production Amazon EC2 instances. At the end of the month, you decide you have more images than are necessary for recovery points. Going forward, you want to automatically delete the oldest AMI of an Amazon EC2 instance when a new AMI is created. To accomplish this, you create a new runbook that does the following:

1. Runs the `aws:createImage` action and specifies the instance ID in the image description.

2. Runs the `aws:waitForAwsResourceProperty` action to poll the state of the image until it's available.
3. After the image state is available, the `aws:executeScript` action runs a custom Python script that gathers the IDs of all images associated with your Amazon EC2 instance. The script does this by filtering, using the instance ID in the image description you specified at creation. Then, the script sorts the list of image IDs based on the `creationDate` of the image and outputs the ID of the oldest AMI.
4. Lastly, the `aws:deleteImage` action runs to delete the oldest AMI using the ID from the output of the previous step.

In this scenario, you were already using the `AWS-CreateImage` runbook but found that your use case required greater flexibility. This is a common situation because there can be overlap between runbooks and automation actions. As a result, you might have to adjust which runbooks or actions you use to address your use case.

For example, the `aws:executeScript` and `aws:invokeLambdaFunction` actions both allow you to run custom scripts as part of your automation. To choose between them, you might prefer `aws:invokeLambdaFunction` because of the additional supported runtime languages. However, you might prefer `aws:executeScript` because it allows you to author your script content directly in YAML runbooks and provide script content as attachments for JSON runbooks. You might also consider `aws:executeScript` to be simpler in terms of AWS Identity and Access Management (IAM) setup. Because it uses the permissions provided in the `AutomationAssumeRole`, `aws:executeScript` doesn't require an additional AWS Lambda function execution role.

In any given scenario, one action might provide more flexibility, or added functionality, over another. Therefore, we recommend that you review the available input parameters for the runbook or action you want to use to determine which best fits your use case and preferences.

## Set up your development environment

After you've identified your use case and the pre-defined runbooks or automation actions you want to use in your runbook, it's time to set up your development environment for the content of your runbook. To develop your runbook content, we recommend using the AWS Toolkit for Visual Studio Code instead of the Systems Manager Documents console.

The Toolkit for VS Code is an open-source extension for Visual Studio Code (VS Code) that offers more features than the Systems Manager Documents console. Helpful features include schema validation for both YAML and JSON, snippets for automation action types, and auto-complete support for various options in both YAML and JSON.

For more information about installing the Toolkit for VS Code, see [Installing the AWS Toolkit for Visual Studio Code](#). For more information about using the Toolkit for VS Code to develop runbooks, see [Working with Systems Manager Automation documents](#) in the [AWS Toolkit for Visual Studio Code User Guide](#).

## Develop runbook content

With your use case identified and environment set up, you're ready to develop the content for your runbook. Your use case and preferences will largely dictate the automation actions or runbooks you use in your runbook content. Some actions support only a subset of input parameters when compared to another action that allows you to accomplish a similar task. Other actions have specific outputs, such as `aws:createImage`, where some actions allow you to define your own outputs, such as `aws:executeAwsApi`.

If you're unsure how to use a particular action in your runbook, we recommend reviewing the corresponding entry for the action in the [Systems Manager Automation actions reference \(p. 477\)](#). We also recommend reviewing the content of pre-defined runbooks to see real-world examples of how these actions are used. For more examples of real-world applications of runbooks, see [Sample scenarios and custom runbook solutions \(p. 574\)](#).

To demonstrate the differences in simplicity and flexibility that runbook content provides, the following tutorials provide an example of how to patch groups of Amazon EC2 instances in stages:

- [the section called “Example 1: Creating parent-child runbooks” \(p. 606\)](#) – In this example, two runbooks are used in a parent-child relationship. The parent runbook initiates a rate control automation of the child runbook.
- [the section called “Example 2: Scripted runbook” \(p. 622\)](#) – This example demonstrates how you can accomplish the same tasks of Example 1 by condensing the content into a single runbook and using scripts in your runbook.

## Example 1: Creating parent-child runbooks

The following example demonstrates how to create two runbooks that patch tagged groups of Amazon Elastic Compute Cloud (Amazon EC2) instances in stages. These runbooks are used in a parent-child relationship with the parent runbook used to initiate a rate control automation of the child runbook. For more information about rate control automations, see [Running automations that use targets and rate controls \(p. 421\)](#). For more information about the automation actions used in this example, see the [Systems Manager Automation actions reference \(p. 477\)](#).

### Create the child runbook

This example runbook addresses the following scenario. Emily is a Systems Engineer at AnyCompany Consultants, LLC. She needs to configure patching for groups of Amazon Elastic Compute Cloud (Amazon EC2) instances that host primary and secondary databases. Applications access these databases 24 hours a day, so one of the database instances must always be available.

She decides that patching the instances in stages is the best approach. The primary group of database instances will be patched first, followed by the secondary group of database instances. Also, to avoid incurring additional costs by leaving instances running that were previously stopped, Emily wants the patched instances to be returned to their original state before the patching occurred.

Emily identifies the primary and secondary groups of database instances by the tags associated with the instances. She decides to create a parent runbook that starts a rate control automation of a child runbook. By doing that, she can target the tags associated with the primary and secondary groups of database instances and manage the concurrency of the child automations. After reviewing the available Systems Manager (SSM) documents for patching, she chooses the `AWS-RunPatchBaseline` document. By using this SSM document, her colleagues can review the associated patch compliance information after the patching operation completes.

To start creating her runbook content, Emily reviews the available automation actions and begins authoring the content for the child runbook as follows:

1. First, she provides values for the schema and description of the runbook, and defines the input parameters for the child runbook.

By using the `AutomationAssumeRole` parameter, Emily and her colleagues can use an existing IAM role that allows Automation to perform the actions in the runbook on their behalf. Emily uses the `InstanceId` parameter to determine the instance that should be patched. Optionally, the `Operation`, `RebootOption`, and `SnapshotId` parameters can be used to provide values to document parameters for `AWS-RunPatchBaseline`. To prevent invalid values from being provided to those document parameters, she defines the `allowedValues` as needed.

#### YAML

```
schemaVersion: '0.3'
description: 'An example of an Automation runbook that patches groups of Amazon EC2
instances in stages.'
assumeRole: '{{AutomationAssumeRole}}'
parameters:
```

```

AutomationAssumeRole:
 type: String
 description: >-
 '(Optional) The Amazon Resource Name (ARN) of the IAM role that allows
Automation to perform the
actions on your behalf. If no role is specified, Systems Manager
Automation uses your IAM permissions to operate this runbook.'
 default: ''
InstanceId:
 type: String
 description: >-
 '(Required) The instance you want to patch.'
SnapshotId:
 type: String
 description: '(Optional) The snapshot ID to use to retrieve a patch baseline
snapshot.'
 default: ''
RebootOption:
 type: String
 description: '(Optional) Reboot behavior after a patch Install operation. If you
choose NoReboot and patches are installed, the instance is marked as non-compliant
until a subsequent reboot and scan.'
 allowedValues:
 - NoReboot
 - RebootIfNeeded
 default: RebootIfNeeded
Operation:
 type: String
 description: '(Optional) The update or configuration to perform on the instance.
The system checks if patches specified in the patch baseline are installed on the
instance. The install operation installs patches missing from the baseline.'
 allowedValues:
 - Install
 - Scan
 default: Install

```

#### JSON

```
{
 "schemaVersion": "0.3",
 "description": "An example of an Automation runbook that patches groups of Amazon
EC2 instances in stages.",
 "assumeRole": "{{AutomationAssumeRole}}",
 "parameters": {
 "AutomationAssumeRole": {
 "type": "String",
 "description": "(Optional) The Amazon Resource Name (ARN) of the IAM
role that allows Automation to perform the actions on your behalf. If no role is
specified, Systems Manager Automation uses your IAM permissions to operate this
runbook.",
 "default": ""
 },
 "InstanceId": {
 "type": "String",
 "description": "(Required) The instance you want to patch."
 },
 "SnapshotId": {
 "type": "String",
 "description": "(Optional) The snapshot ID to use to retrieve a patch
baseline snapshot.",
 "default": ""
 },
 "RebootOption": {
 "type": "String",

```

```

 "description": "(Optional) Reboot behavior after a patch Install operation.
If you choose NoReboot and patches are installed, the instance is marked as non-compliant until a subsequent reboot and scan.",
 "allowedValues": [
 "NoReboot",
 "RebootIfNeeded"
],
 "default": "RebootIfNeeded"
},
 "Operation": {
 "type": "String",
 "description": "(Optional) The update or configuration to perform on the instance. The system checks if patches specified in the patch baseline are installed on the instance. The install operation installs patches missing from the baseline.",
 "allowedValues": [
 "Install",
 "Scan"
],
 "default": "Install"
 }
}
},
},
}

```

- With the top-level elements defined, Emily proceeds with authoring the actions that make up the mainSteps of the runbook. The first step outputs the current state of the target instance specified in the InstanceId input parameter using the aws:executeAwsApi action. The output of this action is used in later actions.

#### YAML

```

mainSteps:
 - name: getInstanceState
 action: 'aws:executeAwsApi'
 onFailure: Abort
 inputs:
 inputs:
 Service: ec2
 Api: DescribeInstances
 InstanceIds:
 - '{{InstanceId}}'
 outputs:
 - Name: instanceState
 Selector: '$.Reservations[0].Instances[0].State.Name'
 Type: String
 nextStep: branchOnInstanceState

```

#### JSON

```

"mainSteps": [
 {
 "name": "getInstanceState",
 "action": "aws:executeAwsApi",
 "onFailure": "Abort",
 "inputs": {
 "inputs": null,
 "Service": "ec2",
 "Api": "DescribeInstances",
 "InstanceIds": [
 "{{InstanceId}}"
]
 },
 "outputs": [
 {
 "Name": "instanceState",

```

```

 "Selector":("$.Reservations[0].Instances[0].State.Name",
 "Type":"String"
 },
],
"nextStep": "branchOnInstanceState"
},

```

3. Rather than manually starting and keeping track of the original state of every instance that needs to be patched, Emily uses the output from the previous action to branch the automation based on the state of the target instance. This allows the automation to run different steps depending on the conditions defined in the `aws:branch` action and improves the overall efficiency of the automation without manual intervention.

If the instance state is already running, the automation proceeds with patching the instance with the `AWS-RunPatchBaseline` document using the `aws:runCommand` action.

If the instance state is stopping, the automation polls for the instance to reach the stopped state using the `aws:waitForAwsResourceProperty` action, starts the instance using the `executeAwsApi` action, and polls for the instance to reach a running state before patching the instance.

If the instance state is stopped, the automation starts the instance and polls for the instance to reach a running state before patching the instance using the same actions.

#### YAML

```

- name: branchOnInstanceState
 action: 'aws:branch'
 onFailure: Abort
 inputs:
 Choices:
 - NextStep: startInstance
 Variable: '{{getInstanceState.instanceState}}'
 StringEquals: stopped
 - Or:
 - Variable: '{{getInstanceState.instanceState}}'
 StringEquals: stopping
 NextStep: verifyInstanceStopped
 - NextStep: patchInstance
 Variable: '{{getInstanceState.instanceState}}'
 StringEquals: running
 isEnd: true
- name: startInstance
 action: 'aws:executeAwsApi'
 onFailure: Abort
 inputs:
 Service: ec2
 Api: StartInstances
 InstanceIds:
 - '{{InstanceId}}'
 nextStep: verifyInstanceRunning
- name: verifyInstanceRunning
 action: 'aws:waitForAwsResourceProperty'
 timeoutSeconds: 120
 inputs:
 Service: ec2
 Api: DescribeInstances
 InstanceIds:
 - '{{InstanceId}}'
 PropertySelector: '$.Reservations[0].Instances[0].State.Name'
 DesiredValues:
 - running
 nextStep: patchInstance
- name: verifyInstanceStopped

```

```

action: 'aws:waitForAwsResourceProperty'
timeoutSeconds: 120
inputs:
 Service: ec2
 Api: DescribeInstances
 InstanceIds:
 - '{{InstanceId}}'
 PropertySelector: '$.Reservations[0].Instances[0].State.Name'
 DesiredValues:
 - stopped
 nextStep: startInstance
- name: patchInstance
 action: 'aws:runCommand'
 onFailure: Abort
 timeoutSeconds: 5400
 inputs:
 DocumentName: 'AWS-RunPatchBaseline'
 InstanceIds:
 - '{{InstanceId}}'
 Parameters:
 SnapshotId: '{{SnapshotId}}'
 RebootOption: '{{RebootOption}}'
 Operation: '{{Operation}}'

```

JSON

```
{
 "name": "branchOnInstanceState",
 "action": "aws:branch",
 "onFailure": "Abort",
 "inputs": {
 "Choices": [
 {
 "NextStep": "startInstance",
 "Variable": "{{getInstanceState.instanceState}}",
 "StringEquals": "stopped"
 },
 {
 "Or": [
 {
 "Variable": "{{getInstanceState.instanceState}}",
 "StringEquals": "stopping"
 }
],
 "NextStep": "verifyInstanceStopped"
 },
 {
 "NextStep": "patchInstance",
 "Variable": "{{getInstanceState.instanceState}}",
 "StringEquals": "running"
 }
]
 },
 "isEnd": true
},
{
 "name": "startInstance",
 "action": "aws:executeAwsApi",
 "onFailure": "Abort",
 "inputs": {
 "Service": "ec2",
 "Api": "StartInstances",
 "InstanceIds": [
 "{{InstanceId}}"
]
 }
}
```

```

],
 },
 "nextStep": "verifyInstanceRunning"
},
{
 "name": "verifyInstanceRunning",
 "action": "aws:waitForAwsResourceProperty",
 "timeoutSeconds": 120,
 "inputs": {
 "Service": "ec2",
 "Api": "DescribeInstances",
 "InstanceIds": [
 "{{InstanceId}}"
],
 "PropertySelector": "$.Reservations[0].Instances[0].State.Name",
 "DesiredValues": [
 "running"
]
 },
 "nextStep": "patchInstance"
},
{
 "name": "verifyInstanceStopped",
 "action": "aws:waitForAwsResourceProperty",
 "timeoutSeconds": 120,
 "inputs": {
 "Service": "ec2",
 "Api": "DescribeInstances",
 "InstanceIds": [
 "{{InstanceId}}"
],
 "PropertySelector": "$.Reservations[0].Instances[0].State.Name",
 "DesiredValues": [
 "stopped"
],
 "nextStep": "startInstance"
 }
},
{
 "name": "patchInstance",
 "action": "aws:runCommand",
 "onFailure": "Abort",
 "timeoutSeconds": 5400,
 "inputs": {
 "DocumentName": "AWS-RunPatchBaseline",
 "InstanceIds": [
 "{{InstanceId}}"
],
 "Parameters": {
 "SnapshotId": "{{SnapshotId}}",
 "RebootOption": "{{RebootOption}}",
 "Operation": "{{Operation}}"
 }
 }
},

```

- After the patching operation completes, Emily wants the automation to return the target instance to the same state it was in before the automation started. She does this by again using the output from the first action. The automation branches based on the original state of the target instance using the `aws:branch` action. If the instance was previously in any state other than `running`, the instance is stopped. Otherwise, if the instance state is `running`, the automation ends.

**YAML**

```
- name: branchOnOriginalInstanceState
```

```

action: 'aws:branch'
onFailure: Abort
inputs:
 Choices:
 - NextStep: stopInstance
 Not:
 Variable: '{{getInstanceState.instanceState}}'
 StringEquals: running
 isEnd: true
- name: stopInstance
 action: 'aws:executeAwsApi'
 onFailure: Abort
 inputs:
 Service: ec2
 Api: StopInstances
 InstanceIds:
 - '{{InstanceId}}'

```

JSON

```
{
 "name": "branchOnOriginalInstanceState",
 "action": "aws:branch",
 "onFailure": "Abort",
 "inputs": {
 "Choices": [
 {
 "NextStep": "stopInstance",
 "Not": {
 "Variable": "{{getInstanceState.instanceState}}",
 "StringEquals": "running"
 }
 }
],
 "isEnd": true
 },
 {
 "name": "stopInstance",
 "action": "aws:executeAwsApi",
 "onFailure": "Abort",
 "inputs": {
 "Service": "ec2",
 "Api": "StopInstances",
 "InstanceIds": [
 "{{InstanceId}}"
]
 }
 }
}
```

- Emily reviews the completed child runbook content and creates the runbook in the same AWS account and AWS Region as the target instances. Now she's ready to continue with the creation of the parent runbook's content. The following is the completed child runbook content.

YAML

```

schemaVersion: '0.3'
description: 'An example of an Automation runbook that patches groups of Amazon EC2
instances in stages.'
assumeRole: '{{AutomationAssumeRole}}'
parameters:
 AutomationAssumeRole:

```

```

 type: String
 description: >-
 '(Optional) The Amazon Resource Name (ARN) of the IAM role that allows
Automation to perform the
 actions on your behalf. If no role is specified, Systems Manager
 Automation uses your IAM permissions to operate this runbook.'
 default: ''
 InstanceId:
 type: String
 description: >-
 '(Required) The instance you want to patch.'
 SnapshotId:
 type: String
 description: '(Optional) The snapshot ID to use to retrieve a patch baseline
snapshot.'
 default: ''
 RebootOption:
 type: String
 description: '(Optional) Reboot behavior after a patch Install operation. If you
choose NoReboot and patches are installed, the instance is marked as non-compliant
until a subsequent reboot and scan.'
 allowedValues:
 - NoReboot
 - RebootIfNeeded
 default: RebootIfNeeded
 Operation:
 type: String
 description: '(Optional) The update or configuration to perform on the instance.
The system checks if patches specified in the patch baseline are installed on the
instance. The install operation installs patches missing from the baseline.'
 allowedValues:
 - Install
 - Scan
 default: Install
 mainSteps:
 - name: getInstanceState
 action: 'aws:executeAwsApi'
 onFailure: Abort
 inputs:
 inputs:
 Service: ec2
 Api: DescribeInstances
 InstanceIds:
 - '{{InstanceId}}'
 outputs:
 - Name: instanceState
 Selector: '$.Reservations[0].Instances[0].State.Name'
 Type: String
 nextStep: branchOnInstanceState
 - name: branchOnInstanceState
 action: 'aws:branch'
 onFailure: Abort
 inputs:
 Choices:
 - NextStep: startInstance
 Variable: '{{getInstanceState.instanceState}}'
 StringEquals: stopped
 - Or:
 - Variable: '{{getInstanceState.instanceState}}'
 StringEquals: stopping
 NextStep: verifyInstanceStopped
 - NextStep: patchInstance
 Variable: '{{getInstanceState.instanceState}}'
 StringEquals: running
 isEnd: true
 - name: startInstance

```

```
action: 'aws:executeAwsApi'
onFailure: Abort
inputs:
 Service: ec2
 Api: StartInstances
 InstanceIds:
 - '{{InstanceId}}'
nextStep: verifyInstanceRunning
- name: verifyInstanceRunning
 action: 'aws:waitForAwsResourceProperty'
 timeoutSeconds: 120
 inputs:
 Service: ec2
 Api: DescribeInstances
 InstanceIds:
 - '{{InstanceId}}'
 PropertySelector: '$.Reservations[0].Instances[0].State.Name'
 DesiredValues:
 - running
nextStep: patchInstance
- name: verifyInstanceStopped
 action: 'aws:waitForAwsResourceProperty'
 timeoutSeconds: 120
 inputs:
 Service: ec2
 Api: DescribeInstances
 InstanceIds:
 - '{{InstanceId}}'
 PropertySelector: '$.Reservations[0].Instances[0].State.Name'
 DesiredValues:
 - stopped
nextStep: startInstance
- name: patchInstance
 action: 'aws:runCommand'
 onFailure: Abort
 timeoutSeconds: 5400
 inputs:
 DocumentName: 'AWS-RunPatchBaseline'
 InstanceIds:
 - '{{InstanceId}}'
 Parameters:
 SnapshotId: '{{SnapshotId}}'
 RebootOption: '{{RebootOption}}'
 Operation: '{{Operation}}'
- name: branchOnOriginalInstanceState
 action: 'aws:branch'
 onFailure: Abort
 inputs:
 Choices:
 - NextStep: stopInstance
 Not:
 variable: '{{getInstanceState.instanceState}}'
 StringEquals: running
 isEnd: true
- name: stopInstance
 action: 'aws:executeAwsApi'
 onFailure: Abort
 inputs:
 Service: ec2
 Api: StopInstances
 InstanceIds:
 - '{{InstanceId}}'
```

## JSON

```
{
 "schemaVersion": "0.3",
 "description": "An example of an Automation runbook that patches groups of Amazon EC2 instances in stages.",
 "assumeRole": "{{AutomationAssumeRole}}",
 "parameters": {
 "AutomationAssumeRole": {
 "type": "String",
 "description": "(Optional) The Amazon Resource Name (ARN) of the IAM role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to operate this runbook.",
 "default": ""
 },
 "InstanceId": {
 "type": "String",
 "description": "(Required) The instance you want to patch."
 },
 "SnapshotId": {
 "type": "String",
 "description": "(Optional) The snapshot ID to use to retrieve a patch baseline snapshot.",
 "default": ""
 },
 "RebootOption": {
 "type": "String",
 "description": "(Optional) Reboot behavior after a patch Install operation. If you choose NoReboot and patches are installed, the instance is marked as non-compliant until a subsequent reboot and scan.",
 "allowedValues": [
 "NoReboot",
 "RebootIfNeeded"
],
 "default": "RebootIfNeeded"
 },
 "Operation": {
 "type": "String",
 "description": "(Optional) The update or configuration to perform on the instance. The system checks if patches specified in the patch baseline are installed on the instance. The install operation installs patches missing from the baseline.",
 "allowedValues": [
 "Install",
 "Scan"
],
 "default": "Install"
 }
 },
 "mainSteps": [
 {
 "name": "getInstanceState",
 "action": "aws:executeAwsApi",
 "onFailure": "Abort",
 "inputs": {
 "inputs": null,
 "Service": "ec2",
 "Api": "DescribeInstances",
 "InstanceIds": [
 "{{InstanceId}}"
]
 },
 "outputs": [
]
 }
]
}
```

```

 "Name":"instanceState",
 "Selector":("$.Reservations[0].Instances[0].State.Name",
 "Type":"String"
 }
],
"nextStep":"branchOnInstanceState"
},
{
 "name":"branchOnInstanceState",
 "action":"aws:branch",
 "onFailure":"Abort",
 "inputs": {
 "Choices": [
 {
 "NextStep":"startInstance",
 "Variable":"{{getInstanceState.instanceState}}",
 "StringEquals":"stopped"
 },
 {
 "Or": [
 {
 "Variable":"{{getInstanceState.instanceState}}",
 "StringEquals":"stopping"
 }
],
 "NextStep":"verifyInstanceStopped"
 },
 {
 "NextStep":"patchInstance",
 "Variable":"{{getInstanceState.instanceState}}",
 "StringEquals":"running"
 }
]
 },
 "isEnd":true
},
{
 "name":"startInstance",
 "action":"aws:executeAwsApi",
 "onFailure":"Abort",
 "inputs": {
 "Service":"ec2",
 "Api":"StartInstances",
 "InstanceIds": [
 "{{InstanceId}}"
]
 },
 "nextStep":"verifyInstanceRunning"
},
{
 "name":"verifyInstanceRunning",
 "action":"aws:waitForAwsResourceProperty",
 "timeoutSeconds":120,
 "inputs": {
 "Service":"ec2",
 "Api":"DescribeInstances",
 "InstanceIds": [
 "{{InstanceId}}"
],
 "PropertySelector":("$.Reservations[0].Instances[0].State.Name",
 "DesiredValues": [
 "running"
]
 },
 "nextStep":"patchInstance"
}
,
```

```
{
 "name": "verifyInstanceStopped",
 "action": "aws:waitForAwsResourceProperty",
 "timeoutSeconds": 120,
 "inputs": {
 "Service": "ec2",
 "Api": "DescribeInstances",
 "InstanceIds": [
 "{{InstanceId}}"
],
 "PropertySelector": "$.Reservations[0].Instances[0].State.Name",
 "DesiredValues": [
 "stopped"
],
 "nextStep": "startInstance"
 }
},
{
 "name": "patchInstance",
 "action": "aws:runCommand",
 "onFailure": "Abort",
 "timeoutSeconds": 5400,
 "inputs": {
 "DocumentName": "AWS-RunPatchBaseline",
 "InstanceIds": [
 "{{InstanceId}}"
],
 "Parameters": {
 "SnapshotId": "{{SnapshotId}}",
 "RebootOption": "{{RebootOption}}",
 "Operation": "{{Operation}}"
 }
 }
},
{
 "name": "branchOnOriginalInstanceState",
 "action": "aws:branch",
 "onFailure": "Abort",
 "inputs": {
 "Choices": [
 {
 "NextStep": "stopInstance",
 "Not": {
 "Variable": "{{getInstanceState.instanceState}}",
 "StringEquals": "running"
 }
 }
]
 },
 "isEnd": true
},
{
 "name": "stopInstance",
 "action": "aws:executeAwsApi",
 "onFailure": "Abort",
 "inputs": {
 "Service": "ec2",
 "Api": "StopInstances",
 "InstanceIds": [
 "{{InstanceId}}"
]
 }
}
]
```

For more information about the automation actions used in this example, see the [Systems Manager Automation actions reference \(p. 477\)](#).

### Create the parent runbook

This example runbook continues the scenario described in the previous section. Now that Emily has created the child runbook, she begins authoring the content for the parent runbook as follows:

1. First, she provides values for the schema and description of the runbook, and defines the input parameters for the parent runbook.

By using the `AutomationAssumeRole` parameter, Emily and her colleagues can use an existing IAM role that allows Automation to perform the actions in the runbook on their behalf. Emily uses the `PatchGroupPrimaryKey` and `PatchGroupPrimaryValue` parameters to specify the tag associated with the primary group of database instances that will be patched. She uses the `PatchGroupSecondaryKey` and `PatchGroupSecondaryValue` parameters to specify the tag associated with the secondary group of database instances that will be patched.

#### YAML

```
description: 'An example of an Automation runbook that patches groups of Amazon EC2 instances in stages.'
schemaVersion: '0.3'
assumeRole: '{{AutomationAssumeRole}}'
parameters:
 AutomationAssumeRole:
 type: String
 description: '(Optional) The Amazon Resource Name (ARN) of the IAM role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to operate this runbook.'
 default: ''
 PatchGroupPrimaryKey:
 type: String
 description: '(Required) The key of the tag for the primary group of instances you want to patch.'
 PatchGroupPrimaryValue:
 type: String
 description: '(Required) The value of the tag for the primary group of instances you want to patch.'
 PatchGroupSecondaryKey:
 type: String
 description: '(Required) The key of the tag for the secondary group of instances you want to patch.'
 PatchGroupSecondaryValue:
 type: String
 description: '(Required) The value of the tag for the secondary group of instances you want to patch.'
```

#### JSON

```
{
 "schemaVersion": "0.3",
 "description": "An example of an Automation runbook that patches groups of Amazon EC2 instances in stages.",
 "assumeRole": "{{AutomationAssumeRole}}",
 "parameters": {
 "AutomationAssumeRole": {
 "type": "String",
 "description": "(Optional) The Amazon Resource Name (ARN) of the IAM role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to operate this runbook.",
 "default": ""
 }
 }
}
```

```

 },
 "PatchGroupPrimaryKey": {
 "type": "String",
 "description": "(Required) The key of the tag for the primary group of
instances you want to patch."
 },
 "PatchGroupPrimaryValue": {
 "type": "String",
 "description": "(Required) The value of the tag for the primary group of
instances you want to patch."
 },
 "PatchGroupSecondaryKey": {
 "type": "String",
 "description": "(Required) The key of the tag for the secondary group of
instances you want to patch."
 },
 "PatchGroupSecondaryValue": {
 "type": "String",
 "description": "(Required) The value of the tag for the secondary group of
instances you want to patch."
 }
 }
},

```

- With the top-level elements defined, Emily proceeds with authoring the actions that make up the `mainSteps` of the runbook.

The first action starts a rate control automation using the child runbook she just created that targets instances associated with the tag specified in the `PatchGroupPrimaryKey` and `PatchGroupPrimaryValue` input parameters. She uses the values provided to the input parameters to specify the key and value of the tag associated with the primary group of database instances she wants to patch.

After the first automation completes, the second action starts another rate control automation using the child runbook that targets instances associated with the tag specified in the `PatchGroupSecondaryKey` and `PatchGroupSecondaryValue` input parameters. She uses the values provided to the input parameters to specify the key and value of the tag associated with the secondary group of database instances she wants to patch.

#### YAML

```

mainSteps:
- name: patchPrimaryTargets
 action: 'aws:executeAutomation'
 onFailure: Abort
 timeoutSeconds: 7200
 inputs:
 DocumentName: RunbookTutorialChildAutomation
 Targets:
 - Key: 'tag:{{PatchGroupPrimaryKey}}'
 Values:
 - '{{PatchGroupPrimaryValue}}'
 TargetParameterName: 'InstanceId'
- name: patchSecondaryTargets
 action: 'aws:executeAutomation'
 onFailure: Abort
 timeoutSeconds: 7200
 inputs:
 DocumentName: RunbookTutorialChildAutomation
 Targets:
 - Key: 'tag:{{PatchGroupSecondaryKey}}'
 Values:
 - '{{PatchGroupSecondaryValue}}'

```

```
TargetParameterName: 'InstanceId'
```

#### JSON

```
"mainSteps": [
 {
 "name": "patchPrimaryTargets",
 "action": "aws:executeAutomation",
 "onFailure": "Abort",
 "timeoutSeconds": 7200,
 "inputs": {
 "DocumentName": "RunbookTutorialChildAutomation",
 "Targets": [
 {
 "Key": "tag:{{PatchGroupPrimaryKey}}",
 "Values": [
 "{{PatchGroupPrimaryValue}}"
]
 }
],
 "TargetParameterName": "InstanceId"
 }
 },
 {
 "name": "patchSecondaryTargets",
 "action": "aws:executeAutomation",
 "onFailure": "Abort",
 "timeoutSeconds": 7200,
 "inputs": {
 "DocumentName": "RunbookTutorialChildAutomation",
 "Targets": [
 {
 "Key": "tag:{{PatchGroupSecondaryKey}}",
 "Values": [
 "{{PatchGroupSecondaryValue}}"
]
 }
],
 "TargetParameterName": "InstanceId"
 }
 }
]
```

- Emily reviews the completed parent runbook content and creates the runbook in the same AWS account and AWS Region as the target instances. Now, she is ready to test her runbooks to make sure the automation operates as desired before implementing them into her production environment. The following is the completed parent runbook content.

#### YAML

```
description: An example of an Automation runbook that patches groups of Amazon EC2 instances in stages.
schemaVersion: '0.3'
assumeRole: '{{AutomationAssumeRole}}'
parameters:
 AutomationAssumeRole:
 type: String
 description: '(Optional) The Amazon Resource Name (ARN) of the IAM role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to operate this runbook.'
 default: ''
 PatchGroupPrimaryKey:
 type: String
```

```

 description: (Required) The key of the tag for the primary group of instances you
want to patch.
 PatchGroupPrimaryKey:
 type: String
 description: '(Required) The value of the tag for the primary group of instances
you want to patch. '
 PatchGroupSecondaryKey:
 type: String
 description: (Required) The key of the tag for the secondary group of instances
you want to patch.
 PatchGroupSecondaryValue:
 type: String
 description: '(Required) The value of the tag for the secondary group of
instances you want to patch. '
mainSteps:
- name: patchPrimaryTargets
 action: 'aws:executeAutomation'
 onFailure: Abort
 timeoutSeconds: 7200
 inputs:
 DocumentName: RunbookTutorialChildAutomation
 Targets:
 - Key: 'tag:{{PatchGroupPrimaryKey}}'
 Values:
 - '{{PatchGroupPrimaryValue}}'
 TargetParameterName: 'InstanceId'
- name: patchSecondaryTargets
 action: 'aws:executeAutomation'
 onFailure: Abort
 timeoutSeconds: 7200
 inputs:
 DocumentName: RunbookTutorialChildAutomation
 Targets:
 - Key: 'tag:{{PatchGroupSecondaryKey}}'
 Values:
 - '{{PatchGroupSecondaryValue}}'
 TargetParameterName: 'InstanceId'

```

#### JSON

```
{
 "description": "An example of an Automation runbook that patches groups of Amazon
EC2 instances in stages.",
 "schemaVersion": "0.3",
 "assumeRole": "{{AutomationAssumeRole}}",
 "parameters": {
 "AutomationAssumeRole": {
 "type": "String",
 "description": "(Optional) The Amazon Resource Name (ARN) of the IAM
role that allows Automation to perform the actions on your behalf. If no role is
specified, Systems Manager Automation uses your IAM permissions to operate this
runbook.",
 "default": ""
 },
 "PatchGroupPrimaryKey": {
 "type": "String",
 "description": "(Required) The key of the tag for the primary group of
instances you want to patch."
 },
 "PatchGroupPrimaryValue": {
 "type": "String",
 "description": "(Required) The value of the tag for the primary group of
instances you want to patch. "
 },

```

```

 "PatchGroupSecondaryKey": {
 "type": "String",
 "description": "(Required) The key of the tag for the secondary group of
instances you want to patch."
 },
 "PatchGroupSecondaryValue": {
 "type": "String",
 "description": "(Required) The value of the tag for the secondary group of
instances you want to patch. "
 }
},
"mainSteps": [
{
 "name": "patchPrimaryTargets",
 "action": "aws:executeAutomation",
 "onFailure": "Abort",
 "timeoutSeconds": 7200,
 "inputs": {
 "DocumentName": "RunbookTutorialChildAutomation",
 "Targets": [
 {
 "Key": "tag:{{PatchGroupPrimaryKey}}",
 "Values": [
 "{{PatchGroupPrimaryValue}}"
]
 }
],
 "TargetParameterName": "InstanceId"
 }
},
{
 "name": "patchSecondaryTargets",
 "action": "aws:executeAutomation",
 "onFailure": "Abort",
 "timeoutSeconds": 7200,
 "inputs": {
 "DocumentName": "RunbookTutorialChildAutomation",
 "Targets": [
 {
 "Key": "tag:{{PatchGroupSecondaryKey}}",
 "Values": [
 "{{PatchGroupSecondaryValue}}"
]
 }
],
 "TargetParameterName": "InstanceId"
 }
}
]
}
}

```

For more information about the automation actions used in this example, see the [Systems Manager Automation actions reference \(p. 477\)](#).

## Example 2: Scripted runbook

This example runbook addresses the following scenario. Emily is a Systems Engineer at AnyCompany Consultants, LLC. She previously created two runbooks that are used in a parent-child relationship to patch groups of Amazon Elastic Compute Cloud (Amazon EC2) instances that host primary and secondary databases. Applications access these databases 24 hours a day, so one of the database instances must always be available.

Based on this requirement, she built a solution that patches the instances in stages using the AWS-RunPatchBaseline Systems Manager (SSM) document. By using this SSM document, her colleagues can review the associated patch compliance information after the patching operation completes.

The primary group of database instances are patched first, followed by the secondary group of database instances. Also, to avoid incurring additional costs by leaving instances running that were previously stopped, Emily made sure that the automation returned the patched instances to their original state before the patching occurred. Emily used tags that are associated with the primary and secondary groups of database instances to identify which instances should be patched in her desired order.

Her existing automated solution works, but she wants to improve her solution if possible. To help with the maintenance of the runbook content and to ease troubleshooting efforts, she would like to condense the automation into a single runbook and simplify the number of input parameters. Also, she would like to avoid creating multiple child automations.

After Emily reviews the available automation actions, she determines that she can improve her solution by using the `aws:executeScript` action to run her custom Python scripts. She now begins authoring the content for the runbook as follows:

1. First, she provides values for the schema and description of the runbook, and defines the input parameters for the parent runbook.

By using the `AutomationAssumeRole` parameter, Emily and her colleagues can use an existing IAM role that allows Automation to perform the actions in the runbook on their behalf. Unlike [Example 1 \(p. 606\)](#), the `AutomationAssumeRole` parameter is now required rather than optional. Because this runbook includes `aws:executeScript` actions, an AWS Identity and Access Management (IAM) service role (or assume role) is always required. This requirement is necessary because some of the Python scripts specified for the actions call AWS API operations.

Emily uses the `PrimaryPatchGroupTag` and `SecondaryPatchGroupTag` parameters to specify the tags associated with the primary and secondary group of database instances that will be patched. To simplify the required input parameters, she decides to use `StringMap` parameters rather than using multiple `String` parameters as she used in the Example 1 runbook. Optionally, the `Operation`, `RebootOption`, and `SnapshotId` parameters can be used to provide values to document parameters for `AWS-RunPatchBaseline`. To prevent invalid values from being provided to those document parameters, she defines the `allowedValues` as needed.

#### YAML

```
description: 'An example of an Automation runbook that patches groups of Amazon EC2 instances in stages.'
schemaVersion: '0.3'
assumeRole: '{{AutomationAssumeRole}}'
parameters:
 AutomationAssumeRole:
 type: String
 description: '(Required) The Amazon Resource Name (ARN) of the IAM role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to operate this runbook.'
 PrimaryPatchGroupTag:
 type: StringMap
 description: '(Required) The tag for the primary group of instances you want to patch. Specify a key-value pair. Example: {"key" : "value"}'
 SecondaryPatchGroupTag:
 type: StringMap
 description: '(Required) The tag for the secondary group of instances you want to patch. Specify a key-value pair. Example: {"key" : "value"}'
 SnapshotId:
 type: String
 description: '(Optional) The snapshot ID to use to retrieve a patch baseline snapshot.'
 default: ''
```

```

RebootOption:
 type: String
 description: '(Optional) Reboot behavior after a patch Install operation. If you choose NoReboot and patches are installed, the instance is marked as non-compliant until a subsequent reboot and scan.'
 allowedValues:
 - NoReboot
 - RebootIfNeeded
 default: RebootIfNeeded
Operation:
 type: String
 description: '(Optional) The update or configuration to perform on the instance. The system checks if patches specified in the patch baseline are installed on the instance. The install operation installs patches missing from the baseline.'
 allowedValues:
 - Install
 - Scan
 default: Install

```

#### JSON

```
{
 "description": "An example of an Automation runbook that patches groups of Amazon EC2 instances in stages.",
 "schemaVersion": "0.3",
 "assumeRole": "{{AutomationAssumeRole}}",
 "parameters": {
 "AutomationAssumeRole": {
 "type": "String",
 "description": "(Required) The Amazon Resource Name (ARN) of the IAM role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to operate this runbook."
 },
 "PrimaryPatchGroupTag": {
 "type": "StringMap",
 "description": "(Required) The tag for the primary group of instances you want to patch. Specify a key-value pair. Example: {\\"key\\": \\"value\\"}"
 },
 "SecondaryPatchGroupTag": {
 "type": "StringMap",
 "description": "(Required) The tag for the secondary group of instances you want to patch. Specify a key-value pair. Example: {\\"key\\": \\"value\\"}"
 },
 "SnapshotId": {
 "type": "String",
 "description": "(Optional) The snapshot ID to use to retrieve a patch baseline snapshot.",
 "default": ""
 },
 "RebootOption": {
 "type": "String",
 "description": "(Optional) Reboot behavior after a patch Install operation. If you choose NoReboot and patches are installed, the instance is marked as non-compliant until a subsequent reboot and scan.",
 "allowedValues": [
 "NoReboot",
 "RebootIfNeeded"
],
 "default": "RebootIfNeeded"
 },
 "Operation": {
 "type": "String",

```

```
 "description": "(Optional) The update or configuration to perform on the instance. The system checks if patches specified in the patch baseline are installed on the instance. The install operation installs patches missing from the baseline.",
 "allowedValues": [
 "Install",
 "Scan"
],
 "default": "Install"
 }
},
}
```

2. With the top-level elements defined, Emily proceeds with authoring the actions that make up the mainSteps of the runbook. The first step gathers the IDs of all instances associated with the tag specified in the PrimaryPatchGroupTag parameter and outputs a StringMap parameter containing the instance ID and the current state of the instance. The output of this action is used in later actions.

Note that the `script` input parameter isn't supported for JSON runbooks. JSON runbooks must provide script content using the `attachment` input parameter.

YAML

```
mainSteps:
- name: getPrimaryInstanceState
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: getInstanceStates
 InputPayload:
 primaryTag: '{{PrimaryPatchGroupTag}}'
 Script: |-
 def getInstanceStates(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 tag = events['primaryTag']
 tagKey, tagValue = list(tag.items())[0]
 instanceQuery = ec2.describe_instances(
 Filters=[
 {
 "Name": "tag:" + tagKey,
 "Values": [tagValue]
 }
]
)
 if not instanceQuery['Reservations']:
 noInstancesForTagString = "No instances found for specified tag."
 return({ 'noInstancesFound' : noInstancesForTagString })
 else:
 queryResponse = instanceQuery['Reservations']
 originalInstanceState = {}
 for results in queryResponse:
 instanceSet = results['Instances']
 for instance in instanceSet:
 instanceId = instance['InstanceId']
 originalInstanceState[instanceId] = instance['State']['Name']
 return originalInstanceState
 outputs:
 - Name: originalInstanceState
 Selector: $.Payload
 Type: StringMap
 nextStep: verifyPrimaryInstancesRunning
```

## JSON

```

"mainSteps": [
 {
 "name": "getPrimaryInstanceState",
 "action": "aws:executeScript",
 "timeoutSeconds": 120,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "getInstanceStates",
 "InputPayload": {
 "primaryTag": "{{PrimaryPatchGroupTag}}"
 },
 "Script": "..."
 },
 "outputs": [
 {
 "Name": "originalInstanceStates",
 "Selector": "$.Payload",
 "Type": "StringMap"
 }
],
 "nextStep": "verifyPrimaryInstancesRunning"
 },
]

```

- Emily uses the output from the previous action in another `aws:executeScript` action to verify all instances associated with the tag specified in the `PrimaryPatchGroupTag` parameter are in a running state.

If the instance state is already running or shutting-down, the script continues to loop through the remaining instances.

If the instance state is stopping, the script polls for the instance to reach the stopped state and starts the instance.

If the instance state is stopped, the script starts the instance.

## YAML

```

- name: verifyPrimaryInstancesRunning
 action: 'aws:executeScript'
 timeoutSeconds: 600
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: verifyInstancesRunning
 InputPayload:
 targetInstances: '{{getPrimaryInstanceState.originalInstanceStates}}'
 Script: |
 def verifyInstancesRunning(events, context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 instanceDict = events['targetInstances']
 for instance in instanceDict:
 if instanceDict[instance] == 'stopped':
 print("The target instance " + instance + " is stopped. The instance will now be started.")
 ec2.start_instances(
 InstanceIds=[instance]
)

```

```

 elif instanceDict[instance] == 'stopping':
 print("The target instance " + instance + " is stopping. Polling for
instance to reach stopped state.")
 while instanceDict[instance] != 'stopped':
 poll = ec2.get_waiter('instance_stopped')
 poll.wait(
 InstanceIds=[instance]
)
 ec2.start_instances(
 InstanceIds=[instance]
)
 else:
 pass
nextStep: waitForPrimaryRunningInstances

```

JSON

```
{
 "name": "verifyPrimaryInstancesRunning",
 "action": "aws:executeScript",
 "timeoutSeconds": 600,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "verifyInstancesRunning",
 "InputPayload": {
 "targetInstances": "{{getPrimaryInstanceState.originalInstanceStates}}"
 },
 "Script": "...",
 },
 "nextStep": "waitForPrimaryRunningInstances"
},
```

- Emily verifies that all instances associated with the tag specified in the PrimaryPatchGroupTag parameter were started or already in a running state. Then she uses another script to verify that all instances, including those that were started in the previous action, have reached the running state.

YAML

```

- name: waitForPrimaryRunningInstances
 action: 'aws:executeScript'
 timeoutSeconds: 300
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: waitForRunningInstances
 InputPayload:
 targetInstances: '{{getPrimaryInstanceState.originalInstanceStates}}'
 Script: |-
 def waitForRunningInstances(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 instanceDict = events['targetInstances']
 for instance in instanceDict:
 poll = ec2.get_waiter('instance_running')
 poll.wait(
 InstanceIds=[instance]
)
 nextStep: returnPrimaryTagKey

```

## JSON

```
{
 "name": "waitForPrimaryRunningInstances",
 "action": "aws:executeScript",
 "timeoutSeconds": 300,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "waitForRunningInstances",
 "InputPayload": {
 "targetInstances": "{{getPrimaryInstanceState.originalInstanceState}}"
 },
 "Script": "..."
 },
 "nextStep": "returnPrimaryTagKey"
},
```

- Emily uses two more scripts to return individual string values of the key and value of the tag specified in the PrimaryPatchGroupTag parameter. The values returned by these actions allows her to provide values directly to the Targets parameter for the AWS-RunPatchBaseline document. The automation then proceeds with patching the instance with the AWS-RunPatchBaseline document using the aws:runCommand action.

## YAML

```
- name: returnPrimaryTagKey
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: returnTagValues
 InputPayload:
 primaryTag: '{{PrimaryPatchGroupTag}}'
 Script: |-
 def returnTagValues(events,context):
 tag = events['primaryTag']
 tagKey = list(tag)[0]
 stringKey = "tag:" + tagKey
 return {'tagKey' : stringKey}
 outputs:
 - Name: Payload
 Selector: $.Payload
 Type: StringMap
 - Name: primaryPatchGroupKey
 Selector: $.Payload.tagKey
 Type: String
 nextStep: returnPrimaryTagValue
- name: returnPrimaryTagValue
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: returnTagValues
 InputPayload:
 primaryTag: '{{PrimaryPatchGroupTag}}'
 Script: |-
 def returnTagValues(events,context):
 tag = events['primaryTag']
 tagKey = list(tag)[0]
 tagValue = tag[tagKey]
```

```

 return {'tagValue' : tagValue}
outputs:
- Name: Payload
 Selector: $.Payload
 Type: StringMap
- Name: primaryPatchGroupValue
 Selector: $.Payload.tagValue
 Type: String
nextStep: patchPrimaryInstances
- name: patchPrimaryInstances
 action: 'aws:runCommand'
 onFailure: Abort
 timeoutSeconds: 7200
inputs:
 DocumentName: AWS-RunPatchBaseline
 Parameters:
 SnapshotId: '{{SnapshotId}}'
 RebootOption: '{{RebootOption}}'
 Operation: '{{Operation}}'
 Targets:
 - Key: '{{returnPrimaryTagKey.primaryPatchGroupKey}}'
 Values:
 - '{{returnPrimaryTagValue.primaryPatchGroupValue}}'
 MaxConcurrency: 10%
 MaxErrors: 10%
nextStep: returnPrimaryToOriginalState

```

JSON

```
{
 "name": "returnPrimaryTagKey",
 "action": "aws:executeScript",
 "timeoutSeconds": 120,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "returnTagValues",
 "InputPayload": {
 "primaryTag": "{{PrimaryPatchGroupTag}}"
 },
 "Script": "..."
 },
 "outputs": [
 {
 "Name": "Payload",
 "Selector": "$.Payload",
 "Type": "StringMap"
 },
 {
 "Name": "primaryPatchGroupKey",
 "Selector": "$.Payload.tagKey",
 "Type": "String"
 }
],
 "nextStep": "returnPrimaryTagValue"
},
{
 "name": "returnPrimaryTagValue",
 "action": "aws:executeScript",
 "timeoutSeconds": 120,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "returnTagValues",
 "InputPayload": {
 "tagKey": "{{PrimaryPatchGroupKey}}"
 }
 },
 "outputs": [
 {
 "Name": "tagValue",
 "Selector": "$.Payload",
 "Type": "String"
 }
],
 "nextStep": "returnPrimaryToOriginalState"
}
}
```

```

 "InputPayload": {
 "primaryTag": "{{PrimaryPatchGroupTag}}"
 },
 "Script": "...",
},
"outputs": [
{
 "Name": "Payload",
 "Selector": "$.Payload",
 "Type": "StringMap"
},
{
 "Name": "primaryPatchGroupValue",
 "Selector": "$.Payload.tagValue",
 "Type": "String"
}
],
"nextStep": "patchPrimaryInstances"
},
{
 "name": "patchPrimaryInstances",
 "action": "aws:runCommand",
 "onFailure": "Abort",
 "timeoutSeconds": 7200,
 "inputs": {
 "DocumentName": "AWS-RunPatchBaseline",
 "Parameters": {
 "SnapshotId": "{{SnapshotId}}",
 "RebootOption": "{{RebootOption}}",
 "Operation": "{{Operation}}"
 },
 "Targets": [
{
 "Key": "{{returnPrimaryTagKey.primaryPatchGroupKey}}",
 "Values": [
 "{{returnPrimaryTagValue.primaryPatchGroupValue}}"
]
}
],
 "MaxConcurrency": "10%",
 "MaxErrors": "10%"
 },
 "nextStep": "returnPrimaryToOriginalState"
},

```

- After the patching operation completes, Emily wants the automation to return the target instances associated with the tag specified in the PrimaryPatchGroupTag parameter to the same state they were before the automation started. She does this by again using the output from the first action in a script. Based on the original state of the target instance, if the instance was previously in any state other than running, the instance is stopped. Otherwise, if the instance state is running, the script continues to loop through the remaining instances.

#### YAML

```

- name: returnPrimaryToOriginalState
 action: 'aws:executeScript'
 timeoutSeconds: 600
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: returnToOriginalState
 InputPayload:
 targetInstances: '{{getPrimaryInstanceState.originalInstanceState}}'
 Script: |
 def returnToOriginalState(events, context):

```

```

import boto3

#Initialize client
ec2 = boto3.client('ec2')
instanceDict = events['targetInstances']
for instance in instanceDict:
 if instanceDict[instance] == 'stopped' or instanceDict[instance] ==
'stopping':
 ec2.stop_instances(
 InstanceIds=[instance]
)
 else:
 pass
nextStep: getSecondaryInstanceState

```

JSON

```
{
 "name": "returnPrimaryToOriginalState",
 "action": "aws:executeScript",
 "timeoutSeconds": 600,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "returnToOriginalState",
 "InputPayload": {
 "targetInstances": "{{getPrimaryInstanceState.originalInstanceState}}"
 },
 "Script": "...",
 },
 "nextStep": "getSecondaryInstanceState"
},
```

7. The patching operation is completed for the instances associated with the tag specified in the PrimaryPatchGroupTag parameter. Now Emily duplicates all of the previous actions in her runbook content to target the instances associated with the tag specified in the SecondaryPatchGroupTag parameter.

YAML

```

- name: getSecondaryInstanceState
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: getInstanceStates
 InputPayload:
 secondaryTag: '{{SecondaryPatchGroupTag}}'
 Script: |
 def getInstanceStates(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 tag = events['secondaryTag']
 tagKey, tagValue = list(tag.items())[0]
 instanceQuery = ec2.describe_instances(
 Filters=[
 {
 "Name": "tag:" + tagKey,
 "Values": [tagValue]
 }
]
)

```

```

if not instanceQuery['Reservations']:
 noInstancesForTagString = "No instances found for specified tag."
 return({ 'noInstancesFound' : noInstancesForTagString })
else:
 queryResponse = instanceQuery['Reservations']
 originalInstanceState = {}
 for results in queryResponse:
 instanceSet = results['Instances']
 for instance in instanceSet:
 instanceId = instance['InstanceId']
 originalInstanceState[instanceId] = instance['State']['Name']
 return originalInstanceState

outputs:
- Name: originalInstanceState
 Selector: $.Payload
 Type: StringMap
nextStep: verifySecondaryInstancesRunning
- name: verifySecondaryInstancesRunning
 action: 'aws:executeScript'
 timeoutSeconds: 600
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: verifyInstancesRunning
 InputPayload:
 targetInstances: '{{getSecondaryInstanceState.originalInstanceState}}'
 Script: |-
 def verifyInstancesRunning(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 instanceDict = events['targetInstances']
 for instance in instanceDict:
 if instanceDict[instance] == 'stopped':
 print("The target instance " + instance + " is stopped. The instance will now be started.")
 ec2.start_instances(
 InstanceIds=[instance]
)
 elif instanceDict[instance] == 'stopping':
 print("The target instance " + instance + " is stopping. Polling for instance to reach stopped state.")
 while instanceDict[instance] != 'stopped':
 poll = ec2.get_waiter('instance_stopped')
 poll.wait(
 InstanceIds=[instance]
)
 ec2.start_instances(
 InstanceIds=[instance]
)
 else:
 pass
 nextStep: waitForSecondaryRunningInstances
- name: waitForSecondaryRunningInstances
 action: 'aws:executeScript'
 timeoutSeconds: 300
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: waitForRunningInstances
 InputPayload:
 targetInstances: '{{getSecondaryInstanceState.originalInstanceState}}'
 Script: |-
 def waitForRunningInstances(events,context):
 import boto3

```

```

#Initialize client
ec2 = boto3.client('ec2')
instanceDict = events['targetInstances']
for instance in instanceDict:
 poll = ec2.get_waiter('instance_running')
 poll.wait(
 InstanceIds=[instance]
)
nextStep: returnSecondaryTagKey
- name: returnSecondaryTagKey
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: returnTagValues
 InputPayload:
 secondaryTag: '{{SecondaryPatchGroupTag}}'
 Script: |-
 def returnTagValues(events,context):
 tag = events['secondaryTag']
 tagKey = list(tag)[0]
 stringKey = "tag:" + tagKey
 return {'tagKey' : stringKey}
 outputs:
 - Name: Payload
 Selector: $.Payload
 Type: StringMap
 - Name: secondaryPatchGroupKey
 Selector: $.Payload.tagKey
 Type: String
nextStep: returnSecondaryTagValue
- name: returnSecondaryTagValue
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: returnTagValues
 InputPayload:
 secondaryTag: '{{SecondaryPatchGroupTag}}'
 Script: |-
 def returnTagValues(events,context):
 tag = events['secondaryTag']
 tagKey = list(tag)[0]
 tagValue = tag[tagKey]
 return {'tagValue' : tagValue}
 outputs:
 - Name: Payload
 Selector: $.Payload
 Type: StringMap
 - Name: secondaryPatchGroupValue
 Selector: $.Payload.tagValue
 Type: String
nextStep: patchSecondaryInstances
- name: patchSecondaryInstances
 action: 'aws:runCommand'
 onFailure: Abort
 timeoutSeconds: 7200
 inputs:
 DocumentName: AWS-RunPatchBaseline
 Parameters:
 SnapshotId: '{{SnapshotId}}'
 RebootOption: '{{RebootOption}}'
 Operation: '{{Operation}}'

```

```

Targets:
 - Key: '{{returnSecondaryTagKey.secondaryPatchGroupKey}}'
 Values:
 - '{{returnSecondaryTagValue.secondaryPatchGroupValue}}'
 MaxConcurrency: 10%
 MaxErrors: 10%
 nextStep: returnSecondaryToOriginalState
- name: returnSecondaryToOriginalState
 action: 'aws:executeScript'
 timeoutSeconds: 600
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: returnToOriginalState
 InputPayload:
 targetInstances: '{{getSecondaryInstanceState.originalInstanceState}}'
 Script: |-
 def returnToOriginalState(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 instanceDict = events['targetInstances']
 for instance in instanceDict:
 if instanceDict[instance] == 'stopped' or instanceDict[instance] ==
'stopping':
 ec2.stop_instances(
 InstanceIds=[instance]
)
 else:
 pass

```

### JSON

```
{
 "name": "getSecondaryInstanceState",
 "action": "aws:executeScript",
 "timeoutSeconds": 120,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "getInstanceStates",
 "InputPayload": {
 "secondaryTag": "{{SecondaryPatchGroupTag}}"
 },
 "Script": "..."
 },
 "outputs": [
 {
 "Name": "originalInstanceState",
 "Selector": "$.Payload",
 "Type": "StringMap"
 }
],
 "nextStep": "verifySecondaryInstancesRunning"
},
{
 "name": "verifySecondaryInstancesRunning",
 "action": "aws:executeScript",
 "timeoutSeconds": 600,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "verifyInstancesRunning",

```

```

 "InputPayload":{

"targetInstances":"{{getSecondaryInstanceState.originalInstanceState}}"
},
"Script":"..."
},
"nextStep":"waitForSecondaryRunningInstances"
},
{
 "name":"waitForSecondaryRunningInstances",
 "action":"aws:executeScript",
 "timeoutSeconds":300,
 "onFailure":"Abort",
 "inputs":{
 "Runtime":"python3.7",
 "Handler":"waitForRunningInstances",
 "InputPayload":{

"targetInstances":"{{getSecondaryInstanceState.originalInstanceState}}"
},
"Script":"..."
},
"nextStep":"returnSecondaryTagKey"
},
{
 "name":"returnSecondaryTagKey",
 "action":"aws:executeScript",
 "timeoutSeconds":120,
 "onFailure":"Abort",
 "inputs":{
 "Runtime":"python3.7",
 "Handler":"returnTagValues",
 "InputPayload":{
 "secondaryTag":"{{SecondaryPatchGroupTag}}"
},
"Script":"..."
},
"outputs":[
{
 "Name":"Payload",
 "Selector":("$.Payload",
 "Type":"StringMap"
},
{
 "Name":"secondaryPatchGroupKey",
 "Selector":("$.Payload.tagKey",
 "Type":"String"
}
],
"nextStep":"returnSecondaryTagValue"
},
{
 "name":"returnSecondaryTagValue",
 "action":"aws:executeScript",
 "timeoutSeconds":120,
 "onFailure":"Abort",
 "inputs":{
 "Runtime":"python3.7",
 "Handler":"returnTagValues",
 "InputPayload":{
 "secondaryTag":"{{SecondaryPatchGroupTag}}"
},
"Script":"..."
},
"outputs":[
{

```

```

 "Name":"Payload",
 "Selector":("$.Payload",
 "Type":"StringMap"
 },
 {
 "Name":"secondaryPatchGroupValue",
 "Selector":("$.Payload.tagValue",
 "Type":"String"
 }
],
"nextStep":"patchSecondaryInstances"
},
{
 "name":"patchSecondaryInstances",
 "action":"aws:runCommand",
 "onFailure":"Abort",
 "timeoutSeconds":7200,
 "inputs":{
 "DocumentName":"AWS-RunPatchBaseline",
 "Parameters":{
 "SnapshotId":"{{SnapshotId}}",
 "RebootOption":"{{RebootOption}}",
 "Operation":"{{Operation}}"
 },
 "Targets":[
 {
 "Key":"{{returnSecondaryTagKey.secondaryPatchGroupKey}}",
 "Values":[
 "{{returnSecondaryTagValue.secondaryPatchGroupValue}}"
]
 }
],
 "MaxConcurrency":"10%",
 "MaxErrors":"10%"
 },
 "nextStep":"returnSecondaryToOriginalState"
},
{
 "name":"returnSecondaryToOriginalState",
 "action":"aws:executeScript",
 "timeoutSeconds":600,
 "onFailure":"Abort",
 "inputs":{
 "Runtime":"python3.7",
 "Handler":"returnToOriginalState",
 "InputPayload":{

"targetInstances":"{{getSecondaryInstanceState.originalInstanceState}}"
 },
 "Script":"..."
 }
}
]
}

```

- Emily reviews the completed scripted runbook content and creates the runbook in the same AWS account and AWS Region as the target instances. Now she's ready to test her runbook to make sure the automation operates as desired before implementing it into her production environment. The following is the completed scripted runbook content.

YAML

```

description: An example of an Automation runbook that patches groups of Amazon EC2
instances in stages.
schemaVersion: '0.3'

```

```

assumeRole: '{{AutomationAssumeRole}}'
parameters:
 AutomationAssumeRole:
 type: String
 description: '(Required) The Amazon Resource Name (ARN) of the IAM role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to operate this runbook.'
 PrimaryPatchGroupTag:
 type: StringMap
 description: '(Required) The tag for the primary group of instances you want to patch. Specify a key-value pair. Example: {"key" : "value"}'
 SecondaryPatchGroupTag:
 type: StringMap
 description: '(Required) The tag for the secondary group of instances you want to patch. Specify a key-value pair. Example: {"key" : "value"}'
 SnapshotId:
 type: String
 description: '(Optional) The snapshot ID to use to retrieve a patch baseline snapshot.'
 default: ''
 RebootOption:
 type: String
 description: '(Optional) Reboot behavior after a patch Install operation. If you choose NoReboot and patches are installed, the instance is marked as non-compliant until a subsequent reboot and scan.'
 allowedValues:
 - NoReboot
 - RebootIfNeeded
 default: RebootIfNeeded
 Operation:
 type: String
 description: '(Optional) The update or configuration to perform on the instance. The system checks if patches specified in the patch baseline are installed on the instance. The install operation installs patches missing from the baseline.'
 allowedValues:
 - Install
 - Scan
 default: Install
mainSteps:
 - name: getPrimaryInstanceState
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: getInstanceStates
 InputPayload:
 primaryTag: '{{PrimaryPatchGroupTag}}'
 Script: |-
 def getInstanceStates(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 tag = events['primaryTag']
 tagKey, tagValue = list(tag.items())[0]
 instanceQuery = ec2.describe_instances(
 Filters=[{
 {
 "Name": "tag:" + tagKey,
 "Values": [tagValue]
 }
]
)
 if not instanceQuery['Reservations']:
 noInstancesForTagString = "No instances found for specified tag."
 return({ 'noInstancesFound' : noInstancesForTagString })

```

```

else:
 queryResponse = instanceQuery['Reservations']
 originalInstanceState = {}
 for results in queryResponse:
 instanceSet = results['Instances']
 for instance in instanceSet:
 instanceId = instance['InstanceId']
 originalInstanceState[instanceId] = instance['State']['Name']
 return originalInstanceState
outputs:
- Name: originalInstanceState
 Selector: $.Payload
 Type: StringMap
nextStep: verifyPrimaryInstancesRunning
- name: verifyPrimaryInstancesRunning
 action: 'aws:executeScript'
 timeoutSeconds: 600
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: verifyInstancesRunning
 InputPayload:
 targetInstances: '{{getPrimaryInstanceState.originalInstanceState}}'
 Script: |-
 def verifyInstancesRunning(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 instanceDict = events['targetInstances']
 for instance in instanceDict:
 if instanceDict[instance] == 'stopped':
 print("The target instance " + instance + " is stopped. The instance will now be started.")
 ec2.start_instances(
 InstanceIds=[instance]
)
 elif instanceDict[instance] == 'stopping':
 print("The target instance " + instance + " is stopping. Polling for instance to reach stopped state.")
 while instanceDict[instance] != 'stopped':
 poll = ec2.get_waiter('instance_stopped')
 poll.wait(
 InstanceIds=[instance]
)
 ec2.start_instances(
 InstanceIds=[instance]
)
 else:
 pass
 nextStep:waitForPrimaryRunningInstances
- name: waitForPrimaryRunningInstances
 action: 'aws:executeScript'
 timeoutSeconds: 300
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: waitForRunningInstances
 InputPayload:
 targetInstances: '{{getPrimaryInstanceState.originalInstanceState}}'
 Script: |-
 def waitForRunningInstances(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')

```

```

instanceDict = events['targetInstances']
for instance in instanceDict:
 poll = ec2.get_waiter('instance_running')
 poll.wait(
 InstanceIds=[instance]
)
nextStep: returnPrimaryTagKey
- name: returnPrimaryTagKey
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: returnTagValues
 InputPayload:
 primaryTag: '{{PrimaryPatchGroupTag}}'
 Script: |-
 def returnTagValues(events,context):
 tag = events['primaryTag']
 tagKey = list(tag)[0]
 stringKey = "tag:" + tagKey
 return {'tagKey' : stringKey}
 outputs:
 - Name: Payload
 Selector: $.Payload
 Type: StringMap
 - Name: primaryPatchGroupKey
 Selector: $.Payload.tagKey
 Type: String
nextStep: returnPrimaryTagValue
- name: returnPrimaryTagValue
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: returnTagValues
 InputPayload:
 primaryTag: '{{PrimaryPatchGroupTag}}'
 Script: |-
 def returnTagValues(events,context):
 tag = events['primaryTag']
 tagKey = list(tag)[0]
 tagValue = tag[tagKey]
 return {'tagValue' : tagValue}
 outputs:
 - Name: Payload
 Selector: $.Payload
 Type: StringMap
 - Name: primaryPatchGroupValue
 Selector: $.Payload.tagValue
 Type: String
nextStep: patchPrimaryInstances
- name: patchPrimaryInstances
 action: 'aws:runCommand'
 onFailure: Abort
 timeoutSeconds: 7200
 inputs:
 DocumentName: AWS-RunPatchBaseline
 Parameters:
 SnapshotId: '{{SnapshotId}}'
 RebootOption: '{{RebootOption}}'
 Operation: '{{Operation}}'
 Targets:
 - Key: '{{returnPrimaryTagKey.primaryPatchGroupKey}}'
 Values:

```

```

 - '{{returnPrimaryTagValue.primaryPatchGroupValue}}'
 MaxConcurrency: 10%
 MaxErrors: 10%
 nextStep: returnPrimaryToOriginalState
- name: returnPrimaryToOriginalState
 action: 'aws:executeScript'
 timeoutSeconds: 600
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: returnToOriginalState
 InputPayload:
 targetInstances: '{{getPrimaryInstanceState.originalInstanceState}}'
 Script: |-
 def returnToOriginalState(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 instanceDict = events['targetInstances']
 for instance in instanceDict:
 if instanceDict[instance] == 'stopped' or instanceDict[instance] ==
'stopping':
 ec2.stop_instances(
 InstanceIds=[instance]
)
 else:
 pass
 nextStep: getSecondaryInstanceState
- name: getSecondaryInstanceState
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: getInstanceStates
 InputPayload:
 secondaryTag: '{{SecondaryPatchGroupTag}}'
 Script: |-
 def getInstanceStates(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 tag = events['secondaryTag']
 tagKey, tagValue = list(tag.items())[0]
 instanceQuery = ec2.describe_instances(
 Filters=[
 {
 "Name": "tag:" + tagKey,
 "Values": [tagValue]
 }
]
)
 if not instanceQuery['Reservations']:
 noInstancesForTagString = "No instances found for specified tag."
 return({ 'noInstancesFound' : noInstancesForTagString })
 else:
 queryResponse = instanceQuery['Reservations']
 originalInstanceState = {}
 for results in queryResponse:
 instanceSet = results['Instances']
 for instance in instanceSet:
 instanceId = instance['InstanceId']
 originalInstanceState[instanceId] = instance['State']['Name']
 return originalInstanceState
 outputs:

```

```

- Name: originalInstanceState
 Selector: $.Payload
 Type: StringMap
nextStep: verifySecondaryInstancesRunning
- name: verifySecondaryInstancesRunning
 action: 'aws:executeScript'
 timeoutSeconds: 600
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: verifyInstancesRunning
 InputPayload:
 targetInstances: '{{getSecondaryInstanceState.originalInstanceState}}'
 Script: |-
 def verifyInstancesRunning(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 instanceDict = events['targetInstances']
 for instance in instanceDict:
 if instanceDict[instance] == 'stopped':
 print("The target instance " + instance + " is stopped. The instance will now be started.")
 ec2.start_instances(
 InstanceIds=[instance]
)
 elif instanceDict[instance] == 'stopping':
 print("The target instance " + instance + " is stopping. Polling for instance to reach stopped state.")
 while instanceDict[instance] != 'stopped':
 poll = ec2.get_waiter('instance_stopped')
 poll.wait(
 InstanceIds=[instance]
)
 ec2.start_instances(
 InstanceIds=[instance]
)
 else:
 pass
nextStep: waitForSecondaryRunningInstances
- name: waitForSecondaryRunningInstances
 action: 'aws:executeScript'
 timeoutSeconds: 300
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: waitForRunningInstances
 InputPayload:
 targetInstances: '{{getSecondaryInstanceState.originalInstanceState}}'
 Script: |-
 def waitForRunningInstances(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 instanceDict = events['targetInstances']
 for instance in instanceDict:
 poll = ec2.get_waiter('instance_running')
 poll.wait(
 InstanceIds=[instance]
)
nextStep: returnSecondaryTagKey
- name: returnSecondaryTagKey
 action: 'aws:executeScript'
 timeoutSeconds: 120

```

```

onFailure: Abort
inputs:
 Runtime: python3.7
 Handler: returnTagValues
 InputPayload:
 secondaryTag: '{{SecondaryPatchGroupTag}}'
 Script: |-
 def returnTagValues(events,context):
 tag = events['secondaryTag']
 tagKey = list(tag)[0]
 stringKey = "tag:" + tagKey
 return {'tagKey' : stringKey}
outputs:
 - Name: Payload
 Selector: $.Payload
 Type: StringMap
 - Name: secondaryPatchGroupKey
 Selector: $.Payload.tagKey
 Type: String
nextStep: returnSecondaryTagValue
- name: returnSecondaryTagValue
 action: 'aws:executeScript'
 timeoutSeconds: 120
 onFailure: Abort
 inputs:
 Runtime: python3.7
 Handler: returnTagValues
 InputPayload:
 secondaryTag: '{{SecondaryPatchGroupTag}}'
 Script: |-
 def returnTagValues(events,context):
 tag = events['secondaryTag']
 tagKey = list(tag)[0]
 tagValue = tag[tagKey]
 return {'tagValue' : tagValue}
outputs:
 - Name: Payload
 Selector: $.Payload
 Type: StringMap
 - Name: secondaryPatchGroupValue
 Selector: $.Payload.tagValue
 Type: String
nextStep: patchSecondaryInstances
- name: patchSecondaryInstances
 action: 'aws:runCommand'
 onFailure: Abort
 timeoutSeconds: 7200
 inputs:
 DocumentName: AWS-RunPatchBaseline
 Parameters:
 SnapshotId: '{{SnapshotId}}'
 RebootOption: '{{RebootOption}}'
 Operation: '{{Operation}}'
 Targets:
 - Key: '{{returnSecondaryTagKey.secondaryPatchGroupKey}}'
 Values:
 - '{{returnSecondaryTagValue.secondaryPatchGroupValue}}'
 MaxConcurrency: 10%
 MaxErrors: 10%
nextStep: returnSecondaryToOriginalState
- name: returnSecondaryToOriginalState
 action: 'aws:executeScript'
 timeoutSeconds: 600
 onFailure: Abort
 inputs:
 Runtime: python3.7

```

```

Handler: returnToOriginalState
InputPayload:
 targetInstances: '{{getSecondaryInstanceState.originalInstanceStates}}'
Script: |-
 def returnToOriginalState(events,context):
 import boto3

 #Initialize client
 ec2 = boto3.client('ec2')
 instanceDict = events['targetInstances']
 for instance in instanceDict:
 if instanceDict[instance] == 'stopped' or instanceDict[instance] ==
'stopping':
 ec2.stop_instances(
 InstanceIds=[instance]
)
 else:
 pass

```

#### JSON

```
{
 "description": "An example of an Automation runbook that patches groups of Amazon EC2 instances in stages.",
 "schemaVersion": "0.3",
 "assumeRole": "{{AutomationAssumeRole}}",
 "parameters": {
 "AutomationAssumeRole": {
 "type": "String",
 "description": "(Required) The Amazon Resource Name (ARN) of the IAM role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to operate this runbook."
 },
 "PrimaryPatchGroupTag": {
 "type": "StringMap",
 "description": "(Required) The tag for the primary group of instances you want to patch. Specify a key-value pair. Example: {\\"key\\": \\"value\\"}"
 },
 "SecondaryPatchGroupTag": {
 "type": "StringMap",
 "description": "(Required) The tag for the secondary group of instances you want to patch. Specify a key-value pair. Example: {\\"key\\": \\"value\\"}"
 },
 "SnapshotId": {
 "type": "String",
 "description": "(Optional) The snapshot ID to use to retrieve a patch baseline snapshot.",
 "default": ""
 },
 "RebootOption": {
 "type": "String",
 "description": "(Optional) Reboot behavior after a patch Install operation. If you choose NoReboot and patches are installed, the instance is marked as non-compliant until a subsequent reboot and scan.",
 "allowedValues": [
 "NoReboot",
 "RebootIfNeeded"
],
 "default": "RebootIfNeeded"
 },
 "Operation": {
 "type": "String",

```

```

 "description": "(Optional) The update or configuration to perform on the
instance. The system checks if patches specified in the patch baseline are installed
on the instance. The install operation installs patches missing from the baseline.",
 "allowedValues": [
 "Install",
 "Scan"
],
 "default": "Install"
},
"mainSteps": [
{
 "name": "getPrimaryInstanceState",
 "action": "aws:executeScript",
 "timeoutSeconds": 120,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "getInstanceStates",
 "InputPayload": {
 "primaryTag": "{{PrimaryPatchGroupTag}}"
 },
 "Script": "..."
 },
 "outputs": [
 {
 "Name": "originalInstanceState",
 "Selector": "$.Payload",
 "Type": "StringMap"
 }
],
 "nextStep": "verifyPrimaryInstancesRunning"
},
{
 "name": "verifyPrimaryInstancesRunning",
 "action": "aws:executeScript",
 "timeoutSeconds": 600,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "verifyInstancesRunning",
 "InputPayload": {
 "targetInstances": "{{getPrimaryInstanceState.originalInstanceState}}"
 },
 "Script": "..."
 },
 "nextStep": "waitForPrimaryRunningInstances"
},
{
 "name": "waitForPrimaryRunningInstances",
 "action": "aws:executeScript",
 "timeoutSeconds": 300,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "waitForRunningInstances",
 "InputPayload": {
 "targetInstances": "{{getPrimaryInstanceState.originalInstanceState}}"
 },
 "Script": "..."
 },
 "nextStep": "returnPrimaryTagKey"
},
{
 "name": "returnPrimaryTagKey",
 "action": "aws:executeScript",

```

```

 "timeoutSeconds":120,
 "onFailure":"Abort",
 "inputs":{
 "Runtime":"python3.7",
 "Handler":"returnTagValues",
 "InputPayload":{
 "primaryTag":"{{PrimaryPatchGroupTag}}"
 },
 "Script": "..."
 },
 "outputs": [
 {
 "Name": "Payload",
 "Selector": "$.Payload",
 "Type": "StringMap"
 },
 {
 "Name": "primaryPatchGroupKey",
 "Selector": "$.Payload.tagKey",
 "Type": "String"
 }
],
 "nextStep": "returnPrimaryTagValue"
},
{
 "name": "returnPrimaryTagValue",
 "action": "aws:executeScript",
 "timeoutSeconds":120,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "returnTagValues",
 "InputPayload": {
 "primaryTag": "{{PrimaryPatchGroupTag}}"
 },
 "Script": "..."
 },
 "outputs": [
 {
 "Name": "Payload",
 "Selector": "$.Payload",
 "Type": "StringMap"
 },
 {
 "Name": "primaryPatchGroupValue",
 "Selector": "$.Payload.tagValue",
 "Type": "String"
 }
],
 "nextStep": "patchPrimaryInstances"
},
{
 "name": "patchPrimaryInstances",
 "action": "aws:runCommand",
 "onFailure": "Abort",
 "timeoutSeconds": 7200,
 "inputs": {
 "DocumentName": "AWS-RunPatchBaseline",
 "Parameters": {
 "SnapshotId": "{{SnapshotId}}",
 "RebootOption": "{{RebootOption}}",
 "Operation": "{{Operation}}"
 },
 "Targets": [
 {
 "Key": "{{returnPrimaryTagKey.primaryPatchGroupKey}}"
 }
]
 }
}

```

```

 "Values":[
 "{{returnPrimaryTagValue.primaryPatchGroupValue}}"
]
 }
],
"MaxConcurrency":"10%",
"MaxErrors":"10%"
},
"nextStep":"returnPrimaryToOriginalState"
},
{
 "name":"returnPrimaryToOriginalState",
 "action":"aws:executeScript",
 "timeoutSeconds":600,
 "onFailure":"Abort",
 "inputs":{
 "Runtime":"python3.7",
 "Handler":"returnToOriginalState",
 "InputPayload":{
 "targetInstances":"{{getPrimaryInstanceState.originalInstanceState}}"
 },
 "Script":"..."
 },
 "nextStep":"getSecondaryInstanceState"
},
{
 "name":"getSecondaryInstanceState",
 "action":"aws:executeScript",
 "timeoutSeconds":120,
 "onFailure":"Abort",
 "inputs":{
 "Runtime":"python3.7",
 "Handler":"getInstanceStates",
 "InputPayload":{
 "secondaryTag":"{{SecondaryPatchGroupTag}}"
 },
 "Script":"..."
 },
 "outputs":[
 {
 "Name":"originalInstanceState",
 "Selector":("$.Payload",
 "Type":"StringMap"
 }
],
 "nextStep":"verifySecondaryInstancesRunning"
},
{
 "name":"verifySecondaryInstancesRunning",
 "action":"aws:executeScript",
 "timeoutSeconds":600,
 "onFailure":"Abort",
 "inputs":{
 "Runtime":"python3.7",
 "Handler":"verifyInstancesRunning",
 "InputPayload":{

"targetInstances":"{{getSecondaryInstanceState.originalInstanceState}}"
 },
 "Script":"..."
 },
 "nextStep":"waitForSecondaryRunningInstances"
},
{
 "name":"waitForSecondaryRunningInstances",
 "action":"aws:executeScript",

```

```
"timeoutSeconds":300,
"OnFailure":"Abort",
"inputs": {
 "Runtime":"python3.7",
 "Handler":"waitForRunningInstances",
 "InputPayload": {

"targetInstances": "{{getSecondaryInstanceState.originalInstanceState}}"
 },
 "Script": "..."
},
"nextStep": "returnSecondaryTagKey"
},
{
 "name": "returnSecondaryTagKey",
 "action": "aws:executeScript",
 "timeoutSeconds": 120,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "returnTagValues",
 "InputPayload": {
 "secondaryTag": "{{SecondaryPatchGroupTag}}"
 },
 "Script": ...
 },
 "outputs": [
 {
 "Name": "Payload",
 "Selector": "$.Payload",
 "Type": "StringMap"
 },
 {
 "Name": "secondaryPatchGroupKey",
 "Selector": "$.Payload.tagKey",
 "Type": "String"
 }
],
 "nextStep": "returnSecondaryTagValue"
},
{
 "name": "returnSecondaryTagValue",
 "action": "aws:executeScript",
 "timeoutSeconds": 120,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "returnTagValues",
 "InputPayload": {
 "secondaryTag": "{{SecondaryPatchGroupTag}}"
 },
 "Script": ...
 },
 "outputs": [
 {
 "Name": "Payload",
 "Selector": "$.Payload",
 "Type": "StringMap"
 },
 {
 "Name": "secondaryPatchGroupValue",
 "Selector": "$.Payload.tagValue",
 "Type": "String"
 }
],
 "nextStep": "patchSecondaryInstances"
}
```

```

},
{
 "name": "patchSecondaryInstances",
 "action": "aws:runCommand",
 "onFailure": "Abort",
 "timeoutSeconds": 7200,
 "inputs": {
 "DocumentName": "AWS-RunPatchBaseline",
 "Parameters": {
 "SnapshotId": "{{SnapshotId}}",
 "RebootOption": "{{RebootOption}}",
 "Operation": "{{Operation}}"
 },
 "Targets": [
 {
 "Key": "{{returnSecondaryTagKey.secondaryPatchGroupKey}}",
 "Values": [
 "{{returnSecondaryTagValue.secondaryPatchGroupValue}}"
]
 }
],
 "MaxConcurrency": "10%",
 "MaxErrors": "10%"
 },
 "nextStep": "returnSecondaryToOriginalState"
},
{
 "name": "returnSecondaryToOriginalState",
 "action": "aws:executeScript",
 "timeoutSeconds": 600,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.7",
 "Handler": "returnToOriginalState",
 "InputPayload": {
 "targetInstances": "{{getSecondaryInstanceState.originalInstanceState}}"
 },
 "Script": "..."
 }
}
]
}
}

```

For more information about the automation actions used in this example, see the [Systems Manager Automation actions reference \(p. 477\)](#).

## Walkthrough: Using Document Builder to create a custom runbook

The following walkthrough shows how to use Document Builder in the AWS Systems Manager Automation console to create a custom runbook and then run the custom runbook. Automation is a capability of AWS Systems Manager.

The first step of the runbook you create runs a script to launch an Amazon Elastic Compute Cloud (Amazon 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.

### 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 runbooks \(p. 401\)](#).
  - 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 automations \(p. 401\)](#).  
For information about the IAM service role requirement for running `aws:executeScript`, see [Permissions for using runbooks \(p. 545\)](#).
- Verify that you have permission to launch EC2 instances.  
For information, see [IAM and Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.

#### Topics

- [Step 1: Create the custom runbook \(p. 649\)](#)
- [Step 2: Run the custom runbook \(p. 653\)](#)

### Step 1: Create the custom runbook

Use the following procedure to create a custom runbook that launches an Amazon 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 runbook using Document Builder

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 runbook: `LaunchInstanceAndCheckStatus`.
5. (Optional) For **Document description**, replace the default text with a description for this runbook, using Markdown. The following is an example.

```
##Title: LaunchInstanceAndCheckState

Purpose: This runbook first launches an EC2 instance using the AMI ID provided in the parameter ```imageId```. The second step of this runbook continuously checks the instance status check value for the launched instance until the status ```ok``` is returned.

##Parameters:

Name | Type | Description | Default Value
----- | ----- | ----- | -----
assumeRole | String | (Optional) The ARN of the role that allows Automation to perform the actions on your behalf. | -
```

```
imageId | String | (Optional) The AMI ID to use for launching the instance. The default value uses the latest Amazon Linux AMI ID available. | {{ ssm:/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2 }}
```

6. For **Assume role**, enter the ARN of the IAM service role for Automation (Assume role) for the automation, 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.

**Important**

For runbooks not owned by Amazon that use the `aws:executeScript` action, a role must be specified. For information, see [Permissions for using runbooks \(p. 545\)](#).

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 Amazon 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 runbook. 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: **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

```
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 runbook. 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:

**WaitForInstanceStateOk**.

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**

```
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]['InstanceState']['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

```
Install-Module AWS.Tools.EC2 -Force

$inputPayload = $env:InputPayload | ConvertFrom-Json

$instanceId = $inputPayload.payload.InstanceId

$status = Get-EC2InstanceState -InstanceId $instanceId

while ($status.Status -ne 'ok'){
 Write-Host 'Polling get status of the instance', $instanceId

 Start-Sleep -Seconds 5

 $status = Get-EC2InstanceState -InstanceId $instanceId
}

return @{Status = $status.Status; InstanceId = $instanceId}
```

7. Expand **Additional inputs**.

8. For **Input name**, choose **InputPayload**. For **Input value**, enter the following:

```
{{ LaunchEc2Instance.payload }}
```

16. Choose **Create automation** to save the runbook.

## Step 2: Run the custom runbook

Use the following procedure to run the custom runbook created in Step 1. The custom runbook launches an EC2 instance and waits for the instance check to change to the **ok** status.

### To run the custom runbook

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 runbook 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 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*.

You can identify the instance by the name **LaunchedBySsmAutomation** that you tagged it with in [Step 1: Create the custom runbook \(p. 649\)](#).

## Patching Amazon Machine Images

This section includes walkthroughs that describe how to patch or update Amazon Machine Image (AMIs).

### Topics

- [Walkthrough: Patch a Linux AMI \(console\) \(p. 654\)](#)
- [Walkthrough: Patch a Linux AMI \(AWS CLI\) \(p. 657\)](#)
- [Walkthrough: Patch a Windows Server AMI \(p. 661\)](#)
- [Walkthrough: Simplify AMI patching using Automation, AWS Lambda, and Parameter Store \(p. 665\)](#)
- [Walkthrough: Patch an AMI and update an Auto Scaling group \(p. 671\)](#)

### Walkthrough: Patch a Linux AMI (console)

This Systems Manager Automation walkthrough shows you how to use the console and the Systems Manager `AWS-UpdateLinuxAmi` runbook to automatically patch a Linux AMI with the latest versions of packages that you specify. Automation is a capability of AWS Systems Manager. The `AWS-UpdateLinuxAmi` runbook 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 \(p. 1094\)](#).

The `AWS-UpdateLinuxAmi` runbook enables you to automate image-maintenance tasks without having to author the runbook in JSON or YAML. You can use the `AWS-UpdateLinuxAmi` runbook to perform the following types of tasks.

- Upgrade all distribution packages and Amazon software on an Amazon Linux, Red Hat Enterprise Linux, Ubuntu Server, SUSE Linux Enterprise Server, or CentOS Amazon Machine Image (AMI). This is the default runbook behavior.
- Install AWS Systems Manager SSM Agent on an existing image to enable Systems Manager capabilities, such as running remote commands using AWS Systems Manager Run Command or software inventory collection using Inventory.
- Install additional software packages.

### Before you begin

Before you begin working with runbooks, configure roles and, optionally, EventBridge for Automation. For more information, see [Setting up Automation \(p. 401\)](#). 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 \(p. 21\)](#).

The `AWS-UpdateLinuxAmi` runbook accepts the following input parameters.

Parameter	Type	Description
<code>SourceAmiId</code>	String	(Required) The source AMI ID.
<code>IamInstanceProfileName</code>	String	(Required) The name of the IAM instance profile role you created in <a href="#">Create an IAM instance profile for Systems Manager (p. 21)</a> . The instance profile role gives Automation permission to perform actions on your instances, such as running commands or starting and stopping services. The runbook uses only the name of the instance profile role. If you specify the Amazon Resource Name (ARN), the automation fails.
<code>AutomationAssumeRole</code>	String	(Required) The name of the IAM service role you created in <a href="#">Setting up Automation (p. 401)</a> . 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 a runbook. For this parameter, the complete ARN must be specified.
<code>TargetAmiName</code>	String	(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.
<code>InstanceType</code>	String	(Optional) The type of instance to launch as the workspace host. Instance types vary by region. The default type is <code>t2.micro</code> .
<code>PreUpdateScript</code>	String	(Optional) URL of a script to run before updates are applied.

Parameter	Type	Description
		Default ("none") is to not run a script.
PostUpdateScript	String	(Optional) URL of a script to run after package updates are applied. Default ("none") is to not run a script.
IncludePackages	String	(Optional) Only update these named packages. By default ("all"), all available updates are applied.
ExcludePackages	String	(Optional) Names of packages to hold back from updates, under all conditions. By default ("none"), no package is excluded.

## Automation Steps

The `AWS-UpdateLinuxAmi` runbook includes the following automation actions, by default.

### Step 1: `launchInstance` (`aws:runInstances` action)

This step launches an instance using Amazon Elastic Compute Cloud (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 automation by terminating the running instance.

#### Output

The automation returns the new AMI ID as output.

#### Note

By default, when Automation runs the `AWS-UpdateLinuxAmi` runbook, 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` runbook and specify a subnet ID. For more information, see [VPC not defined 400 \(p. 691\)](#).

## To create a patched AMI using Automation (AWS Systems Manager)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/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 automation finishes, launch a test instance from the updated AMI to verify changes.

#### Note

If any step in the automation fails, information about the failure is listed on the **Automation Executions** page. The automation 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 AWS Systems Manager Automation walkthrough shows you how to use the AWS Command Line Interface (AWS CLI) and the Systems Manager `AWS-UpdateLinuxAmi` runbook to automatically patch a Linux Amazon Machine Image (AMI) with the latest versions of packages that you specify. Automation is a capability of AWS Systems Manager. The `AWS-UpdateLinuxAmi` runbook 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 \(p. 1094\)](#).

The `AWS-UpdateLinuxAmi` runbook enables you to automate image-maintenance tasks without having to author the runbook in JSON or YAML. You can use the `AWS-UpdateLinuxAmi` runbook to perform the following types of tasks.

- Upgrade all distribution packages and Amazon software on an Amazon Linux, Red Hat, Ubuntu, SLES, or Cent OS Amazon Machine Image (AMI). This is the default runbook behavior.
- Install AWS Systems Manager SSM Agent on an existing image to enable Systems Manager capabilities, such as running remote commands using AWS Systems Manager Run Command or software inventory collection using Inventory.
- Install additional software packages.

### Before you begin

Before you begin working with runbooks, configure roles and, optionally, EventBridge for Automation. For more information, see [Setting up Automation \(p. 401\)](#). 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 \(p. 21\)](#).

The `AWS-UpdateLinuxAmi` runbook accepts the following input parameters.

Parameter	Type	Description
SourceAmiId	String	(Required) The source AMI ID. You can automatically reference the latest ID of an Amazon EC2 AMI for Linux by using a AWS Systems Manager Parameter Store <i>public</i> parameter. For more information, see <a href="#">Query for the latest Amazon Linux AMI IDs using AWS Systems Manager Parameter Store</a> .
IamInstanceProfileName	String	(Required) The name of the IAM instance profile role you created in <a href="#">Create an IAM instance profile for Systems Manager (p. 21)</a> . The instance profile role gives Automation permission to perform actions on your instances, such as running commands or starting and stopping services. The runbook uses only the name of the instance profile role.
AutomationAssumeRole	String	(Required) The name of the IAM service role you created in <a href="#">Setting up Automation (p. 401)</a> . 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 a

Parameter	Type	Description
		runbook. For this parameter, the complete ARN must be specified.
TargetAmiName	String	(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.
InstanceType	String	(Optional) The type of instance to launch as the workspace host. Instance types vary by Region. The default type is t2.micro.
PreUpdateScript	String	(Optional) URL of a script to run before updates are applied. Default ("none") is to not run a script.
PostUpdateScript	String	(Optional) URL of a script to run after package updates are applied. Default ("none") is to not run a script.
IncludePackages	String	(Optional) Only update these named packages. By default ("all"), all available updates are applied.
ExcludePackages	String	(Optional) Names of packages to hold back from updates, under all conditions. By default ("none"), no package is excluded.

## Automation Steps

The `AWS-UpdateLinuxAmi` runbook includes the following steps, by default.

### Step 1: `launchInstance` (`aws:runInstances` action)

This step launches an instance using Amazon Elastic Compute Cloud (Amazon EC2) user data and an IAM instance profile role. User data 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 Simple Storage Service (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 automation by terminating the running instance.

#### Output

The automation returns the new AMI ID as output.

#### Note

By default, when Automation runs the `AWS-UpdateLinuxAmi` runbook, 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` runbook and specify a subnet ID. For more information, see [VPC not defined 400 \(p. 691\)](#).

### To create a patched AMI using Automation

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to run the `AWS-UpdateLinuxAmi` runbook. Replace each *example resource placeholder* with your own information.

```
aws ssm start-automation-execution \
 --document-name "AWS-UpdateLinuxAmi" \
 --parameters \
 SourceAmiId=AMI ID, \
 IamInstanceProfileName=IAM instance profile, \
 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 automation.

```
{ \
 "AutomationExecutionId": "automation execution ID" \
}
```

3. To view the automation using the AWS CLI, run the following command:

```
aws ssm describe-automation-executions
```

4. To view details about the automation progress, run the following command. Replace *automation execution ID* with your own information.

```
aws ssm get-automation-execution --automation-execution-id automation execution ID
```

The update process can take 30 minutes or more to complete.

**Note**

You can also monitor the status of the automation in the console. In the list, choose the automation you just ran and then choose the **Steps** tab. This tab shows you the status of the automation actions.

After the automation finishes, launch a test instance from the updated AMI to verify changes.

**Note**

If any step in the automation fails, information about the failure is listed on the **Automation Executions** page. The automation 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 runbook enables you to automate image maintenance tasks on your Amazon Windows Amazon Machine Image (AMI) without having to author the runbook in JSON or YAML. This runbook is supported for Windows Server 2008 R2 or later. You can use the AWS-UpdateWindowsAmi runbook 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 runbooks, [configure roles for Automation \(p. 403\)](#) to add an `iam:PassRole` policy that references the ARN of the instance profile you want to grant access to. Optionally, configure Amazon EventBridge for Automation, a capability of AWS Systems Manager. For more information, see [Setting up Automation \(p. 401\)](#). 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 \(p. 21\)](#).

**Note**

Updates to AWS Systems Manager 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 runbook accepts the following input parameters.

Parameter	Type	Description
SourceAmiId	String	(Required) The source AMI ID. You can automatically reference the latest Windows Server AMI ID by using a

Parameter	Type	Description
		Systems Manager Parameter Store <i>public</i> parameter. For more information, see <a href="#">Query for the latest Windows AMI IDs using AWS Systems ManagerParameter Store</a> .
IamInstanceProfileName	String	(Required) The name of the IAM instance profile role you created in <a href="#">Create an IAM instance profile for Systems Manager (p. 21)</a> . The instance profile role gives Automation permission to perform actions on your instances, such as running commands or starting and stopping services. The runbook uses only the name of the instance profile role.
AutomationAssumeRole	String	(Required) The name of the IAM service role you created in <a href="#">Setting up Automation (p. 401)</a> . 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 a runbook. For this parameter, the complete ARN must be specified.
TargetAmiName	String	(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.
InstanceType	String	(Optional) The type of instance to launch as the workspace host. Instance types vary by region. The default type is <code>t2.medium</code> .
PreUpdateScript	String	(Optional) A script to run before updating the AMI. Enter a script in the runbook or at runtime as a parameter.
PostUpdateScript	String	(Optional) A script to run after updating the AMI. Enter a script in the runbook or at runtime as a parameter.

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 runbook includes the following 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 runbook 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 runbook.

### **Step 5: updateAWSPVDriver (aws : runCommand action)**

This step updates AWS PV drivers by using the AWS-ConfigureAWS.Package runbook.

### **Step 6: updateAwsEnaNetworkDriver (aws : runCommand action)**

This step updates AWS ENA Network drivers by using the AWS-ConfigureAWS.Package runbook.

### **Step 7: installWindowsUpdates (aws : runCommand action)**

This step installs Windows updates by using the AWS-InstallWindowsUpdates runbook. 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 runbook 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 automation 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 automation.

### **Note**

By default, when Automation runs the AWS-UpdateWindowsAmi runbook 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` runbook and specify a subnet ID. For more information, see [VPC not defined 400 \(p. 691\)](#).

### To create a patched Windows AMI by using Automation

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to run the `AWS-UpdateWindowsAmi` runbook. Replace each *example resource placeholder* with your own information. 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.

```
aws ssm start-automation-execution \
 --document-name="AWS-UpdateWindowsAmi" \
 --parameters SourceAmiId='AMI ID',IamInstanceProfileName='IAM
instance profile',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 automation.

```
{ "AutomationExecutionId": "automation execution ID" }
```

3. To view the automation using the AWS CLI, run the following command:

```
aws ssm describe-automation-executions
```

4. To view details about the automation progress, run the following command.

```
aws ssm get-automation-execution
--automation-execution-id automation execution ID
```

#### Note

Depending on the number of patches applied, the Windows patching process run in this sample automation 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 \(p. 661\)](#). This example uses the model where an organization maintains and periodically patches their own, proprietary AMIs rather than building from Amazon Elastic Compute Cloud (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 AWS 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. 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. Parameter Store and Automation are capabilities of AWS Systems Manager.

## Before you begin

Configure Automation roles and, optionally, Amazon EventBridge for Automation. For more information, see [Setting up Automation \(p. 401\)](#).

## Contents

- [Task 1: Create a parameter in Systems Manager Parameter Store \(p. 666\)](#)
- [Task 2: Create an IAM role for AWS Lambda \(p. 666\)](#)
- [Task 3: Create an AWS Lambda function \(p. 667\)](#)
- [Task 4: Create a runbook and patch the AMI \(p. 669\)](#)

## 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 \(p. 279\)](#).

## Task 2: Create an IAM role for AWS Lambda

Use the following procedure to create an IAM service role for AWS Lambda. 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 each *example resource placeholder* with your own information.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "logs:CreateLogGroup",
 "Resource": "arn:aws:logs:region:123456789012:*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "logs:CreateLogStream",
 "logs:PutLogEvents"
],
 "Resource": [
 "arn:aws:logs:region:123456789012:log-group:/aws/lambda/function name:*"
]
 }
]
}
```

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 each *example resource placeholder* with your own information.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "ssm:PutParameter",
 "Resource": "arn:aws:ssm:region:123456789012:parameter/latestAmi"
 },
 {
 "Effect": "Allow",
 "Action": "ssm:DescribeParameters",
 "Resource": "*"
 }
]
}
```

10. Choose **Review policy**.
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.

```
from __future__ import print_function

import json
import boto3

print('Loading function')

#Updates an SSM parameter
#Expects parameterName, parameterValue
def lambda_handler(event, context):
 print("Received event: " + json.dumps(event, indent=2))

 # get SSM client
 client = boto3.client('ssm')

 #confirm parameter exists before updating it
 response = client.describe_parameters(
 Filters=[
 {
 'Key': 'Name',
 'Values': [event['parameterName']]
 },
]
)

 if not response['Parameters']:
 print('No such parameter')
 return 'SSM parameter not found.'

 #if parameter has a Description field, update it PLUS the Value
 if 'Description' in response['Parameters'][0]:
 description = response['Parameters'][0]['Description']

 response = client.put_parameter(
 Name=event['parameterName'],
 Value=event['parameterValue'],
 Description=description,
 Type='String',
 Overwrite=True
)

 #otherwise just update Value
 else:
 response = client.put_parameter(
 Name=event['parameterName'],
 Value=event['parameterValue'],
 Type='String',
 Overwrite=True
)

 reponseString = 'Updated parameter %s with value %s.' % (event['parameterName'],
 event['parameterValue'])

 return reponseString
```

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. Replace **AMI ID** with your own information to set your `latestAmi` parameter value.

```
{
 "parameterName": "latestAmi",
 "parameterValue": "AMI ID"
}
```

14. Choose **Create**.
15. Choose **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”.

#### Task 4: Create a runbook and patch the AMI

Use the following procedure to create and run a runbook that patches the AMI you specified for the **latestAmi** parameter. After the automation completes, the value of **latestAmi** is updated with the ID of the newly-patched AMI. Subsequent automations use the AMI created by the previous execution.

##### To create a runbook and patch an AMI

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **UpdateMyLatestWindowsAmi**.
5. Choose the **Editor** tab, and then choose **Edit**.
6. In the following example, replace each **example resource placeholder** with your own information.

```
{
 "description": "Systems Manager Automation Demo - Patch AMI and Update SSM Param",
 "schemaVersion": "0.3",
 "assumeRole": "IAM service role",
 "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",
 "region": "us-east-1"
 }
]
}
```

```

 "timeoutSeconds":1200,
 "maxAttempts":1,
 "onFailure":"Abort",
 "inputs":{
 "ImageId":"{{ sourceAMIid }}",
 "InstanceType":"m3.large",
 "MinInstanceCount":1,
 "MaxInstanceCount":1,
 "IamInstanceProfileName":"IAM instance profile"
 }
},
{
 "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,

```

```
 "maxAttempts":1,
 "onFailure":"Abort",
 "inputs":{
 "FunctionName":"Automation-UpdateSsmParam",
 "Payload":"{\\"parameterName\":\"latestAmi\", \\"parameterValue\":"
 \\"{{createImage.ImageId}}\""
 }
],
 "outputs":[
 "createImage.ImageId"
]
}
```

7. Choose **Create automation** to save the runbook.
8. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
9. In the **Choose document** page, choose the **Owned by me** tab, and then select the button in the **UpdateMyLatestWindowsAmi** card.
10. In the **Document details** section, verify that **Document version** is set to **1 (Default)**.
11. Choose **Next**.
12. Choose **Simple execution**.
13. Choose **Execute**.
14. After the automation completes, choose **Parameter Store** in the navigation pane and confirm that the new value for `latestAmi` matches the value returned by the automation. You can also verify the new AMI ID matches the Automation output in the **AMIs** section of the Amazon EC2 console.

## 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 \(p. 665\)](#) 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 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, a capability of AWS Systems Manager. Systems Manager requires an instance profile role and a service role ARN to process automations. For more information, see [Setting up Automation \(p. 401\)](#).
- 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, choose **AWS Service**.
4. In the **Choose a use case** section, choose **Lambda**, and then choose **Next: Permissions**.
5. On the **Attach permissions policies** 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: 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, verify that **AWSLambdaExecute** and **AutoScalingFullAccess** are listed under **Policies**.



9. Type a name in the **Role name** box, and then type a description.
10. 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 **Function name** field, enter **Automation-UpdateAsg**.
5. In the **Runtime** list, choose **Python 3.7**.
6. Expand **Change default execution role** under **Permissions**, verify that **Use an existing role** is selected.
7. In the **Existing role** list, choose the role you created earlier.
8. Choose **Create function**. The system displays a code and configuration page for Automation-UpdateAsg.
9. Make no changes in the **Designer** section.
10. In the **Code source** section, delete the pre-populated code in the **lambda\_function** field, and then paste the following code example.

The image shows a screenshot of the AWS Lambda 'Function code' editor. The code entry type is set to 'Edit code inline'. The runtime is 'Python 2.7' and the handler is 'lambda\_function.lambda\_handler'. The code editor contains the following Python code:

```
lambda_function
1 from __future__ import print_function
2
3 import json
4 import datetime
5 import time
6 import boto3
```

```

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 **Deploy**.
14. Choose **Test**.
15. In the **Configure test event** page, verify that **Create new test event** is selected.
16. In the **Event template** list, verify that **Hello World** is selected.
17. In the **Event name** field, enter a name.
18. Replace the existing sample with the following JSON. Enter an AMI ID and Auto Scaling group.

```
{
 "newAmiID": "AMI ID",
 "targetASG": "name of your Auto Scaling group"
}
```

19. Choose **Create**.
20. Choose **Test**. The output states that the Auto Scaling group was successfully updated with a new launch configuration.

### Task 3: Create a runbook, patch the AMI, and update the Auto Scaling group

Use the following procedure to create and run a runbook that patches the AMI you specified for the **latestAmi** parameter. The automation then updates the Auto Scaling group to use the latest, patched AMI.

#### To create and run the runbook

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Create document** dropdown, choose **Automation**.
4. In the **Name** field, enter **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 runbook 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 [Setting up Automation \(p. 401\)](#).

```
{
 "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,
 "variables": {
 "amiId": "$sourceAMIid",
 "instanceType": "t2.micro",
 "subnetId": "$subnetId",
 "count": 1
 }
 }
]
}
```

```

 "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\":\"{{targetASG}}\", \"newAmiID\":"
 \"{{createImage.ImageId}}\""
 }
],
 "outputs":[
 "createImage.ImageId"
]
}
```

8. Choose **Create automation** to save the runbook.
9. Choose **Automation**, and then choose **Execute automation**.
10. In the **Automation document** list, choose **PatchAmiandUpdateAsg**.
11. Choose **Next**.
12. In the **Document details** section, verify that **Document version** is set to **1**.
13. In the **Execution mode** section, choose **Execute the entire automation at once**.
14. Leave the **Targets and Rate Control** option disabled.
15. Specify a Windows AMI ID for **sourceAmiId**, your Auto Scaling group name for **targetASG**, and a value for the **subnetId** input parameter.
16. Choose **Execute**.
17. After automation 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.
18. Choose **Auto Scaling**, and then choose **Auto Scaling Groups**. Verify that the Auto Scaling group uses the new launch configuration.
19. 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 use some of the self-service automations created by the AWS Support team. These automations help you manage your AWS resources.

### Support Automation Workflows

Support Automation Workflows (SAW) are automation runbooks written and maintained by the AWS Support team. These runbooks help you troubleshoot common issues with your AWS resources, proactively monitor and identify network issues, collect and analyze logs, and more.

SAW runbooks use the **AWSSupport** prefix. For example, [AWSSupport-ActivateWindowsWithAmazonLicense](#).

Additionally, AWS Enterprise and Business Support customers also have access to runbooks that use the **AWSPremiumSupport** prefix. For example, [AWSPremiumSupport-TroubleshootEC2DiskUsage](#).

To learn more about AWS Support, see [Getting started with AWS Support](#).

## Topics

- [Walkthrough: Run the EC2Rescue tool on unreachable instances \(p. 677\)](#)
- [Walkthrough: Reset passwords and SSH keys on EC2 instances \(p. 681\)](#)

## Walkthrough: Run the EC2Rescue tool on unreachable instances

EC2Rescue can help you diagnose and troubleshoot problems on Amazon Elastic Compute Cloud (Amazon 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** runbook. Automation is a capability of AWS Systems Manager. The **AWSSupport-ExecuteEC2Rescue** runbook 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** runbook 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 and macOS:** 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** runbook works as follows:

- You specify the ID of the unreachable instance and start the runbook.
- 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 about Amazon VPC quotas, see [VPC and Subnets](#) in the [Amazon VPC User Guide](#).
- 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 by using an IAM service role \(p. 460\)](#).

### 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. 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 \(p. 678\)](#)
- [Granting permissions by using an AWS CloudFormation template \(p. 679\)](#)

#### 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

Replace **account ID** with your own information.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "lambda:InvokeFunction",
 "lambda>DeleteFunction",
 "lambda:GetFunction"
],
 "Resource": "arn:aws:lambda:*: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",
 "iamGetInstanceProfile",
 "iam:ListRoles",
 "iam:ListInstanceProfiles",
 "iam:ListAttachedRolePolicies",
 "iam:ListPolicyAttachments",
 "iam:ListRolePolicies",
 "iam:ListRoleTags",
 "iam:ListUserPolicies",
 "iam:ListUserTags",
 "iam:PutRolePolicy",
 "iam:PutUserPolicy",
 "iam:UpdateRole",
 "iam:UpdateUser",
 "iam:UpdateRoleTags",
 "iam:UpdateUserTags",
 "iam:DeleteRole",
 "iam:DeleteUser",
 "iam:DeleteRoleTags",
 "iam:DeleteUserTags",
 "iam:TagRole",
 "iam:UntagRole",
 "iam:TagUser",
 "iam:UntagUser"
],
 "Resource": "account ID:*",
 "Effect": "Allow"
 }
]
}
```

```

 "iam:PutRolePolicy",
 "iam:DetachRolePolicy",
 "iam:AttachRolePolicy",
 "iam:PassRole",
 "iam:AddRoleToInstanceProfile",
 "iam:RemoveRoleFromInstanceProfile",
 "iam:DeleteRole",
 "iam:DeleteRolePolicy",
 "iam:DeleteInstanceProfile"
],
 "Resource": [
 "arn:aws:iam::account ID:role/AWSSupport-EC2Rescue-*",
 "arn:aws:iam::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`.

3. Sign in to the AWS Management Console and open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
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. 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**. This is the ARN of the AssumeRole. You specify this ARN when you run the Automation in the next procedure, [Running the Automation \(p. 680\)](#).

## Running the Automation

### Important

The following automation 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/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** section, choose **Owned by Amazon** from the list.
5. In the runbooks 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 runbook.

9. (Optional) In the **Tags** area, apply one or more tag key name/value pairs to help identify the automation, for example `Key=Purpose,Value=EC2Rescue`.
10. Choose **Execute**.

The runbook creates a backup AMI as part of the automation. All other resources created by the automation 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` runbook to automatically re-enable local Administrator password generation on Amazon Elastic Compute Cloud Amazon EC2 instances for Windows Server and to generate a new SSH key on EC2 instances for Linux. The `AWSSupport-ResetAccess` runbook is designed to perform a combination of AWS Systems Manager actions, AWS CloudFormation actions, and AWS Lambda functions that automate the steps normally required to reset the local administrator password.

You can use Automation, a capability of AWS Systems Manager, with the `AWSSupport-ResetAccess` runbook to solve the following problems:

### Windows

*You lost the EC2 key pair:* To resolve this problem, you can use the `AWSSupport-ResetAccess` runbook 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` runbook 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` runbook to create a new SSH key for your current instance, which enables you to connect to the instance again.

**Note**

If your EC2 instance for Windows Server is configured for Systems Manager, you can also reset your local Administrator password by using EC2Rescue and AWS Systems Manager 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` runbook works as follows:

- You specify the ID of the instance and run the runbook.
- 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 about Amazon VPC quotas, see [VPC and Subnets](#) in the *Amazon VPC User Guide*.
- 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 by using an IAM service role \(p. 460\)](#).

## 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. 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. 683)
  - Granting permissions by using an AWS CloudFormation template (p. 684)

## 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

Replace *account ID* with your own information.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "lambda:InvokeFunction",
 "lambda>DeleteFunction",
 "lambda:GetFunction"
],
 "Resource": "arn:aws:lambda::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",
 "iamGetInstanceProfile",
 "iam:PutRolePolicy",
 "iam:DetachRolePolicy",
 "iam:AttachRolePolicy",
 "iam:PassRole",
 "iam>AddRoleToInstanceProfile",
 "iam:RemoveRoleFromInstanceProfile",
 "iam:DeleteRole",
 "iam:DeleteRolePolicy",
 "iam:DeleteInstanceProfile"
],
 "Resource": [
 "arn:aws:iam::account ID:role/*",
 "arn:aws:iam::account ID:instanceprofile/*",
 "arn:aws:iam::account ID:policy/*",
 "arn:aws:iam::account ID:role/*/*",
 "arn:aws:iam::account ID:instanceprofile/*/*",
 "arn:aws:iam::account ID:policy/*/*"
]
 }
]
}
```

```
 "arn:aws:iam::account ID:role/AWSSupport-EC2Rescue-*",
 "arn:aws:iam::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`.
3. Sign in to the AWS Management Console and open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
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**. This 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** runbook by using the AWS Systems Manager console.

### Important

The following automation 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/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** section, choose **Owned by Amazon** from the list.
5. In the runbooks 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 runbook.

- 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 automation, for example `Key=Purpose,Value=ResetAccess`.
  10. Choose **Execute**.
  11. To monitor the automation progress, choose the running automation, and then choose the **Steps** tab. When the automation 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 runbook creates a backup AMI and a password-enabled AMI as part of the automation. All other resources created by the automation 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 *Instance ID*

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 AWS Systems Manager Automation walkthrough shows how to use the input transformer feature of Amazon EventBridge to extract the `instance-id` of an Amazon Elastic Compute Cloud (Amazon EC2) instance from an instance state change event. Automation is a capability of AWS Systems Manager. We use the input transformer to pass that data to the `AWS-CreateImage` runbook 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 EventBridge User Guide*.

### Before you begin

Verify that you added the required permissions and trust policy for EventBridge to your Systems Manager Automation service role. For more information, see [Overview of Managing Access Permissions to Your EventBridge Resources](#) in the *Amazon EventBridge User Guide*.

### To use input transformers with Automation

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**, and then choose **Create rule**.

-or-

If the Amazon EventBridge home page opens first, choose **Create rule**.

3. Enter a name and description for the rule.

A rule can't have the same name as another rule in the same Region and on the same event bus.

4. For **Define pattern**, do the following:

- a. Choose **Event pattern**.
  - b. For **Event matching pattern**, choose **Pre-defined pattern by service**.
  - c. For **Event matching pattern**, choose **Pre-defined pattern by service**.
  - d. For **Service provider**, choose **AWS**.
  - e. For **Service name**, choose **EC2**.
  - f. For **Event type**, choose **EC2 Instance State-change Notification**.
  - g. Choose **Specific state(s)** and **stopped** from the dropdown.
  - h. Choose **Any instance**.
5. For **Select event bus**, choose the event bus that you want to associate with this rule. If you want this rule to trigger on matching events that come from your own AWS account, select **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
  6. For **Target**, choose **Systems Manager Automation**.
  7. For **Document**, choose **AWS-CreateImage**.
  8. Expand **Configure automation parameter(s)** and choose **Input Transformer**.
  9. In the first box under **Input Transformer**, enter `{"instance": "$.detail.instance-id"}`.
  10. In the second box, enter `{"InstanceId": [<instance>]}`.
  11. Choose **Use existing role** and choose your Automation service role.
  12. (Optional) Enter one or more tags for the rule. For more information, see [Tagging Your Amazon EventBridge Resources](#) in the *Amazon EventBridge User Guide*.
  13. Choose **Create**.

## 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 Machine Images (AMIs). Automation is a capability of AWS Systems Manager. 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 Elastic Compute Cloud (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 \(p. 665\)](#) example. The following example uses the **UpdateMyLatestWindowsAmi** runbook created in that example.
- Configure IAM roles for Automation. Systems Manager requires an instance profile role and a service role ARN to process automations. For more information, see [Setting up Automation \(p. 401\)](#).
- After you configure IAM roles for Automation, use the following procedure to create an IAM user account for your Jenkins server. The automation uses the IAM user account's Access key and Secret key to authenticate the Jenkins server during the automation.

### 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 each *example resource placeholder* with your own information.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "ssm:StartAutomationExecution",
 "Resource": [
 "arn:aws:ssm:region:account ID:document/UpdateMyLatestWindowsAmi",
 "arn:aws:ssm:region:account ID:automation-definition/
 UpdateMyLatestWindowsAmi:$DEFAULT"
]
 }
]
}
```

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, enter 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. Connect to your Jenkins server on port 8080 using your preferred browser to access the management interface.
2. Enter the password found in `/var/lib/jenkins/secrets/initialAdminPassword`. To display your password, run the following command.

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

3. The Jenkins installation script directs you to the **Customize Jenkins** page. Select **Install suggested plugins**.

4. Once the installation is complete, choose **Administrator Credentials**, select **Save Credentials**, and then select **Start Using Jenkins**.
5. In the left navigation pane, choose **Manage Jenkins**, and then choose **Manage Plugins**.
6. Choose the **Available** tab, and then enter **Amazon EC2 plugin**.
7. Select the check box for **Amazon EC2 plugin**, and then select **Install without restart**.
8. When the installation completes, select **Go back to the top page**.
9. Choose **Manage Jenkins**, and then choose **Configure System**.
10. In the **Cloud** section, select **Add a new cloud**, and then choose **Amazon EC2**.
11. Enter your information in the remaining fields. You must enter your AWS credentials in the **Add Credentials** field.

Use the following procedure to configure your Jenkins project to invoke Automation.

### To configure your Jenkins server to invoke Automation

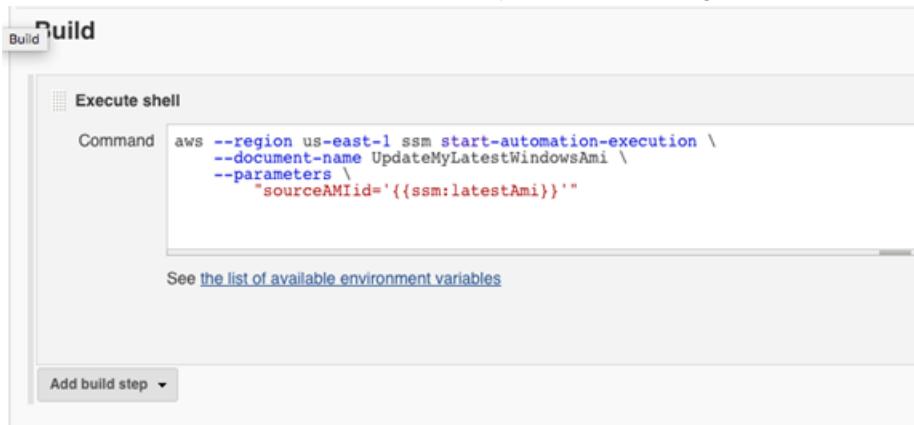
1. Open the Jenkins console in a web browser.
2. Choose the project that you want to configure with Automation, and then choose **Configure**.
3. On the **Build** tab, choose **Add Build Step**.
4. Choose **Execute shell** or **Execute Windows batch command** (depending on your operating system).
5. In the **Command** field, run an AWS CLI command like the following. Replace each *example resource placeholder* with your own information.

```
aws ssm start-automation-execution \
--document-name runbook name \
--region AWS Region of your source AMI \
--parameters runbook parameters
```

The following example command uses the **UpdateMyLatestWindowsAmi** runbook and the Systems Manager Parameter `latestAmi` created in [Walkthrough: Simplify AMI patching using Automation, AWS Lambda, and Parameter Store \(p. 665\)](#).

```
aws ssm start-automation-execution \
--document-name UpdateMyLatestWindowsAmi \
--parameters \
"sourceAMIid='{{ssm:latestAmi}}'" \
--region region
```

In Jenkins, the command looks like the example in the following screenshot.



- 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 AMI
[Build AMI] $ /bin/sh -xe /tmp/hudson3259912997441414819.sh
+ aws --region us-east-1 ssm start-automation-execution --document-name UpdateMyLatestWindowsAmi --parameters 'sourceAmiId=\'{ssm:latestAmi}\''
{
 "AutomationExecutionId": "7badf13a-ff8c-11e6-9503-9d48daa849f3"
}
Finished: SUCCESS
```

## Understanding automation statuses

AWS Systems Manager Automation reports detailed status information about the various statuses an automation action or step goes through when you run an automation and for the overall automation. Automation is a capability of AWS Systems Manager. You can monitor automation statuses using the following methods:

- Monitor the **Execution status** in the Systems Manager Automation console.
- Use your preferred command line tools. For the AWS Command Line Interface (AWS CLI), you can use [describe-automation-step-executions](#) or [get-automation-execution](#). For the AWS Tools for Windows PowerShell, you can use [Get-SMAutomationStepExecution](#) or [Get-SMAutomationExecution](#).
- Configure Amazon EventBridge to respond to action or automation status changes.

## About automation statuses

Automation reports status details for individual automation actions in addition to the overall automation.

The overall automation status can be different than the status reported by an individual action or step as noted in the following tables.

### Detailed status for actions

Status	Details
Pending	The step hasn't started running. If your automation uses conditional actions, steps remain in this state after an automation has completed if the condition wasn't met to run the step. Steps also remain in this state if the automation is canceled before the step runs.
InProgress	The step is running.
Waiting	The step is waiting for input.
Success	The step completed successfully. This is a terminal state.
TimedOut	A step or approval wasn't completed before the specified timeout period. This is a terminal state.
Cancelling	The step is in the process of stopping after being canceled by a requester.

Status	Details
Cancelled	The step was stopped by a requester before it completed. This is a terminal state.
Failed	The step didn't complete successfully. This is a terminal state.

### Detailed status for an automation

Status	Details
Pending	The automation hasn't started running.
InProgress	The automation is running.
Waiting	The automation is waiting for input.
Success	The automation completed successfully. This is a terminal state.
TimedOut	A step or approval wasn't completed before the specified timeout period. This is a terminal state.
Cancelling	The automation is in the process of stopping after being canceled by a requester.
Cancelled	The automation was stopped by a requester before it completed. This is a terminal state.
Failed	The automation didn't complete successfully. This is a terminal state.

## Troubleshooting Systems Manager Automation

Use the following information to help you troubleshoot problems with AWS Systems Manager Automation, a capability of AWS Systems Manager. This topic includes specific tasks to resolve issues based on Automation error messages.

### Topics

- [Common Automation errors \(p. 691\)](#)
- [Automation execution failed to start \(p. 692\)](#)
- [Execution started, but status is failed \(p. 693\)](#)
- [Execution started, but timed out \(p. 694\)](#)

## Common Automation errors

This section includes information about common Automation errors.

### VPC not defined 400

By default, when Automation runs either the `AWS-UpdateLinuxAmi` runbook or the `AWS-UpdateWindowsAmi` runbook, 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 can fail with an access denied error or an invalid assume role error if you haven't properly configured AWS Identity and Access Management (IAM) users, roles, and policies for Automation.

### Access denied

The following examples describe situations when an automation failed to start with an access denied error.

#### Access Denied to Systems Manager API

**Error message:** User: user arn isn't authorized to perform:  
`ssm:StartAutomationExecution` on resource: document arn (Service: AWSSimpleSystemsManagement; Status Code: 400; Error Code: AccessDeniedException; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)

- Possible cause 1: The IAM user attempting to start the automation doesn't 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 automation. For more information, see [Task 3: Configure user access to Automation \(p. 406\)](#).
- Possible cause 2: The IAM user attempting to start the automation has permission to invoke the `StartAutomationExecution` API but doesn't have permission to invoke the API by using the specific runbook. To resolve this issue, attach the required IAM policy to the user account that was used to start the automation. For more information, see [Task 3: Configure user access to Automation \(p. 406\)](#).

#### Access Denied Because of Missing PassRole Permissions

**Error message:** User: user arn isn't 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 doesn't 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. For more information, see [Task 2: Attach the iam:PassRole policy to your Automation role \(p. 406\)](#).

### Invalid assume role

When you run an Automation, an assume role is either provided in the runbook or passed as a parameter value for the runbook. Different types of errors can occur if the assume role isn't specified or configured properly.

#### Malformed Assume Role

**Error message:** The format of the supplied assume role ARN isn't valid. The assume role is improperly formatted. To resolve this issue, verify that a valid assume role is specified in your runbook or as a runtime parameter when starting 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 doesn't exist. To resolve this issue, create the role. For more information, see [the section called "Setting up Automation" \(p. 401\)](#). Specific details for creating this role are described in the following topic, [Task 1: Create a service role for Automation \(p. 403\)](#).
- Possible cause 2: The assume role doesn't 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

Runbooks 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 Amazon Elastic Compute Cloud (Amazon 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's launching the instance(s). Get Exception from RunInstances API of ec2 Service. Exception Message from RunInstances API: [The image id '[ami id]' doesn't 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 runbook or parameter values with the correct AMI ID.

#### Assume Role Policy Doesn't Have Sufficient Permissions

**Error message:** Automation Step Execution fails when it's launching the instance(s). Get Exception from RunInstances API of ec2 Service. Exception Message from RunInstances API: [You aren't authorized to perform this operation. Encoded authorization failure message: xxxxxxxx (Service: AmazonEC2; Status Code: 403; Error Code: UnauthorizedOperation; Request ID: xxxxxxxx-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 \(p. 403\)](#).

#### Unexpected State

**Error message:** Step fails when it's 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.

Action	AWS Service(s) invoked by this action	For information about this service	Troubleshooting content
<code>aws:runInstances</code>	Amazon EC2	<a href="#">Amazon EC2 User Guide for Linux Instances</a>	<a href="#">Troubleshooting EC2 Instances</a>
<code>aws:changeInstanceState</code>	Amazon EC2	<a href="#">Amazon EC2 User Guide for Linux Instances</a>	<a href="#">Troubleshooting EC2 instances</a>
<code>aws:runCommand</code>	Systems Manager	<a href="#">AWS Systems Manager Run Command (p. 991)</a>	<a href="#">Troubleshooting Systems Manager Run Command (p. 1033)</a>
<code>aws:createImage</code>	Amazon EC2	<a href="#">Amazon Machine Images</a>	
<code>aws:createStack</code>	AWS CloudFormation	<a href="#">AWS CloudFormation User Guide</a>	<a href="#">Troubleshooting AWS CloudFormation</a>
<code>aws:deleteStack</code>	AWS CloudFormation	<a href="#">AWS CloudFormation User Guide</a>	<a href="#">Troubleshooting AWS CloudFormation</a>
<code>aws:deleteImage</code>	Amazon EC2	<a href="#">Amazon Machines Images</a>	
<code>aws:copyImage</code>	Amazon EC2	<a href="#">Amazon Machine Images</a>	
<code>aws:createTag</code>	Amazon EC2, Systems Manager	<a href="#">EC2 Resource and Tags</a>	
<code>aws:invokeLambdaFunction</code>	AWS Lambda	<a href="#">AWS Lambda Developer Guide</a>	<a href="#">Troubleshooting Lambda</a>

### Automation service internal error

**Error message:** Internal Server Error. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

A problem with the Automation service is preventing the specified runbook from running correctly. To resolve this issue, contact AWS Support. Provide the execution ID and customer ID, if available.

### Execution started, but timed out

**Error message:** Step timed out while step is verifying launched instance(s) are ready to be used. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

A step in the `aws:runInstances` action timed out. This can happen if the step action takes longer to run than the value specified for `timeoutSeconds` in the step. To resolve this issue, specify a longer value for the `timeoutSeconds` parameter in the `aws:runInstances` action. If that doesn't solve the problem, investigate why the step takes longer to run than expected.

## AWS Systems Manager Change Calendar

Change Calendar, a capability of AWS Systems Manager, allows you to set up date and time ranges when actions you specify (for example, in [Systems Manager Automation \(p. 397\)](#) runbooks) might or might not be performed in your AWS account. In Change Calendar, these ranges are called *events*. When you create a Change Calendar entry, you're creating a [Systems Manager document \(p. 1287\)](#) 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. To get started with Change Calendar, open the [Systems Manager console](#). In the navigation pane, choose **Change Calendar**.

You can create a calendar and its events in the Systems Manager console. You can also import an iCalendar (.ics) file that you have exported from a supported third-party calendar provider to add its events to your calendar. Supported providers include Google Calendar, Microsoft Outlook, and iCloud Calendar.

A Change Calendar entry can be one of two types:

**DEFAULT\_OPEN**, or Open by default

All actions can run by default, except during calendar events. During events, the state of a `DEFAULT_OPEN` calendar is `CLOSED` and events are blocked from running.

**DEFAULT\_CLOSED**, or Closed by default

All actions are blocked by default, except during calendar events. During events, the state of a `DEFAULT_CLOSED` calendar is `OPEN` and actions are permitted to run.

## Who should use Change Calendar?

- Any Amazon Web Services customer who creates or runs Automation runbooks, creates change requests in Change Manager, or creates associations in State Manager. (Automation, Change Manager, and State Manager are all capabilities of AWS Systems Manager.) By integrating these capabilities with Change Calendar, you can allow or block these three action types depending on the current state of the change calendar you associate with each one.
- Administrators who are responsible for keeping the configurations of Systems Manager managed nodes consistent, stable, and functional.

## Benefits of Change Calendar

The following are some benefits of Change Calendar.

- **Review changes before they're applied**

A Change Calendar entry can help ensure that potentially destructive 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.

- **EventBridge support**

This Systems Manager capability is supported as an *event* type in Amazon EventBridge rules. For information, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#) and [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#).

#### Topics

- [Setting up Change Calendar \(p. 696\)](#)
- [Working with Change Calendar \(p. 697\)](#)
- [Adding Change Calendar dependencies to Automation runbooks \(p. 705\)](#)
- [Troubleshooting Change Calendar \(p. 705\)](#)

## Setting up Change Calendar

Complete the following before using Change Calendar, a capability of AWS Systems Manager.

### Install latest command line tools

Install the latest command line tools to get state information about calendars.

Requirement	Description
AWS CLI	(Optional) To use the AWS Command Line Interface (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 <a href="#">Installing, updating, and uninstalling the AWS CLI in the AWS Command Line Interface User Guide</a> .
AWS Tools for PowerShell	(Optional) To use the Tools for PowerShell to get state information about calendars, 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 <a href="#">Installing the AWS Tools for PowerShell in the AWS Tools for PowerShell User Guide</a> .

## 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:DeleteDocument`
- `ssm:DescribeDocument`
- `ssm:DescribeDocumentPermission`
- `ssm:GetCalendar`
- `ssm>ListDocuments`
- `ssm:ModifyDocumentPermission`
- `ssm:PutCalendar`
- `ssm:UpdateDocument`
- `ssm:UpdateDocumentDefaultVersion`

To get information about the current or upcoming state of the calendar, a policy attached to your 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're shared with your account.

## Working with Change Calendar

You can use the AWS Systems Manager console to add, manage, or delete entries in Change Calendar, a capability of AWS Systems Manager. You can also import events from supported third-party calendar providers by importing an iCalendar (.ics) file that you exported from the source calendar. And, you can use the `GetCalendarState` API operation or the `get-calendar-state` AWS Command Line Interface (AWS CLI) command to get information about the state of Change Calendar at a specific time.

### Topics

- [Creating a change calendar \(p. 697\)](#)
- [Creating and managing events in Change Calendar \(p. 698\)](#)
- [Importing and managing events from third-party calendars \(p. 700\)](#)
- [Updating a change calendar \(p. 702\)](#)
- [Sharing a change calendar \(p. 703\)](#)
- [Deleting a change calendar \(p. 703\)](#)
- [Getting the state of a change calendar \(p. 703\)](#)

## Creating a change calendar

When you create an entry in Change Calendar, a capability of AWS Systems Manager, you're creating a Systems Manager document (SSM document) that uses the `text` format.

## To create a change calendar

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Calendar**.
3. Choose **Create calendar**.  
-or-  
If the **Change Calendar** home page opens first, choose **Create change 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 can't update this name after you create the calendar entry.
5. (Optional) For **Description**, enter a description for your calendar entry.
6. (Optional) In the **Import calendar** area, choose **Choose file** to select an iCalendar (.ics) file that you have exported from a third-party calendar provider. Importing the file will add its events to your calendar.  
Supported providers include Google Calendar, Microsoft Outlook, and iCloud Calendar.  
For more information, see [Importing events from third-party calendar providers \(p. 701\)](#).
7. 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 can't run until an event starts) but open for the duration of an associated event.
8. 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 can't prevent or allow any actions until you have created or imported at least one event. For information about creating an event, see [Creating a Change Calendar event \(p. 699\)](#). For information about importing events, see [Importing events from third-party calendar providers \(p. 701\)](#).

## Creating and managing events in Change Calendar

After you create a calendar in AWS Systems Manager Change Calendar, you can create, update, and delete events that are included in your open or closed calendar. Change Calendar is a capability of AWS Systems Manager.

### Tip

As an alternative to creating events directly in the Systems Manager console, you can import an iCalendar (.ics) file from a supported third-party calendar application. For information, see [Importing and managing events from third-party calendars \(p. 700\)](#).

### Topics

- [Creating a Change Calendar event \(p. 699\)](#)
- [Updating a Change Calendar event \(p. 699\)](#)
- [Deleting a Change Calendar event \(p. 700\)](#)

## Creating a Change Calendar event

When you add an event to an entry in Change Calendar, a capability in AWS Systems Manager, you're 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. (Alternatively, you can create an advisory event, which serves an informational role on the calendar only.)

Currently, you can only create a Change Calendar event by using the console. Events are added to the Change Calendar document that you create when you create a Change Calendar entry.

### To create a Change Calendar event

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Calendar**.
3. In the list of calendars, 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**.
6. For **Description**, enter a description for your event. For example, **The support team isn't available during these hours**.
7. (Optional) If you want this event to serve as a visual notification or reminder only, select the **Advisory** check box. Advisory events play no functional role on your calendar. They serve informational purposes only for those who view your calendar.
8. For **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.
9. For **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.
10. For **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 Coordinated Universal Time (UTC).
11. (Optional) To create an event that recurs daily, weekly, or monthly, turn on **Recurrence**, and then specify the frequency and optional end date for the recurrence.
12. 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.

## Updating a Change Calendar event

Use the following procedure to update a Change Calendar event in the AWS Systems Manager console. Change Calendar is a capability of AWS Systems Manager.

### To update a Change Calendar event

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Calendar**.
3. In the list of calendars, 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.

#### Tip

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 list on the upper right.

6. In **Event details**, choose **Edit**.

- To change the event name and description, add to or replace the current text values.
7. To change the **Event start date** value, choose the current start date, and then choose a new date from the calendar. To change the start time, choose the current start time, and then choose a new time from the list.
  8. To change the **Event end date** value, choose the current date, and then choose a new end date from the calendar. To change the end time, choose the current end time, and then choose a new time from the list.
  9. To change the **Schedule time zone** value, choose a time zone to apply 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 Coordinated Universal Time (UTC).
  10. (Optional) If you want this event to serve as a visual notification or reminder only, select the **Advisory** check box. Advisory events play no functional role on your calendar. They serve informational purposes only for those who view your calendar.
  11. Choose **Save**. 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.

## Deleting a Change Calendar event

You can delete one event at a time in Change Calendar, a capability in AWS Systems Manager, by using the AWS Management Console.

### To delete a Change Calendar event

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Calendar**.
3. In the list of calendars, 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.

#### Tip

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 list on the upper right.

6. On the **Event details** page, choose **Delete**. When you're prompted to confirm that you want to delete the event, choose **Confirm**.

## Importing and managing events from third-party calendars

As an alternative to creating events directly in the AWS Systems Manager console, you can import an iCalendar (.ics) file from a supported third-party calendar application. Your calendar can include both imported events and events that you create in Change Calendar, which is a capability of AWS Systems Manager.

### Before you begin

Before you attempt to import a calendar file, review the following requirements and constraints:

#### Calendar file format

Only valid iCalendar files (.ics) are supported.

#### Supported calendar providers

Only .ics files exported from the following third-party calendar providers are supported:

- Google Calendar ([Export instructions](#))
- Microsoft Outlook ([Export instructions](#))
- iCloud Calendar ([Export instructions](#))

#### File size

You can import any number of valid .ics files. However, the total size of all imported files for each calendar can't exceed 64KB.

#### Tip

To minimize the size of the .ics file, ensure that you are exporting only basic details about your calendar entries. If necessary, reduce the length of the time period that you are exporting.

#### Time zone

In addition to a calendar name, a calendar provider, and at least one event, your exported .ics file should also indicate the time zone for the calendar. If it does not, or there is a problem identifying the time zone, you will be prompted to specify one after you import the file.

#### Recurring event limitation

Your exported .ics file can include recurring events. However, if one or more occurrences of a recurring event had been deleted in the source calendar, the import fails.

#### Topics

- [Importing events from third-party calendar providers \(p. 701\)](#)
- [Updating all events from a third-party calendar provider \(p. 702\)](#)
- [Deleting all events imported from a third-party calendar \(p. 702\)](#)

## Importing events from third-party calendar providers

Use the following procedure to import an iCalendar (.ics) file from a supported third-party calendar application. The events contained in the file are incorporated into the rules for your open or closed calendar. You can import a file into a new calendar you are creating with Change Calendar (a capability of AWS Systems Manager) or into an existing calendar.

After you import the .ics file, you can remove individual events from it using the Change Calendar interface. For information, see [Deleting a Change Calendar event \(p. 700\)](#). You can also delete all events from the source calendar by deleting the .ics file. For information, see [Deleting all events imported from a third-party calendar \(p. 702\)](#).

### To import events from third-party calendar providers

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Calendar**.
3. To start with a new calendar, choose **Create calendar**. In the **Import calendar** area, choose **Choose file**. For information about other steps to create a new calendar, see [Creating a change calendar \(p. 697\)](#).

-or-

To import third-party events into an existing calendar, choose the name of an existing calendar to open it.

4. Choose **Actions, Edit**, and then in the **Import calendar** area, choose **Choose file**.
5. Browse to and select the exported .ics file on your local computer.
6. If prompted, for **Select a time zone**, select which time zone applies to the calendar.
7. Choose **Save**.

## Updating all events from a third-party calendar provider

If several events are added to or removed from your source calendar after you have imported its iCalendar .ics file, you can reflect those changes in Change Calendar. First, re-export the source calendar, and then import the new file into Change Calendar, which is a capability of AWS Systems Manager. Events in your change calendar will be updated to reflect the contents of the newer file.

### To update all events from a third-party calendar provider

1. In your third-party calendar, add or remove events as you want them to be reflected in Change Calendar, and then re-export the calendar to a new .ics file.
2. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
3. In the navigation pane, choose **Change Calendar**.
4. From the list of calendars, choose the calendar name from the list.
5. Choose **Choose file**, and then browse to and select the replacement .ics file.
6. In response to the notification about overwriting the existing file, choose **Confirm**.

## Deleting all events imported from a third-party calendar

If you no longer want any of the events that you imported from a third-party provider included in your calendar, you can delete the imported iCalendar .ics file.

### To delete all events imported from a third-party calendar

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Calendar**.
3. From the list of calendars, choose the calendar name from the list.
4. In the **Import calendar** area, under **My imported calendars**, locate the name of the imported calendar, and then choose the X in its card.
5. Choose **Save**.

## Updating a change calendar

You can update the description of a change calendar, but not its name. Although you can change the default state of a calendar, be aware that this reverses the behavior of change actions during events that are associated with the calendar. 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 aren't expecting changes.

When you update a change calendar, you're editing the Change Calendar document that you created when you created the entry. Change Calendar is a capability of AWS Systems Manager.

### To update a change calendar

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Calendar**.
3. In the list of calendars, choose the name of the calendar that you want to update.
4. On the calendar's details page, choose **Actions, Edit**.
5. In **Description**, you can change the description text. You can't edit the name of a change calendar.
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. Before you

change the calendar type, you should verify with other Change Calendar users that changing the calendar type doesn't 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 can't run until an event starts) but open for the duration of an associated event.
7. Choose **Save**.

Your calendar can't prevent or allow any actions until you add at least one event. For information about how to add an event, see [Creating a Change Calendar event \(p. 699\)](#).

## Sharing a change calendar

You can share a calendar in Change Calendar, a capability of AWS Systems Manager, with other AWS accounts by using the AWS Systems Manager console. When you share a calendar, the calendar is readable only to users in the shared account.

### To share a change calendar

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Calendar**.
3. In the list of calendars, choose the name of the calendar that you want to share.
4. On the calendar's details page, choose the **Sharing** tab.
5. Choose **Actions, Share**.
6. 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 they can't make changes.

## Deleting a change calendar

You can delete a calendar in Change Calendar, a capability of AWS Systems Manager, by using either the Systems Manager console or the AWS Command Line Interface (AWS CLI). Deleting a change calendar deletes all associated events.

### To delete a change calendar

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Change Calendar**.
3. In the list of calendars, choose the name of the calendar that you want to delete.
4. On the calendar's details page, choose **Actions, Delete**. When you're prompted to confirm that you want to delete the calendar, choose **Delete**.

## Getting the state of a change calendar

You can get the overall state of a calendar or the state of a calendar at a specific time in Change Calendar, a capability in AWS Systems Manager. You can also show the next time that the calendar state changes from OPEN to CLOSED, or the reverse.

You can do this task only by using the `GetCalendarState` API operation. The procedure in this section uses the AWS Command Line Interface (AWS CLI).

## To get the state of a change calendar

- Run the following command to show the state of one or more calendars at a specific time. The --calendar-names parameter is required, but --at-time is optional.

Linux & macOS

```
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.

```
aws ssm get-calendar-state \
--calendar-names "arn:aws:ssm:us-east-2:123456789012:document/
MyChangeCalendarDocument" "arn:aws:ssm:us-east-2:123456789012:document/
SupportOffHours" \
--at-time "2020-07-30T11:05:14-0700"
```

Windows

```
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.

```
aws ssm get-calendar-state ^
--calendar-names "arn:aws:ssm:us-east-2:123456789012:document/
MyChangeCalendarDocument" "arn:aws:ssm:us-east-2:123456789012:document/
SupportOffHours" ^
--at-time "2020-07-30T11:05:14-0700"
```

The command returns information like the following.

```
{
 "State": "OPEN",
 "AtTime": "2020-07-30T16:18:18Z",
 "NextTransitionTime": "2020-07-31T00:00:00Z"
}
```

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 don't add the `--at-time` parameter, the current time is used.

### Note

If you specify more than one calendar in a request, the command returns the status of `OPEN` only if all calendars in the request are open. If one or more calendars in the request are closed, the status returned is `CLOSED`.

## Adding Change Calendar dependencies to Automation runbooks

To make Automation actions adhere to Change Calendar, a capability of AWS Systems Manager, add a step in an Automation runbook that uses the [aws:assertAwsResourceProperty \(p. 486\)](#) 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 runbook 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 runbook that can't advance to the next step, LaunchInstance, unless the calendar state matches OPEN, the state specified in DesiredValues.

The following is an example.

```
mainSteps:
 - name: MyCheckCalendarStateStep
 action: 'aws:assertAwsResourceProperty'
 inputs:
 Service: ssm
 Api: GetCalendarState
 CalendarNames: ["arn:aws:ssm:us-east-2:123456789012:document/SaleDays"]
 PropertySelector: '$.State'
 DesiredValues:
 - OPEN
 description: "Use GetCalendarState to determine whether a calendar is open or closed."
 nextStep: LaunchInstance
 - name: LaunchInstance
 action: 'aws:executeScript'
 inputs:
 Runtime: python3.6
...
...
```

## Troubleshooting Change Calendar

Use the following information to help you troubleshoot problems with Change Calendar, a capability of AWS Systems Manager.

### Topics

- ['Calendar import failed' error \(p. 705\)](#)

## 'Calendar import failed' error

**Problem:** When importing an iCalendar (.ics) file, the system reports that the calendar import failed.

- **Solution 1** – Ensure that you are importing a file that was exported from a supported third-party calendar provider, which include the following:
  - Google Calendar ([Export instructions](#))
  - Microsoft Outlook ([Export instructions](#))
  - iCloud Calendar ([Export instructions](#))
- **Solution 2** – If your source calendar contains any recurring events, ensure that no individual occurrences of the event have been canceled or deleted. Currently, Change Calendar doesn't support importing recurring events with individual cancellations. To resolve the issue, remove the recurring event from the source calendar, re-export the calendar and re-import it into Change Calendar, and then add the recurring event using the Change Calendar interface. For information, see [Creating a Change Calendar event \(p. 699\)](#).

- **Solution 3** – Ensure that your source calendar contains at least one event. Uploads of .ics files that don't contain events don't succeed.
- **Solution 4** – If the system reports that the import failed because the .ics is too large, ensure that you are exporting only basic details about your calendar entries. If necessary, reduce the length of the time period that you export.
- **Solution 5** – If Change Calendar is unable to determine the time zone of your exported calendar when you attempt to import it from the **Events** tab, you might receive this message: "Calendar import failed. Change Calendar couldn't locate a valid time zone. You can import the calendar from the Edit menu." In this case, choose **Actions**, **Edit**, and then try importing the file from the **Edit calendar** page.
- **Solution 6** – Don't edit the .ics file before import. Attempting to modify the contents of the file can corrupt the calendar data. If you have modified the file before attempting the import, export the calendar from the source calendar again, and then re-attempt the upload.

## AWS Systems Manager Maintenance Windows

Maintenance Windows, a capability of AWS Systems Manager, helps you define a schedule for when to perform potentially disruptive actions on your nodes such as patching an operating system, updating drivers, or installing software or patches. With Maintenance Windows, you can 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 [Resources you can use with AWS Resource Groups and Tag Editor](#) in the *AWS Resource Groups User Guide*. To get started with Maintenance Windows, open the [Systems Manager console](#). In the navigation pane, choose **Maintenance Windows**.

### Note

State Manager and Maintenance Windows can perform some similar types of updates on your managed nodes. Which one you choose depends on whether you need to automate system compliance or perform high-priority, time-sensitive tasks during periods you specify.

For more information, see [Choosing between State Manager and Maintenance Windows \(p. 154\)](#).

Each maintenance window has a schedule, a maximum duration, a set of registered targets (the nodes 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 shouldn't 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 [Maintenance window scheduling and active period options \(p. 790\)](#).

For more information about working with the --schedule option, see [Reference: Cron and rate expressions for Systems Manager \(p. 1540\)](#).

### Supported task types

Maintenance windows support running four types of tasks:

- Commands in Run Command, a capability of Systems Manager

For more information about Run Command, see [AWS Systems Manager Run Command \(p. 991\)](#).

- Workflows in Automation, a capability of Systems Manager

For more information about Automation workflows, see [AWS Systems Manager Automation \(p. 397\)](#).

- Functions in AWS Lambda

For more information about Lambda functions, see [Getting started with Lambda in the AWS Lambda Developer Guide](#).

- Tasks in AWS Step Functions

For more information about Step Functions, see the [AWS Step Functions Developer Guide](#).

**Note**

One or more targets must be specified for maintenance window Run Command-type tasks.

Depending on the task, targets are optional for other maintenance window task types (Automation, AWS Lambda, and AWS Step Functions). For more information about running tasks that don't specify targets, see [Registering maintenance window tasks without targets \(p. 794\)](#).

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 nodes by using a Systems Manager Automation task.
- Run AWS Lambda functions that invokes additional actions, such as scanning your nodes for patch updates.
- Run AWS Step Functions state machines to perform tasks such as removing a node from an Elastic Load Balancing environment, patching the node, and then adding the node back to the Elastic Load Balancing environment.
- Target nodes that are offline by specifying an AWS resource group as the target.

**EventBridge support**

This Systems Manager capability is supported as an *event* type in Amazon EventBridge rules. For information, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#) and [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#).

**Contents**

- [Setting up Maintenance Windows \(p. 707\)](#)
- [Working with maintenance windows \(console\) \(p. 724\)](#)
- [Systems Manager Maintenance Windows tutorials \(AWS CLI\) \(p. 732\)](#)
- [Maintenance window walkthroughs \(p. 777\)](#)
- [Maintenance window scheduling and active period options \(p. 790\)](#)
- [Registering maintenance window tasks without targets \(p. 794\)](#)
- [Troubleshooting maintenance windows \(p. 795\)](#)

## Setting up Maintenance Windows

Before users in your AWS account can create and schedule maintenance window tasks using Maintenance Windows, a capability of AWS Systems Manager, they must be granted the necessary permissions. To grant these permissions to users, an administrator must perform these two tasks:

**Task 1: Configure node permissions**

Provide the Maintenance Windows service with the AWS Identity and Access Management (IAM) permissions needed to run maintenance window tasks on your nodes 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 Systems Manager Inventory and Systems Manager Explorer. For more information, see [Using roles to collect inventory, run maintenance window tasks, and view OpsData: AWSServiceRoleForAmazonSSM \(p. 1410\)](#).

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? \(p. 708\)](#).

**Task 2: Configure permissions for users who are allowed to register maintenance window tasks**

Allow `iam:PassRole` permissions for the users in your AWS account who assign tasks to maintenance windows. This allows them to pass the role to the maintenance window service. Without this explicit IAM permission, a user can't assign tasks to a maintenance window when using a custom service role to run maintenance window tasks.

**Task 3: Configure permissions for users who aren't allowed to register maintenance window tasks**

Deny `ssm:RegisterTaskWithMaintenanceWindow` permissions for the users in your AWS account who you don't want to register tasks with maintenance windows. This prevents users from registering a maintenance window task by using the service-linked role in a maintenance window task registration request.

**Before you begin**

To complete the tasks in the section, you need one or both of the following resources.

- You're 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 \(p. 19\)](#) and [Create users and assign permissions \(p. 20\)](#).
- (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 \(p. 1462\)](#).

## Should I use a service-linked role or a custom service role to run maintenance window tasks?

To run maintenance tasks on your target nodes, the Maintenance Windows service must have permission to access and run tasks on your nodes. 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 Amazon SNS to send notifications related to status changes for Run Command tasks registered with your maintenance windows. For information, see the following topics:
  - [Monitoring Systems Manager status changes using Amazon SNS notifications \(p. 1462\)](#)
  - [Use a maintenance window to send a command that returns status notifications \(p. 1471\)](#)
- 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 nodes, or you want to allow only certain Systems Manager documents (SSM documents) run on your target nodes. 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 runbooks 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 runbook `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 runbooks, see the document descriptions in [Systems Manager Automation runbook reference \(p. 604\)](#).

**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 \(p. 1410\)](#).

#### Topics

- [Control access to maintenance windows \(console\) \(p. 709\)](#)
- [Control access to maintenance windows \(AWS CLI\) \(p. 714\)](#)
- [Control access to maintenance windows \(Tools for Windows PowerShell\) \(p. 720\)](#)

## 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.

#### Topics

- [Task 1: \(Optional\) Create a custom service role for maintenance windows \(console\) \(p. 709\)](#)
- [Task 2: Configure permissions for users who are allowed to register maintenance window tasks \(console\) \(p. 711\)](#)
- [Task 3: Configure permissions for users who aren't allowed to register maintenance window tasks \(console\) \(p. 713\)](#)

### Task 1: (Optional) Create a custom service role for maintenance windows (console)

Use the following procedure to create a custom service role for Maintenance Windows, a capability of Systems Manager, so that Systems Manager can run tasks on your behalf.

### Important

A custom service role isn't required if you choose to use a Systems Manager service-linked role to let maintenance windows run tasks on your behalf instead. If you don't 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? \(p. 708\)](#)
- [Using service-linked roles for Systems Manager \(p. 1410\)](#)
- [Assign tasks to a maintenance window \(console\) \(p. 727\)](#)

### 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 a use case area: Systems Manager**
  3. **Select your use case area: Systems Manager**
4. Choose **Next: Permissions**.
5. In the search box, enter **AmazonSSMMaintenanceWindowRole**, 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. For **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 is displayed in the **Policy Document** field.

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

}

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 Simple Notification Service (Amazon SNS), when run through a Systems Manager 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**, replace the default contents with the following.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": "arn:aws::iam::account-id:role/my-sns-access-role"
 }
]
}
```

**account-id** represents the the 12-digit identifier for your AWS account, in the format 123456789012.

**my-sns-access-role** represents the name of the existing AWS Identity and Access Management (IAM) role to use to send Amazon SNS notifications related to the maintenance window.

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 \(p. 1462\)](#).

**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 \(console\) \(p. 727\)](#). 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: Configure permissions for users who are allowed to register maintenance window tasks (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, 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're 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 configure permissions for users who are allowed to register maintenance window tasks (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** tab, 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 isn't included already. For information, see [Adding and removing IAM identity permissions](#) in the *IAM User Guide*.
4. Choose **Add inline policy**, and then choose the **JSON** tab.
5. Replace the default contents of the box with the following.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": "arn:aws:iam::account-id:role/my-maintenance-window-role"
 },
 {
 "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/"
 }
]
}
```

`my-maintenance-window-role` represents the name of the custom maintenance window role you created earlier.

`account-id` represents 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.

6. Choose **Review policy**.
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 configure permissions for groups that are allowed to register maintenance window tasks (console)

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **User 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, choose **Add permissions**, **Create inline policy**, and then choose the **JSON** tab.
5. Replace the default contents of the box with the following.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": "arn:aws:iam::account-id:role/my-maintenance-window-role"
 },
 {
 "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/"
 }
]
}
```

`my-maintenance-window-role` represents the name of the custom maintenance window role you created earlier.

`account-id` represents 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.

6. Choose **Review policy**.
7. On the **Review policy** page, enter a name in the **Name** box to identify this `PassRole` policy, such as `my-group-iam-passrole-policy`, and then choose **Create policy**.

### Task 3: Configure permissions for users who aren't allowed to register maintenance window tasks (console)

Depending on whether you're denying the `ssm:RegisterTaskWithMaintenanceWindow` permission for an individual user or a group, use one of the following procedures to prevent users from registering tasks with a maintenance window.

#### To configure permissions for users who aren't allowed to register maintenance window tasks (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. Choose **Add inline policy**, and then choose the **JSON** tab.
4. Replace the default contents of the box with the following.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Deny",
 "Action": "ssm:RegisterTaskWithMaintenanceWindow",
 "Resource": "*"
 }
]
}
```

5. Choose **Review policy**.
6. On the **Review policy** page, enter a name in the **Name** box to identify this policy, such as **my-deny-mw-tasks-policy**, and then choose **Create policy**.

#### To configure permissions for groups that aren't allowed to register maintenance window tasks (console)

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **User groups**.
3. In the list of groups, select the name of the group you want to deny the `ssm:RegisterTaskWithMaintenanceWindow` permission from.
4. On the **Permissions** tab, choose **Add permissions, Create inline policy**.
5. Choose the **JSON** tab, and then replace the default contents of the box with the following.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Deny",
 "Action": "ssm:RegisterTaskWithMaintenanceWindow",
 "Resource": "*"
 }
]
}
```

6. Choose **Review policy**.
7. On the **Review policy** page, for **Name**, enter a name to identify this policy, such as **my-groups-deny-mw-tasks-policy**, and then choose **Create policy**.

## Control access to maintenance windows (AWS CLI)

The following procedures describe how to use the AWS Command Line Interface (AWS CLI) to create the required roles and permissions for Maintenance Windows, a capability of AWS Systems Manager.

### Topics

- [Task 1: \(Optional\) Create a custom service role for maintenance windows \(AWS CLI\) \(p. 715\)](#)
- [Task 2: Configure permissions for users who are allowed to register maintenance window tasks \(AWS CLI\) \(p. 716\)](#)
- [Task 3: Configure permissions for users who aren't allowed to register maintenance window tasks \(AWS CLI\) \(p. 719\)](#)

## Task 1: (Optional) Create a custom service role for maintenance windows (AWS CLI)

### Important

A custom service role isn't required if you choose to use a Systems Manager service-linked role to let maintenance windows run tasks on your behalf instead. If you don't 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? \(p. 708\)](#)
- [Using service-linked roles for Systems Manager \(p. 1410\)](#)
- [Assign tasks to a maintenance window \(console\) \(p. 727\)](#)

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`.

```
{
 "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 & macOS

```
aws iam create-role \
 --role-name "my-maintenance-window-role" \
 --assume-role-policy-document file://mw-role-trust-policy.json
```

Windows

```
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.

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

```

 "Principal": {
 "Service": "ssm.amazonaws.com"
 }
 }
],
"RoleId": "AROAIIZKPBK52LEXAMPLE",
"CreateDate": "2017-04-04T03:40:17.373Z",
"RoleName": "my-maintenance-window-role",
"Path": "/",
"Arn": "arn:aws:iam::123456789012:role/my-maintenance-window-role"
}
}

```

### 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.

- Run the following command to attach the `AmazonSSMMaintenanceWindowRole` managed policy to the role you created in step 2.

Linux & macOS

```
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: Configure permissions for users who are allowed to register maintenance window tasks (AWS CLI)

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, 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 configure permissions for users who are allowed to register maintenance window tasks (AWS CLI)

- Copy and paste the following AWS Identity and Access Management (IAM) policy into a text editor and save it with the following name and file extension: `mw-passrole-policy.json`.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": "arn:aws:iam::account-id:role/my-maintenance-window-role"
 }
]
}
```

```

 },
 {
 "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 *my-maintenance-window-role* with the name of the custom maintenance window role you created earlier.

Replace *account-id* 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're assigning the permission to an IAM user or group, run one of the following commands.

- **For an IAM user:**

Linux & macOS

```
aws iam put-user-policy \
--user-name "user-name" \
--policy-name "policy-name" \
--policy-document file:///path-to-document
```

Windows

```
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 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 \(p. 19\)](#) and [Create users and assign permissions \(p. 20\)](#). Run the following command to assign the **AmazonSSMFullAccess** policy to your account.

Linux & macOS

```
aws iam attach-user-policy \
```

```
--policy-arn "arn:aws:iam::aws:policy/AmazonSSMFullAccess" \
--user-name "user-name"
```

### Windows

```
aws iam attach-user-policy ^
--policy-arn "arn:aws:iam::aws:policy/AmazonSSMFullAccess" ^
--user-name "user-name"
```

- **For an IAM group:**

### Linux & macOS

```
aws iam put-group-policy \
--group-name "group-name" \
--policy-name "policy-name" \
--policy-document file://path-to-document
```

### Windows

```
aws iam put-group-policy ^
--group-name "group-name" ^
--policy-name "policy-name" ^
--policy-document file://path-to-document
```

For *group-name*, specify the IAM group whose members assign 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 members of a group to register tasks for maintenance windows using the Systems Manager console, you must also assign the **AmazonSSMFullAccess** policy to your group. Run the following command to assign this policy to your group.

### Linux & macOS

```
aws iam attach-group-policy \
--policy-arn "arn:aws:iam::aws:policy/AmazonSSMFullAccess" \
--group-name "group-name"
```

### Windows

```
aws iam attach-group-policy ^
--policy-arn "arn:aws:iam::aws:policy/AmazonSSMFullAccess" ^
--group-name "group-name"
```

4. Run the following command to verify that the policy has been assigned to the group.

### Linux & macOS

```
aws iam list-group-policies \
--group-name "group-name"
```

### Windows

```
aws iam list-group-policies ^
--group-name "group-name"
```

## Task 3: Configure permissions for users who aren't allowed to register maintenance window tasks (AWS CLI)

1. Copy and paste the following IAM policy into a text editor and save it with the following name and file extension: deny-mw-tasks-policy.json.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Deny",
 "Action": "ssm:RegisterTaskWithMaintenanceWindow",
 "Resource": "*"
 }
]
}
```

2. Open the AWS CLI.
3. Depending on whether you're assigning the permission to an IAM user or group, run one of the following commands.

- For an IAM user:

Linux & macOS

```
aws iam put-user-policy \
 --user-name "user-name" \
 --policy-name "policy-name" \
 --policy-document file:///path-to-document
```

Windows

```
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 to prevent from assigning tasks to maintenance windows.

For *policy-name*, specify the name you want to use to identify the policy, such as **my-deny-mw-tasks-policy**. For *path-to-document*, specify the path to the file you saved in step 1. For example: file://C:\Temp\deny-mw-tasks-policy.json

- For an IAM group:

Linux & macOS

```
aws iam put-group-policy \
 --group-name "group-name" \
 --policy-name "policy-name" \
 --policy-document file:///path-to-document
```

Windows

```
aws iam put-group-policy ^
 --group-name "group-name" ^
 --policy-name "policy-name" ^
 --policy-document file:///path-to-document
```

For `group-name`, specify the IAM group whose to prevent from assigning tasks to maintenance windows. For `policy-name`, specify the name you want to use to identify the policy, such as `my-deny-mw-tasks-policy`. For `path-to-document`, specify the path to the file you saved in step 1. For example: `file://C:\Temp\deny-mw-tasks-policy.json`

- Run the following command to verify that the policy has been assigned to the group.

Linux & macOS

```
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 AWS Tools for Windows PowerShell to create the required roles and permissions for Maintenance Windows, a capability of AWS Systems Manager.

### Topics

- [Task 1: \(Optional\) Create a custom service role for maintenance windows \(Tools for Windows PowerShell\) \(p. 720\)](#)
- [Task 2: Configure permissions for users who are allowed to register maintenance window tasks \(PowerShell\) \(p. 721\)](#)
- [Task 3: Configure permissions for users who aren't allowed to register maintenance window tasks \(PowerShell\) \(p. 723\)](#)

### Task 1: (Optional) Create a custom service role for maintenance windows (Tools for Windows PowerShell)

#### Important

A custom service role isn't required if you choose to use a Systems Manager service-linked role to let maintenance windows run tasks on your behalf instead. If you don't 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? \(p. 708\)](#)
- [Using service-linked roles for Systems Manager \(p. 1410\)](#)
- [Assign tasks to a maintenance window \(console\) \(p. 727\)](#)

- 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`.

```
{
 "Version": "2012-10-17",
 "Statement": {
 "Effect": "Allow",
 "Principal": "*",
 "Action": "sts:AssumeRole",
 "Condition": {}
 }
}
```

```

 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "sts:AssumeRole"
 }
]
}

```

- 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.

```

New-IAMRole `
 -RoleName "my-maintenance-window-role" `
 -AssumeRolePolicyDocument (Get-Content -raw .\mw-role-trust-policy.json)

```

The system returns information like the following.

```

Arn : arn:aws:iam::123456789012:role/mw-task-role
AssumeRolePolicyDocument : ExampleDoc12345678
CreateDate : 4/4/2017 11:24:43
Path : /
RoleId : AROAIIZKPBKS2LEXAMPLE
RoleName : my-maintenance-window-role

```

- Run the following command to attach the `AmazonSSMMaintenanceWindowRole` managed policy to the role you created in the previous step.

```

Register-IAMRolePolicy `
 -RoleName "my-maintenance-window-role" `
 -PolicyArn "arn:aws:iam::aws:policy/service-role/AmazonSSMMaintenanceWindowRole"

```

## Task 2: Configure permissions for users who are allowed to register maintenance window tasks (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, 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*.

- Copy and paste the following AWS Identity and Access Management (IAM) policy into a text editor and save it with the following name and file extension: `mw-passrole-policy.json`.

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": "arn:aws:iam::account-id:role/my-maintenance-window-role"
 }
]
}

```

```

 "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 `my-maintenance-window-role` with the name of the custom maintenance window role you created earlier.

Replace `account-id` 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 Tools for Windows PowerShell.
3. Depending on whether you're assigning the permission to an IAM user or group, run one of the following commands.

- **For an IAM user:**

```

Write-IAMUserPolicy `
 -UserName "user-name" `
 -PolicyDocument (Get-Content -raw path-to-document) `
 -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.

```

Register-IAMUserPolicy `
 -UserName "user-name" `
 -PolicyArn "arn:aws:iam::aws:policy/AmazonSSMFullAccess"

```

- **For an IAM group:**

```

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.

```
Register-IAMGroupPolicy `
-GroupName "group-name" `
-PolicyArn "arn:aws:iam::aws:policy/AmazonSSMFullAccess"
```

- Run the following command to verify that the policy has been assigned to the group.

```
Get-IAMGroupPolicies `
-GroupName "group-name"
```

### Task 3: Configure permissions for users who aren't allowed to register maintenance window tasks (PowerShell)

- Copy and paste the following IAM policy into a text editor and save it with the following name and file extension: `deny-mw-tasks-policy.json`.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Deny",
 "Action": "ssm:RegisterTaskWithMaintenanceWindow",
 "Resource": "*"
 }
]
}
```

- Open Tools for Windows PowerShell.
- Depending on whether you're assigning the permission to an IAM user or group, run one of the following commands.

- For an IAM user:

```
Write-IAMUserPolicy `
-UserName "user-name" `
-PolicyDocument (Get-Content -raw path-to-document) `
-PolicyName "policy-name"
```

For `user-name`, specify the IAM user to prevent from assigning tasks to maintenance windows. For `policy-name`, specify the name you want to use to identify the policy, such as `my-deny-mw-tasks-policy`. For `path-to-document`, specify the path to the file you saved in step 1. For example: `C:\temp\deny-mw-tasks-policy.json`

- For an IAM group:

```
Write-IAMGroupPolicy `
-GroupName "group-name" `
-PolicyDocument (Get-Content -raw path-to-document) `
-PolicyName "policy-name"
```

For *group-name*, specify the IAM group to prevent from assigning tasks to maintenance windows. For *policy-name*, specify the name you want to use to identify the policy, such as **my-deny-mw-tasks-policy**. For *path-to-document*, specify the path to the file you saved in step 1. For example: C:\temp\deny-mw-tasks-policy.json

4. Run the following command to verify that the policy has been assigned to the group.

```
Get-IAMGroupPolicies `
-GroupName "group-name"
```

## Working with maintenance windows (console)

This section describes how to create, configure, update, and delete maintenance windows using the AWS 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, a capability of AWS Systems Manager. For more information, see [Setting up Maintenance Windows \(p. 707\)](#).

### Topics

- [Create a maintenance window \(console\) \(p. 724\)](#)
- [Assign targets to a maintenance window \(console\) \(p. 726\)](#)
- [Assign tasks to a maintenance window \(console\) \(p. 727\)](#)
- [Updating or deleting maintenance window resources \(console\) \(p. 730\)](#)

## Create a maintenance window (console)

In this procedure, you create a maintenance window in Maintenance Windows, a capability of AWS Systems Manager. You can 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 when the maintenance window runs.

**Note**

For an explanation of how the various schedule-related options for maintenance windows relate to one another, see [Maintenance window scheduling and active period options \(p. 790\)](#).

For more information about working with the --schedule option, see [Reference: Cron and rate expressions for Systems Manager \(p. 1540\)](#).

### To create a maintenance window (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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.
5. (Optional) For **Description**, enter a description to identify how this maintenance window will be used.

6. (Optional) If you want to allow a maintenance window task to run on managed nodes, even if you haven't registered those nodes as targets, choose **Allow unregistered targets**.

If you choose this option, then you can choose the unregistered nodes (by node 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 \(p. 1540\)](#).

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 **Window start date**, 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 **Window end date**, 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 **Schedule timezone**, specify the time zone to use as the basis for when scheduled maintenance windows run, 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) For **Schedule offset**, enter the number of days to wait after the date and time specified by a cron or rate expression before running the maintenance window. You can specify between one and six days.

**Note**

This option is available only if you specified a schedule by entering a cron or rate expression manually.

14. (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 allow 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

15. 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.

### Note

If a single maintenance window task is registered with multiple targets, its task invocations occur sequentially and not in parallel. If your task must run on multiple targets at the same time, register a task for each target individually and assign each task the same priority level.

### To assign targets to a maintenance window (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Maintenance Windows**.
3. In the list of maintenance windows, choose the maintenance window to add targets to.
4. Choose **Actions**, and then choose **Register targets**.
5. (Optional) For **Target name**, enter a name for the targets.
6. (Optional) For **Description**, enter a description.
7. (Optional) For **Owner information**, specify information to include in any Amazon EventBridge event raised while running tasks for these targets in this maintenance window.

For information about using EventBridge to monitor Systems Manager events, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#).

8. In the **Targets** area, choose one of the options described in the following table.

Option	Description
<b>Specify instance tags</b>	For the <b>Specify instance tags</b> boxes, specify one or more tag keys and (optional) values that have been or will be added to managed nodes in your account. When the maintenance window runs, it attempts to perform tasks on all of the managed nodes to which these tags have been added.  If you specify more than one tag key, a node must be tagged with <i>all</i> the tag keys and values you specify to be included in the target group.
<b>Choose instances manually</b>	From the list, select the box for each node that you want to include in the maintenance window target.  The list includes all nodes in your account that are configured for use with Systems Manager.  If a managed node you expect to see isn't listed, see <a href="#">Troubleshooting managed node availability (p. 816)</a> for troubleshooting tips.  For edge devices and on-premises servers and virtual machines (VMs), see <a href="#">Setting up AWS Systems Manager for hybrid environments (p. 34)</a>

Option	Description
<b>Choose a resource group</b>	<p>For <b>Resource group</b>, choose the name of an existing resource group in your account from the list.</p> <p>For information about creating and working with resource groups, see the following topics:</p> <ul style="list-style-type: none"> <li>• <a href="#">What are resource groups?</a> in the <i>AWS Resource Groups User Guide</i></li> <li>• <a href="#">Resource Groups and Tagging for AWS</a> in the <i>AWS News Blog</i></li> </ul> <p>(Optional) For <b>Resource types</b>, select up to five available resource types, or choose <b>All resource types</b>.</p> <p>If the tasks you assign to the maintenance window don't act on one of the resource types you added to the target, the system might report an error. Tasks for which a supported resource type is found continue to run despite these errors.</p> <p>For example, suppose you add the following resource types to this target:</p> <ul style="list-style-type: none"> <li>• AWS::S3::Bucket</li> <li>• AWS::DynamoDB::Table</li> <li>• AWS::EC2::Instance</li> </ul> <p>But later, when you add tasks to the maintenance window, you include only tasks that perform actions on nodes, such as applying a patch baseline or rebooting a node. In the maintenance window log, an error might be reported for no Amazon Simple Storage Service (Amazon S3) buckets or Amazon DynamoDB tables being found. However, the maintenance window still runs tasks on the nodes in your resource group.</p>

9. Choose **Register target**.

If you want to assign more targets to this maintenance window, choose the **Targets** tab, and then choose **Register target**. With this option, you can choose a different means of targeting. For example, if you previously targeted nodes by node ID, you can register new targets and target nodes by specifying tags applied to managed nodes or choosing resource types from a resource group.

## Assign tasks to a maintenance window (console)

In this procedure, you add a task to a maintenance window. Tasks are the actions performed when a maintenance window runs.

The following four types of tasks can be added to a maintenance window:

- AWS Systems Manager Run Command commands
- Systems Manager Automation workflows
- AWS Lambda functions

**Important**

The IAM policy for Maintenance Windows requires that you add the prefix `ssm` to Lambda function (or alias) names. Before you proceed to register this type of task, update its name in AWS Lambda to include `ssm`. For example, if your Lambda function name is `MyLambdaFunction`, change it to `SSMMyLambdaFunction`.

- AWS Step Functions tasks

### To assign tasks to a maintenance window

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Maintenance Windows**.
3. In the list of maintenance windows, choose a maintenance window.
4. 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**
5. (Optional) For **Name**, enter a name for the task.
6. (Optional) For **Description**, enter a description.
7. For **New task invocation cutoff**, if you don't want any new task invocations to start after the maintenance window cutoff time is reached, choose **Enabled**.

When this option is *not* enabled, the task continues running when the cutoff time is reached and starts new task invocations until completion.

**Note**

The status for tasks that are not completed when you enable this option is `TIMED_OUT`.

8. Do one of the following:
  - For a Run Command task: In the **Command document** list, choose the Systems Manager Command document (SSM document) that defines the tasks to run. For **Document version**, choose the document version to use.
  - For an Automation task: In the **Automation document** list, choose the Automation runbook that defines the tasks to run. For **Document version**, choose the runbook version to use.
  - For a Lambda task: In the **Lambda parameters** area, choose a Lambda function from the list. (Optional) Provide any content for **Payload**, **Client Context**, or **Qualifier** that you want to include.
  - For a Step Functions task: In the **Step Functions parameters** area, choose a state machine from the list. (Optional) Provide a name for the state machine execution and any content for **Input** that you want to include.
9. 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.
10. In the **Targets** area, choose one of the following:
  - **Selecting registered target groups:** Select one or more maintenance window targets you have registered with the current maintenance window.
  - **Selecting unregistered targets:** Choose available resources one by one as targets for the task.

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

- **Task target not required:** Targets for the task might already be specified in other functions for all but Run Command-type tasks.

Specify one or more targets for maintenance window Run Command-type tasks. Depending on the task, targets are optional for other maintenance window task types (Automation, AWS Lambda, and AWS Step Functions). For more information about running tasks that don't specify targets, see [Registering maintenance window tasks without targets \(p. 794\)](#).

**Note**

In many cases, you don't need to explicitly specify a target for an automation task. For example, say that you're creating an Automation-type task to update an Amazon Machine Image (AMI) for Linux using the `AWS-UpdateLinuxAmi` runbook. When the task runs, the AMI is updated with the latest available Linux distribution packages and Amazon software. New instances created from the AMI already have these updates installed. Because the ID of the AMI to be updated is specified in the input parameters for the runbook, there is no need to specify a target again in the maintenance window task.

11. Automation tasks only:

In the **Input parameters** area, provide values for any required or optional parameters needed to run your task.

12. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes still processing the command might also send errors.

13. In the **IAM service role** area, choose one of the following options to provide permissions for Systems Manager to run tasks on your target nodes:

- **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 \(console\) \(p. 709\)](#)
- [Control access to maintenance windows \(AWS CLI\) \(p. 714\)](#)

- [Control access to maintenance windows \(Tools for Windows PowerShell\) \(p. 720\)](#)

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? \(p. 708\)](#).

14. Run Command tasks only:

(Optional) For **Output options**, do the following:

- Select the **Enable writing to S3** check box to save the command output to a file. Enter 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 node, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile associated with the node has the necessary permissions to write to that bucket.

- Select the **CloudWatch output** check box to write complete output to Amazon CloudWatch Logs. Enter the name of a CloudWatch Logs log group.

15. Run Command tasks only:

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 \(p. 1462\)](#).

16. Run Command tasks only:

In the **Parameters** area, specify parameters for the document.

17. Choose **Register task**.

## Updating or deleting maintenance window resources (console)

You can update or delete a maintenance window in Maintenance Windows, a capability of AWS Systems Manager. 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 Systems Manager console. For examples of how to do this by using the AWS Command Line Interface (AWS CLI), see [Tutorial: Update a maintenance window \(AWS CLI\) \(p. 769\)](#).

### Topics

- [Updating or deleting a maintenance window \(console\) \(p. 730\)](#)
- [Updating or deregistering maintenance window targets \(console\) \(p. 731\)](#)
- [Updating or deregistering maintenance window tasks \(console\) \(p. 731\)](#)

## Updating or deleting 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Maintenance Windows**.
3. Select the button next to 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 **Save changes**.

For information about the configuration choices you can make, see [Create a maintenance window \(console\) \(p. 724\)](#).

### Updating or deregistering maintenance window targets (console)

You can update or deregister 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Maintenance Windows**.
3. Choose the name of the maintenance window that you want to update, choose the **Targets** tab, and then do one of the following:
  - To update targets, select the button next to the target to update, and then choose **Edit**.
  - To deregister targets, select the button next to the target to deregister, and then choose **Deregister target**. In the **Deregister maintenance windows target** dialog box, choose **Deregister**.

### Updating or deregistering maintenance window tasks (console)

You can update or deregister 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)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Maintenance Windows**.
3. Choose the name of the maintenance window that you want to update.
4. Choose the **Tasks** tab, and then select the button next to the task to update.
5. Do one of the following:
  - To deregister a task, choose **Deregister task**.
  - To edit the task, choose **Edit**. Change the values and options that you want, and then choose **Edit task**.

# 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 \(p. 60\)](#).
- **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 [Setting up Maintenance Windows \(p. 707\)](#).
- **Create or configure a Systems Manager-compatible instance:** You need at least one Amazon Elastic Compute Cloud (Amazon 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 AWS managed Amazon Machine Image (AMI) with the agent preinstalled. For more information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

For information about installing SSM Agent on an instance, see the following topics:

- [Working with SSM Agent on EC2 instances for Windows Server \(p. 123\)](#)
- [Working with SSM Agent on EC2 instances for Linux \(p. 76\)](#)

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 \(p. 21\)](#)
- [Attach an IAM instance profile to an EC2 instance \(p. 26\)](#)
- **Create additional resources as needed:** Run Command, a capability of Systems Manager, includes many tasks that don't 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 Amazon Elastic Compute Cloud (EC2) instance that is configured to use with Systems Manager, as described earlier in this topic. 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:

- 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:

```
{
 "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-02573cafefEXAMPLE`

### Tutorials

- [Tutorial: Create and configure a maintenance window \(AWS CLI\) \(p. 733\)](#)
- [Tutorial: View information about maintenance windows \(AWS CLI\) \(p. 755\)](#)
- [Tutorial: View information about tasks and task executions \(AWS CLI\) \(p. 766\)](#)
- [Tutorial: Update a maintenance window \(AWS CLI\) \(p. 769\)](#)
- [Tutorial: Delete a maintenance window \(AWS CLI\) \(p. 777\)](#)

## Tutorial: Create and configure a maintenance window (AWS CLI)

This tutorial demonstrates how to use the AWS Command Line Interface (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-02573cafefEXAMPLE* with IDs of resources you create.

### Contents

- [Step 1: Create the maintenance window \(AWS CLI\) \(p. 734\)](#)
- [Step 2: Register a target node with the maintenance window \(AWS CLI\) \(p. 735\)](#)
- [Step 3: Register a task with the maintenance window \(AWS CLI\) \(p. 739\)](#)

## 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 wouldn't run a maintenance window this frequently. However, with this rate you can see your tutorial results quickly. We will show you how to change to a less frequent rate after the task has run successfully.

### Note

For an explanation of how the various schedule-related options for maintenance windows relate to one another, see [Maintenance window scheduling and active period options \(p. 790\)](#).

For more information about working with the --schedule option, see [Reference: Cron and rate expressions for Systems Manager \(p. 1540\)](#).

### To create a maintenance window (AWS CLI)

1. Open the AWS Command Line Interface (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 operation.
  - 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 & macOS

```
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"
```

#### Windows

```
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 the following command to view details about this and any other maintenance windows already in your account.

```
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,
 "NextExecutionTime": "2019-05-11T16:46:16.991Z"
 }
]
}
```

Continue to [Step 2: Register a target node with the maintenance window \(AWS CLI\) \(p. 735\)](#).

## Step 2: Register a target node with the maintenance window (AWS CLI)

In this step, you register a target with your new maintenance window. In this case, you specify which node to update when the maintenance window runs.

For an example of registering more than one node at a time using node IDs, examples of using tags to identify multiple nodes, and examples of specifying resource groups as targets, see [Examples: Register targets with a maintenance window \(p. 736\)](#).

### Note

You should already have created an Amazon Elastic Compute Cloud (Amazon EC2) instance to use in this step, as described in the [Maintenance Windows tutorial prerequisites \(p. 732\)](#).

### To register a target node with a maintenance window (AWS CLI)

- Run the following command on your local machine:

Linux & macOS

```
aws ssm register-target-with-maintenance-window \
 --window-id "mw-0c50858d01EXAMPLE" \
 --resource-type "INSTANCE" \
 --target "Key=InstanceIds,Values=i-02573cafcfEXAMPLE"
```

Windows

```
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"
}
```

- Now run the following command on your local machine to view details about your maintenance window target:

## Linux & macOS

```
aws ssm describe-maintenance-window-targets \
--window-id "mw-0c50858d01EXAMPLE"
```

## Windows

```
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-02573cafccEXAMPLE"
]
 }
]
 }
]
}
```

Continue to [Step 3: Register a task with the maintenance window \(AWS CLI\) \(p. 739\)](#).

### Examples: Register targets with a maintenance window

You can register a single node as a target using its node ID, as demonstrated in [Step 2: Register a target node with the maintenance window \(AWS CLI\) \(p. 735\)](#). You can also register one or more nodes as targets using the command formats on this page.

In general, there are two methods for identifying the nodes you want to use as maintenance window targets: specifying individual nodes, 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 nodes and many other types of supported AWS resources. Examples 4 and 5, next, demonstrate how to add resource groups to your maintenance window targets.

#### Note

If a single maintenance window task is registered with multiple targets, its task invocations occur sequentially and not in parallel. If your task must run on multiple targets at the same time, register a task for each target individually and assign each task the same priority level.

For more information about creating and managing resource groups, see [What are resource groups?](#) in the *AWS Resource Groups User Guide* and [Resource Groups and Tagging for AWS](#) in the *AWS News Blog*.

For information about quotas for Maintenance Windows, a capability of AWS Systems Manager, in addition to those specified in the following examples, see [Systems Manager service quotas](#) in the *Amazon Web Services General Reference*.

### Example 1: Register multiple targets using node IDs

Run the following command on your local machine format to register multiple nodes as targets using their node IDs.

Linux & macOS

```
aws ssm register-target-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--resource-type "INSTANCE" \
--target
"Key=InstanceIds,Values=i-02573cafefEXAMPLE,i-0471e04240EXAMPLE,i-07782c72faEXAMPLE"
```

Windows

```
aws ssm register-target-with-maintenance-window ^
--window-id "mw-0c50858d01EXAMPLE" ^
--resource-type "INSTANCE" ^
--target
"Key=InstanceIds,Values=i-02573cafefEXAMPLE,i-0471e04240EXAMPLE,i-07782c72faEXAMPLE"
```

**Recommended use:** Most useful when registering a unique group of nodes with any maintenance window for the first time and they do *not* share a common node tag.

**Quotas:** You can specify up to 50 nodes total for each maintenance window target.

### Example 2: Register targets using resource tags applied to nodes

Run the following command on your local machine to register nodes that are all already tagged with a key-value pair you have assigned.

Linux & macOS

```
aws ssm register-target-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--resource-type "INSTANCE" \
--target "Key>tag:Region,Values=East"
```

Windows

```
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 nodes with any maintenance window for the first time and they *do* share a common node tag.

**Quotas:** You can specify up to five key-value pairs total for each target. If you specify more than one key-value pair, a node 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 nodes with the tag-key **Patch Group** and assign the nodes a common key value, such as `my-patch-group`. Patch Manager, a capability of Systems Manager, evaluates the **Patch Group** key on nodes 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 allows you to use a maintenance window to update patches on a group of nodes that are already associated with the same patch baseline. For more information, see [About patch groups \(p. 1163\)](#).

### Example 3: Register targets using a group of tag keys (without tag values)

Run the following command on your local machine to register nodes that all have one or more tag keys assigned to them, regardless of their key values.

Linux & macOS

```
aws ssm register-target-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--resource-type "INSTANCE" \
--target "Key=tag-key,Values=Name,Instance-Type,CostCenter"
```

Windows

```
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 nodes 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, a node 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 don't 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 & macOS

```
aws ssm register-target-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--resource-type "RESOURCE_GROUP" \
--target "Key=resource-groups:Name,Values=MyResourceGroup"
```

Windows

```
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 & macOS

```
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

```
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 an AWS Systems Manager Run Command task that runs the `df` command on your Amazon Elastic Compute Cloud (Amazon 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're targeting an Amazon 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're ready to register other task types, or use more of the available Systems Manager Run Command options, see [Examples: Register tasks with a maintenance window \(p. 743\)](#). 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 & macOS

```
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

```
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-02573cafefEXAMPLE" ^
--task-type "RUN_COMMAND" ^
--task-invocation-parameters={"RunCommand":{"Parameters":{"commands":["df"]}}}
```

The system returns information similar to the following:

```
{
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
```

- Now run the following command to view details about the maintenance window task you created.

### Linux & macOS

```
aws ssm describe-maintenance-window-tasks \
--window-id mw-0c50858d01EXAMPLE
```

### Windows

```
aws ssm describe-maintenance-window-tasks ^
--window-id mw-0c50858d01EXAMPLE
```

- The system returns information similar to the following.

```
{
 "Tasks": [
 {
 "WindowId": "mw-0c50858d01EXAMPLE",
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
 "TaskArn": "AWS-RunShellScript",
 "Type": "RUN_COMMAND",
 "Targets": [
 {
 "Key": "InstanceIds",
 "Values": [
 "i-02573cafefEXAMPLE"
]
 }
],
 "TaskParameters": {},
 "Priority": 10,
 "ServiceRoleArn": "arn:aws:iam::123456789012:role/
MyMaintenanceWindowServiceRole",
 "MaxConcurrency": "1",
 "MaxErrors": "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 \(AWS CLI\) \(p. 734\)](#). 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 & macOS

```
aws ssm describe-maintenance-window-executions \
--window-id mw-0c50858d01EXAMPLE
```

Windows

```
aws ssm describe-maintenance-window-executions ^
--window-id mw-0c50858d01EXAMPLE
```

The system returns information similar to the following.

```
{
 "WindowExecutions": [
 {
 "WindowId": "mw-0c50858d01EXAMPLE",
 "WindowExecutionId": "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",
 "Status": "SUCCESS",
 "StartTime": 1557593493.096,
 "EndTime": 1557593498.611
 }
]
}
```

**Tip**

After the task runs 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 & macOS

```
aws ssm update-maintenance-window \
--window-id mw-0c50858d01EXAMPLE \
--schedule "rate(7 days)"
```

Windows

```
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 \(p. 1540\)](#) and [Maintenance window scheduling and active period options \(p. 790\)](#).

For information about using the AWS Command Line Interface (AWS CLI) to modify a maintenance window, see [Tutorial: Update a maintenance window \(AWS CLI\) \(p. 769\)](#).

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 \(AWS CLI\) \(p. 766\)](#).

### 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 Amazon Simple Storage Service (Amazon 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

Ethernet adapter Ethernet 2:

Connection-specific DNS Suffix . : example.com
IPv4 Address : 10.24.34.0/23
Subnet Mask : 255.255.255.255
Default Gateway : 0.0.0.0

Ethernet adapter Ethernet:

Media State : Media disconnected
Connection-specific DNS Suffix . : abc1.wa.example.net

Wireless LAN adapter Local Area Connection* 1:

Media State : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :
Link-local IPv6 Address : fe80::100b:c234:66d6:d24f%4
IPv4 Address : 192.0.2.0
Subnet Mask : 255.255.255.0
Default Gateway : 192.0.2.0

Ethernet adapter Bluetooth Network Connection:

Media State : Media disconnected
Connection-specific DNS Suffix . :
```

## Examples: Register tasks with a maintenance window

You can register a task in Run Command, a capability of AWS Systems Manager, with a maintenance window using the AWS Command Line Interface (AWS CLI), as demonstrated in [Register tasks with the maintenance window \(p. 739\)](#). You can also register tasks for Systems Manager Automation workflows, AWS Lambda functions, and AWS Step Functions tasks, as demonstrated later in this topic.

### Note

Specify one or more targets for maintenance window Run Command-type tasks. Depending on the task, targets are optional for other maintenance window task types (Automation, AWS Lambda, and AWS Step Functions). For more information about running tasks that don't specify targets, see [Registering maintenance window tasks without targets \(p. 794\)](#).

In this topic, we provide examples of using the AWS Command Line Interface (AWS CLI) command `register-task-with-maintenance-window` 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`.

#### Linux & macOS

```
aws ssm register-task-with-maintenance-window \
--cli-input-json file://MyFile.json
```

#### Windows

```
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, `{{TARGET_ID}}` and `{{RESOURCE_ID}}` can be 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 \(p. 751\)](#).

### More information

For information about some fundamental `register-task-with-maintenance-window` options, see [About register-task-with-maintenance-windows options \(p. 748\)](#).

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.

#### Linux & macOS

```
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-02573cafefEXAMPLE" \
--task-type "RUN_COMMAND" \
--task-invocation-parameters '{"RunCommand":{"Parameters":{"commands":["df"]}}}'
```

#### Windows

```
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-02573cafefEXAMPLE" ^
--task-type "RUN_COMMAND" ^
--task-invocation-parameters "{\"RunCommand\":{\"Parameters\":{\"commands\":[\"df\"]}}}"
```

### JSON content to use with `--cli-input-json` file option:

```
{
 "TaskType": "RUN_COMMAND",
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Description": "My Run Command task to update SSM Agent on an instance",
 "MaxConcurrency": "1",
 "MaxErrors": "1",
 "Name": "My-Run-Command-Task",
 "Priority": 10,
 "Targets": [
 {
 "Key": "WindowTargetIds",
 "Values": [
 "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
]
 }
],
 "TaskArn": "AWS-UpdateSSMAgent",
 "TaskInvocationParameters": {
 "RunCommand": {
 "Comment": "A TaskInvocationParameters test comment",
 "NotificationConfig": {
 "NotificationArn": "arn:aws:sns:region:123456789012:my-sns-topic-name",
 "NotificationEvents": [
 "All"
],
 "NotificationType": "Invocation"
 }
 }
 }
}
```

```

 },
 "OutputsS3BucketName": "DOC-EXAMPLE-BUCKET",
 "OutputsS3KeyPrefix": "DOC-EXAMPLE-FOLDER",
 "TimeoutSeconds": 3600
 }
}

```

## 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 & macOS

```
aws ssm register-task-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--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 --name "My-Restart-EC2-Instances-Automation-Task" \
--description "Automation task to restart EC2 instances"
```

Windows

```
aws ssm register-task-with-maintenance-window ^
--window-id "mw-0c50858d01EXAMPLE" ^
--task-arn "AWS-RestartEC2Instance" ^
--service-role-arn arn:aws:iam::123456789012:role/MyMaintenanceWindowServiceRole ^
--task-type AUTOMATION ^
--task-invocation-parameters
"Automation={DocumentVersion=5,Parameters={InstanceId='{{TARGET_ID}}'}}" ^
--priority 0 --name "My-Restart-EC2-Instances-Automation-Task" ^
--description "Automation task to restart EC2 instances"
```

### JSON content to use with `--cli-input-json` file option:

```
{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "TaskArn": "AWS-PatchInstanceWithRollback",
 "TaskType": "AUTOMATION", "TaskInvocationParameters": {
 "Automation": {
 "DocumentVersion": "1",
 "Parameters": {
 "InstanceId": [
 "{{RESOURCE_ID}}"
]
 }
 }
 }
}
```

## Register an AWS Lambda task

The following examples demonstrate how to register 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 add the prefix `SSM` to Lambda function (or alias) names. Before you proceed to register this type of task, 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 & macOS

**Important**

If you are using version 2 of the AWS CLI, you must include the option `--cli-binary-format raw-in-base64-out` in the following command if your Lambda payload is not base64 encoded. The `cli_binary_format` option is available only in version 2. For information about this and other AWS CLI config file settings, see [Supported config file settings](#) in the *AWS Command Line Interface User Guide*.

```
aws ssm register-task-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-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-arn "arn:aws:lambda:region:123456789012:function:serverlessrepo-SSMrestart-
my-instances-C4JF9EXAMPLE" \
--task-invocation-parameters '{"Lambda":{"Payload":"{\\"InstanceId\\":\
\"{{RESOURCE_ID}}\\",\\"targetType\\":\"{{TARGET_TYPE}}\\\"}}","Qualifier": "$LATEST"}'`
```

PowerShell

**Important**

If you are using version 2 of the AWS CLI, you must include the option `--cli-binary-format raw-in-base64-out` in the following command if your Lambda payload is not base64 encoded. The `cli_binary_format` option is available only in version 2. For information about this and other AWS CLI config file settings, see [Supported config file settings](#) in the *AWS Command Line Interface User Guide*.

```
aws ssm register-task-with-maintenance-window `
--window-id "mw-0c50858d01EXAMPLE" `
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-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-arn "arn:aws:lambda:region:123456789012:function:serverlessrepo-SSMrestart-
my-instances-C4JF9EXAMPLE" `
--task-invocation-parameters '{\"Lambda\":{\"Payload\":\"{\\"InstanceId\\\":\
\"{{RESOURCE_ID}}\\\",\\\"targetType\\\":\"{{TARGET_TYPE}}\\\"}\",\\\"Qualifier\\\":
\"$LATEST\"}}'
```

**JSON content to use with `--cli-input-json` file option:**

```
{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Targets": [
 {
 "Key": "WindowTargetIds",
 "Values": [
 "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
]
 }
]
}
```

```

 },
],
 "TaskArn": "SSM_RestartMyInstances",
 "TaskType": "LAMBDA",
 "MaxConcurrency": "10",
 "MaxErrors": "10",
 "TaskInvocationParameters": {
 "Lambda": {
 "ClientContext": "ewOKICAi--truncated--0KIEEXAMPLE",
 "Payload": "{ \"instanceId\": \"{{RESOURCE_ID}}\", \"targetType\": \"{{TARGET_TYPE}}\" }",
 "Qualifier": "$LATEST"
 }
 },
 "Name": "My-Lambda-Task",
 "Description": "A description for my LAMBDA task",
 "Priority": 5
}

```

## Register a Step Functions task

The following examples demonstrate how to register 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 AWS Identity and Access Management (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 & macOS

```

aws ssm register-task-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" \
--task-arn arn:aws:states:region:123456789012:stateMachine:SSMMyStateMachine-
MggiqEXAMPLE \
--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"

```

#### PowerShell

```

aws ssm register-task-with-maintenance-window `
--window-id "mw-0c50858d01EXAMPLE" `
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" `
--task-arn arn:aws:states:region:123456789012:stateMachine:SSMMyStateMachine-
MggiqEXAMPLE `
--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:**

```
{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Targets": [
 {
 "Key": "WindowTargetIds",
 "Values": [
 "e32eecb2-646c-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 operates on. The value that you specify depends on the type of task you're registering, as described in the following table.

#### TaskArn formats for maintenance window tasks

Maintenance window task type	TaskArn value
<b>RUN_COMMAND</b> and <b>AUTOMATION</b>	TaskArn is the SSM document name or Amazon Resource Name (ARN). For example:  AWS-RunBatchShellScript  -or-  arn:aws:ssm: <i>region</i> :111122223333:document/ My-Document.
<b>LAMBDA</b>	TaskArn is the function name or ARN. For example:  SSMMy-Lambda-Function  -or-

Maintenance window task type	TaskArn value
	<p>arn:aws:lambda:<b>region</b>:111122223333:function:SSMMY</p> <p><b>Important</b> The IAM policy for Maintenance Windows requires that you add the prefix SSM to Lambda function (or alias) names. Before you proceed to register this type of task, update its name in AWS Lambda to include SSM. For example, if your Lambda function name is MyLambdaFunction, change it to SSMMyLambdaFunction.</p>
<b>STEP_FUNCTIONS</b>	<p>TaskArn is the state machine ARN. For example:</p> <p>arn:aws:states:us-east-2:111122223333:stateMachine:SSMMystateMachine</p> <p><b>Important</b> 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.</p>

## About the --service-role-arn option

The role for AWS Systems Manager to assume when running the maintenance window task.

Specifying a service role ARN is optional. If you don't specify a service role ARN, Systems Manager creates a service-linked role or uses your account's service-linked role.

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? \(p. 708\)](#)

## 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 \(p. 751\)](#).

## Task invocation parameters options for maintenance window tasks

Maintenance window task type	Available parameters	Example
RUN_COMMAND	Comment DocumentHash DocumentHashType	<pre>"TaskInvocationParameters": {     "RunCommand": {         "Comment": "My Run Command task comment",         "DocumentHash": "12345678901234567890123456789012",         "DocumentHashType": "SHA256"     } }</pre>

Maintenance window task type	Available parameters	Example
	NotificationConfig OutputS3BucketName OutPutS3KeyPrefix Parameters ServiceRoleArn TimeoutSeconds	<pre>     "DocumentHash": "6554ed3d--truncated--5EXAMPLE",     "DocumentHashType": "Sha256",     "NotificationConfig": {         "NotificationArn": "arn:aws:sns:<b>region</b>:123456789012:my-sns-topic-name",         "NotificationEvents": [             "FAILURE"         ],         "NotificationType": "Invocation"     },     "OutputS3BucketName": "<b>DOC-EXAMPLE-BUCKET</b>",     "OutputS3KeyPrefix": "<b>DOC-EXAMPLE-FOLDER</b>",     "Parameters": {         "commands": [             {                 "Get-ChildItem\$env: temp-Recurse Remove-Item-Recurse-force"             }         ],         "ServiceRoleArn": "arn:aws:iam::123456789012:role/MyMaintenanceWindowServiceRole",         "TimeoutSeconds": 3600     } } </pre>
AUTOMATION	DocumentVersion Parameters	<pre> "TaskInvocationParameters": {     "Automation": {         "DocumentVersion": "3",         "Parameters": {             "instanceid": [                 "{{TARGET_ID}}"             ]         }     } } </pre>

Maintenance window task type	Available parameters	Example
LAMBDA	ClientContext Payload Qualifier	<pre>"TaskInvocationParameters": {     "Lambda": {         "ClientContext": "ewOKICAI-- truncated--OKIEXAMPLE",         "Payload": "{"\\"targetId\\": "\\"{{TARGET_ID}}\\", "\\"targetType\\": "\\"{{TARGET_TYPE}}\\\"",         "Qualifier": "\\$LATEST"     } }</pre>
STEP_FUNCTIONS	Input Name	<pre>"TaskInvocationParameters": {     "StepFunctions": {         "Input": "{"\\"targetId\\": "\\"{{TARGET_ID}}\\\"",         "Name": "\\"{{INVOCATION_ID}}\"     } }</pre>

## About pseudo parameters

When you register a task in Maintenance Windows, a capability of AWS Systems Manager, 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 \(p. 753\)](#).

### 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-02573cafefEXAMPLE`
- `\{\{RESOURCE_ID\}\}` format: `i-02573cafefEXAMPLE`

For target type `INSTANCE`, both the `\{\{TARGET_ID\}\}` and `\{\{RESOURCE_ID\}\}` parameters yield the instance ID only. For more information, see [Supported pseudo parameters \(p. 753\)](#). `\{\{TARGET_ID\}\}` and `\{\{RESOURCE_ID\}\}` can be used to pass IDs of AWS resources only to Automation, Lambda, and Step Functions tasks. These two pseudo parameters can't be used with Run Command tasks.

## Pseudo parameter examples

Suppose that your payload for an AWS 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. For example:

```
"TaskArn": "arn:aws:lambda:us-east-2:111122223333:function:SSMTestFunction",
"TaskType": "LAMBDA",
"TaskInvocationParameters": {
 "Lambda": {
 "ClientContext": "ewOKICAI--truncated--0KIEEXAMPLE",
 "Payload": "{ \"instanceId\": \"{{RESOURCE_ID}}\" }",
 "Qualifier": "$LATEST"
 }
}
```

If your Lambda task is intended to run against another supported target type in addition to Amazon Elastic Compute Cloud (Amazon EC2) instances, such as an Amazon 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.

```
"TaskArn": "arn:aws:lambda:us-east-2:111122223333:function:SSMTestFunction",
"TaskType": "LAMBDA",
"TaskInvocationParameters": {
 "Lambda": {
 "ClientContext": "ewOKICAI--truncated--0KIEEXAMPLE",
 "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 \(p. 753\)](#).

As another example, to run an Automation task that stops your EC2 instances, you specify the `AWS-StopEC2Instance` Systems Manager document (SSM document) as the `TaskArn` value and use the `{ { RESOURCE_ID } }` pseudo parameter:

```
"TaskArn": "AWS-StopEC2Instance",
"TaskType": "AUTOMATION",
"TaskInvocationParameters": {
 "Automation": {
 "DocumentVersion": "1",
 "Parameters": {
 "instanceId": [
 "{ { RESOURCE_ID } }"
]
 }
 }
}
```

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 running.
- **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.

**Note**

TARGET\_ID isn't supported for Run Command tasks.

Resource type	Example TARGET_ID
AWS::EC2::Instance	<code>arn:aws:ec2:us-east-1:123456789012:instance/i-078a280217EXAMPLE</code>
AWS::EC2::Image	<code>arn:aws:ec2:us-east-1:123456789012:image/ami-02250b3732EXAMPLE</code>
AWS::EC2::SecurityGroup	<code>arn:aws:ec2:us-east-1:123456789012:security-group/sg-cEXAMPLE</code>
AWS::EC2::Snapshot	<code>arn:aws:ec2:us-east-1:123456789012:snapshot/snap-03866bf003EXAMPLE</code>
AWS::EC2::Volume	<code>arn:aws:ec2:us-east-1:123456789012:volume/vol-0912e04d78EXAMPLE</code>

Resource type	Example TARGET_ID
AWS::DynamoDB::Table	arn:aws:dynamodb:us-east-1:123456789012:table/MyTable
AWS::RDS::DBCluster	arn:aws:rds:us-east-2:123456789012:cluster:My-Cluster
AWS::RDS::DBInstance	arn:aws:rds:us-east-1:123456789012:db:My-SQL-Instance
AWS::S3::Bucket	arn:aws:s3:::DOC-EXAMPLE-BUCKET
AWS::SSM::ManagedInstance	arn:aws:ssm:us-east-1:123456789012:managed-instance/mi-0feadcf2d9EXAMPLE

- **RESOURCE\_ID:** The short ID of a resource type contained in a resource group. The following table provides sample RESOURCE\_ID values for particular resource types in a resource group.

**Note**

RESOURCE\_ID isn't supported for Run Command tasks.

Resource type	Example RESOURCE_ID
AWS::EC2::Instance	i-078a280217EXAMPLE
AWS::EC2::Image	ami-02250b3732EXAMPLE
AWS::EC2::SecurityGroup	sg-cEXAMPLE
AWS::EC2::Snapshot	snap-03866bf003EXAMPLE
AWS::EC2::Volume	vol-0912e04d78EXAMPLE
AWS::DynamoDB::Table	MyTable
AWS::RDS::DBCluster	My-Cluster
AWS::RDS::DBInstance	My-SQL-Instance
AWS::S3::Bucket	DOC-EXAMPLE-BUCKET
AWS::SSM::ManagedInstance	mi-0feadcf2d9EXAMPLE

**Note**

If the AWS resource group you specify includes resource types that don't yield a RESOURCE\_ID value, and aren't listed in the preceding table, then the RESOURCE\_ID parameter isn't 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 AWS Command Line Interface (AWS CLI), see [Installing, updating, and uninstalling the AWS CLI](#) and [Configuring the AWS CLI](#).

### Command examples

- Examples for 'describe-maintenance-windows' (p. 755)
- Examples for 'describe-maintenance-window-targets' (p. 757)
- Examples for 'describe-maintenance-window-tasks' (p. 758)
- Examples for 'describe-maintenance-windows-for-target' (p. 761)
- Examples for 'describe-maintenance-window-executions' (p. 761)
- Examples for 'describe-maintenance-window-schedule' (p. 763)

### 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.

```
{
 "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.

```
aws ssm describe-maintenance-windows --filters "Key=Enabled,Values=true"
```

The system returns information like the following.

```
{
 "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 disabled maintenance windows

Run the following command.

```
aws ssm describe-maintenance-windows --filters "Key=Enabled,Values=false"
```

The system returns information like the following.

```
{
 "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.

```
aws ssm describe-maintenance-windows --filters "Key=Name,Values=My"
```

The system returns information like the following.

```
{
 "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"
 }
]
}
```

```

 "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"
 },
 {
 "WindowId": "mw-6e5c9d4b7cEXAMPLE",
 "Name": "My-Disabled-Maintenance-Window",
 "Enabled": false,
 "Duration": 2,
 "Cutoff": 1
 }
]
}

```

## Examples for 'describe-maintenance-window-targets'

### Display the targets for a maintenance window matching a specific owner information value

Run the following command.

**Linux & macOS**

```
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": "e32eeccb2-646c-4f4b-8ed1-205fbEXAMPLE",
 "ResourceType": "INSTANCE",
 "Targets": [
 {
 "Key": "tag:Name",
 "Values": [
 "Production"
]
 }
]
 }
]
}
```

```
],
 "OwnerInformation": "CostCenter1",
 "Name": "Target1"
 }
}
```

## Examples for 'describe-maintenance-window-tasks'

### Show all registered tasks that invoke the SSM command document AWS-RunPowerShellScript

Run the following command.

Linux & macOS

```
aws ssm describe-maintenance-window-tasks \
--window-id "mw-0c50858d01EXAMPLE" \
--filters "Key=TaskArn,Values=AWS-RunPowerShellScript"
```

Windows

```
aws ssm describe-maintenance-window-tasks ^
--window-id "mw-0c50858d01EXAMPLE" ^
--filters "Key=TaskArn,Values=AWS-RunPowerShellScript"
```

The system returns information like the following.

```
{
 "Tasks": [
 {
 "ServiceRoleArn": "arn:aws:iam::111122223333:role/MyMaintenanceWindowServiceRole",
 "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-02573cafccEXAMPLE",
 "TaskTargetType": "INSTANCE"
 }
]
 },
 {
 "ServiceRoleArn": "arn:aws:iam::111122223333:role/MyMaintenanceWindowServiceRole",
 "MaxErrors": "1",
 "TaskArn": "AWS-RunPowerShellScript",
 "MaxConcurrency": "1",
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
 "TaskParameters": {
 "commands": {
 "Values": [

```

```
 "ipconfig"
]
}
"Priority":1,
"Type":"RUN_COMMAND",
"Targets":[
{
 "TaskTargetId":"i-02573cafccEXAMPLE",
 "TaskTargetType":"WINDOW_TARGET"
}
]
}
```

### Show all registered tasks that have a priority of "3"

Run the following command.

Linux & macOS

```
aws ssm describe-maintenance-window-tasks \
--window-id "mw-9a8b7c6d5eEXAMPLE" \
--filters "Key=Priority,Values=3"
```

Windows

```
aws ssm describe-maintenance-window-tasks ^
--window-id "mw-9a8b7c6d5eEXAMPLE" ^
--filters "Key=Priority,Values=3"
```

The system returns information like the following.

```
{
 "Tasks": [
 {
 "ServiceRoleArn": "arn:aws:iam::111122223333:role/MyMaintenanceWindowServiceRole",
 "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-02573cafccEXAMPLE",
 "TaskTargetType": "INSTANCE"
 }
]
 }
]
}
```

### Show all registered tasks that have a priority of "1" and use Run Command

Run the following command.

Linux & macOS

```
aws ssm describe-maintenance-window-tasks \
--window-id "mw-0c50858d01EXAMPLE" \
--filters "Key=Priority,Values=1" "Key=TaskType,Values=RUN_COMMAND"
```

Windows

```
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-02573cafefEXAMPLE"
]
 }
],
 "TaskParameters": {},
 "Priority": 1,
 "ServiceRoleArn": "arn:aws:iam::111122223333:role/
MyMaintenanceWindowServiceRole",
 "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::111122223333:role/
MyMaintenanceWindowServiceRole",
 "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 node

Run the following command.

Linux & macOS

```
aws ssm describe-maintenance-windows-for-target \
 --resource-type INSTANCE \
 --targets "Key=InstanceIds,Values=i-02573cafefEXAMPLE" \
 --max-results 10
```

Windows

```
aws ssm describe-maintenance-windows-for-target ^
 --resource-type INSTANCE ^
 --targets "Key=InstanceIds,Values=i-02573cafefEXAMPLE" ^
 --max-results 10
```

The system returns information like the following.

```
{
 "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 & macOS

```
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 & macOS**

```
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",
 "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
 }
]
}
```

## Examples for 'describe-maintenance-window-schedule'

### Display the next ten scheduled maintenance window runs for a particular node

Run the following command.

Linux & macOS

```
aws ssm describe-maintenance-window-schedule \
--resource-type INSTANCE \
--targets "Key=InstanceIds,Values=i-07782c72faEXAMPLE" \
--max-results 10
```

Windows

```
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.

```
{
 "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"
 },
 {
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Name": "My-First-Maintenance-Window",
 "ExecutionTime": "2019-06-08T23:35:24.902Z"
 },
 {
 "WindowId": "mw-9a8b7c6d5eEXAMPLE",
 "Name": "My-Second-Maintenance-Window",
 "ExecutionTime": "2019-06-15T23:35:24.902Z"
 }
]
}
```

```

 "WindowId": "mw-0c50858d01EXAMPLE",
 "Name": "My-First-Maintenance-Window",
 "ExecutionTime": "2019-06-22T23:35:24.902Z"
 },
 {
 "WindowId": "mw-9a8b7c6d5eEXAMPLE",
 "Name": "My-Second-Maintenance-Window",
 "ExecutionTime": "2019-06-29T23:35:24.902Z"
 },
 {
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Name": "My-First-Maintenance-Window",
 "ExecutionTime": "2019-07-06T23:35:24.902Z"
 },
 {
 "WindowId": "mw-9a8b7c6d5eEXAMPLE",
 "Name": "My-Second-Maintenance-Window",
 "ExecutionTime": "2019-07-13T23:35:24.902Z"
 },
 {
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Name": "My-First-Maintenance-Window",
 "ExecutionTime": "2019-07-20T23:35:24.902Z"
 }
],
"NextToken": "AAEABUXdceT92FvtKld/dGHELj5Mi+GKW/EXAMPLE"
}

```

### Display the maintenance window schedule for nodes tagged with a certain key-value pair

Run the following command.

**Linux & macOS**

```
aws ssm describe-maintenance-window-schedule \
--resource-type INSTANCE \
--targets "Key=tag:prod,Values=rhel7"
```

**Windows**

```
aws ssm describe-maintenance-window-schedule ^
--resource-type INSTANCE ^
--targets "Key=tag:prod,Values=rhel7"
```

The system returns information like the following.

```
{
 "ScheduledWindowExecutions": [
 {
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Name": "DemoRateStartDate",
 "ExecutionTime": "2019-10-20T05:34:56-07:00"
 },
 {
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Name": "DemoRateStartDate",
 "ExecutionTime": "2019-10-21T05:34:56-07:00"
 },
 {
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Name": "DemoRateStartDate",
 "ExecutionTime": "2019-10-22T05:34:56-07:00"
 }
]
}
```

```
 },
 {
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Name": "DemoRateStartDate",
 "ExecutionTime": "2019-10-23T05:34:56-07:00"
 },
 {
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Name": "DemoRateStartDate",
 "ExecutionTime": "2019-10-24T05:34:56-07:00"
 }
],
 "NextToken": "AAEABccwSXqQRGKitZ1yzGELR6cxW4W/EXAMPLE"
}
```

### Display start times for next four runs of a maintenance window

Run the following command.

Linux & macOS

```
aws ssm describe-maintenance-window-schedule \
--window-id "mw-0c50858d01EXAMPLE" \
--max-results "4"
```

Windows

```
aws ssm describe-maintenance-window-schedule ^
--window-id "mw-0c50858d01EXAMPLE" ^
--max-results "4"
```

The system returns information like the following.

```
{
 "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"
 },
 {
 "ExecutionTime": "2019-10-18T10:10:10Z",
 "Name": "My-First-Maintenance-Window",
 "WindowId": "mw-0c50858d01EXAMPLE"
 },
 {
 "ExecutionTime": "2019-10-25T10:10:10Z",
 "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 Command Line Interface (AWS CLI) to view details about your completed maintenance window tasks.

If you're continuing directly from [Tutorial: Create and configure a maintenance window \(AWS CLI\) \(p. 733\)](#), 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.

Linux & macOS

```
aws ssm describe-maintenance-window-executions \
--window-id "mw-0c50858d01EXAMPLE"
```

Windows

```
aws ssm describe-maintenance-window-executions ^
--window-id "mw-0c50858d01EXAMPLE"
```

The system returns information like the following.

```
{
 "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.

Linux & macOS

```
aws ssm get-maintenance-window-execution \
--window-execution-id "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE"
```

Windows

```
aws ssm get-maintenance-window-execution ^
--window-execution-id "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE"
```

The system returns information like the following.

```
{
 "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.

Linux & macOS

```
aws ssm describe-maintenance-window-execution-tasks \
--window-execution-id "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE"
```

Windows

```
aws ssm describe-maintenance-window-execution-tasks ^
--window-execution-id "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE"
```

The system returns information like the following.

```
{
 "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.

Linux & macOS

```
aws ssm get-maintenance-window-execution-task \
--window-execution-id "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE" \
```

```
--task-id "c9b05aba-197f-4d8d-be34-e73fbEXAMPLE"
```

## Windows

```
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.

```
{
 "WindowExecutionId": "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",
 "TaskExecutionId": "c9b05aba-197f-4d8d-be34-e73fbEXAMPLE",
 "TaskArn": "AWS-RunShellScript",
 "ServiceRole": "arn:aws:iam::111122223333:role/MyMaintenanceWindowServiceRole",
 "Type": "RUN_COMMAND",
 "TaskParameters": [
 {
 "aws:InstanceId": {
 "Values": [
 "i-02573cafefEXAMPLE"
]
 },
 "commands": {
 "Values": [
 "df"
]
 }
 }
],
 "Priority": 10,
 "MaxConcurrency": "1",
 "MaxErrors": "1",
 "Status": "SUCCESS",
 "StartTime": 1557593493.162,
 "EndTime": 1557593498.57
}
```

5. Run the following command to get the specific task invocations performed for a task execution.

## Linux & macOS

```
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\": [
 \"i-02573cafefEXAMPLE\"]
], \"maxConcurrency\": \"1\", \"maxErrors\": \"1\", \"parameters\": {
 \"commands\": [\"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 Command Line Interface (AWS CLI) to update a maintenance window. It also shows you how to update different task types, including those for AWS Systems Manager Run Command and 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 [Updating or deleting maintenance window resources \(console\) \(p. 730\)](#).

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-02573cafefEXAMPLE* 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 & macOS

```
aws ssm update-maintenance-window-target \
--window-id "mw-0c50858d01EXAMPLE" \
--window-target-id "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" \
--name "My-Maintenance-Window-Target" \
--description "Description for my maintenance window target"
```

Windows

```
aws ssm update-maintenance-window-target ^
--window-id "mw-0c50858d01EXAMPLE" ^
--window-target-id "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" ^
--name "My-Maintenance-Window-Target" ^
--description "Description for my maintenance window target"
```

The system returns information like the following.

```
{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE",
 "Targets": [
 {
 "Key": "InstanceIds",
 "Values": [
 "i-02573cafefEXAMPLE"
]
 }
],
 "Name": "My-Maintenance-Window-Target",
 "Description": "Description for my maintenance window target"
}
```

- 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 doesn't include the field (a null value). Be sure to specify an additional node that has been configured for use with Systems Manager.

Linux & macOS

```
aws ssm update-maintenance-window-target \
 --window-id "mw-0c50858d01EXAMPLE" \
 --window-target-id "d208dedf-3f6b-41ff-ace8-8e751EXAMPLE" \
 --targets "Key=InstanceIds,Values=i-02573cafefEXAMPLE,i-0471e04240EXAMPLE" \
 --name "My-Maintenance-Window-Target" \
 --replace
```

Windows

```
aws ssm update-maintenance-window-target ^
 --window-id "mw-0c50858d01EXAMPLE" ^
 --window-target-id "d208dedf-3f6b-41ff-ace8-8e751EXAMPLE" ^
 --targets "Key=InstanceIds,Values=i-02573cafefEXAMPLE,i-0471e04240EXAMPLE" ^
 --name "My-Maintenance-Window-Target" ^
 --replace
```

The system returns information like the following.

```
{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE",
 "Targets": [
 {
 "Key": "InstanceIds",
 "Values": [
 "i-02573cafefEXAMPLE",
 "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 & macOS

```
aws ssm update-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--start-date "2020-10-01T10:10:10Z" \
--end-date "2020-11-01T10:10:10Z"
```

Windows

```
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 Amazon Elastic Compute Cloud (Amazon EC2) instance for Windows Server, change `df` to `ipconfig`, and `AWS-RunShellScript` to `AWS-RunPowerShellScript` in the following command.

Linux & macOS

```
aws ssm update-maintenance-window-task \
--window-id "mw-0c50858d01EXAMPLE" \
--window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" \
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" \
--task-arn "AWS-RunShellScript" \
--service-role-arn "arn:aws:iam::account-id: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

```
aws ssm update-maintenance-window-task ^
--window-id "mw-0c50858d01EXAMPLE" ^
--window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" ^
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" ^
--task-arn "AWS-RunShellScript" ^
--service-role-arn "arn:aws:iam::account-id: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.

```
{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
 "Priority": 1,
 "MaxConcurrency": 10,
 "MaxErrors": 4,
 "Name": "My-Task-Name",
 "Description": "A description for my Run Command task",
 "ServiceRoleArn": "arn:aws:iam::account-id:role/MaintenanceWindowsRole",
 "TaskArn": "AWS-RunShellScript",
 "TaskInvocationParameters": "RunCommand={Comment=Revising my Run Command task,Parameters={commands=df}}",
 "Targets": "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE",
 "LastModified": "2020-10-01T10:10:10Z",
 "LastUpdated": "2020-10-01T10:10:10Z",
 "Status": "PENDING",
 "StatusReason": null}
```

```

"Targets": [
 {
 "Key": "WindowTargetIds",
 "Values": [
 "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
]
 }
],
"TaskArn": "AWS-RunShellScript",
"ServiceRoleArn": "arn:aws:iam::111122223333:role/MaintenanceWindowsRole",
"TaskParameters": {},
"TaskInvocationParameters": {
 "RunCommand": {
 "Comment": "Revising my Run Command task",
 "Parameters": {
 "commands": [
 "df"
]
 }
 },
 "Priority": 1,
 "MaxConcurrency": "10",
 "MaxErrors": "4",
 "Name": "My-Task-Name",
 "Description": "A description for my Run Command task"
}
}

```

- Adapt and run the following command to update a Lambda task.

Linux & macOS

```

aws ssm update-maintenance-window-task \
--window-id mw-0c50858d01EXAMPLE \
--window-task-id 4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE \
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" \
--task-arn "arn:aws:lambda:region:111122223333:function:SSMTestLambda" \
--service-role-arn "arn:aws:iam:account-id: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

```

aws ssm update-maintenance-window-task ^
--window-id mw-0c50858d01EXAMPLE ^
--window-task-id 4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE ^
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" ^
--task-arn --task-arn
"arn:aws:lambda:region:111122223333:function:SSMTestLambda" ^
--service-role-arn "arn:aws:iam:account-id: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.

```
{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
 "Targets": [
 {
 "Key": "WindowTargetIds",
 "Values": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
 }
],
 "TaskArn": "arn:aws:lambda:us-east-2:111122223333:function:SSMTestLambda",
 "ServiceRoleArn": "arn:aws:iam::111122223333: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"
}
```

- If you're updating a Step Functions task, adapt and run the following command to update its task-invocation-parameters.

#### Linux & macOS

```
aws ssm update-maintenance-window-task \
--window-id "mw-0c50858d01EXAMPLE" \
--window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" \
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" \
--task-arn "arn:aws:states:region:execution:SSMStepFunctionTest" \
--service-role-arn "arn:aws:iam:account-id:role/MaintenanceWindowsRole" \
--task-invocation-parameters '{"StepFunctions":{"Input":{"\"InstanceId\":": \
"\\"{{RESOURCE_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 update-maintenance-window-task ^
--window-id "mw-0c50858d01EXAMPLE" ^
--window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" ^
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" ^
--task-arn "arn:aws:states:region:execution:SSMStepFunctionTest" ^
--service-role-arn "arn:aws:iam:account-id:role/MaintenanceWindowsRole" ^
--task-invocation-parameters '{"StepFunctions":{"Input":{"\"InstanceId\":": \
"\\"{{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.

```
{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
```

```

"Targets": [
 {
 "Key": "WindowTargetIds",
 "Values": [
 "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
]
 }
],
"TaskArn": "arn:aws:states:us-east-2:111122223333:execution:SSMStepFunctionTest",
"ServiceRoleArn": "arn:aws:iam::111122223333:role/MaintenanceWindowsRole",
"TaskParameters": {},
"TaskInvocationParameters": {
 "StepFunctions": {
 "Input": "{\"instanceId\": \"{{RESOURCE_ID}}\"}"
 }
},
"Priority": 0,
"MaxConcurrency": "10",
"MaxErrors": "5",
"Name": "My-Step-Functions-Task",
"Description": "A description for my Step Functions task"
}
}

```

- 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 & macOS

```

aws ssm deregister-target-from-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--window-target-id "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" \
--safe

```

#### Windows

```

aws ssm deregister-target-from-maintenance-window ^
--window-id "mw-0c50858d01EXAMPLE" ^
--window-target-id "e32eecb2-646c-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

```

- 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 & macOS

```

aws ssm deregister-target-from-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--window-target-id "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" \
--no-safe

```

## Windows

```
aws ssm deregister-target-from-maintenance-window ^
--window-id "mw-0c50858d01EXAMPLE" ^
--window-target-id "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" ^
--no-safe
```

The system returns information like the following.

```
{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
}
```

- 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 & macOS

```
aws ssm update-maintenance-window-task \
--window-id "mw-0c50858d01EXAMPLE" \
--window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" \
--targets "Key=InstanceIds,Values=i-02573cafefEXAMPLE" \
--task-invocation-parameters "RunCommand={Comment=A comment for my task
update,Parameters={UpdateLevel='{{ssm:UpdateLevel}}'}}"
```

## Windows

```
aws ssm update-maintenance-window-task ^
--window-id "mw-0c50858d01EXAMPLE" ^
--window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" ^
--targets "Key=InstanceIds,Values=i-02573cafefEXAMPLE" ^
--task-invocation-parameters "RunCommand={Comment=A comment for my task
update,Parameters={UpdateLevel='{{ssm:UpdateLevel}}'}}"
```

The system returns information like the following.

```
{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
 "Targets": [
 {
 "Key": "InstanceIds",
 "Values": [
 "i-02573cafefEXAMPLE"
]
 }
],
 "TaskArn": "AWS-RunShellScript",
 "ServiceRoleArn": "arn:aws:iam::111122223333:role/MyMaintenanceWindowServiceRole",
 "TaskParameters": {},
 "TaskInvocationParameters": {
 "RunCommand": {
 "Comment": "A comment for my task update",
 "Parameters": {
 "UpdateLevel": [

```

```

 "{{ssm:UpdateLevel}}"
]
}
},
"Priority": 10,
"MaxConcurrency": "1",
"MaxErrors": "1"
}

```

- 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 & macOS

```

aws ssm update-maintenance-window-task \
--window-id "mw-0c50858d01EXAMPLE" \
--window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" \
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" \
--task-arn "AutoTestDoc" \
--service-role-arn "arn:aws:iam:account-id:role/MyMaintenanceWindowServiceRole" \
--task-invocation-parameters
"Automation={Parameters={InstanceId='{{RESOURCE_ID}}',initiator='{{WINDOW_ID}}.Task{{WINDOW_TASK_ID}}'}}" \
--priority 3 --max-concurrency 10 --max-errors 5

```

Windows

```

aws ssm update-maintenance-window-task ^
--window-id "mw-0c50858d01EXAMPLE" ^
--window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" ^
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" ^
--task-arn "AutoTestDoc" ^
--service-role-arn "arn:aws:iam:account-id:role/MyMaintenanceWindowServiceRole" ^
--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.

```

{
 "WindowId": "mw-0c50858d01EXAMPLE",
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
 "Targets": [
 {
 "Key": "WindowTargetIds",
 "Values": [
 "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
]
 }
],
 "TaskArn": "AutoTestDoc",
 "ServiceRoleArn": "arn:aws:iam::111122223333:role/MyMaintenanceWindowServiceRole",
 "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.

```
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 (AWS CLI) or the Systems Manager console. The maintenance window that you create updates SSM Agent on managed nodes.

### Contents

- [Walkthrough: Create a maintenance window to update SSM Agent \(AWS CLI\) \(p. 777\)](#)
- [Walkthrough: Create a maintenance window to automatically update SSM Agent \(console\) \(p. 782\)](#)
- [Walkthrough: Creating a maintenance window for patching \(console\) \(p. 787\)](#)

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 Command Line Interface (AWS CLI) to create an AWS Systems Manager maintenance window. The walkthrough also describes how to register your managed nodes as targets and register a Systems Manager Run Command task to update SSM Agent.

### Before you begin

Before you complete the following procedure, you must either have administrator permissions on the nodes 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 Amazon Elastic Compute Cloud (Amazon EC2) instance for Linux or Windows Server that is configured for Systems Manager. For more information, see [Systems Manager prerequisites \(p. 13\)](#).

## Topics

- [Step 1: Get started \(p. 778\)](#)
- [Step 2: Create the maintenance window \(p. 778\)](#)
- [Step 3: Register maintenance window targets \(AWS CLI\) \(p. 779\)](#)
- [Step 4: Register a Run Command task for the maintenance window to update SSM Agent \(p. 781\)](#)

## Step 1: Get started

### To run commands using the AWS CLI

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Verify that a node is ready to be registered as a target for a maintenance window.

Run the following command to view which nodes are online.

```
aws ssm describe-instance-information --query "InstanceInformationList[*]"
```

Run the following command to view details about a particular node.

```
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 & macOS

```
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

```
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 \(p. 1540\)](#).

For an explanation of how the various schedule-related options for maintenance windows relate to one another, see [Maintenance window scheduling and active period options \(p. 790\)](#).

For more information about working with the --schedule option, see [Reference: Cron and rate expressions for Systems Manager \(p. 1540\)](#).

The system returns information like the following.

```
{
 "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.

```
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 nodes to update.

### To register maintenance window targets (AWS CLI)

1. Run the following command.

Linux & macOS

```
aws ssm register-target-with-maintenance-window \
 --window-id "mw-0c50858d01EXAMPLE" \
 --target "Key=InstanceIds,Values=i-02573cafefEXAMPLE" \
 --resource-type "INSTANCE"
```

Windows

```
aws ssm register-target-with-maintenance-window ^
 --window-id "mw-0c50858d01EXAMPLE" ^
 --target "Key=InstanceIds,Values=i-02573cafefEXAMPLE" ^
```

```
--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.

```
{
 "WindowTargetId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

## Alternative commands

Use the following command to register multiple managed nodes.

Linux & macOS

```
aws ssm register-target-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--targets "Key=InstanceIds,Values=i-02573cafefEXAMPLE,i-0471e04240EXAMPLE" \
--resource-type "INSTANCE"
```

## Windows

```
aws ssm register-target-with-maintenance-window ^
--window-id "mw-0c50858d01EXAMPLE" ^
--targets "Key=InstanceIds,Values=i-02573cafefEXAMPLE,i-0471e04240EXAMPLE" ^
--resource-type "INSTANCE"
```

Use the following command to register nodes by using tags.

## Linux & macOS

```
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"
```

## Windows

```
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.

```
aws ssm describe-maintenance-window-targets --window-id "mw-0c50858d01EXAMPLE"
```

The system returns information like the following.

```
{
 "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)

- 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 AWS-UpdateSSMAgent document.

Linux & macOS

```

aws ssm register-task-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--task-arn "AWS-UpdateSSMAgent" \
--name "UpdateSSMAgent" \
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" \
--service-role-arn "arn:aws:iam:account-id:role/MW-Role" \
--task-type "RUN_COMMAND" \
--max-concurrency 1 --max-errors 1 --priority 10

```

Windows

```

aws ssm register-task-with-maintenance-window ^
--window-id "mw-0c50858d01EXAMPLE" ^
--task-arn "AWS-UpdateSSMAgent" ^
--name "UpdateSSMAgent" ^

```

```
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" ^
--service-role-arn "arn:aws:iam:account-id:role/MW-Role" ^
--task-type "RUN_COMMAND" ^
--max-concurrency 1 --max-errors 1 --priority 10
```

**Note**

If the targets you registered in the preceding step are Windows Server 2012 R2 or earlier, you must use the [AWS-UpdateEC2Config](#) document.

The system returns information like the following.

```
{
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
```

- Run the following command to list all registered tasks for a maintenance window.

```
aws ssm describe-maintenance-window-tasks --window-id "mw-0c50858d01EXAMPLE"
```

The system returns information like the following.

```
{
 "Tasks": [
 {
 "ServiceRoleArn": "arn:aws:iam::111122223333:role/MW-Role",
 "MaxErrors": "1",
 "TaskArn": "AWS-UpdateSSMAgent",
 "MaxConcurrency": "1",
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
 "TaskParameters": {},
 "Priority": 10,
 "WindowId": "mw-0c50858d01EXAMPLE",
 "Type": "RUN_COMMAND",
 "Targets": [
 {
 "Values": [
 "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
],
 "Key": "WindowTargetIds"
 }
],
 "Name": "UpdateSSMAgent"
 }
]
}
```

## Walkthrough: Create a maintenance window to automatically update SSM Agent (console)

The following walkthrough shows you how to use the AWS Systems Manager console to create a maintenance window. The walkthrough also describes how to register your managed nodes as targets and register a Systems Manager Run Command task to update SSM Agent.

### Before you begin

Before you complete the following procedure, you must either have administrator permissions on the nodes 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 Amazon Elastic Compute Cloud (Amazon EC2) instance for Linux or Windows Server that is configured for Systems Manager. For more information, see [Systems Manager prerequisites \(p. 13\)](#).

## Topics

- [Step 1: Create the maintenance window \(console\) \(p. 783\)](#)
- [Step 2: Register maintenance window targets \(console\) \(p. 784\)](#)
- [Step 3: Register a Run Command task for the maintenance window to update SSM Agent \(console\) \(p. 785\)](#)

## Step 1: Create the maintenance window (console)

### To create a maintenance window (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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.
5. (Optional) For **Description**, enter a description.
6. Choose **Allow unregistered targets** if you want to allow a maintenance window task to run on managed nodes, even if you haven't registered those nodes as targets. If you choose this option, then you can choose the unregistered nodes (by node 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 \(p. 1540\)](#).

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 **Window 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 **Window 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 **Schedule 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 allow 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 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 nodes 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. (Optional) For **Target name**, enter a name for the target.
4. (Optional) For **Description**, enter a description.
5. (Optional) For **Owner information**, specify your name or work alias. Owner information is included in any Amazon EventBridge event raised while running tasks for these targets in this maintenance window.

For information about using EventBridge to monitor Systems Manager events, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#).

6. In the **Targets** area, choose one of the options described in the following table.

Option	Description
<b>Specify instance tags</b>	For the <b>Specify instance tags</b> boxes, specify one or more tag keys and (optional) values that have been or will be added to managed nodes in your account. When the maintenance window runs, it attempts to perform tasks on all of the managed nodes to which these tags have been added.  If you specify more than one tag key, a node must be tagged with <i>all</i> the tag keys and values you specify to be included in the target group.
<b>Choose nodes manually</b>	From the list, select the box for each node that you want to include in the maintenance window target.  The list includes all nodes in your account that are configured for use with Systems Manager.  If a managed node you expect to see isn't listed, see <a href="#">Troubleshooting managed node availability (p. 816)</a> for troubleshooting tips.  For edge devices and on-premises servers and virtual machines (VMs), see <a href="#">Setting up AWS Systems Manager for hybrid environments (p. 34)</a>

Option	Description
<b>Choose a resource group</b>	<p>For <b>Resource group</b>, choose the name of an existing resource group in your account from the list.</p> <p>For information about creating and working with resource groups, see the following topics:</p> <ul style="list-style-type: none"> <li>• <a href="#">What are resource groups?</a> in the <i>AWS Resource Groups User Guide</i></li> <li>• <a href="#">Resource Groups and Tagging for AWS</a> in the <i>AWS News Blog</i></li> </ul> <p>For <b>Resource types</b>, select up to five available resource types, or choose <b>All resource types</b>.</p> <p>If the tasks you assign to the maintenance window don't act on one of the resource types you added to the target, the system might report an error. Tasks for which a supported resource type is found continue to run despite these errors.</p> <p>For example, suppose you add the following resource types to this target:</p> <ul style="list-style-type: none"> <li>• AWS::S3::Bucket</li> <li>• AWS::DynamoDB::Table</li> <li>• AWS::EC2::Instance</li> </ul> <p>But later, when you add tasks to the maintenance window, you include only tasks that perform actions on nodes, such as applying a patch baseline or rebooting a node. In the maintenance window log, an error might be reported for no Amazon Simple Storage Service (Amazon S3) buckets or Amazon DynamoDB tables being found. However, the maintenance window still runs tasks on the nodes in your resource group.</p>

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. (Optional) For **Name**, enter a name for the task, such as `UpdateSSMAgent`.

4. (Optional) For **Description**, enter a description.
5. In the **Command document** area, 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 **Document version**, 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 nodes on which you want to run this operation by choosing **Selecting registered target groups** or **Selecting unregistered targets**.
9. For **Rate control**:
  - For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes still processing the command might also send errors.
10. For **IAM service role**, choose a role. If you need to create a custom service role, see one of the following topics:
  - [Control access to maintenance windows \(console\) \(p. 709\)](#)
  - [Control access to maintenance windows \(AWS CLI\) \(p. 714\)](#)
  - [Control access to maintenance windows \(Tools for Windows PowerShell\) \(p. 720\)](#)

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? \(p. 708\)](#)

11. (Optional) For **Output options**, do one of the following:

- Select the **Enable writing to S3** check box to save the command output to a file. Enter 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 node, not those of the IAM user performing this task. For more information, see [Create an IAM Instance profile for Systems Manager \(p. 21\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile associated with the node has the necessary permissions to write to that bucket.

- Select the **CloudWatch output** check box to write complete output to Amazon CloudWatch Logs. Enter the name of a CloudWatch Logs log group.
12. In the **SNS notifications** section, you can optionally allow Systems Manager to send notifications about command statuses using Amazon Simple Notification Service (Amazon SNS). If you choose to turn on this option, you need to specify the following:
    - a. The IAM role to start 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 nodes, choose **Invocation** to receive notification on an invocation (per-node) basis when the status of each invocation changes.
13. In the **Parameters** area, 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 updated to the latest version.
  14. Choose **Register Run command task**.

## Walkthrough: Creating a maintenance window for patching (console)

### 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 [Creating a patching configuration \(console\) \(p. 1203\)](#). Although many patching use cases benefit from patching nodes on a schedule with a maintenance window, you can also run a one-time patching operation manually without a maintenance window. For more information, see [Patching managed nodes on demand \(console\) \(p. 1190\)](#).

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 \(p. 706\)](#).

You must configure roles and permissions for Maintenance Windows, a capability of AWS Systems Manager, before beginning this procedure. For more information, see [Setting up Maintenance Windows \(p. 707\)](#).

### To create a maintenance window for patching

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 nodes, even if you haven't registered those nodes as targets. If you choose this option, then you can choose the unregistered nodes (by node 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 \(p. 1540\)](#).

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. Choose **Create maintenance window**.
14. In the maintenance windows list, choose the maintenance window you just created, and then choose **Actions, Register targets**.
15. (Optional) In the **Maintenance window target details** section, provide a name, a description, and owner information (your name or alias) for this target.
16. For **Targets**, choose **Specifying instance tags**.
17. For **Instance tags**, enter a tag key and a tag value to identify the nodes to register with the maintenance window, and then choose **Add**.
18. Choose **Register target**. The system creates a maintenance window target.
19. In the details page of the maintenance window you created, choose **Actions, Register Run command task**.
20. (Optional) For **Maintenance window task details**, provide a name and description for this task.
21. For **Command document**, choose `AWS-RunPatchBaseline`.
22. For **Task priority**, choose a priority. Zero (0) is the highest priority.
23. For **Targets**, under **Target by**, choose the maintenance window target you created earlier in this procedure.
24. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

#### Note

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes still processing the command might also send errors.
25. For **IAM service role**, choose a role. If you need to create a custom service role, see one of the following topics:
    - [Control access to maintenance windows \(console\) \(p. 709\)](#)
    - [Control access to maintenance windows \(AWS CLI\) \(p. 714\)](#)

- [Control access to maintenance windows \(Tools for Windows PowerShell\) \(p. 720\)](#)

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? \(p. 708\)](#).

26. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Enter 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 managed node, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

To stream the output to an Amazon CloudWatch Logs log group, select the **CloudWatch output** box. Enter 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 \(p. 1462\)](#).

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 \(p. 1132\)](#).
- For **Reboot option**, specify whether you want nodes to reboot if patches are installed during the **Install** operation, or if Patch Manager detects other patches that were installed since the last node reboot. For information, see [Parameter name: RebootOption \(p. 1136\)](#).
- (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 is complete, 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 node by choosing the node 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 \(p. 840\)](#).

## About patching schedules using maintenance windows

After you configure a patch baseline (and optionally a patch group), you can apply patches to your node 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 Elastic Compute Cloud (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 nodes or scan and install patches on the nodes. If you choose to scan nodes, Patch Manager, a capability of AWS Systems Manager, scans each node and generates a list of missing patches for you to review.

If you choose to scan and install patches, Patch Manager scans each node 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 node. The reboot is required to make sure that patches are installed correctly and to ensure that the system didn't leave the node in a potentially bad state. (Exception: If the `RebootOption` parameter is set to `NoReboot` in the `AWS-RunPatchBaseline` document, the managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).)

## 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 \(p. 1540\)](#). Optionally, you can specify a date range during which the maintenance window can run on its regular schedule and a time zone on which to base that regular schedule.

Be aware, however, that the time zone option and the start date and end date options don't 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:

2021-04-07T14:29:00-08:00

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 \(p. 791\)](#)
- [Example 2: Specify a maintenance window start date and end date \(p. 791\)](#)

- [Example 3: Create a maintenance window that runs only once \(p. 792\)](#)
- [Example 4: Specify the number of schedule offset days for a maintenance window \(p. 793\)](#)

## Example 1: Specify a maintenance window start date

Say that you use the AWS Command Line Interface (AWS CLI) to create a maintenance window with the following options:

- `--start-date 2021-01-01T00:00:00-08:00`
- `--schedule-timezone "America/Los_Angeles"`
- `--schedule "cron(0 09 ? * WED *)"`

For example:

Linux & macOS

```
aws ssm create-maintenance-window \
--name "My-LAX-Maintenance-Window" \
--allow-unassociated-targets \
--duration 3 \
--cutoff 1 \
--start-date 2021-01-01T00:00:00-08:00 \
--schedule-timezone "America/Los_Angeles" \
--schedule "cron(0 09 ? * WED *)"
```

Windows

```
aws ssm create-maintenance-window ^
--name "My-LAX-Maintenance-Window" ^
--allow-unassociated-targets ^
--duration 3 ^
--cutoff 1 ^
--start-date 2021-01-01T00:00:00-08:00 ^
--schedule-timezone "America/Los_Angeles" ^
--schedule "cron(0 09 ? * WED *)"
```

This means that the first run of the maintenance window won't occur until *after* its specified start date and time, which is at 12:00 AM US Pacific Time on Friday, January 1, 2021. (This time zone is eight hours behind UTC time.) In this case, the start date and time of the window period don't represent when the maintenance windows first runs. Taken together, the `--schedule-timezone` and `--schedule` values mean that the maintenance window runs at 9 AM every Wednesday in the US Pacific Time Zone (represented by "America/Los Angeles" in IANA format). The first execution in the allowed period will be on Wednesday, January 4, 2021, 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:

Linux & macOS

```
aws ssm create-maintenance-window \
--name "My-NRT-Maintenance-Window" \
--allow-unassociated-targets \
--duration 3 \
--cutoff 1 \
--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)"
```

Windows

```
aws ssm create-maintenance-window ^
--name "My-NRT-Maintenance-Window" ^
--allow-unassociated-targets ^
--duration 3 ^
--cutoff 1 ^
--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 allowed 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 1. The last execution is at 3:15 AM Japan Standard Time on Tuesday, June 25, 2019. This is the last Tuesday before the allowed 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:

Linux & macOS

```
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
```

Windows

```
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 allowed 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.

## Example 4: Specify the number of schedule offset days for a maintenance window

Now you create a maintenance window with this option:

```
--schedule-offset 2
```

For example:

Linux & macOS

```
aws ssm create-maintenance-window \
--name "My-Cron-Offset-Maintenance-Window" \
--schedule "cron(0 30 23 ? * TUE#3 *)" \
--duration 4 \
--cutoff 1 \
--schedule-offset 2 \
--allow-unassociated-targets
```

Windows

```
aws ssm create-maintenance-window ^
--name "My-Cron-Offset-Maintenance-Window" ^
--schedule "cron(0 30 23 ? * TUE#3 *)" ^
--duration 4 ^
--cutoff 1 ^
--schedule-offset 2 ^
--allow-unassociated-targets
```

A schedule offset is the number of days to wait after the date and time specified by a CRON expression before running the maintenance window.

In the preceding example, the CRON expression schedules a maintenance window to run the third Tuesday of every month at 11:30 PM:

```
--schedule "cron(0 30 23 ? * TUE#3 *)"
```

However, including `--schedule-offset 2` means that the maintenance window won't run until 11:30 PM two days *after* the third Tuesday of each month.

Schedule offsets are supported for CRON expressions only.

### Related content

- [Reference: Cron and rate expressions for Systems Manager \(p. 1540\)](#)
- [Create a maintenance window \(console\) \(p. 724\)](#)
- [Tutorial: Create and configure a maintenance window \(AWS CLI\) \(p. 733\)](#)

- [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

## Registering maintenance window tasks without targets

For each maintenance window you create, you can specify one or more tasks to perform when the maintenance window runs. In most cases, you must specify the resources, or targets, that the task is to run on. In some cases, however, you don't need to specify targets explicitly in the task.

One or more targets must be specified for maintenance window Systems Manager Run Command-type tasks. Depending on the nature of the task, targets are optional for other maintenance window task types (Systems Manager Automation, AWS Lambda, and AWS Step Functions).

For Lambda and Step Functions task types, whether a target is required depends on the content of the function or state machine you have created.

In many cases, you don't need to explicitly specify a target for an automation task. For example, say that you're creating an Automation-type task to update an Amazon Machine Image (AMI) for Linux using the `AWS-UpdateLinuxAmi` runbook. When the task runs, the AMI is updated with the latest available Linux distribution packages and Amazon software. New instances created from the AMI already have these updates installed. Because the ID of the AMI to be updated is specified in the input parameters for the runbook, there is no need to specify a target again in the maintenance window task.

Similarly, suppose you're using the AWS Command Line Interface (AWS CLI) to register a maintenance window Automation task that uses the document [AWS-RestartEC2Instance](#). Because the node to restart is specified in the `--task-invocation-parameters` argument, you don't need to also specify a `--targets` option.

### Note

For maintenance window tasks without a target specified, you can't supply values for `--max-errors` and `--max-concurrency`. Instead, the system inserts a placeholder value of 1, which might be reported in the response to commands such as [describe-maintenance-window-tasks](#) and [get-maintenance-window-task](#). These values don't affect the running of your task and can be ignored.

The following example demonstrates omitting the `--targets`, `--max-errors`, and `--max-concurrency` options for a targetless maintenance window task.

### Linux & macOS

```
aws ssm register-task-with-maintenance-window \
--window-id "mw-ab12cd34eEXAMPLE" \
--service-role-arn "arn:aws:iam::123456789012:role/
MaintenanceWindowAndAutomationRole" \
--task-type "AUTOMATION" \
--name "RestartInstanceWithoutTarget" \
--task-arn "AWS-RestartEC2Instance" \
--task-invocation-parameters "{\"Automation\":{\"Parameters\":{\"InstanceId\":[\"i-02573cafceEXAMPLE\"]}}}" \
--priority 10
```

### Windows

```
aws ssm register-task-with-maintenance-window ^
```

```
--window-id "mw-ab12cd34eEXAMPLE" ^
--service-role-arn "arn:aws:iam::123456789012:role/
MaintenanceWindowAndAutomationRole" ^
--task-type "AUTOMATION" ^
--name "RestartInstanceWithoutTarget" ^
--task-arn "AWS-RestartEC2Instance" ^
--task-invocation-parameters "{\"Automation\":{\"Parameters\":{\"InstanceId\":[\"i-02573cafefEXAMPLE\"]}}}" ^
--priority 10
```

### Note

For maintenance window tasks registered before December 23, 2020: If you specified targets for the task and one is no longer required, you can update that task to remove the targets using the Systems Manager console or the [update-maintenance-window-task](#) AWS CLI command.

### Related content

Error messages: "Maintenance window tasks without targets don't support MaxConcurrency values" and "Maintenance window tasks without targets don't support MaxErrors values" (p. 797)

## Troubleshooting maintenance windows

Use the following information to help you troubleshoot problems with maintenance windows.

### Topics

- [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."](#) (p. 795)
- [Not all maintenance window targets are updated](#) (p. 796)
- [Task fails with error message: "Step fails when it is validating and resolving the step inputs"](#) (p. 796)
- [Error messages: "Maintenance window tasks without targets don't support MaxConcurrency values" and "Maintenance window tasks without targets don't support MaxErrors values"](#) (p. 797)

**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 AWS Identity and Access Management (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 an AWS 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. 708).

**Problem 2:** If the task was created using the AWS Command Line Interface (AWS CLI), AWS 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? \(p. 708\)](#).

## Not all maintenance window targets are updated

**Problem:** You notice that maintenance window tasks didn't run on all the resources targeted by your maintenance window. For example, in the maintenance window run results, the task for that resource is marked as failed or timed out.

**Solution:** The most common reasons for a maintenance window task not running on a target resource involve connectivity and availability. For example:

- Systems Manager lost connection to the resource before or during the maintenance window operation.
- The resource was offline or stopped during the maintenance window operation.

You can wait for the next scheduled maintenance window time to run tasks on the resources. You can manually run the maintenance window tasks on the resources that weren't available or were offline.

## Task fails with error message: "Step fails when it is validating and resolving the step inputs"

**Problem:** An Automation runbook or Systems Manager Command document you're using in a task requires that you specify inputs such as `InstanceId` or `SnapshotId`, but a value isn't supplied or isn't supplied correctly.

- **Solution 1:** If your task is targeting a single resource, such as a single node or single snapshot, enter its ID in the input parameters for the task.
- **Solution 2:** If your task is targeting multiple resources, such as creating images from multiple nodes when you use the runbook `AWS-CreateImage`, you can use one of the pseudo parameters supported for maintenance window tasks in the input parameters to represent node IDs in the command.

The following commands register a Systems Manager Automation task with a maintenance window using the AWS CLI. The `--targets` value indicates a maintenance window target ID. Also, even though the `--targets` parameter specifies a window target ID, parameters of the Automation runbook require that a node ID be provided. In this case, the command uses the pseudo parameter `{{RESOURCE_ID}}` as the `InstanceId` value.

### AWS CLI command:

Linux & macOS

The following example command restarts Amazon Elastic Compute Cloud (Amazon EC2) instances that belong to the maintenance window target group with the ID `e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE`.

```
aws ssm register-task-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--targets Key=WindowTargetIds,Values=e32eecb2-646c-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

```
aws ssm register-task-with-maintenance-window ^
--window-id "mw-0c50858d01EXAMPLE" ^
--targets Key=WindowTargetIds,Values=e32eecb2-646c-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"
```

For more information about working with pseudo parameters for maintenance window tasks, see [About pseudo parameters \(p. 751\)](#) and [Task registration examples \(p. 744\)](#).

## Error messages: "Maintenance window tasks without targets don't support MaxConcurrency values" and "Maintenance window tasks without targets don't support MaxErrors values"

**Problem:** When you register a Run Command-type task, you must specify at least one target for the task to run on. For other task types (Automation, AWS Lambda, and AWS Step Functions), depending on the nature of the task, targets are optional. The options `MaxConcurrency` (the number of resources to run a task on at the same time) and `MaxErrors` (the number of failures to run the task on target resources before the task fails) aren't required or supported for maintenance window tasks that don't specify targets. The system generates these error messages if values are specified for either of these options when no task target is specified.

**Solution:** If you receive either of these errors, remove the values for concurrency and error threshold before continuing to register or update the maintenance window task.

For more information about running tasks that don't specify targets, see [Registering maintenance window tasks without targets \(p. 794\)](#) in the *AWS Systems Manager User Guide*.

# AWS Systems Manager Node Management

AWS Systems Manager provides the following capabilities for accessing, managing, and configuring your *managed nodes*. A managed node is any machine configured for Systems Manager. You can configure Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and on-premises servers and virtual machines (VMs) in a hybrid environment as managed nodes.

## Topics

- [AWS Systems Manager Fleet Manager \(p. 798\)](#)
- [AWS Systems Manager Compliance \(p. 836\)](#)
- [AWS Systems Manager Inventory \(p. 848\)](#)
- [AWS Systems Manager Hybrid Activations \(p. 912\)](#)
- [AWS Systems Manager Session Manager \(p. 912\)](#)
- [AWS Systems Manager Run Command \(p. 991\)](#)
- [AWS Systems Manager State Manager \(p. 1034\)](#)
- [AWS Systems Manager Patch Manager \(p. 1093\)](#)
- [AWS Systems Manager Distributor \(p. 1252\)](#)

## AWS Systems Manager Fleet Manager

Fleet Manager, a capability of AWS Systems Manager, is a unified user interface (UI) experience that helps you remotely manage your nodes running on AWS or on premises. With Fleet Manager, you can view the health and performance status of your entire server fleet from one console. You can also gather data from individual nodes to perform common troubleshooting and management tasks from the console. This includes connecting to Windows instances using the Remote Desktop Protocol (RDP), viewing folder and file contents, Windows registry management, operating system user management, and more. To get started with Fleet Manager, open the [Systems Manager console](#). In the navigation pane, choose **Fleet Manager**.

## Who should use Fleet Manager?

Any AWS customer should use Fleet Manager who wants to have a centralized way to do the following:

- Manage their node fleet
- Manage their Amazon ECS clusters

## How can Fleet Manager benefit my organization?

Fleet Manager offers these benefits:

- Perform a variety of common systems administration tasks without having to manually connect to your managed nodes.
- Manage nodes running on multiple platforms from a single unified console.

- Manage nodes running different operating systems from a single unified console.
- Manage ECS containers from a single unified console.
- Improve the efficiency of your systems administration.

## What are the features of Fleet Manager?

Key features of Fleet Manager include the following:

- **Access the Red Hat Knowledgebase Portal**

Access binaries, knowledge-shares, and discussion forums on the Red Hat Knowledgebase Portal through your Red Hat Enterprise Linux (RHEL) instances.

- **Managed node status**

View which managed instances are `running` and which are `stopped`. For more information about stopped instances, see [Stop and start your instance](#) in the *Amazon EC2 User Guide for Linux Instances*. For AWS IoT Greengrass core devices, you can view which are `online`, `offline`, or show a status of `Connection lost`.

**Note**

If you stopped your managed instance before July 12, 2021, it won't display the `stopped` marker. To show the marker, start and stop the instance.

- **View instance information**

View information about the folder and file data stored on the volumes attached to your managed instances, performance data about your instances in real-time, and log data stored on your instances.

- **View edge device information**

View the AWS IoT Greengrass Thing name for the device, the SSM Agent ping status and version, and more.

- **Manage accounts and registry**

Manage operating system (OS) user accounts on your instances and registry on your Windows instances.

- **Control access to features**

Control access to Fleet Manager features using AWS Identity and Access Management (IAM) policies. With these policies, you can control which individual users or groups in your organization can use various Fleet Manager features, and which managed nodes they can manage.

### Topics

- [Getting started with Fleet Manager \(p. 799\)](#)
- [Working with Fleet Manager \(p. 803\)](#)

## Getting started with Fleet Manager

Before you can use Fleet Manager, a capability of AWS Systems Manager, to monitor and manage your managed nodes, complete the steps in the following topics.

### Topics

- [Step 1: Create an IAM policy with Fleet Manager permissions \(p. 800\)](#)
- [Step 2: Verify your instances and edge devices can be managed by Systems Manager \(p. 803\)](#)

## Step 1: Create an IAM policy with Fleet Manager permissions

To use Fleet Manager, a capability of AWS Systems Manager, your AWS Identity and Access Management (IAM) user or role must have the required permissions. You can create an IAM policy that provides access to all Fleet Manager features, or modify your policy to grant access to the features you choose. The following policy samples provide the required permissions for all Fleet Manager features and the permissions needed for subsets of features.

### Note

In all the following policies, replace *key-name* with the name of the AWS Key Management Service (AWS KMS) key you want to use to encrypt Session Manager session data. Replace *region* and *account-id* with your AWS Region and AWS account ID, such as `us-east-2` and `111122223333`.

### Administrator policy

The following policy provides permissions to all Fleet Manager features. This means a user can create and delete local users and groups, modify group membership for any local group, and modify Windows registry keys or values.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "EC2",
 "Effect": "Allow",
 "Action": [
 "ec2:CreateTags",
 "ec2:DeleteTags",
 "ec2:DescribeInstances",
 "ec2:DescribeTags"
],
 "Resource": "*"
 },
 {
 "Sid": "General",
 "Effect": "Allow",
 "Action": [
 "ssm:AddTagsToResource",
 "ssm:DescribeInstanceAssociationsStatus",
 "ssm:DescribeInstancePatches",
 "ssm:DescribeInstancePatchStates",
 "ssm:DescribeInstanceProperties",
 "ssm:GetCommandInvocation",
 "ssm:GetServiceSetting",
 "ssm:GetInventorySchema",
 "ssm>ListComplianceItems",
 "ssm>ListInventoryEntries",
 "ssm>ListTagsForResource",
 "ssm>ListCommandInvocations",
 "ssm>ListAssociations",
 "ssm:RemoveTagsFromResource"
],
 "Resource": "*"
 },
 {
 "Sid": "SendCommand",
 "Effect": "Allow",
 "Action": [
 "ssm:GetDocument",
 "ssm:SendCommand",
 "ssm:StartSession"
],
 "Resource": "*"
 }
]
}
```

```

"Resource": [
 "arn:aws:ec2:*:account-id:instance/*",
 "arn:aws:ssm:*:account-id:managed-instance/*",
 "arn:aws:ssm:*:account-id:document/SSM-SessionManagerRunShell",
 "arn:aws:ssm:*:*:document/AWS-PasswordReset",
 "arn:aws:ssm:*:*:document/AWSFleetManager-AddUsersToGroups",
 "arn:aws:ssm:*:*:document/AWSFleetManager-CreateGroup",
 "arn:aws:ssm:*:*:document/AWSFleetManager-CreateUser",
 "arn:aws:ssm:*:*:document/AWSFleetManager-CreateUserInteractive",
 "arn:aws:ssm:*:*:document/AWSFleetManager-CreateWindowsRegistryKey",
 "arn:aws:ssm:*:*:document/AWSFleetManager-DeleteGroup",
 "arn:aws:ssm:*:*:document/AWSFleetManager-DeleteUser",
 "arn:aws:ssm:*:*:document/AWSFleetManager-DeleteWindowsRegistryKey",
 "arn:aws:ssm:*:*:document/AWSFleetManager-DeleteWindowsRegistryValue",
 "arn:aws:ssm:*:*:document/AWSFleetManager-GetFileContent",
 "arn:aws:ssm:*:*:document/AWSFleetManager-GetFileSystemContent",
 "arn:aws:ssm:*:*:document/AWSFleetManager-GetGroups",
 "arn:aws:ssm:*:*:document/AWSFleetManager-GetPerformanceCounters",
 "arn:aws:ssm:*:*:document/AWSFleetManager-GetUsers",
 "arn:aws:ssm:*:*:document/AWSFleetManager-GetWindowsEvents",
 "arn:aws:ssm:*:*:document/AWSFleetManager-GetWindowsRegistryContent",
 "arn:aws:ssm:*:*:document/AWSFleetManager-RemoveUsersFromGroups",
 "arn:aws:ssm:*:*:document/AWSFleetManager-SetWindowsRegistryValue"
],
"Condition": {
 "BoolIfExists": {
 "ssm:SessionDocumentAccessCheck": "true"
 }
}
},
{
 "Sid": "TerminateSession",
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession"
],
 "Resource": "*",
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/aws:ssmmessages:session-id": [
 "${aws:userid}"
]
 }
 }
},
{
 "Sid": "KMS",
 "Effect": "Allow",
 "Action": [
 "kms:GenerateDataKey"
],
 "Resource": [
 "arn:aws:kms:region:account-id:key/key-name"
]
}
]
}

```

### Read-only features policy

The following policy provides permissions to read-only Fleet Manager features.

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

```

```
{
 "Sid": "EC2",
 "Effect": "Allow",
 "Action": [
 "ec2:DescribeInstances",
 "ec2:DescribeTags"
],
 "Resource": "*"
},
{
 "Sid": "General",
 "Effect": "Allow",
 "Action": [
 "ssm:DescribeInstanceAssociationsStatus",
 "ssm:DescribeInstancePatches",
 "ssm:DescribeInstancePatchStates",
 "ssm:DescribeInstanceProperties",
 "ssm:GetCommandInvocation",
 "ssm:GetServiceSetting",
 "ssm:GetInventorySchema",
 "ssm>ListComplianceItems",
 "ssm>ListInventoryEntries",
 "ssm>ListTagsForResource",
 "ssm>ListCommandInvocations",
 "ssm>ListAssociations"
],
 "Resource": "*"
},
{
 "Sid": "SendCommand",
 "Effect": "Allow",
 "Action": [
 "ssm:GetDocument",
 "ssm:SendCommand",
 "ssm:StartSession"
],
 "Resource": [
 "arn:aws:ec2:::account-id:instance/*",
 "arn:aws:ssm:::account-id:managed-instance/*",
 "arn:aws:ssm:::account-id:document/SSM-SessionManagerRunShell",
 "arn:aws:ssm::*:document/AWSFleetManager-GetFileContent",
 "arn:aws:ssm::*:document/AWSFleetManager-GetFileSystemContent",
 "arn:aws:ssm::*:document/AWSFleetManager-GetGroups",
 "arn:aws:ssm::*:document/AWSFleetManager-GetPerformanceCounters",
 "arn:aws:ssm::*:document/AWSFleetManager-GetUsers",
 "arn:aws:ssm::*:document/AWSFleetManager-GetWindowsEvents",
 "arn:aws:ssm::*:document/AWSFleetManager-GetWindowsRegistryContent"
],
 "Condition": {
 "BoolIfExists": {
 "ssm:SessionDocumentAccessCheck": "true"
 }
 }
},
{
 "Sid": "TerminateSession",
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession"
],
 "Resource": "*",
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/aws:ssmmessages:session-id": [
 "${aws:userid}"
]
 }
 }
}
```

```
 }
 },
{
 "Sid": "KMS",
 "Effect": "Allow",
 "Action": [
 "kms:GenerateDataKey"
],
 "Resource": [
 "arn:aws:kms:region:account-id:key/key-name"
]
}
}
```

## Step 2: Verify your instances and edge devices can be managed by Systems Manager

For Amazon Elastic Compute Cloud (Amazon EC2) instances; AWS IoT Greengrass core devices; and on-premises servers, edge devices, and virtual machines (VMs) to be monitored and managed using Fleet Manager, a capability of AWS Systems Manager, they must be Systems Manager *managed nodes*. This means your nodes must meet certain prerequisites and be configured with the AWS Systems Manager Agent (SSM Agent). For more information, see [Setting up AWS Systems Manager \(p. 16\)](#).

You can use Quick Setup, a capability of AWS Systems Manager, to help you quickly configure your Amazon EC2 instances as managed instances in an individual account. If your business or organization uses AWS Organizations, you can also configure instances across multiple organizational units (OUs) and AWS Regions . For more information about using Quick Setup to configure managed instances, see [Quick Setup Host Management \(p. 150\)](#).

### Note

For servers or virtual machines that aren't running on AWS, use a hybrid activation to configure the server, edge device, or VM as a managed instance. For information about hybrid activations, see [AWS Systems Manager Hybrid Activations \(p. 912\)](#).

## Working with Fleet Manager

You can use Fleet Manager, a capability of AWS Systems Manager, to perform various tasks on your managed nodes from the AWS Systems Manager console. The following topics describe the features provided by Fleet Manager.

### Note

The only supported feature for macOS instances is viewing the file system.

### Topics

- [Managed nodes \(p. 804\)](#)
- [Connect using Remote Desktop \(p. 825\)](#)
- [Working with the file system \(p. 826\)](#)
- [Monitoring managed node performance \(p. 829\)](#)
- [Working with processes \(p. 829\)](#)
- [View logs \(p. 830\)](#)
- [User management \(p. 831\)](#)
- [Windows registry management \(p. 834\)](#)
- [Accessing the Red Hat Knowledgebase portal \(p. 836\)](#)

## Managed nodes

A *managed node* is any machine configured for AWS Systems Manager. You can configure Amazon Elastic Compute Cloud (Amazon EC2) instances; AWS IoT Greengrass core devices; and on-premises servers, edge devices, and virtual machines (VMs) in a hybrid environment as managed nodes.

### Note

In the Systems Manager console, any machine prefixed with "mi-" is an on-premises server, edge device, or VM configured as a managed node. Edge devices display their AWS IoT Thing name.

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for servers, edge devices, and VMs in your hybrid environment. The standard-instances tier allows you to register a maximum of 1,000 machines per AWS account per AWS Region. If you need to register more than 1,000 machines in a single account and Region, then use the advanced-instances tier. You can create as many managed nodes as you like in the advanced-instances tier. All managed nodes configured for Systems Manager are priced on a pay-per-use basis. For more information about enabling the advanced instances tier, see [Turning on the advanced-instances tier \(p. 805\)](#). For more information about pricing, see [AWS Systems Manager Pricing](#).

### Note

- Advanced instances also allow 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 \(p. 912\)](#).
- The standard-instances quota also applies to EC2 instances that use a Systems Manager on-premises activation (which isn't a common scenario).
- To patch applications released by Microsoft on virtual machines (VMs) on-premises instances, activate the advanced-instances tier. There is a charge to use the advanced-instances tier. There is no additional charge to patch applications released by Microsoft on Amazon Elastic Compute Cloud (Amazon EC2) instances. For more information, see [About patching applications released by Microsoft on Windows Server \(p. 1166\)](#).

## Display managed nodes

If you don't see your managed nodes listed in the console, then do the following:

1. Verify that the console is open in the AWS Region where you created your managed nodes. You can switch Regions by using the list in the top, right corner of the console.
2. Verify that your managed nodes meet Systems Manager requirements. For information, see [Systems Manager prerequisites \(p. 13\)](#).
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 \(p. 34\)](#).

### Note

Note the following information.

- Systems Manager requires accurate time references in order to perform operations on your machines. If the date and time aren't set correctly on your managed nodes, the machines might not match the signature date of your API requests. For more information, see [Use cases and best practices \(p. 1550\)](#).
- When you create or edit tags, the system can take up to one hour to display changes in the table filter.

## Verify Systems Manager support on a managed node

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 Amazon EC2 instances in your account are managed by Systems Manager. For more information, see [AWS Config Managed Rules](#).

### Increase security posture on managed nodes

For information about increasing your security posture against unauthorized root-level commands on your managed nodes, see [Restricting access to root-level commands through SSM Agent \(p. 134\)](#).

### Deregister managed nodes

You can deregister managed nodes at any time. For example, if you're managing multiple nodes with the same AWS Identity and Access Management (IAM) role and you notice any kind of malicious behavior, you can deregister any number of machines at any point. For information about deregistering managed nodes, see [Deregistering managed nodes in a hybrid environment \(p. 815\)](#).

#### Topics

- [Configuring instance tiers \(p. 805\)](#)
- [Resetting passwords on managed nodes \(p. 812\)](#)
- [Deregistering managed nodes in a hybrid environment \(p. 815\)](#)
- [Troubleshooting managed node availability \(p. 816\)](#)

## Configuring instance tiers

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for servers, edge devices, and VMs in a hybrid environment. The standard-instances tier lets you register a maximum of 1,000 on-premises machines per AWS account per AWS Region. If you need to register more than 1,000 on-premises machines in a single account and Region, then use the advanced-instances tier. You can activate as many managed nodes in a hybrid environment as you like in the advanced-instances tier. However, all Systems Manager managed nodes that use the managed-instance activation process described in [Create a managed-instance activation for a hybrid environment \(p. 41\)](#) are made available on a pay-per-use basis. This also applies to Amazon Elastic Compute Cloud (Amazon EC2) instances that use a Systems Manager on-premises activation (which isn't 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. You can have up to 1,000 standard instances per account per AWS Region at no additional cost. If you run advanced instances, you are charged based on the number of advanced instances activated as Systems Manager managed instances and the hours those instances are running. Review pricing details for advanced instances. Advanced instances are an account-level feature. For more information, see [AWS Systems Manager Pricing](#).

#### Topics

- [Turning on the advanced-instances tier \(p. 805\)](#)
- [Reverting from the advanced-instances tier to the standard-instances tier \(p. 810\)](#)

### Turning on the advanced-instances tier

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for servers, edge devices, and VMs in a hybrid environment. The standard-instances tier lets you register a maximum of 1,000 on-premises machines per AWS account per AWS Region. If you need to register more than 1,000 on-premises machines in a single account and Region, then use the advanced-instances tier. You can activate as many managed nodes in a hybrid environment as you like in the advanced-instances tier. However, all Systems Manager managed nodes that use the managed-instance activation process

described in [Create a managed-instance activation for a hybrid environment \(p. 41\)](#) are made available on a pay-per-use basis. This also applies to Amazon Elastic Compute Cloud (Amazon EC2) instances that use a Systems Manager on-premises activation (which isn't a common scenario).

#### Note

- Advanced instances can use Session Manager to connect to your hybrid machines. Session Manager provides interactive shell access to your instances. For more information, see [AWS Systems Manager Session Manager \(p. 912\)](#).
- The standard-instances limit also applies to Amazon EC2 instances that use a Systems Manager on-premises activation (which isn't a common scenario).
- To patch applications released by Microsoft on virtual machines (VMs) and on-premises instances, turn on the advanced-instances tier. There is a charge to use the advanced-instances tier. There is no additional charge to patch applications released by Microsoft on Amazon Elastic Compute Cloud (Amazon EC2) instances. For more information, see [About patching applications released by Microsoft on Windows Server \(p. 1166\)](#).

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 turn on 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 operations:

- [GetServiceSetting](#)
- [UpdateServiceSetting](#)
- [ResetServiceSetting](#)

Use the following procedure to add an inline IAM policy to a user account. This policy allows a user to view the current managed-instance tier setting. This policy also allows the user to change or reset the current setting in the specified AWS account and AWS 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:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:DescribeActivationTier",
 "ssm:UpdateActivationTier"
]
 }
]
}
```

```
 "ssm:GetServiceSetting"

],
 "Resource": "*"
},
{
 "Effect": "Allow",
 "Action": [
 "ssm:ResetServiceSetting",
 "ssm:UpdateServiceSetting"
],
 "Resource": "arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier"
}
]
```

8. Choose **Review policy**.
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.

```
{
 "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*.

#### Turning on the advanced-instances tier (console)

The following procedure shows you how to use the Systems Manager console to change *all* on-premises servers, edge devices, and virtual machines (VMs) that were added using managed-instance activation, in the specified AWS account and AWS 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 turn on the advanced-instances tier (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.

2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. In the **Account management** menu, choose **Instance tier settings**.

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 \(p. 13\)](#).
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 \(p. 34\)](#).
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 \(p. 810\)](#).

### Turning on the advanced-instances tier (AWS CLI)

The following procedure shows you how to use the AWS Command Line Interface to change *all* on-premises servers and VMs that were added using managed-instance activation, in the specified AWS account and AWS 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 turn on the advanced-instances tier using the AWS CLI

1. Open the AWS CLI and run the following command.

Linux & macOS

```
aws ssm update-service-setting \
--setting-id arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier \
--setting-value advanced
```

Windows

```
aws ssm update-service-setting ^
--setting-id arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier ^
--setting-value advanced
```

There is no output if the command succeeds.

- Run the following command to view the current service settings for managed nodes in the current AWS account and AWS Region.

Linux & macOS

```
aws ssm get-service-setting \
--setting-id arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier
```

Windows

```
aws ssm get-service-setting ^
--setting-id arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier
```

The command returns information like the following.

```
{
 "ServiceSetting": {
 "SettingId": "/ssm/managed-instance/activation-tier",
 "SettingValue": "advanced",
 "LastModifiedDate": 1555603376.138,
 "LastModifiedUser": "arn:aws:sts::123456789012:assumed-role/
Administrator/User_1",
 "ARN": "arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/managed-instance/
activation-tier",
 "Status": "PendingUpdate"
 }
}
```

### Turning on 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 AWS 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 turn on the advanced-instances tier using PowerShell

- Open AWS Tools for Windows PowerShell and run the following command.

```
Update-SSMServiceSetting `
 -SettingId "arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-instance/
activation-tier" `
 -SettingValue "advanced"
```

There is no output if the command succeeds.

- Run the following command to view the current service settings for managed nodes in the current AWS account and AWS Region.

```
Get-SSMServiceSetting `
 -SettingId "arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-instance/
activation-tier"
```

The command returns information like the following.

```
ARN:arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/managed-instance/activation-tier
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 nodes 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 \(p. 810\)](#).

### [Reverting from the advanced-instances tier to the standard-instances tier](#)

This section describes how to change hybrid machines running in the advanced-instances tier back to the standard-instances tier. This configuration applies to all hybrid machines 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're 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 Amazon Elastic Compute Cloud (Amazon EC2) instances that use a Systems Manager on-premises activation (which isn't a common scenario). For more information, see [Deregistering managed nodes in a hybrid environment \(p. 815\)](#).
- After you revert, you won't be able to use Session Manager, a capability of AWS Systems Manager, to interactively access your hybrid instances.
- After you revert, you won't be able to use Patch Manager, a capability of AWS Systems Manager, to patch applications released by Microsoft on hybrid servers and virtual machines (VMs).
- The process of reverting all hybrid machines back to the standard-instance tier can take 30 minutes or more to complete.

This section describes how to revert all hybrid machines in an AWS account and AWS 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, edge devices, and virtual machines (VMs) in your hybrid environment to use the standard-instances tier in the specified AWS account and AWS Region.

#### **To revert to the standard-instances tier (console)**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

- If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Select the **Account settings** dropdown and choose **Instance tier settings**.
  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 Command Line Interface to change all on-premises servers, edge devices, and VMs in your hybrid environment to use the standard-instances tier in the specified AWS account and AWS Region.

#### To revert to the standard-instances tier using the AWS CLI

1. Open the AWS CLI and run the following command.

Linux & macOS

```
aws ssm update-service-setting \
--setting-id arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier \
--setting-value standard
```

Windows

```
aws ssm update-service-setting ^
--setting-id arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier ^
--setting-value standard
```

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 AWS Region.

Linux & macOS

```
aws ssm get-service-setting \
--setting-id arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier
```

Windows

```
aws ssm get-service-setting ^
--setting-id arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier
```

The command returns information like the following.

```
{
 "ServiceSetting": {
 "SettingId": "/ssm/managed-instance/activation-tier",
 "SettingValue": "standard",
 },
}
```

```
 "LastModifiedDate": 1555603376.138,
 "LastModifiedUser": "System",
 "ARN": "arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/managed-instance/
activation-tier",
 "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, edge devices, and VMs in your hybrid environment to use the standard-instances tier in the specified AWS account and AWS Region.

#### To revert to the standard-instances tier using PowerShell

1. Open AWS Tools for Windows PowerShell and run the following command.

```
Update-SSMServiceSetting ^
 -SettingId "arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-instance/
activation-tier" ^
 -SettingValue "standard"
```

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 AWS Region.

```
Get-SSMServiceSetting ^
 -SettingId "arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-instance/
activation-tier"
```

The command returns information like the following.

```
ARN: arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/managed-instance/activation-
tier
LastModifiedDate : 4/18/2019 4:02:56 PM
LastModifiedUser : System
SettingId : /ssm/managed-instance/activation-tier
SettingValue : standard
Status : Default
```

The status changes to *Default* after the request has been approved.

### Resetting passwords on managed nodes

You can reset the password for any user on a managed node. This includes Amazon Elastic Compute Cloud (Amazon EC2) instances; AWS IoT Greengrass core devices; and on-premises servers, edge devices, and virtual machines (VMs) that are managed by AWS Systems Manager. The password reset functionality is built on Session Manager, a capability of AWS Systems Manager. You can use this functionality to connect to managed nodes without opening inbound ports, maintaining bastion hosts, or managing SSH keys.

Password reset is 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 a managed node.

## Prerequisites

Before you can reset the password on a managed node, the following requirements must be met:

- The managed node on which you want to change a password must be a Systems Manager managed node. Also, SSM Agent version 2.3.668.0 or later must be installed on the managed node.) For information about installing or updating SSM Agent, see [Working with SSM Agent \(p. 68\)](#).
- The password reset functionality uses the Session Manager configuration that is set up for your account to connect to the managed node. Therefore, the prerequisites for using Session Manager must have been completed for your account in the current AWS Region. For more information, see [Setting up Session Manager \(p. 916\)](#).

### Note

Session Manager support for on-premises machines is provided for the advanced-instances tier only. For more information, see [Turning on the advanced-instances tier \(p. 805\)](#).

- The AWS user who is changing the password must have the `ssm:SendCommand` permission for the managed node. For more information, see [Restricting Run Command access based on tags \(p. 993\)](#).

## Restricting access

You can limit a user's ability to reset passwords to specific managed nodes. This is done by using identity-based policies for the Session Manager `ssm:StartSession` operation with the [AWS-PasswordReset SSM document](#). For more information, see [Control user session access to instances \(p. 926\)](#).

## Encrypting data

Turn on AWS Key Management Service (AWS KMS) complete encryption for Session Manager data to use the password reset option for managed nodes. For more information, see [Turn on KMS key encryption of session data \(console\) \(p. 945\)](#).

## Reset a password on a managed node

You can reset a password on a Systems Manager managed node using the Systems Manager **Fleet Manager** console or the AWS Command Line Interface (AWS CLI).

### To change the password on a managed node (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the button next to the node that needs a new password.
4. In the **Instance actions** menu, choose **Reset password**.
5. For **User name**, enter the name of the user for which you're changing the password. This can be any user name that has an account on the node.
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 managed node doesn't support password resets, you're prompted to install a supported version using Run Command, a capability of AWS Systems Manager.

## To reset the password on a managed node (AWS CLI)

1. To reset the password for a user on a managed node, 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 \(p. 964\)](#).

Linux & macOS

```
aws ssm start-session \
--target instance-id \
--document-name "AWS-PasswordReset" \
--parameters '{"username": ["user-name"]}'
```

Windows

```
aws ssm start-session ^
--target instance-id ^
--document-name "AWS-PasswordReset" ^
--parameters username="user-name"
```

*instance-id* is the ID of a managed node configured for Systems Manager.

*user-name* is the name of the user for whom you want to reset password on the managed node.

2. Follow the prompts in the **Enter new password** command window to specify the new password.

## Troubleshoot password resets on managed nodes

Many password reset issues can be resolved by ensuring that you have completed the [password reset prerequisites \(p. 812\)](#). For other problems, use the following information to help you troubleshoot password reset issues.

### Topics

- [Managed node not available \(p. 814\)](#)
- [SSM Agent not up-to-date \(console\) \(p. 815\)](#)
- [Password reset options aren't provided \(AWS CLI\) \(p. 815\)](#)
- [No authorization to run ssm:SendCommand \(p. 815\)](#)
- [Session Manager error message \(p. 815\)](#)

### Managed node not available

**Problem:** You want to reset the password for a managed node on the **Managed instances** console page, but the node isn't in the list.

- **Solution:** The managed node you want to connect to might not be configured for Systems Manager. To use an EC2 instance with Systems Manager, an AWS Identity and Access Management (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 \(p. 21\)](#).

To use an on-premises server, edge device, or virtual machine (VM) with Systems Manager, create an IAM service role that gives Systems Manager permission to perform actions on your machines. For more information, see [Create an IAM service role for a hybrid environment \(p. 36\)](#). (Session Manager

support for on-premises servers and VMs is provided for the advanced-instances tier only. For more information, see [Turning on the advanced-instances tier \(p. 805\)](#).)

### [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 update the agent on the managed node 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. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

### [Password reset options aren't provided \(AWS CLI\)](#)

**Problem:** You connect successfully to a managed node 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 aren't displayed.

- **Solution:** The version of SSM Agent on the managed node isn't 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. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

### [No authorization to run ssm:SendCommand](#)

**Problem:** You attempt to connect to a managed node to change the password but receive an error message saying that you aren't authorized to run `ssm:SendCommand` on the managed node.

- **Solution:** Your IAM user policy must include permission to run the `ssm:SendCommand` command. For information, see [Restricting Run Command access based on tags \(p. 993\)](#).

### [Session Manager error message](#)

**Problem:** You receive an error message related to Session Manager.

- **Solution:** Password reset support requires that Session Manager is configured correctly. For information, see [Setting up Session Manager \(p. 916\)](#) and [Troubleshooting Session Manager \(p. 987\)](#).

### [Deregistering managed nodes in a hybrid environment](#)

If you no longer want to manage an on-premises server, edge device, 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 nodes in Systems Manager. AWS Systems Manager Agent (SSM Agent) running on the hybrid machine won't be able to refresh its authorization token because it's no longer registered. SSM Agent hibernates and reduce its ping frequency to Systems Manager in the cloud to once per hour.

You can reregister an on-premises server, edge device, or VM again at any time. Systems Manager stores the command history for a deregistered managed node for 30 days.

The following procedure describes how to deregister a hybrid machine by using the Systems Manager console. For information about how to do this by using the AWS Command Line Interface , see [deregister-managed-instance](#).

### To deregister a hybrid machine (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the button next to the managed node that you want to deregister.
4. In the **Instance actions** menu, choose **Deregister this managed instance**.
5. Review the information in the **Deregister this managed instance** pop-up, and then if you approve, choose **Deregister**.

## Troubleshooting managed node availability

For several AWS Systems Manager capabilities like Run Command, Distributor, and Session Manager, you can choose to manually select the managed nodes on which you want to run an operation. In cases like these, after you specify that you want to choose nodes manually, the system displays a list of managed nodes where you can run the operation.

This topic provides information to help you diagnose why a managed node *that you have confirmed is running* isn't included in your lists of managed nodes in Systems Manager.

In order for a node to be managed by Systems Manager and available in lists of managed nodes, it must meet three requirements:

- **SSM Agent** must be installed and running on the node with a supported operating system.

#### Note

Some AWS managed Amazon Machine Images (AMIs) are configured to launch instances with [SSM Agent \(p. 68\)](#) preinstalled. (You can also configure a custom AMI to preinstall SSM Agent.) For more information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

- For Amazon Elastic Compute Cloud (Amazon EC2) instances, you must attach an AWS Identity and Access Management (IAM) instance profile to the instance. The instance profile enables the instance to communicate with the Systems Manager service. If you don't assign an instance profile to the instance, you register it using a hybrid activation, which is not a common scenario.
- **SSM Agent** must be able to connect to a Systems Manager endpoint in order to register itself with the service. Thereafter, the managed node must be available to the service, which is confirmed by the service sending a signal every five minutes to check the instance's health.

After you verify that a managed node is running, you can use the following command to check whether SSM Agents successfully registered with the Systems Manager service. This command doesn't return results until a successful registration has taken place.

#### Linux & macOS

```
aws ssm describe-instance-associations-status \
```

```
--instance-id instance-id
```

### Windows

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

### PowerShell

```
Get-SSMInstanceAssociationsStatus ^
-InstanceId instance-id
```

If registration was successful and the managed node is now available for Systems Manager operations, the command returns results similar to the following.

```
{
 "InstanceAssociationStatusInfos": [
 {
 "AssociationId": "fa262de1-6150-4a90-8f53-d7eb5EXAMPLE",
 "Name": "AWS-GatherSoftwareInventory",
 "DocumentVersion": "1",
 "AssociationVersion": "1",
 "InstanceId": "i-02573cafefEXAMPLE",
 "Status": "Pending",
 "DetailedStatus": "Associated"
 },
 {
 "AssociationId": "f9ec7a0f-6104-4273-8975-82e34EXAMPLE",
 "Name": "AWS-RunPatchBaseline",
 "DocumentVersion": "1",
 "AssociationVersion": "1",
 "InstanceId": "i-02573cafefEXAMPLE",
 "Status": "Queued",
 "AssociationName": "SystemAssociationForScanningPatches"
 }
]
}
```

If registration hasn't completed yet or was unsuccessful, the command returns results similar to the following:

```
{
 "InstanceAssociationStatusInfos": []
}
```

If the command doesn't return results after 5 minutes or so, use the following information to help you troubleshoot problems with your managed nodes.

### Topics

- [Solution 1: Verify that SSM Agent is installed and running on the managed node \(p. 818\)](#)
- [Solution 2: Verify that an IAM instance profile has been specified for the instance \(EC2 instances only\) \(p. 818\)](#)
- [Solution 3: Verify service endpoint connectivity \(p. 819\)](#)
- [Solution 4: Verify target operating system support \(p. 819\)](#)
- [Solution 5: Verify you're working in the same AWS Region as the Amazon EC2 instance \(p. 819\)](#)
- [Solution 6: Verify the proxy configuration you applied to the SSM Agent on your managed node \(p. 819\)](#)

- [Solution 7: Install a TLS certificate on managed instances \(p. 819\)](#)
- [Troubleshooting Amazon EC2 managed instance availability using ssm-cli \(p. 820\)](#)

### Solution 1: Verify that SSM Agent is installed and running on the managed node

Make sure the latest version of SSM Agent is installed and running on the managed node.

To determine whether SSM Agent is installed and running on a managed node, see [Checking SSM Agent status and starting the agent \(p. 128\)](#).

To install or reinstall SSM Agent on a managed node, see the following topics:

- [Working with SSM Agent on EC2 instances for Linux \(p. 76\)](#)
- [Install SSM Agent for a hybrid environment \(Linux\) \(p. 45\)](#)
- [Working with SSM Agent on EC2 instances for Windows Server \(p. 123\)](#)
- [Install SSM Agent for a hybrid environment \(Windows\) \(p. 50\)](#)

### Solution 2: Verify that an IAM instance profile has been specified for the instance (EC2 instances only)

For Amazon Elastic Compute Cloud (Amazon EC2) instances, verify that the instance is configured with an AWS Identity and Access Management (IAM) instance profile that allows the instance to communicate with the Systems Manager API. Also verify that your user account has an IAM user trust policy that allows your account to communicate with the Systems Manager API.

#### Note

On-premises servers, edge devices, and virtual machines (VMs) use an IAM service role instead of an instance profile. For more information, see [Create an IAM service role for a hybrid environment \(p. 36\)](#).

#### To determine whether an instance profile with the necessary permissions is attached to an EC2 instance

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Instances**.
3. Choose the instance to check for an instance profile.
4. On the **Description** tab in the bottom pane, locate **IAM role** and choose the name of the role.
5. On the role **Summary** page for the instance profile, on the **Permissions** tab, ensure that **AmazonSSMManagedInstanceCore** is listed under **Permissions policies**.

If a custom policy is used instead, ensure that it provides the same permissions as **AmazonSSMManagedInstanceCore**.

[Open \*\*AmazonSSMManagedInstanceCore\*\* in the console](#)

For information about other policies that can be attached to an instance profile for Systems Manager, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#).

#### To create and attach an instance profile with the necessary permissions to a new EC2 instance

- Complete the tasks in the following topics:
  - [Create an IAM instance profile for Systems Manager \(p. 21\)](#)

- [Launch an instance that uses the Systems Manager instance profile \(console\) \(p. 27\)](#)

### To attach an existing instance profile with the necessary permissions to an existing Amazon EC2 instance

- Complete the task in the following topic:
  - [Attach the Systems Manager instance profile to an existing instance \(console\) \(p. 28\)](#)

### Solution 3: Verify 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, or by allowing HTTPS (port 443) outbound traffic to the service endpoints.

For Amazon EC2 instances, the Systems Manager service endpoint for the AWS Region the instance is used to register the instance if your virtual private cloud (VPC) configuration allows outbound traffic. However, if the VPC configuration the instance was launched in does not allow outbound traffic and you can't change this configuration to allow connectivity to the public service endpoints, you must configure interface endpoints for your VPC instead.

For more information, see [\(Optional\) Create a VPC endpoint \(p. 28\)](#).

### Solution 4: Verify target operating system support

Verify that the operation you have chosen can be run on the type of managed node you expect to see listed. Some Systems Manager operations can target only Windows instances or only Linux instances. For example, the Systems Manager (SSM) documents [AWS-InstallPowerShellModule](#) and [AWS-ConfigureCloudWatch](#) can be run only on Windows instances. In the **Run a command** page, if you choose either of these documents and select **Choose instances manually**, only your Windows instances are listed and available for selection.

### Solution 5: Verify you're working in the same AWS Region as the Amazon EC2 instance

Amazon EC2 instances are created and available in specific AWS Regions, such as the US East (Ohio) Region (us-east-2) or Europe (Ireland) Region (eu-west-1). Ensure that you're working in the same AWS Region as the Amazon EC2 instance that you want to work with. For more information, see [Choosing a Region](#) in *Getting Started with the AWS Management Console*.

### Solution 6: Verify the proxy configuration you applied to the SSM Agent on your managed node

Verify that the proxy configuration you applied to the SSM Agent on your managed node is correct. If the proxy configuration is incorrect, the node can't connect to the required service endpoints, or Systems Manager might identify the operating system of the managed node incorrectly. For more information, see [Configuring SSM Agent to use a proxy \(Linux\) \(p. 117\)](#) and [Configure SSM Agent to use a proxy for Windows Server instances \(p. 125\)](#).

### Solution 7: Install a TLS certificate on managed instances

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

A TLS certificate is already installed by default on each Amazon 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.

To verify whether the required certificate is installed on your instance run the following command based on the operating system of your instance. Be sure to replace the `region` portion of the URL with the AWS Region where your managed instance is located.

## Linux & macOS

```
curl -L https://ssm.region.amazonaws.com
```

## Windows

```
Invoke-WebRequest -Uri https://ssm.region.amazonaws.com
```

The command should return an `UnknownOperationException` error. If you receive an SSL/TLS error message instead then the required certificate might not be installed.

If you find the required Amazon Trust Services CA certificates aren't installed on your base operating systems, on instances created from AMIs that aren't supplied by Amazon, or on your own on-premises servers and VMs, you must install and allow 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](#).

## Troubleshooting Amazon EC2 managed instance availability using `ssm-cli`

For an Amazon EC2 instance to be managed by AWS Systems Manager and available in lists of managed instances, it must meet three primary requirements:

- SSM Agent must be installed and running on an instance with a supported operating system.

Some AWS managed Amazon Machine Images (AMIs) are configured to launch instances with [SSM Agent \(p. 68\)](#) preinstalled. (You can also configure a custom AMI to preinstall SSM Agent.) For more information, see [Amazon Machine Images \(AMIs\) with SSM Agent preinstalled \(p. 72\)](#).

- An AWS Identity and Access Management (IAM) instance profile that supplies the required permissions for the instance to communicate with the Systems Manager service must be attached to the instance.
- SSM Agent must be able to connect to a Systems Manager endpoint to register itself with the service. Thereafter, the instance must be available to the service, which is confirmed by the service sending a signal every five minutes to check the instance's health.

Starting with SSM Agent version 3.1.501.0, you can use `ssm-cli` to determine whether an instance meets these requirements. The `ssm-cli` is a standalone command line tool included in the SSM Agent installation. Preconfigured commands are included that gather the required information to help you diagnose why an Amazon EC2 instance that you have confirmed is running isn't included in your lists of managed instances in Systems Manager. These commands are run when you specify the `get-diagnostics` option.

Run the following command to use `ssm-cli` to help you troubleshoot Amazon EC2 managed instance availability. On a Windows instance, you must navigate to the `C:\Program Files\Amazon\SSM` directory before running the command.

Linux & macOS

```
ssm-cli get-diagnostics --output table
```

Windows

```
ssm-cli.exe get-diagnostics --output table
```

PowerShell

```
.\ssm-cli.exe get-diagnostics --output table
```

The command returns output similar to the following. Connectivity checks to the `ssmmessages`, `s3`, `kms`, `logs`, and `monitoring` endpoints are for additional optional features such as Session Manager that can log to Amazon Simple Storage Service (Amazon S3) or Amazon CloudWatch Logs, and use AWS Key Management Service (AWS KMS) encryption.

Linux & macOS

```
[root@instance]# ssm-cli get-diagnostics --output table
#####
Check # Status # Note
#
#####
EC2 IMDS # Success # IMDS is accessible and has instance
id i-0123456789abcdefa in Region
us-east-2
#
#####
Hybrid instance registration # Skipped # Instance does not have hybrid
registration
#####
Connectivity to ssm endpoint # Success # ssm.us-east-2.amazonaws.com is
reachable
#####
Connectivity to ec2messages endpoint # Success # ec2messages.us-east-2.amazonaws.com
is reachable
#####
Connectivity to ssmmessages endpoint # Success # ssmmessages.us-east-2.amazonaws.com
is reachable
#####
Connectivity to s3 endpoint # Success # s3.us-east-2.amazonaws.com is
reachable
#####
Connectivity to kms endpoint # Success # kms.us-east-2.amazonaws.com is
reachable
#####
Connectivity to logs endpoint # Success # logs.us-east-2.amazonaws.com is
reachable
#####
Connectivity to monitoring endpoint # Success # monitoring.us-east-2.amazonaws.com
is reachable
#####
AWS Credentials # Success # Credentials are for
#
```

```

arn:aws:sts::123456789012:assumed-
role/Fullaccess/i-0123456789abcdefa #
and will expire at 2021-08-17
18:47:49 +0000 UTC #
#####
Agent service # Success # Agent service is running and is
running as expected user #
#####
Proxy configuration # Skipped # No proxy configuration detected
#
#####
SSM Agent version # Success # SSM Agent version is 3.0.1209.0,
latest available agent version is #
#
3.1.192.0
#
#####

```

## Windows

```

PS C:\Program Files\Amazon\SSM> ssm-cli.exe get-diagnostics --output table
#####
Check # Status # Note
#
#####
EC2 IMDS # Success # IMDS is accessible and has instance
id i-0123456789EXAMPLE in #
Region us-east-2
#
#####
Hybrid instance registration # Skipped # Instance does not have hybrid
registration #
#####
Connectivity to ssm endpoint # Success # ssm.us-east-2.amazonaws.com is
reachable #
#####
Connectivity to ec2messages endpoint # Success # ec2messages.us-east-2.amazonaws.com
is reachable #
#####
Connectivity to ssmmessages endpoint # Success # ssmmessages.us-east-2.amazonaws.com
is reachable #
#####
Connectivity to s3 endpoint # Success # s3.us-east-2.amazonaws.com is
reachable #
#####
Connectivity to kms endpoint # Success # kms.us-east-2.amazonaws.com is
reachable #
#####
Connectivity to logs endpoint # Success # logs.us-east-2.amazonaws.com is
reachable #
#####
Connectivity to monitoring endpoint # Success # monitoring.us-east-2.amazonaws.com
is reachable #
#####
AWS Credentials # Success # Credentials are for
#
arn:aws:sts::123456789012:assumed-
role/SSM-Role/i-123abc45EXAMPLE #
and will expire at 2021-09-02
13:24:42 +0000 UTC #
#####
Agent service # Success # Agent service is running and is
running as expected user #
#####
Proxy configuration # Skipped # No proxy configuration detected
#

```

```
#####
Windows sysprep image state # Success # Windows image state value is at
desired value IMAGE_STATE_COMPLETE #
#####
SSM Agent version # Success # SSM Agent version is 3.0.1209.0,
latest agent version in us-east-2 #
#
is 3.1.192.0
#
#####
```

The following table provides additional details for each of the checks performed by `ssm-cli`.

#### **ssm-cli diagnostic checks**

Check	Details
Amazon EC2 instance metadata service	The instance is able to reach the instance metadata service. A failed test indicates a connectivity issue to <code>http://169.254.169.254</code> which can be caused by local route, proxy, or operating system (OS) firewall and proxy configurations.
Hybrid instance registration	Whether the SSM Agent is registered using a hybrid activation.
Connectivity to ssm endpoint	The instance is able to reach the service endpoints for Systems Manager on TCP port 443. A failed test indicates connectivity issues to <code>https://ssm.region.amazonaws.com</code> depending on the AWS Region where the instance is located. Connectivity issues can be caused by the VPC configuration including security groups, network access control lists, route tables, or OS firewalls and proxies.
Connectivity to ec2messages endpoint	The instance is able to reach the service endpoints for Systems Manager on TCP port 443. A failed test indicates connectivity issues to <code>https://ec2messages.region.amazonaws.com</code> depending on the AWS Region where the instance is located. Connectivity issues can be caused by the VPC configuration including security groups, network access control lists, route tables, or OS firewalls and proxies.
Connectivity to ssmmessages endpoint	The instance is able to reach the service endpoints for Systems Manager on TCP port 443. A failed test indicates connectivity issues to <code>https://ssmmessages.region.amazonaws.com</code> depending on the AWS Region where the instance is located. Connectivity issues can be caused by the VPC configuration including security groups, network access control lists, route tables, or OS firewalls and proxies.
Connectivity to s3 endpoint	The instance is able to reach the service endpoint for Amazon Simple Storage Service on TCP port 443. A failed test indicates connectivity issues to

Check	Details
	<a href="https://s3.region.amazonaws.com">https://s3.region.amazonaws.com</a> depending on the AWS Region where the instance is located. Connectivity to this endpoint is not required for an instance to appear in your managed instances list.
Connectivity to kms endpoint	The instance is able to reach the service endpoint for AWS Key Management Service on TCP port 443. A failed test indicates connectivity issues to <a href="https://kms.region.amazonaws.com">https://kms.region.amazonaws.com</a> depending on the AWS Region where the instance is located. Connectivity to this endpoint is not required for an instance to appear in your managed instances list.
Connectivity to logs endpoint	The instance is able to reach the service endpoint for Amazon CloudWatch Logs on TCP port 443. A failed test indicates connectivity issues to <a href="https://logs.region.amazonaws.com">https://logs.region.amazonaws.com</a> depending on the AWS Region where the instance is located. Connectivity to this endpoint is not required for an instance to appear in your managed instances list.
Connectivity to monitoring endpoint	The instance is able to reach the service endpoint for CloudWatch on TCP port 443. A failed test indicates connectivity issues to <a href="https://monitoring.region.amazonaws.com">https://monitoring.region.amazonaws.com</a> depending on the AWS Region where the instance is located. Connectivity to this endpoint is not required for an instance to appear in your managed instances list.
AWS Credentials	Whether the SSM Agent has the required credentials based on the IAM instance profile attached to the instance. A failed test indicates that no instance profile is attached to the instance, or it does not contain the required permissions for Systems Manager.
Agent service	Whether the SSM Agent service is running, and whether the service is running as root for Linux, or SYSTEM for Windows. A failed test indicates the SSM Agent service is not running or is not running as root or SYSTEM.
Proxy configuration	Whether the SSM Agent is configured to use a proxy.
Sysprep image state (Windows only)	The state of Sysprep on the instance. The SSM Agent will not start on the instance if the Sysprep state is a value other than IMAGE_STATE_COMPLETE.
SSM Agent version	Whether the latest available version of the SSM Agent is installed.

## Connect using Remote Desktop

You can use Fleet Manager, a capability of AWS Systems Manager, to connect to your Windows Server 2012R2 and later instances using the Remote Desktop Protocol (RDP). These Remote Desktop sessions powered by NICE DCV provide secure connections to your instances directly from your browser. With Fleet Manager, you can connect a maximum of four instances per browser window. Currently, Fleet Manager supports only English language inputs. Though you can connect to instances with Fleet Manager using RDP without opening any inbound ports, Fleet Manager only supports operating system configurations that use the default RDP port 3389 at this time. If you've changed the value of the listening port for RDP on your instance, Fleet Manager fails to establish the connection. When connecting to your instance, you can use Windows credentials or the Amazon EC2 key pair (.pem file) associated with the instance for authentication. For information about Amazon EC2 key pairs, see [Amazon EC2 key pairs and Linux instances](#) and [Amazon EC2 key pairs and Windows instances](#) in the [Amazon EC2 User Guide for Linux Instances](#) and [Amazon EC2 User Guide for Windows Instances](#).

Alternatively, if you're authenticated to the AWS Management Console using AWS Single Sign-On, Fleet Manager integrates with AWS SSO so you can connect to your instances without providing additional credentials. Fleet Manager supports AWS SSO authenticated RDP connections in the same AWS Region where you enabled AWS SSO and user names can be a maximum of 16 characters. For AWS SSO authenticated RDP connections, Fleet Manager creates a local user on the instance that persists after the connection ends. AWS SSO authenticated RDP connections are not supported for nodes that are Microsoft Active Directory domain controllers.

### Important

Note the following important details.

- To use Fleet Manager with RDP, you must have SSM Agent version 3.0.222.0 or higher running on your instances. For information about how to determine the version number running on an instance, see [Checking the SSM Agent version number \(p. 129\)](#). For information about manually installing or automatically updating SSM Agent, see [Working with SSM Agent \(p. 68\)](#).
- Fleet Manager RDP connections have a maximum session duration of 60 minutes. When that duration is reached, Fleet Manager disconnects the session. You can reconnect to the same session by using your credentials.
- Fleet Manager RDP connections have an idle session timeout of 10 minutes. When that duration is reached, Fleet Manager disconnects the session. You can reconnect to the same session by using your credentials.

Because Fleet Manager uses Session Manager to connect to Windows instances using RDP, you must complete the prerequisites for Session Manager before using this feature. Session Manager is a capability of AWS Systems Manager. Session preferences in the AWS account and AWS Region are applied when connecting to your instances using RDP. For information about setting up Session Manager, see [Setting up Session Manager \(p. 916\)](#).

In addition to the required AWS Identity and Access Management (IAM) permissions for Systems Manager and Session Manager, the user or role you use to access the console must allow the following actions:

- `ssm-guiconnect:CancelConnection`
- `ssm-guiconnect:GetConnection`
- `ssm-guiconnect:StartConnection`

### To connect to instances using RDP with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the button next to the instance that you want to connect to using RDP.
4. In the **Node actions** menu, choose **Connect with Remote Desktop**.
5. Choose your preferred **Authentication type**. If you choose **User credentials**, enter the user name and password for the Windows user account that you want to use when connecting to the instance. If you choose **Key pair**, choose the **Browse local machine** option to browse your local machine and choose the PEM key associated with your instance, or copy and paste the contents into the empty field after choosing the **Paste key pair content** option.
6. Select **Connect**.

## Working with the file system

You can use Fleet Manager, a capability of AWS Systems Manager, to work with the file system on your managed nodes. Using Fleet Manager, you can view information about the directory and file data stored on the volumes attached to your managed nodes. For example, you can view the name, size, extension, owner, and permissions for your directories and files. Up to 10,000 lines of file data can be previewed as text from the Fleet Manager console. You can also use this feature to `tail` files. When using `tail` to view file data, the last 10 lines of the file are displayed initially. As new lines of data are written to the file, the view is updated in real time. As a result, you can review log data from the console, which can improve the efficiency of your troubleshooting and systems administration. Additionally, you can create directories and copy, cut, paste, rename, or delete files and directories.

We recommend creating regular backups, or taking snapshots of the Amazon Elastic Block Store (Amazon EBS) volumes attached to your managed nodes. When copying, or cutting and pasting files, existing files and directories in the destination path with the same name as the new files or directories are replaced. Serious problems can occur if you replace or modify system files and directories. AWS doesn't guarantee that these problems can be solved. Modify system files at your own risk. You're responsible for all file and directory changes, and ensuring you have backups. Deleting or replacing files and directories can't be undone.

### Note

Fleet Manager uses Session Manager, a capability of AWS Systems Manager, to view text previews and `tail` files. For Amazon Elastic Compute Cloud (Amazon EC2) instances, the instance profile attached to your managed instances must provide permissions for Session Manager to use this feature. For more information about adding Session Manager permissions to an instance profile, see [Adding Session Manager permissions to an existing IAM role \(p. 921\)](#). Also, AWS Key Management Service (AWS KMS) encryption must be turned on in your session preferences to use Fleet Manager features. For more information about enabling AWS KMS encryption for Session Manager, see [Turn on KMS key encryption of session data \(console\) \(p. 945\)](#).

### To view the file system with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Select the link of the managed node with the file system you want to view.

4. In the **Tools** menu, choose **File system**.

#### To view text previews of files with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Select the link of the managed node with the files you want to preview.
4. In the **Tools** menu, choose **File system**.
5. Select the **File name** of the directory that contains the file you want to preview.
6. Choose the button next to the file whose content you want to preview.
7. In the **Actions** menu, choose **View text preview**.

#### To tail files with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Select the link of the managed node with the files you want to tail.
4. In the **Tools** menu, choose **File system**.
5. Select the **File name** of the directory that contains the file you want to tail.
6. Choose the button next to the file whose content you want to tail.
7. In the **Actions** menu, choose **View with tail**.

#### To copy or cut and paste files or directories with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Select the link of the managed node with the files you want to copy, or cut and paste.
4. In the **Tools** menu, choose **File system**.
5. To copy or cut a file, select the **File name** of the directory that contains the file you want to copy or cut. To copy or cut a directory, choose the button next to the directory that you want to copy or cut and then proceed to step 8.
6. Choose the button next to the file you want to copy or cut.
7. In the **Actions** menu, choose **Copy or Cut**.
8. In the **File system** view, choose the button next to the directory you want to paste the file in.

9. In the **Actions** menu, choose **Paste**.

#### To rename files or directories with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Select the link of the managed node with the files or directories you want to rename.
4. In the **Tools** menu, choose **File system**.
5. To rename a file, select the **File name** of the directory that contains the file you want to rename. To rename a directory, choose the button next to the directory that you want to rename and then proceed to step 8.
6. Choose the button next to the file whose content you want to rename.
7. In the **Actions** menu, choose **Rename**.
8. In the **File name** field, enter the new name for the file and select **Rename**.

#### To delete files or directories with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Select the link of the managed node with the files or directories you want to delete.
4. In the **Tools** menu, choose **File system**.
5. To delete a file, select the **File name** of the directory that contains the file you want to delete. To delete a directory, choose the button next to the directory that you want to delete and then proceed to step 7.
6. Choose the button next to the file with the content you want to delete.
7. In the **Actions** menu, choose **Delete**.

#### To create a directory with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Select the link of the managed node you want to create a directory in.
4. In the **Tools** menu, choose **File system**.
5. Select the **File name** of the directory where you want to create a new directory.
6. Select **Create directory**.

7. In the **Directory name** field, enter the name for the new directory and select **Create directory**.

## Monitoring managed node performance

You can use Fleet Manager, a capability of AWS Systems Manager, to view performance data about your managed nodes in real time. The performance data is retrieved from performance counters.

The following performance counters are available in Fleet Manager:

- CPU utilization
- Disk input/output (I/O) utilization
- Network traffic
- Memory usage

### Note

Fleet Manager uses Session Manager, a capability of AWS Systems Manager, to retrieve performance data. For Amazon Elastic Compute Cloud (Amazon EC2) instances, the instance profile attached to your managed instances must provide permissions for Session Manager to use this feature. For more information about adding Session Manager permissions to an instance profile, see [Adding Session Manager permissions to an existing IAM role \(p. 921\)](#). Also, AWS Key Management Service (AWS KMS) encryption must be turned on in your session preferences to use Fleet Manager features. For more information about turning on AWS KMS encryption for Session Manager, see [Turn on KMS key encryption of session data \(console\) \(p. 945\)](#).

### To view performance data with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the button next to the managed node whose performance you want to monitor.
4. Choose **View details**.
5. In the **Tools** menu, choose **Performance counters**.

## Working with processes

You can use Fleet Manager, a capability of AWS Systems Manager, to work with processes on your managed instances. Using Fleet Manager, you can view information about processes. For example, you can see the CPU utilization and memory usage of processes in addition to their handles and threads. With Fleet Manager, you can start and terminate processes from the console.

### To view details about processes with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Select the link of the instance whose processes you want to view.
4. In the **Tools** menu, choose **Processes**.

### To start a process with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

- If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Select the link of the instance you want to start a process on.
  4. In the **Tools** menu, choose **Processes**.
  5. Select **Start new process**.
  6. In the **Process name or full path** field, enter the name of the process or the full path to the executable.
  7. (Optional) In the **Working directory** field, enter the directory path where you want the process to run.

### To terminate a process with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

- If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Select the link of the instance you want to start a process on.
  4. In the **Tools** menu, choose **Processes**.
  5. Choose the button next to the process you want to terminate.
  6. In the **Actions** dropdown, choose **Terminate process** or **Terminate process tree**.

#### Note

Terminating a process tree also terminates all processes and applications using that process.

## View logs

You can use Fleet Manager, a capability of AWS Systems Manager, to view log data stored on your managed nodes. For Windows managed nodes, you can view Windows event logs and copy their details from the console. To help you search events, filter Windows event logs by **Event level**, **Event ID**, **Event source**, and **Time created**. You can also view other log data using the procedure to view the file system. For more information about viewing the file system with Fleet Manager, see [Working with the file system \(p. 826\)](#).

### To view Windows event logs with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

- If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Choose the button next to the managed node whose event logs you want to view.
  4. Choose **View details**.
  5. In the **Tools** menu, choose **Windows event logs**.
  6. Choose the **Log name** that contains the events you want to view.
  7. Choose the button next to the **Log name** you want to view, and then select **View events**.
  8. Choose the button next to the event you want to view, and then select **View event details**.
  9. (Optional) Select **Copy as JSON** to copy the event details to your clipboard.

## User management

You can use Fleet Manager, a capability of AWS Systems Manager, to manage operating system (OS) user accounts on your managed nodes. For example, you can create and delete users and groups. Additionally, you can view details like group membership, user roles, and status.

### Important

Fleet Manager uses Run Command and Session Manager, capabilities of AWS Systems Manager, for various user management operations. As a result, a user could grant permissions to an operating system user account that they would otherwise be unable to. This is because AWS Systems Manager Agent (SSM Agent) runs on Amazon Elastic Compute Cloud (Amazon EC2) instances using root permissions (Linux) or SYSTEM permissions (Windows Server). For more information about restricting access to root-level commands through the SSM Agent, see [Restricting access to root-level commands through SSM Agent \(p. 134\)](#). To restrict access to this feature, we recommend creating AWS Identity and Access Management (IAM) policies for your users that only allow access to the actions you define. For more information about creating IAM policies for Fleet Manager, see [Step 1: Create an IAM policy with Fleet Manager permissions \(p. 800\)](#).

### Create a user or group

#### Note

Fleet Manager uses Session Manager to set passwords for new users. For Amazon EC2 instances, the instance profile attached to your managed instances must provide permissions for Session Manager to use this feature. For more information about adding Session Manager permissions to an instance profile, see [Adding Session Manager permissions to an existing IAM role \(p. 921\)](#). Also, AWS Key Management Service (AWS KMS) encryption must be turned on in your session preferences to use Fleet Manager features. For more information about enabling AWS KMS encryption for Session Manager, see [Turn on KMS key encryption of session data \(console\) \(p. 945\)](#).

#### To create an OS user account with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
  2. In the navigation pane, choose **Fleet Manager**.
- or-
- If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Choose the button next to the managed node you want to create a new user on.
  4. Choose **View details**.
  5. In the **Tools** menu, choose **Users and groups**.
  6. Choose the **Users** tab, and then choose **Create user**.

7. Enter a value for the **Name** of the new user.
8. (Recommended) Select the check box next to **Set password**. You will be prompted to provide a password for the new user at the end of the procedure.
9. Select **Create user**. If you selected the check box to create a password for the new user, you will be prompted to enter a value for the password and select **Done**. If the password you specify doesn't meet the requirements specified by your managed node's local or domain policies, an error is returned.

### To create an OS group with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the button next to the managed node you want to create a group in.
4. Choose **View details**.
5. In the **Tools** menu, choose **Users and groups**.
6. Choose the **Groups** tab, and then choose **Create group**.
7. Enter a value for the **Name** of the new group.
8. (Optional) Enter a value for the **Description** of the new group.
9. (Optional) Select users to add to the **Group members** for the new group.
10. Select **Create group**.

### Update user or group membership

#### To add an OS user account to a new group with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the button next to the managed node where the user account exists that you want to update.
4. Choose **View details**.
5. In the **Tools** menu, choose **Users and groups**.
6. Choose the **Users** tab.
7. Choose the button next to the user you want to update.
8. In the **Actions** menu, choose **Add user to group**.
9. Choose the group you want to add the user to under **Add to group**.
10. Select **Add user to group**.

#### To edit an OS group's membership with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.

2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the button next to the managed node where the group exists that you want to update.
4. Choose **View details**.
5. In the **Tools** menu, choose **Users and groups**.
6. Choose the **Groups** tab.
7. Choose the button next to the group you want to update.
8. In the **Actions** menu, choose **Modify group**.
9. Choose the users you want to add or remove under **Group members**.
10. Select **Modify group**.

## Delete a user or group

### To delete an OS user account with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the button next to the managed node where the user account exists that you want to delete.
4. Choose **View details**.
5. In the **Tools** menu, choose **Users and groups**.
6. Choose the **Users** tab.
7. Choose the button next to the user you want to delete.
8. In the **Actions** menu, choose **Delete local user**.

### To delete an OS group with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the button next to the managed node where the group exists that you want to delete.
4. Choose **View details**.
5. In the **Tools** menu, choose **Users and groups**.
6. Choose the **Group** tab.
7. Choose the button next to the group you want to update.
8. In the **Actions** menu, choose **Delete local group**.

## Windows registry management

You can use Fleet Manager, a capability of AWS Systems Manager, to manage the registry on your Windows managed nodes. From the Fleet Manager console you can create, copy, update, and delete registry entries and values.

### Important

We recommend creating a backup of the registry, or taking a snapshot of the root Amazon Elastic Block Store (Amazon EBS) volume attached to your managed node, before you modify the registry. Serious problems can occur if you modify the registry incorrectly. These problems might require you to reinstall the operating system, or restore the root volume of your node from a snapshot. AWS doesn't guarantee that these problems can be solved. Modify the registry at your own risk. You're responsible for all registry changes, and ensuring you have backups.

## Create a Windows registry key or entry

### To create a Windows registry key with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.  
-or-
- If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Choose the button next to the managed node you want to create a registry key on.
4. Choose **View details**.
5. In the **Tools** menu, choose **Windows registry**.
6. Choose the hive you want to create a new registry key in by selecting the **Registry name**.
7. In the **Create** menu, choose **Create registry key**.
8. Choose the button next to the registry entry you want to create a new key in.
9. Choose **Create registry key**.
10. Enter a value for the **Name** of the new registry key, and select **Submit**.

### To create a Windows registry entry with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.  
-or-
- If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Choose the button next to the instance you want to create a registry entry on.
4. Choose **View details**.
5. In the **Tools** menu, choose **Windows registry**.
6. Choose the hive, and subsequent registry key you want to create a new registry entry in by selecting the **Registry name**.
7. In the **Create** menu, choose **Create registry entry**.
8. Enter a value for the **Name** of the new registry entry.
9. Choose the **Type** of value you want to create for the registry entry. For more information about registry value types, see [Registry value types](#).

10. Enter a value for the **Value** of the new registry entry.

## Update a Windows registry entry

### To update a Windows registry entry with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.  
-or-  
If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Choose the button next to the managed node you want to update a registry entry on.
4. Choose **View details**.
5. In the **Tools** menu, choose **Windows registry**.
6. Choose the hive, and subsequent registry key you want to update by selecting the **Registry name**.
7. Choose the button next to the registry entry you want to update.
8. In the **Actions** menu, choose **Update registry entry**.
9. Enter the new value for the **Value** of the registry entry.
10. Choose **Update**.

## Delete a Windows registry entry or key

### To delete a Windows registry key with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.  
-or-  
If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Choose the button next to the managed node you want to delete a registry key on.
4. In the **Tools** menu, choose **Windows registry**.
5. Choose the hive, and subsequent registry key you want to delete by selecting the **Registry name**.
6. Choose the button next to the registry key you want to delete.
7. In the **Actions** menu, choose **Delete registry key**.

### To delete a Windows registry entry with Fleet Manager

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.  
-or-  
If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Choose the button next to the managed node you want to delete a registry entry on.
4. Choose **View details**.

5. In the **Tools** menu, choose **Windows registry**.
6. Choose the hive, and subsequent registry key containing the entry you want to delete by selecting the **Registry name**.
7. Choose the button next to the registry entry you want to delete.
8. In the **Actions** menu, choose **Delete registry entry**.

## Accessing the Red Hat Knowledgebase portal

You can use Fleet Manager, a capability of AWS Systems Manager, to access the Knowledgebase portal if you are a Red Hat customer. You are considered a Red Hat customer if you run Red Hat Enterprise Linux (RHEL) instances or use RHEL services on AWS. The Knowledgebase portal includes binaries, and knowledge-share and discussion forums for community support that are available only to Red Hat licensed customers.

In addition to the required AWS Identity and Access Management (IAM) permissions for Systems Manager and Fleet Manager, the user or role you use to access the console must allow the `rhelkb:GetRhelURL` action to access the Knowledgebase portal.

### To access the Red Hat Knowledgebase Portal

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.  
-or-  
If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Choose the RHEL instance you want to use to connect to the Red Hat Knowledgebase Portal.
4. Select the **Account management** dropdown. Then choose **Access Red Hat Knowledgebase**. You will then be redirected to the Red Hat Knowledgebase page.

If you use RHEL on AWS to run fully supported RHEL workloads, you can also access the Red Hat Knowledgebase through Red Hat's website by using your AWS credentials.

## AWS Systems Manager Compliance

You can use Compliance, a capability of AWS Systems Manager, to scan your fleet of managed nodes 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, Compliance displays current compliance data about patching in Patch Manager and associations in State Manager. (Patch Manager and State Manager are also both capabilities of AWS Systems Manager.) To get started with Compliance, open the [Systems Manager console](#). In the navigation pane, choose **Compliance**.

Patch compliance data from Patch Manager can be sent to AWS Security Hub. Security Hub gives you a comprehensive view of your high-priority security alerts and compliance status. It also monitors the patching status of your fleet. For more information, see [Integrating Patch Manager with AWS Security Hub \(p. 1207\)](#).

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 Compliance to create your own compliance types based on your IT or business requirements.
- Remediate issues by using Run Command, another capability of AWS Systems Manager, State Manager, or Amazon EventBridge.
- Port data to Amazon Athena and Amazon QuickSight to generate fleet-wide reports.

### EventBridge support

This Systems Manager capability is supported as an *event* type in Amazon EventBridge rules. For information, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#) and [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#).

### Chef InSpec integration

Systems Manager integrates with [Chef InSpec](#). InSpec is an open-source, runtime framework that allows you to create human-readable profiles on GitHub or Amazon Simple Storage Service (Amazon S3). You can then use Systems Manager to run compliance scans and view compliant and noncompliant managed nodes. For more information, see [Using Chef InSpec profiles with Systems Manager Compliance \(p. 1498\)](#).

### Pricing

Compliance is offered at no additional charge. You only pay for the AWS resources that you use.

### Contents

- [Getting started with Compliance \(p. 837\)](#)
- [Creating a resource data sync for Compliance \(p. 838\)](#)
- [Working with Compliance \(p. 839\)](#)
- [Deleting a resource data sync for Compliance \(p. 842\)](#)
- [Remediating compliance issues using EventBridge \(p. 843\)](#)
- [Compliance walkthrough \(AWS CLI\) \(p. 844\)](#)

## Getting started with Compliance

To get started with Compliance, a capability of AWS Systems Manager, complete the following tasks.

Task	For more information
Compliance works with patch data in Patch Manager and associations in State Manager. (Patch Manager and State Manager are also both capabilities of AWS Systems Manager.) Compliance also works with custom compliance types on managed nodes that are managed using Systems Manager. Verify that your Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and on-premises servers and virtual machines are configured as managed nodes by verifying Systems Manager prerequisites.	<a href="#">Systems Manager prerequisites (p. 13)</a>
Update Systems Manager SSM Agent (SSM Agent) on your managed nodes to the latest version.	<a href="#">Working with SSM Agent (p. 68)</a>

Task	For more information
If you plan to monitor patch compliance, verify that you've configured Patch Manager. You must perform patching operations by using Patch Manager before Compliance can display patch compliance data.	<a href="#">AWS Systems Manager Patch Manager (p. 1093)</a>
If you plan to monitor association compliance, verify that you've created State Manager associations. You must create associations before Compliance can display association compliance data.	<a href="#">AWS Systems Manager State Manager (p. 1034)</a>
(Optional) Configure the system to view compliance history and change tracking.	<a href="#">Viewing compliance configuration history and change tracking (p. 842)</a>
(Optional) Create custom compliance types.	<a href="#">Compliance walkthrough (AWS CLI) (p. 844)</a>
(Optional) Create a resource data sync to aggregate all compliance data in a target Amazon Simple Storage Service (Amazon S3) bucket.	<a href="#">Creating a resource data sync for Compliance (p. 838)</a>

## Creating a resource data sync for Compliance

You can use the resource data sync feature in AWS Systems Manager to send compliance data from all of your managed nodes to a target Amazon Simple Storage Service (Amazon S3) bucket. When you create the sync, you can specify managed nodes 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 Compliance is a one-time operation.

Use the following procedure to create a resource data sync for Compliance by using the AWS Management Console.

### To create and configure an Amazon 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 compliance data. For more information, see [Create a Bucket](#) in the *Amazon Simple Storage Service User 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 **DOC-EXAMPLE-BUCKET** 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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "SSMBucketPermissionsCheck",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 }
]
}
```

```
 "Action": "s3:GetBucketAcl",
 "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
 },
 {
 "Sid": "SSMBucketDelivery",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "s3:PutObject",
 "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/Bucket-Prefix/*",
accountid=Account_ID_number/*"],
 "Condition": {
 "StringEquals": {
 "s3:x-amz-acl": "bucket-owner-full-control"
 }
 }
 }
]
```

### To create a resource data sync

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.  
-or-  
If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. Choose **Account management, Resource Data Syncs**, and then choose **Create resource data sync**.
4. In the **Sync name** field, enter a name for the sync configuration.
5. In the **Bucket name** field, enter the name of the Amazon S3 bucket you created at the start of this procedure.
6. (Optional) In the **Bucket prefix** field, enter the name of an Amazon S3 bucket prefix (subdirectory).
7. In the **Bucket region** field, choose **This region** if the Amazon 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 enter the name of the Region.

**Note**

If the sync and the target Amazon S3 bucket are located in different Regions, you might be subject to data transfer pricing. For more information, see [Amazon S3 Pricing](#).

8. Choose **Create**.

## Working with Compliance

Compliance, a capability of AWS Systems Manager, collects and reports data about the status of patching in Patch Manager patching and associations in State Manager. (Patch Manager and State Manager are also both capabilities of AWS Systems Manager.) Compliance also reports on custom compliance types you have specified for your managed nodes. 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 allows you to create human-readable profiles on GitHub or Amazon Simple Storage Service (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 \(p. 1498\)](#).

## About patch compliance

After you use Patch Manager to install patches on your instances, compliance status information is immediately available to you in the console or in response to AWS Command Line Interface (AWS CLI) commands or corresponding Systems Manager API operations.

For information about patch compliance status values, see [Understanding patch compliance state values \(p. 1186\)](#).

## 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 operations. For associations, 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 node. 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 nodes. 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 operation to explicitly note which managed nodes are running older versions of software X. 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::.

Linux & macOS

```
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
```

Windows

```
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
```

### Note

The ResourceType parameter only supports ManagedInstance. If you add custom compliance to a managed AWS IoT Greengrass core device, you must specify a ResourceType of ManagedInstance.

Compliance managers can then view summaries or create reports about which managed nodes are or aren't compliant. You can assign a maximum of 10 different custom compliance types to a managed node.

For an example of how to create a custom compliance type and view compliance data, see [Compliance walkthrough \(AWS CLI\) \(p. 844\)](#).

## Viewing current compliance data

This section describes how to view compliance data in the 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 \(p. 842\)](#).

### Topics

- [Viewing current compliance data \(console\) \(p. 841\)](#)
- [Viewing current compliance data \(AWS CLI\) \(p. 841\)](#)

### 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Compliance dashboard filtering** section, choose an option to filter compliance data. The **Compliance resources summary** section displays counts of compliance data based on the filter you chose.
  4. To drill down into a resource for more information, scroll down to the **Details overview for resources** area and choose the ID of a managed node.
  5. On the **Instance ID or Name** details page, choose the **Configuration compliance** tab to view a detailed configuration compliance report for the managed node.

#### Note

For information about fixing compliance issues, see [Remediating compliance issues using EventBridge \(p. 843\)](#).

### Viewing current compliance data (AWS CLI)

You can view summaries of compliance data for patching, associations, and custom compliance types 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 \(AWS CLI\) \(p. 1238\)](#).

## Viewing compliance configuration history and change tracking

Systems Manager Compliance displays *current* patching and association compliance data for your managed nodes. 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 turn on 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](#).

## Deleting a resource data sync for Compliance

If you no longer want to use AWS Systems Manager Compliance to view compliance data, then we also recommend deleting resource data syncs used for Compliance data collection.

#### To delete a Compliance resource data sync

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose **Account management, Resource data syncs**.
4. Choose a sync in the list.

#### Important

Make sure you choose the sync used for Compliance. Systems Manager supports resource data sync for multiple capabilities. If you choose the wrong sync, you could disrupt data aggregation for Systems Manager Explorer or Systems Manager Inventory.

5. Choose **Delete**.

6. Delete the Amazon Simple Storage Service (Amazon S3) bucket where the data was stored. For information about deleting an Amazon S3 bucket, see [Deleting a bucket](#).

## Remediating compliance issues using EventBridge

You can quickly remediate patch and association compliance issues by using Run Command, a capability of AWS Systems Manager. You can target instance or AWS IoT Greengrass core device IDs or 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 operations didn't resolve the problem.

For more information about patching, see [AWS Systems Manager Patch Manager \(p. 1093\)](#) and [About the AWS-RunPatchBaseline SSM document \(p. 1128\)](#).

For more information about associations, see [Working with associations in Systems Manager \(p. 1039\)](#).

For more information about running a command, see [Sending commands using Systems Manager Run Command \(p. 995\)](#).

### Specify Compliance as the target of an EventBridge event

You can also configure Amazon EventBridge to perform an action in response to Systems Manager Compliance events. For example, if one or more managed nodes fail to install Critical patch updates or run an association that installs anti-virus software, then you can configure EventBridge to run the AWS-RunPatchBaseline document or the AWS-RefreshAssociation document when the Compliance event occurs.

Use the following procedure to configure Compliance as the target of an EventBridge event.

#### To configure Compliance as the target of a EventBridge event (console)

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**, and then choose **Create rule**.  
-or-  
If the EventBridge home page opens first, choose **Create rule**.
3. Enter a name and description for the rule.  
A rule can't have the same name as another rule in the same AWS Region and on the same event bus.
4. For **Define pattern**, choose **Event pattern**. Use **Event pattern** to build a rule that generates events for specific actions in AWS services.
5. Choose **Pre-defined pattern by service**.
6. For **Service provider**, choose **AWS**.
7. For **Service Name**, choose **Systems Manager**.
8. For **Event type**, choose **Configuration Compliance**.
9. For **Select event bus**, choose the event bus that you want to associate with this rule. If you want this rule to initiate on matching events that come from your own AWS account, select **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
10. For **Target**, choose **Systems Manager Run Command**.

11. In the **Document** list, choose a Systems Manager document (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.
12. 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.
13. (Optional) Enter one or more tags for the rule. For more information, see [Tagging Your Amazon EventBridge Resources](#) in the *Amazon EventBridge User Guide*.
14. Choose **Create** and complete the wizard.

## Compliance walkthrough (AWS CLI)

The following procedure walks you through the process of using the AWS Command Line Interface (AWS CLI) to call the AWS Systems Manager [PutComplianceItems](#) API operation to assign custom compliance metadata to a resource. You can also use this API operation to manually assign patch or association compliance metadata to a managed nodes, as shown in the following walkthrough. For more information about custom compliance, see [About custom compliance \(p. 840\)](#).

### To assign custom compliance metadata to a managed instance (AWS CLI)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to assign custom compliance metadata to a managed node. The `ResourceType` parameter only supports a value of `ManagedInstance`. Specify this value even if you are assigning custom compliance metadata to a managed AWS IoT Greengrass core device.

Linux & macOS

```
aws ssm put-compliance-items \
--resource-id instance_ID \
--resource-type ManagedInstance \
--compliance-type Custom:user-defined_string \
--execution-summary ExecutionTime=user-defined_time_and/or_date_value \
--items Id=user-defined_ID,Title=user-defined_title,Severity=one_or_more_comma-separated_severities:CRITICAL, MAJOR, MINOR, INFORMATIONAL, or UNSPECIFIED,Status=COMPLIANT or NON_COMPLIANT
```

Windows

```
aws ssm put-compliance-items ^
--resource-id instance_ID ^
--resource-type ManagedInstance ^
--compliance-type Custom:user-defined_string ^
--execution-summary ExecutionTime=user-defined_time_and/or_date_value ^
--items Id=user-defined_ID,Title=user-defined_title,Severity=one_or_more_comma-separated_severities:CRITICAL, MAJOR, MINOR, INFORMATIONAL, or UNSPECIFIED,Status=COMPLIANT or NON_COMPLIANT
```

3. Repeat the previous step to assign additional custom compliance metadata to one or more nodes. You can also manually assign patch or association compliance metadata to managed nodes by using the following commands:

Association compliance metadata

Linux & macOS

```
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
```

Windows

```
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
```

Patch compliance metadata

Linux & macOS

```
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}"
```

Windows

```
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}"
```

- Run the following command to view a list of compliance items for a specific managed node. Use filters to drill down into specific compliance data.

Linux & macOS

```
aws ssm list-compliance-items \
```

```
--resource-ids instance_ID \
--resource-types ManagedInstance \
--filters one_or_more_filters
```

#### Windows

```
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.

#### Linux & macOS

```
aws ssm list-compliance-items \
--resource-ids i-02573cafccEXAMPLE \
--resource-type ManagedInstance \
--filters Key=DocumentName,Values=AWS-RunPowerShellScript
Key>Status,Values=NON_COMPLIANT,Type=NotEqual
Key=Id,Values=cee20ae7-6388-488e-8be1-a88ccEXAMPLE Key=Severity,Values=UNSPECIFIED
```

#### Windows

```
aws ssm list-compliance-items ^
--resource-ids i-02573cafccEXAMPLE ^
--resource-type ManagedInstance ^
--filters Key=DocumentName,Values=AWS-RunPowerShellScript
Key>Status,Values=NON_COMPLIANT,Type=NotEqual
Key=Id,Values=cee20ae7-6388-488e-8be1-a88ccEXAMPLE Key=Severity,Values=UNSPECIFIED
```

#### Linux & macOS

```
aws ssm list-resource-compliance-summaries \
--filters Key=OverallSeverity,Values=UNSPECIFIED
```

#### Windows

```
aws ssm list-resource-compliance-summaries ^
--filters Key=OverallSeverity,Values=UNSPECIFIED
```

#### Linux & macOS

```
aws ssm list-resource-compliance-summaries \
--filters Key=OverallSeverity,Values=UNSPECIFIED
Key=ComplianceType,Values=Association Key=InstanceId,Values=i-02573cafccEXAMPLE
```

#### Windows

```
aws ssm list-resource-compliance-summaries ^
--filters Key=OverallSeverity,Values=UNSPECIFIED
Key=ComplianceType,Values=Association Key=InstanceId,Values=i-02573cafccEXAMPLE
```

- 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 One or more filters.
```

The following examples show you how to use this command with filters.

#### Linux & macOS

```
aws ssm list-resource-compliance-summaries \
--filters Key=ExecutionType,Values=Command
```

#### Windows

```
aws ssm list-resource-compliance-summaries ^
--filters Key=ExecutionType,Values=Command
```

#### Linux & macOS

```
aws ssm list-resource-compliance-summaries \
--filters Key=AWS:InstanceInformation.PlatformType,Values=Windows
Key=OverallSeverity,Values=CRITICAL
```

#### Windows

```
aws ssm list-resource-compliance-summaries ^
--filters Key=AWS:InstanceInformation.PlatformType,Values=Windows
Key=OverallSeverity,Values=CRITICAL
```

- 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 One or more filters.
```

The following examples show you how to use this command with filters.

#### Linux & macOS

```
aws ssm list-compliance-summaries \
--filters Key=AWS:InstanceInformation.PlatformType,Values=Windows
Key=PatchGroup,Values=TestGroup
```

#### Windows

```
aws ssm list-compliance-summaries ^
--filters Key=AWS:InstanceInformation.PlatformType,Values=Windows
Key=PatchGroup,Values=TestGroup
```

#### Linux & macOS

```
aws ssm list-compliance-summaries \
```

```
--filters Key=AWS:InstanceInformation.PlatformType,Values=Windows
Key=ExecutionId,Values=4adf0526-6aed-4694-97a5-14522EXAMPLE
```

Windows

```
aws ssm list-compliance-summaries ^
--filters Key=AWS:InstanceInformation.PlatformType,Values=Windows
Key=ExecutionId,Values=4adf0526-6aed-4694-97a5-14522EXAMPLE
```

## AWS Systems Manager Inventory

AWS Systems Manager Inventory provides visibility into your AWS computing environment. You can use Inventory to collect *metadata* from your managed nodes. 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 nodes are running the software and configurations required by your software policy, and which nodes need to be updated. You can configure Inventory on all of your managed nodes by using a one-click procedure. You can also configure and view inventory data from multiple AWS Regions and AWS accounts. To get started with Inventory, open the [Systems Manager console](#). In the navigation pane, choose **Inventory**.

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 node in a specific directory. When Systems Manager Inventory collects data, it captures this custom inventory data. For example, if you run a large data center, 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 nodes. Inventory doesn't access proprietary information or data.

The following table describes the types of data you can collect with Systems Manager Inventory. The table also describes different offerings for targeting nodes and the collection intervals you can specify.

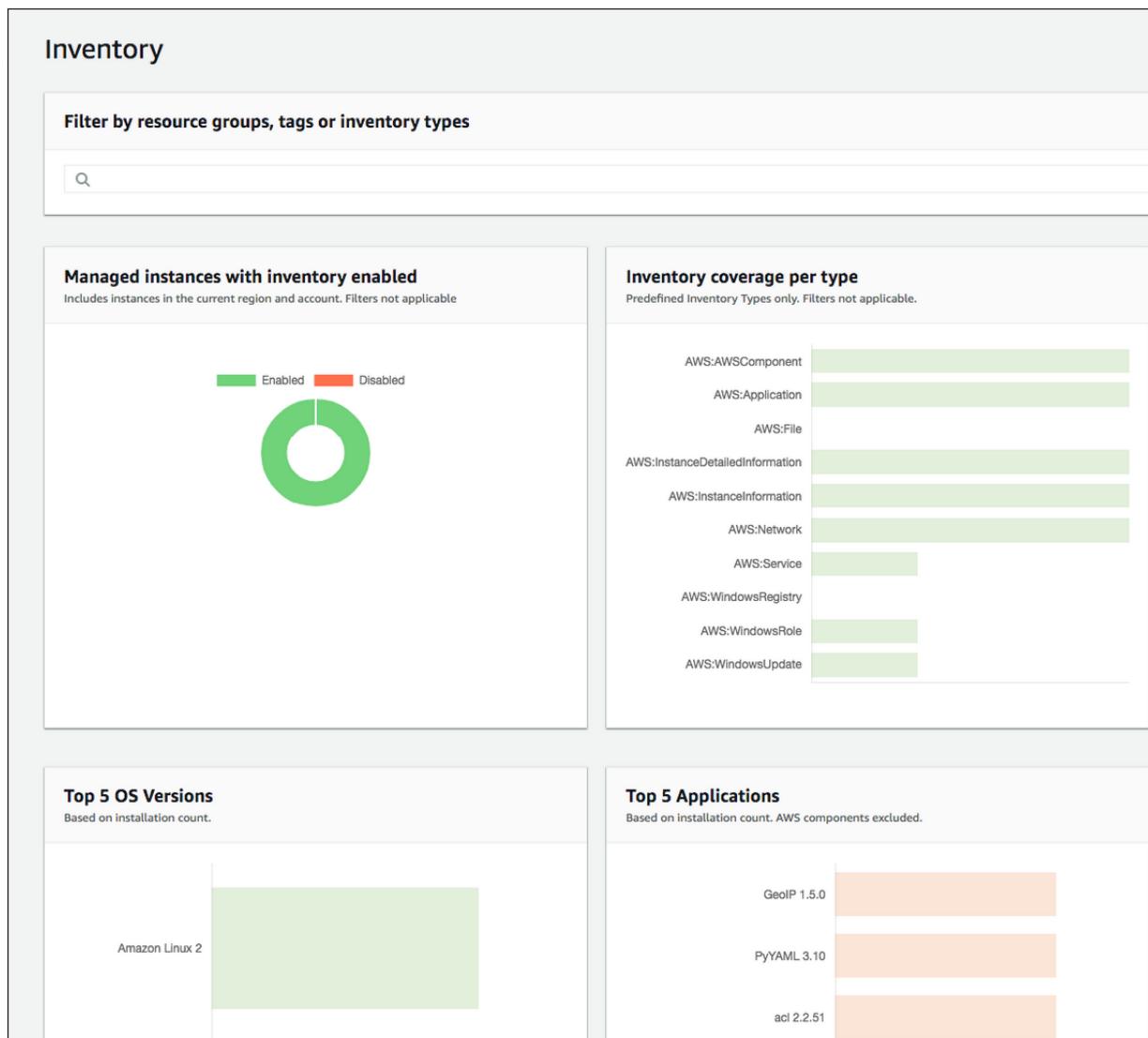
Configuration	Details
Metadata types	You can configure Inventory to collect the following types of data: <ul style="list-style-type: none"><li><b>Applications:</b> Application names, publishers, versions, etc.</li><li><b>AWS components:</b> EC2 driver, agents, versions, etc.</li><li><b>Files:</b> Name, size, version, installed date, modification and last accessed times, etc.</li><li><b>Network configuration:</b> IP address, MAC address, DNS, gateway, subnet mask, etc.</li><li><b>Windows updates:</b> Hotfix ID, installed by, installed date, etc.</li><li><b>Instance details:</b> System name, operating systems (OS) name, OS version, DNS, domain, work group, OS architecture, etc.</li><li><b>Services:</b> Name, display name, status, dependent services, service type, start type, etc.</li></ul>

Configuration	Details
	<ul style="list-style-type: none"> <li>• <b>Tags:</b> Tags assigned to your nodes.</li> <li>• <b>Windows Registry:</b> Registry key path, value name, value type, and value.</li> <li>• <b>Windows roles:</b> Name, display name, path, feature type, installed state, etc.</li> <li>• <b>Custom inventory:</b> Metadata that was assigned to a managed node as described in <a href="#">Working with custom inventory (p. 885)</a>.</li> </ul> <p><b>Note</b> To view a list of all metadata collected by Inventory, see <a href="#">Metadata collected by inventory (p. 851)</a>.</p>
Nodes to target	You can choose to inventory all managed nodes in your AWS account, individually select nodes, or target groups of nodes by using tags. For more information about collecting inventory data from all of your managed nodes, see <a href="#">Inventory all managed nodes in your AWS account (p. 867)</a> .
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 data 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 Systems Manager console on the **Inventory** page, which includes several predefined cards to help you query the data.



### Note

Inventory cards automatically filter out Amazon EC2 managed instances with a state of *Terminated* and *Stopped*. For on-premises and AWS IoT Greengrass core device managed nodes, Inventory cards automatically filter out nodes with a state of *Terminated*.

If you create a resource data sync to synchronize and store all of your data in a single Amazon S3 bucket, then you can drill down into the data on the **Inventory Detailed View** page. For more information, see [Querying inventory data from multiple Regions and accounts \(p. 870\)](#).

### EventBridge support

This Systems Manager capability is supported as an *event* type in Amazon EventBridge rules. For information, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#) and [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#).

### Contents

- [Learn more about Systems Manager Inventory \(p. 851\)](#)
- [Setting up Systems Manager Inventory \(p. 858\)](#)
- [Configuring inventory collection \(p. 866\)](#)

- [Working with Systems Manager inventory data \(p. 870\)](#)
- [Working with custom inventory \(p. 885\)](#)
- [Viewing inventory history and change tracking \(p. 895\)](#)
- [Stopping data collection and deleting inventory data \(p. 896\)](#)
- [Systems Manager Inventory walkthroughs \(p. 897\)](#)
- [Troubleshooting problems with Systems Manager Inventory \(p. 909\)](#)

## Learn more about Systems Manager Inventory

When you configure AWS Systems Manager Inventory, you specify the type of metadata to collect, the managed nodes from where the metadata should be collected, and a schedule for metadata collection. These configurations are saved with your AWS account as an AWS Systems Manager State Manager association. An association is simply a configuration.

### Note

Inventory only collects metadata. It doesn't collect any personal or proprietary data.

### Topics

- [Metadata collected by inventory \(p. 851\)](#)
- [Working with file and Windows registry inventory \(p. 856\)](#)
- [Related AWS services \(p. 857\)](#)

## Metadata collected by inventory

The following sample shows the complete list of metadata collected by each AWS Systems Manager Inventory plugin.

```
{
 "typeName": "AWS:InstanceInformation",
 "version": "1.0",
 "attributes": [
 { "name": "AgentType", "dataType": "STRING"},
 { "name": "AgentVersion", "dataType": "STRING"},
 { "name": "ComputerName", "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": "AgentStatus", "dataType": "STRING"},
 { "name": "InstanceStatus", "dataType": "STRING"}
],
 {
 "typeName": "AWS:Application",
 "version": "1.1",
 "attributes": [
 { "name": "Name", "dataType": "STRING"},
 { "name": "ApplicationType", "dataType": "STRING"},
 { "name": "Publisher", "dataType": "STRING"},
 { "name": "Version", "dataType": "STRING"},
 { "name": "Release", "dataType": "STRING"},
 { "name": "Epoch", "dataType": "STRING"},
 { "name": "InstalledTime", "dataType": "STRING"},
 { "name": "Architecture", "dataType": "STRING"},
 { "name": "URL", "dataType": "STRING"}
]
 }
}
```

```

 { "name": "Summary", "dataType": "STRING"},

 { "name": "PackageId", "dataType": "STRING"}

]

},

{

 "typeName" : "AWS:File",

 "version": "1.0",

 "attributes": [

 { "name": "Name", "dataType": "STRING"},

 { "name": "Size", "dataType": "STRING"},

 { "name": "Description", "dataType": "STRING"},

 { "name": "FileVersion", "dataType": "STRING"},

 { "name": "InstalledDate", "dataType": "STRING"},

 { "name": "ModificationTime", "dataType": "STRING"},

 { "name": "LastAccessTime", "dataType": "STRING"},

 { "name": "ProductName", "dataType": "STRING"},

 { "name": "InstalledDir", "dataType": "STRING"},

 { "name": "ProductLanguage", "dataType": "STRING"},

 { "name": "CompanyName", "dataType": "STRING"},

 { "name": "ProductVersion", "dataType": "STRING"}

]

},

{

 "typeName" : "AWS:Process",

 "version": "1.0",

 "attributes": [

 { "name": "StartTime", "dataType": "STRING"},

 { "name": "CommandLine", "dataType": "STRING"},

 { "name": "User", "dataType": "STRING"},

 { "name": "FileName", "dataType": "STRING"},

 { "name": "FileVersion", "dataType": "STRING"},

 { "name": "FileDescription", "dataType": "STRING"},

 { "name": "FileSize", "dataType": "STRING"},

 { "name": "CompanyName", "dataType": "STRING"},

 { "name": "ProductName", "dataType": "STRING"},

 { "name": "ProductVersion", "dataType": "STRING"},

 { "name": "InstalledDate", "dataType": "STRING"},

 { "name": "InstalledDir", "dataType": "STRING"},

 { "name": "UsageId", "dataType": "STRING"}

]

},

{

 "typeName": "AWS:AWSComponent",

 "version": "1.0",

 "attributes": [

 { "name": "Name", "dataType": "STRING"},

 { "name": "ApplicationType", "dataType": "STRING"},

 { "name": "Publisher", "dataType": "STRING"},

 { "name": "Version", "dataType": "STRING"},

 { "name": "InstalledTime", "dataType": "STRING"},

 { "name": "Architecture", "dataType": "STRING"},

 { "name": "URL", "dataType": "STRING"}

]

},

{

 "typeName": "AWS:WindowsUpdate",

 "version": "1.0",

 "attributes": [

 { "name": "HotFixId", "dataType": "STRING"},

 { "name": "Description", "dataType": "STRING"},

 { "name": "InstalledTime", "dataType": "STRING"},

 { "name": "InstalledBy", "dataType": "STRING"}

]

},

{

 "typeName": "AWS:Network",

```

```

 "version": "1.0",
 "attributes": [
 { "name": "Name", "dataType": "STRING" },
 { "name": "SubnetMask", "dataType": "STRING" },
 { "name": "Gateway", "dataType": "STRING" },
 { "name": "DHCPServer", "dataType": "STRING" },
 { "name": "DNSServer", "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": "InstalledPendingRebootCount", "dataType": "NUMBER" },
 { "name": "InstalledOtherCount", "dataType": "NUMBER" },
 { "name": "InstalledRejectedCount", "dataType": "NUMBER" },
 { "name": "NotApplicableCount", "dataType": "NUMBER" },
 { "name": "UnreportedNotApplicableCount", "dataType": "NUMBER" },
 { "name": "MissingCount", "dataType": "NUMBER" },
 { "name": "FailedCount", "dataType": "NUMBER" },
 { "name": "OperationType", "dataType": "STRING" },
 { "name": "OperationStartTime", "dataType": "STRING" },
 { "name": "OperationEndTime", "dataType": "STRING" },
 { "name": "InstallOverrideList", "dataType": "STRING" },
 { "name": "RebootOption", "dataType": "STRING" },
 { "name": "LastNoRebootInstallOperationTime", "dataType": "STRING" },
 { "name": "ExecutionId", "dataType": "STRING" },
 "isOptional": "true",
 { "name": "NonCompliantSeverity", "dataType": "STRING" },
 "isOptional": "true",
 { "name": "SecurityNonCompliantCount", "dataType": "NUMBER" },
 "isOptional": "true",
 { "name": "CriticalNonCompliantCount", "dataType": "NUMBER" },
 "isOptional": "true",
 { "name": "OtherNonCompliantCount", "dataType": "NUMBER" },
 "isOptional": "true"
]
},
{
 "typeName": "AWS:PatchCompliance",
 "version": "1.0",
 "attributes": [
 { "name": "Title", "dataType": "STRING" },
 { "name": "KBId", "dataType": "STRING" },
 { "name": "Classification", "dataType": "STRING" },
 { "name": "Severity", "dataType": "STRING" },
 { "name": "State", "dataType": "STRING" },
 { "name": "InstalledTime", "dataType": "STRING" }
]
},
{
 "typeName": "AWS:ComplianceItem",
 "version": "1.0",
 "attributes": [
 { "name": "ComplianceType", "dataType": "STRING" },
 "isContext": "true",
 { "name": "ExecutionId", "dataType": "STRING" },
 "isContext": "true"
]
}

```

```

 {
 "name": "ExecutionType",
 "isContext": "true",
 {
 "name": "ExecutionTime",
 "isContext": "true",
 {
 "name": "Id",
 {
 "name": "Title",
 {
 "name": "Status",
 {
 "name": "Severity",
 {
 "name": "DocumentName",
 {
 "name": "DocumentVersion",
 {
 "name": "Classification",
 {
 "name": "PatchBaselineId",
 {
 "name": "PatchSeverity",
 {
 "name": "PatchState",
 {
 "name": "PatchGroup",
 {
 "name": "InstalledTime",
 {
 "name": "InstallOverrideList",
 "isOptional": "true",
 {
 "name": "DetailedText",
 "isOptional": "true",
 {
 "name": "DetailedLink",
 "isOptional": "true",
 {
 "name": "CVEIds",
 "isOptional": "true"
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 },
 {
 "typeName": "AWS:ComplianceSummary",
 "version": "1.0",
 "attributes": [
 {
 "name": "ComplianceType",
 {
 "name": "PatchGroup",
 {
 "name": "PatchBaselineId",
 {
 "name": "Status",
 {
 "name": "OverallSeverity",
 {
 "name": "ExecutionId",
 {
 "name": "ExecutionType",
 {
 "name": "ExecutionTime",
 {
 "name": "CompliantCriticalCount",
 {
 "name": "CompliantHighCount",
 {
 "name": "CompliantMediumCount",
 {
 "name": "CompliantLowCount",
 {
 "name": "CompliantInformationalCount",
 {
 "name": "CompliantUnspecifiedCount",
 {
 "name": "NonCompliantCriticalCount",
 {
 "name": "NonCompliantHighCount",
 {
 "name": "NonCompliantMediumCount",
 {
 "name": "NonCompliantLowCount",
 {
 "name": "NonCompliantInformationalCount",
 "name": "NonCompliantUnspecifiedCount",
 "dataType": "NUMBER"
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
 }
]
 },
 {
 "typeName": "AWS:InstanceDetailedInformation",
 "version": "1.0",
 "attributes": [
 {
 "name": "CPUModel",
 "dataType": "STRING",
 {
 "name": "CPUCores",
 "dataType": "NUMBER",
 {
 "name": "CPUs",
 "dataType": "NUMBER",
 {
 "name": "CPUSpeedMHz",
 "dataType": "NUMBER",
 {
 "name": "CPUSockets",
 "dataType": "NUMBER",
 {
 "name": "CPUHyperThreadEnabled",
 "dataType": "STRING",
 {
 "name": "OSServicePack",
 "dataType": "STRING"
 }
 }
 }
 }
 }
 }
]
 }
 }
 }
}

```

```

"typeName": "AWS:Service",
"version": "1.0",
"attributes": [
 { "name": "Name", "dataType": "STRING" },
 { "name": "DisplayName", "dataType": "STRING" },
 { "name": "ServiceType", "dataType": "STRING" },
 { "name": "Status", "dataType": "STRING" },
 { "name": "DependentServices", "dataType": "STRING" },
 { "name": "ServicesDependedOn", "dataType": "STRING" },
 { "name": "StartType", "dataType": "STRING" }
],
},
{
 "typeName": "AWS:WindowsRegistry",
 "version": "1.0",
 "attributes": [
 { "name": "KeyPath", "dataType": "STRING" },
 { "name": "ValueName", "dataType": "STRING" },
 { "name": "ValueType", "dataType": "STRING" },
 { "name": "Value", "dataType": "STRING" }
],
},
{
 "typeName": "AWS:WindowsRole",
 "version": "1.0",
 "attributes": [
 { "name": "Name", "dataType": "STRING" },
 { "name": "DisplayName", "dataType": "STRING" },
 { "name": "Path", "dataType": "STRING" },
 { "name": "FeatureType", "dataType": "STRING" },
 { "name": "DependsOn", "dataType": "STRING" },
 { "name": "Description", "dataType": "STRING" },
 { "name": "Installed", "dataType": "STRING" },
 { "name": "InstalledState", "dataType": "STRING" },
 { "name": "SubFeatures", "dataType": "STRING" },
 { "name": "ServerComponentDescriptor", "dataType": "STRING" },
 { "name": "Parent", "dataType": "STRING" }
],
},
{
 "typeName": "AWS:Tag",
 "version": "1.0",
 "attributes": [
 { "name": "Key", "dataType": "STRING" },
 { "name": "Value", "dataType": "STRING" }
],
},
{
 "typeName": "AWS:ResourceGroup",
 "version": "1.0",
 "attributes": [
 { "name": "Name", "dataType": "STRING" },
 { "name": "Arn", "dataType": "STRING" }
],
},
{
 "typeName": "AWS:BillingInfo",
 "version": "1.0",
 "attributes": [
 { "name": "BillingProductId", "dataType": "STRING" }
]
}
}

```

### Note

- For "typeName": "AWS:InstanceInformation", InstanceStatus can be one of the following: Active, ConnectionLost, Stopped, Terminated.
- 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, 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

AWS Systems Manager Inventory allows you to search and inventory files on Windows, Linux, and macOS 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 time 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.

```
{
 "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 managed nodes. 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 and macOS, collect metadata of .sh files in the /home/ec2-user directory, excluding all subdirectories.

```
[{"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.

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

- On Windows, collect metadata of specific log patterns.

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

- Limit the directory count when performing recursive collection.

```
[{"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.

```
{
 "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.

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

- Collect all keys and values for a specific path (recursive search turned off).

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

- Collect a specific key by using the **ValueNames** option.

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

## Related AWS services

AWS 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](#).

# Setting up Systems Manager Inventory

Before you use AWS Systems Manager Inventory to collect metadata about the applications, services, AWS components and more running on your managed nodes, we recommend that you configure resource data sync to centralize the storage of your inventory data in a single Amazon Simple Storage Service (Amazon S3) bucket. We also recommend that you configure Amazon EventBridge monitoring of inventory events. These processes make it easier to view and manage inventory data and collection.

## Topics

- [Configuring resource data sync for Inventory \(p. 858\)](#)
- [About EventBridge monitoring of Inventory events \(p. 865\)](#)

## Configuring resource data sync for Inventory

This topic describes how to set up and configure resource data sync for AWS 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 \(p. 167\)](#).

### About resource data sync

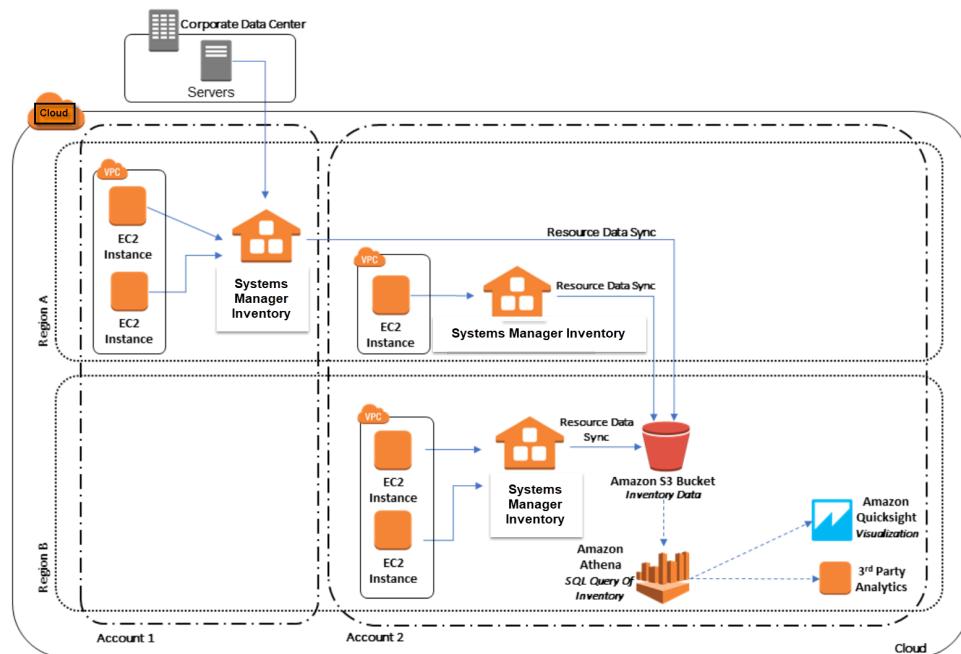
You can use Systems Manager resource data sync to send inventory data collected from all of your managed nodes to a single Amazon Simple Storage Service (Amazon S3) bucket. Resource data sync then automatically updates the centralized data when new inventory data is collected. With all inventory data stored in a target Amazon 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 nodes. Some of these nodes are located in a hybrid data center, and others are running in Amazon Elastic Compute Cloud (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 managed node, 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 nodes. After the sync is successfully created, Systems Manager creates a baseline of all inventory data and saves it in the target Amazon S3 bucket. When new inventory data is collected, Systems Manager automatically updates the data in the Amazon 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 nodes in Amazon EC2 and a hybrid environment to a target Amazon 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 node, resource data sync preserves the inventory file for the deleted node. For running nodes, however, resource data sync automatically overwrites old inventory files when new files are created and written to the Amazon 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 Amazon S3 bucket. For an example of how to create an encrypted sync by using the AWS Command Line Interface (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 \(p. 903\)](#).

## Before you begin

Before you create a resource data sync, use the following procedure to create a central Amazon S3 bucket to store aggregated inventory data. The procedure describes how to assign a bucket policy that allows Systems Manager to write inventory data to the bucket from multiple accounts. If you already have an Amazon 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.

#### Note

Systems Manager Inventory can't add data to a specified Amazon S3 bucket if that bucket is configured to use Object Lock. Verify that the Amazon S3 bucket you create or choose for resource data sync isn't configured to use Amazon S3 Object Lock. For more information, see [How Amazon S3 Object Lock works](#) in the *Amazon Simple Storage Service User Guide*.

## To create and configure an Amazon 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 User 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 **DOC-EXAMPLE-BUCKET** and **account-id** with the name of the S3 bucket you created and a valid AWS account ID.

To allow 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:::DOC-EXAMPLE-BUCKET/* accountId=123456789012/*",
 "arn:aws:s3:::DOC-EXAMPLE-BUCKET/* accountId=444455556666/*",
 "arn:aws:s3:::DOC-EXAMPLE-BUCKET/* accountId=777788889999/*"
],
```

### Note

For information about viewing your AWS account ID, see [Your Amazon Web Services Account ID and Its Alias](#) in the *IAM User Guide*.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "SSMBucketPermissionsCheck",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "s3:GetBucketAcl",
 "Resource": "arn:aws:s3:::S3_bucket_name"
 },
 {
 "Sid": "SSMBucketDelivery",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "s3:PutObject",
 "Resource": [
 "arn:aws:s3:::S3_bucket_name/* accountId=ID_number/*",
 "arn:aws:s3:::S3_bucket_name/* accountId=ID_number/*",
 "arn:aws:s3:::S3_bucket_name/* accountId=ID_number/*",
 "arn:aws:s3:::S3_bucket_name/* accountId=ID_number/*"
],
 "Condition": {
 "StringEquals": {
 "s3:x-amz-acl": "bucket-owner-full-control",
 "aws:SourceAccount": "123456789012"
 },
 "ArnLike": {
 "aws:SourceArn": "arn:aws:ssm::*:123456789012:resource-data-sync/*"
 }
 }
 }
]
}
```

**Note**

If you create a resource data sync for an AWS Region that came online since April 25, 2019 or later, you must enter a Region-specific service principal entry in the `SSMBucketDelivery` section. This requirement includes the following Regions:

- Asia Pacific (Hong Kong) Region (ap-east-1)
- Asia Pacific (Jakarta) Region (ap-southeast-3)
- Africa (Cape Town) Region (af-south-1)
- Europe (Milan) Region (eu-south-1)
- EU (Zaragoza) Region (eu-south-2)

The following example includes a Region-specific service principal entry for `ssm.ap-east-1.amazonaws.com`.

```
{
 "Sid": " SSMBucketDelivery",
 "Effect": "Allow",
 "Principal": {
 "Service": ["ssm.amazonaws.com", "ssm.ap-east-1.amazonaws.com"]
 },
```

## Create a resource data sync for Inventory

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 nodes for Inventory by using the CLI \(p. 899\)](#).

### To create a resource data sync

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. In the **Account management** menu, choose **Resource data sync**.
4. Choose **Create resource data sync**.
5. In the **Sync name** field, enter a name for the sync configuration.
6. In the **Bucket name** field, enter the name of the Amazon S3 bucket you created using the [To create and configure an Amazon S3 bucket for resource data sync](#) procedure.
7. (Optional) In the **Bucket prefix** field, enter the name of an Amazon S3 bucket prefix (subcategory).
8. In the **Bucket region** field, choose **This region** if the Amazon 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 enter the name of the Region.

**Note**

If the sync and the target Amazon S3 bucket are located in different regions, you might be subject to data transfer pricing. For more information, see [Amazon S3 Pricing](#).

9. (Optional) In the **KMS Key ARN** field, type or paste a KMS Key ARN to encrypt inventory data in Amazon S3.
10. 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 Amazon 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 Amazon S3 bucket.

Resource data sync

Sync name

TestSync

Sync name can be between 1 and 64 characters

Bucket name

Type a name of a bucket in S3.

MyCentralInventoryBucketForRDS

Bucket name can be between 3 and 63 characters. See [Amazon S3 naming convention](#).

Bucket prefix - optional

Type a prefix for the bucket that receives the output.

Bucket region

The region of a bucket in Amazon S3

This region (us-east-2)

Another region

US East (Ohio)

## Creating an inventory resource data sync for accounts defined in AWS Organizations

You can synchronize inventory data from AWS accounts defined in AWS Organizations to a central Amazon S3 bucket. After you complete the following procedures, inventory data is synchronized to *individual* Amazon S3 key prefixes in the central bucket. Each key prefix represents a different AWS account ID.

### Before you begin

Before you begin, verify that you set up and configured AWS accounts in AWS Organizations. For more information, see [in the AWS Organizations User Guide](#).

Also, be aware that you must create the organization-based resource data sync for each AWS Region and AWS account defined in AWS Organizations.

### Creating a central Amazon S3 bucket

Use the following procedure to create a central Amazon S3 bucket to store aggregated inventory data. The procedure describes how to assign a bucket policy that allows Systems Manager to write inventory data to the bucket from your AWS Organizations account ID. If you already have an Amazon 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 Amazon 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 User 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 `DOC-EXAMPLE-BUCKET` 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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "SSMBucketPermissionsCheck",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "s3:GetBucketAcl",
 "Resource": "arn:aws:s3:::S3_bucket_name"
 },
 {
 "Sid": "SSMBucketDelivery",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "s3:PutObject",
 "Resource": [
 "arn:aws:s3:::DOC-EXAMPLE-BUCKET/bucket-prefix/*/accountid=*"/*"
],
 "Condition": {
 "StringEquals": {
 "s3:x-amz-acl": "bucket-owner-full-control",
 "s3:RequestObjectTag/OrgId": "organization-id",
 "aws:SourceAccount": "123456789012"
 },
 "ArnLike": {
 "aws:SourceArn": "arn:aws:ssm::123456789012:resource-data-sync/*"
 }
 }
 },
 {
 "Sid": "SSMBucketDeliveryTagging",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "s3:PutObjectTagging",
 "Resource": [
 "arn:aws:s3:::DOC-EXAMPLE-BUCKET/bucket-prefix/*/accountid=*"/*"
]
 }
]
}
```

#### Note

If you create a resource data sync for an AWS Region that came online since April 25, 2019 or later, you must enter a Region-specific service principal entry in the

`SSMBucketDelivery` and `SSMBucketDeliveryTagging` sections. This requirement includes the following Regions:

- Asia Pacific (Hong Kong) Region (ap-east-1)
- Asia Pacific (Jakarta) Region (ap-southeast-3)
- Africa (Cape Town) Region (af-south-1)
- Europe (Milan) Region (eu-south-1)
- EU (Zaragoza) Region (eu-south-2)

The following example includes a Region-specific service principal entry for `ssm.ap-east-1.amazonaws.com`.

```
{
 "Sid": " SSMBucketDelivery",
 "Effect": "Allow",
 "Principal": {
 "Service": [
 "ssm.amazonaws.com",
 "ssm.ap-east-1.amazonaws.com"
]
 },
 ...
 "Sid": " SSMBucketDeliveryTagging",
 "Effect": "Allow",
 "Principal": {
 "Service": [
 "ssm.amazonaws.com",
 "ssm.ap-east-1.amazonaws.com"
]
 },
}
```

## Create an inventory resource data sync for accounts defined in AWS Organizations

The following procedure describes how to use the AWS CLI to create a resource data sync for accounts that are defined in AWS Organizations. You must use the AWS CLI to perform this task. You must also perform this procedure for each AWS Region and AWS account defined in AWS Organizations.

### To create a resource data sync for an account defined in AWS Organizations (AWS CLI)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to verify that you don't have any other resource data syncs. You can only have one organization-based resource data sync.

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

If the command returns another resource data sync, you must delete it or choose not to create a new one.

3. Run the following command to create a resource data sync for an account defined in AWS Organizations. For `DOC-EXAMPLE-BUCKET`, specify the name of the Amazon S3 bucket you created earlier in this topic. If you created a prefix (subdirectory) for your bucket, then specify this information for `prefix-name`.

```
aws ssm create-resource-data-sync --sync-name name --s3-destination "BucketName=DOC-EXAMPLE-BUCKET,Prefix=prefix-name,SyncFormat=JsonSerDe,Region=AWS Region, for example us-east-2,DestinationDataSharing={DestinationDataSharingType=Organization}"
```

4. Repeat Steps 2 and 3 for every AWS Region and AWS account where you want to synchronize data to the central Amazon S3 bucket.

## About EventBridge monitoring of Inventory events

You can configure a rule in Amazon EventBridge to create an event in response to AWS Systems Manager Inventory resource state changes. EventBridge supports events for the following Inventory state changes. All events are sent on a best effort basis.

**Custom inventory type deleted for a specific instance:** If a rule is configured to monitor for this event, EventBridge creates an event when a custom inventory type on a specific managed is deleted. EventBridge sends one event per node per custom inventory type. Here is a sample event pattern.

```
{
 "timestampMillis": 1610042981103,
 "source": "SSM",
 "account": "123456789012",
 "type": "INVENTORY_RESOURCE_STATE_CHANGE",
 "startTime": "Jan 7, 2021 6:09:41 PM",
 "resources": [
 {
 "arn": "arn:aws:ssm:us-east-1:123456789012:managed-instance/i-12345678"
 }
],
 "body": {
 "action-status": "succeeded",
 "action": "delete",
 "resource-type": "managed-instance",
 "resource-id": "i-12345678",
 "action-reason": "",
 "type-name": "Custom:MyCustomInventoryType"
 }
}
```

**Custom inventory type deleted event for all instances:** If a rule is configured to monitor for this event, EventBridge creates an event when a custom inventory type for all managed nodes is deleted. Here is a sample event pattern.

```
{
 "timestampMillis": 1610042904712,
 "source": "SSM",
 "account": "123456789012",
 "type": "INVENTORY_RESOURCE_STATE_CHANGE",
 "startTime": "Jan 7, 2021 6:08:24 PM",
 "resources": [
],
 "body": {
 "action-status": "succeeded",
 "action": "delete-summary",
 "resource-type": "managed-instance",
 "resource-id": "",
 "action-reason": "The delete for type name Custom:SomeCustomInventoryType was completed. The deletion summary is: {\\"totalCount\\":1,\\"remainingCount\\":0,\\"summaryItems\\":[\{\\"version\\\":\"1.1\",\\"count\\":1,\\"remainingCount\\":0\}]}",
 "type-name": "Custom:MyCustomInventoryType"
 }
}
```

```
}
```

**PutInventory call with old schema version event:** If a rule is configured to monitor for this event, EventBridge creates an event when a PutInventory call is made that uses a schema version that is lower than the current schema. This event applies to all inventory types. Here is a sample event pattern.

```
{
 "timestampMillis": 1610042629548,
 "source": "SSM",
 "account": "123456789012",
 "type": "INVENTORY_RESOURCE_STATE_CHANGE",
 "startTime": "Jan 7, 2021 6:03:49 PM",
 "resources": [
 {
 "arn": "arn:aws:ssm:us-east-1:123456789012:managed-instance/i-12345678"
 }
],
 "body": {
 "action-status": "failed",
 "action": "put",
 "resource-type": "managed-instance",
 "resource-id": "i-01f017c1b2efbe2bc",
 "action-reason": "The inventory item with type name Custom:MyCustomInventoryType was sent with a disabled schema version 1.0. You must send a version greater than 1.0",
 "type-name": "Custom:MyCustomInventoryType"
 }
}
```

For information about how to configure EventBridge to monitor for these events, see [Configuring EventBridge for Systems Manager events \(p. 1450\)](#).

## Configuring inventory collection

This section describes how to configure AWS Systems Manager Inventory collection on one or more managed nodes by using the Systems Manager console. For an example of how to configure inventory collection by using the AWS Command Line Interface (AWS CLI), see [Systems Manager Inventory walkthroughs \(p. 897\)](#).

When you configure inventory collection, you start by creating a AWS 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, AWS Systems Manager Run Command, the system returns the following error: The aws:softwareInventory plugin can only be invoked via ssm-associate.

### Note

Be aware of the following behavior if you create multiple inventory associations for a managed node:

- Each node can be assigned an inventory association that targets *all* nodes (--targets "Key=InstanceIds,Values=\*").
- Each node can also be assigned a specific association that uses either tag key/value pairs or an AWS resource group.
- If a node is assigned multiple inventory associations, the status shows *Skipped* for the association that hasn't run. The association that ran most recently displays the actual status of the inventory association.
- If a node is assigned multiple inventory associations and each uses a tag key/value pair, then those inventory associations fail to run on the node because of the tag conflict. The association still runs on nodes that don't have the tag key/value conflict.

## Before You Begin

Before you configure inventory collection, complete the following tasks.

- Update AWS Systems Manager SSM Agent on the nodes 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 [Walkthrough: Automatically update SSM Agent \(CLI\) \(p. 1089\)](#).
- Verify that your nodes meet Systems Manager prerequisites. For more information, see [Systems Manager prerequisites \(p. 13\)](#).
- For Microsoft Windows nodes, verify that your managed node is configured with Windows PowerShell 3.0 (or later). SSM Agent uses the `ConvertTo-Json` cmdlet in PowerShell to convert Windows update inventory data to the required format.
- (Optional) Create a resource data sync to centrally store inventory data in an Amazon S3 bucket. resource data sync then automatically updates the centralized data when new inventory data is collected. For more information, see [Configuring resource data sync for Inventory \(p. 858\)](#).
- (Optional) Create a JSON file to collect custom inventory. For more information, see [Working with custom inventory \(p. 885\)](#).

## Inventory all managed nodes in your AWS account

You can inventory all managed nodes 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 nodes in your AWS account. Managed nodes that already have an inventory association are skipped when the global inventory association is applied and runs. When a node is skipped, the detailed status message states `Overridden By Explicit Inventory Association`. Those nodes are skipped by the global association, but they will still report inventory when they run their assigned inventory association.
- Automatically adds new nodes created in your AWS account to the global inventory association.

### Note

- If a managed node is configured for the global inventory association, and you assign a specific association to that node, 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 nodes, see [Update SSM Agent by using Run Command \(p. 997\)](#).

## Configuring inventory collection with one click (console)

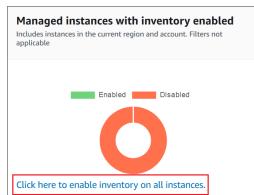
Use the following procedure to configure Systems Manager Inventory for all managed nodes in your AWS account and in a single AWS Region.

### To configure all of your managed nodes in the current Region for Systems Manager inventory

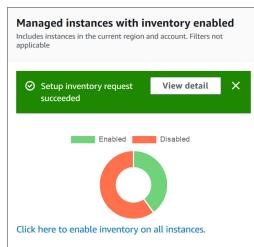
1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Inventory**.

-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**.



If successful, the console displays the following message.



Depending on the number of managed nodes 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 nodes.

## Configuring collection by using the console

This section includes information about how to configure Systems Manager Inventory to collect metadata from your managed nodes by using the Systems Manager console. You can quickly collect metadata from all nodes in a specific AWS account (and any future nodes that might be created in that account) or you can selectively collect inventory data by using tags or node IDs.

### To configure inventory collection

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 nodes 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 nodes for which there is no existing inventory association. If you choose this option, nodes 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 nodes in your AWS account \(p. 867\)](#).
  - **Specifying a tag** - Use this option to specify a single tag to identify nodes in your account from which you want to collect inventory. If you use a tag, any nodes created in the future with the same tag will also report inventory. If there is an existing inventory association with all nodes, using a tag to select specific nodes as a target for a different inventory overrides node

membership in the **All managed instances** target group. Managed nodes with the specified tag are skipped on future inventory collection from **All managed instances**.

- **Manually selecting instances** - Use this option to choose specific managed nodes in your account. Explicitly choosing specific nodes by using this option overrides inventory associations on the **All managed instances** target. The node is skipped on future inventory collection from **All managed instances**.

#### Note

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

5. In the **Schedule** section, choose how often you want the system to collect inventory metadata from your nodes.
6. In the **Parameters** section, use the lists to turn on or turn off 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 and macOS, collect metadata of .sh files in the /home/ec2-user directory, excluding all subdirectories.

```
[{"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.

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

- On Windows, collect metadata of specific log patterns.

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

- Limit the directory count when performing recursive collection.

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

### Windows registry

- Collect all keys and values recursively for a specific path.

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

- Collect all keys and values for a specific path (recursive search turned off).

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

- Collect a specific key by using the ValueNames option.

```
{"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 \(p. 856\)](#).

7. In the **Advanced** section, choose **Sync inventory execution logs to an Amazon S3 bucket** if you want to store the association execution status in an Amazon S3 bucket.

8. Choose **Setup Inventory**. Systems Manager creates a State Manager association and immediately runs Inventory on the nodes.
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 Amazon 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 node, then choose **Managed Instances** in the navigation pane.
10. Choose a node, and then choose **View details**.
11. On the node 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 AWS Systems Manager Inventory data.

### Topics

- [Querying inventory data from multiple Regions and accounts \(p. 870\)](#)
- [Querying an inventory collection by using filters \(p. 875\)](#)
- [Aggregating inventory data \(p. 876\)](#)

## 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 AWS accounts. Athena integration uses resource data sync so that you can view inventory data from all of your managed nodes on the **Detailed 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're 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 **Detailed 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 \(p. 858\)](#).

Also, be aware that the **Detailed View** page displays inventory data for the *owner* of the central Amazon S3 bucket used by resource data sync. If you aren't the owner of the central Amazon S3 bucket, then you won't see inventory data on the **Detailed View** page.

### Configuring access

Before you can query and view data from multiple accounts and Regions on the **Detailed View** page in the Systems Manager console, you must configure your AWS Identity and Access Management (IAM) user account with permission to view the data.

Optionally, if the inventory data is stored in an Amazon S3 bucket that uses AWS Key Management Service (AWS KMS) encryption, you must also configure your IAM account and the `AmazonGlueServiceRoleForSSM` service role for AWS KMS encryption. If you don't configure your IAM account and this role, Systems Manager displays `Cannot load Glue tables` when you choose the **Detailed View** tab in the console.

### Topics

- [Configuring your IAM user account to access the Detailed View page \(p. 871\)](#)
- [\(Optional\) Configure permissions for viewing AWS KMS encrypted data \(p. 873\)](#)

### Configuring your IAM user account to access the Detailed View page

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 **Detailed View** page.

#### To configure IAM user account access to the Detailed 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, 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": [
 {
 "Sid": "AllowGlue",
 "Effect": "Allow",
 "Action": [
 "glue:GetCrawler",
 "glue:GetCrawlers",
 "glue:GetTables",
 "glue:StartCrawler",
 "glue>CreateCrawler"
],
 "Resource": "*"
 },
 {
 "Sid": "iamPassRole",
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "iam:PassedToService": "glue.amazonaws.com"
 }
 }
 }
]
}
```

```
 },
 {
 "Sid": "iamRoleCreation",
 "Effect": "Allow",
 "Action": [
 "iam:CreateRole",
 "iam:AttachRolePolicy"
],
 "Resource": "arn:aws:iam::account_ID:role/*"
 },
 {
 "Sid": "iamPolicyCreation",
 "Effect": "Allow",
 "Action": "iam:CreatePolicy",
 "Resource": "arn:aws:iam::account_ID:policy/*"
 }
]
}
```

**Note**

(Optional) If the Amazon S3 bucket used to store inventory data is encrypted by using AWS KMS, you must also add the following block to the policy.

```
{
 "Effect": "Allow",
 "Action": [
 "kms:Decrypt"
],
 "Resource": [
 "arn:aws:kms:Region:account_ID:key/key_ARN"
]
}
```

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 allows AWS Glue to access the Amazon S3 bucket for resource data sync. Systems Manager automatically attaches the following policies to the role:

- **Amazon-GlueServicePolicyForSSM-{Amazon S3 bucket name}**: This policy allows communication between AWS Glue and Systems Manager Inventory.
- **AWSGlueServiceRole**: This is an AWS managed policy that allows access to AWS Glue.

If a policy with the name **Amazon-GlueServicePolicyForSSM-{Amazon S3 bucket name}** already exists in your IAM user account, and this policy isn't 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.

### (Optional) Configure permissions for viewing AWS KMS encrypted data

If the Amazon S3 bucket used to store inventory data is encrypted by using the AWS Key Management Service (AWS KMS), you must configure your IAM user account and the **Amazon-GlueServiceRoleForSSM** role with `kms:Decrypt` permissions for the AWS KMS key. If you don't configure your IAM account and this role, Systems Manager displays `Cannot load Glue tables` when you choose the **Detailed View** tab in the console.

#### Before you begin

To configure your IAM user account with `kms:Decrypt` permissions for the AWS KMS key, you add the following policy block to your IAM account as an inline policy, as described in the previous procedure ([Configuring your IAM user account to access the Detailed View page \(p. 871\)](#)).

```
{
 "Effect": "Allow",
 "Action": [
 "kms:Decrypt"
],
 "Resource": [
 "arn:aws:kms:Region:account_ID:key/key_ARN"
]
}
```

If you haven't done so already, complete that procedure and add `kms:Decrypt` permissions for the AWS KMS key.

Use the following procedure to configure the **Amazon-GlueServiceRoleForSSM** role with `kms:Decrypt` permissions for the AWS KMS key.

#### To configure the **Amazon-GlueServiceRoleForSSM** role with `kms:Decrypt` permissions

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**, and then use the search field to locate the **Amazon-GlueServiceRoleForSSM** role. The **Summary** page opens.
3. Use the search field to find the **Amazon-GlueServiceRoleForSSM** role. Choose the role name. The **Summary** page opens.
4. Choose the role name. The **Summary** page opens.
5. Choose **Add inline policy**. The **Create policy** page opens.
6. Choose the **JSON** tab.
7. 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": [
 "kms:Decrypt"
],
 "Resource": [
 "arn:aws:kms:Region:account_ID:key/key_ARN"
]
 }
]
}
```

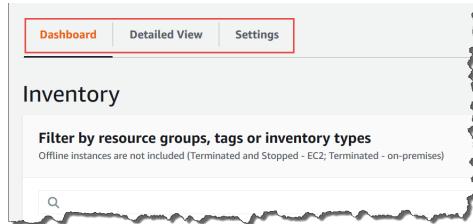
8. Choose **Review policy**
9. On the **Review Policy** page, enter a name in the **Name** field.
10. Choose **Create policy**.

## Querying data on the inventory detailed view page

Use the following procedure to view inventory data from multiple AWS Regions and AWS 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 aren't displayed on the Systems Manager Inventory page, it means Athena isn't available in the Region and you can't use the **Detailed View** to query data.



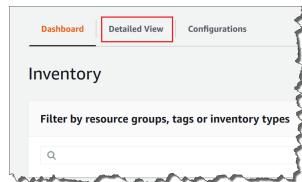
### To view inventory data from multiple Regions and accounts in the AWS Systems Manager console

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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.

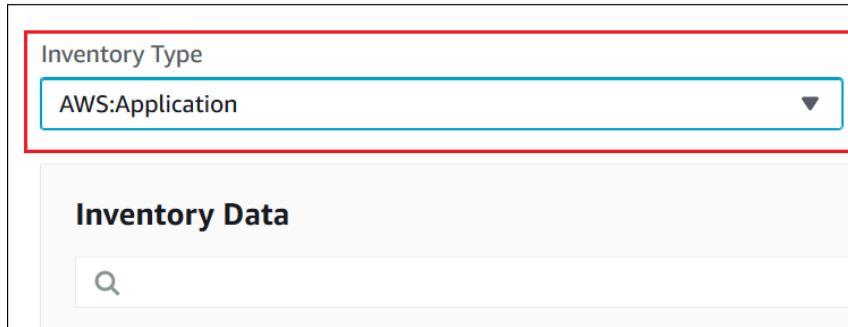
### Resource data syncs

Select a resource data sync to see the detailed data.

#### Inventory-Resource-Data-Sync

Sync created: Fri Jan 25 2019 14:17:12 GMT-0800 (Pacific Standard Time) Last status: Successful Last sync: Fri Jan 25 2019 14:21:39 GMT-0800 (Pacific Standard Time)

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.



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 Amazon S3 bucket twice daily, by default. If you frequently change the types of data to collect on your nodes then you might want to crawl the data more frequently, as described in the following procedure.

##### Important

AWS Glue charges your AWS 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

1. Open the AWS Glue console at <https://console.aws.amazon.com/glue/>.
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-*DOC-EXAMPLE-BUCKET-Region-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 isn't current.

## Querying an inventory collection by using filters

After you collect inventory data, you can use the filter capabilities in AWS Systems Manager to query a list of managed nodes that meet certain filter criteria.

## To query nodes based on inventory filters

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 is displayed.
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 is displayed in the filter.
6. Enter a value in the text box. For example, enter *Amazon* (SSM Agent is named *Amazon SSM Agent*).
7. Press **Enter**. The system returns a list of managed nodes 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 nodes for AWS Systems Manager Inventory, you can view aggregated counts of inventory data. For example, say you configured dozens or hundreds of managed nodes to collect the **AWS : Application** inventory type. By using the information in this section, you can see an exact count of how many nodes 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 nodes are running Windows, how many are running Linux, and how many are running macOS.

The procedures in this section describe how to view aggregated counts of inventory data by using the AWS Command Line Interface (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:

- If you terminate a managed node that is configured to collect inventory data, Systems Manager retains the inventory data for 30 days and then deletes it. For running nodes, the system deletes inventory data that is older than 30 days. If you need to store inventory data longer than 30 days, you can use AWS Config to record history or periodically query and upload the data to an Amazon Simple Storage Service (Amazon S3) bucket.
- If a node 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 node has been terminated and 30 days have passed.

For information about how to quickly configure and collect inventory data from all nodes in a specific AWS account (and any future nodes that might be created in that account), see [Configuring collection by using the console \(p. 868\)](#).

### Topics

- [Aggregating inventory data to see counts of nodes that collect specific types of data \(p. 877\)](#)

- [Aggregating inventory data with groups to see which nodes are and aren't configured to collect an inventory type \(p. 881\)](#)

## Aggregating inventory data to see counts of nodes that collect specific types of data

You can use the AWS Systems Manager [GetInventory API](#) operation to view aggregated counts of nodes that collect one or more inventory types and data types. For example, the `AWS:InstanceInformation` inventory type allows you to view an aggregate of operating systems by using the `GetInventory` API operation with the `AWS:InstanceInformation.PlatformType` data type. Here is an example AWS CLI command and output.

```
aws ssm get-inventory --aggregators "Expression=AWS:InstanceInformation.PlatformType"
```

The system returns information like the following.

```
{
 "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.

```
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": [
 {
 "Name": "Name",
 "Type": "String",
 "Required": true,
 "Description": "The name of the application."
 },
 {
 "Name": "Version",
 "Type": "String",
 "Required": true,
 "Description": "The version of the application."
 }
]
 }
]
}
```

```
 "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": "ServiceType"
 },
 {
 "DataType": "STRING",
 "Name": "Status"
 },
 {
 "DataType": "STRING",
 "Name": "StartType"
 }
]
},
```

```
{
 "TypeName": "AWS:WindowsRole",
 "Version": "1.0",
 "DisplayName": "WindowsRole",
 "Attributes": [
 {
 "Name": "Name",
 "Type": "String"
 },
 {
 "Name": "Installed",
 "Type": "String"
 },
 {
 "Name": "FeatureType",
 "Type": "String"
 }
]
}
```

You can aggregate data for any of the listed inventory types by creating a command that uses the following syntax.

```
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 nodes.

```
aws ssm get-inventory --aggregators "Expression=AWS:WindowsRole.Name"
```

### Example 2

This example aggregates a count of the applications installed on your nodes.

```
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 nodes. It also returns the specific name of the operating systems.

```
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 nodes and the specific version of each application.

```
aws ssm get-inventory --aggregators '[{"Expression": "AWS:Application.Name", "Aggregators": [{"Expression": "AWS:Application.Version"}]}]
```

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.

```
[
 {
 "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.

```
[
 {
 "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.

```
aws ssm get-inventory --aggregators file://file_name.json
```

The command returns information like the following.

```
{"Entities":
[
 {"Data":
 {"AWS:Application":
 {"Content":
 [
 {"Count": "3",
 "PlatformType": "linux",
 "Version": "2.6.5",
 "Name": "audit-libs"},
 {"Count": "2",
 "PlatformType": "windows",
 "Version": "2.6.5",
 "Name": "aws-lambda"},
 {"Count": "1",
 "PlatformType": "macos",
 "Version": "10.14",
 "Name": "aws-lambda"},
]
 }
 }
 }
]
```

```

 "Name": "audit-libs"},

 {"Count": "4",

 "PlatformType": "windows",

 "Version": "6.2.8",

 "Name": "microsoft office"},

 {"Count": "2",

 "PlatformType": "windows",

 "Version": "2.6.5",

 "Name": "chrome"},

 {"Count": "1",

 "PlatformType": "linux",

 "Version": "2.6.5",

 "Name": "chrome"},

 {"Count": "2",

 "PlatformType": "linux",

 "Version": "6.3",

 "Name": "authconfig"}

]

},

"ResourceType": "ManagedInstance"}

]
}

```

## Aggregating inventory data with groups to see which nodes are and aren't configured to collect an inventory type

Groups in Systems Manager Inventory allow you to quickly see a count of which managed nodes 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 nodes configured to collect the following inventory types:

- Node 1: AWS:Application
- Node 2: AWS:File
- Node 3: AWS:Application, AWS:File
- Node 4: AWS:Network

You can run the following command from the AWS CLI to see how many nodes are configured to collect both the AWS:Application and AWS:File inventory types. The response also returns a count of how many nodes aren't configured to collect both of these inventory types.

```
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 node is configured to collect both the AWS:Application and AWS:File inventory types.

```
{
 "Entities": [
 {
 "Data": {
 "ApplicationAndFile": {
 "Content": [
 {
 "notMatchingCount": "3"
 }
]
 }
 }
 }
]
}
```

```
 "matchingCount": "1"
 }
}
]
}
}
```

**Note**

Groups don't return data type counts. Also, you can't drill-down into the results to see the IDs of nodes 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:

```
{
 "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 nodes 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 nodes 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.

```
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.

```
{
 "Aggregators": [
 {
 "Groups": [
 {
 "Name": "ApplicationORFile",
 "Filters": [
 {
 "Key": "TypeName",
 "Values": [
 "AWS:Application",
 "AWS:File"
],
 "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": {
 "ApplicationORFile": {
 "Content": [
 {
 "notMatchingCount": "1"
 },
 {
 "matchingCount": "3"
 }
]
 }
 }
]
 }
}
```

### Example 2

This example aggregates a count of nodes 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.

```
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.

```
{
 "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 nodes by creating AWS Systems Manager Inventory *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 nodes. Currently, you store information about server rack location in a spreadsheet. With custom inventory, you can specify the rack location of each node as metadata on the node. 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 a node, you can either use the Systems Manager [PutInventory](#) API operation, as described in [Walkthrough: Assign custom inventory metadata to a managed node \(p. 898\)](#). Or, you can create a custom inventory JSON file and upload it to the node. 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.

```
{
 "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.

```
{
"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.

```
{
"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 aren't 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.

#### Note

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 node. The following table shows the location where custom inventory JSON files must be stored on the node.

Operating system	Path
Linux	/var/lib/amazon/ssm/ <i>node-id</i> /inventory/ custom

Operating system	Path
macOS	/opt/aws/ssm/data/ <i>node-id</i> /inventory/custom
Windows	%SystemDrive%\ProgramData\Amazon\SSM\InstanceData\ <i>node-id</i> \inventory\custom

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 operation to delete a custom inventory type and the data associated with that type. You call the delete-inventory command by using the AWS Command Line Interface (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 turns off the current version, deletes all data for it, and ignores all new data if the version is less than or equal to the turned off version. You can allow this inventory type again by calling the [PutInventory](#) action for a version greater than the turned off version.

### To delete or turn off custom inventory by using the AWS CLI

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

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.

```
{
 "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"
}

```

For information about how to understand the delete inventory summary, see [Understanding the delete inventory summary \(p. 891\)](#).

- Run the following command to delete all data for a custom inventory type.

```
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 hasn't 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.

```

{
 "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.

- Run the following command. The command performs the following actions for the current version of the inventory type: turns off the current version, deletes all data for it, and ignores all new data if the version is less than or equal to the turned off version.

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

The system returns information like the following.

```

{
 "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"
}
```

You can view a turned off inventory type by using the following command.

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

- Run the following command to delete an inventory type.

```
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.

```
{
 "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.

- Run the following command to view the status of a deletion operation. The system returned the deletion ID in the delete-inventory summary.

```
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.

```
{"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.

```
{"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.

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

```
{"InventoryDeletions": [
 {
 "DeletionId": "system_generated_deletion_ID",
 "DeletionStartTime": 1521682552,
 "DeletionSummary": {
 "RemainingCount": 0,
 "SummaryItems": [
 {
 "Count": 1,
 "RemainingCount": 0,
 "Version": "1.0"
 }
]
 }
 }
]}
```

```

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

```

## 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 nodes. 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

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

### RackSpace custom inventory 2

```
{
```

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

### RackSpace custom inventory 3

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

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

The user runs the following command to delete the Custom:RackSpace inventory.

**Note**

The output of this command doesn't show the deletion progress. For this reason, TotalCount and RemainingCount are always the same because the system hasn't deleted anything yet. You can use the describe-inventory-deletions command to show the deletion progress.

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

The system returns information like the following.

```
{
 "DeletionId": "1111-2222-333-444-7777777",
 "DeletionSummary": {
 "RemainingCount": 3, There are three items to delete
 "SummaryItems": [
 {
 "Count": 2, The system found two items that use SchemaVersion 1.0.
 "RemainingCount": 2,
 "Version": "1.0"
 },
 {
 "Count": 1, The system found one item that uses SchemaVersion 2.0.
 "RemainingCount": 1,
 "Version": "2.0"
 }
],
 "TotalCount": 3
 },
 "TypeName": "RackSpace"
}
```

## Viewing inventory delete actions in EventBridge

You can configure Amazon EventBridge to create an event anytime a user deletes custom inventory. EventBridge offers three types of events for custom inventory delete operations:

- **Delete action for an instance:** If the custom inventory for a specific managed node was successfully deleted or not.
- **Delete action summary:** A summary of the delete action.
- **Warning for turned off custom inventory type:** A warning event if a user called the [PutInventory](#) API operation for a custom inventory type version that was previously turned off.

Here are examples of each event.

### Delete action for an instance

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

### Delete action summary

```
{
 "version": "0",
 "id": "83898300-f576-5181-7a67-fb3e45e4fad4",
 "detail-type": "Inventory Resource State Change",
 "source": "aws.ssm",
 "account": "478678815555",
 "time": "2018-05-24T22:28:25Z",
 "region": "us-east-1",
 "resources": [

],
 "detail": {
 "action-status": "succeeded",
 "action": "delete-summary",
 "resource-type": "managed-instance",
 "resource-id": "",
 "action-reason": "The delete for type name Custom:MyInfo was completed. The deletion summary is: {\\"totalCount\\":2,\\"remainingCount\\":0,\\"summaryItems\\":[\{\\"version\\\":\"1.0\\",
 "\\"count\\":2,\\"remainingCount\\":0\}]}",
 "type-name": "Custom:MyInfo"
 }
}
```

### Warning for turned off custom inventory type

```
{
 "version": "0",
 "id": "49c1855c-9c57-b5d7-8518-b64aeeef5e4a",
 "detail-type": "Inventory Resource State Change",
 "source": "aws.ssm",
 "account": "478678815555",
 "time": "2018-05-24T22:46:58Z",
 "region": "us-east-1",
 "resources": [
 "arn:aws:ssm:us-east-1:478678815555:managed-instance/i-0ee2d86a2cf371f6"
],
 "detail": {
 "action-status": "failed",
 "action": "put",
 "resource-type": "managed-instance",
 "resource-id": "i-0ee2d86a2cf371f6",
 "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 an EventBridge 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 EventBridge for delete inventory operations

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**, and then choose **Create rule**.

-or-

- If the Amazon EventBridge home page opens first, choose **Create rule**.
3. Enter a name and description for the rule.

A rule can't have the same name as another rule in the same Region and on the same event bus.
  4. For **Define pattern**, choose **Event pattern**.
  5. Choose **Pre-defined pattern by service**.
  6. For **Service provider**, choose **AWS**.
  7. For **Service name**, choose **Systems Manager**.
  8. For **Event type**, choose **Inventory**.
  9. Verify that **Any detail type** is selected.
  10. For **Select event bus**, choose the event bus that you want to associate with this rule. If you want this rule to initiate on matching events that come from your own AWS account, select **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
  11. For **Target**, choose **SNS topic**, and then choose your topic from the **Topic** list.
  12. Expand **Configure input** and verify that **Matched event** is selected.
  13. (Optional) Enter one or more tags for the rule. For more information, see [Tagging Your Amazon EventBridge Resources](#) in the *Amazon EventBridge User Guide*.
  14. Choose **Create**.

## Viewing inventory history and change tracking

You can view AWS Systems Manager Inventory history and change tracking for all of your managed nodes 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 turn on the following resources in AWS Config:

- **SSM:ManagedInstanceInventory**
- **SSM:PatchCompliance**
- **SSM:AssociationCompliance**
- **SSM:FileData**

### Note

Note the following important details about Inventory history and change tracking:

- If you use AWS Config to track changes in your system, you must configure Systems Manager Inventory to collect **AWS:File** metadata so that you can view file changes in AWS Config (**SSM:FileData**). If you don't, then AWS Config doesn't track file changes on your system.
- By turning on **SSM:PatchCompliance** and **SSM:AssociationCompliance**, you can view Systems Manager Patch Manager patching and Systems Manager State Manager association compliance history and change tracking. For more information about compliance management for these resources, see [Working with Compliance \(p. 839\)](#).

The following procedure describes how to turn on inventory history and change-track recording in AWS Config by using the AWS Command Line Interface (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 `AWS_ConfigRole` managed policy to the IAM role that you assign to AWS Config. For more information about this role, see [AWS managed policy: AWS\\_ConfigRole in the AWS Config Developer Guide](#). For information about how to create an IAM role and assign the `AWS_ConfigRole` managed policy to that role, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.

### To turn on inventory history and change-track recording in AWS Config

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Copy and paste the following JSON sample into a simple text file and save it as `recordingGroup.json`.

```
{
 "allSupported":false,
 "includeGlobalResourceTypes":false,
 "resourceTypes": [
 "AWS::SSM::AssociationCompliance",
 "AWS::SSM::PatchCompliance",
 "AWS::SSM::ManagedInstanceInventory",
 "AWS::SSM::FileData"
]
}
```

3. Run the following command to load the `recordingGroup.json` file into AWS Config.

```
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.

```
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 node 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.

## Stopping data collection and deleting inventory data

If you no longer want to use AWS Systems Manager Inventory to view metadata about your AWS resources, you can stop data collection and delete data that has already been collected. This section includes the following information.

### Topics

- [Stopping data collection \(p. 896\)](#)
- [Deleting an Inventory resource data sync \(p. 897\)](#)

## Stopping data collection

When you initially configure Systems Manager to collect inventory data, the system creates a State Manager association that defines the schedule and the resources from which to collect

metadata. You can stop data collection by deleting any State Manager associations that use the AWS-GatherSoftwareInventory document.

#### To delete an Inventory association

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 an association that uses the AWS-GatherSoftwareInventory document and then choose **Delete**.
4. Repeat step three for any remaining associations that use the AWS-GatherSoftwareInventory document.

## Deleting an Inventory resource data sync

If you no longer want to use AWS Systems Manager Inventory to view metadata about your AWS resources, then we also recommend deleting resource data syncs used for inventory data collection.

#### To delete an Inventory resource data sync

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Resource Data Syncs**.
4. Choose a sync in the list.

#### Important

Make sure you choose the sync used for Inventory. Systems Manager supports resource data sync for multiple capabilities. If you choose the wrong sync, you could disrupt data aggregation for Systems Manager Explorer or Systems Manager Compliance.

5. Choose **Delete**.
6. Repeat these steps for any remaining resource data syncs you want to delete.
7. Delete the Amazon Simple Storage Service (Amazon S3) bucket where the data was stored. For information about deleting an Amazon S3 bucket, see [Deleting a bucket](#).

## Systems Manager Inventory walkthroughs

Use the following walkthroughs to collect and manage inventory data by using AWS Systems Manager Inventory. We recommend that you initially perform these walkthroughs with managed nodes in a test environment.

#### Before you begin

Before you start these walkthroughs, complete the following tasks:

- Update AWS Systems Manager SSM Agent on the nodes 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 [Walkthrough: Automatically update SSM Agent \(CLI\) \(p. 1089\)](#).

- Verify that your nodes meet Systems Manager prerequisites. For more information, see [Systems Manager prerequisites \(p. 13\)](#).
- (Optional) Create a JSON file to collect custom inventory. For more information, see [Working with custom inventory \(p. 885\)](#).

## Contents

- [Walkthrough: Assign custom inventory metadata to a managed node \(p. 898\)](#)
- [Walkthrough: Configure your managed nodes for Inventory by using the CLI \(p. 899\)](#)
- [Walkthrough: Use resource data sync to aggregate inventory data \(p. 903\)](#)

## Walkthrough: Assign custom inventory metadata to a managed node

The following procedure walks you through the process of using the AWS Systems Manager `PutInventory` API operation to assign custom inventory metadata to a managed node. This example assigns rack location information to a node. For more information about custom inventory, see [Working with custom inventory \(p. 885\)](#).

### To assign custom inventory metadata to a node

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to assign rack location information to a node.

#### Linux

```
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"}]'
```

#### Windows

```
aws ssm put-inventory --instance-id "ID" --items "TypeName=Custom:RackInfo,SchemaVersion=1.0,CaptureTime=2021-05-22T10:01:01Z,Content=[{RackLocation=B/Row C/Rack D/Shelf F'}]"
```

3. Run the following command to view custom inventory entries for this node.

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

The system responds with information like the following.

```
{
 "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.

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

The system responds with information like the following.

```
{
 "Schemas": [
 {
 "TypeName": "Custom:RackInfo",
 "Version": "1.0",
 "Attributes": [
 {
 "Name": "RackLocation",
 "Type": "STRING"
 }
]
 }
]
}
```

## Walkthrough: Configure your managed nodes for Inventory by using the CLI

The following procedures walk you through the process of configuring AWS Systems Manager Inventory to collect metadata from your managed nodes. 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, Systems Manager Run Command, the system returns the following error:

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

**Note**

A node can have only one inventory association configured at a time. If you configure a node with two or more inventory associations, the association doesn't run and no inventory data is collected.

### Quickly configure all of your managed nodes for Inventory (CLI)

You can quickly configure all managed nodes 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 nodes in your AWS account and in the current Region (CLI)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command.

## Linux & macOS

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

## Windows

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

### Note

This command doesn't allow Inventory to collect metadata for the Windows Registry or files. To inventory these datatypes, use the next procedure.

## Manually configuring Inventory on your managed nodes (CLI)

Use the following procedure to manually configure AWS Systems Manager Inventory on your managed nodes by using node IDs or tags.

### To manually configure your managed nodes for inventory (CLI)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to create a State Manager association that runs Systems Manager Inventory on the node. This command configures the service to run every six hours and to collect network configuration, Windows Update, and application metadata from a node.

## Linux & macOS

```
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, for example us-east-2\", \"OutputS3BucketName\": \"DOC-EXAMPLE-BUCKET\", \"OutputS3KeyPrefix \
\": \"Test\" } }" \
--parameters "networkConfig=Enabled,windowsUpdates=Enabled,applications=Enabled"
```

## Windows

```
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, for example us-east-2\", \"OutputS3BucketName\": \"DOC-EXAMPLE-BUCKET\", \"OutputS3KeyPrefix \
\": \"Test\" } }" ^
--parameters "networkConfig=Enabled,windowsUpdates=Enabled,applications=Enabled"
```

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-1a2b3c4d5e6f7g",
 "DocumentVersion": "$DEFAULT",
 "LastUpdateAssociationDate": 1480544990.06,
 "Date": 1480544990.06,
 "Targets": [
 {
 "Values": [
 "i-02573cafefEXAMPLE"
],
 "Key": "InstanceIds"
 }
]
 }
}
```

You can target large groups of nodes by using the Targets parameter with EC2 tags. See the following example.

#### Linux & macOS

```
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\": \"DOC-EXAMPLE-BUCKET\", \"OutputS3KeyPrefix\": \"Test \
\" } }" \
--parameters "networkConfig=Enabled,windowsUpdates=Enabled,applications=Enabled"
```

#### Windows

```
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\": \"DOC-EXAMPLE-BUCKET\",
 \"OutputS3KeyPrefix\": \"Test\" } }" ^
--parameters "networkConfig=Enabled,windowsUpdates=Enabled,applications=Enabled"
```

You can also inventory files and Windows Registry keys on a Windows Server node 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 \(p. 856\)](#).

#### Linux & macOS

```
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}]]"}' \
--profile dev-pdx
```

#### Windows

```
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}]]"}' ^
--profile dev-pdx
```

- 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-1a2b3c4d5e6f7g",
 "DocumentVersion": "1"
 }
]
}
```

## Walkthrough: Use resource data sync to aggregate inventory data

The following walkthrough describes how to create a resource data sync configuration for AWS Systems Manager Inventory by using the AWS Command Line Interface (AWS CLI). A resource data sync automatically ports inventory data from all of your managed nodes to a central Amazon Simple Storage Service (Amazon S3) bucket. The sync automatically updates the data in the central Amazon 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 Systems Manager in the AWS Management Console, see [Configuring resource data sync for Inventory \(p. 858\)](#). For information about querying inventory from multiple AWS Regions and accounts by using Systems Manager in the AWS Management Console, see [Querying inventory data from multiple Regions and accounts \(p. 870\)](#).

### Note

This walkthrough includes information about how to encrypt the sync by using AWS Key Management Service (AWS KMS). Inventory doesn't 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

Review or complete the following tasks before you begin the walkthrough in this section:

- Collect inventory data from your managed nodes. For the purpose of the Amazon Athena and Amazon QuickSight sections in this walkthrough, we recommend that you collect Application data. For more information about how to collect inventory data, see [Configuring inventory collection \(p. 866\)](#) or [Walkthrough: Configure your managed nodes for Inventory by using the CLI \(p. 899\)](#).
- (Optional) If the inventory data is stored in an Amazon Simple Storage Service (Amazon S3) bucket that uses AWS Key Management Service (AWS KMS) encryption, you must also configure your IAM account and the `AmazonGlueServiceRoleForSSM` service role for AWS KMS encryption. If you don't configure your IAM account and this role, Systems Manager displays `Cannot load Glue tables` when you choose the **Detailed View** tab in the console. For more information, see [\(Optional\) Configure permissions for viewing AWS KMS encrypted data \(p. 873\)](#).
- (Optional) If you want to encrypt the resource data 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": [
 {
 "Sid": "ssm-access-policy-statement",
 "Action": [
 "kms:GenerateDataKey"
],
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Resource": "arn:aws:kms:us-east-2:123456789012:key/KMS_key_id",
 "Condition": {
 "StringLike": {
 "aws:SourceAccount": "123456789012"
 },
 "ArnLike": {
 "aws:SourceAccount:123456789012"
 }
 }
 }
]
}
```

```

 "aws:SourceArn": "arn:aws:ssm:*:123456789012:resource-data-sync/*"
 }
}
]
```

### 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 User 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 **DOC-EXAMPLE-BUCKET** and **account-id** with the name of the Amazon 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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "SSMBucketDelivery",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "s3:PutObject",
 "Resource": [
 "arn:aws:s3:::DOC-EXAMPLE-BUCKET/bucket-prefix/*/accountid=account-id/*"
],
 "Condition": {
 "StringEquals": {
 "s3:x-amz-acl": "bucket-owner-full-control",
 "aws:SourceAccount": "account-id"
 },
 "ArnLike": {
 "aws:SourceArn": "arn:aws:ssm:*:account-id:resource-data-sync/*"
 }
 }
 }
]
}
```

5. (Optional) If you want to encrypt the sync, then you must add the following conditions to the policy listed in the previous step. Add these in the **StringEquals** section.

```
"s3:x-amz-server-side-encryption": "aws:kms",
"s3:x-amz-server-side-encryption-aws-kms-key-
id": "arn:aws:kms:region:account_ID:key/KMS_key_ID"
```

Here is an example:

```
"StringEquals": {
 "s3:x-amz-acl": "bucket-owner-full-control",
 "aws:SourceAccount": "account-id",
 "s3:x-amz-server-side-encryption": "aws:kms",
```

```
"s3:x-amz-server-side-encryption-aws-kms-key-
id":"arn:aws:kms:region:account_ID:key/KMS_key_ID"
}
```

6. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

7. (Optional) If you want to encrypt the sync, run the following command to verify that the bucket policy is enforcing the AWS KMS key requirement.

Linux & macOS

```
aws s3 cp ./A_file_in_the_bucket s3://DOC-EXAMPLE-BUCKET/prefix/ \
--sse aws:kms \
--sse-kms-key-id "arn:aws:kms:region:account_ID:key/KMS_key_id" \
--region region, for example, us-east-2
```

Windows

```
aws s3 cp ./A_file_in_the_bucket s3://DOC-EXAMPLE-BUCKET/prefix/ ^
--sse aws:kms ^
--sse-kms-key-id "arn:aws:kms:region:account_ID:key/KMS_key_id" ^
--region region, for example, us-east-2
```

8. Run the following command to create a resource data sync configuration with the Amazon S3 bucket you created at the start of this procedure. This command creates a sync from the AWS Region you're logged into.

#### Note

If the sync and the target Amazon S3 bucket are located in different regions, you might be subject to data transfer pricing. For more information, see [Amazon S3 Pricing](#).

Linux & macOS

```
aws ssm create-resource-data-sync \
--sync-name a_name \
--s3-destination "BucketName=DOC-EXAMPLE-BUCKET,Prefix=prefix_name, if_specified,SyncFormat=JsonSerDe,Region=bucket_region"
```

Windows

```
aws ssm create-resource-data-sync ^
--sync-name a_name ^
--s3-destination "BucketName=DOC-EXAMPLE-BUCKET,Prefix=prefix_name, if_specified,SyncFormat=JsonSerDe,Region=bucket_region"
```

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 Amazon S3 bucket in the `us-west-2` Region.

Linux & macOS

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

## Windows

```
aws ssm create-resource-data-sync ^
--sync-name InventoryDataWest ^
--s3-destination "BucketName=DOC-EXAMPLE-BUCKET,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 Amazon S3 bucket must be in the same Region.

## Linux & macOS

```
aws ssm create-resource-data-sync \
--sync-name sync_name \
--s3-destination "BucketName=DOC-EXAMPLE-BUCKET,Prefix=prefix_name,
if_specified,SyncFormat=JsonSerDe,AWSKMSKeyARN=arn:aws:kms:region:account_ID:key/KMS_key_ID,Region=bucket_region" \
--region region
```

## Windows

```
aws ssm create-resource-data-sync ^
--sync-name sync_name ^
--s3-destination "BucketName=DOC-EXAMPLE-BUCKET,Prefix=prefix_name,
if_specified,SyncFormat=JsonSerDe,AWSKMSKeyARN=arn:aws:kms:region:account_ID:key/KMS_key_ID,Region=bucket_region" ^
--region region
```

9. Run the following command to view the status of sync configuration.

```
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, examine the target bucket in Amazon S3. Inventory data should be displayed 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

1. Open the Athena console at <https://console.aws.amazon.com/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 **DOC-EXAMPLE-BUCKET** and **bucket\_prefix** with the name and prefix of the Amazon S3 target.

```
CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_Application (
 Name string,
 ResourceId 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://DOC-EXAMPLE-BUCKET/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 AWS accounts, then you must run this command again to update the partitions. You might also need to update your Amazon 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**.

```
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 Amazon Elastic Compute Cloud (Amazon EC2) instances for Linux, macOS, and Windows Server.

7. Individually copy and paste the following statements into the query editor, replace **DOC-EXAMPLE-BUCKET** and **bucket-prefix** with information for Amazon S3, and then choose **Run Query**. These statements set up additional inventory tables in Athena.

```
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://DOC-EXAMPLE-BUCKET/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://DOC-EXAMPLE-BUCKET/bucket-prefix/AWS:WindowsUpdate/'
```

```
MSCK REPAIR TABLE ssminventory.AWS_WindowsUpdate
```

```

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

```
MSCK REPAIR TABLE ssminventory.AWS_Network
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_PatchSummary (
 `ResourceId` string,
 `PatchGroup` string,
 `BaselineId` string,
 `SnapshotId` string,
 `OwnerInformation` string,
 `InstalledCount` int,
 `InstalledOtherCount` int,
 `NotApplicableCount` int,
 `MissingCount` int,
 `FailedCount` int,
 `OperationType` string,
 `OperationStartTime` string,
 `OperationEndTime` string
)
PARTITIONED BY (AccountId string, Region string, ResourceType string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES (
 'serialization.format' = '1'
) LOCATION 's3://DOC-EXAMPLE-BUCKET/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 AWS Systems Manager Inventory. If you're having trouble viewing your nodes in Systems Manager, see [Troubleshooting managed node availability \(p. 816\)](#).

### Topics

- [Multiple apply all associations with document 'AWS-GatherSoftwareInventory' are not supported \(p. 910\)](#)

- [Inventory execution status never exits pending \(p. 910\)](#)
- [The AWS-ListWindowsInventory document fails to run \(p. 911\)](#)
- [Console doesn't display Inventory Dashboard | Detailed View | Settings tabs \(p. 911\)](#)
- [UnsupportedAgent \(p. 911\)](#)
- [Skipped \(p. 911\)](#)
- [Failed \(p. 911\)](#)

## Multiple apply all associations with document 'AWS-GatherSoftwareInventory' are not supported

An error that **Multiple apply all associations with document 'AWS-GatherSoftwareInventory'** are not supported means that one or more AWS Regions where you're trying to configure an Inventory association for *all nodes* are already configured with an inventory association for all nodes. If necessary, you can delete the existing inventory association for all nodes and then create a new one. To view existing inventory associations, choose **State Manager** in the Systems Manager console and then locate associations that use the [AWS-GatherSoftwareInventory SSM document](#). If the existing inventory association for all nodes was created across multiple Regions, and you want to create a new one, you must delete the existing association from each Region where it exists.

## Inventory execution status never exits pending

There are two reasons why inventory collection never exits the `Pending` status:

- No nodes in the selected AWS Region:

If you create a global inventory association by using Systems Manager Quick Setup, the status of the inventory association ([AWS-GatherSoftwareInventory document](#)) shows `Pending` if there are no nodes available in the selected Region.

- Insufficient permissions:

An inventory association shows `Pending` if one or more nodes don't have permission to run Systems Manager Inventory. Verify that the AWS Identity and Access Management (IAM) instance profile includes the **AmazonSSMManagedInstanceCore** managed policy. For information about how to add this policy to an instance profile, see [Task 1: Add permissions to a Systems Manager instance profile \(console\) \(p. 23\)](#).

At a minimum, the instance profile must have the following IAM permissions.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:DescribeAssociation",
 "ssm>ListAssociations",
 "ssm>ListInstanceAssociations",
 "ssm:PutInventory",
 "ssm:PutComplianceItems",
 "ssm:UpdateAssociationStatus",
 "ssm:UpdateInstanceAssociationStatus",
 "ssm:UpdateInstanceInformation",
 "ssm:GetDocument",
 "ssm:DescribeDocument"
],
 "Resource": "*"
 }
]
}
```

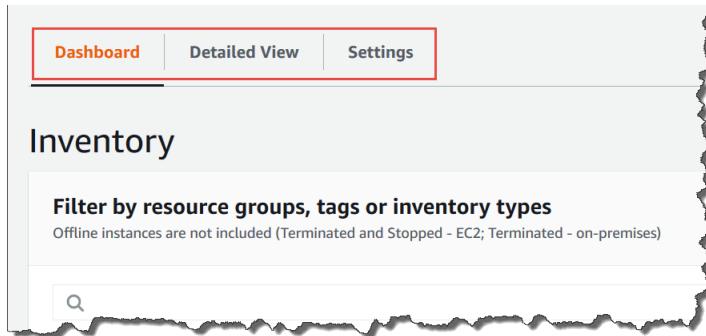
```
]
 }
}
```

## 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 \(p. 866\)](#).

## 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 aren't displayed on the Inventory page, it means Athena isn't 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 AWS Systems Manager SSM Agent on the managed node isn't correct. To create a global inventory association (to inventory all nodes 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 nodes on the **Managed Instances** page in the **Agent version** column. For information about how to update SSM Agent on your nodes, see [Update SSM Agent by using Run Command \(p. 997\)](#).

## Skipped

If the status of the inventory association for a node shows **Skipped**, this means that you created a *global inventory association* (to collect inventory from all nodes), but the skipped node already had an inventory association assigned to it. The global inventory association wasn't assigned to this node, and no inventory was collected by the global inventory association. However, the node will still report inventory data when the existing inventory association runs.

If you don't want the node to be skipped by the global inventory association, you must delete the existing inventory association. To view existing inventory associations, choose **State Manager** in the Systems Manager console and then locate associations that use the `AWS-GatherSoftwareInventory` SSM document.

## Failed

If the status of the inventory association for a node shows **Failed**, this could mean that the node has multiple inventory associations assigned to it. A node can only have one inventory association assigned

at a time. An inventory association uses the `AWS-GatherSoftwareInventory` AWS Systems Manager document (SSM document). You can run the following command by using the AWS Command Line Interface (AWS CLI) to view a list of associations for a node.

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

## AWS Systems Manager Hybrid Activations

To set up servers, edge devices, and virtual machines (VMs) in your hybrid environment as managed nodes, you create a managed-node hybrid activation. After you complete the activation, you receive an activation code and ID. This code and ID combination functions like an Amazon Elastic Compute Cloud (Amazon EC2) access ID and secret key to provide secure access to the AWS Systems Manager service from your managed nodes.

For information about configuring AWS IoT devices, non-AWS IoT devices, and on-premises servers and VMs as managed nodes, see [Setting up AWS Systems Manager for hybrid environments \(p. 34\)](#). For information about configuring AWS IoT Greengrass core devices for Systems Manager, see [Setting up AWS Systems Manager for edge devices \(p. 52\)](#).

**Note**

macOS isn't supported for Systems Manager hybrid environments.

### About Systems Manager instances tiers

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for servers, edge devices, and VMs in your hybrid environment. The standard-instances tier allows you to register a maximum of 1,000 machines per AWS account per AWS Region. If you need to register more than 1,000 machines in a single account and Region, then use the advanced-instances tier. You can create as many managed nodes as you like in the advanced-instances tier. All managed nodes configured for Systems Manager are priced on a pay-per-use basis. For more information about enabling the advanced instances tier, see [Turning on the advanced-instances tier \(p. 805\)](#). For more information about pricing, see [AWS Systems Manager Pricing](#).

**Note**

- Advanced instances also allow 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 \(p. 912\)](#).
- The standard-instances quota also applies to EC2 instances that use a Systems Manager on-premises activation (which isn't a common scenario).
- To patch applications released by Microsoft on virtual machines (VMs) on-premises instances, activate the advanced-instances tier. There is a charge to use the advanced-instances tier. There is no additional charge to patch applications released by Microsoft on Amazon Elastic Compute Cloud (Amazon EC2) instances. For more information, see [About patching applications released by Microsoft on Windows Server \(p. 1166\)](#).

## AWS Systems Manager Session Manager

Session Manager is a fully managed AWS Systems Manager capability. With Session Manager, you can manage your Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and on-premises servers and virtual machines (VMs). You can use either an interactive one-click browser-based shell or the AWS Command Line Interface (AWS CLI). Session Manager provides secure and auditable node

management without the need to open inbound ports, maintain bastion hosts, or manage SSH keys. Session Manager also allows you to comply with corporate policies that require controlled access to managed nodes, strict security practices, and fully auditable logs with node access details, while providing end users with simple one-click cross-platform access to your managed nodes. To get started with Session Manager, open the [Systems Manager console](#). In the navigation pane, choose **Session Manager**.

## How can Session Manager benefit my organization?

Session Manager offers these benefits:

- **Centralized access control to managed nodes using IAM policies**

Administrators have a single place to grant and revoke access to managed nodes. 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 managed nodes 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 managed nodes greatly increases the risk of entities running unauthorized or malicious commands on the managed nodes. 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 managed nodes 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 managed nodes 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 managed node to a local port on a client. After that, connect to the local port and access the server application that is running inside the node.

- **Cross-platform support for Windows, Linux, and macOS**

Session Manager provides support for Windows, Linux, and macOS from a single tool. For example, you don't need to use an SSH client for Linux and macOS managed nodes or an RDP connection for Windows Server managed nodes.

- **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 managed nodes 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 Amazon Simple Storage Service (Amazon S3) bucket you specify. One bucket is used for all CloudTrail logs for your account. For more information, see [Logging AWS Systems Manager API calls with AWS CloudTrail \(p. 1442\)](#).
- **Amazon Simple Storage Service** – You can choose to store session log data in an Amazon S3 bucket of your choice for debugging and troubleshooting purposes. Log data can be sent to your Amazon S3 bucket with or without encryption using your AWS KMS key. For more information, see [Logging session data using Amazon S3 \(console\) \(p. 979\)](#).
- **Amazon CloudWatch Logs** – CloudWatch Logs allows you to monitor, store, and access log files from various AWS services. You can send session log data to a CloudWatch Logs log group for debugging and troubleshooting purposes. Log data can be sent to your log group with or without

AWS KMS encryption using your KMS key. For more information, see [Logging session data using Amazon CloudWatch Logs \(console\) \(p. 981\)](#).

- **Amazon EventBridge and Amazon Simple Notification Service** – EventBridge allows you to set up rules to detect when changes happen to AWS resources that you specify. You can create a rule to detect when a user in your organization starts or stops a session, and then receive a notification through Amazon SNS (for example, a text or email message) about the event. You can also configure a CloudWatch event to initiate other responses. For more information, see [Monitoring session activity using Amazon EventBridge \(console\) \(p. 977\)](#).

**Note**

Logging isn't 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.

## 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 managed nodes, and reduce inbound node access.
- Information Security experts who want to monitor and track managed node access and activity, close down inbound ports on managed nodes, or allow connections to managed nodes that don't 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 Linux, macOS, and Windows Server managed nodes.
- Users who want to connect to a managed node with just one click from the browser or AWS CLI without having to provide SSH keys.

## What are the main features of Session Manager?

- **Support for Windows Server, Linux and macOS managed nodes**

Session Manager enables you to establish secure connections to your Amazon Elastic Compute Cloud (EC2) instances, edge devices, and on-premises servers and virtual machines (VMs). For a list of supported operating system types, see [Setting up Session Manager \(p. 916\)](#).

**Note**

Session Manager support for on-premises machines is provided for the advanced-instances tier only. For information, see [Turning on the advanced-instances tier \(p. 805\)](#).

- **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 is 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 \(p. 964\)](#).

The **Session Manager SDK** consists of libraries and sample code that allows application developers to build front-end applications, such as custom shells or self-service portals for internal users that natively use Session Manager to connect to managed nodes. 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 managed nodes and which nodes they can access. You can also provide temporary access to your managed nodes. 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 session activity \(p. 977\)](#) and [Logging session activity \(p. 978\)](#).

- **Configurable shell profiles**

Session Manager provides you with options to configure preferences within sessions. These customizable profiles allow you to define preferences such as shell preferences, environment variables, working directories, and running multiple commands when a session is started.

- **Customer key data encryption support**

You can configure Session Manager to encrypt the session data logs that you send to an Amazon Simple Storage Service (Amazon 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 managed nodes during your sessions. For information, see [Logging session activity \(p. 978\)](#) and [Configure session preferences \(p. 940\)](#).

- **AWS PrivateLink support for managed nodes without public IP addresses**

You can also set up VPC Endpoints for Systems Manager using AWS PrivateLink to further secure your sessions. AWS PrivateLink limits all network traffic between your managed nodes, Systems Manager, and Amazon EC2 to the Amazon network. For more information, see [\(Optional\) Create a VPC endpoint \(p. 28\)](#).

- **Tunneling**

In a session, use a Session-type AWS Systems Manager (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 a managed node.

- **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 a managed node.

## What is a session?

A session is a connection made to a managed node using Session Manager. Sessions are based on a secure bi-directional communication channel between the client (you) and the remote managed node that streams inputs and outputs for commands. Traffic between a client and a managed node is encrypted using TLS 1.2, and requests to create the connection are signed using Sigv4. This two-way communication allows interactive bash and PowerShell access to managed nodes. 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 a managed node, such as a failure that requires

troubleshooting or a directive to change a simple configuration option on a node. Using the AWS Systems Manager console, the Amazon EC2 console, or the AWS CLI, John starts a session connecting him to the managed node, runs commands on the node 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

- [Setting up Session Manager \(p. 916\)](#)
- [Working with Session Manager \(p. 964\)](#)
- [Auditing session activity \(p. 977\)](#)
- [Logging session activity \(p. 978\)](#)
- [Session document schema \(p. 981\)](#)
- [Troubleshooting Session Manager \(p. 987\)](#)

## Setting up Session Manager

Before you use AWS Systems Manager Session Manager to connect to the managed nodes in your account, complete the steps in the following topics.

#### Topics

- [Step 1: Complete Session Manager prerequisites \(p. 916\)](#)
- [Step 2: Verify or create an IAM role with Session Manager permissions \(p. 920\)](#)
- [Step 3: Control user session access to managed nodes \(p. 926\)](#)
- [Step 4: Configure session preferences \(p. 940\)](#)
- [Step 5: \(Optional\) Restrict access to commands in a session \(p. 952\)](#)
- [Step 6: \(Optional\) Use AWS PrivateLink to set up a VPC endpoint for Session Manager \(p. 958\)](#)
- [Step 7: \(Optional\) Turn on or turn off ssm-user account administrative permissions \(p. 958\)](#)
- [Step 8: \(Optional\) Allow and controlling permissions for SSH connections through Session Manager \(p. 961\)](#)

## Step 1: Complete Session Manager prerequisites

Before using Session Manager, make sure your environment meets the following requirements.

#### Session Manager prerequisites

Requirement	Description
Supported operating systems	Session Manager supports connecting to Amazon Elastic Compute Cloud (Amazon EC2) instances, in addition to servers or virtual machines (VMs) in your hybrid environment that use the <i>advanced-instances</i> tier.  Session Manager supports the following operating system versions:

Requirement	Description
	<p><b>Note</b> Session Manager supports EC2 instances, edge devices, and on-premises servers and virtual machines (VMs) in your hybrid environment that use the <i>advanced-instances</i> tier. For more information about advanced instances, see <a href="#">Turning on the advanced-instances tier (p. 805)</a>.</p> <p><b>Linux</b> Session Manager supports all the versions of Linux that are supported by AWS Systems Manager. For information, see <a href="#">Systems Manager prerequisites (p. 13)</a>.</p> <p><b>macOS</b> Session Manager supports all the versions of macOS that are supported by AWS Systems Manager. For information, see <a href="#">Systems Manager prerequisites (p. 13)</a>.</p> <p><b>Windows</b> Session Manager supports Windows Server 2012 through Windows Server 2019.</p> <p><b>Note</b> Microsoft Windows Server 2016 Nano isn't supported.</p>

Requirement	Description
SSM Agent	<p>At minimum, AWS Systems Manager SSM Agent version 2.3.68.0 or later must be installed on the managed nodes you want to connect to through sessions.</p> <p>To use the option to encrypt session data using a key created in AWS Key Management Service (AWS KMS), version 2.3.539.0 or later of SSM Agent must be installed on the managed node.</p> <p>To use shell profiles in a session, SSM Agent version 3.0.161.0 or later must be installed on the managed node.</p> <p>To start a Session Manager port forwarding or SSH session, SSM Agent version 3.0.222.0 or later must be installed on the managed node.</p> <p>To stream session data using Amazon CloudWatch Logs, SSM Agent version 3.0.284.0 or later must be installed on the managed node.</p> <p>For information about how to determine the version number running on an instance, see <a href="#">Checking the SSM Agent version number (p. 129)</a>. For information about manually installing or automatically updating SSM Agent, see <a href="#">Working with SSM Agent (p. 68)</a>.</p> <p><b>About the <code>ssm-user</code> account</b></p> <p>Starting with version 2.3.50.0 of SSM Agent, the agent creates a user account on the managed node, with root or administrator permissions, called <code>ssm-user</code>. (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, <code>ssm-user</code> is created the first time a session starts on the managed node.) Sessions are launched using the administrative credentials of this user account. For information about restricting administrative control for this account, see <a href="#">Turn off or turn on <code>ssm-user</code> account administrative permissions (p. 958)</a>.</p> <p><b><code>ssm-user</code> on Windows Server domain controllers</b></p> <p>Beginning with SSM Agent version 2.3.612.0, the <code>ssm-user</code> account isn't created automatically on managed nodes 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 <code>ssm-user</code> account manually if it isn't already present, and assign Domain Administrator permissions to the user. On Windows Server, SSM Agent sets a new password</p>

Requirement	Description
	for the <code>ssm-user</code> account each time a session starts, so you don't need to specify a password when you create the account.
Connectivity to endpoints	<p>The managed nodes you connect to must also allow HTTPS (port 443) outbound traffic to the following endpoints:</p> <ul style="list-style-type: none"> <li>• <code>ec2messages.region.amazonaws.com</code></li> <li>• <code>ssm.region.amazonaws.com</code></li> <li>• <code>ssmmessages.region.amazonaws.com</code></li> </ul> <p>Alternatively, you can connect to the required endpoints by using interface endpoints. For more information, see <a href="#">Step 6: (Optional) Use AWS PrivateLink to set up a VPC endpoint for Session Manager (p. 958)</a>.</p>
AWS CLI	<p>(Optional) If you use the AWS Command Line Interface (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.</p> <p>You can call <code>aws --version</code> to check the version.</p> <p>If you need to install or upgrade the CLI, see <a href="#">Installing the AWS Command Line Interface</a> in the AWS Command Line Interface User Guide.</p> <p><b>Important</b>  An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see <a href="#">Automating updates to SSM Agent (p. 135)</a>. Subscribe to the <a href="#">SSM Agent Release Notes</a> page on GitHub to get notifications about SSM Agent updates.</p> <p>In addition, to use the CLI to manage your nodes with Session Manager, you must first install the Session Manager plugin on your local machine. For information, see <a href="#">(Optional) Install the Session Manager plugin for the AWS CLI (p. 964)</a>.</p>

Requirement	Description
Turn on advanced-instances tier (hybrid environments)	To connect to on-premises machines using Session Manager, you must turn on the advanced-instances tier in the AWS account and AWS Region where you create hybrid activations to register on-premises machines as managed instances. For more information about the advanced-instance tier, see <a href="#">Turning on the advanced-instances tier (p. 805)</a> .
Verify IAM service role permissions (hybrid environments)	<p>Hybrid instances use the AWS Identity and Access Management (IAM) service role specified in the hybrid activation to communicate with Systems Manager API operations. This service role must contain the permissions required to connect to your on-premises machines using Session Manager. If your service role contains the AWS managed policy <code>AmazonSSMManagedInstanceCore</code>, the required permissions for Session Manager are already provided.</p> <p>If you find that the service role does not contain the required permissions, you must deregister the managed instance and register it with a new hybrid activation that uses an IAM service role with the required permissions. For more information about deregistering managed instances, see <a href="#">Deregistering managed nodes in a hybrid environment (p. 815)</a>. For more information about creating IAM policies with Session Manager permissions, see <a href="#">Verify or create an IAM role with Session Manager permissions</a>.</p>

## Step 2: Verify or create an IAM role 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 AWS Identity and Access Management (IAM). For Amazon Elastic Compute Cloud (Amazon EC2) instances, permissions are provided by an instance profile. An instance profile passes an IAM role to an Amazon EC2 instance. You can attach an IAM instance profile to an Amazon EC2 instance as you launch it or to a previously launched instance. For more information, see [Using instance profiles](#).

For on-premises servers or virtual machines (VMs), permissions are provided by the IAM service role associated with the hybrid activation used to register your on-premises servers and VMs with Systems Manager. On-premises servers and VMs do not use instance profiles.

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 Amazon EC2 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. This is also true if the IAM service role used in your hybrid activation contains the `AmazonSSMManagedInstanceCore` managed policy.

### Important

You can't change the IAM service role associated with a hybrid activation. If you find that the service role does not contain the required permissions, you must deregister the managed instance and register it with a new hybrid activation that uses a service role with the required permissions. For more information about deregistering managed instances, see [Deregistering managed nodes in a hybrid environment \(p. 815\)](#). For more information about creating an IAM service role for on-premises machines, see [Create an IAM service role for a hybrid environment](#).

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 Simple Storage Service (Amazon S3) encryption or AWS Key Management Service (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 IAM role**

To add permissions for Session Manager actions to an existing IAM role that doesn't rely on the AWS-provided default policy `AmazonSSMManagedInstanceCore`, follow the steps in [Adding Session Manager permissions to an existing IAM role \(p. 921\)](#).

- **Create a custom IAM role with Session Manager permissions only**

To create an IAM role that contains permissions only for Session Manager actions, follow the steps in [Create a custom IAM role for Session Manager \(p. 922\)](#).

- **Create and use a new IAM role with permissions for all Systems Manager actions**

To create an IAM role 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 \(p. 21\)](#).

### Topics

- [Adding Session Manager permissions to an existing IAM role \(p. 921\)](#)
- [Create a custom IAM role for Session Manager \(p. 922\)](#)

## Adding Session Manager permissions to an existing IAM role

Follow these steps to embed Session Manager permissions in an existing AWS Identity and Access Management (IAM) role that doesn't rely on the AWS-provided default policy `AmazonSSMManagedInstanceCore` for instance permissions. This procedure assumes that your existing role already includes other Systems Manager `ssm` permissions for actions you want to allow access to. This policy alone isn't enough to use Session Manager.

### To add Session Manager permissions to an existing role (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. Choose **Add inline policy**. The link is located on the right side of the page.
6. Choose the **JSON** tab.
7. Replace the default content with the following policy. Replace `key-name` with the Amazon Resource Name (ARN) of the KMS key you want to use.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssmmessages:CreateControlChannel",
 "ssmmessages:CreateDataChannel",
 "ssmmessages:OpenControlChannel",
 "ssmmessages:OpenDataChannel"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "s3:GetEncryptionConfiguration"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "kms:Decrypt"
],
 "Resource": "key-name"
 }
]
```

For information about using a KMS key to encrypt session data, see [Turn on KMS key encryption of session data \(console\) \(p. 945\)](#).

If you won't 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"
 }
}'
```

8. Choose **Next: Tags**.
9. (Optional) Add tags by choosing **Add tag**, and entering the preferred tags for the policy.
10. Choose **Next: Review**.
11. On the **Review policy** page, for **Name**, enter a name for the inline policy, such as **SessionManagerPermissions**.
12. (Optional) For **Description**, enter a description for the policy.

Choose **Create policy**.

For information about the `ssmmessages` actions, see [Reference: ec2messages, ssmmessages, and other API operations \(p. 1546\)](#).

## Create a custom IAM role for Session Manager

You can create a custom AWS Identity and Access Management (IAM) role that provides permissions for only Session Manager actions on your instances. You can also include a policy to provide the permissions

needed for session logs to be sent to Amazon Simple Storage Service (Amazon S3) and Amazon CloudWatch Logs.

After you create the IAM role, see [Attaching an IAM Role to an Instance](#) and [Attach or Replace an Instance Profile](#) for information about how to attach the role 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*. For more information about creating an IAM service role for on-premises machines, see [Create an IAM service role for a hybrid environment](#).

### Topics

- [Creating an IAM role with minimal Session Manager permissions \(console\) \(p. 923\)](#)
- [Creating an IAM role with permissions for Session Manager and Amazon S3 and CloudWatch Logs \(console\) \(p. 924\)](#)

#### Creating an IAM role with minimal Session Manager permissions (console)

Use the following procedure to create a custom IAM role 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 is displayed, choose it, and then choose **Create Policy**.)
3. Choose the **JSON** tab.
4. Replace the default content with the following policy. Replace **key-name** with the Amazon Resource Name (ARN) of the KMS key you want to use.

```
{
 "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"
 }
]
}
```

For information about using a KMS key to encrypt session data, see [Turn on KMS key encryption of session data \(console\) \(p. 945\)](#).

If you won't 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"
 }
```

5. Choose **Next: Tags**.
6. (Optional) Add tags by choosing **Add tag**, and entering the preferred tags for the policy.
7. Choose **Next: Review**.
8. On the **Review policy** page, for **Name**, enter a name for the inline policy, such as **SessionManagerPermissions**.
9. (Optional) For **Description**, enter a description for the policy.
10. Choose **Create policy**.
11. In the navigation pane, choose **Roles**, and then choose **Create role**.
12. On the **Create role** page, choose **AWS service**, and for **Use case**, choose **EC2**.
13. Choose **Next**.
14. On the **Add permissions** page, select the check box to the left of name of the policy you just created, such as **SessionManagerPermissions**.
15. Choose **Next**.
16. On the **Name, review, and create** page, for **Role name**, enter a name for the IAM role, such as **MySessionManagerRole**.
17. (Optional) For **Role description**, enter a description for the instance profile.
18. (Optional) Add tags by choosing **Add tag**, and entering the preferred tags for the role.

Choose **Create role**.

For information about `ssmmessages` actions, see [Reference: ec2messages, ssmmessages, and other API operations \(p. 1546\)](#).

### [Creating an IAM role with permissions for Session Manager and Amazon S3 and CloudWatch Logs \(console\)](#)

Use the following procedure to create a custom IAM role 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 Amazon Simple Storage Service (Amazon S3) buckets and Amazon CloudWatch Logs log groups.

#### **Important**

To output session logs to an Amazon S3 bucket owned by a different AWS account, you must add the IAM `s3:PutObjectAcl` permission to the policy. If this permission isn't added, the account that owns the Amazon S3 bucket can't access the session output logs.

For information about specifying preferences for storing session logs, see [Logging session activity \(p. 978\)](#).

## To create an IAM role 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 is displayed, choose it, and then choose **Create Policy**.)
3. Choose the **JSON** tab.
4. Replace the default content with the following policy. Replace each *example resource placeholder* with your own information.

```
{
 "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:::DOC-EXAMPLE-BUCKET/s3-bucket-prefix/*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "s3:GetEncryptionConfiguration"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "kms:Decrypt"
],
 "Resource": "key-name"
 },
 {
 "Effect": "Allow",
 "Action": "kms:GenerateDataKey",
 "Resource": "*"
 }
]
}
```

}

5. Choose **Next: Tags**.
6. (Optional) Add tags by choosing **Add tag**, and entering the preferred tags for the policy.
7. Choose **Next: Review**.
8. On the **Review policy** page, for **Name**, enter a name for the inline policy, such as **SessionManagerPermissions**.
9. (Optional) For **Description**, enter a description for the policy.
10. Choose **Create policy**.
11. In the navigation pane, choose **Roles**, and then choose **Create role**.
12. On the **Create role** page, choose **AWS service**, and for **Use case**, choose **EC2**.
13. Choose **Next**.
14. On the **Add permissions** page, select the check box to the left of name of the policy you just created, such as **SessionManagerPermissions**.
15. Choose **Next**.
16. On the **Name, review, and create** page, for **Role name**, enter a name for the IAM role, such as **MySessionManagerRole**.
17. (Optional) For **Role description**, enter a description for the role.
18. (Optional) Add tags by choosing **Add tag**, and entering the preferred tags for the role.
19. Choose **Create role**.

## Step 3: Control user session access to managed nodes

AWS Systems Manager Session Manager allows you to centrally grant and revoke user access to managed nodes. Using AWS Identity and Access Management (IAM) policies, you control which managed nodes specific users or groups can connect to, and you control what Session Manager API operations they can perform on the managed nodes they're 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 \(p. 927\)](#)
- [Quickstart default IAM policies for Session Manager \(p. 928\)](#)

- [Additional sample IAM policies for Session Manager \(p. 935\)](#)

## Enforce a session document permission check for the AWS CLI

When you configure Session Manager for your account, the system creates a Session type document `SSM-SessionManagerRunShell`. This document stores your session preferences, such as whether session data is saved in an Amazon Simple Storage Service (Amazon S3) bucket or Amazon CloudWatch Logs log group, whether session data is encrypted using AWS Key Management Service (AWS KMS), and whether Run As support is allowed for your sessions. The following is an example.

```
{
 "schemaVersion": "1.0",
 "description": "Document to hold regional settings for Session Manager",
 "sessionType": "Standard_Stream",
 "inputs": {
 "s3BucketName": "doc-example-bucket",
 "s3KeyPrefix": "BucketPrefix",
 "s3EncryptionEnabled": true,
 "cloudWatchLogGroupName": "LogGroupName",
 "cloudWatchEncryptionEnabled": true,
 "kmsKeyId": "kms-key",
 "runAsEnabled": true,
 "runAsDefaultUser": "RunAsUser"
 }
}
```

By default, if a user in your account was granted permission in their AWS Identity and Access Management (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 don't 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.

```
aws ssm start-session \
 --target i-02573cafccEXAMPLE
```

The following example specifies the default Session Manager Session document.

```
aws ssm start-session \
 --document-name SSM-SessionManagerRunShell \
 --target i-02573cafccEXAMPLE
```

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.

```
"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. To ensure the condition element is enforced, it must be included in all policy statements which allow the `ssm:StartSession` action. The following is an example.

```
{
 "Effect": "Allow",
 "Action": [
 "ssm:StartSession"
],
 "Resource": [
 "arn:aws:ec2:us-west-2:123456789012:instance/i-02573cafccEXAMPLE",
 "arn:aws:ssm:us-west-2:123456789012:document/SSM-SessionManagerRunShell"
]
}
```

If the `SessionDocumentAccessCheck` condition element is set to `false`, it will not affect the evaluation of the IAM policy. That means if the `SessionDocumentAccessCheck` condition element is set to `false`, and you specify a document name in the `Resource`, you must provide the specified document name when starting a session. If you provide a different document name when starting a session, the request fails.

If the `SessionDocumentAccessCheck` condition element is set to `false`, and a document name is not specified in the `Resource`, you do not need to provide a document name when you start a session. By default, the `SSM-SessionManagerRunShell` document is used in the request.

For an example of specifying a Session Manager Session document in an IAM policy, see [Quickstart end user policies for Session Manager \(p. 929\)](#).

### 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 managed node and the user's local machine. For information, see [\(Optional\) Enable and control permissions for SSH connections through Session Manager \(p. 961\)](#).

## Quickstart default IAM policies for Session Manager

Use the samples in this section to help you create AWS Identity and Access Management (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 KMS key. 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 \(p. 929\)](#)
- [Quickstart administrator policy for Session Manager \(p. 932\)](#)

## 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 Command Line Interface (AWS CLI), from only the Amazon Elastic Compute Cloud (Amazon EC2) console, or from all three.

These policies provide end users the ability to start a session to a particular managed node and the ability to end only their own sessions. Refer to [Additional sample IAM policies for Session Manager \(p. 935\)](#) for examples of customizations you might want to make to the policy.

In the following sample policies, replace each *example resource placeholder* with your own information.

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 give users the ability to start and resume 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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:StartSession"
],
 "Resource": [
 "arn:aws:ec2:region:account-id:instance/instance-id",
 "arn:aws:ssm:region:account-id:document/SSM-SessionManagerRunShell" 1
],
 "Condition": {
 "BoolIfExists": {
 "ssm:SessionDocumentAccessCheck": "true" 2
 }
 },
 "Resource": "arn:aws:ssm:*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:DescribeSessions",
 "ssm:GetConnectionStatus",
 "ssm:DescribeInstanceProperties",
 "ec2:DescribeInstances"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession",
 "ssm:ResumeSession"
],
 "Resource": [
 "arn:aws:ssm:*:*:session/${aws:username}-*"
]
 },
]
}
```

```
{
 "Effect": "Allow",
 "Action": [
 "kms:GenerateDataKey" ③
],
 "Resource": "key-name"
}
]
```

### Amazon EC2

Use this sample policy to give users the ability to start and resume 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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:StartSession",
 "ssm:SendCommand" ④
],
 "Resource": [
 "arn:aws:ec2:region:account-id:instance/instance-id"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetConnectionStatus",
 "ssm:DescribeInstanceInformation"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession",
 "ssm:ResumeSession"
],
 "Resource": [
 "arn:aws:ssm:*::*:session/${aws:username}-*"
]
 }
]
}
```

### Session Manager, CLI, and Amazon EC2

Use this sample policy to give users the ability to start and resume sessions from the Session Manager console, the AWS CLI, and the Amazon EC2 console.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:StartSession",

```

```

 "ssm:SendCommand" ④
],
 "Resource": [
 "arn:aws:ec2:region:account-id:instance/instance-id",
 "arn:aws:ssm:region:account-id:document/SSM-SessionManagerRunShell" ①
],
 "Condition": {
 "BoolIfExists": {
 "ssm:SessionDocumentAccessCheck": "true" ②
 }
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:DescribeSessions",
 "ssm:GetConnectionStatus",
 "ssm:DescribeInstanceInformation",
 "ssm:DescribeInstanceProperties",
 "ec2:DescribeInstances"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession",
 "ssm:ResumeSession"
],
 "Resource": [
 "arn:aws:ssm:*:*:session/${aws:username}-*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "kms:GenerateDataKey" ③
],
 "Resource": "key-name"
 }
]
}

```

<sup>1</sup> 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 and control permissions for SSH connections through Session Manager \(p. 961\)](#).

<sup>2</sup> 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 \(p. 927\)](#).

<sup>3</sup> 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 Amazon Resource Name (ARN) of the KMS key 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 KMS key encryption for your session data, remove the following content from the policy.

```
'
 {
 "Effect": "Allow",
 "Action": [
 "kms:GenerateDataKey"
],
 "Resource": "key-name"
 }
'
```

For information about using AWS KMS for encrypting session data, see [Turn on KMS key encryption of session data \(console\) \(p. 945\)](#).

<sup>4</sup> 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 managed nodes 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 \(p. 935\)](#) 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.

In the following sample policies, replace each *example resource placeholder* with your own information.

Refer to the following sections to view sample policies for the three permissions scenarios.

#### Session Manager and CLI

Use this sample policy to give administrators 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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:StartSession"
],
 "Resource": [
 "arn:aws:ec2:region:account-id: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"
],
 "Resource": "arn:aws:ssm:region:account-id:document/SSM-SessionManagerRunShell"
},
{
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession",
 "ssm:ResumeSession"
],
 "Resource": [
 "arn:aws:ssm:*::session/${aws:username}-*"
]
}
]
```

## Amazon EC2

Use this sample policy to give administrators the ability to perform session-related tasks from only the Amazon EC2 console. 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" ①
],
 "Resource": [
 "arn:aws:ec2:region:account-id:instance/*"
],
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/tag-keytag-value"
]
 }
 }
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetConnectionStatus",
 "ssm:DescribeInstanceInformation"
]
 }
]
}
```

```

],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession",
 "ssm:ResumeSession"
],
 "Resource": [
 "arn:aws:ssm:*:*:session/${aws:username}-*"
]
 }
]
}

```

### Session Manager, CLI, and Amazon EC2

Use this sample policy to give administrators the ability to perform session-related tasks from the Session Manager console, the AWS CLI, and the Amazon EC2 console.

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:StartSession",
 "ssm:SendCommand" ①
],
 "Resource": [
 "arn:aws:ec2:region:account-id:instance/*"
],
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/tag-keytag-value"
]
 }
 }
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:DescribeSessions",
 "ssm:GetConnectionStatus",
 "ssm:DescribeInstanceInformation",
 "ssm:DescribeInstanceProperties",
 "ec2:DescribeInstances"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm>CreateDocument",
 "ssm:UpdateDocument",
 "ssm:GetDocument"
],
 "Resource": "arn:aws:ssm:region:account-id:document/SSM-SessionManagerRunShell"
 },
 {
 "Effect": "Allow",

```

```
 "Action": [
 "ssm:TerminateSession",
 "ssm:ResumeSession"
],
 "Resource": [
 "arn:aws:ssm:*:*:session/${aws:username}-*"
]
 }
}
```

<sup>1</sup> 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 AWS Identity and Access Management (IAM) policy for any Session Manager user access scenarios you want to support.

### Topics

- [Example 1: Restrict access to specific managed nodes \(p. 935\)](#)
- [Example 2: Restrict access based on tags \(p. 936\)](#)
- [Example 3: Allow a user to end only sessions they started \(p. 937\)](#)
- [Example 4: Allow full \(administrative\) access to all sessions \(p. 940\)](#)

### Example 1: Restrict access to specific managed nodes

You can restrict access to specific managed nodes by creating an IAM user policy that includes the IDs of the nodes. In the following example, the user is allowed Session Manager access to three specific managed nodes only, and allowed to end only their sessions on those nodes. If the user sends a command to any other managed node or tries to end any other session, the command result will include `AccessDenied`.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:StartSession"
],
 "Resource": [
 "arn:aws:ec2:us-east-2:123456789012:instance/i-1234567890EXAMPLE",
 "arn:aws:ec2:us-east-2:123456789012:instance/i-abcdefghiJEXAMPLE",
 "arn:aws:ec2:us-east-2:123456789012:instance/i-0e9d8c7b6aEXAMPLE"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession",
 "ssm:ResumeSession"
],
 "Resource": [
 "arn:aws:ssm:*:*:session/${aws:username}-*"
]
 }
]
}
```

}

### Example 2: Restrict access based on tags

You can restrict access to managed nodes based on specific tags. In the following example, the user is allowed to start and resume sessions (`Effect: Allow, Action: ssm:StartSession, ssm:ResumeSession`) on any managed node (`Resource: arn:aws:ec2:region:987654321098:instance/*`) with the condition that the node is a Finance WebServer (`ssm:resourceTag/Finance: WebServer`). If the user sends a command to a managed node that isn't tagged or that has any tag other than `Finance: WebServer`, the command result will include `AccessDenied`.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:StartSession"
],
 "Resource": [
 "arn:aws:ec2:us-east-2:123456789012:instance/*"
],
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/Finance": [
 "WebServers"
]
 }
 }
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession",
 "ssm:ResumeSession"
],
 "Resource": [
 "arn:aws:ssm:*:*:session/${aws:username}-*"
]
 }
]
}
```

You can create IAM policies that allow a user to start sessions to managed nodes that are tagged with multiple tags. The following policy allows the user to start sessions to managed nodes that have both the specified tags applied to them. If a user sends a command to a managed node that isn't tagged with both of these tags, the command result will include `AccessDenied`.

```
{
 "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 managed nodes, see [Tagging managed nodes \(p. 1515\)](#) and [Tagging your Amazon EC2 resources](#) in the *Amazon EC2 User Guide for Linux Instances* (content applies to Windows and Linux managed nodes). For more information about increasing your security posture against unauthorized root-level commands on your managed nodes, see [Restricting access to root-level commands through SSM Agent \(p. 134\)](#)

### 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 doesn't 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 allows a user to view the IDs of all sessions in your account. However, users can interact with managed nodes 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 doesn't work for accounts that grant access to AWS using federated IDs.

```

{
 "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 managed node ID as the value in policy. In the following policy block, the condition statement allows a user to end only the node `i-02573cafccfEXAMPLE`.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession"
],
 "Resource": "*",
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/aws:ssmmessages:target-id": [
 "i-02573cafccfEXAMPLE"
]
 }
 }
 }
]
}
```

If the user tries to end a session for which they haven't been granted this `TerminateSession` permission, they receive an `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 `aws:ssmmessages:session-id` is the ID of the user. In this example, `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*.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:StartSession"
],
 "Resource": "
 arn:aws:ssm:
 <region>:
 <account>:
 session/
 AIDIODR4TAW7CSEXAMPLE
 "
 }
]
}
```

```

 "Action": [
 "ssm:TerminateSession"
],
 "Resource": "*",
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/aws:ssmmessages:session-id": [
 "AIDIODR4TAW7CSEEXAMPLE"
]
 }
 }
}
]
}
}

```

The following example demonstrates a policy for cases where the caller type is `AssumedRole`. You can use the `{aws:userid}` variable for the value you supply for `aws:ssmmessages:session-id`. Alternatively, you can hardcode a role ID for the value you supply for `aws:ssmmessages:session-id`. If you hardcode a role ID, you must provide the value in the format `role-id:caller-specified-role-name`. For example, `AIDIODR4TAW7CSEEXAMPLE: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.

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:TerminateSession"
],
 "Resource": "*",
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/aws:ssmmessages:session-id": [
 "${aws:userid}"
]
 }
 }
 }
]
}

```

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 allow a user to end sessions that are tagged with both system tags, as shown in this example.

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",

```

```
"Action": [
 "ssm:TerminateSession"
],
"Resource": "*",
"Condition": {
 "StringLike": {
 "ssm:resourceTag/aws:ssmmessages:target-id": [
 "i-02573cafefEXAMPLE"
],
 "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 managed nodes and all sessions created by all users for all nodes. It should be granted only to an Administrator who needs full control over your organization's Session Manager activities.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "ssm:StartSession",
 "ssm:TerminateSession",
 "ssm:ResumeSession",
 "ssm:DescribeSessions",
 "ssm:GetConnectionStatus"
],
 "Effect": "Allow",
 "Resource": [
 "*"
]
 }
]
}
```

## Step 4: Configure session preferences

An AWS Identity and Access Management (IAM) user with administrator permissions can do the following:

- Turn on Run As support for Linux managed nodes. 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 AWS Systems Manager Session Manager can create on a managed node.
- Configure Session Manager to use AWS KMS key encryption to provide additional protection to the data transmitted between client machines and managed nodes.
- 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 managed nodes and the commands run on them during the sessions.
- Configure session timeouts. You can use this setting to specify when to end a session after a period of inactivity.

- Configure Session Manager to use configurable shell profiles. These customizable profiles allow you to define preferences within sessions such as shell preferences, environment variables, working directories, and running multiple commands when a session is started.

**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 don't 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 \(p. 941\)](#)
- [Specify an idle session timeout value \(p. 942\)](#)
- [Specify maximum session duration \(p. 942\)](#)
- [Allow configurable shell profiles \(p. 943\)](#)
- [Turn on run as support for Linux and macOS managed nodes \(p. 944\)](#)
- [Turn on KMS key encryption of session data \(console\) \(p. 945\)](#)
- [Create Session Manager preferences \(command line\) \(p. 946\)](#)
- [Update Session Manager preferences \(command line\) \(p. 950\)](#)

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 \(console\) \(p. 979\)](#)
- [Streaming session data using Amazon CloudWatch Logs \(console\) \(p. 979\)](#)
- [Logging session data using Amazon CloudWatch Logs \(console\) \(p. 981\)](#)

## Grant or deny a user permissions to update Session Manager preferences

Account preferences are stored as AWS Systems Manager (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 AWS Identity and Access Management (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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "ssm:CreateDocument",
 "ssm:GetDocument",
 "ssm:UpdateDocument",
 "ssm:DeleteDocument"
],
 "Effect": "Allow",
 "Resource": [
 "arn:aws:ssm:us-east-2:123456789012:document/SSM-SessionManagerRunShell"
]
 }
]
}
```

```
]
}
```

### 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.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "ssm:CreateDocument",
 "ssm:GetDocument",
 "ssm:UpdateDocument",
 "ssm:DeleteDocument"
],
 "Effect": "Deny",
 "Resource": [
 "arn:aws:ssm:us-east-2:123456789012:document/SSM-SessionManagerRunShell"
]
 }
]
}
```

### Specify an idle session timeout value

Session Manager, a capability of AWS Systems Manager, allows you to specify the amount of time to allow a user to be inactive before the system ends a session. By default, sessions time out after 20 minutes of inactivity. You can modify this setting to specify that a session times out between 1 and 60 minutes of inactivity. Some professional computing security agencies recommend setting idle session timeouts to a maximum of 15 minutes.

#### To allow idle session timeout (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Session Manager**.
3. Choose the **Preferences** tab, and then choose **Edit**.
4. Specify the amount of time to allow a user to be inactive before a session ends in the **minutes** field under **Idle session timeout**.
5. Choose **Save**.

### Specify maximum session duration

Session Manager, a capability of AWS Systems Manager, allows you to specify the maximum duration of a session before it ends. The value you specify for maximum session duration must be between 1 and 1,440 minutes.

#### To specify maximum session duration (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 maximum session duration**.
5. Specify the maximum duration of session before it ends in the **minutes** field under **Maximum session duration**.

6. Choose **Save**.

## Allow configurable shell profiles

By default, sessions on EC2 instances for Linux start using the Bourne shell (sh). However, you might prefer to use another shell like bash. By allowing configurable shell profiles, you can customize preferences within sessions such as shell preferences, environment variables, working directories, and running multiple commands when a session is started.

### Important

Systems Manager doesn't check the commands or scripts in your shell profile to see what changes they would make to an instance before they're run. To restrict a user's ability to modify commands or scripts entered in their shell profile, we recommend the following:

- Create a customized Session-type document for your AWS Identity and Access Management (IAM) users and roles. Then modify the IAM policy for these users and roles so the StartSession API operation can only use the Session-type document you have created for them. For information see, [Create Session Manager preferences \(command line\) \(p. 946\)](#) and [Quickstart end user policies for Session Manager \(p. 929\)](#).
- Modify the IAM policy for your IAM users and roles to deny access to the UpdateDocument API operation for the Session-type document resource you create. This allows your users and roles to use the document you created for their session preferences without allowing them to modify any of the settings.

## To turn on configurable shell profiles

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Session Manager**.
3. Choose the **Preferences** tab, and then choose **Edit**.
4. Specify the environment variables, shell preferences, or commands you want to run when your session starts in the fields for the applicable operating systems.
5. Choose **Save**.

The following are some example commands that can be added to your shell profile.

Change to the bash shell and change to the /usr directory on Linux instances.

```
/bin/bash
cd /usr
```

Output a timestamp and welcome message at the start of a session.

Linux & macOS

```
timestamp=$(date '+%Y-%m-%dT%H:%M:%SZ')
user=$(whoami)
echo $timestamp && echo "Welcome $user"!
echo "You have logged in to a production instance. Note that all session activity is
being logged."
```

Windows

```
$timestamp = (Get-Date).ToString("yyyy-MM-ddTH:mm:ssZ")
$splitName = (whoami).Split("\")
$user = $splitName[1]
Write-Host $timestamp
```

```
Write-Host "Welcome $user!"
Write-Host "You have logged in to a production instance. Note that all session activity
is being logged."
```

**View dynamic system activity at the start of a session.**

Linux & macOS

top

## Windows

```

while ($true) { Get-Process | Sort-Object -Descending CPU | Select-Object -First 30; `

Start-Sleep -Seconds 2; cls
Write-Host "Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName";
Write-Host "----- ----- ----- ----- ----- ----- -----"

```

Turn on run as support for Linux and macOS managed nodes

By default, sessions are launched using the credentials of a system-generated `ssm-user` account that is created on a managed node. (On Linux and macOS machines, this account is added to `/etc/sudoers/`.) You can instead launch sessions using the credentials of an operating system (OS) account. The `root` account is not supported. OS level and directory policies, like log in restrictions or system resource usage restrictions, might not apply to the OS account since sessions are authenticated by AWS Identity and Access Management (IAM) and run within the context of the OS account that you specify. AWS Systems Manager Session Manager verifies the OS account that you specify exists before starting a session. If you specify an OS account that doesn't exist, the session fails to start.

## How it works

If you turn on 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 managed node? If it does, start the session. If it doesn't, don't 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 managed node? If it does, start the session. If it doesn't, don't allow a session to start.

At this point, Session Manager doesn't fall back on the default `ssm-user` account. In other words, allowing Run As support prevents sessions from being started using an `ssm-user` account on a managed node.

If you complete the following procedure without specifying an OS user account, or tagging an IAM user or role, sessions fail.

## To turn on Run As support for Linux and macOS managed nodes

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.

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:** In the **Operating system user name** field, enter the name of the OS user account on the target managed node that you want to use to start sessions. Using this option, all sessions are run by the same OS user for all the IAM users in your account who connect to the managed node using Session Manager.
  - **Option 2 (Recommended):** 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 managed node for the key value. Choose **Save changes**.

Using this option, 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 resources, see [Tagging IAM resources](#) in the *IAM User Guide*

The following is an example.

Key	Value (optional)	Remove
SSMSessionRunAs	My-OS-User-Name	
<a href="#">Add new key</a>		

You can add 49 more tags.

6. Choose **Save**.

### Turn on 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 managed nodes and the local machines of users in your AWS account is encrypted using KMS key encryption. (This is in addition to the TLS 1.2 encryption that AWS already provides by default.) KMS key encryption for sessions is accomplished using a key that is created in AWS KMS. AWS KMS encryption is available for `Standard_Stream`, `InteractiveCommands`, and `NonInteractiveCommands` session types. To use the option to encrypt session data using a key created in AWS KMS, version 2.3.539.0 or later of AWS Systems Manager SSM Agent must be installed on the managed node.

#### Note

You must allow AWS KMS encryption in order to reset passwords on your managed nodes from the AWS Systems Manager console. For more information, see [Reset a password on a managed node \(p. 813\)](#).

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 turn on KMS key encryption for your session data, both the users who start sessions and the managed nodes that they connect to must have permission to use the key. You provide permission to use the KMS key with Session Manager through AWS Identity and Access Management (IAM) policies. For information, see the following topics:

- Add AWS KMS permissions for users in your account: [Quickstart default IAM policies for Session Manager \(p. 928\)](#).
- Add AWS KMS permissions for managed nodes in your account: [Step 2: Verify or create an IAM role with Session Manager permissions \(p. 920\)](#).

For more information about creating and managing KMS keys, see the [AWS Key Management Service Developer Guide](#).

For information about using the AWS CLI to turn on KMS key encryption of session data in your account, see [Create Session Manager preferences \(command line\) \(p. 946\)](#) or [Update Session Manager preferences \(command line\) \(p. 950\)](#).

**Note**

There is a charge to use KMS keys. For information, see [AWS Key Management Service pricing](#).

### To turn on KMS key encryption of session data (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 KMS encryption**.
5. Do one of the following:
  - Choose the button next to **Select a KMS key in my current account**, then select a key from the list.

-or-

Choose the button next to **Enter a KMS key alias or KMS key ARN**. Manually enter a KMS key alias for a key created in your current account, or enter the key Amazon Resource Name (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

-or-

Choose **Create new key** to create a new KMS key 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 key](#) 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 AWS Systems Manager Session Manager preferences for your AWS account in the selected AWS Region. Use Session Manager preferences to specify options for logging session data in an Amazon Simple Storage

Service (Amazon 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 \(command line\) \(p. 950\)](#).

For an example of how to create session preferences using AWS CloudFormation, see [Create a Systems Manager document for Session Manager preferences](#) in the *AWS CloudFormation User Guide*.

**Note**

You can use this procedure to create custom Session documents for your Session Manager preferences that override account level settings. When you create your custom Session documents, specify a value other than `SSM-SessionManagerRunShell` for the name parameter and modify the inputs as needed. To use your custom Session documents, you must provide the name of your custom Session document for the `--document-name` parameter when starting a session from the AWS Command Line Interface (AWS CLI). When you start a session from the console, you can't specify custom Session documents.

### 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.

```
{
 "schemaVersion": "1.0",
 "description": "Document to hold regional settings for Session Manager",
 "sessionType": "Standard_Stream",
 "inputs": {
 "s3BucketName": "",
 "s3KeyPrefix": "",
 "s3EncryptionEnabled": true,
 "cloudWatchLogGroupName": "",
 "cloudWatchEncryptionEnabled": true,
 "cloudWatchStreamingEnabled": false,
 "kmsKeyId": "",
 "runAsEnabled": false,
 "runAsDefaultUser": "",
 "idleSessionTimeout": "",
 "maxSessionDuration": "",
 "shellProfile": {
 "windows": "date",
 "linux": "pwd;ls"
 }
 }
}
```

You can also pass values to your session preferences using parameters instead of hardcoding the values as shown in the following example.

```
{
 "schemaVersion": "1.0",
 "description": "Session Document Parameter Example JSON Template",
 "sessionType": "Standard_Stream",
 "parameters": {
 "s3BucketName": {
 "type": "String",
 "default": ""
 },
 "s3KeyPrefix": {
 "type": "String",
 "default": ""
 },
 },
}
```

```

 "s3EncryptionEnabled": {
 "type": "Boolean",
 "default": "false"
 },
 "cloudWatchLogGroupName": {
 "type": "String",
 "default": ""
 },
 "cloudWatchEncryptionEnabled": {
 "type": "Boolean",
 "default": "false"
 }
},
"inputs": {
 "s3BucketName": "{{s3BucketName}}",
 "s3KeyPrefix": "{{s3KeyPrefix}}",
 "s3EncryptionEnabled": "{{s3EncryptionEnabled}}",
 "cloudWatchLogGroupName": "{{cloudWatchLogGroupName}}",
 "cloudWatchEncryptionEnabled": "{{cloudWatchEncryptionEnabled}}",
 "kmsKeyId": ""
}
}
}

```

- 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 managed nodes, provide the KMS key to use for encryption. The following is an example.

```

{
 "schemaVersion": "1.0",
 "description": "Document to hold regional settings for Session Manager",
 "sessionType": "Standard_Stream",
 "inputs": {
 "s3BucketName": "DOC-EXAMPLE-BUCKET",
 "s3KeyPrefix": "MyBucketPrefix",
 "s3EncryptionEnabled": true,
 "cloudWatchLogGroupName": "MyLogGroupName",
 "cloudWatchEncryptionEnabled": true,
 "cloudWatchStreamingEnabled": false,
 "kmsKeyId": "MyKMSKeyID",
 "runAsEnabled": true,
 "runAsDefaultUser": "MyDefaultRunAsUser",
 "idleSessionTimeout": "20",
 "maxSessionDuration": "60",
 "shellProfile": {
 "windows": "MyCommands",
 "linux": "MyCommands"
 }
 }
}

```

#### Note

If you don't want to encrypt the session log data, change `true` to `false` for `s3EncryptionEnabled`.

If you aren't sending logs to either an Amazon S3 bucket or a CloudWatch Logs log group, don't want to encrypt active session data, or don't want to turn on 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 doesn't end with a comma.

If you add a KMS key ID to encrypt your session data, both the users who start sessions and the managed nodes that they connect to must have permission to use the key. You provide permission to use the KMS key with Session Manager through IAM policies. For information, see the following topics:

- Add AWS KMS permissions for users in your account: [Quickstart default IAM policies for Session Manager \(p. 928\)](#)
  - Add AWS KMS permissions for managed nodes in your account: [Step 2: Verify or create an IAM role with Session Manager permissions \(p. 920\)](#)
3. Save the file.
4. In the directory where you created the JSON file, run the following command.

Linux & macOS

```
aws ssm create-document \
--name SSM-SessionManagerRunShell \
--content "file://SessionManagerRunShell.json" \
--document-type "Session" \
--document-format JSON
```

Windows

```
aws ssm create-document ^
--name SSM-SessionManagerRunShell ^
--content "file://SessionManagerRunShell.json" ^
--document-type "Session" ^
--document-format JSON
```

PowerShell

```
New-SSMDocument `
-Name "SSM-SessionManagerRunShell" `
-Content (Get-Content -Raw SessionManagerRunShell.json) `
-DocumentType "Session" `
-DocumentFormat JSON
```

If successful, the command returns output similar to the following.

```
{
 "DocumentDescription": {
 "Status": "Creating",
 "Hash": "ce4fd0a2ab9b0fae759004ba603174c3ec2231f21a81db8690a33eb66EXAMPLE",
 "Name": "SSM-SessionManagerRunShell",
 "Tags": [],
 "DocumentType": "Session",
 "PlatformTypes": [
 "Windows",
 "Linux"
],
 "DocumentVersion": "1",
 "HashType": "Sha256",
 "CreatedDate": 1547750660.918,
 "Owner": "111122223333",
 "SchemaVersion": "1.0",
 "DefaultVersion": "1",
 "DocumentFormat": "JSON",
 "LatestVersion": "1"
 }
}
```

## Update Session Manager preferences (command line)

The following procedure describes how to use your preferred command line tool to make changes to the AWS Systems Manager Session Manager preferences for your AWS account in the selected AWS Region. Use Session Manager preferences to specify options for logging session data in an Amazon Simple Storage Service (Amazon 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.

```
{
 "schemaVersion": "1.0",
 "description": "Document to hold regional settings for Session Manager",
 "sessionType": "Standard_Stream",
 "inputs": {
 "s3BucketName": "",
 "s3KeyPrefix": "",
 "s3EncryptionEnabled": true,
 "cloudWatchLogGroupName": "",
 "cloudWatchEncryptionEnabled": true,
 "cloudWatchStreamingEnabled": false,
 "kmsKeyId": "",
 "runAsEnabled": true,
 "runAsDefaultUser": "",
 "idleSessionTimeout": "",
 "maxSessionDuration": "",
 "shellProfile": {
 "windows": "date",
 "linux": "pwd;ls"
 }
 }
}
```

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 managed nodes, provide the AWS KMS key to use for encryption. The following is an example.

```
{
 "schemaVersion": "1.0",
 "description": "Document to hold regional settings for Session Manager",
 "sessionType": "Standard_Stream",
 "inputs": {
 "s3BucketName": "DOC-EXAMPLE-BUCKET",
 "s3KeyPrefix": "MyBucketPrefix",
 "s3EncryptionEnabled": true,
 "cloudWatchLogGroupName": "MyLogGroupName",
 "cloudWatchEncryptionEnabled": true,
 "cloudWatchStreamingEnabled": false,
 "kmsKeyId": "MyKMSKeyID",
 "runAsEnabled": true,
 "runAsDefaultUser": "MyDefaultRunAsUser",
 "idleSessionTimeout": "20",
 "maxSessionDuration": "60",
 "shellProfile": {
 "windows": "MyCommands",
 "linux": "MyCommands"
 }
 }
}
```

}

**Note**

If you don't want to encrypt the session log data, change `true` to `false` for `s3EncryptionEnabled`.

If you aren't sending logs to either an Amazon S3 bucket or a CloudWatch Logs log group, don't want to encrypt active session data, or don't want to turn on 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 doesn't end with a comma.

If you add a KMS key ID to encrypt your session data, both the users who start sessions and the managed nodes that they connect to must have permission to use the key. You provide permission to use the KMS key with Session Manager through AWS Identity and Access Management (IAM) policies. For information, see the following topics:

- Add AWS KMS permissions for users in your account: [Quickstart default IAM policies for Session Manager \(p. 928\)](#).
- Add AWS KMS permissions for managed nodes in your account: [Step 2: Verify or create an IAM role with Session Manager permissions \(p. 920\)](#).

3. Save the file.
4. In the directory where you created the JSON file, run the following command.

Linux & macOS

```
aws ssm update-document \
--name "SSM-SessionManagerRunShell" \
--content "file://SessionManagerRunShell.json" \
--document-version "\$LATEST"
```

Windows

```
aws ssm update-document ^
--name "SSM-SessionManagerRunShell" ^
--content "file://SessionManagerRunShell.json" ^
--document-version "\$LATEST"
```

PowerShell

```
Update-SSMDocument `
-Name "SSM-SessionManagerRunShell" `
-Content (Get-Content -Raw SessionManagerRunShell.json) `
-DocumentVersion '$LATEST'
```

If successful, the command returns output similar to the following.

```
{
 "DocumentDescription": {
 "Status": "Updating",
 "Hash": "ce4fd0a2ab9b0fae759004ba603174c3ec2231f21a81db8690a33eb66EXAMPLE",
 "Name": "SSM-SessionManagerRunShell",
 "Tags": [],
 "DocumentType": "Session",
 "PlatformTypes": [
 "Windows",
 "Linux"
],
 "DocumentVersion": "2",
 "LastModifiedDate": "2023-01-12T12:00:00Z",
 "LastSuccessfulRun": "2023-01-12T12:00:00Z",
 "DefaultVersion": true,
 "DefaultVersionHash": "ce4fd0a2ab9b0fae759004ba603174c3ec2231f21a81db8690a33eb66EXAMPLE",
 "DefaultVersionName": "SSM-SessionManagerRunShell",
 "DefaultVersionTags": [],
 "DefaultVersionDocumentType": "Session",
 "DefaultVersionPlatformTypes": [
 "Windows",
 "Linux"
]
 }
}
```

```
 "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 AWS Systems Manager Session Manager session by creating a custom `Session` type AWS Systems Manager (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 AWS Identity and Access Management (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 \(p. 957\)](#).

Custom `Session` type SSM documents can only be used when starting sessions from the AWS Command Line Interface (AWS CLI). 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 \(interactive and noninteractive commands\) \(p. 975\)](#).

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)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Documents**.
3. Choose **Create command or session**.
4. For **Name**, enter 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.

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"
 allowedPattern: "[a-zA-Z0-9-_/.]+(.log)$"
properties:
 linux:
```

```
commands: "tail -f {{ logpath }}"
runAsElevated: true
```

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",
 "allowedPattern": "[a-zA-Z0-9-_/.]+(.log)$"
 }
 },
 "properties": {
 "linux": {
 "commands": "tail -f {{ logpath }}",
 "runAsElevated": true
 }
 }
}
```

7. Choose **Create document**.

## Restrict access to commands in a session (command line)

### Before you begin

If you haven't already, install and configure the AWS Command Line Interface (AWS CLI) or the AWS Tools for PowerShell. For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

### 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

```

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"
 allowedPattern: "[a-zA-Z0-9-_/.]+(.log)$"
properties:
 linux:
 commands: "tail -f {{ logpath }}"
 runAsElevated: true
```

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",
 "allowedPattern": "[a-zA-Z0-9-_/.]+(.log)$"
 }
},
"properties": {
 "linux": {
 "commands": "tail -f {{ logpath }}",
 "runAsElevated": true
 }
}
}

```

- 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 & macOS

```

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"

```

## Interactive command parameters and the AWS CLI

There are a variety of ways you can provide interactive command parameters when using the AWS CLI. Depending on the operating system (OS) of your client machine that you use to connect to managed nodes with the AWS CLI, the syntax you provide for commands that contain special or escape characters might differ. The following examples show some of the different ways you can provide command parameters when using the AWS CLI, and how to handle special or escape characters.

Parameters stored in Parameter Store can be referenced in the AWS CLI for your command parameters as shown in the following example.

#### Linux & macOS

```

aws ssm start-session \
--target instance-id \
--document-name MyInteractiveCommandDocument \

```

```
--parameters '{"command": ["{{ssm:mycommand}}"]}'
```

### Windows

```
aws ssm start-session ^
--target instance-id ^
--document-name MyInteractiveCommandDocument ^
--parameters '{"command": ["{{ssm:mycommand}}"]}'
```

The following example shows how you can use a shorthand syntax with the AWS CLI to pass parameters.

### Linux & macOS

```
aws ssm start-session \
--target instance-id \
--document-name MyInteractiveCommandDocument \
--parameters command="ifconfig"
```

### Windows

```
aws ssm start-session ^
--target instance-id ^
--document-name MyInteractiveCommandDocument ^
--parameters command="ipconfig"
```

You can also provide parameters in JSON as shown in the following example.

### Linux & macOS

```
aws ssm start-session \
--target instance-id \
--document-name MyInteractiveCommandDocument \
--parameters '{"command": ["ifconfig"]}'
```

### Windows

```
aws ssm start-session ^
--target instance-id ^
--document-name MyInteractiveCommandDocument ^
--parameters '{"command": ["ipconfig"]}'
```

Parameters can also be stored in a JSON file and provided to the AWS CLI as shown in the following example. For more information about using AWS CLI parameters from a file, see [Loading AWS CLI parameters from a file](#) in the *AWS Command Line Interface User Guide*.

```
{
 "command": [
 "my command"
]
}
```

### Linux & macOS

```
aws ssm start-session \
```

```
--target instance-id \
--document-name MyInteractiveCommandDocument \
--parameters file://complete/path/to/file/parameters.json
```

#### Windows

```
aws ssm start-session ^
--target instance-id ^
--document-name MyInteractiveCommandDocument ^
--parameters file://complete/path/to/file/parameters.json
```

You can also generate an AWS CLI skeleton from a JSON input file as shown in the following example. For more information about generating AWS CLI skeletons from JSON input files, see [Generating AWS CLI skeleton and input parameters from a JSON or YAML input file](#) in the *AWS Command Line Interface User Guide*.

```
{
 "Target": "instance-id",
 "DocumentName": "MyInteractiveCommandDocument",
 "Parameters": {
 "command": [
 "my command"
]
 }
}
```

#### Linux & macOS

```
aws ssm start-session \
--cli-input-json file://complete/path/to/file/parameters.json
```

#### Windows

```
aws ssm start-session ^
--cli-input-json file://complete/path/to/file/parameters.json
```

To escape characters inside quotation marks, you must add additional backslashes to the escape characters as shown in the following example.

#### Linux & macOS

```
aws ssm start-session \
--target instance-id \
--document-name MyInteractiveCommandDocument \
--parameters '{"command":["printf \"abc\\\\\\tdef\""]}'
```

#### Windows

```
aws ssm start-session ^
--target instance-id ^
--document-name MyInteractiveCommandDocument ^
--parameters '{"command":["printf \"abc\\\\\\tdef\""]}'
```

For information about using quotation marks with command parameters in the AWS CLI, see [Using quotation marks with strings in the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

## 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 managed node

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "ssm:StartSession",
 "Resource": [
 "arn:aws:ec2:region:987654321098:instance/i-02573cafcfEXAMPLE",
 "arn:aws:ssm:region:987654321098:document/exampleAllowedSessionDocument"
],
 "Condition": {
 "BoolIfExists": {
 "ssm:SessionDocumentAccessCheck": "true"
 }
 }
 }
]
}
```

### Allow a user to run an interactive command on all managed nodes

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "ssm:StartSession",
 "Resource": [
 "arn:aws:ec2:us-west-2:987654321098:instance/*",
 "arn:aws:ssm:us-west-2:987654321098:document/exampleAllowedSessionDocument"
],
 "Condition": {
 "BoolIfExists": {
 "ssm:SessionDocumentAccessCheck": "true"
 }
 }
 }
]
}
```

### Allow a user to run multiple interactive commands on all managed nodes

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "ssm:StartSession",
 "Resource": [
 "arn:aws:ec2:us-west-2:987654321098:instance/*",
 "arn:aws:ssm:us-
west-2:987654321098:document/exampleAllowedSessionDocument",
 "arn:aws:ssm:us-
west-2:987654321098:document/exampleAllowedSessionDocument2"
]
 }
]
}
```

```
],
 "Condition": {
 "BoolIfExists": {
 "ssm:SessionDocumentAccessCheck": "true"
 }
 }
]
}
```

## Step 6: (Optional) Use AWS PrivateLink to set up a VPC endpoint for Session Manager

You can further improve the security posture of your managed nodes by configuring AWS Systems Manager to use an interface virtual private cloud (VPC) endpoint. Interface endpoints are powered by AWS PrivateLink, a technology that allows you to privately access Amazon Elastic Compute Cloud (Amazon EC2) and Systems Manager APIs by using private IP addresses.

AWS PrivateLink restricts all network traffic between your managed nodes, Systems Manager, and Amazon EC2 to the Amazon network. (Managed nodes don't have access to the internet.) Also, you don't need an internet gateway, a NAT device, or a virtual private gateway.

For information about creating a VPC endpoint, see [\(Optional\) Create a VPC endpoint \(p. 28\)](#).

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

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

Systems Manager uses the last of these endpoints, `ssmmessages.region.amazonaws.com`, to make calls from SSM Agent to the Session Manager service in the cloud.

To use optional features like AWS Key Management Service (AWS KMS) encryption, streaming logs to Amazon CloudWatch Logs (CloudWatch Logs), and sending logs to Amazon Simple Storage Service (Amazon S3) you must allow HTTPS (port 443) outbound traffic to the following endpoints:

- `kms.region.amazonaws.com`
- `logs.region.amazonaws.com`
- `s3.region.amazonaws.com`

For more information about required endpoints for Systems Manager, see [Reference: ec2messages, ssmmessages, and other API operations \(p. 1546\)](#).

## Step 7: (Optional) Turn on or turn off ssm-user account administrative permissions

Starting with version 2.3.50.0 of AWS Systems Manager SSM Agent, the agent creates a local user account called `ssm-user` and adds it to `/etc/sudoers` (Linux and macOS) or to the Administrators group (Windows). On agent versions earlier than 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 a node. This `ssm-user` is the default operating system (OS) user when a AWS Systems Manager Session Manager session is started. SSM Agent version 2.3.612.0 was released on May 8th, 2019.

If you want to prevent Session Manager users from running administrative commands on a node, 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 and macOS \(p. 959\)](#)
- [Managing ssm-user Administrator account permissions on Windows Server \(p. 960\)](#)

## Managing ssm-user sudo account permissions on Linux and macOS

Use one of the following procedures to turn on or turn off the `ssm-user` account sudo permissions on Linux and macOS managed nodes.

### Use Run Command to modify ssm-user sudo permissions (console)

- Use the procedure in [Running commands from the console \(p. 996\)](#) 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
```

-or-

To restore sudo access, in the **Command parameters** area, paste the following in the **Commands** box.

```
cd /etc/sudoers.d
echo "ssm-user ALL=(ALL) NOPASSWD:ALL" > ssm-agent-users
```

### Use the command line to modify ssm-user sudo permissions (AWS CLI)

1. Connect to the managed node 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 turn on or turn off the ssm-user account Administrator permissions on Windows Server managed nodes.

### Use Run Command to modify Administrator permissions (console)

- Use the procedure in [Running commands from the console \(p. 996\)](#) 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 managed node 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 managed node 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, enter **Administrators** in the text box, and then choose **Add**.

5. Choose **OK**.

## Step 8: (Optional) Allow and controlling permissions for SSH connections through Session Manager

You can allow users in your AWS account to use the AWS Command Line Interface (AWS CLI) to establish Secure Shell (SSH) connections to managed nodes using AWS Systems Manager Session Manager. Users who connect using SSH can also copy files between their local machines and managed nodes using Secure Copy Protocol (SCP). You can use this functionality to connect to managed nodes without opening inbound ports or maintaining bastion hosts.

After allowing SSH connections, you can use AWS Identity and Access Management (IAM) policies to explicitly allow or deny users, groups, or roles to make SSH connections using Session Manager.

**Note**

Logging isn't 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.

**Topics**

- [Allowing SSH connections for Session Manager \(p. 961\)](#)
- [Controlling user permissions for SSH connections through Session Manager \(p. 962\)](#)

### Allowing SSH connections for Session Manager

Use the following steps to allow SSH connections through Session Manager on a managed node.

#### To allow SSH connections for Session Manager

1. On the managed node to which you want to allow SSH connections, do the following:
  - Ensure that SSH is running on the managed node. (You can close inbound ports on the node.)
  - Ensure that SSM Agent version 2.3.672.0 or later is installed on the managed node.

For information about installing or updating SSM Agent on a managed node, see the following topics:

- [Working with SSM Agent on EC2 instances for Windows Server \(p. 123\)](#).
- [Working with SSM Agent on EC2 instances for Linux \(p. 76\)](#)
- [Working with SSM Agent on EC2 instances for macOS \(p. 121\)](#)
- [Install SSM Agent for a hybrid environment \(Windows\) \(p. 50\)](#)
- [Install SSM Agent for a hybrid environment \(Linux\) \(p. 45\)](#)

**Note**

To use Session Manager with on-premises servers, edge devices, and virtual machines (VMs) that you activated as managed nodes, you must use the Advanced-Instances Tier. For more information about advanced instances, see [Turning on the advanced-instances tier \(p. 805\)](#).

2. On the local machine from which you want to connect to a managed node 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 \(p. 964\)](#).

- Update the SSH configuration file to allow running a proxy command that starts a Session Manager session and transfer all data through the connection.

### Linux and macOS

**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 nodes. This must be a key that is already associated with the managed node. The permissions of your private key file must be set so that only you can read it. You can use the following command to set the permissions of your private key file so that only you can read it.

```
chmod 400 <my-key-pair>.pem
```

For example, for an Amazon Elastic Compute Cloud (Amazon 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 \(SSH\) \(p. 972\)](#).)

## Controlling user permissions for SSH connections through Session Manager

After you enable SSH connections through Session Manager on a managed node, you can use IAM policies to allow or deny users, groups, or roles the ability to make SSH connections through Session Manager.

### To use an IAM policy to allow SSH connections through Session Manager

- Use one of the following options:
  - **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 you want to allow to start SSH connections through Session Manager.

For example, add the following element to the Quickstart policy you created in [Quickstart end user policies for Session Manager \(p. 929\)](#). Replace each *example resource placeholder* with your own information.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "ssm:StartSession",
 "Resource": [
 "arn:aws:ec2:region:account-id:instance/instance-id",
 "arn:aws:ssm:*:*:document/AWS-StartSSHSessions"
]
 }
]
}
```

- **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 use an IAM policy to deny SSH connections through Session Manager

- Use one of the following options:
  - **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, add the following element to the Quickstart policy you created in [Quickstart end user policies for Session Manager \(p. 929\)](#).

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "VisualEditor1",
 "Effect": "Deny",
 "Action": "ssm:StartSession",
 "Resource": "arn:aws:ssm:*:*:document/AWS-StartSSHSessions"
 }
]
}
```

- **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 Elastic Compute Cloud (Amazon EC2) console, or the AWS Command Line Interface (AWS CLI) to start sessions that connect you to the managed nodes 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 haven't timed out, and end sessions. After a session is established, it is not affected by IAM role session duration. For information about limiting session duration with Session Manager, see [Specify an idle session timeout value \(p. 942\)](#) and [Specify maximum session duration \(p. 942\)](#).

For more information about sessions, see [What is a session? \(p. 915\)](#)

## Topics

- [\(Optional\) Install the Session Manager plugin for the AWS CLI \(p. 964\)](#)
- [Start a session \(p. 971\)](#)
- [End a session \(p. 975\)](#)
- [View session history \(p. 976\)](#)

## (Optional) Install the Session Manager plugin for the AWS CLI

If you want to use the AWS Command Line Interface (AWS CLI) to start and end sessions that connect you to your managed nodes, 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 Session Manager plugin is updated occasionally with enhanced functionality. We recommend that you regularly ensure you're using the latest version of the plugin. For more information, see [Session Manager plugin latest version and release history \(p. 969\)](#).

### 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 \(p. 964\)](#)
- [Install and uninstall the Session Manager plugin on macOS \(p. 965\)](#)
- [Install the Session Manager plugin on macOS with the signed installer \(p. 966\)](#)
- [Install Session Manager plugin on Linux \(p. 966\)](#)
- [Install the Session Manager plugin on Ubuntu Server \(p. 967\)](#)
- [Verify the Session Manager plugin installation \(p. 967\)](#)
- [Session Manager plugin on GitHub \(p. 968\)](#)
- [\(Optional\) Turn on Session Manager plugin logging \(p. 968\)](#)
- [Session Manager plugin latest version and release history \(p. 969\)](#)

### 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 that you start sessions on Windows clients using Windows PowerShell, version 5 or later. Alternatively, you can use the Command shell in Microsoft Windows 10. The Session Manager plugin only supports PowerShell and the Command shell. Third-party command line tools might not be compatible with the plugin.

### 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
```

Alternatively, you can download a zipped version of the installer using the following URL.

```
https://s3.amazonaws.com/session-manager-downloads/plugin/latest/windows/
SessionManagerPlugin.zip
```

2. Run the downloaded installer, and follow the on-screen instructions. If you downloaded the zipped version of the installer, you must unzip the installer first.

Leave the install location box blank to install the plugin to the default directory.

- %PROGRAMFILES%\Amazon\SessionManagerPlugin\bin\

3. Verify that the installation was successful. For information, see [Verify the Session Manager plugin installation \(p. 967\)](#).

**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 \(Windows\) \(p. 989\)](#).

## 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 doesn't support installing to paths that contain spaces.

### To install the Session Manager plugin using the bundled installer (macOS)

1. Download the bundled installer.

```
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/mac/
sessionmanager-bundle.zip" -o "sessionmanager-bundle.zip"
```

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 either Python 2.6.5 or later, or Python 3.3 or later. 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 \(p. 967\)](#).

**Note**

If you ever want to uninstall the plugin, run the following two commands in the order shown.

```
sudo rm -rf /usr/local/sessionmanagerplugin
```

```
sudo rm /usr/local/bin/session-manager-plugin
```

## Install the Session Manager plugin on macOS with the signed installer

You can install the Session Manager plugin on macOS using the signed installer.

### To install the Session Manager plugin using the signed installer (macOS)

1. Download the signed installer.

```
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/mac/session-manager-plugin.pkg" -o "session-manager-plugin.pkg"
```

2. Run the install command.

```
sudo installer -pkg session-manager-plugin.pkg -target /
sudo ln -s /usr/local/sessionmanagerplugin/bin/session-manager-plugin /usr/local/bin/session-manager-plugin
```

3. Verify that the installation was successful. For information, see [Verify the Session Manager plugin installation \(p. 967\)](#).

## Install Session Manager plugin on Linux

1. Download the Session Manager plugin RPM package.

- x86\_64

```
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/linux_64bit/session-manager-plugin.rpm" -o "session-manager-plugin.rpm"
```

- x86

```
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/linux_32bit/session-manager-plugin.rpm" -o "session-manager-plugin.rpm"
```

- ARM64

```
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.

```
sudo yum install -y session-manager-plugin.rpm
```

3. Verify that the installation was successful. For information, see [Verify the Session Manager plugin installation \(p. 967\)](#).

**Note**

If you ever want to uninstall the plugin, run `sudo yum erase session-manager-plugin -y`

## Install the Session Manager plugin on Ubuntu Server

1. Download the Session Manager plugin deb package.

- x86\_64

```
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_64bit/session-manager-plugin.deb" -o "session-manager-plugin.deb"
```

- x86

```
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_32bit/session-manager-plugin.deb" -o "session-manager-plugin.deb"
```

- ARM64

```
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 \(p. 967\)](#).

**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. In the following command, replace *instance-id* with your own information.

```
aws ssm start-session --target instance-id
```

This command will work only if your Session Manager administrator has granted you the necessary IAM permissions to access the target managed node using Session Manager.

## Session Manager plugin on GitHub

The source code for Session Manager plugin is available on [GitHub](#) so that you can adapt the plugin to meet your needs. We encourage you to submit [pull requests](#) for changes that you would like to have included. However, Amazon Web Services doesn't provide support for running modified copies of this software.

### (Optional) Turn on Session Manager plugin logging

The Session Manager plugin includes an option to allow logging for sessions that you run. By default, logging is turned off.

If you allow 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

- [Turn on logging for the Session Manager plugin \(Windows\) \(p. 968\)](#)
- [Enable logging for the Session Manager plugin \(Linux and macOS\) \(p. 969\)](#)

#### Turn on 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 `formatid="fmtinfo"` to `outputs formatid="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

Don't change the file names or logging won't work correctly.

```
<rollingfile type="size" filename="C:\Program Files\Amazon\SessionManagerPlugin\Logs\session-manager-plugin.log" maxsize="30000000" maxrolls="5"/>
<filter levels="error,critical" formatid="fmterror">
<rollingfile type="size" filename="C:\Program Files\Amazon\SessionManagerPlugin\Logs\errors.log" maxsize="10000000" maxrolls="5"/>
```

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 `formatid="fmtinfo"` to `outputs formatid="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**

Don't change the file names or logging won't work correctly.

```
<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"/>
```

**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. The current minimum supported version is 1.1.17.0. If you're running an earlier version, your Session Manager operations might not succeed.

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.

Version	Release date	Details
1.2.331.0	May 27, 2022	<b>Bug fix:</b> Fix port sessions closing prematurely when the local server doesn't connect before timeout.
1.2.323.0	May 19, 2022	<b>Bug fix:</b> Disable smux keep alive to use idle session timeout feature.
1.2.312.0	March 31, 2022	<b>Enhancement:</b> Supports more output message payload types.
1.2.295.0	January 12, 2022	<b>Bug fix:</b> Hung sessions caused by client resending stream data when agent becomes inactive, and incorrect logs for <code>start_publication</code> and <code>pause_publication</code> messages.
1.2.279.0	October 27, 2021	<b>Enhancement:</b> Zip packaging for Windows platform.
1.2.245.0	August 19, 2021	<b>Enhancement:</b> Upgrade <code>aws-sdk-go</code> to latest version (v1.40.17) to support single sign-on (SSO).
1.2.234.0	July 26, 2021	<b>Bug fix:</b> Handle session abruptly terminated scenario in interactive session type.
1.2.205.0	June 10, 2021	<b>Enhancement:</b> Added support for signed macOS installer.
1.2.54.0	January 29, 2021	<b>Enhancement:</b> Added support for running sessions in NonInteractiveCommands execution mode.
1.2.30.0	November 24, 2020	<b>Enhancement:</b> (Port forwarding sessions only) Improved overall performance.
1.2.7.0	October 15, 2020	<b>Enhancement:</b> (Port forwarding sessions only) Reduced latency and improved overall performance.
1.1.61.0	April 17, 2020	<b>Enhancement:</b> Added ARM support for Linux and Ubuntu Server.
1.1.54.0	January 6, 2020	<b>Bug fix:</b> Handle race condition scenario of packets being dropped when the Session Manager plugin isn't ready.
1.1.50.0	November 19, 2019	<b>Enhancement:</b> Added support for forwarding a port to a local unix socket.
1.1.35.0	November 7, 2019	<b>Enhancement:</b> (Port forwarding sessions only) Send a <code>TerminateSession</code> command to SSM Agent when the local user presses <b>Ctrl+C</b> .
1.1.33.0	September 26, 2019	<b>Enhancement:</b> (Port forwarding sessions only) Send a disconnect signal to the server when the client drops the TCP connection.
1.1.31.0	September 6, 2019	<b>Enhancement:</b> Update to keep port forwarding session open until remote server closes the connection.
1.1.26.0	July 30, 2019	<b>Enhancement:</b> Update to limit the rate of data transfer during a session.

Version	Release date	Details
1.1.23.0	July 9, 2019	<b>Enhancement:</b> Added support for running SSH sessions using Session Manager.
1.1.17.0	April 4, 2019	<b>Enhancement:</b> Added support for further encryption of session data using AWS Key Management Service (AWS KMS).
1.0.37.0	September 20, 2018	<b>Enhancement:</b> Bug fix for Windows version.
1.0.0.0	September 11, 2018	Initial release of the Session Manager plugin.

## Start a session

You can use the AWS Systems Manager console, the Amazon Elastic Compute Cloud (Amazon EC2) console, the AWS Command Line Interface (AWS CLI), or SSH to start a session.

### Topics

- [Starting a session \(Systems Manager console\) \(p. 971\)](#)
- [Starting a session \(Amazon EC2 console\) \(p. 972\)](#)
- [Starting a session \(AWS CLI\) \(p. 972\)](#)
- [Starting a session \(SSH\) \(p. 972\)](#)
- [Starting a session \(port forwarding\) \(p. 973\)](#)
- [Starting a session \(port forwarding to remote host\) \(p. 974\)](#)
- [Starting a session \(interactive and noninteractive commands\) \(p. 975\)](#)

### Starting a session (Systems Manager console)

You can use the AWS Systems Manager console to start a session with a managed node in your account.

#### Note

Before you start a session, make sure that you have completed the setup steps for Session Manager. For information, see [Setting up Session Manager \(p. 916\)](#).

#### To start a session (Systems Manager console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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. (Optional) Enter a reason for the session in the **Reason for session** field.
5. For **Target instances**, choose the option button to the left of the managed node you want to connect to.

If a managed node you want to connect to isn't in the list, or is listed but an error message reports, "The instance you selected isn't configured to use Session Manager," see [Managed node not available or not configured for Session Manager \(p. 988\)](#) for troubleshooting steps.

6. Choose **Start session**.

After the connection is made, you can run bash commands (Linux and macOS) 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 aren't 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're an administrator, see [Quickstart default IAM policies for Session Manager \(p. 928\)](#) 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 and macOS) or PowerShell commands (Windows) as you would through any other connection type.

## Starting a session (AWS CLI)

Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

Before you start a session, make sure that you have completed the setup steps for Session Manager. For information, see [Setting up Session Manager \(p. 916\)](#).

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 \(p. 964\)](#).

To start a session using the AWS CLI, run the following command replacing `instance-id` with your own information.

```
aws ssm start-session \
--target 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 (SSH)

To start a Session Manager SSH session, version 2.3.672.0 or later of SSM Agent must be installed on the managed node.

### SSH connection requirements

Take note of the following requirements and limitations for session connections using SSH:

- Your target managed node must be configured to support SSH connections. For more information, see [\(Optional\) Enable and control permissions for SSH connections through Session Manager \(p. 961\)](#).
- You must connect using the managed node account 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 and macOS, 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 isn't 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.

**Note**

Before you start a session, make sure that you have completed the setup steps for Session Manager. For information, see [Setting up Session Manager \(p. 916\)](#).

To start a session using SSH, run the following command. Replace each *example resource placeholder* with your own information.

```
ssh -i /path/my-key-pair.pem username@instance-id
```

**Tip**

When you start a session using SSH, you can copy local files to the target managed node using the following command format.

```
scp -i /path/my-key-pair.pem /path/ExampleFile.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 Session Manager port forwarding session, version 2.3.672.0 or later of SSM Agent must be installed on the managed node.

**Note**

Before you start a session, make sure that you have completed the setup steps for Session Manager. For information, see [Setting up Session Manager \(p. 916\)](#).

To use the AWS CLI to run session commands, you must install the Session Manager plugin on your local machine. For information, see [\(Optional\) Install the Session Manager plugin for the AWS CLI \(p. 964\)](#).

Depending on your operating system and command line tool, the placement of quotation marks can differ and escape characters might be required.

To start a port forwarding session, run the following command from the CLI. Replace each *example resource placeholder* with your own information.

Linux & macOS

```
aws ssm start-session \
--target instance-id \
--document-name AWS-StartPortForwardingSession \
--parameters '{"portNumber":["80"], "localPortNumber":["56789"]}'
```

Windows

```
aws ssm start-session ^
--target instance-id ^
```

```
--document-name AWS-StartPortForwardingSession ^
--parameters portNumber="3389",localPortNumber="56789"
```

The value you specify for `portNumber` represents the remote port on the managed node where traffic should be redirected to, such as 3389 for connecting to a Windows node using the Remote Desktop Protocol (RDP). If this parameter isn't specified, Session Manager assumes 80 as the default remote port.

The value you specify for `localPortNumber` represents the local port on the client where traffic should be redirected to, such as 56789. This value is what you enter when connecting to a managed node using a client. For example, **localhost: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.

For more information about port forwarding sessions, see [Port Forwarding Using AWS Systems ManagerSession Manager](#) in the *AWS News Blog*.

## Starting a session (port forwarding to remote host)

To start a Session Manager port forwarding session to a remote host, version 3.x or later of SSM Agent must be installed on the managed node. The remote host isn't required to be managed by Systems Manager.

### Note

Before you start a session, make sure that you have completed the setup steps for Session Manager. For information, see [Setting up Session Manager \(p. 916\)](#).

To use the AWS CLI to run session commands, you must install the Session Manager plugin on your local machine. For information, see [\(Optional\) Install the Session Manager plugin for the AWS CLI \(p. 964\)](#).

Depending on your operating system and command line tool, the placement of quotation marks can differ and escape characters might be required.

To start a port forwarding session, run the following command from the CLI. Replace each *example resource placeholder* with your own information.

### Linux & macOS

```
aws ssm start-session \
--target instance-id \
--document-name AWS-StartPortForwardingSessionToRemoteHost \
--parameters '{"host":["mydb.example.us-east-2.rds.amazonaws.com"], "portNumber": ["3306"], "localPortNumber": ["3306"]}'
```

### Windows

```
aws ssm start-session ^
--target instance-id ^
--document-name AWS-StartPortForwardingSessionToRemoteHost ^
--parameters host="mydb.example.us-east-2.rds.amazonaws.com",portNumber="3306",localPortNumber="3306"
```

The value you specify for `host` represents the hostname or IP address of the remote host you want to connect to. General connectivity and name resolution requirements between the managed node and the remote host still apply.

The value you specify for `portNumber` represents the remote port on the managed node where traffic should be redirected to, such as 3306 for connecting to a MySQL database. If this parameter isn't specified, Session Manager assumes 80 as the default remote port.

The value you specify for `localPortNumber` represents the local port on the client where traffic should be redirected to, such as 56789. This value is what you enter when connecting to a managed node using a client. For example, `localhost: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 and noninteractive commands)

Before you start a session, make sure that you have completed the setup steps for Session Manager. For information, see [Setting up Session Manager \(p. 916\)](#).

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 \(p. 964\)](#).

To start an interactive command session, run the following command. Replace each *example resource placeholder* with your own information.

Linux & macOS

```
aws ssm start-session \
--target instance-id \
--document-name CustomCommandSessionDocument \
--parameters '{"logpath":["/var/log/amazon/ssm/amazon-ssm-agent.log"]}'
```

Windows

```
aws ssm start-session ^
--target instance-id ^
--document-name CustomCommandSessionDocument ^
--parameters logpath="/var/log/amazon/ssm/amazon-ssm-agent.log"
```

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 ManagerSession Manager](#) on the [AWS News Blog](#).

## End a session

You can use the AWS Systems Manager console or the AWS Command Line Interface (AWS CLI) to end a session that you started 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

- [Ending a session \(console\) \(p. 975\)](#)
- [Ending a session \(AWS CLI\) \(p. 976\)](#)

## Ending a session (console)

You can use the AWS Systems Manager console to end a session in your account.

### To end a session (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.

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**.

## Ending a session (AWS CLI)

To end a session using the AWS CLI, run the following command. Replace *session-id* with your own information.

```
aws ssm terminate-session \
 --session-id session-id
```

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 Command Line Interface (AWS CLI) to view information about sessions in your account. In the console, you can view session details such as the following:

- The ID of the session
- Which user connected to a managed node through a session
- The ID of the managed node
- When the session began and ended
- The status of the session
- The location specified for storing session logs (if turned on)

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 logging session history information, see [Logging session activity \(p. 978\)](#).

### Topics

- [Viewing session history \(console\) \(p. 976\)](#)
- [Viewing session history \(AWS CLI\) \(p. 976\)](#)

## 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)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Session Manager**.
3. Choose **Configure Preferences**.
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 doesn't 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 session activity

In addition to providing information about current and completed sessions in the Systems Manager console, Session Manager provides you with the ability to audit session activity in your AWS account using AWS CloudTrail.

CloudTrail captures session API calls through the Systems Manager console, the AWS Command Line Interface (AWS CLI), and the Systems Manager SDK. You can view the information on the CloudTrail console or store it in a specified Amazon Simple Storage Service (Amazon S3) bucket. One Amazon S3 bucket is used for all CloudTrail logs for your account. For more information, see [Logging AWS Systems Manager API calls with AWS CloudTrail \(p. 1442\)](#).

## Monitoring session activity using Amazon EventBridge (console)

With EventBridge, you can 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.

EventBridge support for Session Manager relies on records of API operations that were recorded by CloudTrail. (You can use CloudTrail integration with EventBridge to respond to most AWS Systems Manager events.) Actions that take place within a session, such as an `exit` command, that don't make an API call aren't detected by EventBridge.

The following steps outline how to initiate notifications through Amazon Simple Notification Service (Amazon SNS) when a Session Manager API event occurs, such as `StartSession`.

### To monitor session activity using Amazon EventBridge (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](#) in the *Amazon Simple Notification Service Developer Guide*.
2. Create an EventBridge 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 an EventBridge Rule That Triggers on an Event from an AWS Resource](#) in the *Amazon EventBridge User Guide*.

As you follow the steps to create the rule, make the following selections:

- For **Service Name**, choose **Systems Manager**.
- For **Event Type**, choose **AWS API Call through 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**. (EventBridge 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 EventBridge User Guide](#) and the [Amazon Simple Notification Service Getting Started Guide](#).

## 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 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 managed nodes 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:

- Session Manager logs the commands you enter and their output during a session depending on your session preferences. To prevent sensitive data, such as passwords, from being viewed in your session logs we recommend using the following commands when entering sensitive data during a session.

#### Linux & macOS

```
stty -echo; read passwd; stty echo;
```

#### Windows

```
$Passwd = Read-Host -AsSecureString
```

- If you're 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're using Linux or macOS managed nodes, ensure that the screen utility is installed. If it isn't, 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 isn't 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 IAM role with permissions for Session Manager and Amazon S3 and CloudWatch Logs \(console\) \(p. 924\)](#).

Refer to the following topics for more information about logging options for Session Manager.

## Topics

- [Streaming session data using Amazon CloudWatch Logs \(console\) \(p. 979\)](#)
- [Logging session data using Amazon S3 \(console\) \(p. 979\)](#)
- [Logging session data using Amazon CloudWatch Logs \(console\) \(p. 981\)](#)

## Streaming session data using Amazon CloudWatch Logs (console)

You can send a continual stream of session data logs to Amazon CloudWatch Logs. Essential details, such as the commands a user has run in a session, the ID of the user who ran the commands, and timestamps for when the session data is streamed to CloudWatch Logs, are included when streaming session data. When streaming session data, the logs are JSON-formatted to help you integrate with your existing logging solutions. Streaming session data isn't supported for interactive commands.

### Note

To stream session data from Windows Server managed nodes, you must have PowerShell 5.1 or later installed. By default, Windows Server 2016 and later have the required PowerShell version installed. However, Windows Server 2012 and 2012 R2 don't have the required PowerShell version installed by default. If you haven't already updated PowerShell on your Windows Server 2012 or 2012 R2 managed nodes, you can do so using Run Command. For information on updating PowerShell using Run Command, see [Update PowerShell using Run Command \(p. 999\)](#).

### Important

If you have the **PowerShell Transcription** policy setting configured on your Windows Server managed nodes, you won't be able to stream session data.

### To stream session data using Amazon CloudWatch Logs (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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** under **CloudWatch logging**.
5. Choose the **Stream session logs** option.
6. (Recommended) Select the check box next to **Allow only encrypted CloudWatch log groups**. With this option turned on, log data is encrypted using the server-side encryption key specified for the log group. If you don't want to encrypt the log data that is sent to CloudWatch Logs, clear the check box. You must also clear the check box if encryption isn't allowed on the log group.
7. 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:
  - Enter the name of a log group in the text box that has already been created in your account to store session log data.
  - **Browse log groups:** Select a log group that has already been created in your account to store session log data.
8. Choose **Save**.

## Logging session data using Amazon S3 (console)

You can choose to store session log data in a specified Amazon Simple Storage Service (Amazon S3) bucket for debugging and troubleshooting purposes. The default option is for logs to be sent to an

encrypted Amazon S3 bucket. Encryption is performed using the key specified for the bucket, either an 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 don't use periods (".") in bucket names when using virtual hosted-style buckets.

### Amazon S3 bucket encryption

In order to send logs to your Amazon S3 bucket with encryption, encryption must be allowed on the bucket. For more information about Amazon S3 bucket encryption, see [Amazon S3 Default Encryption for S3 Buckets](#).

#### Customer managed key

If you're using a KMS key that you manage yourself to encrypt your bucket, then the IAM instance profile attached to your instances must have explicit permissions to read the key. If you use an AWS managed key, the instance doesn't require this explicit permission. For more information about providing the instance profile with access to use the key, see [Allows Key Users to Use the key in the AWS Key Management Service Developer Guide](#).

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 Amazon S3 bucket that session data is sent to. For information, see [Update Session Manager preferences \(command line\) \(p. 950\)](#).

### To log session data using Amazon S3 (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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** under **S3 logging**.
5. (Recommended) Select the check box next to **Allow only encrypted S3 buckets**. With this option turned on, log data is encrypted using the server-side encryption key specified for the bucket. If you don't want to encrypt the log data that is sent to Amazon S3, clear the check box. You must also clear the check box if encryption isn't allowed on the S3 bucket.
6. For **S3 bucket name**, select one of the following:

**Note**

We recommend that you don't use periods (".") in bucket names when using virtual hosted-style buckets. For more information about Amazon S3 bucket-naming conventions, see [Bucket Restrictions and Limitations in the Amazon Simple Storage Service User Guide](#).

- **Choose a bucket name from the list:** Select an Amazon 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 Amazon 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 Amazon S3 buckets, see the [Amazon Simple Storage Service User Guide](#) and the [Amazon Simple Storage Service User Guide](#).

## Logging session data using Amazon CloudWatch Logs (console)

With Amazon CloudWatch Logs, you can monitor, store, and access log files from various AWS services. You can send session log data to a CloudWatch Logs log group for debugging and troubleshooting purposes. The default option is for log data to be sent with encryption using your KMS key, but you can send the data to your log group with or without encryption.

Follow these steps to configure AWS Systems Manager Session Manager to send session log data to a CloudWatch Logs log group at the end of your sessions.

**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 \(command line\) \(p. 950\)](#).

### To log session data using Amazon CloudWatch Logs (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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** under **CloudWatch logging**.
5. Choose the **Upload session logs** option.
6. (Recommended) Select the check box next to **Allow only encrypted CloudWatch log groups**. With this option turned on, log data is encrypted using the server-side encryption key specified for the log group. If you don't want to encrypt the log data that is sent to CloudWatch Logs, clear the check box. You must also clear the check box if encryption isn't allowed on the log group.
7. 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.
8. Choose **Save**.

For more information about working with CloudWatch Logs, see the [Amazon CloudWatch Logs User Guide](#).

## Session document schema

The following information describes the schema elements of a Session document. AWS Systems Manager Session Manager uses Session documents to determine which type of session to start, such as a standard session, a port forwarding session, or a session to run an interactive command.

### [schemaVersion \(p. 985\)](#)

The schema version of the Session document. Session documents only support version 1.0.

Type: String

Required: Yes

### [description \(p. 985\)](#)

A description you specify for the Session document. For example, "Document to start port forwarding session with Session Manager".

Type: String

Required: No

[sessionType \(p. 985\)](#)

The type of session the Session document is used to establish.

Type: String

Required: Yes

Valid values: `InteractiveCommands` | `NonInteractiveCommands` | `Port` | `Standard_Stream`  
[inputs \(p. 985\)](#)

The session preferences to use for sessions established using this Session document. This element is required for Session documents that are used to create `Standard_Stream` sessions.

Type: `StringMap`

Required: No

[s3BucketName \(p. 985\)](#)

The Amazon Simple Storage Service (Amazon S3) bucket you want to send session logs to at the end of your sessions.

Type: String

Required: No

[s3KeyPrefix \(p. 985\)](#)

The prefix to use when sending logs to the Amazon S3 bucket you specified in the `s3BucketName` input. For more information about using a shared prefix with objects stored in Amazon S3, see [How do I use folders in an S3 bucket?](#) in the *Amazon Simple Storage Service User Guide*.

Type: String

Required: No

[s3EncryptionEnabled \(p. 985\)](#)

If set to `true`, the Amazon S3 bucket you specified in the `s3BucketName` input must be encrypted.

Type: Boolean

Required: Yes

[cloudWatchLogGroupName \(p. 985\)](#)

The name of the Amazon CloudWatch Logs (CloudWatch Logs) group you want to send session logs to at the end of your sessions.

Type: String

Required: No

[cloudWatchEncryptionEnabled \(p. 985\)](#)

If set to `true`, the log group you specified in the `cloudWatchLogGroupName` input must be encrypted.

Type: Boolean

Required: Yes

[cloudWatchStreamingEnabled \(p. 985\)](#)

If set to `true`, a continual stream of session data logs are sent to the log group you specified in the `cloudWatchLogGroupName` input. If set to `false`, session logs are sent to the log group you specified in the `cloudWatchLogGroupName` input at the end of your sessions.

Type: Boolean

Required: Yes

[kmsKeyId \(p. 985\)](#)

The ID of the AWS KMS key you want to use to further encrypt data between your local client machines and the Amazon Elastic Compute Cloud (Amazon EC2) managed nodes you connect to.

Type: String

Required: No

[runAsEnabled \(p. 985\)](#)

If set to `true`, you must specify a user account that exists on the managed nodes you will be connecting to in the `runAsDefaultUser` input. Otherwise, sessions will fail to start. By default, sessions are started using the `ssm-user` account created by the AWS Systems Manager SSM Agent. The Run As feature is only supported for connecting to Linux managed nodes.

Type: Boolean

Required: Yes

[runAsDefaultUser \(p. 985\)](#)

The name of the user account to start sessions with on Linux managed nodes when the `runAsEnabled` input is set to `true`. The user account you specify for this input must exist on the managed nodes you will be connecting to; otherwise, sessions will fail to start.

Type: String

Required: No

[idleSessionTimeout \(p. 985\)](#)

The amount of time of inactivity you want to allow before a session ends. This input is measured in minutes.

Type: String

Valid values: 1-60

Required: No

[maxSessionDuration \(p. 985\)](#)

The maximum amount of time you want to allow before a session ends. This input is measured in minutes.

Type: String

Valid values: 1-1440

Required: No

[shellProfile \(p. 985\)](#)

The preferences you specify per operating system to apply within sessions such as shell preferences, environment variables, working directories, and running multiple commands when a session is started.

Type: StringMap

Required: No

[windows \(p. 985\)](#)

The shell preferences, environment variables, working directories, and commands you specify for sessions on Windows managed nodes.

Type: String

Required: No

[linux \(p. 985\)](#)

The shell preferences, environment variables, working directories, and commands you specify for sessions on Linux managed nodes.

Type: String

Required: No

[parameters \(p. 985\)](#)

An object that defines the parameters the document accepts. For more information about defining document parameters, see [parameters](#) in the [Top-level elements \(p. 1308\)](#). 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. You can reference a Parameter Store parameter using the following format.

```
 {{ssm:parameter-name}}
```

For more information about Parameter Store, see [AWS Systems Manager Parameter Store \(p. 256\)](#).

Type: StringMap

Required: No

[properties \(p. 986\)](#)

An object whose values you specify that are used in the `StartSession` API operation.

For Session documents that are used for `InteractiveCommands` sessions, the properties object includes the commands to run on the operating systems you specify. For more information, see [Restrict access to commands in a session \(p. 952\)](#).

For Session documents that are used for `Port` sessions, the properties object contains the port number where traffic should be redirected to. For an example, see the `Port` type Session document example later in this topic.

Type: StringMap

Required: No

**Standard\_Stream** type Session document example

YAML

```

schemaVersion: '1.0'
description: Document to hold regional settings for Session Manager
sessionType: Standard_Stream
inputs:
 s3BucketName: ''
 s3KeyPrefix: ''
 s3EncryptionEnabled: true
 cloudWatchLogGroupName: ''
 cloudWatchEncryptionEnabled: true
 cloudWatchStreamingEnabled: true
 kmsKeyId: ''
 runAsEnabled: true
 runAsDefaultUser: ''
 idleSessionTimeout: '20'
 maxSessionDuration: '60'
 shellProfile:
 windows: ''
 linux: ''
```

JSON

```
{
 "schemaVersion": "1.0",
 "description": "Document to hold regional settings for Session Manager",
 "sessionType": "Standard_Stream",
 "inputs": {
 "s3BucketName": "",
 "s3KeyPrefix": "",
 "s3EncryptionEnabled": true,
 "cloudWatchLogGroupName": "",
 "cloudWatchEncryptionEnabled": true,
 "cloudWatchStreamingEnabled": true,
 "kmsKeyId": "",
 "runAsEnabled": true,
 "runAsDefaultUser": "",
 "idleSessionTimeout": "20",
 "maxSessionDuration": "60",
 "shellProfile": {
 "windows": "date",
 "linux": "pwd;ls"
 }
 }
}
```

**InteractiveCommands** type Session document example

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"
allowedPattern: "^[a-zA-Z0-9_-]+(.log)$"
properties:
 linux:
 commands: "tail -f {{ logpath }}"
 runAsElevated: true
```

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",
 "allowedPattern": "^[a-zA-Z0-9_-]+(.log)$"
 }
 },
 "properties": {
 "linux": {
 "commands": "tail -f {{ logpath }}",
 "runAsElevated": true
 }
 }
}
```

Port type Session document example

YAML

```

schemaVersion: '1.0'
description: Document to open given port connection over Session Manager
sessionType: Port
parameters:
 paramExample:
 type: string
 description: document parameter
properties:
 portNumber: anyPortNumber
```

JSON

```
{
 "schemaVersion": "1.0",
 "description": "Document to open given port connection over Session Manager",
 "sessionType": "Port",
 "parameters": {
 "paramExample": {
 "type": "string",
 "description": "document parameter"
 }
 },
 "properties": {
 "portNumber": "anyPortNumber"
 }
}
```

}

### Session document example with special characters

YAML

```

schemaVersion: '1.0'
description: Example document with quotation marks
sessionType: InteractiveCommands
parameters:
 Test:
 type: String
 description: Test Input
 maxChars: 32
properties:
 windows:
 commands: |
 $Test = '{{ Test }}'
 $myVariable = \"Computer name is $env:COMPUTERNAME\"
 Write-Host "Test variable: $myVariable`.\`nInput parameter: $Test"
 runAsElevated: false
```

JSON

```
{
 "schemaVersion": "1.0",
 "description": "Test document with quotation marks",
 "sessionType": "InteractiveCommands",
 "parameters": {
 "Test": {
 "type": "String",
 "description": "Test Input",
 "maxChars": 32
 }
 },
 "properties": {
 "windows": {
 "commands": [
 "$Test = '{{ Test }}'",

 "$myVariable = \\\"Computer name is $env:COMPUTERNAME\\\"",

 "Write-Host \"Test variable: $myVariable`.\`nInput parameter: $Test\""
],
 "runAsElevated": false
 }
 }
}
```

## Troubleshooting Session Manager

Use the following information to help you troubleshoot problems with AWS Systems Manager Session Manager.

### Topics

- [No permission to start a session \(p. 988\)](#)
- [No permission to change session preferences \(p. 988\)](#)
- [Managed node not available or not configured for Session Manager \(p. 988\)](#)
- [Session Manager plugin not found \(p. 989\)](#)

- [Session Manager plugin not automatically added to command line path \(Windows\) \(p. 989\)](#)
- [Session Manager plugin becomes unresponsive \(p. 990\)](#)
- [TargetNotConnected \(p. 990\)](#)
- [Blank screen displays after starting a session \(p. 990\)](#)
- [Managed node becomes unresponsive during long running sessions \(p. 991\)](#)

## No permission to start a session

**Problem:** You try to start a session, but the system tells you that you don't have the necessary permissions.

- **Solution:** A system administrator hasn't granted you AWS Identity and Access Management (IAM) policy permissions for starting Session Manager sessions. For information, see [Control user session access to instances \(p. 926\)](#).

## No permission to change session preferences

**Problem:** You try to update global session preferences for your organization, but the system tells you that you don't have the necessary permissions.

- **Solution:** A system administrator hasn't granted you IAM policy permissions for setting Session Manager preferences. For information, see [Grant or deny a user permissions to update Session Manager preferences \(p. 941\)](#).

## Managed node not available or not configured for Session Manager

**Problem 1:** You want to start a session on the **Start a session** console page, but a managed node isn't in the list.

- **Solution A:** The managed node you want to connect to might not have been configured for AWS Systems Manager. For more information, see [Setting up AWS Systems Manager \(p. 16\)](#).

### Note

If AWS Systems Manager SSM Agent is already running on a managed node 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.

- **Solution B:** The proxy configuration you applied to the SSM Agent on your managed node might be incorrect. If the proxy configuration is incorrect, the managed node won't be able to reach the needed service endpoints, or the node might report as a different operating system to Systems Manager. For more information, see [Configuring SSM Agent to use a proxy \(Linux\) \(p. 117\)](#) and [Configure SSM Agent to use a proxy for Windows Server instances \(p. 125\)](#).

**Problem 2:** A managed node you want to connect is in the list on the **Start a session** console page, but the page reports that "The instance you selected isn't configured to use Session Manager."

- **Solution A:** The managed node has been configured for use with the Systems Manager service, but the IAM instance profile attached to the node might not include permissions for the Session Manager capability. For information, see [Verify or Create an IAM Instance Profile with Session Manager Permissions \(p. 920\)](#).
- **Solution B:** The managed node isn't running a version of SSM Agent that supports Session Manager. Update SSM Agent on the node to version 2.3.68.0 or later.

Update SSM Agent manually on a managed node by following the steps in [Manually installing SSM Agent on EC2 instances for Windows Server \(p. 123\)](#), [Manually installing SSM Agent on EC2 instances for Linux \(p. 76\)](#), or [Working with SSM Agent on EC2 instances for macOS \(p. 121\)](#), depending on the operating system.

Alternatively, use the Run Command document [AWS-UpdateSSMAgent](#) to update the agent version on one or more managed nodes at a time. For information, see [Update SSM Agent by using Run Command \(p. 997\)](#).

**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 [Walkthrough: Automatically update SSM Agent \(CLI\) \(p. 1089\)](#).
- Run `AWS-UpdateSSMAgent` as part of a maintenance window. For information about working with maintenance windows, see [Working with maintenance windows \(console\) \(p. 724\)](#) and [Tutorial: Create and configure a maintenance window \(AWS CLI\) \(p. 733\)](#).
- **Solution C:** The managed node can't reach the requisite service endpoints. You can improve the security posture of your managed nodes by using interface endpoints powered by AWS PrivateLink to connect to Systems Manager endpoints. The alternative to using interface endpoints is to allow outbound internet access on your managed nodes. For more information, see [Use PrivateLink to set up a VPC endpoint for Session Manager](#).
- **Solution D:** The managed node has limited available CPU or memory resources. Although your managed node might otherwise be functional, if the node doesn't 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 \(p. 964\)](#).

## 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\`

- **32-bit machines:** C:\Program Files (x86)\Amazon\SessionManagerPlugin\bin\
5. Choose **OK** twice to apply the new settings.
  6. Close any running command prompts and re-open.

## Session Manager plugin becomes unresponsive

During a port forwarding session, traffic might stop forwarding if you have antivirus software installed on your local machine. In some cases, antivirus software interferes with the Session Manager plugin causing process deadlocks. To resolve this issue, allow or exclude the Session Manager plugin from the antivirus software. For information about the default installation path for the Session Manager plugin, see [\(Optional\) Install the Session Manager plugin for the AWS CLI \(p. 964\)](#).

## 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* isn't connected."

- **Solution A:** This error is returned when the specified target managed node for the session isn't fully configured for use with Session Manager. For information, see [Setting up Session Manager \(p. 916\)](#).
- **Solution B:** This error is also returned if you attempt to start a session on a managed node that is located in a different AWS account or AWS Region.

## 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 managed node is full. Due to lack of disk space, SSM Agent on the node 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 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.

```
https://us-west-2.console.aws.amazon.com/systems-manager/session-manager/sessions?
region=us-west-1
```

- **Solution C:** The managed node is connecting to Systems Manager using VPC endpoints, and your Session Manager preferences write session output to an Amazon S3 bucket or Amazon CloudWatch Logs log group, but an s3 gateway endpoint or logs interface endpoint doesn't exist in the VPC. An s3 endpoint in the format **com.amazonaws.region.s3** is required if your managed nodes are connecting to Systems Manager using VPC endpoints, and your Session Manager preferences write session output to an Amazon S3 bucket. Alternatively, a logs endpoint in the format **com.amazonaws.region.logs** is required if your managed nodes are connecting to Systems Manager using VPC endpoints, and your Session Manager preferences write session output to a CloudWatch Logs log group. For more information, see [Creating VPC endpoints for Systems Manager \(p. 30\)](#).
- **Solution D:** The log group or Amazon S3 bucket you specified in your session preferences has been deleted. To resolve this issue, update your session preferences with a valid log group or S3 bucket.
- **Solution E:** The log group or Amazon S3 bucket you specified in your session preferences isn't encrypted, but you have set the `cloudWatchEncryptionEnabled` or `s3EncryptionEnabled` input to `true`. To resolve this issue, update your session preferences with a log group or Amazon S3 bucket.

that is encrypted, or set the `cloudWatchEncryptionEnabled` or `s3EncryptionEnabled` input to `false`. This scenario is only applicable to customers who create session preferences using command line tools.

## Managed node becomes unresponsive during long running sessions

**Problem:** Your managed node becomes unresponsive or crashes during a long running session.

**Solution:** Decrease the SSM Agent log retention duration for Session Manager.

### To decrease the SSM Agent log retention duration for sessions

1. Locate the `amazon-ssm-agent.json.template` in the `/etc/amazon/ssm/` directory for Linux, or `C:\Program Files\Amazon\SSM` for Windows.
2. Copy the contents of the `amazon-ssm-agent.json.template` to a new file in the same directory named `amazon-ssm-agent.json`.
3. Decrease the default value of the `SessionLogsRetentionDurationHours` value in the `SSM` property, and save the file.
4. Restart the SSM Agent.

# AWS Systems Manager Run Command

Using Run Command, a capability of AWS Systems Manager, you can remotely and securely manage the configuration of your managed nodes. A *managed node* is any Amazon Elastic Compute Cloud (Amazon EC2) instance, edge device, or on-premises server or virtual machine (VM) in your hybrid environment that has been configured for Systems Manager. Run Command allows you to automate common administrative tasks and perform one-time configuration changes at scale. You can use Run Command from the AWS Management Console, the AWS Command Line Interface (AWS CLI), AWS Tools for Windows PowerShell, or the AWS SDKs. Run Command is offered at no additional cost. To get started with Run Command, open the [Systems Manager console](#). In the navigation pane, choose **Run Command**.

Administrators use Run Command to install or bootstrap applications, build a deployment pipeline, capture log files when an instance is removed from an Auto Scaling group, join instances to a Windows domain, and more.

### Getting Started

The following table includes information to help you get started with Run Command.

Topic	Details
<a href="#">Systems Manager prerequisites (p. 13)</a>	(Required) Verify that your managed nodes meet the minimum requirements for Run Command, configure required roles, and install the SSM Agent.
<a href="#">Setting up AWS Systems Manager for hybrid environments (p. 34)</a>	(Optional) Register on-premises servers and VMs with AWS so you can manage them using Run Command.
<a href="#">the section called "Setting up edge devices" (p. 52)</a>	(Optional) Configure edge devices so you can manage them using Run Command.

Topic	Details
<a href="#">Sending commands using Systems Manager Run Command (p. 995)</a>	Learn how to run a command that targets one or more managed nodes by using the AWS Management Console.
<a href="#">Run Command walkthroughs (p. 1019)</a>	Learn how to run commands using either Tools for Windows PowerShell or the AWS CLI.

### EventBridge support

This Systems Manager capability is supported as both an *event* type and a *target* type in Amazon EventBridge rules. For information, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#) and [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#).

### Related content

- [Remotely Run Command 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 \(p. 992\)](#)
- [Sending commands using Systems Manager Run Command \(p. 995\)](#)
- [Handling exit codes with scripts \(p. 1011\)](#)
- [Understanding command statuses \(p. 1013\)](#)
- [Run Command walkthroughs \(p. 1019\)](#)
- [Troubleshooting Systems Manager Run Command \(p. 1033\)](#)

## Setting up Run Command

Before you can manage nodes by using Run Command, a capability of AWS Systems Manager, 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 \(p. 16\)](#).

You must also configure your nodes for Systems Manager. For more information, see [Setting up AWS Systems Manager \(p. 16\)](#).

We recommend completing the following optional setup tasks to help minimize the security posture and day-to-day management of your managed nodes.

#### Monitor command executions using Amazon EventBridge

You can use EventBridge 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 an EventBridge event occurs. For more information, see [Configuring EventBridge for Systems Manager events \(p. 1450\)](#).

#### Monitor command executions using Amazon CloudWatch Logs

You can configure Run Command to periodically send all command output and error logs to an Amazon CloudWatch 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 \(p. 1447\)](#).

#### Restrict Run Command access to specific managed nodes

You can restrict a user's ability to run commands on managed nodes by using AWS Identity and Access Management (IAM). Specifically, you can create an IAM user policy with a condition that the user can only run commands on managed nodes that are tagged with specific tags. For more information, see [Restricting Run Command access based on tags \(p. 993\)](#).

## Restricting Run Command access based on tags

This section describes how to restrict a user's ability to run commands on managed instances by specifying a tag condition in an IAM policy. Managed instances include Amazon EC2 instances and on-premises servers, edge devices, and VMs in a hybrid environment that are configured for Systems Manager. Though the information is not explicitly presented, you can also restrict access to managed AWS IoT Greengrass core devices. To get started, you must tag your AWS IoT Greengrass devices. For more information, see [Tag your AWS IoT Greengrass Version 2 resources](#) in the *AWS IoT Greengrass Version 2 Developer Guide*.

You can restrict command execution to specific managed nodes by creating an IAM user policy that includes a condition that the user can only run commands on nodes with specific 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 node (`Resource: arn:aws:ec2:*:*:instance/*`) with the condition that the node is a Finance WebServer (`ssm:resourceTag/Finance: WebServer`). If the user sends a command to a node that isn't tagged or that has any tag other than `Finance: WebServer`, the execution results show `AccessDenied`.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:SendCommand"
],
 "Resource": [
 "arn:aws:ssm:*:*:document/*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:SendCommand"
],
 "Resource": [
 "arn:aws:ec2:*:*:instance/*"
],
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/Finance": [
 "WebServers"
]
 }
 }
 }
]
}
```

You can create IAM policies that allow a user to run commands on managed nodes that are tagged with multiple tags. The following policy allows the user to run commands on managed nodes that have two

tags. If a user sends a command to a node that isn't tagged with both of these tags, the execution results show `AccessDenied`.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:SendCommand"
],
 "Resource": "*",
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/tag_key1": [
 "tag_value1"
],
 "ssm:resourceTag/tag_key2": [
 "tag_value2"
]
 }
 }
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:SendCommand"
],
 "Resource": [
 "arn:aws:ssm:us-west-1::document/AWS-*",
 "arn:aws:ssm:us-east-2::document/AWS-*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:UpdateInstanceInformation",
 "ssm>ListCommands",
 "ssm>ListCommandInvocations",
 "ssm:GetDocument"
],
 "Resource": "*"
 }
]
}
```

You can also create IAM policies that allows a user to run commands on multiple groups of tagged managed nodes. The following example policy allows the user to run commands on either group of tagged nodes, or both groups.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:SendCommand"
],
 "Resource": "*",
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/tag_key1": [
 "tag_value1"
]
 }
 }
 }
]
}
```

```
 }
 },
{
 "Effect": "Allow",
 "Action": [
 "ssm:SendCommand"
],
 "Resource": "*",
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/tag_key2": [
 "tag_value2"
]
 }
 }
},
{
 "Effect": "Allow",
 "Action": [
 "ssm:SendCommand"
],
 "Resource": [
 "arn:aws:ssm:us-west-1::document/AWS-*",
 "arn:aws:ssm:us-east-2::document/AWS-*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "ssm:UpdateInstanceInformation",
 "ssm>ListCommands",
 "ssm>ListCommandInvocations",
 "ssm:GetDocument"
],
 "Resource": "*"
}
]
```

For more information about creating IAM user policies, see [Managed policies and inline policies](#) in the *IAM User Guide*. For more information about tagging managed nodes, see [Tag Editor](#) in the *AWS Resource Groups User Guide*.

## Sending commands using Systems Manager Run Command

This section includes information about how to send commands from the AWS Systems Manager console to managed nodes. 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 \(p. 1025\)](#) 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 Command Line Interface (AWS CLI), see the [Walkthrough: Use the AWS CLI with Run Command \(p. 1019\)](#) or the examples in the [SSM CLI Reference](#).

### Important

When you send a command using Run Command, don't include sensitive information formatted as plaintext, such as passwords, configuration data, or other secrets. All Systems Manager API activity in your account is logged in an Amazon S3 bucket for AWS CloudTrail logs. This means that any user with access to S3 bucket can view the plaintext values of those secrets. For this

reason, we recommend creating and using `SecureString` parameters to encrypt sensitive data you use in your Systems Manager operations.

For more information, see [Restricting access to Systems Manager parameters using IAM policies \(p. 260\)](#).

## Contents

- [Running commands from the console \(p. 996\)](#)
- [Running PowerShell scripts on Linux managed nodes \(p. 1000\)](#)
- [Running commands using the document version parameter \(p. 1002\)](#)
- [Using targets and rate controls to send commands to a fleet \(p. 1003\)](#)
- [Canceling a command \(p. 1010\)](#)

## Running commands from the console

You can use Run Command, a capability of AWS Systems Manager, from the AWS Management Console to configure managed nodes without having to log into them. This topic includes an example that shows how to [update SSM Agent \(p. 997\)](#) on a managed node by using Run Command.

### Before you begin

Before you send a command using Run Command, verify that your managed nodes meet Systems Manager [requirements \(p. 13\)](#).

### To send a command using Run Command

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

#### Note

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Other parameters**:

- For **Comment**, enter information about this command.
- For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

#### Note

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then

restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

11. Choose **Run**.

For information about canceling a command, see [the section called “Canceling a command” \(p. 1010\)](#).

## Rerunning commands

Systems Manager includes two options to help you rerun a command from the **Run Command** page in the Systems Manager console.

- **Rerun:** This button allows 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 ran 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 update the SSM Agent running on your managed nodes. You can update to either the latest version of SSM Agent 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.

To be notified about SSM Agent updates, subscribe to the [SSM Agent Release Notes](#) page on GitHub.

### To update SSM Agent using Run Command

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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**, enter the version of SSM Agent to install. You can install [older versions](#) of the agent. If you don't 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, specify the [earlier](#) version number. Choose **false** to install only the newest version of the service.
6. In the **Targets** section, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

#### Note

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Other parameters**:
  - For **Comment**, enter information about this command.
  - For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.
8. For **Rate control**:
  - For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

#### Note

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role](#)

for a hybrid environment (p. 36). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

11. Choose **Run**.

## Update PowerShell using Run Command

The following procedure describes how to update PowerShell to version 5.1 on your Windows Server 2012 and 2012 R2 managed nodes. The script provided in this procedure downloads the Windows Management Framework (WMF) version 5.1 update, and starts the installation of the update. The node reboots during this process because this is required when installing WMF 5.1. The download and installation of the update takes approximately five minutes to complete.

### To update PowerShell using Run Command

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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-RunPowerShellScript**.
5. In the **Commands** section, paste the following commands for your operating system.

Windows Server 2012 R2

```
Set-Location -Path "C:\Windows\Temp"

Invoke-WebRequest "https://go.microsoft.com/fwlink/?linkid=839516" -OutFile
"Win8.1AndW2K12R2-KB3191564-x64.msu"

Start-Process -FilePath "$env:systemroot\system32\wusa.exe" -Verb RunAs -
ArgumentList ('Win8.1AndW2K12R2-KB3191564-x64.msu', '/quiet')
```

Windows Server 2012

```
Set-Location -Path "C:\Windows\Temp"

Invoke-WebRequest "https://go.microsoft.com/fwlink/?linkid=839513" -OutFile "W2K12-
KB3191565-x64.msu"

Start-Process -FilePath "$env:systemroot\system32\wusa.exe" -Verb RunAs -
ArgumentList ('W2K12-KB3191565-x64.msu', '/quiet')
```

6. In the **Targets** section, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

#### Note

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Other parameters**:

- For **Comment**, enter information about this command.
- For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

11. Choose **Run**.

After the managed node reboots and the installation of the update is complete, connect to your node to confirm that PowerShell successfully upgraded to version 5.1. To check the version of PowerShell on your node, open PowerShell and enter `$PSVersionTable`. The `PSVersion` value in the output table shows 5.1 if the upgrade was successful.

If the `PSVersion` value is different than 5.1, for example 3.0 or 4.0, review the **Setup** logs in Event Viewer under **Windows Logs**. These logs indicate why the update installation failed.

## Running PowerShell scripts on Linux managed nodes

Using the `aws:runPowerShellScript` plugin or the `AWS-RunPowerShellScript` command document, along with PowerShell Core, you can run PowerShell scripts on Linux managed nodes. 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 managed node and follow the [PowerShell Core](#) installation procedure for the appropriate operating system.

Many PowerShell commands (cmdlets) aren't available on Linux. To see which commands are available, use the `Get-Command` cmdlet after starting PowerShell using the `pwsh` command on your Linux managed node. For more information, see [Get-Command](#).

The following procedure describes how to run a PowerShell script on a Linux managed node using the console.

### To run a PowerShell script on a Linux managed node using the console

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Other parameters**:

- For **Comment**, enter information about this command.
- For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

11. Choose **Run**.

To see examples that use the `aws:runPowerShellScript` plugin, see [aws:runPowerShellScript \(p. 1341\)](#).

## Running commands using the document version parameter

You can use the document version parameter to specify which version of an AWS Systems Manager document to use when the command runs. You can specify one of the following options for this parameter:

- `$DEFAULT`
- `$LATEST`
- Version number

Run the following procedure to run a command using the document version parameter.

Linux

### To run commands using the AWS CLI on local Linux machines

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. List all available documents

This command lists all of the documents available for your account based on AWS Identity and Access Management (IAM) permissions.

```
aws ssm list-documents
```

3. Run the following command to view the different versions of a document.

```
aws ssm list-document-versions \
--name "document-name"
```

4. Run 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 local Windows machines

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. List all available documents

This command lists all of the documents available for your account based on AWS Identity and Access Management (IAM) permissions.

```
aws ssm list-documents
```

3. Run the following command to view the different versions of a document.

```
aws ssm list-document-versions ^
--name "document-name"
```

4. Run 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 Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. List all available documents

This command lists all of the documents available for your account based on AWS Identity and Access Management (IAM) permissions.

```
Get-SSMDocumentList
```

3. Run the following command to view the different versions of a document.

```
Get-SSMDocumentVersionList ^
-Name "document-name"
```

4. Run 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 use Run Command, a capability of AWS Systems Manager, to send commands to tens, hundreds, or thousands of managed nodes by using the `targets` parameter. The `targets` parameter accepts a Key,Value combination based on tags that you specified for your managed nodes. When you run the command, the system locates and attempts to run the command on all managed nodes that match the specified tags. For more information about tagging managed instances, see [Tag Editor](#) in the [AWS Resource Groups User Guide](#). For information about tagging your managed IoT devices, see [Tag your AWS IoT Greengrass Version 2 resources](#) in the [AWS IoT Greengrass Version 2 Developer Guide](#).

You can also use the `targets` parameter to target a list of specific managed node IDs, as described in the next section.

To control how commands run across hundreds or thousands of managed nodes, Run Command also includes parameters for restricting how many nodes can simultaneously process a request and how many errors can be thrown by a command before the command is canceled.

### Contents

- [Targeting multiple managed nodes \(p. 1004\)](#)
- [Using rate controls \(p. 1008\)](#)

## Targeting multiple managed nodes

You can run a command and target managed nodes by specifying tags, AWS resource group names, or managed node IDs.

The following examples show the command format when using Run Command from the AWS Command Line Interface (AWS CLI). Sample commands in this section are truncated using [ ... ].

### Example 1: Targeting tags

Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--targets Key=tag:tag-name,Values=tag-value \
[...]
```

Windows

```
aws ssm send-command ^
--document-name document-name ^
--targets Key=tag:tag-name,Values=tag-value ^
[...]
```

### Example 2: Targeting an AWS resource group by name

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 AWS Identity and Access Management (IAM) permissions to list or view the resources that belong to that group. For more information, see [Set up permissions](#) in the *AWS Resource Groups User Guide*.

Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--targets Key=resource-groups:Name,Values=resource-group-name \
[...]
```

Windows

```
aws ssm send-command ^
```

```
--document-name document-name ^
--targets Key=resource-groups:Name,Values=resource-group-name ^
[...]
```

### Example 3: Targeting an AWS resource group by resource type

You can specify a maximum of five resource group types 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](#) in the *AWS Resource Groups User Guide*.

#### Linux & macOS

```
aws ssm send-command \
 --document-name document-name \
 --targets Key=resource-groups:ResourceTypeFilters,Values=resource-type-1,resource-type-2 \
 [...]
```

#### Windows

```
aws ssm send-command ^
 --document-name document-name ^
 --targets Key=resource-groups:ResourceTypeFilters,Values=resource-type-1,resource-type-2 ^
 [...]
```

### Example 4: Targeting instance IDs

The following examples show how to target managed nodes by using the `instanceids` key with the `targets` parameter. You can use this key to target managed AWS IoT Greengrass core devices because each device is assigned an [\*mi-ID number\*](#). You can view device IDs in Fleet Manager, a capability of AWS Systems Manager.

#### Linux & macOS

```
aws ssm send-command \
 --document-name document-name \
 --targets Key=instanceids,Values=instance-ID-1,instance-ID-2,instance-ID-3 \
 [...]
```

#### Windows

```
aws ssm send-command ^
 --document-name document-name ^
 --targets Key=instanceids,Values=instance-ID-1,instance-ID-2,instance-ID-3 ^
 [...]
```

If you tagged managed nodes 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 managed nodes in *one* of these environments by using the `targets` parameter with the following syntax.

Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--targets Key=tag:Environment,Values=Development \
[...]
```

Windows

```
aws ssm send-command ^
--document-name document-name ^
--targets Key=tag:Environment,Values=Development ^
[...]
```

You could target additional managed nodes in other environments by adding to the `Values` list. Separate items using commas.

Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--targets Key=tag:Environment,Values=Development,Test,Pre-production \
[...]
```

Windows

```
aws ssm send-command ^
--document-name document-name ^
--targets Key=tag:Environment,Values=Development,Test,Pre-production ^
[...]
```

**Variation:** 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 managed nodes that meet *all* of the criteria. The following command targets all managed nodes tagged for the Finance Department *and* tagged for the database server role.

Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--targets Key=tag:Department,Values=Finance Key=tag:ServerRole,Values=Database \
[...]
```

Windows

```
aws ssm send-command ^
--document-name document-name ^
--targets Key=tag:Department,Values=Finance Key=tag:ServerRole,Values=Database ^
[...]
```

**Variation:** 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.

## Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--targets Key=tag:Department,Values=Finance,Marketing
Key=tag:ServerRole,Values=WebServer,Database \
[...]
```

## Windows

```
aws ssm send-command ^
--document-name document-name ^
--targets Key=tag:Department,Values=Finance,Marketing
Key=tag:ServerRole,Values=WebServer,Database ^
[...]
```

### Variation: Targeting tagged managed nodes using multiple values criteria

If you tagged managed nodes for different environments using a Key named `Department` and Values of `Sales` and `Finance`, then you could send a command to all of the nodes in these environments by using the `targets` parameter with the following syntax.

## Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--targets Key=tag:Department,Values=Sales,Finance \
[...]
```

## Windows

```
aws ssm send-command ^
--document-name document-name ^
--targets Key=tag:Department,Values=Sales,Finance ^
[...]
```

You can specify a maximum of five keys, and five values for each key.

If either a tag key (the tag name) or a tag value includes spaces, enclose the tag key or the value in quotation marks, as shown in the following examples.

### Example: Spaces in value tag

## Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--targets Key=tag:OS,Values="Windows Server 2016 Nano" \
[...]
```

## Windows

```
aws ssm send-command ^
--document-name document-name ^
--targets Key=tag:OS,Values="Windows Server 2016 Nano" ^
[...]
```

**Example:** Spaces in tag key and value

Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--targets Key="tag:Operating System",Values="Windows Server 2016 Nano" \
[...]
```

Windows

```
aws ssm send-command ^
--document-name document-name ^
--targets Key="tag:Operating System",Values="Windows Server 2016 Nano" ^
[...]
```

**Example:** Spaces in one item in a list of values

Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--targets Key=tag:Department,Values="Sales", "Finance", "Systems Mgmt" \
[...]
```

Windows

```
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 managed nodes in a group by using *concurrency controls* and *error controls*.

### Topics

- [Using concurrency controls \(p. 1008\)](#)
- [Using error controls \(p. 1009\)](#)

### Using concurrency controls

You can control the number of managed nodes that run a command simultaneously by using the `max-concurrency` parameter (the **Concurrency** options in the [Run a command](#) page). You can specify either an absolute number of managed nodes, for example `10`, or a percentage of the target set, for example `10%`. The queueing system delivers the command to a single node and waits until the system acknowledges the initial invocation before sending the command to two more nodes. The system exponentially sends commands to more nodes until the system meets the value of `max-concurrency`. The default for value `max-concurrency` is `50`. The following examples show you how to specify values for the `max-concurrency` parameter.

Linux & macOS

```
aws ssm send-command \
```

```
--document-name document-name \
--max-concurrency 10 \
--targets Key=tag:Environment,Values=Development \
[...]
```

```
aws ssm send-command \
--document-name document-name \
--max-concurrency 10% \
--targets Key=tag:Department,Values=Finance,Marketing
Key=tag:ServerRole,Values=WebServer,Database \
[...]
```

## Windows

```
aws ssm send-command ^
--document-name document-name ^
--max-concurrency 10 ^
--targets Key=tag:Environment,Values=Development ^
[...]
```

```
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 managed nodes 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 managed nodes. 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 managed nodes after the first error result is returned. If you send a command to 50 managed nodes and set `max-errors` to **10%**, then the system stops sending the command to additional nodes 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 might 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.

## Linux & macOS

```
aws ssm send-command \
--document-name document-name \
--max-errors 10 \
--targets Key=tag:Database,Values=Development \
[...]
```

```
aws ssm send-command \
--document-name document-name \
--max-errors 10% \
--targets Key=tag:Environment,Values=Development \
```

[...]

```
aws ssm send-command \
--document-name document-name \
--max-concurrency 1 \
--max-errors 1 \
--targets Key=tag:Environment,Values=Production \
[...]
```

## Windows

```
aws ssm send-command ^
--document-name document-name ^
--max-errors 10 ^
--targets Key=tag:Database,Values=Development ^
[...]
```

```
aws ssm send-command ^
--document-name document-name ^
--max-errors 10% ^
--targets Key=tag:Environment,Values=Development ^
[...]
```

```
aws ssm send-command ^
--document-name document-name ^
--max-concurrency 1 ^
--max-errors 1 ^
--targets Key=tag:Environment,Values=Production ^
[...]
```

## Cancelling a command

You can attempt to cancel a command as long as the service shows that it's in either a Pending or Executing state. However, even if a command is still in one of these states, we can't guarantee that the command will be canceled and the underlying process stopped.

### To cancel a command using the console

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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

Run the following command.

Linux & macOS

```
aws ssm cancel-command \
```

```
--command-id "command-ID" \
--instance-ids "instance-ID"
```

#### Windows

```
aws ssm cancel-command ^
--command-id "command-ID" ^
--instance-ids "instance-ID"
```

For information about the status of a canceled command, see [Understanding command statuses \(p. 1013\)](#).

## Handling exit codes with scripts

In some cases, you might need to manage how exit codes are handled in your commands by using scripts.

#### Topics

- [Rebooting managed nodes from scripts \(p. 1011\)](#)
- [Managing exit codes in Run Command commands \(p. 1012\)](#)

## Rebooting managed nodes from scripts

If you use Run Command, a capability of AWS Systems Manager, to run scripts that reboot managed nodes, we recommend that you specify an exit code in your script. If you attempt to reboot a node 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 nodes, you specify `exit 3010` in your script. For Linux and macOS managed nodes, you specify `exit 194`. The exit code instructs AWS Systems Manager Agent (SSM Agent) to reboot the managed node, 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.

#### Note

The reboot script can't be part of an `aws:runDocument` plugin. If a document contains the reboot script and another document tries to run that document through the `aws:runDocument` plugin, SSM Agent returns an error.

#### Create idempotent scripts

When developing scripts that reboot managed nodes, 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's only intended to run once.

Here is an outline example of an idempotent script that reboots a managed node multiple times.

```
$name = Get current computer name
If ($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
}
```

## Examples

The following script samples use exit codes to restart managed nodes. The Linux example installs package updates on Amazon Linux, and then restarts the node. The Windows example installs the Telnet-Client on the node, and then restarts it.

### Amazon Linux

```
#!/bin/bash
yum -y update
needs-restarting -r
if [$? -eq 1]
then
 exit 194
else
 exit 0
fi
```

### Windows

```
$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

Using Run Command, a capability of AWS Systems Manager, you can 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 script at the first command 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

Run Command, a capability of AWS Systems Manager, reports detailed status information about the different states a command experiences during processing and for each managed node that processed the command. You can monitor command statuses using the following methods:

- Choose the **Refresh** icon on the **Run Command** page in the Amazon Elastic Compute Cloud (Amazon EC2) console.
- Call [list-commands](#) or [list-command-invocations](#) using the AWS Command Line Interface (AWS CLI). Or call [Get-SSMCommand](#) or [Get-SSMCommandInvocation](#) using AWS Tools for Windows PowerShell.
- Configure Amazon EventBridge to respond to state or status changes.
- Configure Amazon Simple Notification Service (Amazon SNS) to send notifications for all status changes or specific statuses such as 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 SSM document. For more information about plugins, see [Systems Manager Command document plugin reference \(p. 1314\)](#).

When you send a command to multiple managed nodes at the same time, each copy of the command targeting each node is a *command invocation*. For example, if you use the `AWS-RunShellScript` document and send an `ifconfig` command to 20 Linux 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 managed nodes 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 \(p. 1003\)](#).

### Detailed status for command plugins and invocations

Status	Details
Pending	The command hasn't yet been sent to the managed node or hasn't been received by SSM Agent. If the command isn't received by the agent before the length of time passes that is equal to the sum of the <b>Timeout (seconds)</b> parameter and the <b>Execution timeout</b> parameter, the status changes to <b>Delivery Timed Out</b> .
In Progress	Systems Manager is attempting to send the command to the managed node, or the command was received by SSM Agent and has started running on the instance. Depending on the result of all command plugins, the status changes to <b>Success</b> , <b>Failed</b> , <b>Delivery Timed Out</b> , or <b>Execution Timed Out</b> . <i>Exception:</i> If the agent isn't running or available on the node, the command status remains at <b>In Progress</b> until the agent is available again, or until the execution timeout limit is reached. The status then changes to a terminal state.
Delayed	The system attempted to send the command to the managed node but wasn't successful. The system retries again.
Success	The command was received by SSM Agent on the managed node and returned an exit code of zero. This status <i>doesn't</i> mean the command was processed on the node. This is a terminal state. <p><b>Note</b></p> <p>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).</p>
Delivery Timed Out	The command wasn't delivered to the managed node before the total timeout expired. Total timeouts don't count against the parent command's <code>max-errors</code> limit, but they do contribute to whether the parent command status is <b>Success</b> , <b>Incomplete</b> , or <b>Delivery Timed Out</b> . This is a terminal state.
Execution Timed Out	Command automation started on the managed node, but the command wasn't completed before the execution timeout expired. Execution timeouts

Status	Details
	count as a failure, which will send a non-zero reply and Systems Manager will exit the attempt to run the command automation, and report a failure status.
Failed	The command wasn't successful on the managed node. For a plugin, this indicates that the result code wasn't zero. For a command invocation, this indicates that the result code for one or more plugins wasn't zero. Invocation failures count against the <code>max-errors</code> limit of the parent command. This is a terminal state.
Canceled	The command was canceled before it was completed. This is a terminal state.
Undeliverable	The command can't be delivered to the managed node. The node might not exist or it might not be responding. Undeliverable invocations don't count against the parent command's <code>max-errors</code> limit, but they do contribute to whether the parent command status is Success or Incomplete. For example, if all invocations in a command have the status Undeliverable, then the command status returned is Failed. However, if a command has five invocations, four of which return the status Undeliverable and one of which returns the status Success, then the parent command's status is Success. This is a terminal state.
Terminated	The parent command exceeded its <code>max-errors</code> limit and subsequent command invocations were canceled by the system. This is a terminal state.
Invalid Platform	The command was sent to a managed node that didn't match the required platforms specified by the chosen document. Invalid Platform doesn't count against the parent command's <code>max-errors</code> limit, but it does contribute to whether the parent command status is Success or Failed. For example, if all invocations in a command have the status Invalid Platform, then the command status returned is Failed. However, if a command has five invocations, four of which return the status Invalid Platform and one of which returns the status Success, then the parent command's status is Success. This is a terminal state.

Status	Details
Access Denied	The AWS Identity and Access Management (IAM) user or role initiating the command doesn't have access to the targeted managed node. <code>Access Denied</code> doesn't count against the parent command's <code>max-errors</code> limit, but it 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>Access Denied</code> , then the command status returned is <code>Failed</code> . However, if a command has five invocations, four of which return the status <code>Access Denied</code> and one of which returns the status <code>Success</code> , then the parent command's status is <code>Success</code> . This is a terminal state.

### Detailed status for a command

Status	Details
Pending	The command wasn't yet received by an agent on any managed nodes.
In Progress	The command has been sent to at least one managed node but hasn't reached a final state on all nodes.
Delayed	The system attempted to send the command to the node but wasn't successful. The system retries again.
Success	The command was received by SSM Agent on all specified or targeted managed nodes and returned an exit code of zero. All command invocations have reached a terminal state, and the value of <code>max-errors</code> wasn't reached. This status <i>doesn't</i> mean the command was successfully processed on all specified or targeted managed nodes. This is a terminal state. <p><b>Note</b>            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).</p>
Delivery Timed Out	The command wasn't delivered to the managed node before the total 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.
Failed	The command wasn't successful on the managed node. The value of <code>max-errors</code> or more

Status	Details
	command invocations shows a status of Failed. This is a terminal state.
Incomplete	The command was attempted on all managed nodes and one or more of the invocations doesn't have a value of Success. However, not enough invocations failed for the status to be Failed. This is a terminal state.
Canceled	The command was canceled before it was completed. This is a terminal state.
Rate Exceeded	The number of managed nodes targeted by the command exceeded the account quota for pending invocations. The system has canceled the command before executing it on any node. This is a terminal state.
Access Denied	The IAM user or role initiating the command doesn't have access to the targeted resource group. AccessDenied doesn't 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.
No Instances In Tag	The tag key-pair value or resource group targeted by the command doesn't match any managed nodes. This is a terminal state.

## Understanding command timeout values

Systems Manager enforces the following timeout values when running commands.

### Total Timeout

In the Systems Manager console, you specify the timeout value in the **Timeout (seconds)** field. After a command is sent, Run Command checks whether the command has expired or not. If a command reaches the command expiration limit (total timeout), it changes status to `DeliveryTimedOut` for all invocations that have the status `InProgress`, `Pending` or `Delayed`.



On a more technical level, total timeout (**Timeout (seconds)**) is a combination of two timeout values, as shown here:

Total timeout = "Timeout(seconds)" from the console + "timeoutSeconds": "{{ executionTimeout }}" from your SSM document

For example, the default value of **Timeout (seconds)** in the Systems Manager console is 600 seconds. If you run a command by using the AWS-RunShellScript SSM document, the default value of "timeoutSeconds": "{{ executionTimeout }}" is 3600 seconds, as shown in the following document sample:

```
"executionTimeout": {
 "type": "String",
 "default": "3600",

 "runtimeConfig": {
 "aws:runShellScript": {
 "properties": [
 {
 "timeoutSeconds": "{{ executionTimeout }}"
```

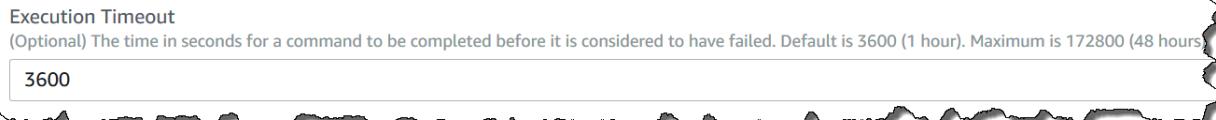
This means the command runs for 4,200 seconds (70 minutes) before the system sets the command status to `DeliveryTimedOut`.

### 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.

#### Note

Run Command relies on the SSM Agent document terminal response to determine whether or not the command was delivered to the agent. SSM Agent must send an `ExecutionTimedOut` signal for an invocation or command to be marked as `ExecutionTimedOut`.



### Default Execution Timeout

If a SSM document doesn't require that you explicitly specify an execution timeout value, then Systems Manager enforces the hard-coded 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 Run Command doesn't receive a document terminal response from SSM Agent, the command invocation is marked as `deliveryTimeout`.

To determine timeout status on a target, SSM Agent combines all 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.

The default for **Timeout (seconds)** is 3600 seconds. The default for **Execution Timeout** is also 3600 seconds. Therefore, the total default timeout for a command is 7200 seconds.

#### 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, a capability of AWS Systems Manager, using either the AWS Command Line Interface (AWS CLI) or AWS Tools for Windows PowerShell.

## Contents

- [Walkthrough: Use the AWS CLI with Run Command \(p. 1019\)](#)
- [Walkthrough: Use the AWS Tools for Windows PowerShell with Run Command \(p. 1025\)](#)

You can also view sample commands in the following references.

- [Systems Manager AWS CLI Reference](#)
- [AWS Tools for Windows PowerShell - AWS Systems Manager](#)

## Walkthrough: Use the AWS CLI with Run Command

The following sample walkthrough shows you how to use the AWS Command Line Interface (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 AWS Systems Manager pre-configured documents shown in this topic. The commands or scripts specified in Systems Manager documents run with administrative permissions on your managed nodes. 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 node. For all other users, you should create restrictive documents and share them with specific users. For more information about restricting access to Run Command, a capability of AWS Systems Manager, see [Create non-Admin IAM users and groups for Systems Manager \(p. 18\)](#).

### Topics

- [Step 1: Getting started \(p. 1019\)](#)
- [Step 2: Run shell scripts to view resource details \(p. 1020\)](#)
- [Step 3: Send simple commands using the AWS-RunShellScript document \(p. 1021\)](#)
- [Step 4: Run a simple Python script using Run Command \(p. 1023\)](#)
- [Step 5: Run a Bash script using Run Command \(p. 1023\)](#)

### Step 1: Getting started

You must either have administrator permissions on the managed node you want to configure or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM). Also note, this example uses the US East (Ohio) Region (`us-east-2`). Run Command is 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 \(p. 13\)](#).

#### To run commands using the AWS CLI

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
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. Verify that a managed node is ready to receive commands.

The output of the following command shows if managed nodes are online.

Linux & macOS

```
aws ssm describe-instance-information \
--output text --query "InstanceInformationList[*]"
```

Windows

```
aws ssm describe-instance-information ^
--output text --query "InstanceInformationList[*]"
```

4. Run the following command to view details about a particular managed node.

**Note**

To run the commands in this walkthrough, replace the instance and command IDs. For managed AWS IoT Greengrass core devices, use the `mi-ID_number` for instance ID. The command ID is returned as a response to **send-command**. Instance IDs are available from Fleet Manager, a capability of AWS Systems Manager..

Linux & macOS

```
aws ssm describe-instance-information \
--instance-information-filter-list key=InstanceIds,valueSet=instance-ID
```

Windows

```
aws ssm describe-instance-information ^
--instance-information-filter-list key=InstanceIds,valueSet=instance-ID
```

## Step 2: Run shell scripts to view resource details

Using Run Command and the `AWS-RunShellScript` document, you can run any command or script on a managed node as if you were logged on locally.

### View the description and available parameters

Run the following command to view a description of the Systems Manager JSON document.

Linux & macOS

```
aws ssm describe-document \
--name "AWS-RunShellScript" \
--query "[Document.Name,Document.Description]"
```

Windows

```
aws ssm describe-document ^
--name "AWS-RunShellScript" ^
--query "[Document.Name,Document.Description]"
```

Run the following command to view the available parameters and details about those parameters.

Linux & macOS

```
aws ssm describe-document \
--name "AWS-RunShellScript" \
--query "Document.Parameters[*]"
```

Windows

```
aws ssm describe-document ^
--name "AWS-RunShellScript" ^
--query "Document.Parameters[*]"
```

### Step 3: Send simple commands using the AWS-RunShellScript document

Run the following command to get IP information for a Linux managed node.

If you're targeting a Windows Server managed node, change the document-name to AWS-RunPowerShellScript and change the command from ifconfig to ipconfig.

Linux & macOS

```
aws ssm send-command \
--instance-ids "instance-ID" \
--document-name "AWS-RunShellScript" \
--comment "IP config" \
--parameters commands=ifconfig \
--output text
```

Windows

```
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 run this command again to see the response data.

Linux & macOS

```
aws ssm list-command-invocations \
--command-id $sh-command-id \
--details
```

Windows

```
aws ssm list-command-invocations ^
```

```
--command-id $sh-command-id ^
--details
```

### Identify user account

The following command displays the default user account running the commands.

Linux & macOS

```
sh_command_id=$(aws ssm send-command \
 --instance-ids "instance-ID" \
 --document-name "AWS-RunShellScript" \
 --comment "Demo run shell script on Linux managed node" \
 --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 managed node. This example uses the Command ID that was returned in the previous command.

Linux & macOS

```
aws ssm list-commands \
 --command-id "command-ID"
```

Windows

```
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 managed node basis.

Linux & macOS

```
aws ssm list-command-invocations \
 --command-id "command-ID" \
 --details
```

Windows

```
aws ssm list-command-invocations ^
 --command-id "command-ID" ^
 --details
```

### Get command information with response data for a specific managed node

The following command returns the output of the original aws ssm send-command request for a specific managed node.

## Linux & macOS

```
aws ssm list-command-invocations \
--instance-id instance-ID \
--command-id "command-ID" \
--details
```

## Windows

```
aws ssm list-command-invocations ^
--instance-id instance-ID ^
--command-id "command-ID" ^
--details
```

## Display Python version

The following command returns the version of Python running on a node.

## Linux & macOS

```
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]"'
```

## Step 4: Run a simple Python script using Run Command

The following command runs a simple Python "Hello World" script using Run Command.

## Linux & macOS

```
sh_command_id=$(aws ssm send-command \
--instance-ids "instance-ID" \
--document-name "AWS-RunShellScript" \
--comment "Demo run shell script on Linux Instances" \
--parameters '{"commands":[{"#": "/usr/bin/python", "print \\"Hello World from python\\n"}]}' \
--output text \
--query "Command.CommandId") sh -c 'aws ssm list-command-invocations \
--command-id "$sh_command_id" \
--details \
--query "CommandInvocations[].CommandPlugins[].[Status:Status,Output:Output]"'
```

## Step 5: Run a Bash script using Run Command

The examples in this section demonstrate how to run the following bash script using Run Command.

For examples of using Run Command to run scripts stored in remote locations, see [Running scripts from Amazon S3 \(p. 1485\)](#) and [Running scripts from GitHub \(p. 1493\)](#).

```
#!/bin/bash
```

```
yum -y update
yum install -y ruby
cd /home/ec2-user
curl -O https://aws-codedeploy-us-east-2.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

This script installs the AWS CodeDeploy agent on Amazon Linux and Red Hat Enterprise Linux (RHEL) instances, as described in [Create an Amazon EC2 instance for CodeDeploy](#) in the *AWS CodeDeploy User Guide*.

The script installs the CodeDeploy agent from an AWS managed Amazon S3 bucket in the US East (Ohio) Region (us-east-2), aws-codedeploy-us-east-2.

#### Run a bash script in an AWS CLI command

The following sample demonstrates how to include the bash script in a CLI command using the `--parameters` option.

##### Linux & macOS

```
aws ssm send-command \
--document-name "AWS-RunShellScript" \
--targets '[{"Key": "InstanceIds", "Values": ["instance-id"]}]' \
--parameters '{"commands": ["#!/bin/bash", "yum -y update", "yum install -y
ruby", "cd /home/ec2-user", "curl -O https://aws-codedeploy-us-east-2.s3.amazonaws.com/
latest/install", "chmod +x ./install", "./install auto"]}'
```

#### Run a bash script in a JSON file

In the following example, the content of the bash script is stored in a JSON file, and the file is included in the command using the `--cli-input-json` option.

##### Linux & macOS

```
aws ssm send-command \
--document-name "AWS-RunShellScript" \
--targets "Key=InstanceIds,Values=instance-id" \
--cli-input-json file://installCodeDeployAgent.json
```

##### Windows

```
aws ssm send-command ^
--document-name "AWS-RunShellScript" ^
--targets "Key=InstanceIds,Values=instance-id" ^
--cli-input-json file://installCodeDeployAgent.json
```

The contents of the referenced `installCodeDeployAgent.json` file is shown in the following example.

```
{
 "Parameters": {
 "commands": [
 "#!/bin/bash",
 "yum -y update",
 "yum install -y ruby",
 "cd /home/ec2-user",
 "curl -O https://aws-codedeploy-us-east-2.s3.amazonaws.com/latest/install",
 "chmod +x ./install",
```

```
 " ./install auto"
 }
}
```

## Walkthrough: Use the AWS Tools for Windows PowerShell with Run Command

The following examples show how to use the AWS 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 AWS 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 permission on your managed nodes. If a user has permission to run any of the predefined Systems Manager documents (any document that begins with AWS), then that user also has administrator access to the node. For all other users, you should create restrictive documents and share them with specific users. For more information about restricting access to Run Command, a capability of AWS Systems Manager, see [Create non-Admin IAM users and groups for Systems Manager \(p. 18\)](#).

### Topics

- [Configure AWS Tools for Windows PowerShell session settings \(p. 1025\)](#)
- [List all available documents \(p. 1026\)](#)
- [Run PowerShell commands or scripts \(p. 1026\)](#)
- [Install an application using the AWS-InstallApplication document \(p. 1027\)](#)
- [Install a PowerShell module using the AWS-InstallPowerShellModule JSON document \(p. 1028\)](#)
- [Join a managed node to a Domain using the AWS-JoinDirectoryServiceDomain JSON document \(p. 1029\)](#)
- [Send Windows metrics to Amazon CloudWatch Logs using the AWS-ConfigureCloudWatch document \(p. 1030\)](#)
- [Update EC2Config using the AWS-UpdateEC2Config document \(p. 1031\)](#)
- [Turn on or turn off Windows automatic update using the AWS-ConfigureWindowsUpdate document \(p. 1032\)](#)
- [Manage Windows updates using Run Command \(p. 1033\)](#)

## Configure AWS Tools for Windows PowerShell session settings

### Specify your credentials

Open **Tools for Windows PowerShell** on your local computer and run the following command to specify your credentials. You must either have administrator permissions on the managed nodes you want to configure or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM). For more information, see [Systems Manager prerequisites \(p. 13\)](#).

```
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 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 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 a managed node as if you were logged on locally. You can issue commands or enter a path to a local script to run the command.

### Note

For information about rebooting managed nodes when using Run Command to call scripts, see [Rebooting managed nodes from scripts \(p. 1011\)](#).

### 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 managed nodes.

```
$runPSCmd = Send-SSMCommand `~
-InstanceIds @("instance-ID-1", "instance-ID-2") `~
-DocumentName "AWS-RunPowerShellScript" `~
-Comment "Demo AWS-RunPowerShellScript with two instances" `~
-Parameter @{'commands'=@('dir C:\Users', 'dir C:\')}
```

### Get command request details

The following command uses the `CommandId` to get the status of the command execution on both managed nodes. This example uses the `CommandId` that was returned in the previous command.

```
Get-SSMCommand `~
-CommandId $runPSCmd.CommandId
```

The status of the command in this example can be Success, Pending, or InProgress.

### Get command information per managed node

The following command uses the `CommandId` from the previous command to get the status of the command execution on a per managed node basis.

```
Get-SSMCommandInvocation `~
```

```
-CommandId $runPSCmd.CommandId
```

### Get command information with response data for a specific managed node

The following command returns the output of the original `Send-SSMCommand` for a specific managed node.

```
Get-SSMCommandInvocation `
-CommandId $runPSCmd.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 @("iinstance-ID-1","iinstance-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 managed nodes. The command requires the path or address to an MSI.

#### Note

For information about rebooting managed nodes when using Run Command to call scripts, see [Rebooting managed nodes from scripts \(p. 1011\)](#).

#### 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 managed node 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'='/norestart /quiet /log c:\pythoninstall.txt'}
```

### Get command information per managed node

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 managed node

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 managed nodes. 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".

```
$installPSCmd = 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 managed node

The following command uses the `CommandId` to get the status of the command execution.

```
Get-SSMCommandInvocation `
-CommandId $installPSCmd.CommandId `
-Details $true
```

## Get command information with response data for the managed node

The following command returns the output of the original Send-SSMCommand for the specific CommandId.

```
Get-SSMCommandInvocation `
 -CommandId $installPSCmd.CommandId `
 -Details $true | Select -ExpandProperty CommandPlugins
```

## Join a managed node to a Domain using the AWS-JoinDirectoryServiceDomain JSON document

Using Run Command, you can quickly join a managed node to an AWS Directory Service domain. Before executing this command, [create a directory](#). We also recommend that you learn more about the AWS Directory Service. For more information, see the [AWS Directory Service Administration Guide](#).

You can only join a managed node to a domain. You can't remove a node from a domain.

### Note

For information about managed nodes when using Run Command to call scripts, see [Rebooting managed nodes from scripts \(p. 1011\)](#).

## 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 a managed node to a domain

The following command joins a managed node to the given AWS Directory Service domain and uploads any generated output to the example Amazon Simple Storage Service (Amazon 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 managed node

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 managed node

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 Logs 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 allow 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 aren't included. If you turn off logging and then later turn logging back on, 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 CloudWatch Logs.

If you previously turned on CloudWatch integration in EC2Config, the Systems Manager settings override any settings stored locally on the managed node 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 a single managed node, see [Collecting metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch agent](#) in the *Amazon CloudWatch User Guide*.

### 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 managed node and moves Windows Applications logs to CloudWatch.

```
$cloudWatchCommand = Send-SSMCommand `
-InstanceId instance-ID `
-DocumentName "AWS-ConfigureCloudWatch" `
-Parameter @{'properties'='{"engineConfiguration":`
{"PollInterval":"00:00:15", "Components": [{"Id":"ApplicationEventLog", `
"FullName":"AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch", `
"Parameters": {"LogName":"Application", "Levels":"7"}}, {"Id":"CloudWatch", `
"FullName":"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch", `
"Parameters": {"Region":"region", "LogGroup":"my-log-group", "LogStream":"instance-id"}]}, `
"Flows": {"Flows": ["ApplicationEventLog", "CloudWatch"]}}}'`
```

### Get command information per managed node

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 managed node

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.
Parameters":{"CategoryName": "Memory", "CounterName": "Available
MBytes", "InstanceName": "", "MetricName": "AvailableMemory",
"Unit": "Megabytes", "DimensionName": "", "DimensionValue": ""}}, {"Id": "CloudWatch",
"FullName": "AWS.EC2.Windows.CloudWatch.CloudWatchOutputComponent,AWS.EC2.Windows.CloudWatch
Parameters":{"AccessKey": "", "SecretKey": "", "Region": "region", "NameSpace": "Windows-
Default"}]}, "Flows": {"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 managed nodes. 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

```
$ec2ConfigCommand = Send-SSMCommand `
-InstanceId instance-ID `
-DocumentName "AWS-UpdateEC2Config"
```

## Get command information with response data for the managed node

This command returns the output of the specified command from the previous Send-SSMCommand.

```
Get-SSMCommandInvocation `
-CommandId $ec2ConfigCommand.CommandId `
-Details $true `
-InstanceId instance-ID | Select -ExpandProperty CommandPlugins
```

## Update EC2Config to a specific version

The following command downgrades EC2Config to an older version.

```
Send-SSMCommand `
-InstanceId instance-ID `
-DocumentName "AWS-UpdateEC2Config" `
-Parameter @{'version'='4.9.3519'; 'allowDowngrade'='true'}
```

## Turn on or turn off Windows automatic update using the AWS-ConfigureWindowsUpdate document

Using Run Command and the AWS-ConfigureWindowsUpdate document, you can turn on or turn off automatic Windows updates on your Windows Server managed nodes. 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 managed node 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

```
Get-SSMDocumentDescription `
-Name "AWS-ConfigureWindowsUpdate"
```

### View more information about parameters

```
Get-SSMDocumentDescription `
-Name "AWS-ConfigureWindowsUpdate" | Select -ExpandProperty Parameters
```

## Turn on Windows automatic update

The following command configures Windows Update to automatically download and install updates daily at 10:00 PM.

```
$configureWindowsUpdateCommand = Send-SSMCommand `
-InstanceId instance-ID `
-DocumentName "AWS-ConfigureWindowsUpdate" `
-Parameters @{'updateLevel'='InstallUpdatesAutomatically';
'scheduledInstallDay'='Daily'; 'scheduledInstallTime'='22:00'}
```

### View command status for allowing Windows automatic update

The following command uses the `CommandId` to get the status of the command execution for allowing Windows automatic update.

```
Get-SSMCommandInvocation `
-Details $true `
-CommandId $configureWindowsUpdateCommand.CommandId | Select -ExpandProperty
CommandPlugins
```

## Turn off Windows automatic update

The following command lowers the Windows Update notification level so the system checks for updates but doesn't automatically update the managed node.

```
$configureWindowsUpdateCommand = Send-SSMCommand `
-InstanceId instance-ID `
-DocumentName "AWS-ConfigureWindowsUpdate" `
-Parameters @{'updateLevel'='NeverCheckForUpdates'}
```

## View command status for turning off Windows automatic update

The following command uses the `CommandId` to get the status of the command execution for turning off 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 Windows Server managed nodes. This command scans for or installs missing updates on your managed nodes 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 managed nodes when using Run Command to call scripts, see [Rebooting managed nodes from scripts \(p. 1011\)](#).

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" `~
 -Parameters @{'Action'='Install';'IncludeKbs'='kb-ID-1, kb-ID-2, kb-ID-3';'AllowReboot'='True'}
```

### 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, a capability of AWS Systems Manager, provides status details with each command execution. For more information about the details of command statuses, see [Understanding command statuses \(p. 1013\)](#). You can also use the information in this topic to help troubleshoot problems with Run Command.

## Topics

- [Some of my managed nodes are missing \(p. 1034\)](#)
- [A step in my script failed, but the overall status is 'succeeded' \(p. 1034\)](#)
- [SSM Agent isn't running properly \(p. 1034\)](#)

## Some of my managed nodes are missing

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 managed nodes you can choose to run the command on.

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

After you create, activate, reboot, or restart a managed node, install Run Command on a node, or attach an AWS Identity and Access Management (IAM) instance profile to a node, it can take a few minutes for the managed node to be added to the list.

## A step in my script failed, but the overall status is 'succeeded'

Using Run Command, you can define how your scripts handle exit codes. 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 \(p. 1012\)](#).

## SSM Agent isn't running properly

If you experience problems running commands using Run Command, there might be a problem with the SSM Agent. For information about investigating issues with SSM Agent, see [Troubleshooting SSM Agent \(p. 142\)](#).

# AWS Systems Manager State Manager

State Manager, a capability of AWS Systems Manager, is a secure and scalable configuration management service that automates the process of keeping your managed nodes and other AWS resources in a state that you define. To get started with State Manager, open the [Systems Manager console](#). In the navigation pane, choose **State Manager**.

### Note

State Manager and Maintenance Windows can perform some similar types of updates on your managed nodes. Which one you choose depends on whether you need to automate system compliance or perform high-priority, time-sensitive tasks during periods you specify.

For more information, see [Choosing between State Manager and Maintenance Windows \(p. 1554\)](#).

## How can State Manager benefit my organization?

By using pre-configured Systems Manager documents (SSM documents), State Manager offers the following benefits for managing your nodes:

- Bootstrap nodes with specific software at start-up.
- Download and update agents on a defined schedule, including the SSM Agent.
- Configure network settings.

- Join nodes to a Microsoft Active Directory domain.
- Patch nodes with software updates throughout their lifecycle.
- Run scripts on Linux, macOS, and Windows managed nodes throughout their lifecycle.

To manage configuration drift across other AWS resources, you can use Automation, a capability of Systems Manager, with State Manager to perform the following types of tasks:

- Attach a Systems Manager role to Amazon Elastic Compute Cloud (Amazon EC2) instances to make them *managed nodes*.
- Enforce desired ingress and egress rules for a security group.
- Create or delete Amazon DynamoDB backups.
- Create or delete Amazon Elastic Block Store (Amazon EBS) snapshots.
- Turn off read and write permissions on Amazon Simple Storage Service (Amazon S3) buckets.
- Start, restart, or stop managed nodes and Amazon Relational Database Service (Amazon RDS) instances.
- Apply patches to Linux, macOS, and Windows AMIs.

For information about using State Manager with Automation runbooks, see [Running automations with triggers using State Manager \(p. 441\)](#).

## Who should use State Manager?

State Manager is appropriate for any AWS customer that wants to improve the management and governance of their AWS resources and reduce configuration drift.

## What are the features of State Manager?

Key features of State Manager include the following:

- **State Manager associations**

A State Manager *association* is a configuration that you assign to your AWS resources. The configuration defines the state that you want to maintain on your resources. For example, an association can specify that antivirus software must be installed and running on a managed node, or that certain ports must be closed.

An association specifies a schedule for when to apply the configuration and the targets for the association. For example, an association for antivirus software might run once a day on all managed nodes in an AWS account. If the software isn't installed on a node, then the association could instruct State Manager to install it. If the software is installed, but the service isn't running, then the association could instruct State Manager to start the service.

- **Flexible scheduling options**

State Manager offers the following options for scheduling when an association runs:

- **Immediate or delayed processing**

When you create an association, by default, the system immediately runs it on the specified resources. After the initial run, the association runs in intervals according to the schedule that you defined.

You can instruct State Manager not to run an association immediately by using the **Apply association only at the next specified Cron interval** option in the console or the `ApplyOnlyAtCronInterval` parameter from the command line.

- **Cron and rate expressions**

When you create an association, you specify a schedule for when State Manager applies the configuration. State Manager supports standard cron and rate expressions for scheduling when an association runs. State Manager also supports cron expressions that include a day of the week and the number sign (#) to designate the *n*th day of a month to run an association and the (L) sign to indicate the last X day of the month.

To further control when an association runs, for example if you want to run an association two days after patch Tuesday, you can specify an offset. An *offset* defines how many days to wait after the scheduled day to run an association.

- **Multiple targeting options**

An association also specifies the targets for the association. State Manager supports targeting AWS resources by using tags, AWS Resource Groups, individual node IDs, or all managed nodes in the current AWS Region and AWS account.

- **EventBridge support**

This Systems Manager capability is supported as both an *event* type and a *target* type in Amazon EventBridge rules. For information, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#) and [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#).

- **AWS CloudTrail integration**

State Manager integrates with AWS CloudTrail to provide a record of all executions that you can audit, and Amazon EventBridge to track state changes. You can also choose to store and view detailed command output in Amazon Simple Storage Service (Amazon S3). For more information, see the following topics:

- [Logging AWS Systems Manager API calls with AWS CloudTrail \(p. 1442\)](#)
- [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#)
- [\(Optional\) Set up integrations with other AWS services \(p. 34\)](#)

## Is there a charge to use State Manager?

State Manager is available at no additional charge.

## How do I get started with State Manager?

Complete the following tasks to get started with State Manager.

Task	For More Information
Verify Systems Manager prerequisites	<a href="#">Systems Manager prerequisites (p. 13)</a>
Learn more about State Manager	<a href="#">About State Manager (p. 1037)</a>
Create and assign a State Manager association to your nodes	<a href="#">Creating associations (p. 1042)</a>

### Related content

- [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 \(p. 1037\)](#)
- [Working with associations in Systems Manager \(p. 1039\)](#)
- [AWS Systems Manager State Manager walkthroughs \(p. 1066\)](#)

## About State Manager

State Manager, a capability of AWS Systems Manager, is a secure and scalable service that automates the process of keeping your Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and hybrid infrastructure in a state that you define.

Here's how State Manager works:

**1. Determine the state you want to apply to your AWS resources.**

Do you want to guarantee that your managed nodes are configured with specific applications, such as antivirus 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 guarantee that specific ports are closed or open? To get started with State Manager, determine the state that you want to apply to your AWS resources. The state that you want to apply determines which SSM document you use to create a State Manager association.

A State Manager *association* is a configuration that you assign to your AWS resources. The configuration defines the state that you want to maintain on your resources. For example, an association can specify that antivirus software must be installed and running on a managed node, or that certain ports must be closed.

An association specifies a schedule for when to apply the configuration and the targets for the association. For example, an association for antivirus software might run once a day on all managed nodes in an AWS account. If the software isn't installed on a node, then the association could instruct State Manager to install it. If the software is installed, but the service isn't running, then the association could instruct State Manager to start the service.

**2. Determine if a preconfigured SSM document can help you create the desired state on your AWS resources.**

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 AWS Systems Manager automations, running PowerShell and Shell scripts, and joining managed nodes to a directory service domain for Active Directory.

You can view all SSM documents in the [Systems Manager console](#). Choose the name of a document to learn more about each one. Here are two examples: [AWS-ConfigureAWSPackage](#) and [AWS-InstallApplication](#).

**3. Create an association.**

You can create an association by using the Systems Manager console, the AWS Command Line Interface (AWS CLI), AWS Tools for Windows PowerShell (Tools for Windows PowerShell), or the Systems Manager API. When you create an association, you specify the following information:

- A name for the association.
- The parameters for the SSM document (for example, the path to the application to install or the script to run on the nodes).
- Targets for the association. You can target managed nodes by specifying tags, by choosing individual node IDs, or by choosing a group in AWS Resource Groups. You can also target *all* managed nodes in the current AWS Region and AWS account.

- 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 \(p. 1544\)](#).

When you run the command to create the association, Systems Manager binds the information you specified (schedule, targets, SSM document, and parameters) to the targeted resources. 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 times out and the new association runs.

Systems Manager reports the status of the request to create associations on the resources. You can view status details in the console or (for managed nodes) by using the [DescribeInstanceAssociationsStatus](#) API operation. If you choose to write the output of the command to Amazon Simple Storage Service (Amazon S3) when you create an association, you can also view the output in the Amazon S3 bucket you specified.

For more information, see [Creating associations \(p. 1042\)](#).

#### 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 \(p. 1059\)](#). You can update your association documents and reapply them as necessary. You can also create multiple versions of an association. For more information, see [Editing and creating a new version of an association \(p. 1052\)](#).

## About association scheduling

A State Manager association is a configuration that you assign to your AWS resources. The configuration defines the state that you want to maintain on your resources. When you create an association, you specify a schedule for when to apply the configuration and the targets for the association. By default, State Manager runs the association when you create it and then according to the specified schedule. State Manager also attempts to run the association in following situations:

- **Association edit:** State Manager runs the association after a user edits and saves their changes to any of the following association fields: DOCUMENT\_VERSION, PARAMETERS, SCHEDULE\_EXPRESSION, OUTPUT\_S3\_LOCATION.
- **Document edit:** State Manager runs the association after a user edits and saves changes to the underlying SSM document used to create the association. Specifically, the association runs after the following edits to the document:
  - A user specifies a new \$DEFAULT document version and the association was created using the \$DEFAULT version.
  - A user updates a document and the association was created using the \$LATEST version.
  - A user deletes a document that was used to create an association.
- **Target updates:** State Manager runs the association the first time a targeted instance comes online and the first time a targeted instance comes online after missing a scheduled association run.

**Note**

Target updates don't affect associations created using Systems Manager Automation.

- **Parameter Store parameter value change:** State Manager runs the association after a user edits the value of a parameter defined in the association.
- **Manual start:** State Manager runs the association anytime a user chooses to manually run the association from either the Systems Manager console or programmatically.

# Working with associations in Systems Manager

This section describes how to create and manage State Manager associations by using the AWS Systems Manager console, the AWS Command Line Interface (AWS CLI), and AWS Tools for PowerShell.

## Topics

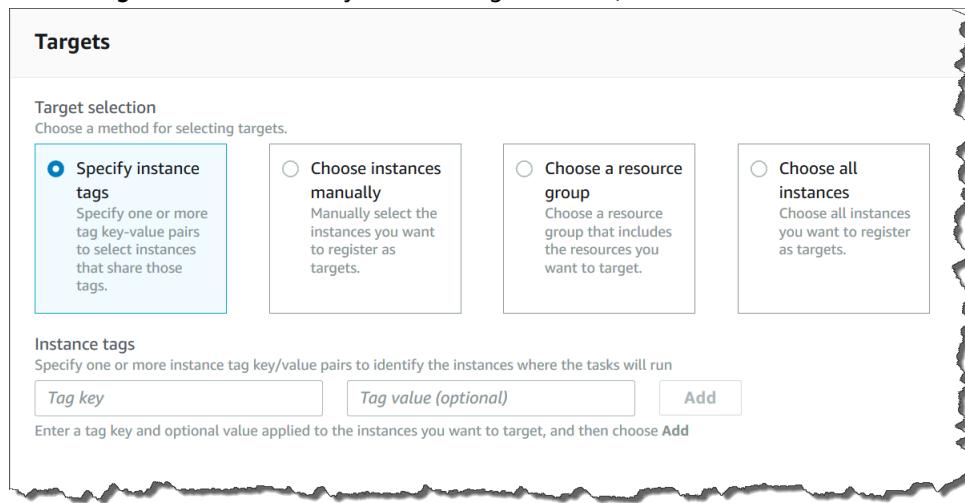
- [About targets and rate controls in State Manager associations \(p. 1039\)](#)
- [Creating associations \(p. 1042\)](#)
- [Editing and creating a new version of an association \(p. 1052\)](#)
- [Deleting an association \(p. 1058\)](#)
- [Running Auto Scaling groups with associations \(p. 1059\)](#)
- [Viewing association histories \(p. 1059\)](#)
- [Working with associations using IAM \(p. 1065\)](#)

## About targets and rate controls in State Manager associations

This topic describes State Manager, a capability of AWS Systems Manager, features that help you deploy an association to dozens or hundreds of nodes while controlling the number of nodes that run the association at the scheduled time.

### Targets

When you create a State Manager association, you choose which nodes 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 Command Line Interface (AWS CLI), then you specify the `targets` parameter. Targeting nodes allows you to configure tens, hundreds, or thousands of nodes with an association without having to specify or choose individual node IDs.

State Manager includes the following target options when creating an association.

#### Specify tags

Use this option to specify a tag key and (optionally) a tag value assigned to your nodes. When you run the request, the system locates and attempts to create the association on all nodes that match the specified tag key and value. If you specified multiple tag values, the association targets any node with at

least one of those tag values. 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 nodes and assign the specified tag key and value to those nodes, the system automatically applies the association, runs it immediately, and then runs it according to the schedule. This applies when the association uses a Command or Policy document and doesn't apply if the association uses an Automation runbook. If you delete the specified tags from a node, the system no longer runs the association on those nodes.

It's a best practice to use tags when creating associations that use a Command or Policy document; this doesn't apply if the association uses an Automation runbook. If you delete the specified tags from a node, the system no longer runs the association on those nodes.

It's also a best practice to use tags when creating associations to run Auto Scaling groups. For more information, see [Running Auto Scaling groups with associations \(p. 1059\)](#).

**Note**

When using tags, you can use one tag key maximum in the console. If you want to target your nodes using more than one tag key, use the resource group option. You can also specify multiple tag keys using the AWS CLI. In this case, all tag keys are required for the nodes to be targeted.

For information about assigning tags to your nodes, see [Tagging Systems Manager resources \(p. 1505\)](#).

### Choose nodes manually

Use this option to manually select the nodes where you want to create the association. The **Instances** pane displays all Systems Manager managed nodes in the current AWS account and AWS Region. You can manually select as many nodes as you want. 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.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

### Choose a resource group

Use this option to create an association on all nodes returned by an AWS Resource Groups tag-based or AWS CloudFormation stack-based query.

Below are details about targeting resource groups for an association.

- If you add new nodes to a group, the system automatically maps the nodes to the association that targets the resource group. The system applies the association to the nodes when it discovers the change. After this initial run, the system runs the association according to the schedule you specified.
- If you create an association that targets a resource group and the `AWS::SSM::ManagedInstance` resource type was specified for that group, then the association runs on hybrid machines *and* Amazon Elastic Compute Cloud (Amazon EC2) instances (machines with a managed instance ID prefix of either "mi-" or "i-"). This is by design.
- If you create an association that targets a resource group, the resource group must not have more than five tag keys assigned to it or more than five values specified for any one tag key. If either of these conditions applies to the tags and keys assigned to your resource group, the association fails to run and returns an `InvalidTarget` error.
- If you delete a resource group, all instances in that group no longer run the association. As a best practice, delete associations targeting the group.
- At most you can target a single resource group for an association. Multiple or nested groups aren't 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 determine 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.

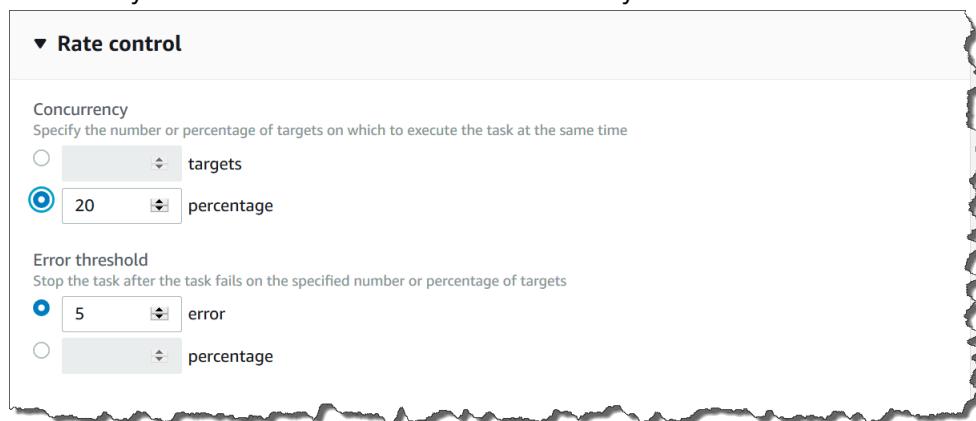
For more information about Resource Groups, see [What Is AWS Resource Groups?](#) in the *AWS Resource Groups User Guide*.

### Choose all nodes

Use this option to target all nodes in the current AWS account and AWS Region. When you run the request, the system locates and attempts to create the association on all nodes in the current AWS account and AWS 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 nodes, 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 nodes by specifying a concurrency value and an error threshold. The concurrency value specifies how many nodes can run the association simultaneously. An error threshold specifies how many association executions can fail before Systems Manager sends a command to each node 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 nodes by allowing you to specify that only a certain number of nodes can process an association at one time. You can specify either an absolute number of nodes, for example 20, or a percentage of the target set of nodes, 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 nodes.
- If new nodes that match the target criteria come online while an association that uses concurrency is running, then the new nodes run the association if the concurrency value isn't exceeded. If the concurrency value is exceeded, then the nodes are ignored during the current association execution interval. The nodes 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 nodes are processing that association when it's updated, then any node that is running the association is allowed to complete.

Those associations that haven't started are stopped. After running associations complete, all target nodes 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 node 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.

## Creating associations

State Manager, a capability of AWS Systems Manager, helps you keep your AWS resources in a state that you define and reduce configuration drift. To do this, State Manager uses associations. An *association* is a configuration that you assign to your AWS resources. The configuration defines the state that you want to maintain on your resources. For example, an association can specify that antivirus software must be installed and running on a managed node, or that certain ports must be closed.

An association specifies a schedule for when to apply the configuration and the targets for the association. For example, an association for antivirus software might run once a day on all managed nodes in an AWS account. If the software isn't installed on a node, then the association could instruct State Manager to install it. If the software is installed, but the service isn't running, then the association could instruct State Manager to start the service.

The following procedures describe how to create an association that uses either a **Command** or a **Policy** document to target managed nodes. For information about creating an association that uses an Automation runbook to target nodes or other types of AWS resources, see [Running automations with triggers using State Manager \(p. 441\)](#).

### Before you begin

When you create an association, you can either specify a schedule for the association or run the association immediately. If you specify a schedule, you must enter the schedule in the form of either a cron or rate expression. For more information about cron and rate expressions, see [Reference: Cron and rate expressions for Systems Manager \(p. 1540\)](#).

An association also specifies which managed nodes, or targets, should receive the association. State Manager includes several features to help you target your managed nodes and control how the association is deployed to those targets. For more information about targets and rate controls, see [About targets and rate controls in State Manager associations \(p. 1039\)](#).

## About creating associations

When you create an association, by default, the system immediately runs it on the specified resources. After the initial run, the association runs in intervals according to the schedule that you defined and according to the following rules:

- State Manager attempts to run the association on all specified or targeted nodes during an interval.
- If an association doesn't run during an interval (because, for example, a concurrency value limited the number of nodes 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.

For associations that run according to a schedule, you can specify standard cron or rate expressions that define when the association runs. State Manager also supports cron expressions that include a day of the week and the number sign (#) to designate the *n*th day of a month to run an association. Here is an example that runs a cron schedule on the third Tuesday of every month at 23:30 UTC:

```
cron(30 23 ? * TUE#3 *)
```

Here is an example that runs on the second Thursday of every month at midnight UTC:

```
cron(0 0 ? * THU#2 *)
```

State Manager also supports the (L) sign to indicate the last *X* day of the month. Here is an example that runs a cron schedule on the last Tuesday of every month at midnight UTC:

```
cron(0 0 ? * 3L *)
```

To further control when an association runs, for example if you want to run an association two days after patch Tuesday, you can specify an offset. An *offset* defines how many days to wait after the scheduled day to run an association. For example, if you specified a cron schedule of `cron(0 0 ? * THU#2 *)`, you could specify the number 3 in the **Schedule offset** field to run the association each Sunday after the second Thursday of the month.

### Note

To use offsets, you must either choose the **Apply association only at the next specified Cron interval** option in the console or you must specify the `ApplyOnlyAtCronInterval` parameter from the command line. This option tells State Manager not to run an association immediately after you create it.

## Create an association (console)

The following procedure describes how to use the Systems Manager console to create a State Manager association.

### Warning

When you create an association, you can choose an AWS resource group of managed nodes as the target for the association. If an AWS Identity and Access Management (IAM) user, group, or role has permission to create an association that targets a resource group of managed nodes, then that user, group, or role automatically has root-level control of all nodes in the group. Permit only trusted administrators to create associations.

### To create a State Manager association

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Name** field, specify a name.
5. In the **Document** list, choose the option next to a document name. Note the document type. This procedure applies to **Command** and **Policy** documents. For information about creating an association that uses an Automation runbook, see [Running automations with triggers using State Manager \(p. 441\)](#).

**Important**

State Manager doesn't support running associations that use a new version of a document if that document is shared from another account. State Manager always runs the **default** version of a document if shared from another account, even though the Systems Manager console shows that a new version was processed. If you want to run an association using a new version of a document shared from another account, you must set the document version to **default**.

6. For **Parameters**, specify the required input parameters.
7. For **Targets**, choose an option. For information about using targets, see [About targets and rate controls in State Manager associations \(p. 1039\)](#).
8. 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**.

9. (Optional) In the **Schedule offset** field, specify a number between 1 and 6.
10. In the **Advanced options** section use **Compliance severity** to choose a severity level for the association and use **Change Calendars** to choose a change calendar 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 \(p. 840\)](#).

The change calendar determines when the association runs. If the calendar is closed, the association isn't applied. If the calendar is open, the association runs accordingly. For more information, see [AWS Systems Manager Change Calendar \(p. 695\)](#).

11. In the **Rate control** section, choose options to control how the association runs on multiple nodes. For more information about using rate controls, see [About targets and rate controls in State Manager associations \(p. 1039\)](#).

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.

12. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Enter 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 managed node, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

Following are the minimal permissions required to turn on Amazon S3 output for an association. You might further restrict access to individual IAM users or roles within an account. At minimum, an Amazon EC2 instance profile should have an IAM role with the `AmazonSSMManagedInstanceCore` managed policy and the following inline policy.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "s3:PutObject",
 "s3:GetObject",
 "s3:PutObjectAcl"
],
 "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
 }
]
}
```

For minimal permissions, the Amazon S3 bucket you export to must have the default settings defined by the Amazon S3 console. For more information about creating Amazon S3 buckets, see [Creating a bucket](#) in the *Amazon S3 User Guide*.

13. 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 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 allow you to assign an association to dozens or hundreds of nodes 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 \(p. 1039\)](#).

### Before you begin

The `targets` parameter is an array of search criteria that targets nodes using a `Key,Value` combination that you specify. If you plan to create an association on dozens or hundreds of node by using the `targets` parameter, review the following targeting options before you begin the procedure.

#### Target specific nodes by specifying IDs

```
--targets Key=InstanceIds,Values=instance-id-1,instance-id-2,instance-id-3
```

```
--targets
Key=InstanceIds,Values=i-02573cafefEXAMPLE,i-0471e04240EXAMPLE,i-07782c72faEXAMPLE
```

Target instances by using Amazon EC2 tags

```
--targets Key=tag:tag-key,Values=tag-value-1,tag-value-2,tag-value-3
```

```
--targets Key=tag:Environment,Values=Development,Test,Pre-production
```

**Note**

When using tags, you can only use one tag key. If you want to target your nodes using more than one tag key, use the resource group option.

Target nodes by using AWS Resource Groups

```
--targets Key=resource-groups:Name,Values=resource-group-name
```

```
--targets Key=resource-groups:Name,Values=WindowsInstancesGroup
```

Target all instances in the current AWS account and AWS Region

```
--targets Key=InstanceIds,Values=*
```

**Note**

Note the following information.

- State Manager doesn't support running associations that use a new version of a document if that document is shared from another account. State Manager always runs the `default` version of a document if shared from another account, even though the Systems Manager console shows that a new version was processed. If you want to run an association using a new version of a document shared from another account, you must set the document version to `default`.
- When you create an association, you specify when the schedule runs. 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 \(p. 1544\)](#).

## To create an association

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Use the following format to create a command that creates a State Manager association.

Linux & macOS

```
aws ssm create-association \
--name document_name \
--document-version version_of_document_applied \
--instance-id instances_to_apply_association_on \
--parameters (if any) \
--targets target_options \
--schedule "cron_or_rate_expression" \
--apply-only-at-cron-interval required_parameter_for_schedule_offsets \
--schedule-offset number_between_1_and_6 \
--output-location s3_bucket_to_store_output_details \
--association-name association_name \
--max-errors a_number_of_errors_or_a_percentage_of_target_set \
--max-concurrency a_number_of_instances_or_a_percentage_of_target_set \
--compliance-severity severity_level \
```

```
--calendar-names change_calendar_names \
--target-locations aws_region_or_account
```

### Windows

```
aws ssm create-association ^
--name document_name ^
--document-version version_of_document_applied ^
--instance-id instances_to_apply_association_on ^
--parameters (if any) ^
--targets target_options ^
--schedule "cron_or_rate_expression" ^
--apply-only-at-cron-interval required_parameter_for_schedule_offsets ^
--schedule-offset number_between_1_and_6 ^
--output-location s3_bucket_to_store_output_details ^
--association-name association_name ^
--max-errors a_number_of_errors_or_a_percentage_of_target_set ^
--max-concurrency a_number_of_instances_or_a_percentage_of_target_set ^
--compliance-severity severity_level ^
--calendar-names change_calendar_names ^
--target-locations aws_region_or_account
```

### PowerShell

```
New-SSMAssociation `-
-Name document_name `-
-DocumentVersion version_of_document_applied `-
-InstanceId instances_to_apply_association_on `-
-Parameters (if any) `-
-Target target_options `-
-ScheduleExpression "cron_or_rate_expression" `-
-ApplyOnlyAtCronInterval required_parameter_for_schedule_offsets `-
-ScheduleOffSet number_between_1_and_6 `-
-OutputLocation s3_bucket_to_store_output_details `-
-AssociationName association_name `-
-MaxError a_number_of_errors_or_a_percentage_of_target_set `-
-MaxConcurrency a_number_of_instances_or_a_percentage_of_target_set `-
-ComplianceSeverity severity_level `-
-CalendarNames change_calendar_names `-
-TargetLocations aws_region_or_account
```

The following example creates an association on nodes tagged with "Environment, Linux". The association uses the AWS-UpdateSSMAgent document to update the SSM Agent on the targeted nodes at 2:00 UTC every Sunday morning. This association runs simultaneously on 10 nodes maximum at any given time. Also, this association stops running on more nodes for a particular execution interval if the error count exceeds 5. For compliance reporting, this association is assigned a severity level of Medium.

### Linux & macOS

```
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

```
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"
```

## PowerShell

```
New-SSMAssociation `
-AssociationName Update_SSM_Agent_Linux `
-Name AWS-UpdateSSMAgent `
-Target @{
 "Key"="tag:Environment"
 "Values"="Linux"
} `
-ComplianceSeverity MEDIUM `
-ScheduleExpression "cron(0 2 ? * SUN *)" `
-MaxConcurrency 10 `
-MaxError 5
```

The following example creates an association that scans nodes for missing patch updates by using the AWS-RunPatchBaseline document. This association targets all managed nodes in the account in the us-east-2 Region. The association specifies the Operation and RebootOption parameters.

## Linux & macOS

```
aws ssm create-association \
--name "AWS-RunPatchBaseline" \
--association-name "ScanningInstancesForMissingUpdate" \
--targets "Key=instanceids,Values=*" \
--parameters "Operation=Scan,RebootOption=NoReboot" \
--region us-east-2
```

## Windows

```
aws ssm create-association ^
--name "AWS-RunPatchBaseline" ^
--association-name "ScanningInstancesForMissingUpdate" ^
--targets "Key=instanceids,Values=*" ^
--parameters "Operation=Scan,RebootOption=NoReboot" ^
--region us-east-2
```

## PowerShell

```
New-SSMAssociation `
-AssociationName ScanningInstancesForMissingUpdate `
-Name AWS-RunPatchBaseline `
-Target @{
 "Key"="instanceids"
 "Values"="*"
} `
-Parameters "Operation=Scan,RebootOption=NoReboot" `
```

```
-Region us-east-2
```

The following example targets node IDs by specifying a wildcard value (\*). This allows Systems Manager to create an association on *all* nodes in the current AWS account and AWS Region. This association runs simultaneously on 10 nodes maximum at any given time. Also, this association stops running on more nodes 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 uses a schedule offset, which means it runs two days after the specified cron schedule. It also includes the `ApplyOnlyAtCronInterval` parameter, which is required to use the schedule offset and which means the association won't run immediately after it is created.

#### Linux & macOS

```
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#2 *)" \
--apply-only-at-cron-interval \
--schedule-offset 2 \
--max-errors "5" \
--max-concurrency "10" \
```

#### 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#2 *)" ^
--apply-only-at-cron-interval ^
--schedule-offset 2 ^
--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"="*"
} `
-ScheduleExpression "cron(0 2 ? * SUN#2 *)" `
-ApplyOnlyAtCronInterval `
-ScheduleOffset 2 `
-MaxConcurrency 10 `
-MaxError 5 `
-ComplianceSeverity MEDIUM `
-ApplyOnlyAtCronInterval
```

The following example creates an association on nodes in Resource Groups. The group is named "HR-Department". The association uses the AWS-UpdateSSMAgent document to update SSM Agent

on the targeted nodes at 2:00 UTC every Sunday morning. This association runs simultaneously on 10 nodes maximum at any given time. Also, this association stops running on more nodes 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 at the specified cron schedule. It doesn't run immediately after the association is created.

#### Linux & macOS

```
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

```
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

```
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
```

The following example creates an association that runs on nodes tagged with a specific node id. The association uses the SSM Agent document to update SSM Agent on the targeted nodes once when the change calendar is open. The association checks the calendar state when it runs. If the calendar is closed at launch time and the association is only run once, it won't run again because the association run window has passed. If the calendar is open, the association runs accordingly.

#### Note

If you add new nodes to the tags or resource groups that an association acts on when the change calendar is closed, the association is applied to those nodes once the change calendar opens.

#### Linux & macOS

```
aws ssm create-association \
```

```
--association-name CalendarAssociation \
--targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f" \
--name AWS-UpdateSSMAgent \
--calendar-names "arn:aws:ssm:us-east-1:123456789012:document/testCalendar1" \
--schedule "rate(1day)"
```

## Windows

```
aws ssm create-association ^
--association-name CalendarAssociation ^
--targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f" ^
--name AWS-UpdateSSMAgent ^
--calendar-names "arn:aws:ssm:us-east-1:123456789012:document/testCalendar1" ^
--schedule "rate(1day)"
```

## PowerShell

```
New-SSMAssociation `
-AssociationName CalendarAssociation `
-Target @{
 "Key"="tag:instanceids"
 "Values"="i-0cb2b964d3e14fd9f"
} `
-Name AWS-UpdateSSMAgent `
-CalendarNames "arn:aws:ssm:us-east-1:123456789012:document/testCalendar1" `
-ScheduleExpression "rate(1day)"
```

The following example creates an association that runs on nodes tagged with a specific node id. The association uses the SSM Agent document to update SSM Agent on the targeted nodes on the targeted nodes at 2:00 AM every Sunday. This association runs only at the specified cron schedule when the change calendar is open. When the association is created, it checks the calendar state. If the calendar is closed, the association isn't applied. When the interval to apply the association starts at 2:00 AM on Sunday, the association checks to see if the calendar is open. If the calendar is open, the association runs accordingly.

### Note

If you add new nodes to the tags or resource groups that an association acts on when the change calendar is closed, the association is applied to those nodes once the change calendar opens.

## Linux & macOS

```
aws ssm create-association \
--association-name MultiCalendarAssociation \
--targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f" \
--name AWS-UpdateSSMAgent \
--calendar-names "arn:aws:ssm:us-east-1:123456789012:document/testCalendar1" \
"arn:aws:ssm:us-east-2:123456789012:document/testCalendar2" \
--schedule "cron(0 2 ? * SUN *)"
```

## Windows

```
aws ssm create-association ^
--association-name MultiCalendarAssociation ^
--targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f" ^
--name AWS-UpdateSSMAgent ^
--calendar-names "arn:aws:ssm:us-east-1:123456789012:document/testCalendar1" ^
"arn:aws:ssm:us-east-2:123456789012:document/testCalendar2" ^
```

```
--schedule "cron(0 2 ? * SUN *)"
```

#### PowerShell

```
New-SMMAssociation `‐
 -AssociationName MultiCalendarAssociation `‐
 -Name AWS-UpdateSSMAgent `‐
 -Target @{
 "Key"="tag:instanceids"
 "Values"="i-0cb2b964d3e14fd9f"
 } `‐
 -CalendarNames "arn:aws:ssm:us-east-1:123456789012:document/testCalendar1"
 "arn:aws:ssm:us-east-2:123456789012:document/testCalendar2" `‐
 -ScheduleExpression "cron(0 2 ? * SUN *)"
```

#### 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.

## Editing and creating a new version of an association

You can edit a State Manager 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 Simple Storage Service (Amazon S3) bucket. After you edit an association, State 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 Systems Manager console, AWS Command Line Interface (AWS CLI), and AWS Tools for PowerShell (Tools for PowerShell).

#### Important

State Manager doesn't support running associations that use a new version of a document if that document is shared from another account. State Manager always runs the default version of a document if shared from another account, even though the Systems Manager console shows that a new version was processed. If you want to run an association using a new version of a document shared from another account, you must set the document version to default.

### 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 Amazon S3 bucket. If you haven't 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 [Create an association \(command line\) \(p. 1045\)](#) and then choose **Edit**.
4. In the **Name** field, enter a new name.
5. In the **Specify schedule** section, choose a new option.
6. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Enter 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 managed node, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

7. Choose **Edit association**. Configure the association to meet your current requirements.
8. In the **Associations** page, choose the name of the association you edited, and then choose the **Versions** tab. The system lists each version of the association you created and edited.
9. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
10. Choose the name of the Amazon S3 bucket you specified for storing command output, and then choose the folder named with the ID of the node 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 `awsrunPowerShell1` 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 haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Use the following format to create a command to edit and create a new version of an existing State Manager association.

**Important**

When you call `UpdateAssociation`, the system drops all optional parameters from the request and overwrites the association with null values for those parameters. This is by design. You must specify all optional parameters in the call, even if you are not changing the parameters. This includes the `Name` parameter. Before calling this API action, we recommend that you call the [DescribeAssociation](#) API operation and make a note of all optional parameters required for your `UpdateAssociation` call.

Linux & macOS

```
aws ssm update-association \
--name document_name \
--document-version version_of_document_applied \
--instance-id instances_to_apply_association_on \
--parameters (if any) \
--targets target_options \
--schedule "cron_or_rate_expression" \
--schedule-offset "number_between_1_and_6" \
--output-location s3_bucket_to_store_output_details \
```

```
--association-name association_name \
--max-errors a_number_of_errors_or_a_percentage_of_target_set \
--max-concurrency a_number_of_instances_or_a_percentage_of_target_set \
--compliance-severity severity_level \
--calendar-names change_calendar_names \
--target-locations aws_region_or_account
```

### Windows

```
aws ssm update-association ^
--name document_name ^
--document-version version_of_document_applied ^
--instance-id instances_to_apply_association_on ^
--parameters (if any) ^
--targets target_options ^
--schedule "cron_or_rate_expression" ^
--schedule-offset "number_between_1_and_6" ^
--output-location s3_bucket_to_store_output_details ^
--association-name association_name ^
--max-errors a_number_of_errors_or_a_percentage_of_target_set ^
--max-concurrency a_number_of_instances_or_a_percentage_of_target_set ^
--compliance-severity severity_level ^
--calendar-names change_calendar_names ^
--target-locations aws_region_or_account
```

### PowerShell

```
Update-SSMAssociation `
-Name document_name `
-DocumentVersion version_of_document_applied `
-InstanceId instances_to_apply_association_on `
-Parameters (if any) `
-Target target_options `
-ScheduleExpression "cron_or_rate_expression" `
-ScheduleOffset "number_between_1_and_6" `
-OutputLocation s3_bucket_to_store_output_details `
-AssociationName association_name `
-MaxError a_number_of_errors_or_a_percentage_of_target_set `
-MaxConcurrency a_number_of_instances_or_a_percentage_of_target_set `
-ComplianceSeverity severity_level `
-CalendarNames change_calendar_names `
-TargetLocations aws_region_or_account
```

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 & macOS

```
aws ssm update-association \
--association-id 8dfe3659-4309-493a-8755-01234EXAMPLE \
--association-name TestHostnameAssociation2 \
--parameters commands="echo Association" \
--output-location S3Location='{OutputS3Region=us-east-1,OutputS3BucketName=DOC-EXAMPLE-BUCKET,OutputS3KeyPrefix=logs}' \
--schedule-expression "cron(0 */1 * * ? *)"
```

## Windows

```
aws ssm update-association ^
--association-id 8dfe3659-4309-493a-8755-01234EXAMPLE ^
--association-name TestHostnameAssociation2 ^
--parameters commands="echo Association" ^
--output-location S3Location='{OutputS3Region=us-east-1,OutputS3BucketName=DOC-
EXAMPLE-BUCKET,OutputS3KeyPrefix=logs}' ^
--schedule-expression "cron(0 */1 * * ? *)"
```

## PowerShell

```
Update-SSMAssociation `
-AssociationId b85ccafe-9f02-4812-9b81-01234EXAMPLE `
-AssociationName TestHostnameAssociation2 `
-Parameter @{"commands"="echo Association"} `
-S3Location_OutputS3BucketName DOC-EXAMPLE-BUCKET `
-S3Location_OutputS3KeyPrefix logs `
-S3Location_OutputS3Region us-east-1 `
-ScheduleExpression "cron(0 */1 * * ? *)"
```

The following example updates an existing association to change the name to `CalendarAssociation`. The new association runs when the calendar is open and writes command output to the specified Amazon S3 bucket.

## Linux & macOS

```
aws ssm update-association \
--association-id 8dfe3659-4309-493a-8755-01234EXAMPLE \
--association-name CalendarAssociation \
--parameters commands="echo Association" \
--output-location S3Location='{OutputS3Region=us-east-1,OutputS3BucketName=DOC-
EXAMPLE-BUCKET,OutputS3KeyPrefix=logs}' \
--calendar-names "arn:aws:ssm:us-east-1:123456789012:document/testCalendar2"
```

## Windows

```
aws ssm update-association ^
--association-id 8dfe3659-4309-493a-8755-01234EXAMPLE ^
--association-name CalendarAssociation ^
--parameters commands="echo Association" ^
--output-location S3Location='{OutputS3Region=us-east-1,OutputS3BucketName=DOC-
EXAMPLE-BUCKET,OutputS3KeyPrefix=logs}' ^
--calendar-names "arn:aws:ssm:us-east-1:123456789012:document/testCalendar2"
```

## PowerShell

```
Update-SSMAssociation `
-AssociationId b85ccafe-9f02-4812-9b81-01234EXAMPLE `
-AssociationName CalendarAssociation `
-AssociationName OneTimeAssociation `
-Parameter @{"commands"="echo Association"} `
-S3Location_OutputS3BucketName DOC-EXAMPLE-BUCKET `
-CalendarNames "arn:aws:ssm:us-east-1:123456789012:document/testCalendar2"
```

The following example updates an existing association to change the name to MultiCalendarAssociation. The new association runs when the calendars are open and writes command output to the specified Amazon S3 bucket.

Linux & macOS

```
aws ssm update-association \
--association-id 8dfe3659-4309-493a-8755-01234EXAMPLE \
--association-name MultiCalendarAssociation \
--parameters commands="echo Association" \
--output-location S3Location='{OutputS3Region=us-east-1,OutputS3BucketName=DOC-EXAMPLE-BUCKET,OutputS3KeyPrefix=logs}' \
--calendar-names "arn:aws:ssm:us-east-1:123456789012:document/testCalendar1" \
"arn:aws:ssm:us-east-2:123456789012:document/testCalendar2"
```

Windows

```
aws ssm update-association ^
--association-id 8dfe3659-4309-493a-8755-01234EXAMPLE ^
--association-name MultiCalendarAssociation ^
--parameters commands="echo Association" ^
--output-location S3Location='{OutputS3Region=us-east-1,OutputS3BucketName=DOC-EXAMPLE-BUCKET,OutputS3KeyPrefix=logs}' ^
--calendar-names "arn:aws:ssm:us-east-1:123456789012:document/testCalendar1" ^
"arn:aws:ssm:us-east-2:123456789012:document/testCalendar2"
```

PowerShell

```
Update-SMMAssociation `
-AssociationId b85ccafe-9f02-4812-9b81-01234EXAMPLE `
-AssociationName MultiCalendarAssociation `
-Parameter @{"commands"="echo Association"} `
-S3Location_OutputS3BucketName DOC-EXAMPLE-BUCKET `
-CalendarNames "arn:aws:ssm:us-east-1:123456789012:document/testCalendar1" `
"arn:aws:ssm:us-east-2:123456789012:document/testCalendar2"
```

3. To view the new version of the association, run the following command.

Linux & macOS

```
aws ssm describe-association \
--association-id b85ccafe-9f02-4812-9b81-01234EXAMPLE
```

Windows

```
aws ssm describe-association ^
--association-id b85ccafe-9f02-4812-9b81-01234EXAMPLE
```

PowerShell

```
Get-SMMAssociation `
-AssociationId b85ccafe-9f02-4812-9b81-01234EXAMPLE | Select-Object *
```

The system returns information like the following.

Linux & macOS

```
{
 "AssociationDescription": {
 "ScheduleExpression": "cron(0 */1 * * ? *)",
 "OutputLocation": {
 "S3Location": {
 "OutputS3KeyPrefix": "logs",
 "OutputS3BucketName": "DOC-EXAMPLE-BUCKET",
 "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

```
{
 "AssociationDescription": {
 "ScheduleExpression": "cron(0 */1 * * ? *)",
 "OutputLocation": {
 "S3Location": {
 "OutputS3KeyPrefix": "logs",
 "OutputS3BucketName": "DOC-EXAMPLE-BUCKET",
 "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"
 }
]
 }
}
```

```
 "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"
 }
]
}
```

### PowerShell

```
AssociationId : b85ccafe-9f02-4812-9b81-01234EXAMPLE
AssociationName : TestHostnameAssociation2
AssociationVersion : 2
AutomationTargetParameterName :
ComplianceSeverity :
Date : 5/31/2019 2:47:18 PM
DocumentVersion : $DEFAULT
InstanceId :
LastExecutionDate : 5/31/2019 3:26:40 PM
LastSuccessfulExecutionDate : 5/31/2019 3:26:40 PM
LastUpdateAssociationDate : 5/31/2019 3:26:29 PM
MaxConcurrency :
MaxErrors :
Name : AWS-RunPowerShellScript
OutputLocation :
Amazon.SimpleSystemsManagement.Model.InstanceAssociationOutputLocation
Overview :
Amazon.SimpleSystemsManagement.Model.AssociationOverview
Parameters : {[commands,
Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
ScheduleExpression : cron(0 */1 * * ? *)
Status :
Targets : {tag:Environment}
```

## Deleting an association

The following procedure describes how to delete a State Manager association by using the AWS Management Console.

### To delete an association

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 an association and then choose **Delete**.

## Running Auto Scaling groups with associations

The best practice when using associations to run Auto Scaling groups is to use tag targets. Not using tags might cause you to reach the association limit.

If all nodes are tagged with the same key and value, you only need one association to run your Auto Scaling group. The following procedure describes how to create such an association.

### To create an association that runs Auto Scaling groups

1. Ensure all nodes in the Auto Scaling group are tagged with the same key and value. For more instructions on tagging nodes, see [Tagging Auto Scaling groups and instances in the AWS Auto Scaling User Guide](#).
2. Create an association by using the procedure in [Creating associations \(p. 1042\)](#).

If you're working in the console, choose **Specify instance tags** in the **Targets** field. For **Instance tags**, enter the **Tag** key and value for your Auto Scaling group.

If you're using the AWS Command Line Interface (AWS CLI), specify `--targets Key=tag:tag-key,Values=tag-value` where the key and value match what you tagged your nodes with.

## Viewing association histories

You can view all executions for a specific association ID by using the [DescribeAssociationExecutions](#) API operation. Use this operation to see the status, detailed status, results, last execution time, and more information for a State Manager association. State Manager is a capability of AWS Systems Manager. This API operation also includes filters to help you 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 executions 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 operation. This operation shows you the resources, such as node IDs, where the association ran and the various association statuses. You can then see which resource or node failed to run an association. With the resource ID you can then view the command execution details to see which step in a command failed.

The examples in this section also include information about how to use the [StartAssociationsOnce](#) API operation to run an association once at the time of creation. You can use this API operation 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)

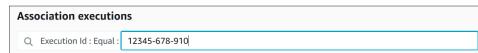
Use the following procedure to view the execution history for a specific association ID and then view execution details for one or more resources.

#### To view execution history for a specific association ID

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 nodes that failed to run the association.

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 the resources that ran the association.
8. Choose a resource ID to view specific information about that resource.

Use the search box filters to locate the resource for which you want to view details.



9. If you're investigating an association that failed to run, you can use the **Apply association now** button to run an association once at the time of creation. 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 Command Line Interface (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 haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
2. Run the following command to view a list of executions for a specific association ID.

Linux & macOS

```
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

```
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

```
Get-SMMAssociationExecution `~
 -AssociationId ID `~
 -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

---

department

#### Linux & macOS

```
{
 "AssociationExecutions": [
 {
 "Status": "Success",
 "DetailedStatus": "Success",
 "AssociationId": "c336d2ab-09de-44ba-8f6a-6136cEXAMPLE",
 "ExecutionId": "76a5a04f-caf6-490c-b448-92c02EXAMPLE",
 "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"
 }
]
}
```

#### Windows

```
{
 "AssociationExecutions": [
 {
 "Status": "Success",
 "DetailedStatus": "Success",
 "AssociationId": "c336d2ab-09de-44ba-8f6a-6136cEXAMPLE",
 "ExecutionId": "76a5a04f-caf6-490c-b448-92c02EXAMPLE",
 "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
CreatedTime : 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

AssociationId : c336d2ab-09de-44ba-8f6a-6136cEXAMPLE
AssociationVersion : 1
CreatedTime : 8/11/2019 2:00:54 AM
DetailedStatus : Success
ExecutionId : 791b72e0-f0da-4021-8b35-f95dfEXAMPLE
LastExecutionDate : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=1}
Status : Success

AssociationId : c336d2ab-09de-44ba-8f6a-6136cEXAMPLE
AssociationVersion : 1
CreatedTime : 8/4/2019 2:01:00 AM
DetailedStatus : Success
ExecutionId : ecec60fa-6bb0-4d26-98c7-140308EXAMPLE
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 & macOS

```
aws ssm describe-association-executions \
--association-id ID \
--filters Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=LESS_THAN
```

## Windows

```
aws ssm describe-association-executions ^
```

```
--association-id ID ^
--filters Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=LESS_THAN
```

### PowerShell

```
Get-SMMAssociationExecution ^
-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 & macOS

```
aws ssm describe-association-executions \
--association-id ID \
--filters Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=GREATERTHAN
Key=Status,Value=Success,Type=EQUAL
```

### Windows

```
aws ssm describe-association-executions ^
--association-id ID ^
--filters Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=GREATERTHAN
Key=Status,Value=Success,Type=EQUAL
```

### PowerShell

```
Get-SMMAssociationExecution ^
-AssociationId 14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE ^
-Filter @{
 "Key"="CreatedTime";
 "Value"="2019-06-01T19:15:38.372Z";
 "Type"="GREATERTHAN"
},
@{
 "Key"="Status";
 "Value"="Success";
 "Type"="EQUAL"
}
```

- Run the following command to view all targets where the specific execution ran.

### Linux & macOS

```
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 & macOS

```
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 ^
-ExecutionId 76a5a04f-caf6-490c-b448-92c02EXAMPLE ^
-Filter @{
 "Key"="Status";
 "Value"="Failed"
}
```

The following example returns information about a specific managed node where an association failed to run.

## Linux & macOS

```
aws ssm describe-association-execution-targets \
--association-id ID \
--execution-id ID \
--filters Key=Status,Value=Failed Key=ResourceId,Value="i-02573cafefEXAMPLE" \
Key=ResourceType,Value=ManagedInstance
```

## Windows

```
aws ssm describe-association-execution-targets ^
--association-id ID ^
--execution-id ID ^
--filters Key=Status,Value=Failed Key=ResourceId,Value="i-02573cafefEXAMPLE" ^
Key=ResourceType,Value=ManagedInstance
```

## PowerShell

```
Get-SSMAssociationExecutionTarget ^
-AssociationId 14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE ^
-ExecutionId 76a5a04f-caf6-490c-b448-92c02EXAMPLE ^
-Filter @{
 "Key"="Status";
 "Value"="Success"
},
@{
 "Key"="ResourceId";
 "Value"="i-02573cafefEXAMPLE"
},
@{
 "Key"="ResourceType";
 "Value"="ManagedInstance"
}
```

4. If you're investigating an association that failed to run, you can use the [StartAssociationsOnce API](#) operation 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 & macOS

```
aws ssm start-associations-once \
--association-id ID
```

## Windows

```
aws ssm start-associations-once ^
--association-id ID
```

## PowerShell

```
Start-SSMAssociationsOnce ^
-AssociationId ID
```

## Working with associations using IAM

State Manager, a capability of AWS Systems Manager, uses [targets \(p. 1039\)](#) to choose which instances you configure your associations with. Originally, associations were created by specifying a document name (`Name`) and instance ID (`InstanceId`). This created an association between a document and an instance or managed instance. Associations used to be identified by these parameters. These parameters are now deprecated, but they're still supported. The resources `instance` and `managed-instance` were added as resources to actions with `Name` and `InstanceId`.

AWS Identity and Access Management (IAM) policy enforcement behavior depends on the type of resource specified. Resources for State Manager operations are only enforced based on the passed-in request. State Manager doesn't perform a deep check for the properties of resources in your account. A request is only validated against policy resources if the request parameter contains the specified policy resources. For example, if you specify an instance in the resource block, the policy is enforced if the request uses the `InstanceId` parameter. The `Targets` parameter for each resource in the account isn't checked for that `InstanceId`.

Following are some cases with confusing behavior:

- [DescribeAssociation](#), [DeleteAssociation](#), and [UpdateAssociation](#) use `instance`, `managed-instance`, and document resources to specify the deprecated way of referring to associations. This includes all associations created with the deprecated `InstanceId` parameter.
- [CreateAssociation](#), [CreateAssociationBatch](#), and [UpdateAssociation](#) use `instance` and `managed-instance` resources to specify the deprecated way of referring to associations. This includes all associations created with the deprecated `InstanceId` parameter. The document resource type is part of the deprecated way of referring to associations and is an actual property of an association. This means you can construct IAM policies with Allow or Deny permissions for both Create and Update actions based on document name.

For more information about using IAM policies with Systems Manager, see [Identity and access management for AWS Systems Manager \(p. 1380\)](#) or [Actions, resources, and condition keys for AWS Systems Manager](#) in the *Service Authorization Reference*.

## AWS Systems Manager State Manager walkthroughs

The following walkthroughs demonstrate how to create and configure State Manager associations by using the Systems Manager console or the AWS Command Line Interface (AWS CLI). The walkthroughs also demonstrate how to automatically perform common administrative tasks by using State Manager, a capability of AWS Systems Manager.

### Topics

- [Walkthrough: Creating associations that run MOF files \(p. 1066\)](#)
- [Walkthrough: Creating associations that run Ansible playbooks \(p. 1076\)](#)
- [Walkthrough: Creating associations that run Chef recipes \(p. 1081\)](#)
- [Walkthrough: Automatically update SSM Agent \(CLI\) \(p. 1089\)](#)
- [Walkthrough: Automatically update PV drivers on EC2 instances for Windows Server \(console\) \(p. 1092\)](#)

## Walkthrough: Creating associations that run MOF files

You can run Managed Object Format (MOF) files to enforce a desired state on Windows Server managed nodes with State Manager, a capability of AWS Systems 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 nodes are 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 nodes based on the resources and their values defined in the MOF files. The `AWS-ApplyDSCMofS` document allows you to download and run MOF configuration files from Amazon Simple Storage Service (Amazon S3), a local share, or from a secure website 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 \(p. 1067\)](#)
- [Resolving credentials in MOF files \(p. 1067\)](#)
- [Using tokens in MOF files \(p. 1068\)](#)
- [Prerequisites \(p. 1069\)](#)
- [Creating an association that runs MOF files \(p. 1069\)](#)
- [Troubleshooting \(p. 1073\)](#)
- [Viewing DSC resource compliance details \(p. 1074\)](#)

## Using Amazon S3 to store artifacts

If you're using Amazon S3 to store PowerShell modules, MOF files, compliance reports, or status reports, then the AWS Identity and Access Management (IAM) role used by AWS Systems Manager 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. Below is important information about storing artifacts in Amazon S3.

- If the bucket is in a different AWS account, 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're using Amazon S3 as a module source, 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 aren't supported.

## Resolving credentials in MOF files

Credentials are resolved by using [AWS Secrets Manager](#) or [AWS Systems Manager Parameter Store \(p. 256\)](#). This allows you to set up automatic credential rotation. This also allows DSC to automatically propagate credentials to your servers without redeploying MOFs.

To use an AWS 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. Following is an example.

```
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
 }
 }
}
```

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 node has `GetSecretValue` access in an IAM Managed Policy, and optionally in the Secret Resource Policy if one exists. To work with DSC, the secret must be in the following format.

```
{ '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 allows you to reuse common MOF files on multiple servers that require similar configurations.

Token substitution only works for Resource Properties of type `String`. However, if your resource has a nested CIM node property, it also resolves tokens from `String` properties in that CIM node. 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 your nodes. In the `ComputerName` property, instead of hardcoding the computer name into the MOF, you can use an Instance Tag type token. The value is resolved during MOF parsing. See the following example.

```
Configuration MyConfig
{
 xComputer Computer
 {
 ComputerName = '{tag:ComputerName}'
 }
}
```

You then set a tag on either the managed node in the Systems Manager console, or an Amazon Elastic Compute Cloud (Amazon EC2) tag in the Amazon 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, as shown in the following example.

```
Configuration MyConfig
{
 File MyFile
 {
 DestinationPath = '{env:TMP}\{tag:ComputerName}'
 Type = 'Directory'
 }
}
```

There are five different types of tokens you can use:

- **tag**: Amazon EC2 or managed node 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**: Parameter Store values. Only String and Secure String types are supported.
- **tagssm**: This is the same as tag, but if the tag isn't set on the node, the system tries to resolve the value from a Systems Manager 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 node (for example, one-box deployments).

Here is a Parameter Store example that uses the `ssm` token type.

```
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 nodes 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 nodes have the following prerequisites installed:

- Windows PowerShell version 5.0 or later. For more information, see [Windows PowerShell System Requirements](#) on Microsoft.com.
- [AWS Tools for Windows PowerShell](#) 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

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **State Manager**, and then choose **Create association**.
4. 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.
5. In the **Document** list, choose **AWS-ApplyDSCMofs**.
6. 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:doc-example-bucket:MOF_file_name.mof
```

If you want to specify an AWS Region, then use the following format.

```
s3:bucket_Region:doc-example-bucket:MOF_file_name.mof
```

- A secure website. Specify this information by using the following format.

```
https://domain_name/MOF_file_name.mof
```

Here is an example.

```
https://www.example.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.

```
\StateManagerAssociationsBox\MOFs_folder\MyMof.mof
```

- 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're 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're using a proxy for Amazon S3 endpoints in a specific Region that doesn't include us-east-1, prefix the bucket name with a Region. If the bucket name isn't prefixed, it automatically discovers the bucket Region by using the us-east-1 endpoint.

- d. **Mof Operation Mode:** Choose State Manager behavior when running the **AWS-ApplyDSCMofs** association:
  - **Apply:** Correct node configurations that aren't compliant.
  - **ReportOnly:** Don't correct node configurations, but instead log all compliance data and report nodes 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 a node. 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:doc-example-bucket. If you're using a proxy for Amazon S3 endpoints in a specific Region that doesn't include us-east-1, prefix the bucket name with a Region. If the bucket name isn't prefixed, it automatically discovers 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**, 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:doc-example-bucket. If you're using a proxy for Amazon S3 endpoints in a specific Region that doesn't include us-east-1, prefix the bucket name

with a Region. If the bucket name isn't prefixed, it automatically discovers 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/>. If you choose **False**, 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 node 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 node whenever a MOF execution requests it. If running multiple MOF files that request reboots, then the nodes are rebooted multiple times.
  - **Never:** Nodes aren't rebooted, even if the MOF execution explicitly requests a reboot.
- j. **Use Computer Name For Reporting:** (Optional) Turn on 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 node ID when reporting compliance information.
- k. **Turn on Verbose Logging:** (Optional) We recommend that you turn on verbose logging when deploying MOF files for the first time.

**Important**

When allowed, verbose logging writes more data to your Amazon S3 bucket than standard association execution logging. This might result in slower performance and higher storage charges for Amazon S3. To mitigate storage size issues, we recommend that you turn on 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 User Guide*.

- l. **Turn on Debug Logging:** (Optional) We recommend that you turn on debug logging to troubleshoot MOF failures. We also recommend that you deactivate this option for normal use.

**Important**

When allowed, debug logging writes more data to your Amazon S3 bucket than standard association execution logging. This might result in slower performance and higher storage charges for Amazon S3. To mitigate storage size issues, we recommend that you turn on 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 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** overwrites 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. Separate additional lines by using semicolons.
7. 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 \(p. 1039\)](#).
  8. 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.
  9. In the **Advanced options** section:

- In **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 \(p. 840\)](#).
10. In the **Rate control** section, configure options for running State Manager associations across of fleet of managed nodes. For more information about these options, see [About targets and rate controls in State Manager associations \(p. 1039\)](#).

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.

11. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Enter 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 managed node, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

12. Choose **Create Association**.

State Manager creates and immediately runs the association on the specified nodes or targets. After the initial execution, the association runs in intervals according to the schedule that you defined and according to the following rules:

- State Manager runs associations on nodes that are online when the interval starts and skips offline nodes.
- State Manager attempts to run the association on all configured nodes during an interval.
- If an association isn't run during an interval (because, for example, a concurrency value limited the number of nodes 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, a capability of AWS Systems Manager. For more information, see [Sending commands using Systems Manager Run Command \(p. 995\)](#).

## Troubleshooting

This section includes information to help you troubleshoot issues creating associations that run MOF files.

### Turn on enhanced logging

As a first step to troubleshooting, turn on enhanced logging. More specifically, do the following:

1. Verify that the association is configured to write command output to either Amazon S3 or Amazon CloudWatch Logs (CloudWatch).
2. Set the **Enable Verbose Logging** parameter to True.
3. Set the **Enable Debug Logging** parameter to True.

With verbose and debug logging turned on, 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 wasn't applied

If State Manager failed to apply the association to your nodes, 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 node 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 and 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're using tags, then ensure that the node 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 node 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 node 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 doesn't reflect the compliance status of your node against the configuration requirements in the MOF file. To view the compliance status of your nodes, 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 an allow list. 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:doc-example-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:doc-example-bucket`

- If Region-specific syntax doesn't fix the problem, then make sure that the targeted nodes 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 node 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 node can't access the internet.

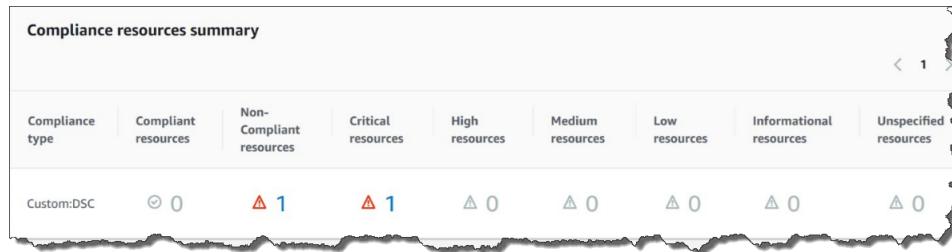
## Viewing DSC resource compliance details

Systems Manager captures compliance information about DSC resource failures in the 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 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 noncompliant.

### 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 noncompliant 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 screenshot shows the 'Details overview for resources' page. At the top, there's a table with columns: ID, Resource type, Compliance type, Overall severity, Overall status, and Execution time. One row is selected, showing 'i-0462a3207a1b63e72' as a ManagedInstance with Custom:DSC compliance, Critical overall severity, Non-compliant status, and execution time on Mon, 20 May 2019 23:50:18 GMT.

Below this is a 'Compliance rule' section with a search bar and filters: Status : Equal : Non-compliant, ComplianceType : Equal : Custom:DSC, Severity : Equal : All, and ResourceId : Equal : i-0462a3207a1b63e72. A red box highlights the 'View output' link next to the third row in the detailed status table.

ID	Compliance type	Resource ID	Severity	Status	Execution time	Detailed status
[Mof]FailingConfig	Custom:DSC	i-0462a3207a1b63e72	Critical	<span style="color: red;">⚠ Non-compliant</span>	Mon, 20 May 2019 23:50:18 GMT	-
[FailingConfig] [Script]EAContinueFailure	Custom:DSC	i-0462a3207a1b63e72	Medium	<span style="color: red;">⚠ Non-compliant</span>	Mon, 20 May 2019 23:50:18 GMT	<span style="border: 1px solid red; padding: 2px;">View output</span>
[FailingConfig][Script]EAStopFailure	Custom:DSC	i-0462a3207a1b63e72	Critical	<span style="color: red;">⚠ Non-compliant</span>	Mon, 20 May 2019 23:50:18 GMT	<span style="border: 1px solid red; padding: 2px;">View output</span>

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 4,000 character quota. This process displays the log messages that were output before the exception was thrown, which are the most relevant messages for troubleshooting.

## View detailed status

```
[2019-05-20 23:50:16.587] LCM: [Start Set]
[2019-05-20 23:50:16.599] Performing the operation "Set-TargetResource" on target "Execu
[2019-05-20 23:50:16.607] WARNING: This resource should fail
[2019-05-20 23:50:16.611] This is verbose message '1' from the SetScript scriptblock
[2019-05-20 23:50:16.612] This is verbose message '2' from the SetScript scriptblock
[2019-05-20 23:50:16.613] This is verbose message '3' from the SetScript scriptblock
[2019-05-20 23:50:16.614] This is verbose message '4' from the SetScript scriptblock
[2019-05-20 23:50:16.616] This is verbose message '5' from the SetScript scriptblock
[2019-05-20 23:50:16.617] This is verbose message '6' from the SetScript scriptblock
[2019-05-20 23:50:16.618] This is verbose message '7' from the SetScript scriptblock
[2019-05-20 23:50:16.619] This is verbose message '8' from the SetScript scriptblock
[2019-05-20 23:50:16.620] This is verbose message '9' from the SetScript scriptblock
[2019-05-20 23:50:16.621] This is verbose message '10' from the SetScript scriptblock
[2019-05-20 23:50:16.649] LCM: [End Set] in 0.0510 seconds.
ERROR: Microsoft.Management.Infrastructure.CimException: PowerShell DSC resource MSFT_Sc
 at Microsoft.Management.Infrastructure.Internal.Operations.CimAsyncObserverProxyBase`
```

For information about how to view compliance information, see [AWS Systems Manager Compliance \(p. 836\)](#).

## 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 fails. For example, if Systems Manager attempts to download a MOF from an Amazon S3 bucket that the node 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.

## Walkthrough: Creating associations that run Ansible playbooks

You can create State Manager associations that run Ansible playbooks by using the AWS-ApplyAnsiblePlaybooks SSM document. State Manager is a capability of AWS Systems Manager. 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 allow 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. Associations that run Ansible playbooks aren't supported on macOS.

### 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, 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, specify the path to the file. Here are two examples.

### Example 1: Download a specific playbook file

```
{
 "path": "https://s3.amazonaws.com/doc-example-bucket/playbook.yml"
}
```

### Example 2: Download the contents of a directory

```
{
 "path": "https://s3.amazonaws.com/doc-example-bucket/ansible/webservers/"
}
```

#### Important

If you specify Amazon S3, then the AWS Identity and Access Management (IAM) instance profile on your managed nodes must be configured with the `AmazonS3ReadOnlyAccess` policy. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#).

#### Support for compressed playbook structure

The `AWS-ApplyAnsiblePlaybooks` document allows 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 your nodes have the following dependencies installed:

- **Ubuntu Server/Debian Server:** Apt-get (Package Management), Python 3, Ansible, Unzip
- **Amazon Linux:** Ansible
- **RHEL:** Python 3, Ansible, Unzip

If one or more of these dependencies aren't found, then Systems Manager automatically installs them.

## 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)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **State Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon ( $\equiv$ ) to open the navigation pane, and then choose **State Manager**.

3. Choose **State Manager**, and then choose **Create association**.
4. For **Name**, specify a name that helps you remember the purpose of the association.
5. In the **Document** list, choose **AWS-ApplyAnsiblePlaybooks**.
6. In the **Parameters** section, for **Source Type**, choose either **GitHub** or **S3**.

### GitHub

If you choose **GitHub**, enter repository information in the following format.

```
{
 "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.

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

7. For **Install Dependencies**, choose an option.
8. (Optional) For **Playbook File**, enter a file name. If a Zip file contains the playbook, specify a relative path to the Zip file.
9. (Optional) For **Extra Variables**, enter variables that you want State Manager to send to Ansible at runtime.
10. (Optional) For **Check**, choose an option.
11. (Optional) For **Verbose**, choose an option.
12. For **Targets**, choose an option. For information about using targets, see [About targets and rate controls in State Manager associations \(p. 1039\)](#).
13. 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.
14. 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 \(p. 840\)](#).
15. In the **Rate control** section, configure options to run State Manager associations across a fleet of managed nodes. For information about using rate controls, see [About targets and rate controls in State Manager associations \(p. 1039\)](#).

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.

16. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Enter 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 managed node, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

17. Choose **Create Association**.

**Note**

If you use tags to create an association on one or more target nodes, and then you remove the tags from a node, that node no longer runs the association. The node is disassociated from the State Manager document.

## Create an association that runs Ansible playbooks (CLI)

The following procedure describes how to use the AWS Command Line Interface (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 Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run one of the following commands to create an association that runs Ansible playbooks by targeting nodes using tags. Command (A) specifies GitHub as the source type. Command (B) specifies Amazon S3 as the source type.

#### (A) GitHub source

Linux & macOS

```
aws ssm create-association --name "AWS-ApplyAnsiblePlaybooks" \
--targets Key=tag:TagKey,Values=TagValue \
--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

```
aws ssm create-association --name "AWS-ApplyAnsiblePlaybooks" ^
--targets Key=tag:TagKey,Values=TagValue ^
```

```
--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.

```
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"]}' \
--association-name "AnsibleAssociation" --schedule-expression "cron(0 2 ? * SUN *)"
```

### (B) S3 source

Linux & macOS

```
aws ssm create-association --name "AWS-ApplyAnsiblePlaybooks" \
--targets Key=tag:TagKey,Values=TagValue \
--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"
```

Windows

```
aws ssm create-association --name "AWS-ApplyAnsiblePlaybooks" ^
--targets Key=tag:TagKey,Values=TagValue ^
--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"
```

Here is an example.

```
aws ssm create-association --name "AWS-ApplyAnsiblePlaybooks" \
--targets "Key=tag:OS,Values=Linux" \
--parameters '{"SourceType":["S3"], "SourceInfo": [{"\"path\": \"https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/playbook.yml\"}], "InstallDependencies": ["True"], "PlaybookFile": ["playbook.yml"], "ExtraVariables": ["SSM=True"], "Check": ["False"], "Verbose": ["-v"]}' \
--association-name "AnsibleAssociation" --schedule-expression "cron(0 2 ? * SUN *)"
```

### Note

State Manager associations don't 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 \(p. 1540\)](#).

The system attempts to create the association on the nodes and immediately apply the state.

3. Run the following command to view an updated status of the association you just created.

```
aws ssm describe-association --association-id "ID"
```

## Walkthrough: Creating associations that run Chef recipes

You can create State Manager associations that run Chef recipes by using the [AWS-ApplyChefRecipes](#) SSM document. State Manager is a capability of AWS Systems Manager. You can target Linux-based Systems Manager managed nodes with the [AWS-ApplyChefRecipes](#) SSM 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 nodes.
- Optionally runs [Systems Manager compliance checks \(p. 836\)](#) on target nodes, and stores the results of compliance checks in an Amazon Simple Storage Service (Amazon 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 change on target nodes without making changes.
- Optionally applies custom JSON attributes to `chef-client` runs.

You can use GitHub or Amazon S3 buckets as sources for Chef cookbooks and recipes that you specify in an [AWS-ApplyChefRecipes](#) document.

**Note**

Associations that run Chef recipes aren't supported on macOS.

### 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 can't officially support cookbooks on the [Chef Supermarket](#) website, many of them work with the [AWS-ApplyChefRecipes](#) document. The following are examples of criteria to determine when you're testing a community cookbook:

- The cookbook should support the Linux-based operating systems of the Systems Manager managed nodes that you're 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, doesn't 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 (SSM document) runs. Using a nested `cookbooks` folder is supported, but not required; you can store cookbooks directly under the root level.

```
<Top-level directory, or the top level of the archive file (ZIP or tgz or tar.gz)>
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 operation 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 nodes 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 Compliance \(p. 836\)](#).

### Logging the document run

When you run a Systems Manager document (SSM 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 (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 Amazon S3 bucket or CloudWatch Logs. For more information, see [Creating associations \(p. 1042\)](#).

### Use GitHub as a cookbook source

The AWS-ApplyChefRecipes document uses the [aws:downloadContent \(p. 1330\)](#) 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.

```
{
 "owner": "TestUser",
 "repository": "GitHubCookbookRepository",
 "path": "cookbooks/HelloWorld",
 "getOptions": "branch:master",
 "tokenInfo": "{{ssm-secure:secure-string-token}}"
}
```

### 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, 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 nodes must be configured with the `AmazonS3ReadOnlyAccess` policy. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#).

#### Topics

- [Create an association that runs Chef recipes \(console\) \(p. 1083\)](#)
- [Create an association that runs Chef recipes \(CLI\) \(p. 1085\)](#)
- [Viewing Chef resource compliance details \(p. 1087\)](#)

### 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.

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
  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 **State Manager**, and then choose **Create association**.
  4. For **Name**, enter a name that helps you remember the purpose of the association.
  5. In the **Document** list, choose **AWS-ApplyChefRecipes**.
  6. In **Parameters**, for **Source Type**, choose either **GitHub** or **S3**.
  7. In **Source info**, enter cookbook source information in one of the following formats.

- a. 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"
}
```

- b. If you chose **S3** in step 5, enter path information in the following format.

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

8. In **Run list**, list the recipes that you want to run in the following format, separating each recipe with a comma as shown. Don't include a space after the comma.

```
recipe[cookbook_name1::recipe_name],recipe[cookbook_name2::recipe_name]
```

9. (Optional) In **JSON attributes content**, add any custom JSON that contains attributes you want the Chef client to pass to your target nodes.

The **JSON attributes content** parameter is best used for the following purposes:

- When you want to override a small number of attributes and you don't 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.

10. 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 nodes. If you specify None, Systems Manager doesn't install the Chef client on target nodes before running the document's recipes. The default value is 14.
11. (Optional) For **Chef client arguments**, specify additional arguments that are supported for the version of Chef you're using. To learn more about supported arguments, run `chef-client -h` on a node that is running the Chef client.
12. (Optional) Turn on **Why-run** to show changes made to target nodes if the recipes are run, without actually changing target nodes.
13. For **Compliance severity**, choose the severity of Systems Manager Compliance results that you want reported. Compliance reporting indicates whether the association state is compliant or noncompliant, along with the severity level you specify. Compliance reports are stored in an S3 bucket that you specify as the value of the **Compliance report bucket** parameter (step 14). For more information about Compliance, see [Working with Compliance \(p. 839\)](#) in this guide.

Compliance scans measure drift between configuration that is specified in your Chef recipes and node resources. Valid values are Critical, High, Medium, Low, Informational, Unspecified, or None. To skip compliance reporting, choose None.

14. 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.
15. 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 Compliance results.
16. In **Rate control**, configure options to run State Manager associations across a fleet of managed nodes. For information about using rate controls, see [About targets and rate controls in State Manager associations \(p. 1039\)](#).

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.
17. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Enter 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 managed node, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

18. Choose **Create Association**.

## Create an association that runs Chef recipes (CLI)

The following procedure describes how to use the AWS Command Line Interface (AWS CLI) to create a State Manager association that runs Chef cookbooks by using the `AWS-ApplyChefRecipes` document.

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run one of the following commands to create an association that runs Chef cookbooks by targeting nodes using tags. Command (A) uses GitHub as the source type. Command (B) uses Amazon S3 as the source type.

### (A) GitHub source

Linux & macOS

```
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": ["DOC-EXAMPLE-BUCKET"]}]' \
--association-name "name" --schedule-expression "cron_or_rate_expression"
```

Windows

```
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": ["DOC-EXAMPLE-BUCKET"]}]' ^
--association-name "name" --schedule-expression "cron_or_rate_expression"
```

Here is an example.

Linux & macOS

```
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": [{"state": "visible","colors": {"foreground": \
"light-blue","background": "dark-gray"}}, "ChefClientVersion": [14], \
"CookbookArguments": [{"fips}"], "WhyRun": false, "ComplianceSeverity": \
[Medium], "ComplianceType": [Custom:Chef], "ComplianceReportBucket": \
["ChefComplianceResultsBucket"]}' \
--association-name "MyChefAssociation" --schedule-expression "cron(0 2 ? * SUN *)"
```

Windows

```
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": [{"state": "visible","colors": {"foreground": \
"light-blue","background": "dark-gray"}}, "ChefClientVersion": [14], \
"CookbookArguments": [{"fips}"], "WhyRun": false, "ComplianceSeverity": \
[Medium], "ComplianceType": [Custom:Chef], "ComplianceReportBucket": \
["ChefComplianceResultsBucket"]}' ^
--association-name "MyChefAssociation" --schedule-expression "cron(0 2 ? * SUN *)"
```

## (B) S3 source

Linux & macOS

```
aws ssm create-association --name "AWS-ApplyChefRecipes" \
--targets Key=tag:TagKey,Values=TagValue \
--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": \
[{"Custom_JSON"}], "ChefClientVersion": [version_number], "ChefClientArguments": \
[{"chef_client_arguments"}], "WhyRun": true_or_false, "ComplianceSeverity": \
[severity_value], "ComplianceType": [Custom:Chef], "ComplianceReportBucket": \
[DOC-EXAMPLE-BUCKET]}' \
--association-name "name" --schedule-expression "cron_or_rate_expression"
```

Windows

```
aws ssm create-association --name "AWS-ApplyChefRecipes" ^
--targets Key=tag:TagKey,Values=TagValue ^
--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": \
[{"Custom_JSON"}], "ChefClientVersion": [version_number], "ChefClientArguments": \
[{"chef_client_arguments"}], "WhyRun": true_or_false, "ComplianceSeverity": \
[severity_value]}
```

```
["severity_value"], "ComplianceType": ["Custom:Chef"], "ComplianceReportBucket":
["DOC-EXAMPLE-BUCKET"]}]' ^
--association-name "name" --schedule-expression "cron_or_rate_expression"
```

Here is an example.

Linux & macOS

```
aws ssm create-association --name "AWS-ApplyChefRecipes" \
--targets "Key=tag:OS,Values= Linux" \
--parameters '{"SourceType":["S3"],"SourceInfo":["{\\"path\\":\\"https://
s3.amazonaws.com/DOC-EXAMPLE-BUCKET>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 *)"
```

Windows

```
aws ssm create-association --name "AWS-ApplyChefRecipes" ^
--targets "Key=tag:OS,Values= Linux" ^
--parameters '{"SourceType":["S3"],"SourceInfo":["{\\"path\\":\\"https://
s3.amazonaws.com/DOC-EXAMPLE-BUCKET>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 don't 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 \(p. 1540\)](#).

The system attempts to create the association on the nodes and immediately apply the state.

3. Run the following command to view an updated status of the association you just created.

```
aws ssm describe-association --association-id "ID"
```

## Viewing Chef resource compliance details

Systems Manager captures compliance information about Chef-managed resources in the 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, systemd 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.

Compliance resources summary						
Compliance type	Compliant resources	Non-Compliant resources	Critical resources	High resources	Medium resources	Low resources
Custom:Chef	✓ 1	⚠ 0	⚠ 0	⚠ 0	⚠ 0	⚠ 0

The **Details overview for resources** section shows information about the AWS resource that isn't 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

Resource	ID	Resource type	Compliance type	Overall severity	Overall status
 i-0 	ManagedInstance	Custom:Chef	Critical	 Compliant	
<b>Compliance rule</b>					
<input data-bbox="372 656 396 688" type="text"/> <span>All</span>					
Status : Equal : Compliant		ComplianceType : Equal : Custom:Chef	Severity : Equal : All	ResourceId : Equal : i-0	
ID	Compliance type	Resource ID	Severity	Status	
aws-site::install-nginx::nginx	Custom:Chef	i-0 	Critical	 Compliant	
aws-site::install-nginx::nginx	Custom:Chef	i-0 	Critical	 Compliant	
aws-site::install-nginx::/var/www/html/	Custom:Chef	i-0 	Critical	 Compliant	
aws-site::install-nginx::/etc/nginx/nginx.conf	Custom:Chef	i-0 	Critical	 Compliant	
aws-site::deploy-app::/usr/share/nginx/html/index.html	Custom:Chef	i-0 	Critical	 Compliant	

**View output** shows the last 4,000 characters of the detailed status. Systems Manager starts with the exception as the first element, finds verbose messages, and shows them until it reaches the 4,000 character quota. 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 Compliance \(p. 836\)](#).

### 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 node doesn't have permission to access, the association fails, and Systems Manager reports no compliance data.

## Walkthrough: 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. The association automatically updates the SSM Agent according to a schedule that you specify. For more information about SSM Agent, see [Working with SSM Agent \(p. 68\)](#). To customize the update schedule for SSM Agent using the console, see [Automatically updating SSM Agent \(p. 136\)](#).

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 Amazon Elastic Compute Cloud (Amazon EC2) instance for Linux, macOS, or Windows Server that is configured for Systems Manager. For more information, see [Systems Manager prerequisites \(p. 13\)](#).

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 Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to create an association by targeting instances using Amazon Elastic Compute Cloud (Amazon EC2) tags. The `Schedule` parameter sets a schedule to run the association every Sunday morning at 2:00 a.m. (UTC).

State Manager associations don't 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 \(p. 1540\)](#).

#### Linux & macOS

```
aws ssm create-association \
--targets Key=tag:tag_key,Values=tag_value \
--name AWS-UpdateSSMAgent \
--schedule-expression "cron(0 2 ? * SUN *)"
```

#### Windows

```
aws ssm create-association ^
--targets Key=tag:tag_key,Values=tag_value ^
--name AWS-UpdateSSMAgent ^
--schedule-expression "cron(0 2 ? * SUN *)"
```

You can target multiple instances by specifying instances IDs in a comma-separated list.

#### Linux & macOS

```
aws ssm create-association \
--targets Key=instanceids,Values=instance_ID,instance_ID,instance_ID \
--name AWS-UpdateSSMAgent \
--schedule-expression "cron(0 2 ? * SUN *)"
```

#### Windows

```
aws ssm create-association ^
--targets Key=instanceids,Values=instance_ID,instance_ID,instance_ID ^
--name AWS-UpdateSSMAgent ^
--schedule-expression "cron(0 2 ? * SUN *)"
```

You can specify the version of the SSM Agent you want to update to.

Linux & macOS

```
aws ssm create-association \
--targets Key=instanceids,Values=instance_ID,instance_ID,instance_ID \
--name AWS-UpdateSSMAgent \
--schedule-expression "cron(0 2 ? * SUN *)" \
--parameters version=ssm_agent_version_number
```

Windows

```
aws ssm create-association ^
--targets Key=instanceids,Values=instance_ID,instance_ID,instance_ID ^
--name AWS-UpdateSSMAgent ^
--schedule-expression "cron(0 2 ? * SUN *)" ^
--parameters version=ssm_agent_version_number
```

The system returns information like the following.

```
{
 "AssociationDescription": {
 "ScheduleExpression": "cron(0 2 ? * SUN *)",
 "Name": "AWS-UpdateSSMAgent",
 "Overview": {
 "Status": "Pending",
 "DetailedStatus": "Creating"
 },
 "AssociationId": "123.....",
 "DocumentVersion": "$DEFAULT",
 "LastUpdateAssociationDate": 1504034257.98,
 "Date": 1504034257.98,
 "AssociationVersion": "1",
 "Targets": [
 {
 "Values": [
 "TagValue"
],
 "Key": "tag:TagKey"
 }
]
 }
}
```

The system attempts to create the association on the instance(s) and applies the state following creation. The association status shows Pending.

3. Run the following command to view an updated status of the association you created.

```
aws ssm list-associations
```

If your instances *aren't* running the most recent version of the SSM Agent, the status shows Failed. 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 Amazon Machine Images (AMIs) contain a set of drivers to permit access to virtualized hardware. These drivers are used by Amazon Elastic Compute Cloud (Amazon EC2) to map instance store and Amazon Elastic Block Store (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. State Manager is a capability of AWS Systems Manager.

### Before you begin

Before you complete the following procedure, verify that you have at least one Amazon EC2 instance for Windows Server running that is configured for Systems Manager. For more information, see [Systems Manager prerequisites \(p. 13\)](#).

### To create a State Manager association that automatically updates PV drivers

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Name** field, enter a descriptive name.
5. In the **Document** list, choose **AWS-ConfigureAWSPackage**.
6. In the **Parameters** section, choose **Install** from the **Action** list.
7. For **Installation type**, choose **Uninstall and reinstall**.
8. In the **Name** field, enter **AWSPVDriver**. You can keep the **Version** and **Additional Arguments** fields empty.
9. In the **Targets** 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**.

10. In the **Specify schedule** section, choose an option. Updated PV drivers are released a several times a year, so you can schedule the association to run once a month, if you want.
11. 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 \(p. 840\)](#).
12. For **Rate control**:
  - For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

#### Note

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then

restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes still processing the command might also send errors.
13. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Enter 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 managed node, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

14. Choose **Create association**, and then choose **Close**. The system attempts to create the association on the instances and immediately apply the state.

If you created the association on one or more Amazon EC2 instances for Windows Server, the status changes to **Success**. If your instances aren't configured for Systems Manager, or if you inadvertently targeted Linux instances, the status shows **Failed**.

If the status is **Failed**, choose the association ID, choose the **Resources** tab, and then 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 the SSM Agent is running on the instance, and verify that the instance is configured with an AWS Identity and Access Management (IAM) role for Systems Manager. For more information, see [Systems Manager prerequisites \(p. 13\)](#).

## AWS Systems Manager Patch Manager

Patch Manager, a capability of AWS Systems Manager, automates the process of patching managed nodes 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 applications released by Microsoft.) You can use Patch Manager to install Service Packs on Windows nodes and perform minor version upgrades on Linux nodes. You can patch fleets of Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, or your on-premises servers and virtual machines (VMs) by operating system type. This includes supported versions of Amazon Linux, Amazon Linux 2, CentOS, Debian Server, macOS, Oracle Linux, Raspberry Pi OS (formerly Raspbian), Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), Ubuntu Server, and Windows Server. You can scan instances to see only a report of missing patches, or you can scan and automatically install all missing patches. To get started with Patch Manager, open the [Systems Manager console](#). In the navigation pane, choose **Patch Manager**.

**Important**

AWS doesn't test patches 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.

For Linux-based operating system types that report a severity level for patches, Patch Manager uses the severity level reported by the software publisher for the update notice or individual patch. Patch Manager doesn't derive severity levels from third-party sources, such as the [Common Vulnerability Scoring System \(CVSS\)](#), or from metrics released by the [National Vulnerability Database \(NVD\)](#).

Patch Manager uses *patch baselines*, which include rules for auto-approving patches within days of their release, in addition to 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 managed nodes by using 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 provides options to scan your managed nodes and report compliance on a schedule, install available patches on a schedule, and patch or scan targets on demand whenever you need to. You can also generate patch compliance reports that are sent to an Amazon Simple Storage Service (Amazon S3) bucket of your choice. You can generate one-time reports, or generate reports on a regular schedule. For a single managed node, reports include details of all patches for the node. For a report on all managed nodes, only a summary of how many patches are missing is provided.

Patch Manager integrates with AWS Identity and Access Management (IAM), AWS CloudTrail, and Amazon EventBridge 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 \(p. 1442\)](#).

For information about using EventBridge to monitor Systems Manager events, see [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#).

### Topics

- [Patch Manager prerequisites \(p. 1094\)](#)
- [How Patch Manager operations work \(p. 1096\)](#)
- [About SSM documents for patching managed nodes \(p. 1124\)](#)
- [About patch baselines \(p. 1156\)](#)
- [Using Kernel Live Patching on Amazon Linux 2 managed nodes \(p. 1168\)](#)
- [Working with Patch Manager \(console\) \(p. 1174\)](#)
- [Working with Patch Manager \(AWS CLI\) \(p. 1210\)](#)
- [AWS Systems Manager Patch Manager walkthroughs \(p. 1234\)](#)
- [Troubleshooting Patch Manager \(p. 1245\)](#)

## Patch Manager prerequisites

Make sure that you have met the required prerequisites before using Patch Manager, a capability of AWS Systems Manager.

### SSM Agent version

Version 2.0.834.0 or later of SSM Agent is running on the managed node 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. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

### Connectivity to the patch source

If your managed nodes don't have a direct connection to the Internet and you're using an Amazon Virtual Private Cloud (Amazon VPC) with a VPC endpoint, you must ensure that the nodes have access to the source patch repositories (repos). On Linux nodes, patch updates are typically downloaded from the

remote repos configured on the node. Therefore, the node must be able to connect to the repos so the patching can be performed. For more information, see [How security patches are selected \(p. 1097\)](#).

Windows Server managed nodes must be able to connect to the Windows Update Catalog or Windows Server Update Services (WSUS). Confirm that your nodes have connectivity to the [Microsoft Update Catalog](#) through an internet gateway, NAT gateway, or NAT instance. If you are using WSUS, confirm that the node has connectivity to the WSUS server in your environment. For more information, see [Issue: managed node doesn't have access to Windows Update Catalog or WSUS \(p. 1249\)](#).

### S3 endpoint access

Whether your managed nodes operate in a private or public network, without access to the required AWS managed Amazon Simple Storage Service (Amazon S3) buckets, patching operations fail. For information about the S3 buckets your managed nodes must be able to access, see [SSM Agent communications with AWS managed S3 buckets \(p. 138\)](#) and [Step 6: \(Optional\) Create a VPC endpoint \(p. 28\)](#).

### Supported operating systems

The Patch Manager capability doesn't support all the same operating systems versions that are supported by other Systems Manager capabilities. For example, Patch Manager doesn't support CentOS 6.3 or Raspberry Pi OS 8 (Jessie). (For the full list of Systems Manager-supported operating systems, see [Systems Manager prerequisites \(p. 13\)](#).) Therefore, ensure that the managed nodes you want to use with Patch Manager are running one of the operating systems listed in the following table.

Operating system	Details
Linux	<ul style="list-style-type: none"><li>Amazon Linux 2012.03 - 2018.03</li><li>Amazon Linux 2 2 - 2.0</li><li>CentOS 6.5 - 7.9, 8.0 - 8.5</li><li>CentOS Stream 8</li><li>CentOS Stream 8</li><li>Debian Server 8.x, 9.x, and 10.x</li><li>Oracle Linux 7.5 - 8.3</li><li>Raspberry Pi OS (formerly Raspbian) 9 (Stretch) and 10 (Buster)</li><li>Red Hat Enterprise Linux (RHEL) 6.5 - 8.5</li><li>Rocky Linux 8.4 and 8.5</li><li>SUSE Linux Enterprise Server (SLES) 12.0 and later 12.x versions, 15.0 and 15.1</li><li>Ubuntu Server 14.04 LTS, 16.04 LTS, 18.04 LTS, 20.04 LTS, and 20.10 STR</li></ul>
macOS	<p><b>Note</b> Managed nodes created from an Amazon Linux AMI that use a proxy must run a current version of the <code>Python requests</code> module to support Patch Manager operations. For more information, see <a href="#">Upgrading the Python requests module on Amazon Linux instances that use a proxy server (p. 119)</a>.</p> <p>macOS 10.14.x (Mojave), 10.15.x (Catalina), and 11.3.1 &amp; 11.4 (Big Sur)</p>

Operating system	Details
	SSM Agent for AWS IoT Greengrass core devices is not supported on macOS. You can't use Patch Manager to patch macOS edge devices.
Windows	<p>Windows Server 2008 through Windows Server 2022, including R2 versions.</p> <p><b>Important</b>  As of January 14, 2020, Windows Server 2008 is no longer supported for feature or security updates from Microsoft. Legacy Amazon Machine Images (AMIs) for Windows Server 2008 and 2008 R2 still include version 2 of SSM Agent preinstalled, but Systems Manager no longer officially supports 2008 versions and no longer updates the agent for these versions of Windows Server. In addition, <a href="#">SSM Agent version 3.0 (p. 75)</a> might not be compatible with all operations on Windows Server 2008 and 2008 R2. The final officially supported version of SSM Agent for Windows Server 2008 versions is 2.3.1644.0.</p> <p>SSM Agent for AWS IoT Greengrass core devices is not supported on Windows 10. You can't use Patch Manager to patch Windows 10 edge devices.</p>

## How Patch Manager operations work

This section provides technical details that explain how Patch Manager, a capability of AWS Systems 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 a managed node. This section also provides details about how patch baseline rules work on different distributions of the Linux operating system.

The information in the following topics applies both when you're patching managed nodes on a schedule and patching nodes on demand.

### Topics

- [How security patches are selected \(p. 1097\)](#)
- [How to specify an alternative patch source repository \(Linux\) \(p. 1101\)](#)
- [How patches are installed \(p. 1103\)](#)
- [How patch baseline rules work on Linux-based systems \(p. 1111\)](#)
- [Key differences between Linux and Windows patching \(p. 1123\)](#)

## How security patches are selected

The primary focus of Patch Manager, a capability of AWS Systems Manager, is on installing operating systems security-related updates on managed nodes. By default, Patch Manager doesn't install all available patches, but rather a smaller set of patches focused on security.

For Linux-based operating system types that report a severity level for patches, Patch Manager uses the severity level reported by the software publisher for the update notice or individual patch. Patch Manager doesn't derive severity levels from third-party sources, such as the [Common Vulnerability Scoring System](#) (CVSS), or from metrics released by the [National Vulnerability Database](#) (NVD).

### Note

On all Linux-based systems supported by Patch Manager, you can choose a different source repository configured for the managed node, typically to install nonsecurity updates. For information, see [How to specify an alternative patch source repository \(Linux\) \(p. 1101\)](#).

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 managed node. There are usually two preconfigured repositories (repos) on a node:

- **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 managed node. Therefore, the node must be able to connect to the repos so the patching can be performed.

Amazon Linux and Amazon Linux 2 managed nodes 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 aren't 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 aren't 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 managed node. The following list provides examples for a fictitious CentOS 8.2 Amazon Machine Image (AMI):

- **Repo ID:** example-centos-8.2-base

**Repo name:** Example CentOS-8.2 - Base

- **Repo ID:** example-centos-8.2-extras

**Repo name:** Example CentOS-8.2 - Extras

- **Repo ID:** example-centos-8.2-updates

**Repo name:** Example CentOS-8.2 - Updates

- **Repo ID:** example-centos-8.x-examplerepo

**Repo name:** Example CentOS-8.x - Example Repo Packages

**Note**

All updates are downloaded from the remote repos configured on the managed node. Therefore, the node must be able to connect to the repos so the patching can be performed.

CentOS 6 and 7 managed nodes use Yum as the package manager. CentOS 8 nodes use DNF as the package manager. Both package managers use the concept of an update notice. An update notice is simply a collection of packages that fix specific problems.

However, CentOS default repos aren't configured with an update notice. This means that Patch Manager doesn't detect packages on a default CentOS repo. To allow Patch Manager to process packages that aren't contained in an update notice, you must turn on the `EnableNonSecurity` flag in the patch baseline rules.

**Note**

CentOS update notices are supported. Repos with update notices can be downloaded after launch.

### Debian Server and Raspberry Pi OS

On Debian Server and Raspberry Pi OS (formerly Raspbian), 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. This means that on Debian Server 8, Patch Manager only identifies upgrades that are part of `debian-security jessie`. On Debian Server 9, only upgrades that are part of `debian-security stretch` are identified. On Debian Server 10, only upgrades that are part of `debian-security buster` are identified.

**Note**

On Debian Server 8 only: Because some Debian Server 8.\* managed nodes refer to an obsolete package repository (`jessie-backports`), Patch Manager performs additional steps to ensure that patching operations succeed. For more information, see [How patches are installed \(p. 1103\)](#).

### Oracle Linux

On Oracle Linux, the Systems Manager patch baseline service uses preconfigured repositories (repos) on the managed node. There are usually two preconfigured repos on a node.

#### Oracle Linux 7:

- **Repo ID:** o17\_UEKR5/x86\_64

**Repo name:** Latest Unbreakable Enterprise Kernel Release 5 for Oracle Linux 7Server (x86\_64)

- **Repo ID:** o17\_latest/x86\_64

**Repo name:** Oracle Linux 7Server Latest (x86\_64)

**Oracle Linux 8:**

- **Repo ID:** ol8\_baseos\_latest

**Repo name:** Oracle Linux 8 BaseOS Latest (x86\_64)

- **Repo ID:** ol8\_appstream

**Repo name:** Oracle Linux 8 Application Stream (x86\_64)

- **Repo ID:** ol8\_UEKR6

**Repo name:** Latest Unbreakable Enterprise Kernel Release 6 for Oracle Linux 8 (x86\_64)

**Note**

All updates are downloaded from the remote repos configured on the managed node. Therefore, the node must be able to connect to the repos so the patching can be performed.

Oracle Linux managed nodes 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. Individual packages aren't assigned classifications or severity levels. For this reason, Patch Manager assigns the attributes of an update notice to the related packages and installs packages based on the Classification filters specified in the patch baseline.

**Note**

If you select the **Approved patches include non-security updates** check box in the **Create patch baseline** page, then packages that aren't 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.

RHEL, CentOS Stream, and Rocky Linux

On Red Hat Enterprise Linux, CentOS Stream, and Rocky Linux the Systems Manager patch baseline service uses preconfigured repositories (repos) on the managed node. There are usually three preconfigured repos on a node.

All updates are downloaded from the remote repos configured on the managed node. Therefore, the node must be able to connect to the repos so the patching can be performed.

**Note**

If you select the **Approved patches include non-security updates** check box in the **Create patch baseline** page, then packages that aren't 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.

Red Hat Enterprise Linux 7 managed nodes use Yum as the package manager. Red Hat Enterprise Linux 8, CentOS Stream, and Rocky Linux managed nodes 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. Individual packages aren't assigned classifications or severity levels. For this reason, Patch Manager assigns the attributes of an update notice to the related packages and installs packages based on the Classification filters specified in the patch baseline.

Repo locations differ between RHEL 7 and the RHEL 8, CentOS Stream, and Rocky Linux systems:

## 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 managed nodes from, you might need to update your commands. For more information, see [Repository IDs for RHEL 7 in AWS Have Changed 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, CentOS Stream, and Rocky Linux

- **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) managed nodes, 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 managed nodes 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 managed node. 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 `codename-security` repos, where the codename is unique to the release version, such as `trusty` for Ubuntu Server 14. Patch Manager only identifies upgrades that are part of these repos:

- Ubuntu Server 14.04 LTS: `trusty-security`

- Ubuntu Server 16.04 LTS: `xenial-security`
- Ubuntu Server 18.04 LTS: `bionic-security`
- Ubuntu Server 20.04 LTS: `focal-security`
- Ubuntu Server 20.10 STR: `groovy-gorilla`

## Windows

On Microsoft Windows operating systems, Patch Manager retrieves a list of available updates that Microsoft publishes to Microsoft Update and are automatically available to Windows Server Update Services (WSUS).

Patch Manager continually 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 were 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.

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.

### Note

In some cases, Microsoft releases patches for applications that might not specify explicitly an updated date and time. In these cases, an updated date and time of 01/01/1970 is supplied by default.

## How to specify an alternative patch source repository (Linux)

When you use the default repositories configured on a managed node for patching operations, Patch Manager, a capability of AWS Systems Manager, scans for or installs security-related patches. This is the default behavior for Patch Manager. For complete information about how Patch Manager selects and installs security patches, see [How security patches are selected \(p. 1097\)](#).

On Linux systems, however, you can also use Patch Manager to install patches that aren't related to security, or that are in a different source repository than the default one configured on the managed node. 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 managed nodes. 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 for a managed node doesn't make them the new default repositories on the operating system. After the patching operation is complete, the repositories previously configured as the defaults for the node's operating system remain the defaults.

For a list of example scenarios for using this option, see [Sample uses for alternative patch source repositories \(p. 1102\)](#) later in this topic.

For information about default and custom patch baselines, see [About predefined and custom patch baselines \(p. 1157\)](#).

### Example: Using the console

To specify alternative patch source repositories when you're working in the Systems Manager console, use the **Patch sources** section on the [Create patch baseline](#) page. For information about using the **Patch sources** options, see [Creating a custom patch baseline \(Linux\) \(p. 1195\)](#).

#### Example: Using the AWS CLI

For an example of using the `--sources` option with the AWS Command Line Interface (AWS CLI), see [Create a patch baseline with custom repositories for different OS versions \(p. 1212\)](#).

#### Topics

- [Important considerations for alternative repositories \(p. 1102\)](#)
- [Sample uses for alternative patch source repositories \(p. 1102\)](#)

### 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 a managed node. 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 managed nodes, the default repositories are `amzn2-core` and `amzn2extra-docker`. 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 for a managed node doesn't make them the new default repositories on the operating system. After the patching operation is complete, the repositories previously configured as the defaults for the node's operating system remain the defaults.

#### 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 won't 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 won't 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're already using Patch Manager to install security patches on a fleet of Ubuntu Server managed nodes 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 nodes, 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.

### Example 2 - Personal Package Archives (PPA) for Ubuntu Server

Your Ubuntu Server managed nodes 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 managed node as the source repository for the patching operation. Then use Run Command to run the patch baseline document on the nodes.

### Example 3 – Internal Corporate Applications on Amazon Linux

You need to run some applications needed for industry regulatory compliance on your Amazon Linux managed nodes. You can configure a repository for these applications on the nodes, 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 managed node. 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, a capability of AWS Systems Manager, uses the appropriate built-in mechanism for an operating system type to install updates on a managed node. 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 managed nodes, the patch installation workflow is as follows:

1. If a list of patches is specified using an https URL or an Amazon Simple Storage Service (Amazon S3) path-style URL using the `InstallOverrideList` parameter for the `AWS-RunPatchBaseline` or `AWS-RunPatchBaselineAssociation` documents, the listed patches are installed and steps 2-7 are skipped.
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.

Approval rules, however, are also subject to whether the **Include nonsecurity updates** check box was selected when creating or last updating a patch baseline.

If nonsecurity updates are excluded, 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.

If nonsecurity updates are included, patches from other repositories are considered as well.

4. Apply [ApprovedPatches](#) as specified in the patch baseline. The approved patches are approved for update even if they're 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 won't be applied.
6. If multiple versions of a patch are approved, the latest version is applied.
7. 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=critical,important --bugfix -y
```

- For custom patch baselines where the **Approved patches include non-security updates** check box *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
```

8. The managed node is rebooted if any updates were installed. (Exception: If the `RebootOption` parameter is set to `NoReboot` in the `AWS-RunPatchBaseline` document, the managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).)

## CentOS

On CentOS managed nodes, the patch installation workflow is as follows:

1. If a list of patches is specified using an https URL or an Amazon Simple Storage Service (Amazon S3) path-style URL using the `InstallOverrideList` parameter for the `AWS-RunPatchBaseline` or `AWS-RunPatchBaselineAssociation` documents, the listed patches are installed and steps 2–7 are skipped.

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.

Approval rules, however, are also subject to whether the **Include nonsecurity updates** check box was selected when creating or last updating a patch baseline.

If nonsecurity updates are excluded, 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.

If nonsecurity updates are included, patches from other repositories are considered as well.

3. Apply [ApprovedPatches](#) as specified in the patch baseline. The approved patches are approved for update even if they're 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 YUM update API (on CentOS 6.x and 7.x versions) or the DNF update (on CentOS 8) is applied to approved patches.

7. The managed node is rebooted if any updates were installed. (Exception: If the `RebootOption` parameter is set to `NoReboot` in the `AWS-RunPatchBaseline` document, the managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).)

## Debian Server and Raspberry Pi OS

On Debian Server and Raspberry Pi OS (formerly Raspbian) instances, the patch installation workflow is as follows:

1. If a list of patches is specified using an https URL or an Amazon Simple Storage Service (Amazon S3) path-style URL using the `InstallOverrideList` parameter for the `AWS-RunPatchBaseline` or `AWS-RunPatchBaselineAssociation` documents, the listed patches are installed and steps 2–7 are skipped.
2. If an update is available for `python3-apt` (a Python library interface to `libapt`), it is upgraded to the latest version. (This nonsecurity package is upgraded even if you did not select the **Include nonsecurity updates** option.)

### Important

On Debian Server 8 only: Because some Debian Server 8.\* managed nodes refer to an obsolete package repository (`jessie-backports`), Patch Manager performs the following additional steps to ensure that patching operations succeed:

- a. On your managed node, 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 Server 8.\* distributions.
- c. The `apt-get` operation is run at this point to ensure that the latest version of `python3-apt` is installed before the patching process begins.
- d. After the installation process is complete, 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.

3. Apply [GlobalFilters](#) as specified in the patch baseline, keeping only the qualified packages for further processing.
4. Apply [ApprovalRules](#) as specified in the patch baseline. Each approval rule can define a package as approved.

### Note

Because it isn't possible to reliably determine the release dates of update packages for Debian Server, the auto-approval options aren't supported for this operating system.

Approval rules, however, are also subject to whether the **Include nonsecurity updates** check box was selected when creating or last updating a patch baseline.

If nonsecurity updates are excluded, 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.

If nonsecurity updates are included, patches from other repositories are considered as well.

### Note

For Debian Server and Raspberry Pi OS, patch candidate versions are limited to patches included in `debian-security`.

5. Apply [ApprovedPatches](#) as specified in the patch baseline. The approved patches are approved for update even if they're discarded by [GlobalFilters](#) or if no approval rule specified in [ApprovalRules](#) grants it approval.

6. Apply [RejectedPatches](#) as specified in the patch baseline. The rejected patches are removed from the list of approved patches and won't be applied.
7. The APT library is used to upgrade packages.
8. The managed node is rebooted if any updates were installed. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).)

## macOS

On macOS managed nodes, the patch installation workflow is as follows:

1. The `/Library/Receipts/InstallHistory.plist` property list is a record of software that has been installed and upgraded using the `softwareupdate` and `installer` package managers. Using the `pkgutil` command line tool (for `installer`) and the `softwareupdate` package manager, CLI commands are run to parse this list.

For `installer`, the response to the CLI commands includes `package name, version, volume, location, and install-time details`, but only the `package name` and `version` are used by Patch Manager.

For `softwareupdate`, the response to the CLI commands includes the `package name (display name), version, and date`, but only the `package name` and `version` are used by Patch Manager.

For Brew and Brew Cask, Homebrew doesn't support its commands running under the root user. As a result, Patch Manager queries for and runs Homebrew commands as either the owner of the Homebrew directory or as a valid user belonging to the Homebrew directory's owner group. The commands are similar to `softwareupdate` and `installer` and are run through a Python subprocess to gather package data, and the output is parsed to identify package names and versions.

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.
4. Apply [ApprovedPatches](#) as specified in the patch baseline. The approved patches are approved for update even if they're 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 won't be applied.
6. If multiple versions of a patch are approved, the latest version is applied.
7. Invokes the appropriate package CLI on the managed node to process approved patches as follows:

### Note

`installer` lacks the functionality to check for and install updates. Therefore, for `installer`, Patch Manager only reports which packages are installed. As a result, `installer` packages are never reported as `Missing`.

- 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 security updates are applied.
  - For custom patch baselines where the **Approved patches include non-security updates** check box *is* selected, both security and nonsecurity updates are applied.
8. The managed node is rebooted if any updates were installed. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the managed

node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).)

### Oracle Linux

On Oracle Linux managed nodes, the patch installation workflow is as follows:

1. If a list of patches is specified using an https URL or an Amazon Simple Storage Service (Amazon S3) path-style URL using the `InstallOverrideList` parameter for the `AWS-RunPatchBaseline` or `AWS-RunPatchBaselineAssociation` documents, the listed patches are installed and steps 2-7 are skipped.
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.

Approval rules, however, are also subject to whether the **Include nonsecurity updates** check box was selected when creating or last updating a patch baseline.

If nonsecurity updates are excluded, 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.

If nonsecurity updates are included, patches from other repositories are considered as well.

4. Apply [ApprovedPatches](#) as specified in the patch baseline. The approved patches are approved for update even if they're 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 won't be applied.
6. If multiple versions of a patch are approved, the latest version is applied.
7. On version 7 managed nodes, 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** check box 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
```

On version 8 managed nodes, the Dnf 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 dnf upgrade-minimal --security --sec-severity Moderate --sec-severity Important
```

- For custom patch baselines where the **Approved patches include non-security updates** check box 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 dnf upgrade --security --bugfix
```

8. The managed node is rebooted if any updates were installed. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).)

#### RHEL, CentOS Stream, and Rocky Linux

On Red Hat Enterprise Linux, CentOS Stream, and Rocky Linux managed nodes, the patch installation workflow is as follows:

1. If a list of patches is specified using an https URL or an Amazon Simple Storage Service (Amazon S3) path-style URL using the InstallOverrideList parameter for the AWS-RunPatchBaseline or AWS-RunPatchBaselineAssociation documents, the listed patches are installed and steps 2–7 are skipped.
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.

Approval rules, however, are also subject to whether the **Include nonsecurity updates** check box was selected when creating or last updating a patch baseline.

If nonsecurity updates are excluded, 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.

If nonsecurity updates are included, patches from other repositories are considered as well.

4. Apply [ApprovedPatches](#) as specified in the patch baseline. The approved patches are approved for update even if they're 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 won't be applied.
6. If multiple versions of a patch are approved, the latest version is applied.
7. The YUM update API (on RHEL 7) or the DNF update API (on RHEL 8, CentOS Stream, and Rocky Linux) 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:

```
sudo yum update-minimal --sec-severity=critical,important --bugfix -y
```

For RHEL 8, CentOS Stream, and Rocky Linux , the equivalent dnf commands for this workflow are:

```
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** check box 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:

```
sudo yum update --security --bugfix -y
```

For RHEL 8, CentOS Stream, and Rocky Linux, the equivalent dnf command for this workflow is:

```
sudo dnf update --security --bugfix -y
```

8. The managed node is rebooted if any updates were installed. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).)

## SLES

On SUSE Linux Enterprise Server (SLES) managed nodes, the patch installation workflow is as follows:

1. If a list of patches is specified using an https URL or an Amazon Simple Storage Service (Amazon S3) path-style URL using the InstallOverrideList parameter for the AWS-RunPatchBaseline or AWS-RunPatchBaselineAssociation documents, the listed patches are installed and steps 2-7 are skipped.
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.

Approval rules, however, are also subject to whether the **Include nonsecurity updates** check box was selected when creating or last updating a patch baseline.

If nonsecurity updates are excluded, 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.

If nonsecurity updates are included, patches from other repositories are considered as well.

4. Apply [ApprovedPatches](#) as specified in the patch baseline. The approved patches are approved for update even if they're 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 won't be applied.
6. If multiple versions of a patch are approved, the latest version is applied.
7. The Zypper update API is applied to approved patches.
8. The managed node is rebooted if any updates were installed. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the managed

node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).)

## Ubuntu Server

On Ubuntu Server managed nodes, the patch installation workflow is as follows:

1. If a list of patches is specified using an https URL or an Amazon Simple Storage Service (Amazon S3) path-style URL using the `InstallOverrideList` parameter for the `AWS-RunPatchBaseline` or `AWS-RunPatchBaselineAssociation` documents, the listed patches are installed and steps 2–7 are skipped.
2. If an update is available for `python3-apt` (a Python library interface to `libapt`), it is upgraded to the latest version. (This nonsecurity package is upgraded even if you did not select the **Include nonsecurity updates** option.)
3. Apply [GlobalFilters](#) as specified in the patch baseline, keeping only the qualified packages for further processing.
4. Apply [ApprovalRules](#) as specified in the patch baseline. Each approval rule can define a package as approved.

### Note

Because it's not possible to reliably determine the release dates of update packages for Ubuntu Server, the auto-approval options aren't supported for this operating system.

Approval rules, however, are also subject to whether the **Include nonsecurity updates** check box was selected when creating or last updating a patch baseline.

If nonsecurity updates are excluded, 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.

If nonsecurity updates are included, patches from other repositories are considered as well.

Approval rules, however, are also subject to whether the **Include nonsecurity updates** check box was selected when creating or last updating a patch baseline.

### Note

For Ubuntu Server, patch candidate versions are limited to patches included in `trusty-security` (Ubuntu Server 14.04 LTS), `xenial-security` (Ubuntu Server 16.04 LTS), `bionic-security` (Ubuntu Server 18.04 LTS), `focal-security` (Ubuntu Server 20.04 LTS), or `groovy-gorilla` (Ubuntu Server 20.10 STR).

5. Apply [ApprovedPatches](#) as specified in the patch baseline. The approved patches are approved for update even if they're discarded by [GlobalFilters](#) or if no approval rule specified in [ApprovalRules](#) grants it approval.
6. Apply [RejectedPatches](#) as specified in the patch baseline. The rejected patches are removed from the list of approved patches and won't be applied.
7. The APT library is used to upgrade packages.
8. The managed node is rebooted if any updates were installed. (Exception: If the `RebootOption` parameter is set to `NoReboot` in the `AWS-RunPatchBaseline` document, the managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).)

## Windows

When a patching operation is performed on a Windows Server managed node, the node 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 were approved for deployment. This list of

updates is sent to the Windows Update API, which determines which of the updates are applicable to the managed node and installs them as needed. If any updates are installed, the managed node 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 managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).) The summary of the patching operation can be found in the output of the Run Command request. Additional logs can be found on the managed node 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 managed nodes 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 managed node 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 managed nodes to target that WSUS server 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 managed nodes, rules are evaluated on each node to take the configured repos on the instance into consideration. Patch Manager, a capability of AWS Systems Manager, uses the native package manager to drive the installation of patches approved by the patch baseline.

For Linux-based operating system types that report a severity level for patches, Patch Manager uses the severity level reported by the software publisher for the update notice or individual patch. Patch Manager doesn't derive severity levels from third-party sources, such as the [Common Vulnerability Scoring System \(CVSS\)](#), or from metrics released by the [National Vulnerability Database \(NVD\)](#).

### Topics

- [How patch baseline rules work on Amazon Linux and Amazon Linux 2 \(p. 1111\)](#)
- [How patch baseline rules work on CentOS \(p. 1113\)](#)
- [How patch baseline rules work on Debian Server and Raspberry Pi OS \(p. 1115\)](#)
- [How patch baseline rules work on macOS \(p. 1116\)](#)
- [How patch baseline rules work on Oracle Linux \(p. 1117\)](#)
- [How patch baseline rules work on RHEL, CentOS Stream, and Rocky Linux \(p. 1119\)](#)
- [How patch baseline rules work on SUSE Linux Enterprise Server \(p. 1121\)](#)
- [How patch baseline rules work on Ubuntu Server \(p. 1122\)](#)

## 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 managed node, the YUM library accesses the `updateinfo.xml` file for each configured repo.

**Note**

If no `updateinfo.xml` file is found, whether patches are installed depend on settings for **Approved patches include non-security updates** and **Auto-approval**. For example, if non-security updates are permitted, they're installed when the auto-approval time arrives.

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

Attribute	Description
type	Corresponds to the value of the Classification key attribute in the patch baseline's <a href="#">PatchFilter</a> 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 <a href="#">describe-patch-properties</a> or the API operation <a href="#">DescribePatchProperties</a> . You can also view the list in the <b>Approval rules</b> area of the <a href="#">Create patch baseline</a> page or <a href="#">Edit patch baseline</a> page in the Systems Manager console.
severity	Corresponds to the value of the Severity key attribute in the patch baseline's <a href="#">PatchFilter</a> data type. Denotes the severity of the packages included in the update notice. Usually only applicable for <i>Security</i> update notices.  You can view the list of supported values by using the AWS CLI command <a href="#">describe-patch-properties</a> or the API operation <a href="#">DescribePatchProperties</a> . You can also view the list in the <b>Approval rules</b> area of the <a href="#">Create patch baseline</a> page or <a href="#">Edit patch baseline</a> page in the Systems Manager console.
update_id	Denotes the advisory ID, such as <i>ALAS-2017-867</i> . The advisory ID can be used in the <a href="#">ApprovedPatches</a> or <a href="#">RejectedPatches</a> attribute in the patch baseline.
references	Contains additional information about the update notice, such as a CVE ID (format: <i>CVE-2017-1234567</i> ). The CVE ID can be used in the <a href="#">ApprovedPatches</a> or <a href="#">RejectedPatches</a> attribute in the patch baseline.
updated	Corresponds to <a href="#">ApproveAfterDays</a> 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 \(p. 1161\)](#).

3. The product of the managed node 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.

Security option	Patch selection
Pre-defined default patch baselines provided by AWS and custom patch baselines where the <b>Approved patches include non-security updates</b> is <i>not</i> selected	<p>For each update notice in <code>updateinfo.xml</code>, 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.</p> <p>The equivalent yum command for this workflow is:</p> <pre>sudo yum update-minimal --sec-severity=critical,important --bugfix -y</pre>
Custom patch baselines where the <b>Approved patches include non-security updates</b> check box <i>is</i> selected	<p>In addition to applying the security updates that were selected from <code>updateinfo.xml</code>, Patch Manager applies nonsecurity updates that otherwise meet the patch filtering rules.</p> <p>The equivalent yum command for this workflow is:</p> <pre>sudo yum update --security --bugfix -y</pre>

For information about patch compliance status values, see [Understanding patch compliance state values \(p. 1186\)](#).

## How patch baseline rules work on CentOS

On CentOS, the patch selection process is as follows:

1. On the managed node, the YUM library (on CentOS 6.x and 7.x versions) or the DNF library (on CentOS 8.x) accesses the `updateinfo.xml` file for each configured repo.

### Note

If there is no `updateinfo.xml` found, whether patches are installed depend on settings for **Approved patches include non-security updates** and **Auto-approval**. For example, if non-security updates are permitted, they're installed when the auto-approval time arrives.

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

Attribute	Description
<code>type</code>	Corresponds to the value of the Classification key attribute in the patch baseline's <a href="#">PatchFilter</a> data type. Denotes the type of package included in the update notice.

Attribute	Description
	You can view the list of supported values by using the AWS CLI command <a href="#">describe-patch-properties</a> or the API operation <a href="#">DescribePatchProperties</a> . You can also view the list in the <b>Approval rules</b> area of the <b>Create patch baseline</b> page or <b>Edit patch baseline</b> page in the Systems Manager console.
severity	Corresponds to the value of the Severity key attribute in the patch baseline's <a href="#">PatchFilter</a> data type. Denotes the severity of the packages included in the update notice. Usually only applicable for <b>Security</b> update notices.  You can view the list of supported values by using the AWS CLI command <a href="#">describe-patch-properties</a> or the API operation <a href="#">DescribePatchProperties</a> . You can also view the list in the <b>Approval rules</b> area of the <b>Create patch baseline</b> page or <b>Edit patch baseline</b> page in the Systems Manager console.
update_id	Denotes the advisory ID, such as <a href="#">CVE-2019-17055</a> . The advisory ID can be used in the <a href="#">ApprovedPatches</a> or <a href="#">RejectedPatches</a> attribute in the patch baseline.
references	Contains additional information about the update notice, such as a CVE ID (format: <a href="#">CVE-2019-17055</a> ) or a Bugzilla ID (format: <a href="#">1463241</a> ). The CVE ID and Bugzilla ID can be used in the <a href="#">ApprovedPatches</a> or <a href="#">RejectedPatches</a> attribute in the patch baseline.
updated	Corresponds to <a href="#">ApproveAfterDays</a> 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 \(p. 1161\)](#).

3. The product of the managed node 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.

Security option	Patch selection
Pre-defined default patch baselines provided by AWS and custom patch baselines where	For each update notice in <code>updateinfo.xml</code> , the patch baseline is used as a filter, allowing

Security option	Patch selection
<p>the <b>Approved patches include non-security updates</b> is <i>not</i> selected</p>	<p>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.</p> <p>For CentOS 6 and 7, the equivalent yum command for this workflow is:</p> <pre data-bbox="926 502 1437 559">sudo yum update-minimal --sec-severity=critical,important --bugfix -y</pre> <p>For CentOS 8, the equivalent dnf commands for this workflow are:</p> <pre data-bbox="926 692 1328 792">sudo dnf update-minimal --sec-severity=Critical --bugfix -y \ sudo dnf update-minimal --sec-severity=Important --bugfix -y</pre>
<p>Custom patch baselines where the <b>Approved patches include non-security updates</b> check box <i>is</i> selected</p>	<p>In addition to applying the security updates that were selected from <code>updateinfo.xml</code>, Patch Manager applies nonsecurity updates that otherwise meet the patch filtering rules.</p> <p>For CentOS 6 and 7, the equivalent yum command for this workflow is:</p> <pre data-bbox="926 1072 1416 1108">sudo yum update --security --bugfix -y</pre> <p>For CentOS 8, the equivalent dnf command for this workflow is:</p> <pre data-bbox="926 1241 1416 1277">sudo dnf update --security --bugfix -y</pre>

For information about patch compliance status values, see [Understanding patch compliance state values \(p. 1186\)](#).

## How patch baseline rules work on Debian Server and Raspberry Pi OS

On Debian Server and Raspberry Pi OS (formerly Raspbian), the patch baseline service offers filtering on the *Priority* and *Section* fields. These fields are typically present for all Debian Server and Raspberry Pi OS packages. To determine whether a patch is selected by the patch baseline, Patch Manager does the following:

1. On Debian Server and Raspberry Pi OS systems, the equivalent of `sudo apt-get update` is run to refresh the list of available packages. Repos aren't configured and the data is pulled from repos configured in a `sources.list`.
2. If an update is available for `python3-apt` (a Python library interface to `libapt`), it is upgraded to the latest version. (This nonsecurity package is upgraded even if you did not select the **Include nonsecurity updates** option.)

### Important

On Debian Server 8 only: Because Debian Server 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 managed node, 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 Server 8.\* distributions.
- c. The `apt-get` operation is run at this point to ensure that the latest version of `python3-apt` is installed before the patching process begins.
- d. After the installation process is complete, 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.

3. Next, the [GlobalFilters](#), [ApprovalRules](#), [ApprovedPatches](#) and [RejectedPatches](#) lists are applied.

### Note

Because it isn't possible to reliably determine the release dates of update packages for Debian Server, the auto-approval options aren't supported for this operating system.

Approval rules, however, are also subject to whether the **Include nonsecurity updates** check box was selected when creating or last updating a patch baseline.

If nonsecurity updates are excluded, 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. In this case, for Debian Server, patch candidate versions are limited to patches included in the following repos:

These repos are named as follows:

- Debian Server 8: `debian-security jessie`
- Debian Server and Raspberry Pi OS 9: `debian-security stretch`
- Debian Server and Raspberry Pi OS 10: `debian-security buster`

If nonsecurity updates are included, patches from other repositories are considered as well.

### 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 \(p. 1161\)](#).

To view the contents of the *Priority* and *Section* fields, run the following `aptitude` command:

### Note

You might need to first install Aptitude on Debian Server systems.

```
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 [Understanding patch compliance state values \(p. 1186\)](#).

## How patch baseline rules work on macOS

On macOS, the patch selection process is as follows:

1. On the managed node, Patch Manager accesses the parsed contents of the `InstallHistory.plist` file and identifies package names and versions.

For details about the parsing process, see the **macOS** section in [How patches are installed \(p. 1103\)](#).

2. The product of the managed node is determined by SSM Agent. This attribute corresponds to the value of the Product key attribute in the patch baseline's `PatchFilter` data type.
3. Packages are selected for the update according to the following guidelines.

Security option	Patch selection
Pre-defined default patch baselines provided by AWS and custom patch baselines where the <b>Approved patches include non-security updates</b> is <i>not</i> selected	For each available package update, 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.
Custom patch baselines where the <b>Approved patches include non-security updates</b> is selected	In addition to applying the security updates that were identified by using <code>InstallHistory.plist</code> , Patch Manager applies nonsecurity updates that otherwise meet the patch filtering rules.

For information about patch compliance status values, see [Understanding patch compliance state values \(p. 1186\)](#).

## How patch baseline rules work on Oracle Linux

On Oracle Linux, the patch selection process is as follows:

1. On the managed node, the YUM library accesses the `updateinfo.xml` file for each configured repo.

### Note

The `updateinfo.xml` file might not be available if the repo isn't one managed by Oracle. If there is no `updateinfo.xml` found, whether patches are installed depend on settings for **Approved patches include non-security updates** and **Auto-approval**. For example, if non-security updates are permitted, they're installed when the auto-approval time arrives.

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

Attribute	Description
<code>type</code>	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 operation <code>DescribePatchProperties</code> . You can also view the list in the <b>Approval rules</b> area of the <b>Create patch baseline</b> page or <b>Edit patch baseline</b> page in the Systems Manager console.

Attribute	Description
severity	Corresponds to the value of the Severity key attribute in the patch baseline's <a href="#">PatchFilter</a> data type. Denotes the severity of the packages included in the update notice. Usually only applicable for <b>Security</b> update notices.  You can view the list of supported values by using the AWS CLI command <a href="#">describe-patch-properties</a> or the API operation <a href="#">DescribePatchProperties</a> . You can also view the list in the <b>Approval rules</b> area of the <a href="#">Create patch baseline</a> page or <a href="#">Edit patch baseline</a> page in the Systems Manager console.
update_id	Denotes the advisory ID, such as <a href="#">CVE-2019-17055</a> . The advisory ID can be used in the <a href="#">ApprovedPatches</a> or <a href="#">RejectedPatches</a> attribute in the patch baseline.
references	Contains additional information about the update notice, such as a CVE ID (format: <a href="#">CVE-2019-17055</a> ) or a Bugzilla ID (format: <a href="#">1463241</a> ). The CVE ID and Bugzilla ID can be used in the <a href="#">ApprovedPatches</a> or <a href="#">RejectedPatches</a> attribute in the patch baseline.
updated	Corresponds to <a href="#">ApproveAfterDays</a> 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 \(p. 1161\)](#).

3. The product of the managed node 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.

Security option	Patch selection
Pre-defined default patch baselines provided by AWS and custom patch baselines where the <b>Approved patches include non-security updates</b> is <i>not</i> selected	<p>For each update notice in <code>updateinfo.xml</code>, 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.</p> <p>For version 7 managed nodes, the equivalent yum command for this workflow is:</p>

Security option	Patch selection
	<pre data-bbox="923 287 1432 340">sudo yum update-minimal --sec-severity=important,moderate --bugfix -y</pre> <p data-bbox="923 375 1480 428">For version 8 managed nodes, the equivalent dnf command for this workflow is:</p> <pre data-bbox="923 470 1470 523">sudo dnf upgrade-minimal --security --sec-severity Moderate --sec-severity Important</pre>
Custom patch baselines where the <b>Approved patches include non-security updates</b> is selected	<p data-bbox="923 561 1470 677">In addition to applying the security updates that were selected from <code>updateinfo.xml</code>, Patch Manager applies nonsecurity updates that otherwise meet the patch filtering rules.</p> <p data-bbox="923 703 1437 756">For version 7 managed nodes, the equivalent yum command for this workflow is:</p> <pre data-bbox="923 804 1380 836">sudo yum update --security --bugfix</pre> <p data-bbox="923 872 1480 925">For version 8 managed nodes, the equivalent dnf command for this workflow is:</p> <pre data-bbox="923 967 1388 998">sudo dnf upgrade --security --bugfix</pre>

For information about patch compliance status values, see [Understanding patch compliance state values \(p. 1186\)](#).

## How patch baseline rules work on RHEL, CentOS Stream, and Rocky Linux

On Red Hat Enterprise Linux (RHEL), CentOS Stream, and Rocky Linux, the patch selection process is as follows:

1. On the managed node, the YUM library (RHEL 7) or the DNF library (RHEL 8, CentOS Stream, and Rocky Linux) accesses the `updateinfo.xml` file for each configured repo.

### Note

The `updateinfo.xml` file might not be available if the repo isn't 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

Attribute	Description
type	<p data-bbox="923 1681 1470 1797">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.</p> <p data-bbox="923 1822 1396 1888">You can view the list of supported values by using the AWS CLI command <code>describe-</code></p>

Attribute	Description
	<b>patch-properties</b> or the API operation <a href="#">DescribePatchProperties</a> . You can also view the list in the <b>Approval rules</b> area of the <a href="#">Create patch baseline</a> page or <a href="#">Edit patch baseline</a> page in the Systems Manager console.
severity	Corresponds to the value of the Severity key attribute in the patch baseline's <a href="#">PatchFilter</a> data type. Denotes the severity of the packages included in the update notice. Usually only applicable for <b>Security</b> update notices.  You can view the list of supported values by using the AWS CLI command <a href="#">describe-patch-properties</a> or the API operation <a href="#">DescribePatchProperties</a> . You can also view the list in the <b>Approval rules</b> area of the <a href="#">Create patch baseline</a> page or <a href="#">Edit patch baseline</a> page in the Systems Manager console.
update_id	Denotes the advisory ID, such as <i>RHSA-2017:0864</i> . The advisory ID can be used in the <a href="#">ApprovedPatches</a> or <a href="#">RejectedPatches</a> attribute in the patch baseline.
references	Contains additional information about the update notice, such as a CVE ID (format: <i>CVE-2017-1000371</i> ) or a Bugzilla ID (format: <i>1463241</i> ). The CVE ID and Bugzilla ID can be used in the <a href="#">ApprovedPatches</a> or <a href="#">RejectedPatches</a> attribute in the patch baseline.
updated	Corresponds to <a href="#">ApproveAfterDays</a> 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 \(p. 1161\)](#).

3. The product of the managed node 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.

Security option	Patch selection
Pre-defined default patch baselines provided by AWS and custom patch baselines where the <b>Approved patches include non-security updates</b> check box is <i>not</i> selected	For each update notice in <code>updateinfo.xml</code> , 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

Security option	Patch selection
	<p>applying the patch baseline definition, the latest version is used.</p> <p>For RHEL 7, the equivalent yum command for this workflow is:</p> <pre data-bbox="915 439 1488 508">sudo yum update-minimal --sec-severity=critical,important --bugfix -y</pre> <p>For RHEL 8, CentOS Stream, and Rocky Linux, the equivalent dnf commands for this workflow are:</p> <pre data-bbox="915 671 1488 762">sudo dnf update-minimal --sec-severity=Critical --bugfix -y \ sudo dnf update-minimal --sec-severity=Important --bugfix -y</pre>
Custom patch baselines where the <b>Approved patches include non-security updates</b> check box is selected	<p>In addition to applying the security updates that were selected from <code>updateinfo.xml</code>, Patch Manager applies nonsecurity updates that otherwise meet the patch filtering rules.</p> <p>For RHEL 7, the equivalent yum command for this workflow is:</p> <pre data-bbox="915 1051 1488 1079">sudo yum update --security --bugfix</pre> <p>For RHEL 8, CentOS Stream, and Rocky Linux, the equivalent dnf command for this workflow is:</p> <pre data-bbox="915 1241 1488 1269">sudo dnf update --security --bugfix</pre>

For information about patch compliance status values, see [Understanding patch compliance state values \(p. 1186\)](#).

## 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 operation [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 patches.

You can view the list of supported values by using the AWS CLI command [describe-patch-properties](#) or the API operation [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 managed node 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 \(p. 1161\)](#).

## 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 aren't configured and the data is pulled from repos configured in a sources list.
2. If an update is available for `python3-apt` (a Python library interface to `libapt`), it is upgraded to the latest version. (This nonsecurity package is upgraded even if you did not select the **Include nonsecurity updates** option.)
3. 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 aren't supported for this operating system.

Approval rules, however, are also subject to whether the **Include nonsecurity updates** check box was selected when creating or last updating a patch baseline.

If nonsecurity updates are excluded, 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. In this case, for Ubuntu Server, patch candidate versions are limited to patches included in the following repos:

- Ubuntu Server 14.04 LTS: `trusty-security`
- Ubuntu Server 16.04 LTS: `xenial-security`
- Ubuntu Server 18.04 LTS: `bionic-security`
- Ubuntu Server 20.04 LTS: `focal-security`
- Ubuntu Server 20.10 STR: `groovy-gorilla`

If nonsecurity updates are included, patches from other repositories are considered as well.

**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 \(p. 1161\)](#).

To view the contents of the *Priority* and *Section* fields, run the following `aptitude` command:

**Note**

You might need to first install Aptitude on Ubuntu Server 16 systems.

```
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 [Understanding patch compliance state values \(p. 1186\)](#).

## Key differences between Linux and Windows patching

This topic describes important differences between Linux and Windows patching in Patch Manager, a capability of AWS Systems Manager.

### Note

To patch Linux managed nodes, your nodes 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. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

### 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 node. Systems Manager must evaluate patching on each node because the service retrieves the list of known patches and updates from the repositories that are configured on the managed node.

#### Windows

Patch Manager uses different processes on Windows managed nodes and Linux managed nodes 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 doesn't report details about patches in the *Not Applicable* state. A *Not Applicable* patch is, for example, a patch for Apache software when the instance doesn't have Apache installed. Systems Manager does report the number of *Not Applicable* patches in the summary, but if you call the [DescribeInstancePatches](#) API for a managed node, the returned data doesn't include patches with a state of *Not Applicable*. This behavior is different from Windows.

### Difference 3: SSM document support

The [AWS-ApplyPatchBaseline](#) Systems Manager document (SSM document) doesn't support Linux managed nodes. For applying patch baselines to Linux, macOS, and Windows Server managed nodes, the recommended SSM document is [AWS-RunPatchBaseline](#). For more information, see [About SSM documents for patching managed nodes \(p. 1124\)](#) and [About the AWS-RunPatchBaseline SSM document \(p. 1128\)](#).

#### Difference 4: Application patches

The primary focus of Patch Manager is applying patches to operating systems. However, you can also use Patch Manager to apply patches to some applications on your managed nodes.

##### Linux

On Linux operating systems, Patch Manager uses the configured repositories for updates, and doesn't 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 \(Linux\) \(p. 1101\)](#).

##### Windows

On Windows Server managed nodes, you can apply approval rules, as well as *Approved* and *Rejected* patch exceptions, for applications released by Microsoft, such as Microsoft Word 2016 and Microsoft Exchange Server 2016. For more information, see [Working with custom patch baselines \(console\) \(p. 1194\)](#).

## About SSM documents for patching managed nodes

This topic describes the nine Systems Manager documents (SSM documents) available to help you keep your managed nodes patched with the latest security-related updates.

We recommend using just five of these documents in your patching operations. Together, these five SSM documents provide you with a full range of patching options using AWS Systems Manager. Four of these documents were released later than the four legacy SSM documents they replace and represent expansions or consolidations of functionality.

The five recommended SSM documents include:

- [AWS-ConfigureWindowsUpdate](#)
- [AWS-InstallWindowsUpdates](#)
- [AWS-RunPatchBaseline](#)
- [AWS-RunPatchBaselineAssociation](#)
- [AWS-RunPatchBaselineWithHooks](#)

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 managed nodes \(p. 1125\)](#)
- [Legacy SSM documents for patching managed nodes \(p. 1127\)](#)
- [About the AWS-RunPatchBaseline SSM document \(p. 1128\)](#)
- [About the AWS-RunPatchBaselineAssociation SSM document \(p. 1137\)](#)

- [About the AWS-RunPatchBaselineWithHooks SSM document \(p. 1146\)](#)
- [Sample scenario for using the InstallOverrideList parameter in AWS-RunPatchBaseline or AWS-RunPatchBaselineAssociation \(p. 1153\)](#)
- [Using the BaselineOverride parameter \(p. 1154\)](#)

## SSM documents recommended for patching managed nodes

The following five SSM documents are recommended for use in your managed node patching operations.

### Recommended SSM documents

- [AWS-ConfigureWindowsUpdate \(p. 1125\)](#)
- [AWS-InstallWindowsUpdates \(p. 1125\)](#)
- [AWS-RunPatchBaseline \(p. 1126\)](#)
- [AWS-RunPatchBaselineAssociation \(p. 1126\)](#)
- [AWS-RunPatchBaselineWithHooks \(p. 1126\)](#)

### [AWS-ConfigureWindowsUpdate](#)

Supports configuring basic Windows Update functions and using them to install updates automatically (or to turn off automatic updates). Available in all AWS Regions.

This SSM document prompts Windows Update to download and install the specified updates and reboot managed nodes as needed. Use this document with State Manager, a capability of AWS Systems Manager, to ensure Windows Update maintains its configuration. You can also run it manually using Run Command, a capability of AWS Systems Manager, to change the Windows Update configuration.

The available parameters in this document support specifying a category of updates to install (or whether to turn off 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 managed node. 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 managed nodes \(p. 1127\)](#).

## AWS-RunPatchBaseline

Installs patches on your managed nodes or scans nodes to determine whether any qualified patches are missing. Available in all AWS Regions.

AWS-RunPatchBaseline allows you to control patch approvals using the patch baseline specified as the "default" for an operating system type. 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 managed nodes, such as which nodes are missing patches and what those patches are. When you use AWS-RunPatchBaseline, patch compliance information is recorded using the PutInventory API command. For Linux operating systems, compliance information is provided for patches from both the default source repository configured on a managed node 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 \(Linux\) \(p. 1101\)](#). For more information about the Systems Manager Compliance tools, see [AWS Systems Manager Compliance \(p. 836\)](#).

**Replaces legacy documents:**

- AWS-ApplyPatchBaseline

The legacy document AWS-ApplyPatchBaseline applies only to Windows Server managed nodes, and doesn't 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 AWS-RunPatchBaseline SSM document \(p. 1128\)](#).

## AWS-RunPatchBaselineAssociation

Installs patches on your instances or scans instances to determine whether any qualified patches are missing. Available in all commercial AWS Regions.

AWS-RunPatchBaselineAssociation differs from AWS-RunPatchBaseline in a few important ways:

- AWS-RunPatchBaselineAssociation is intended for use primarily with associations created using Quick Setup, a capability of AWS Systems Manager. (When using the Quick Setup Host Management configuration type, if you choose the option **Scan instances for missing patches daily**, the system uses AWS-RunPatchBaselineAssociation for the operation.)

In most cases, however, when setting up your own patching operations, you should choose [AWS-RunPatchBaseline \(p. 1128\)](#) or [AWS-RunPatchBaselineWithHooks \(p. 1146\)](#) instead of AWS-RunPatchBaselineAssociation.

For more information, see the following topics:

- [AWS Systems Manager Quick Setup \(p. 145\)](#)
- [About the AWS-RunPatchBaselineAssociation SSM document \(p. 1137\)](#)
- AWS-RunPatchBaselineAssociation supports the use of tags to identify which patch baseline to use with a set of targets when it runs.
- For patching operations that use AWS-RunPatchBaselineAssociation, patch compliance data is compiled in terms of a specific State Manager association. The patch compliance data collected when AWS-RunPatchBaselineAssociation runs is recorded using the PutComplianceItems API command instead of the PutInventory command. This prevents compliance data that isn't associated with this particular association from being overwritten.

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 \(Linux\) \(p. 1101\)](#). For more information about the Systems Manager Compliance tools, see [AWS Systems Manager Compliance \(p. 836\)](#).

**Replaces legacy documents:**

- **None**

For more information about the `AWS-RunPatchBaselineAssociation` SSM document, see [About the AWS-RunPatchBaselineAssociation SSM document \(p. 1137\)](#).

### [AWS-RunPatchBaselineWithHooks](#)

Installs patches on your managed nodes or scans nodes to determine whether any qualified patches are missing, with optional hooks you can use to run SSM documents at three points during the patching cycle. Available in all commercial AWS Regions.

`AWS-RunPatchBaselineWithHooks` differs from `AWS-RunPatchBaseline` in its `Install` operation.

`AWS-RunPatchBaselineWithHooks` supports lifecycle hooks that run at designated points during managed node patching. Because patch installations sometimes require managed nodes to reboot, the patching operation is divided into two events, for a total of three hooks that support custom functionality. The first hook is before the `Install` with `NoReboot` operation. The second hook is after the `Install` with `NoReboot` operation. The third hook is available after the reboot of the node.

**Replaces legacy documents:**

- **None**

For more information about the `AWS-RunPatchBaselineWithHooks` SSM document, see [About the AWS-RunPatchBaselineWithHooks SSM document \(p. 1146\)](#).

## [Legacy SSM documents for patching managed nodes](#)

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 don't recommend their use. Instead, use the documents described in [SSM documents recommended for patching managed nodes \(p. 1125\)](#).

### **Legacy SSM Documents**

- [AWS-ApplyPatchBaseline \(p. 1127\)](#)
- [AWS-FindWindowsUpdates \(p. 1128\)](#)
- [AWS-InstallMissingWindowsUpdates \(p. 1128\)](#)
- [AWS-InstallSpecificWindowsUpdates \(p. 1128\)](#)

### [AWS-ApplyPatchBaseline](#)

Supports only Windows Server managed nodes, but doesn't 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 managed nodes 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 `AWS-RunPatchBaseline` SSM document

AWS Systems Manager supports `AWS-RunPatchBaseline`, a Systems Manager document (SSM document) for Patch Manager, a capability of AWS Systems Manager. This SSM document performs patching operations on managed nodes for both security related and other types of updates. When the document is run, it uses the patch baseline specified as the "default" for an operating system type if no patch group is specified. Otherwise, it uses the patch baseline that is associated with the patch group. For information about patch groups, see [About patch groups \(p. 1163\)](#).

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 applications released by Microsoft.)

This document supports Linux, macOS, and Windows Server managed nodes. 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 managed nodes only. We encourage you to use `AWS-RunPatchBaseline` instead because it supports patching on Linux, macOS, and Windows Server managed nodes. Version 2.0.834.0 or later of SSM Agent is required in order to use the `AWS-RunPatchBaseline` document.

**Windows**

On Windows Server managed nodes, the `AWS-RunPatchBaseline` document downloads and invokes a PowerShell module, which in turn downloads a snapshot of the patch baseline that applies to the managed node. This patch baseline snapshot contains a list of approved patches that is compiled by querying the patch baseline against a Windows Server Update Services (WSUS) server. This list is passed to the Windows Update API, which controls downloading and installing the approved patches as appropriate.

**Linux**

On Linux managed nodes, the `AWS-RunPatchBaseline` document invokes a Python module, which in turn downloads a snapshot of the patch baseline that applies to the managed node. This patch baseline snapshot uses the defined rules and lists of approved and blocked patches to drive the appropriate package manager for each node type:

- Amazon Linux, Amazon Linux 2, CentOS, Oracle Linux, and RHEL 7 managed nodes use YUM. For YUM operations, Patch Manager requires Python 2.6 or later.
- RHEL 8 managed nodes 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 Server, Raspberry Pi OS, and Ubuntu Server instances use APT. For APT operations, Patch Manager requires Python 3.
- SUSE Linux Enterprise Server managed nodes use Zypper. For Zypper operations, Patch Manager requires Python 2.6 or later.

**macOS**

On macOS managed nodes, the `AWS-RunPatchBaseline` document invokes a Python module, which in turn downloads a snapshot of the patch baseline that applies to the managed node. Next, a Python subprocess invokes the AWS Command Line Interface (AWS CLI) on the node to retrieve the installation and update information for the specified package managers and to drive the appropriate package manager for each update package.

Each snapshot is specific to an AWS account, patch group, operating system, and snapshot ID. The snapshot is delivered through a presigned Amazon Simple Storage Service (Amazon S3) URL, which expires 24 hours after the snapshot is created. After the URL expires, however, if you want to apply the same snapshot content to other managed nodes, you can generate a new presigned Amazon S3 URL up to three days after the snapshot was created. To do this, use the [get-deployable-patch-snapshot-for-instance](#) command.

After all approved and applicable updates have been installed, with reboots performed as necessary, patch compliance information is generated on a managed node and reported back to Patch Manager.

**Note**

If the `RebootOption` parameter is set to `NoReboot` in the `AWS-RunPatchBaseline` document, the managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).

For information about viewing patch compliance data, see [About patch compliance \(p. 840\)](#).

## AWS-RunPatchBaseline parameters

AWS-RunPatchBaseline supports five parameters. The `Operation` parameter is required. The `InstallOverrideList`, `BaselineOverride`, 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. Patch Manager can supply the custom value automatically when the document is run as part of a maintenance window operation.

### Parameters

- Parameter name: [Operation \(p. 1130\)](#)
- Parameter name: [Snapshot ID \(p. 1130\)](#)
- Parameter name: [InstallOverrideList \(p. 1132\)](#)
- Parameter name: [RebootOption \(p. 1136\)](#)
- Parameter name: [BaselineOverride \(p. 1137\)](#)

#### Parameter name: Operation

**Usage:** Required.

**Options:** Scan | Install.

Scan

When you choose the `Scan` option, AWS-RunPatchBaseline determines the patch compliance state of the managed node and reports this information back to Patch Manager. `Scan` doesn't prompt updates to be installed or managed nodes to be rebooted. Instead, the operation identifies where updates are missing that are approved and applicable to the node.

Install

When you choose the `Install` option, AWS-RunPatchBaseline attempts to install the approved and applicable updates that are missing from the managed node. Patch compliance information generated as part of an `Install` operation doesn't list any missing updates, but might report updates that are in a failed state if the installation of the update didn't succeed for any reason. Whenever an update is installed on a managed node, the node 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 managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).)

#### Note

If a patch specified by the baseline rules is installed *before* Patch Manager updates the managed node, 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 managed nodes 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're running AWS-RunPatchBaseline in a maintenance window, as described in the following table.

### AWS-RunPatchBaseline best practices

Mode	Best practice	Details
Running AWS-RunPatchBaseline inside a maintenance window	Don't supply a Snapshot ID. Patch Manager will supply it for you.	<p>If you use a maintenance window to run AWS-RunPatchBaseline, you shouldn't 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 AWS-RunPatchBaseline in that maintenance window.</p> <p>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 three days. After that, a new snapshot will be generated even if you specify the same ID after the snapshot expires.</p>
Running AWS-RunPatchBaseline outside of a maintenance window	Generate and specify a custom GUID value for the Snapshot ID. <sup>1</sup>	<p>When you aren't using a maintenance window to run AWS-RunPatchBaseline, we recommend that you generate and specify a unique Snapshot ID for each patch baseline, particularly if you're running the AWS-RunPatchBaseline document on multiple managed nodes in the same operation. If you don't specify an ID in this scenario, Systems Manager generates a different Snapshot ID for each managed node the command is sent to. This might result in varying sets of patches being specified among the managed nodes.</p> <p>For instance, say that you're running the AWS-RunPatchBaseline document directly through Run Command, a capability of AWS Systems Manager, and targeting a group of 50 managed nodes. Specifying a custom Snapshot ID results in the generation of a single baseline snapshot that is used to evaluate and patch all</p>

Mode	Best practice	Details
		the nodes, ensuring that they end up in a consistent state.

<sup>1</sup> 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.

Using `InstallOverrideList`, you specify an https URL or an 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 managed nodes.

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` or `AWS-RunPatchBaselineAssociation` \(p. 1153\)](#).

#### Valid URL formats

##### Note

If your file is stored in a publicly available bucket, you can specify either an https URL format or an Amazon S3 path-style URL. If your file is stored in a private bucket, you must specify an Amazon S3 path-style URL.

- **https URL format:**

```
https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/my-windows-override-list.yaml
```

- **Amazon S3 path-style URL:**

```
s3://DOC-EXAMPLE-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 managed node. The general format, however, is as follows:

```
patches:
 -
 id: '{patch-d}'
 title: '{patch-title}'
 {additional-fields}:{values}
```

Although you can provide additional fields in your YAML file, they're 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 'dhcp\*1.\*'.

### **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 managed node, 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."

## 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.

## macOS

### **id**

The **id** field is required. The value for the **id** field can be supplied using either a {package-name} . {package-version} format or a {package\_name} format.

## Sample patch lists

- **Amazon Linux**

```
patches:
 -
 id: 'kernel.x86_64'
 -
 id: 'bind*.x86_64'
 title: '32:9.8.2-0.62.rc1.57.amzn1'
 -
 id: 'glibc*'
 -
 id: 'dhclient*'
 title: '*12:4.1.1-53.P1.28.amzn1'
 -
 id: 'dhcp*'
 title: '*10:3.1.1-50.P1.26.amzn1'
```

- **CentOS**

```
patches:
 -
 id: 'kernel.x86_64'
 -
 id: 'bind*.x86_64'
 title: '32:9.8.2-0.62.rc1.57.amzn1'
 -
 id: 'glibc*'
 -
 id: 'dhclient*'
 title: '*12:4.1.1-53.P1.28.amzn1'
 -
 id: 'dhcp*'
 title: '*10:3.1.1-50.P1.26.amzn1'
```

- **Debian 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: '*1.2.27'
 -
 id: 'apt-utils.amd64'
 title: '*1.2.25'
```

- **macOS**

```
patches:
 -
 id: 'XProtectPlistConfigData'
 -
 id: 'MRTConfigData.1.61'
 -
 id: 'Command Line Tools for Xcode.11.5'
 -
```

```
 id: 'Gatekeeper Configuration Data'
```

- **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-*.x86_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: '*.el7_5.1'
 -
 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: '0:4.3.3-9.1'
 -
 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: '*1.2.27'
-
 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)'
-
 id: 'KB4284833'
-
 id: 'KB4284835'
 title: '2018-06 Cumulative Update for Windows Server 2016 (1803) for x64-based
Systems (KB4284835)'
-
 id: 'KB4284880'
-
 id: 'KB4338814'
```

**Parameter name:** `RebootOption`

**Usage:** Optional.

**Options:** `RebootIfNeeded` | `NoReboot`

**Default:** `RebootIfNeeded`

**Warning**

The default option is `RebootIfNeeded`. Be sure to select the correct option for your use case. For example, if your managed nodes must reboot immediately to complete a configuration process, choose `RebootIfNeeded`. Or, if you need to maintain managed node availability until a scheduled reboot time, choose `NoReboot`.

`RebootIfNeeded`

When you choose the `RebootIfNeeded` option, the managed node is rebooted in either of the following cases:

- Patch Manager installed one or more patches.

Patch Manager doesn't evaluate whether a reboot is *required* by the patch. The system is rebooted even if the patch doesn't require a reboot.

- Patch Manager detects one or more 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.

**Note**

Patches installed outside of Patch Manager are never given a status of `INSTALLED_PENDING_REBOOT`.

Rebooting managed nodes in these two cases ensures that updated packages are flushed from memory and keeps patching and rebooting behavior consistent across all operating systems.

#### NoReboot

When you choose the `NoReboot` option, Patch Manager doesn't reboot a managed node even if it installed patches during the `Install` operation. This option is useful if you know that your managed nodes don't require rebooting after patches are applied, or you have applications or processes running on a node that shouldn't be disrupted by a patching operation reboot. It's also useful when you want more control over the timing of managed node reboots, such as by using a maintenance window.

#### Note

If you choose the `NoReboot` option and a patch is installed, the patch is assigned a status of `InstalledPendingReboot`. The managed node itself, however, is marked as `Non-Compliant`. After a reboot occurs and a `Scan` operation is run, the managed node status is updated to `Compliant`.

**Patch installation tracking file:** To track patch installation, especially patches that were installed since the last system reboot, Systems Manager maintains a file on the managed node.

#### Important

Don't delete or modify the tracking file. If this file is deleted or corrupted, the patch compliance report for the managed node is inaccurate. If this happens, reboot the node and run a patch `Scan` operation to restore the file.

This tracking file is stored in the following locations on your managed nodes:

- Linux operating systems:
  - `/var/log/amazon/ssm/patch-configuration/patch-states-configuration.json`
  - `/var/log/amazon/ssm/patch-configuration/patch-inventory-from-last-operation.json`
- Windows Server operating system:
  - `C:\ProgramData\Amazon\PatchBaselineOperations\State\PatchStatesConfiguration.json`
  - `C:\ProgramData\Amazon\PatchBaselineOperations\State\PatchInventoryFromLastOperation.json`

#### Parameter name: `BaselineOverride`

**Usage:** Optional.

You can define patching preferences at runtime using the `BaselineOverride` parameter. This baseline override is maintained as a JSON object in an S3 bucket. It ensures patching operations use the provided baselines that match the host operating system instead of applying the rules from the default patch baseline.

For more information about how to use the `BaselineOverride` parameter, see [Using the `BaselineOverride` parameter \(p. 1154\)](#).

## About the AWS-RunPatchBaselineAssociation SSM document

Like the `AWS-RunPatchBaseline` document, `AWS-RunPatchBaselineAssociation` performs patching operations on instances for both security related and other types of updates. You can also use the document `AWS-RunPatchBaselineAssociation` to apply patches for both operating systems and applications. (On Windows Server, application support is limited to updates for applications released by Microsoft.)

**Note**

`AWS-RunPatchBaselineAssociation` isn't supported for on-premises servers and virtual machines (VMs) in a hybrid environment.

This document supports Amazon Elastic Compute Cloud (Amazon EC2) instances for Linux, macOS, and Windows Server. The document will perform the appropriate actions for each platform, invoking a Python module on Linux and macOS instances, and a PowerShell module on Windows instances.

`AWS-RunPatchBaselineAssociation`, however, differs from `AWS-RunPatchBaseline` in the following ways:

- `AWS-RunPatchBaselineAssociation` is intended for use primarily with associations created using [Quick Setup \(p. 145\)](#), a capability of AWS Systems Manager. (When using the Quick Setup Host Management configuration type, if you choose the option **Scan instances for missing patches daily**, the system uses `AWS-RunPatchBaselineAssociation` for the operation.)

In most cases, however, when setting up your own patching operations, you should choose [AWS-RunPatchBaseline \(p. 1128\)](#) or [AWS-RunPatchBaselineWithHooks \(p. 1146\)](#) instead of `AWS-RunPatchBaselineAssociation`.

- When you use the `AWS-RunPatchBaselineAssociation` document, you can specify a tag key pair in the document's `BaselineTags` parameter field. If a custom patch baseline in your AWS account shares these tags, Patch Manager, a capability of AWS Systems Manager, uses that tagged baseline when it runs on the target instances instead of the currently specified "default" patch baseline for the operating system type.

**Important**

If you choose to use `AWS-RunPatchBaselineAssociation` in patching operations other than those set up using Quick Setup, and you want to use its optional `BaselineTags` parameter, you must provide some additional permissions to the [instance profile \(p. 21\)](#) for Amazon Elastic Compute Cloud (Amazon EC2) instances. For more information, see [Parameter name: BaselineTags \(p. 1140\)](#).

Both of the following formats are valid for your `BaselineTags` parameter:

`Key=tag-key,Values=tag-value`

`Key=tag-key,Values=tag-value1,tag-value2,tag-value3`

- When `AWS-RunPatchBaselineAssociation` runs, the patch compliance data it collects is recorded using the `PutComplianceItems` API command instead of the `PutInventory` command, which is used by `AWS-RunPatchBaseline`. This difference means that the patch compliance information that is stored and reported per a specific *association*. Patch compliance data generated outside of this association isn't overwritten.
- The patch compliance information reported after running `AWS-RunPatchBaselineAssociation` indicates whether or not an instance is in compliance. It doesn't include patch-level details, as demonstrated by the output of the following AWS Command Line Interface (AWS CLI) command. The command filters on `Association` as the compliance type:

```
aws ssm list-compliance-items \
--resource-ids "i-02573cafccEXAMPLE" \
--resource-types "ManagedInstance" \
--filters "Key=ComplianceType,Values=Association,Type=EQUAL" \
--region us-east-2
```

The system returns information like the following.

```
{
 "ComplianceItems": [
 {
```

```
 "Status": "NON_COMPLIANT",
 "Severity": "UNSPECIFIED",
 "Title": "MyPatchAssociation",
 "ResourceType": "ManagedInstance",
 "ResourceId": "i-02573cafccEXAMPLE",
 "ComplianceType": "Association",
 "Details": {
 "DocumentName": "AWS-RunPatchBaselineAssociation",
 "PatchBaselineId": "pb-0c10e65780EXAMPLE",
 "DocumentVersion": "1"
 },
 "ExecutionSummary": {
 "ExecutionTime": 1590698771.0
 },
 "Id": "3e5d5694-cd07-40f0-bbea-040e6EXAMPLE"
 }
}
```

If a tag key pair value has been specified as a parameter for the `AWS-RunPatchBaselineAssociation` document, Patch Manager searches for a custom patch baseline that matches the operating system type and has been tagged with that same tag-key pair. This search isn't limited to the current specified default patch baseline or the baseline assigned to a patch group. If no baseline is found with the specified tags, Patch Manager next looks for a patch group, if one was specified in the command that runs `AWS-RunPatchBaselineAssociation`. If no patch group is matched, Patch Manager falls back to the current default patch baseline for the operating system account.

If more than one patch baseline is found with the tags specified in the `AWS-RunPatchBaselineAssociation` document, Patch Manager returns an error message indicating that only one patch baseline can be tagged with that key-value pair in order for the operation to proceed.

**Note**

On Linux instances, the appropriate package manager for each instance type is used to install packages:

- Amazon Linux, Amazon Linux 2, CentOS, Oracle Linux, and RHEL instances use YUM. For YUM operations, Patch Manager requires Python 2.6 or later.
- Debian Server, Raspberry Pi OS, 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 a scan is complete, or 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 the Patch Compliance service.

**Note**

If the `RebootOption` parameter is set to `NoReboot` in the `AWS-RunPatchBaselineAssociation` document, the instance isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1145\)](#).

For information about viewing patch compliance data, see [About patch compliance \(p. 840\)](#).

## [AWS-RunPatchBaselineAssociation parameters](#)

`AWS-RunPatchBaselineAssociation` supports four parameters. The `Operation` and `AssociationId` parameters are required. The `InstallOverrideList`, `RebootOption`, and `BaselineTags` parameters are optional.

### **Parameters**

- Parameter name: [Operation \(p. 1140\)](#)
- Parameter name: [BaselineTags \(p. 1140\)](#)
- Parameter name: [AssociationId \(p. 1141\)](#)
- Parameter name: [InstallOverrideList \(p. 1142\)](#)
- Parameter name: [RebootOption \(p. 1145\)](#)

#### [Parameter name: Operation](#)

**Usage:** Required.

**Options:** Scan | Install.

##### Scan

When you choose the Scan option, `AWS-RunPatchBaselineAssociation` determines the patch compliance state of the instance and reports this information back to Patch Manager. Scan doesn't 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-RunPatchBaselineAssociation` attempts to install the approved and applicable updates that are missing from the instance. Patch compliance information generated as part of an Install operation doesn't list any missing updates, but might report updates that are in a failed state if the installation of the update didn't 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-RunPatchBaselineAssociation` document, the instance isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1145\)](#).)

##### 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: BaselineTags](#)

**Usage:** Optional.

`BaselineTags` is a unique tag key-value pair that you choose and assign to an individual custom patch baseline. You can specify one or more values for this parameter. Both of the following formats are valid:

Key=[tag-key](#),Values=[tag-value](#)

Key=[tag-key](#),Values=[tag-value1](#),[tag-value2](#),[tag-value3](#)

The `BaselineTags` value is 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. When the patching operation runs, Patch Manager checks to see if a patch baseline for the operating system type is tagged with the same key-value pair you specify for `BaselineTags`. If there is a match, this custom patch baseline is used. If there isn't a match, a patch baseline is identified according to any patch group specified for the patching operating. If there is none, the AWS managed predefined patch baseline for that operating system is used.

#### Additional permission requirements

If you use `AWS-RunPatchBaselineAssociation` in patching operations other than those set up using Quick Setup, and you want to use the optional `BaselineTags` parameter, you must add the following permissions to the [instance profile \(p. 21\)](#) for Amazon Elastic Compute Cloud (Amazon EC2) instances.

**Note**

Quick Setup and AWS-RunPatchBaselineAssociation don't support on-premises servers and virtual machines (VMs).

```
{
 "Effect": "Allow",
 "Action": [
 "ssm:DescribePatchBaselines",
 "tag:GetResources"
],
 "Resource": "*"
,
{
 "Effect": "Allow",
 "Action": [
 "ssm:GetPatchBaseline",
 "ssm:DescribeEffectivePatchesForPatchBaseline"
],
 "Resource": "patch-baseline-arn"
}
```

Replace *patch-baseline-arn* with the Amazon Resource Name (ARN) of the patch baseline to which you want to provide access, in the format arn:aws:ssm:us-east-2:123456789012:patchbaseline/pb-0c10e65780EXAMPLE.

**Parameter name:** AssociationId

**Usage:** Required.

AssociationId is the ID of an existing association in State Manager, a capability of AWS Systems Manager. It's used by Patch Manager to add compliance data to the specified Association. By sending patching results as association compliance data instead of inventory compliance data, existing inventory compliance information for your instances isn't overwritten after a patching operation, nor for other association IDs. If you don't already have an association you want to use, you can create one by running [create-association](#) the command. For example:

Linux & macOS

```
aws ssm create-association \
 --name "AWS-RunPatchBaselineAssociation" \
 --association-name "MyPatchAssociation" \
 --targets
 "Key=instanceids,Values=[i-02573cafefEXAMPLE,i-07782c72faEXAMPLE,i-07782c72faEXAMPLE]" \
 --parameters "Operation=Scan" \
 --schedule-expression "cron(0 */30 * * * ? *)" \
 --sync-compliance "MANUAL" \
 --region us-east-2
```

Windows

```
aws ssm create-association ^
 --name "AWS-RunPatchBaselineAssociation" ^
 --association-name "MyPatchAssociation" ^
 --targets
 "Key=instanceids,Values=[i-02573cafefEXAMPLE,i-07782c72faEXAMPLE,i-07782c72faEXAMPLE]" ^
 --parameters "Operation=Scan" ^
 --schedule-expression "cron(0 */30 * * * ? *)" ^
 --sync-compliance "MANUAL" ^
```

```
--region us-east-2
```

### Parameter name: `InstallOverrideList`

**Usage:** Optional.

Using `InstallOverrideList`, 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.

#### Valid URL formats

##### Note

If your file is stored in a publicly available bucket, you can specify either an https URL format or an Amazon S3 path-style URL. If your file is stored in a private bucket, you must specify an Amazon S3 path-style URL.

- **https URL format example:**

```
https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/my-windows-override-list.yaml
```

- **Amazon S3 path-style URL example:**

```
s3://DOC-EXAMPLE-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:

```
patches:
 -
 id: '{patch-d}'
 title: '{patch-title}'
 {additional-fields}:{values}
```

Although you can provide additional fields in your YAML file, they're 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".

- 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.

- 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 'dhcp\*1.\*'.

#### **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."

### **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**

- Windows

```
patches:
-
 id: 'KB4284819'
 title: '2018-06 Cumulative Update for Windows Server 2016 (1709) for x64-based
Systems (KB4284819)'
-
 id: 'KB4284833'
-
 id: 'KB4284835'
 title: '2018-06 Cumulative Update for Windows Server 2016 (1803) for x64-based
Systems (KB4284835)'
-
 id: 'KB4284880'
-
 id: 'KB4338814'
```

- **APT**

```
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'
```

- **Amazon Linux**

```
patches:
 -
 id: 'kernel.x86_64'
 -
 id: 'bind*.x86_64'
 title: '32:9.8.2-0.62.rc1.57.amzn1'
 -
 id: 'glibc*'
 -
 id: 'dhclient*'
 title: '*12:4.1.1-53.P1.28.amzn1'
 -
 id: 'dhcp*'
 title: '*10:3.1.1-50.P1.26.amzn1'
```

- **Red Hat Enterprise Linux (RHEL)**

```
patches:
 -
 id: 'NetworkManager.x86_64'
 title: '*1:1.10.2-14.el7_5'
 -
 id: 'NetworkManager-*.x86_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: '*.el7_5.1'
 -
 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: '0:4.3.3-9.1'

-
 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: '*1.2.27'
-
 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)'
-
 id: 'KB4284833'
-
 id: 'KB4284835'
 title: '2018-06 Cumulative Update for Windows Server 2016 (1803) for x64-based
Systems (KB4284835)'
-
 id: 'KB4284880'
-
 id: 'KB4338814'
```

**Parameter name:** RebootOption

**Usage:** Optional.

**Options:** RebootIfNeeded | NoReboot

**Default:** RebootIfNeeded

**Warning**

The default option is RebootIfNeeded. Be sure to select the correct option for your use case. For example, if your instances must reboot immediately to complete a configuration process, choose RebootIfNeeded. Or, if you need to maintain instances availability until a scheduled reboot time, choose NoReboot.

### RebootIfNeeded

When you choose the `RebootIfNeeded` option, the instance is rebooted in either of the following cases:

- Patch Manager installed one or more patches.

Patch Manager doesn't evaluate whether a reboot is *required* by the patch. The system is rebooted even if the patch doesn't require a reboot.

- Patch Manager detects one or more 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.

#### Note

Patches installed outside of Patch Manager are never given a status of `INSTALLED_PENDING_REBOOT`.

Rebooting instances in these two cases ensures that updated packages are flushed from memory and keeps patching and rebooting behavior consistent across all operating systems.

### NoReboot

When you choose the `NoReboot` option, Patch Manager doesn't 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 shouldn't be disrupted by a patching operation reboot. It's 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

Don't 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`
  - `/var/log/amazon/ssm/patch-configuration/patch-inventory-from-last-operation.json`
- Windows Server operating system:
  - `C:\ProgramData\Amazon\PatchBaselineOperations\State\PatchStatesConfiguration.json`
  - `C:\ProgramData\Amazon\PatchBaselineOperations\State\PatchInventoryFromLastOperation.json`

## About the AWS-RunPatchBaselineWithHooks SSM document

AWS Systems Manager supports `AWS-RunPatchBaselineWithHooks`, a Systems Manager document (SSM document) for Patch Manager, a capability of AWS Systems Manager. This SSM document performs patching operations on managed nodes for both security related and other types of updates.

`AWS-RunPatchBaselineWithHooks` differs from `AWS-RunPatchBaseline` in the following ways:

- **A wrapper document** – `AWS-RunPatchBaselineWithHooks` is a wrapper for `AWS-RunPatchBaseline` and relies on `AWS-RunPatchBaseline` for some of its operations.
- **The Install operation** – `AWS-RunPatchBaselineWithHooks` supports lifecycle hooks that run at designated points during managed node patching. Because patch installations sometimes require managed nodes to reboot, the patching operation is divided into two events, for a total of three hooks that support custom functionality. The first hook is before the `Install` with `NoReboot` operation. The second hook is after the `Install` with `NoReboot` operation. The third hook is available after the reboot of the managed node.
- **No custom patch list support** – `AWS-RunPatchBaselineWithHooks` doesn't support the `InstallOverrideList` parameter.
- **SSM Agent support** – `AWS-RunPatchBaselineWithHooks` requires that SSM Agent 3.0.502 or later be installed on the managed node to patch.

When the document is run, it uses the patch baseline currently specified as the "default" for an operating system type if no patch group is specified. Otherwise, it uses the patch baselines that is associated with the patch group. For information about patch groups, see [About patch groups \(p. 1163\)](#).

You can use the document `AWS-RunPatchBaselineWithHooks` to apply patches for both operating systems and applications. (On Windows, application support is limited to updates for applications released by Microsoft.)

This document supports Linux, macOS, and Windows Server managed nodes. The document will perform the appropriate actions for each platform.

#### Linux

On Linux managed nodes, the `AWS-RunPatchBaselineWithHooks` document invokes a Python module, which in turn downloads a snapshot of the patch baseline that applies to the managed node. This patch baseline snapshot uses the defined rules and lists of approved and blocked patches to drive the appropriate package manager for each node type:

- Amazon Linux, Amazon Linux 2, CentOS, Oracle Linux, and RHEL 7 managed nodes use YUM. For YUM operations, Patch Manager requires Python 2.6 or later.
- RHEL 8 managed nodes 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 Server, Raspberry Pi OS, and Ubuntu Server instances use APT. For APT operations, Patch Manager requires Python 3.
- SUSE Linux Enterprise Server managed nodes use Zypper. For Zypper operations, Patch Manager requires Python 2.6 or later.

#### macOS

On macOS managed nodes, the `AWS-RunPatchBaselineWithHooks` document invokes a Python module, which in turn downloads a snapshot of the patch baseline that applies to the managed node. Next, a Python subprocess invokes the CLI on the node to retrieve the installation and update information for the specified package managers and to drive the appropriate package manager for each update package.

#### Windows

On Windows Server managed nodes, the `AWS-RunPatchBaselineWithHooks` document downloads and invokes a PowerShell module, which in turn downloads a snapshot of the patch baseline that applies to the managed node. This patch baseline snapshot contains a list of approved patches that is compiled by querying the patch baseline against a Windows Server Update Services

(WSUS) server. This list is passed to the Windows Update API, which controls downloading and installing the approved patches as appropriate.

Each snapshot is specific to an AWS account, patch group, operating system, and snapshot ID. The snapshot is delivered through a presigned Amazon Simple Storage Service (Amazon S3) URL, which expires 24 hours after the snapshot is created. After the URL expires, however, if you want to apply the same snapshot content to other managed nodes, you can generate a new presigned Amazon S3 URL up to three days after the snapshot was created. To do this, use the [get-deployable-patch-snapshot-for-instance](#) command.

After all approved and applicable updates have been installed, with reboots performed as necessary, patch compliance information is generated on an managed node and reported back to Patch Manager.

**Note**

If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaselineWithHooks document, the managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1136\)](#).

For information about viewing patch compliance data, see [About patch compliance \(p. 840\)](#).

## AWS-RunPatchBaselineWithHooks operational steps

When the AWS-RunPatchBaselineWithHooks runs, the following steps are performed:

1. **Scan** - A Scan operation using AWS-RunPatchBaseline is run on the managed node, and a compliance report is generated and uploaded.
2. **Verify local patch states** - A script is run to determine what steps will be performed based on the selected operation and Scan result from Step 1.
  - a. If the selected operation is Scan, the operation is marked complete. The operation concludes.
  - b. If the selected operation is Install, Patch Manager evaluates the Scan result from Step 1 to determine what to run next:
    - i. If no missing patches are detected, and no pending reboots required, the operation proceeds directly to the final step (Step 8), which includes a hook you have provided. Any steps in between are skipped.
    - ii. If no missing patches are detected, but there are pending reboots required and the selected reboot option is NoReboot, the operation proceeds directly to the final step (Step 8), which includes a hook you have provided. Any steps in between are skipped.
    - iii. Otherwise, the operation proceeds to the next step.
3. **Pre-patch hook operation** - The SSM document you have provided for the first lifecycle hook, PreInstallHookDocName, is run on the managed node.
4. **Install with NoReboot** - An Install operation with the reboot option of NoReboot using AWS-RunPatchBaseline is run on the managed node, and a compliance report is generated and uploaded.
5. **Post-install hook operation** - The SSM document you have provided for the second lifecycle hook, PostInstallHookDocName, is run on the managed node.
6. **Verify reboot** - A script runs to determine whether a reboot is needed for the managed node and what steps to run:
  - a. If the selected reboot option is NoReboot, the operation proceeds directly to the final step (Step 8), which includes a hook you have provided. Any steps in between are skipped.
  - b. If the selected reboot option is RebootIfNeeded, Patch Manager checks whether there are any pending reboots required from the inventory collected in Step 4.
    - i. If no patches requiring a reboot are found, the managed node patching operation is complete, and the operation proceeds directly to the final step (Step 8), which includes a hook you have provided. Any steps in between are skipped.

- ii. If patches requiring a reboot are found, the operation continues to the next step.
- 7. **Reboot and report** - An installation operation with the reboot option of `RebootIfNeeded` runs on the managed node using `AWS-RunPatchBaseline`, and a compliance report is generated and uploaded.
- 8. **Post-reboot hook operation** - The SSM document you have provided for the third lifecycle hook, `OnExitHookDocName`, is run on the managed node.

For a `Scan` operation, if Step 1 fails, the process of running the document stops and the step is reported as failed, although subsequent steps are reported as successful.

For an `Install` operation, if any of the `aws:runDocument` steps fail during the operation, those steps are reported as failed, and the operation proceeds directly to the final step (Step 8), which includes a hook you have provided. Any steps in between are skipped. This step is reported as failed, the last step reports the status of its operation result, and all steps in between are reported as successful.

## [AWS-RunPatchBaselineWithHooks parameters](#)

`AWS-RunPatchBaselineWithHooks` supports six parameters.

The `Operation` parameter is required.

The `RebootOption`, `PreInstallHookDocName`, `PostInstallHookDocName`, and `OnExitHookDocName` parameters are optional.

`Snapshot-ID` is technically optional, but we recommend that you supply a custom value for it when you run `AWS-RunPatchBaselineWithHooks` outside of a maintenance window. Let Patch Manager supply the value automatically when the document is run as part of a maintenance window operation.

### Parameters

- [Parameter name: Operation \(p. 1130\)](#)
- [Parameter name: Snapshot ID \(p. 1150\)](#)
- [Parameter name: RebootOption \(p. 1151\)](#)
- [Parameter name: PreInstallHookDocName \(p. 1152\)](#)
- [Parameter name: PostInstallHookDocName \(p. 1153\)](#)
- [Parameter name: OnExitHookDocName \(p. 1153\)](#)

#### [Parameter name: Operation](#)

**Usage:** Required.

**Options:** `Scan` | `Install`.

##### Scan

When you choose the `Scan` option, the system uses the `AWS-RunPatchBaseline` document to determine the patch compliance state of the managed node and reports this information back to Patch Manager. `Scan` doesn't prompt updates to be installed or managed nodes to be rebooted. Instead, the operation identifies where updates are missing that are approved and applicable to the node.

##### Install

When you choose the `Install` option, `AWS-RunPatchBaselineWithHooks` attempts to install the approved and applicable updates that are missing from the managed node. Patch compliance information generated as part of an `Install` operation doesn't list any missing updates, but might report updates that are in a failed state if the installation of the update didn't succeed for any reason. Whenever an update is installed on a managed node, the node is rebooted to ensure the

update is both installed and active. (Exception: If the `RebootOption` parameter is set to `NoReboot` in the `AWS-RunPatchBaselineWithHooks` document, the managed node isn't rebooted after Patch Manager runs. For more information, see [Parameter name: RebootOption \(p. 1151\)](#).)

**Note**

If a patch specified by the baseline rules is installed *before* Patch Manager updates the managed node, 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 managed nodes 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're running `AWS-RunPatchBaselineWithHooks` in a maintenance window, as described in the following table.

**AWS-RunPatchBaselineWithHooks best practices**

Mode	Best practice	Details
Running AWS- <code>RunPatchBaselineWithHooks</code> inside a maintenance window	Don't supply a Snapshot ID. Patch Manager will supply it for you.	If you use a maintenance window to run AWS- <code>RunPatchBaselineWithHooks</code> , you shouldn't 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 AWS- <code>RunPatchBaselineWithHooks</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 three days. After that, a new snapshot will be generated even if you specify the same ID after the snapshot expires.
Running AWS- <code>RunPatchBaselineWithHooks</code> outside of a maintenance window	Generate and specify a custom GUID value for the Snapshot ID. <sup>1</sup>	When you aren't using a maintenance window to run AWS- <code>RunPatchBaselineWithHooks</code> , we recommend that you generate and specify a unique Snapshot ID for each patch baseline, particularly if you're running the AWS- <code>RunPatchBaselineWithHooks</code> document on multiple managed

Mode	Best practice	Details
		<p>nodes in the same operation. If you don't specify an ID in this scenario, Systems Manager generates a different Snapshot ID for each managed node the command is sent to. This might result in varying sets of patches being specified among the nodes.</p> <p>For instance, say that you're running the <code>AWS-RunPatchBaselineWithHooks</code> document directly through Run Command, a capability of AWS Systems Manager, and targeting a group of 50 managed nodes. Specifying a custom Snapshot ID results in the generation of a single baseline snapshot that is used to evaluate and patch all the managed nodes, ensuring that they end up in a consistent state.</p>

<sup>1</sup> 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:** `RebootOption`

**Usage:** Optional.

**Options:** `RebootIfNeeded` | `NoReboot`

**Default:** `RebootIfNeeded`

**Warning**

The default option is `RebootIfNeeded`. Be sure to select the correct option for your use case. For example, if your managed nodes must reboot immediately to complete a configuration process, choose `RebootIfNeeded`. Or, if you need to maintain managed node availability until a scheduled reboot time, choose `NoReboot`.

**RebootIfNeeded**

When you choose the `RebootIfNeeded` option, the managed node is rebooted in either of the following cases:

- Patch Manager installed one or more patches.

Patch Manager doesn't evaluate whether a reboot is *required* by the patch. The system is rebooted even if the patch doesn't require a reboot.

- Patch Manager detects one or more 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.

**Note**

Patches installed outside of Patch Manager are never given a status of `INSTALLED_PENDING_REBOOT`.

Rebooting managed nodes in these two cases ensures that updated packages are flushed from memory and keeps patching and rebooting behavior consistent across all operating systems.

**NoReboot**

When you choose the `NoReboot` option, Patch Manager doesn't reboot a managed node even if it installed patches during the `Install` operation. This option is useful if you know that your managed nodes don't require rebooting after patches are applied, or you have applications or processes running on a node that shouldn't be disrupted by a patching operation reboot. It's also useful when you want more control over the timing of managed node reboots, such as by using a maintenance window.

**Note**

If you choose the `NoReboot` option and a patch is installed, the patch is assigned a status of `InstalledPendingReboot`. The managed node itself, however, is marked as `Non-Compliant`. After a reboot occurs and a `Scan` operation is run, the node status is updated to `Compliant`.

**Patch installation tracking file:** To track patch installation, especially patches that were installed since the last system reboot, Systems Manager maintains a file on the managed node.

**Important**

Don't delete or modify the tracking file. If this file is deleted or corrupted, the patch compliance report for the managed node is inaccurate. If this happens, reboot the node and run a patch `Scan` operation to restore the file.

This tracking file is stored in the following locations on your managed nodes:

- Linux operating systems:
  - `/var/log/amazon/ssm/patch-configuration/patch-states-configuration.json`
  - `/var/log/amazon/ssm/patch-configuration/patch-inventory-from-last-operation.json`
- Windows Server operating system:
  - `C:\ProgramData\Amazon\PatchBaselineOperations\State\PatchStatesConfiguration.json`
  - `C:\ProgramData\Amazon\PatchBaselineOperations\State\PatchInventoryFromLastOperation.json`

**Parameter name:** `PreInstallHookDocName`

**Usage:** Optional.

**Default:** `AWS-Noop`.

The value to provide for the `PreInstallHookDocName` parameter is the name or Amazon Resource Name (ARN) of an SSM document of your choice. You can provide the name of an AWS managed document or the name or ARN of a custom SSM document that you have created or that has been shared with you. (For an SSM document that has been shared with you from a different AWS account, you must specify the full resource ARN, such as `aws:arn:ssm:us-east-2:123456789012:document/MySharedDocument`.)

The SSM document you specify is run before the `Install` operation and performs any actions supported by SSM Agent, such as a shell script to check application health check before patching is

performed on the managed node. (For a list of actions, see [Systems Manager Command document plugin reference \(p. 1314\)](#)). The default SSM document name is AWS-Noop, which doesn't perform any operation on the managed node.

For information about creating a custom SSM document, see [Creating SSM documents \(p. 1352\)](#).

**Parameter name:** PostInstallHookDocName

**Usage:** Optional.

**Default:** AWS-Noop.

The value to provide for the PostInstallHookDocName parameter is the name or Amazon Resource Name (ARN) of an SSM document of your choice. You can provide the name of an AWS managed document or the name or ARN of a custom SSM document that you have created or that has been shared with you. (For an SSM document that has been shared with you from a different AWS account, you must specify the full resource ARN, such as aws:arn:ssm:us-east-2:123456789012:document/MySharedDocument.)

The SSM document you specify is run after the Install with NoReboot operation and performs any actions supported by SSM Agent, such as a shell script for installing third party updates before reboot. (For a list of actions, see [Systems Manager Command document plugin reference \(p. 1314\)](#)). The default SSM document name is AWS-Noop, which doesn't perform any operation on the managed node.

For information about creating a custom SSM document, see [Creating SSM documents \(p. 1352\)](#).

**Parameter name:** OnExitHookDocName

**Usage:** Optional.

**Default:** AWS-Noop.

The value to provide for the OnExitHookDocName parameter is the name or Amazon Resource Name (ARN) of an SSM document of your choice. You can provide the name of an AWS managed document or the name or ARN of a custom SSM document that you have created or that has been shared with you. (For an SSM document that has been shared with you from a different AWS account, you must specify the full resource ARN, such as aws:arn:ssm:us-east-2:123456789012:document/MySharedDocument.)

The SSM document you specify is run after the managed node reboot operation and performs any actions supported by SSM Agent, such as a shell script to verify node health after the patching operation is complete. (For a list of actions, see [Systems Manager Command document plugin reference \(p. 1314\)](#)). The default SSM document name is AWS-Noop, which doesn't perform any operation on the managed node.

For information about creating a custom SSM document, see [Creating SSM documents \(p. 1352\)](#).

## Sample scenario for using the InstallOverrideList parameter in AWS-RunPatchBaseline or AWS-RunPatchBaselineAssociation

You can use the InstallOverrideList parameter when you want to override the patches specified by the current default patch baseline in Patch Manager, a capability of AWS Systems Manager. This topic provides examples that show how to use this parameter to achieve the following:

- Apply different sets of patches to a target group of managed nodes.
- 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 managed nodes. 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.

With the following strategy, 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 maintenance window that runs `AWS-RunPatchBaseline` or `AWS-RunPatchBaselineAssociation` each week. Don't 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 Simple Storage Service (Amazon S3) bucket.
4. Create a second maintenance window that runs once a month. However, 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.

## Using the BaselineOverride parameter

You can define patching preferences at runtime using the baseline override feature in Patch Manager, a capability of AWS Systems Manager. Do this by specifying an Amazon Simple Storage Service (Amazon S3) bucket containing a JSON object with a list of patch baselines. The patching operation uses the baselines provided in the JSON object that match the host operating system instead of applying the rules from the default patch baseline.

### Note

Using the `BaselineOverride` parameter doesn't overwrite the patch compliance of the baseline provided in the parameter. The output results are recorded in the Stdout logs from Run Command, a capability of AWS Systems Manager. The results only print out packages that are marked as `NON_COMPLIANT`. This means the package is marked as `Missing`, `Failed`, `InstalledRejected`, or `InstalledPendingReboot`.

## Using the patch baseline override with Snapshot Id or Install Override List parameters

There are two cases where the patch baseline override has noteworthy behavior.

### Using baseline override and Snapshot Id at the same time

Snapshot IDs ensure that all managed nodes in a particular patching command all apply the same thing. For example, if you patch 1,000 nodes at one time, the patches will be the same.

When using both a Snapshot Id and a patch baseline override, the Snapshot Id takes precedence over the patch baseline override. The baseline override rules will still be used, but they will only be evaluated once. In the earlier example, the patches across your 1,000 managed nodes will still always be the same. If, midway through the patching operation, you changed the JSON file in the referenced S3 bucket to be something different, the patches applied will still be the same. This is because the Snapshot Id was provided.

### Using baseline override and Install Override List at the same time

You can't use these two parameters at the same time. The patching document fails if both parameters are supplied, and it doesn't perform any scans or installs on the managed node.

## Code examples

The following code example for Python shows how to generate the patch baseline override.

```
import boto3
import json

ssm = boto3.client('ssm')
s3 = boto3.resource('s3')
s3_bucket_name = 'my-baseline-override-bucket'
s3_file_name = 'MyBaselineOverride.json'
baseline_ids_to_export = ['pb-0000000000000000', 'pb-0000000000000001']

baseline_overrides = []
for baseline_id in baseline_ids_to_export:
 baseline_overrides.append(ssm.get_patch_baseline(
 BaselineId=baseline_id
))

json_content = json.dumps(baseline_overrides, indent=4, sort_keys=True, default=str)
s3.Object(bucket_name=s3_bucket_name, key=s3_file_name).put(Body=json_content)
```

This produces a patch baseline override like the following.

```
[{
 "ApprovalRules": {
 "PatchRules": [
 {
 "ApproveAfterDays": 0,
 "ComplianceLevel": "UNSPECIFIED",
 "EnableNonSecurity": false,
 "PatchFilterGroup": {
 "PatchFilters": [
 {
 "Key": "PRODUCT",
 "Values": [
 "*"
]
 },
 {
 "Key": "CLASSIFICATION",
 "Values": [
 "*"
]
 },
 {
 "Key": "SEVERITY",
 "Values": [
 "*"
]
 }
]
 }
 }
],
 "ApprovedPatches": [],
 "ApprovedPatchesComplianceLevel": "UNSPECIFIED",
 "ApprovedPatchesEnableNonSecurity": false,
 "GlobalFilters": {}}
```

```
 "PatchFilters": [],
 },
 "OperatingSystem": "AMAZON_LINUX_2",
 "RejectedPatches": [],
 "RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
 "Sources": []
},
{
 "ApprovalRules": {
 "PatchRules": [
 {
 "ApproveUntilDate": "2021-01-06",
 "ComplianceLevel": "UNSPECIFIED",
 "EnableNonSecurity": true,
 "PatchFilterGroup": {
 "PatchFilters": [
 {
 "Key": "PRODUCT",
 "Values": [
 "*"
]
 },
 {
 "Key": "CLASSIFICATION",
 "Values": [
 "*"
]
 },
 {
 "Key": "SEVERITY",
 "Values": [
 "*"
]
 }
]
 }
 }
],
 "ApprovedPatches": [
 "open-ssl*"
],
 "ApprovedPatchesComplianceLevel": "UNSPECIFIED",
 "ApprovedPatchesEnableNonSecurity": false,
 "GlobalFilters": {
 "PatchFilters": []
 },
 "OperatingSystem": "CENTOS",
 "RejectedPatches": [
 "python*"
],
 "RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
 "Sources": []
 }
]
```

## About patch baselines

The topics in this section provide information about how patch baselines work in Patch Manager, a capability of AWS Systems Manager, when you run a Scan or Install operation on your managed nodes.

The information in the following topics applies both when you're patching managed nodes on a schedule and patching nodes on demand.

#### Topics

- [About predefined and custom patch baselines \(p. 1157\)](#)
- [About package name formats for approved and rejected patch lists \(p. 1161\)](#)
- [About patch groups \(p. 1163\)](#)
- [About patching applications released by Microsoft on Windows Server \(p. 1166\)](#)

## About predefined and custom patch baselines

Patch Manager, a capability of AWS Systems 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 custom patch baselines. Custom patch baselines allows you greater control over which patches are approved or rejected for your environment. Also, the predefined baselines assign a compliance level of `Unspecified` to all patches installed using those baselines. For compliance values to be assigned, you can create a copy of a predefined baseline and specify the compliance values you want to assign to patches. For more information, see [About custom baselines \(p. 1159\)](#) and [Working with custom patch baselines \(console\) \(p. 1194\)](#).

#### Topics

- [About predefined baselines \(p. 1157\)](#)
- [About custom baselines \(p. 1159\)](#)

## 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 \(p. 1094\)](#).

Name	Supported operating system	Details
AWS-AmazonLinuxDefaultPatchBaseline	Amazon Linux	Approves all operating system patches that are classified as "Security" and that have a severity level of "Critical" or "Important". Patches are auto-approved seven days after release. Also auto-approves all patches with a classification of "Bugfix" seven days after release.
AWS-AmazonLinux2DefaultPatchBaseline	Amazon Linux 2	Approves all operating system patches that are classified as "Security" and that have a severity level of "Critical" or "Important". Patches are auto-approved seven days after release. Also approves all patches with a classification of "Bugfix" seven days after release.
AWS-CentOSDefaultPatchBaseline	CentOS and CentOS Stream	Approves all updates seven days after they become available, including nonsecurity updates.

Name	Supported operating system	Details
AWS-DebianDefaultPatchBaseline	Debian Server	Immediately approves all operating system security-related patches that have a priority of "Required", "Important", "Standard," "Optional," or "Extra." There is no wait before approval because reliable release dates aren't available in the repos.
AWS-MacOSDefaultPatchBaseline	macOS	Approves all operating system patches that are classified as "Security". Also approves all packages with a current update.
AWS-OracleLinuxDefaultPatchBaseline	Oracle Linux	Approves all operating system patches that are classified as "Security" and that have a severity level of "Important" or "Moderate". Patches are auto-approved seven days after release. Also approves all patches that are classified as "Bugfix" seven days after release.
AWS-DefaultRaspbianPatchBaseline	Raspberry Pi OS	Immediately approves all operating system security-related patches that have a priority of "Required", "Important", "Standard," "Optional," or "Extra." There is no wait before approval because reliable release dates aren't available in the repos.
AWS-RedHatDefaultPatchBaseline	Red Hat Enterprise Linux (RHEL)	Approves all operating system patches that are classified as "Security" and that have a severity level of "Critical" or "Important". Patches are auto-approved seven days after release. Also approves all patches that are classified as "Bugfix" seven days after release.
AWS-RockyLinuxDefaultPatchBaseline	Rocky Linux	Approves all operating system patches that are classified as "Security" and that have a severity level of "Critical" or "Important". Patches are auto-approved seven days after release. Also approves all patches that are classified as "Bugfix" seven days after release.

Name	Supported operating system	Details
AWS-SuseDefaultPatchBaseline	SUSE Linux Enterprise Server (SLES)	Approves all operating system patches that are classified as "Security" and with a severity of "Critical" or "Important". Patches are auto-approved seven days after release.
AWS-UbuntuDefaultPatchBaseline	Ubuntu Server	Immediately approves all operating system security-related patches that have a priority of "Required", "Important", "Standard," "Optional," or "Extra." There is no wait before approval because reliable release dates aren't available in the repos.
AWS-DefaultPatchBaseline	Windows Server	Approves all Windows Server operating system patches that are classified as "CriticalUpdates" or "SecurityUpdates" and that have an MSRC severity of "Critical" or "Important". Patches are auto-approved seven days after release.
AWS-WindowsPredefinedPatchBaseline-OS	Windows Server	Approves all Windows Server operating system patches that are classified as "CriticalUpdates" or "SecurityUpdates" and that have an MSRC severity of "Critical" or "Important". Patches are auto-approved seven days after release.
AWS-WindowsPredefinedPatchBaseline-OS-Applications	Windows Server	For the Windows Server operating system, approves all patches that are classified as "CriticalUpdates" or "SecurityUpdates" and that have an MSRC severity of "Critical" or "Important". For applications released by Microsoft, approves all patches. Patches for both OS and applications are auto-approved seven days after release.

## 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 applications released by Microsoft 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 aren't 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 managed nodes. For example, you can create separate patch baselines, auto-approval delays, and cutoff dates for development and production environments. This allows 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, three predefined patch baselines are provided. The patch baselines `AWS-DefaultPatchBaseline` and `AWS-WindowsPredefinedPatchBaseline-OS` support only operating system updates on the Windows operating system itself. `AWS-DefaultPatchBaseline` is used as the default patch baseline for Windows Server managed nodes unless you specify a different patch baseline. The configuration settings in these two patch baselines are the same. The newer of the two, `AWS-WindowsPredefinedPatchBaseline-OS`, was created to distinguish it from the third predefined patch baseline for Windows Server. That patch baseline, `AWS-WindowsPredefinedPatchBaseline-OS-Applications`, can be used to apply patches to both the Windows Server operating system and supported applications released by Microsoft.

- For on-premises servers and virtual machines (VMs), 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.
- A managed node 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're 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 \(p. 1161\)](#).

For information about creating a patch baseline, see [Working with custom patch baselines \(console\) \(p. 1194\)](#) and [Walkthrough: Patch a server environment \(AWS CLI\) \(p. 1238\)](#).

## 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're 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 \(RHEL\) \(p. 1161\)](#)
- [Debian Server, Raspberry Pi OS \(formerly Raspbian\), and Ubuntu Server \(p. 1162\)](#)
- [SUSE Linux Enterprise Server \(SLES\) \(p. 1162\)](#)

### [Amazon Linux, Amazon Linux 2, CentOS, Oracle Linux, and Red Hat Enterprise Linux \(RHEL\)](#)

**Package manager:** YUM, except for RHEL 8 and CentOS 8, which use 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

## Debian Server, Raspberry Pi OS (formerly Raspbian), 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 Debian Server lists, Raspberry Pi OS lists, and Ubuntu Server lists, don't 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 macOS

**Supported package managers:** softwareupdate, installer, Brew, Brew Cask

**Approved patches and rejected patches:** For both approved and rejected patch lists, you specify full package names, in formats such as:

- XProtectPlistConfigData
- MRTConfigData

Wildcards aren't supported in approved and rejected patch lists for macOS.

## 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 managed nodes with a specific patch baseline in Patch Manager, a capability of AWS Systems Manager. Patch groups help ensure that you're deploying the appropriate patches, based on the associated patch baseline rules, to the correct set of nodes. 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.

When you run `AWS-RunPatchBaseline`, you can target managed nodes using their 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 managed node.

You create a patch group by using Amazon Elastic Compute Cloud (Amazon EC2) tags. Unlike other tagging scenarios across Systems Manager, a patch group *must* be defined with the tag key: **Patch Group**. The key is case-sensitive. You can specify any value, for example "web servers," but the key must be **Patch Group**.

After you create a patch group and tag managed nodes, you can register the patch group with a patch baseline. Registering the patch group with a patch baseline ensures that the nodes within the patch group use the rules defined in the associated patch baseline.

For more information about how to create a patch group and associate the patch group to a patch baseline, see [Working with patch groups \(p. 1205\)](#) and [Add a patch group to a patch baseline \(p. 1207\)](#).

To view an example of creating a patch baseline and patch groups by using the AWS Command Line Interface (AWS CLI), see [Walkthrough: Patch a server environment \(AWS CLI\) \(p. 1238\)](#). For more information about Amazon EC2 tags, see [Tag your Amazon EC2 resources](#) in the *Amazon EC2 User Guide for Linux Instances*.

### How it works

When the system runs the task to apply a patch baseline to a managed node, SSM Agent verifies that a patch group value is defined for the node. If the node 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 a node isn't configured for a patch group, Patch Manager automatically notifies SSM Agent to use the currently configured default patch baseline.

#### Important

A managed node can only be in one patch group.

A patch group can be registered with only one patch baseline for each operating system type.

To apply the `Patch Group` tag to an Amazon EC2 instance, the **Allow tags in instance metadata** option must not be enabled on the instance. Allowing tags in instance metadata prevents tag key names from containing spaces. For information about disabling the setting if you have enabled it, see [Turn off access to tags in instance metadata](#) in the *Amazon EC2 User Guide for Linux Instances*.

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:

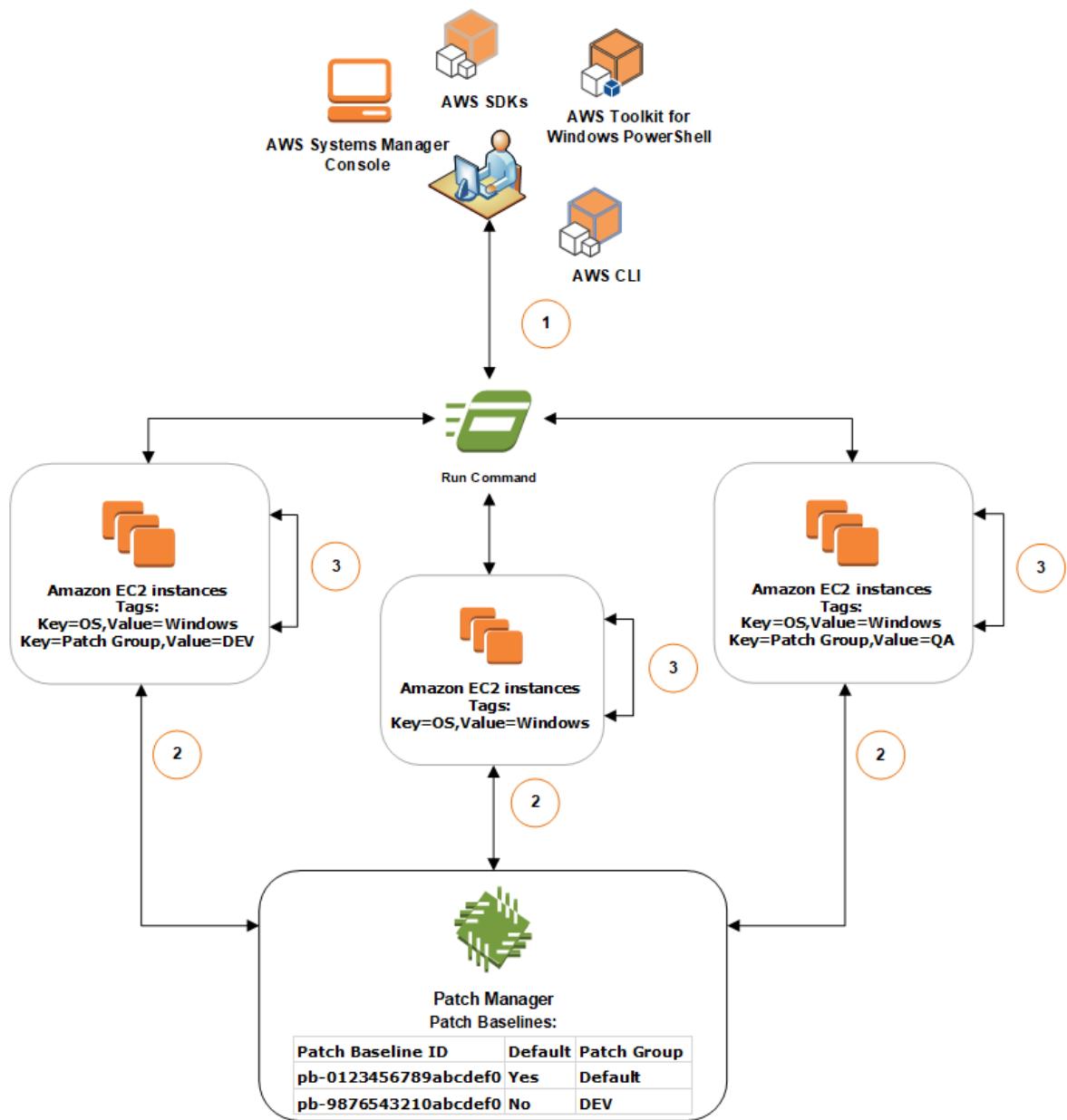
EC2 instances group	Tags
Group 1	key=OS,value=Windows

EC2 instances group	Tags
	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 Server patch baselines:

Patch baseline ID	Default	Associated patch group
pb-0123456789abcdef0	Yes	Default
pb-9876543210abcdef0	No	DEV

**Diagram 1: General example of patching operations process flow**



The general process to scan or install patches using Run Command, a capability of AWS Systems Manager, and Patch Manager is as follows:

- Send a command to patch:** Use the Systems Manager console, SDK, AWS Command Line Interface (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`.
- 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:**
    - 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 doesn't 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

- [Understanding patch compliance state values \(p. 1186\)](#)

## About patching applications released by Microsoft on Windows Server

Use the information in this topic to help you prepare to patch applications on Windows Server using Patch Manager, a capability of AWS Systems Manager.

### Microsoft application patching

Patching support for applications on Windows Server managed nodes is limited to applications released by Microsoft.

#### Note

In some cases, Microsoft releases patches for applications that might not specify explicitly an updated date and time. In these cases, an updated date and time of 01/01/1970 is supplied by default.

### Patch baselines to patch applications released by Microsoft

For Windows Server, three predefined patch baselines are provided. The patch baselines AWS-DefaultPatchBaseline and AWS-WindowsPredefinedPatchBaseline-OS support only operating

system updates on the Windows operating system itself. `AWS-DefaultPatchBaseline` is used as the default patch baseline for Windows Server managed nodes unless you specify a different patch baseline. The configuration settings in these two patch baselines are the same. The newer of the two, `AWS-WindowsPredefinedPatchBaseline-OS`, was created to distinguish it from the third predefined patch baseline for Windows Server. That patch baseline, `AWS-WindowsPredefinedPatchBaseline-OS-Applications`, can be used to apply patches to both the Windows Server operating system and supported applications released by Microsoft.

You can also create a custom patch baseline to update applications released by Microsoft on Windows Server machines.

### Support for patching applications released by Microsoft on virtual machines (VMs) and managed nodes

To patch applications released by Microsoft on virtual machines (VMs) and managed nodes, you must turn on the advanced-instances tier. There is a charge to use the advanced-instances tier. There is no additional charge to patch applications released by Microsoft on Amazon Elastic Compute Cloud (Amazon EC2) instances. For more information, see [Turning on the advanced-instances tier \(p. 805\)](#).

#### Windows update option for "other Microsoft products"

In order for Patch Manager to be able to patch applications released by Microsoft on your Windows Server managed nodes, the Windows Update option **Give me updates for other Microsoft products when I update Windows** must be activated on the managed node.

For information about allowing this option on a single managed node, see [Update Office with Microsoft Update](#) on the Microsoft Support website.

For a fleet of managed nodes running Windows Server 2016 and later, you can use a Group Policy Object (GPO) to turn on the setting. In the Group Policy Management Editor, go to **Computer Configuration**, **Administrative Templates**, **Windows Components**, **Windows Updates**, and choose **Install updates for other Microsoft products**.

For a fleet of managed nodes running Windows Server 2012 or 2012 R2 , you can turn on the option by using a script, as described in [Enabling and Disabling Microsoft Update in Windows 7 via Script](#) on the Microsoft Docs Blog website. For example, you could do the following:

1. Save the script from the blog post in a file.
2. Upload the file to an Amazon Simple Storage Service (Amazon S3) bucket or other accessible location.
3. Use Run Command, a capability of AWS Systems Manager, to run the script on your managed nodes using the Systems Manager document (SSM document) `AWS-RunPowerShellScript` with a command similar to the following.

```
Invoke-WebRequest `
-Uri "https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/script.vbs" `
-Outfile "C:\script.vbs" cscript c:\script.vbs
```

#### Minimum parameter requirements

To include applications released by Microsoft in your custom patch baseline, you must, at a minimum, specify the product that you want to patch. The following AWS Command Line Interface (AWS CLI) command demonstrates the minimal requirements to patch a product, such as Microsoft Office 2016.

#### Linux & macOS

```
aws ssm create-patch-baseline \
--name "My-Windows-App-Baseline" \
--product MicrosoftOffice2016
```

```
--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 matched pairing of product family and product.

### Linux & macOS

```
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]"
```

### 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]"
```

#### Note

If you receive an error message about a mismatched product and family pairing, see [Issue: mismatched product family/product pairs \(p. 1248\)](#) for help resolving the issue.

## Using Kernel Live Patching on Amazon Linux 2 managed nodes

Kernel Live Patching for Amazon Linux 2 allows you to apply security vulnerability and critical bug patches to a running Linux kernel without reboots or disruptions to running applications. This allows you to benefit from improved service and application availability, while keeping your infrastructure secure and up to date. Kernel Live Patching is supported on Amazon EC2 instances, AWS IoT Greengrass core devices, and [on-premises virtual machines](#) running Amazon Linux 2.

For general information about Kernel Live Patching, see [Kernel Live Patching on Amazon Linux 2](#) in the [Amazon EC2 User Guide for Linux Instances](#).

After you turn on Kernel Live Patching on an Amazon Linux 2 managed node, you can use Patch Manager, a capability of AWS Systems Manager, to apply kernel live patches to the managed node. Using Patch Manager is an alternative to using existing yum workflows on the node to apply the updates.

## Before you begin

To use Patch Manager to apply kernel live patches to your Amazon Linux 2 managed nodes, ensure your nodes are based on the correct architecture and kernel version. For information, see [Supported configurations and prerequisites](#) in the *Amazon EC2 User Guide for Linux Instances*.

### Topics

- [About Kernel Live Patching and Patch Manager \(p. 1169\)](#)
- [How it works \(p. 1169\)](#)
- [Turning on Kernel Live Patching using Run Command \(p. 1170\)](#)
- [Applying kernel live patches using Run Command \(p. 1171\)](#)
- [Turning off Kernel Live Patching using Run Command \(p. 1173\)](#)

## About Kernel Live Patching and Patch Manager

### Updating the kernel version

You don't need to reboot a managed node after applying a kernel live patch update. However, AWS provides kernel live patches for an Amazon Linux 2 kernel version for up to three months after its release. After the three-month period, you must update to a later kernel version to continue to receive kernel live patches. We recommend using a maintenance window to schedule a reboot of your node at least once every three months to prompt the kernel version update.

### Uninstalling kernel live patches

Kernel live patches can't be uninstalled using Patch Manager. Instead, you can turn off Kernel Live Patching, which removes the RPM packages for the applied kernel live patches. For more information, see [Turning off Kernel Live Patching using Run Command \(p. 1173\)](#).

### Kernel compliance

In some cases, installing all CVE fixes from live patches for the current kernel version can bring that kernel into the same compliance state that a newer kernel version would have. When that happens, the newer version is reported as `Installed`, and the managed node reported as `Compliant`. No installation time is reported for newer kernel version, however.

### One kernel live patch, multiple CVEs

If a kernel live patch addresses multiple CVEs, and those CVEs have various classification and severity values, only the highest classification and severity from among the CVEs is reported for the patch.

The remainder of this section describes how to use Patch Manager to apply kernel live patches to managed nodes that meet these requirements.

## How it works

AWS releases two types of kernel live patches for Amazon Linux 2: security updates and bug fixes. To apply those types of patches, you use a patch baseline document that targets only the classifications and severities listed in the following table.

Classification	Severity
Security	Critical, Important
Bugfix	All

You can create a custom patch baseline that targets only these patches, or use the predefined AWS-AmazonLinux2DefaultPatchBaseline patch baseline. In other words, you can use AWS-AmazonLinux2DefaultPatchBaseline with Amazon Linux 2 managed nodes on which Kernel Live Patching is turned on, and kernel live updates will be applied.

**Note**

The AWS-AmazonLinux2DefaultPatchBaseline configuration specifies a seven-day waiting period after a patch is released before it's installed automatically. If you don't want to wait seven days for kernel live patches to be auto-approved, you can create and use a custom patch baseline. In your patch baseline, you can specify no auto-approval waiting period, or specify a shorter or longer one. For more information, see [Working with custom patch baselines \(console\) \(p. 1194\)](#).

We recommend the following strategy to patch your managed nodes with kernel live updates:

1. Turn on Kernel Live Patching on your Amazon Linux 2 managed nodes.
2. Use Run Command, a capability of AWS Systems Manager, to run a Scan operation on your managed nodes using the predefined AWS-AmazonLinux2DefaultPatchBaseline or a custom patch baseline that also targets only Security updates with severity classified as Critical and Important, and the Bugfix severity of All.
3. Use Compliance, a capability of AWS Systems Manager, to review whether non-compliance for patching is reported for any of the managed nodes that were scanned. If so, view the node compliance details to determine whether any kernel live patches are missing from the managed node.
4. To install missing kernel live patches, use Run Command with the same patch baseline you specified before, but this time run an Install operation instead of a Scan operation.

Because kernel live patches are installed without the need to reboot, you can choose the NoReboot reboot option for this operation.

**Note**

You can still reboot the managed node if required for other types of patches installed on it, or if you want to update to a newer kernel. In these cases, choose the RebootIfNeeded reboot option instead.

5. Return to Compliance to verify that the kernel live patches were installed.

## Turning on Kernel Live Patching using Run Command

To turn on Kernel Live Patching, you can either run yum commands on your managed nodes or use Run Command and a custom Systems Manager document (SSM document) that you create.

For information about turning on Kernel Live Patching by running yum commands directly on the managed node, see [Enable Kernel Live Patching](#) in the *Amazon EC2 User Guide for Linux Instances*.

**Note**

When you turn on Kernel Live Patching, if the kernel already running on the managed node is *earlier* than kernel-4.14.165-131.185.amzn2.x86\_64 (the minimum supported version), the process installs the latest available kernel version and reboots the managed node. If the node is already running kernel-4.14.165-131.185.amzn2.x86\_64 or later, the process doesn't install a newer version and doesn't reboot the node.

### To turn on Kernel Live Patching using Run Command (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 custom SSM document AWS-ConfigureKernelLivePatch.
5. In the **Command parameters** section, specify whether you want managed nodes to reboot as part of this operation.
6. For information about working with the remaining controls on this page, see [Running commands from the console \(p. 996\)](#).
7. Choose **Run**.

### To turn on Kernel Live Patching (AWS CLI)

- Run the following command on your local machine.

Linux & macOS

```
aws ssm send-command \
--document-name "AWS-ConfigureKernelLivePatch" \
--parameters "EnableOrDisable=Enable" \
--targets "Key=instanceids,Values=instance-id"
```

Windows

```
aws ssm send-command ^
--document-name "AWS-ConfigureKernelLivePatch" ^
--parameters "EnableOrDisable=Enable" ^
--targets "Key=instanceids,Values=instance-id"
```

Replace `instance-id` with the ID of the Amazon Linux 2 managed node on which you want to turn on the feature, such as i-02573cafEXAMPLE. To turn on the feature on multiple managed nodes, you can use either of the following formats.

- `--targets "Key=instanceids,Values=instance-id1,instance-id2"`
- `--targets "Key=tag:tag-key,Values=tag-value"`

For information about other options you can use in the command, see [send-command](#) in the *AWS CLI Command Reference*.

## Applying kernel live patches using Run Command

To apply kernel live patches, you can either run `yum` commands on your managed nodes or use Run Command and the SSM document AWS-RunPatchBaseline.

For information about applying kernel live patches by running `yum` commands directly on the managed node, see [Apply kernel live patches](#) in the *Amazon EC2 User Guide for Linux Instances*.

### To apply kernel live patches using Run Command (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 SSM document **AWS-RunPatchBaseline**.
5. In the **Command parameters** section, do one of the following:
  - If you're checking whether new kernel live patches are available, for **Operation**, choose **Scan**. For **Reboot Option**, if don't want your managed nodes to reboot after this operation, choose **NoReboot**. After the operation is complete, you can check for new patches and compliance status in **Compliance**.
  - If you checked patch compliance already and are ready to apply available kernel live patches, for **Operation**, choose **Install**. For **Reboot Option**, if you don't want your managed nodes to reboot after this operation, choose **NoReboot**.
6. For information about working with the remaining controls on this page, see [Running commands from the console \(p. 996\)](#).
7. Choose **Run**.

### To apply kernel live patches using Run Command (AWS CLI)

1. To perform a **Scan** operation before checking your results in **Compliance**, run the following command from your local machine.

Linux & macOS

```
aws ssm send-command \
--document-name "AWS-RunPatchBaseline" \
--targets "Key=InstanceIds,Values=instance-id" \
--parameters '{"Operation":["Scan"], "RebootOption":["RebootIfNeeded"]}'
```

Windows

```
aws ssm send-command ^
--document-name "AWS-RunPatchBaseline" ^
--targets "Key=InstanceIds,Values=instance-id" ^
--parameters {"Operation":["Scan"], "RebootOption":["RebootIfNeeded"]}
```

For information about other options you can use in the command, see [send-command](#) in the [AWS CLI Command Reference](#).

2. To perform an **Install** operation after checking your results in **Compliance**, run the following command from your local machine.

Linux & macOS

```
aws ssm send-command \
--document-name "AWS-RunPatchBaseline" \
--targets "Key=InstanceIds,Values=instance-id" \
--parameters '{"Operation":["Install"], "RebootOption":["NoReboot"]}'
```

Windows

```
aws ssm send-command ^
```

```
--document-name "AWS-RunPatchBaseline" ^
--targets "Key=InstanceIds,Values=instance-id" ^
--parameters {"Operation\":[\"Install\"], \"RebootOption\":[\"NoReboot\"]}
```

In both of the preceding commands, replace *instance-id* with the ID of the Amazon Linux 2 managed node on which you want to apply kernel live patches, such as i-02573cafefEXAMPLE. To turn on the feature on multiple managed nodes, you can use either of the following formats.

- --targets "Key=instanceids,Values=*instance-id1*,*instance-id2*"
- --targets "Key=tag:*tag-key*,Values=*tag-value*"

For information about other options you can use in these commands, see [send-command](#) in the *AWS CLI Command Reference*.

## Turning off Kernel Live Patching using Run Command

To turn off Kernel Live Patching, you can either run yum commands on your managed nodes or use Run Command and the custom SSM document AWS-ConfigureKernelLivePatching.

### Note

If you no longer need to use Kernel Live Patching, you can turn it off at any time. In most cases, turning off the feature isn't necessary.

For information about turning off Kernel Live Patching by running yum commands directly on the managed node, see [Enable Kernel Live Patching](#) in the *Amazon EC2 User Guide for Linux Instances*.

### Note

When you turn off Kernel Live Patching, the process uninstalls the Kernel Live Patching plugin and then reboots the managed node.

### To turn off Kernel Live Patching using Run Command (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 SSM document **AWS-ConfigureKernelLivePatching**.
5. In the **Command parameters** section, specify values for required parameters.
6. For information about working with the remaining controls on this page, see [Running commands from the console \(p. 996\)](#).
7. Choose **Run**.

### To turn off Kernel Live Patching (AWS CLI)

- Run a command similar to the following.

Linux & macOS

```
aws ssm send-command \
--document-name "AWS-ConfigureKernelLivePatching" \
```

```
--targets "Key=instanceIds,Values=instance-id" \
--parameters "EnableOrDisable=Disable"
```

#### Windows

```
aws ssm send-command ^
--document-name "AWS-ConfigureKernelLivePatching" ^
--targets "Key=instanceIds,Values=instance-id" ^
--parameters "EnableOrDisable=Disable"
```

Replace *instance-id* with the ID of the Amazon Linux 2 managed node on which you want to turn off the feature, such as i-02573cafefEXAMPLE. To turn off the feature on multiple managed nodes, you can use either of the following formats.

- --targets "Key=instanceids,Values=instance-id1,instance-id2"
- --targets "Key>tag:tag-key,Values=tag-value"

For information about other options you can use in the command, see [send-command](#) in the *AWS CLI Command Reference*.

## Working with Patch Manager (console)

To use Patch Manager, a capability of AWS Systems 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 doesn't, create a patch baseline that defines a standard set of patches for that managed node type and set it as the default instead.
2. Organize managed nodes into patch groups by using Amazon EC2 tags (optional, but recommended).
3. Schedule patching by using a maintenance window that defines which managed nodes to patch and when to patch them.

-or-

Patch or scan managed nodes on demand whenever you need to.

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 Command Line Interface (AWS CLI), see [Walkthrough: Patch a server environment \(AWS CLI\) \(p. 1238\)](#).
- For more information about maintenance windows, see [AWS Systems Manager Maintenance Windows \(p. 706\)](#).
- For information about monitoring patch compliance, see [About patch compliance \(p. 840\)](#).

### Topics

- [Viewing patch Dashboard summaries \(console\) \(p. 1175\)](#)
- [Working with patch compliance reports \(p. 1175\)](#)
- [Patching managed nodes on demand \(console\) \(p. 1190\)](#)
- [Working with patch baselines \(p. 1193\)](#)

- [Viewing available patches \(console\) \(p. 1203\)](#)
- [Creating a patching configuration \(console\) \(p. 1203\)](#)
- [Working with patch groups \(p. 1205\)](#)
- [Working with Patch Manager settings \(p. 1207\)](#)

## Viewing patch Dashboard summaries (console)

The **Dashboard** tab in Patch Manager provides you with a summary view in the console that you can use to monitor your patching operations in a consolidated view. Patch Manager is a capability of AWS Systems Manager. On the **Dashboard** tab, you can view the following:

- A snapshot of how many managed nodes are compliant and noncompliant with patching rules.
- A snapshot of the age of patch compliance results for your managed nodes.
- A linked count of how many noncompliant managed nodes there are for each of the most common reasons for noncompliance.
- A linked list of the most recent patching operations.
- A linked list of the recurring patching tasks that have been set up.

### To view patch Dashboard summaries

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Dashboard** tab.
4. Scroll to the section containing summary data that you want to view:
  - **Patch compliance summary**
  - **Compliance reporting age**
  - **Patch states**
  - **Patch operations history**
  - **Recurring patching tasks**
  - **Scheduled reboot tasks**

## Working with patch compliance reports

Use the information in the following topics to help you generate and work with patch compliance reports in Patch Manager, a capability of AWS Systems Manager.

### Topics

- [Viewing patch compliance results \(console\) \(p. 1175\)](#)
- [Generating .csv patch compliance reports \(console\) \(p. 1178\)](#)
- [Remediating out-of-compliance managed nodes with Patch Manager \(p. 1183\)](#)

### Viewing patch compliance results (console)

Use these procedures to view patch compliance information about your managed nodes.

This procedure applies to patch operations that use the `AWS-RunPatchBaseline` document. For information about viewing patch compliance information for patch operations that use the `AWS-RunPatchBaselineAssociation` document, see [Identifying out-of-compliance managed nodes \(p. 1184\)](#).

**Note**

The patch scan processes for Quick Setup and Explorer, AWS Systems Manager capabilities, use the `AWS-RunPatchBaselineAssociation` document.

### Identify the patch solution for a specific CVE issue (Linux)

For many Linux-based operating systems, patch compliance results indicate which Common Vulnerabilities and Exposure (CVE) bulletin issues are resolved by which patches. This information can help you determine how urgently you need to install a missing or failed patch.

CVE details are included for supported versions of the following operating system types:

- Amazon Linux
- Amazon Linux 2
- CentOS Stream
- CentOS Stream
- Oracle Linux
- Red Hat Enterprise Linux (RHEL)
- Rocky Linux
- SUSE Linux Enterprise Server (SLES)

**Note**

By default, CentOS doesn't provide CVE information about updates. You can, however, allow this support by using third-party repositories such as the Extra Packages for Enterprise Linux (EPEL) repository published by Fedora. For information, see [EPEL](#) on the Fedora Wiki.

You can also add CVE IDs to your lists of approved or rejected patches in your patch baselines, as the situation and your patching goals warrant.

For information about working with approved and rejected patch lists, see the following topics:

- [Working with custom patch baselines \(console\) \(p. 1194\)](#)
- [About package name formats for approved and rejected patch lists \(p. 1161\)](#)
- [How patch baseline rules work on Linux-based systems \(p. 1111\)](#)
- [How patches are installed \(p. 1103\)](#)

**Note**

In some cases, Microsoft releases patches for applications that might not specify explicitly an updated date and time. In these cases, an updated date and time of 01/01/1970 is supplied by default.

### Viewing patching compliance results (console)

Use the following procedures to view patch compliance results in the AWS Systems Manager console.

**Note**

For information about generating patch compliance reports that are downloaded to an Amazon Simple Storage Service (Amazon S3) bucket, see [Generating .csv patch compliance reports \(console\) \(p. 1178\)](#).

## To view patch compliance results

1. Do one of the following.

**Option 1** (recommended) – Navigate from Patch Manager, a capability of AWS Systems Manager:

- 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**.

- Choose the **Compliance reporting** tab.
- For **Instance patching summary**, choose the ID of the managed node for which you want to review patch compliance results.
- Choose **View details**.

**Option 2** – Navigate from Compliance, a capability of AWS Systems Manager:

- 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.

- For **Compliance resources summary**, choose a number in the column for the types of patch resources you want to review, such as **Non-Compliant resources**.
- In the **Resource** list, choose the ID of the managed node for which you want to review patch compliance results.

**Option 3** – Navigate from Fleet Manager, a capability of AWS Systems Manager.

- In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

- On the **Managed instances** tab, choose the ID of the managed node for which you want to review patch compliance results.
2. Choose the **Patch** tab.
  3. (Optional) In the Search box (🔍), choose from the available filters.

For example, for Red Hat Enterprise Linux (RHEL), choose from the following:

- Name
- Classification
- State
- Severity

For Windows Server, choose from the following:

- KB
- Classification

- State
  - Severity
4. Choose one of the available values for the filter type you chose. For example, if you chose **State**, now choose a compliance state such as **InstalledPendingReboot**, **Failed** or **Missing**.
  5. Depending on the compliance state of the managed node, you can choose what action to take to remedy any noncompliant nodes.

For example, you can choose to patch your noncompliant managed nodes immediately. For information about patching your managed nodes on demand, see [Patching managed nodes on demand \(console\) \(p. 1190\)](#).

For information about patch compliance states, see [Understanding patch compliance state values \(p. 1186\)](#).

## Generating .csv patch compliance reports (console)

You can use the AWS Systems Manager console to generate patch compliance reports that are saved as a .csv file to an Amazon Simple Storage Service (Amazon S3) bucket of your choice. You can generate a single on-demand report or specify a schedule for generating the reports automatically.

Reports can be generated for a single managed node or for all managed nodes in your selected AWS account and AWS Region. For a single node, a report contains comprehensive details, including the IDs of patches related to a node being noncompliant. For a report on all managed nodes, only summary information and counts of noncompliant nodes' patches are provided.

### Note

After a report is generated, you can use a tool like Amazon QuickSight to import and analyze the data. Amazon QuickSight is a business intelligence (BI) service you can use to explore and interpret information in an interactive visual environment. For more information, see the [Amazon QuickSight User Guide](#).

You can also specify an Amazon Simple Notification Service (Amazon SNS) topic to use for sending notifications when a report is generated.

### Service roles for generating patch compliance reports

The first time you generate a report, Systems Manager creates a service role named `AWS-SystemsManager-PatchSummaryExportRole` to use for the export process. The first time you generate a report on a schedule, Systems Manager creates another service role named `AWS-EventBridge-Start-SSMAutomationRole`, along with the service role `AWS-SystemsManager-PatchSummaryExportRole` (if not created already) to use for the export process. `AWS-EventBridge-Start-SSMAutomationRole` enables Amazon EventBridge to start an automation using the runbook [AWS-ExportPatchReportToS3](#).

We recommend against attempting to modify these policies and roles. Doing so could cause patch compliance report generation to fail. For more information, see [Troubleshooting patch compliance report generation \(p. 1182\)](#).

### Topics

- [What's in a generated patch compliance report? \(p. 1179\)](#)
- [Generating patch compliance reports for a single managed node \(p. 1180\)](#)
- [Generating patch compliance reports for all managed nodes \(p. 1181\)](#)
- [Viewing patch compliance reporting history \(p. 1182\)](#)
- [Viewing patch compliance reporting schedules \(p. 1182\)](#)
- [Troubleshooting patch compliance report generation \(p. 1182\)](#)

## [What's in a generated patch compliance report?](#)

This topic provides information about the types of content included in the patch compliance reports that are generated and downloaded to a specified S3 bucket.

### [Report format for a single managed node](#)

A report generated for a single managed node provides both summary and detailed information.

#### [Download a sample report \(single node\)](#)

Summary information for a single managed node includes the following:

- Index
- Instance ID
- Instance name
- Instance IP
- Platform name
- Platform version
- SSM Agent version
- Patch baseline
- Patch group
- Compliance status
- Compliance severity
- Noncompliant Critical severity patch count
- Noncompliant High severity patch count
- Noncompliant Medium severity patch count
- Noncompliant Low severity patch count
- Noncompliant Informational severity patch count
- Noncompliant Unspecified severity patch count

Detailed information for a single managed node includes the following:

- Index
- Instance ID
- Instance name
- Patch name
- KB ID/Patch ID
- Patch state
- Last report time
- Compliance level
- Patch severity
- Patch classification
- CVE ID
- Patch baseline
- Logs URL
- Instance IP
- Platform name
- Platform version
- SSM Agent version

## Report format for all managed nodes

A report generated for all managed nodes provides only summary information.

### [Download a sample report \(all managed nodes\)](#)

Summary information for all managed nodes includes the following:

- Index
- Instance ID
- Instance name
- Instance IP
- Platform name
- Platform version
- SSM Agent version
- Patch baseline
- Patch group
- Compliance status
- Compliance severity
- Noncompliant Critical severity patch count
- Noncompliant High severity patch count
- Noncompliant Medium severity patch count
- Noncompliant Low severity patch count
- Noncompliant Informational severity patch count
- Noncompliant Unspecified severity patch count

## Generating patch compliance reports for a single managed node

Use the following procedure to generate a patch summary report for a single managed node in your AWS account. The report for a single managed node provides details about each patch that is out of compliance, including patch names and IDs.

### To generate patch compliance reports for a single managed node

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Compliance reporting** tab.
4. Choose the button for the row of the managed node for which you want to generate a report, and then choose **View detail**.
5. In the **Patch summary** section, choose **Export to S3**.
6. For **Report name**, enter a name to help you identify the report later.
7. For **Reporting frequency**, choose one of the following:
  - **On demand** – Create a one-time report. Skip to Step 9.
  - **On a schedule** – Specify a recurring schedule for automatically generating reports. Continue to Step 8.

8. For **Schedule type**, specify either a rate expression, such as every 3 days, or provide a cron expression to set the report frequency.

For information about cron expressions, see [Reference: Cron and rate expressions for Systems Manager \(p. 1540\)](#).
  9. For **Bucket name**, select the name of an S3 bucket where you want to store the .csv report files.
- Important**  
If you're working in an AWS Region that was launched after March 20, 2019, you must select an S3 bucket in that same Region. Regions launched after that date were turned off by default. For more information and a list of these Regions, see [Enabling a Region](#) in the [Amazon Web Services General Reference](#).
10. (Optional) To send notifications when the report is generated, expand the **SNS topic** section, and then choose an existing Amazon SNS topic from **SNS topic Amazon Resource Name (ARN)**.
  11. Choose **Submit**.

For information about viewing a history of generated reports, see [Viewing patch compliance reporting history \(p. 1182\)](#).

For information about viewing details of reporting schedules you have created, see [Viewing patch compliance reporting schedules \(p. 1182\)](#).

### Generating patch compliance reports for all managed nodes

Use the following procedure to generate a patch summary report for all managed nodes in your AWS account. The report for all managed nodes indicates which nodes are out of compliance and the numbers of out-of-compliance patches. It doesn't provide the names or other identifiers of the patches. For these additional details, you can generate a patch compliance report for a single managed node. For information, see [Generating patch compliance reports for a single managed node \(p. 1180\)](#) earlier in this topic.

#### To generate patch compliance reports for all managed nodes

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Compliance reporting** tab.
4. Choose **Export to S3**. (Don't select a node ID first.)
5. For **Report name**, enter a name to help you identify the report later.
6. For **Reporting frequency**, choose one of the following:
  - **On demand** – Create a one-time report. Skip to Step 8.
  - **On a schedule** – Specify a recurring schedule for automatically generating reports. Continue to Step 7.
7. For **Schedule type**, specify either a rate expression, such as every 3 days, or provide a cron expression to set the report frequency.

For information about cron expressions, see [Reference: Cron and rate expressions for Systems Manager \(p. 1540\)](#).
8. For **Bucket name**, select the name of an S3 bucket where you want to store the .csv report files.

**Important**

If you're working in an AWS Region that was launched after March 20, 2019, you must select an S3 bucket in that same Region. Regions launched after that date were turned off by default. For more information and a list of these Regions, see [Enabling a Region](#) in the [Amazon Web Services General Reference](#).

9. (Optional) To send notifications when the report is generated, expand the **SNS topic** section, and then choose an existing Amazon SNS topic from **SNS topic Amazon Resource Name (ARN)**.
10. Choose **Submit**.

For information about viewing a history of generated reports, see [Viewing patch compliance reporting history \(p. 1182\)](#).

For information about viewing details of reporting schedules you have created, see [Viewing patch compliance reporting schedules \(p. 1182\)](#).

### [Viewing patch compliance reporting history](#)

Use the information in this topic to help you view details about the patch compliance reports generated in your AWS account.

#### **To view patch compliance reporting history**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Compliance reporting** tab.
4. Choose **View all S3 exports**, and then choose the **Export history** tab.

### [Viewing patch compliance reporting schedules](#)

Use the information in this topic to help you view details about the patch compliance reporting schedules created in your AWS account.

#### **To view patch compliance reporting history**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Compliance reporting** tab.
4. Choose **View all S3 exports**, and then choose the **Report schedule rules** tab.

### [Troubleshooting patch compliance report generation](#)

Use the following information to help you troubleshoot problems with generating patch compliance report generation in Patch Manager, a capability of AWS Systems Manager.

#### **Topics**

- A message reports that the AWS-SystemsManager-PatchSummaryExportRolePolicy policy is corrupted (p. 1183)
- After deleting patch compliance policies or roles, scheduled reports aren't generated successfully (p. 1183)

**A message reports that the AWS-SystemsManager-PatchSummaryExportRolePolicy policy is corrupted**

**Problem:** You receive an error message similar to the following, indicating the AWS-SystemsManager-PatchSummaryExportRolePolicy is corrupted:

An error occurred while updating the AWS-SystemsManager-PatchSummaryExportRolePolicy policy. If you have edited the policy, you might need to delete the policy, and any role that uses it, then try again. Systems Manager recreates the roles and policies you have deleted.

- **Solution:** Perform the following steps:

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Do one of the following:

**On-demand reports** – If the problem occurred while generating a one-time on-demand report, in the left navigation, choose **Policies**, search for AWS-SystemsManager-PatchSummaryExportRolePolicy, then delete the policy. Next, choose **Roles**, search for AWS-SystemsManager-PatchSummaryExportRole, then delete the role.

**Scheduled reports** – If the report occurred while generating a report on a schedule, in the left navigation, choose **Policies**, search one at a time for AWS-EventBridge-Start-SSMAutomationRolePolicy and AWS-SystemsManager-PatchSummaryExportRolePolicy, and delete each policy. Next, choose **Roles**, search one at a time for AWS-EventBridge-Start-SSMAutomationRole and AWS-SystemsManager-PatchSummaryExportRole, and delete each role.

3. Follow the steps to generate or schedule a new patch compliance report.

**After deleting patch compliance policies or roles, scheduled reports aren't generated successfully**

**Problem:** The first time you generate a report, Systems Manager creates a service role and a policy to use for the export process (AWS-SystemsManager-PatchSummaryExportRole and AWS-SystemsManager-PatchSummaryExportRolePolicy). The first time you generate a report on a schedule, Systems Manager creates another service role and a policy (AWS-EventBridge-Start-SSMAutomationRole and AWS-EventBridge-Start-SSMAutomationRolePolicy). These let Amazon EventBridge start an automation using the runbook [AWS-ExportPatchReportToS3](#).

If you delete any of these policies or roles, the connections between your schedule and your specified S3 bucket and Amazon SNS topic might be lost.

- **Solution:** To work around this problem, we recommend deleting the previous schedule and creating a new schedule to replace the one that was experiencing issues.

## Remediating out-of-compliance managed nodes with Patch Manager

The topics in this section provide overviews of how to identify managed nodes that are out of patch compliance and how to bring nodes into compliance.

### Topics

- [Identifying out-of-compliance managed nodes \(p. 1184\)](#)
- [Understanding patch compliance state values \(p. 1186\)](#)
- [Patching out-of-compliance managed nodes \(p. 1189\)](#)

### Identifying out-of-compliance managed nodes

Out-of-compliance managed nodes are identified when either of two AWS Systems Manager documents (SSM documents) are run. These SSM documents reference the appropriate patch baseline for each managed node in Patch Manager, a capability of AWS Systems Manager. They then evaluate the patch state of the managed node and then make compliance results available to you.

There are two SSM documents that are used to identify or update out-of-compliance managed nodes: `AWS-RunPatchBaseline` and `AWS-RunPatchBaselineAssociation`. Each one is used by different processes, and their compliance results are available through different channels. The following table outlines the differences between these documents.

#### Note

Patch compliance data from Patch Manager can be sent to AWS Security Hub. Security Hub gives you a comprehensive view of your high-priority security alerts and compliance status. It also monitors the patching status of your fleet. For more information, see [Integrating Patch Manager with AWS Security Hub \(p. 1207\)](#).

	<b>AWS-RunPatchBaseline</b>	<b>AWS-RunPatchBaselineAssociation</b>
Processes that use the document	<p><b>Patch on demand</b> – You can scan or patch managed nodes on demand using the <b>Patch now</b> option. For information, see <a href="#">Patching managed nodes on demand (console) (p. 1190)</a>.</p> <p><b>Patch Manager patching configuration</b> – You can create a patching configuration that includes a maintenance window to scan managed nodes for patch compliance on a schedule. For information, see <a href="#">Creating a patching configuration (console) (p. 1203)</a>.</p> <p><b>Run a command</b> – You can manually run <code>AWS-RunPatchBaseline</code> in an operation in Run Command, a capability of AWS Systems Manager. For information, see <a href="#">Running commands from the console (p. 996)</a>.</p> <p><b>Maintenance window</b> – You can create a maintenance window that uses the SSM document <code>AWS-RunPatchBaseline</code> in a Run Command task type. For information, see <a href="#">Walkthrough:</a></p>	<p><b>Systems Manager Quick Setup (p. 145)</b> – You can configure Quick Setup, a capability of AWS Systems Manager, to use Patch Manager to scan your managed instances for patch compliance each day. For information, see <a href="#">Quick Setup Host Management (p. 150)</a> in the topic <a href="#">AWS Systems Manager Quick Setup (p. 145)</a>.</p> <p><b>Systems Manager Explorer (p. 157)</b> – When you allow Explorer, a capability of AWS Systems Manager, it regularly scans your managed instances for patch compliance and reports results in the Explorer dashboard.</p>

	<b>AWS–RunPatchBaseline</b>	<b>AWS–RunPatchBaselineAssociation</b>
	<a href="#">Creating a maintenance window for patching (console) (p. 787)</a> .	
Format of the patch scan result data	After AWS–RunPatchBaseline runs, Patch Manager sends an AWS : PatchSummary object to Inventory, a capability of AWS Systems Manager.	After AWS–RunPatchBaselineAssociation runs, Patch Manager sends an AWS : ComplianceItem object to Systems Manager Inventory.
Viewing patch compliance reports in the console	You can view patch compliance information for processes that use AWS–RunPatchBaseline in <a href="#">Systems Manager Configuration Compliance (p. 836)</a> and <a href="#">Managed nodes (p. 804)</a> . For more information, see <a href="#">Viewing patch compliance results (console) (p. 1175)</a> .	If you use Quick Setup to scan your managed instances for patch compliance, you can see the compliance report in <a href="#">Systems Manager State Manager (p. 1034)</a> , which is accessible using a <b>View results</b> button in Quick Setup.  If you use Explorer to scan your managed instances for patch compliance, you can see the compliance report in both Explorer and <a href="#">Systems Manager OpsCenter (p. 180)</a> .
AWS CLI commands for viewing patch compliance results	For processes that use AWS–RunPatchBaseline, you can use the following AWS CLI commands to view summary information about patches on a managed node. <ul style="list-style-type: none"> <li>• <a href="#">describe-instance-patch-states</a></li> <li>• <a href="#">describe-instance-patch-states-for-patch-group</a></li> <li>• <a href="#">describe-patch-group-state</a></li> </ul>	For processes that use AWS–RunPatchBaselineAssociation, you can use the following AWS CLI command to view summary information about patches on an instance. <ul style="list-style-type: none"> <li>• <a href="#">list-compliance-items</a></li> </ul>
Patching operations	For processes that use AWS–RunPatchBaseline, you specify whether you want the operation to run a Scan operation only, or a Scan and install operation.  If your goal is to identify out-of-compliance managed nodes and not remediate them, run only a Scan operation.	Quick Setup and Explorer processes, which use AWS–RunPatchBaselineAssociation, run only a Scan operation.
More information	<a href="#">About the AWS–RunPatchBaseline SSM document (p. 1128)</a>	<a href="#">About the AWS–RunPatchBaselineAssociation SSM document (p. 1137)</a>

For information about the various patch compliance states you might see reported, see [Understanding patch compliance state values \(p. 1186\)](#)

For information about remediating managed nodes that are out of patch compliance, see [Patching out-of-compliance managed nodes \(p. 1189\)](#).

### Understanding patch compliance state values

The information about patches for a managed node include a report of the state, or status, of each individual patch.

#### Note

If you want to assign a specific patch compliance state to a managed node, you can use the [put-compliance-items](#) AWS Command Line Interface (AWS CLI) command or the [PutComplianceItems](#) API operation. Assigning compliance state isn't supported in the console.

Use the information in the following tables to help you identify why a managed node might be out of patch compliance.

### Patch compliance values for Debian Server, Raspberry Pi OS, and Ubuntu Server

For Debian Server, Raspberry Pi OS, and Ubuntu Server, the rules for package classification into the different compliance states are described in the following table.

#### Note

Keep the following in mind when you're evaluating the **Installed**, **Installed Other**, and **Missing** status values: If you don't select the **Include nonsecurity updates** check box when creating or updating a patch baseline, patch candidate versions are limited to patches included in **trusty-security** (Ubuntu Server 14.04 LTS), **xenial-security** (Ubuntu Server 16.04 LTS), **bionic-security** (Ubuntu Server 18.04 LTS), **focal-security** (Ubuntu Server 20.04 LTS), **groovy-gorilla** (Ubuntu Server 20.10 STR), or **debian-security** (Debian Server and Raspberry Pi OS). If you do select the **Include nonsecurity updates** check box, patches from other repositories are considered as well.

Patch state	Description	Compliance status
<b>Installed</b>	The patch is listed in the patch baseline and is installed on the managed node. It could have been installed either manually by an individual or automatically by Patch Manager when the <a href="#">AWS-RunPatchBaseline</a> document was run on the managed node.	Compliant
<b>Installed Other</b>	The patch isn't included in the baseline or isn't approved by the baseline but is installed on the managed node. The patch might have been installed manually, the package could be a required dependency of another approved patch, or the patch might have been included in an <a href="#">InstallOverrideList</a> operation. If you don't specify <b>Block</b> as the <b>Rejected patches</b> action, <b>Installed_Other</b>	Compliant

Patch state	Description	Compliance status
	patches also includes installed but rejected patches.	
<b>Installed Pending Reboot</b>	The Patch Manager <code>Install</code> operation applied the patch to the managed node, but the node has not been rebooted since the patch was applied. (Note that patches installed outside of Patch Manager are never given a status of <code>INSTALLED_PENDING_REBOOT</code> .) This typically means the <code>NoReboot</code> option was selected for the <code>RebootOption</code> parameter when the <code>AWS-RunPatchBaseline</code> document was last run on the managed node. For more information, see <a href="#">Parameter name: RebootOption (p. 1136)</a> .	Non-Compliant
<b>Installed Rejected</b>	The patch is installed on the managed node but is specified in a <b>Rejected patches</b> list. This typically means the patch was installed before it was added to a list of rejected patches.	Non-Compliant
<b>Missing</b>	Packages that are filtered through the baseline and not already installed.	Non-Compliant
<b>Failed</b>	Packages that failed to install during the patch operation.	Non-Compliant

#### Patch compliance values for other operating systems

For all operating systems besides Debian Server, Raspberry Pi OS, and Ubuntu Server, the rules for package classification into the different compliance states are described in the following table.

Patch state	Description	Compliance value
<b>INSTALLED</b>	The patch is listed in the patch baseline and is installed on the managed node. It could have been installed either manually by an individual or automatically by Patch Manager when the <code>AWS-RunPatchBaseline</code> document was run on the node.	Compliant
<b>INSTALLED_OTHER</b>	The patch isn't in the baseline, but it is installed on the managed node. The patch	Compliant

Patch state	Description	Compliance value
	might have been installed manually, or the package could be a required dependency of another approved patch. If you don't specify Block as the <b>Rejected patches</b> action, <code>Installed_Other</code> patches also includes installed but rejected patches.	
<b>INSTALLED_REJECTED</b>	The patch is installed on the managed node 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.	Non-Compliant
<b>INSTALLED_PENDING_REBOOT</b>	The Patch Manager <code>Install</code> operation applied the patch to the managed node (or a patch was applied to a Windows Server managed node outside of Patch Manager), but the node hasn't been rebooted since the patch was applied. (Note that patches installed outside of Patch Manager are never given a status of <code>INSTALLED_PENDING_REBOOT</code> .) This typically means the <code>NoReboot</code> option was selected for the <code>RebootOption</code> parameter when the <code>AWS-RunPatchBaseline</code> document was last run on the managed node. For more information, see <a href="#">Parameter name: RebootOption (p. 1136)</a> .	Non-Compliant
<b>MISSING</b>	The patch is approved in the baseline, but it isn't installed on the managed node. If you configure the <code>AWS-RunPatchBaseline</code> document task to scan (instead of install), the system reports this status for patches that were located during the scan but haven't been installed.	Non-Compliant

Patch state	Description	Compliance value
<b>NOT_APPLICABLE</b>	<p>The patch is approved in the baseline, but the service or feature that uses the patch isn't installed on the managed node. For example, a patch for a web server service such as Internet Information Services (IIS) would show <b>NOT_APPLICABLE</b> if it was approved in the baseline, but the web service isn't installed on the managed node. A patch can also be marked <b>NOT_APPLICABLE</b> if it has been superseded by a subsequent update. This means that the later update is installed and the <b>NOT_APPLICABLE</b> update is no longer required.</p> <p><b>Note</b> This compliance state is only reported on Windows Server operating systems.</p>	Not applicable
<b>FAILED</b>	<p>The patch is approved in the baseline, but it couldn't be installed. To troubleshoot this situation, review the command output for information that might help you understand the problem.</p>	Non-Compliant

## Patching out-of-compliance managed nodes

Many of the same AWS Systems Manager tools and processes you can use to check managed nodes for patch compliance can be used to bring nodes into compliance with the patch rules that currently apply to them. To bring managed nodes into patch compliance, Patch Manager, a capability of AWS Systems Manager, must run a `Scan` and `install` operation. (If your goal is only to identify out-of-compliance managed nodes and not remediate them, run a `Scan` operation instead. For more information, see [Identifying out-of-compliance managed nodes \(p. 1184\)](#).)

### Install patches using Systems Manager

You can choose from several tools to run a `Scan` and `install` operation:

- Create a patching configuration that includes a maintenance window to install patches on a schedule. For information, see [Creating a patching configuration \(console\) \(p. 1203\)](#).
- Create a maintenance window that uses the Systems Manager document (SSM document) `AWS-RunPatchBaseline` in a Run Command task type. For information, see [Walkthrough: Creating a maintenance window for patching \(console\) \(p. 787\)](#).
- Manually run `AWS-RunPatchBaseline` in a Run Command operation. For information, see [Running commands from the console \(p. 996\)](#).
- Install patches on demand using the **Patch now** option. For information, see [Patching managed nodes on demand \(console\) \(p. 1190\)](#).

## Patching managed nodes on demand (console)

Using the **Patch now** option in Patch Manager, a capability of AWS Systems Manager, you can run on-demand patching operations from the Systems Manager console. This means you don't have to create a schedule in order to update the compliance status of your managed nodes or to install patches on noncompliant nodes. You also don't need to switch the Systems Manager console between Patch Manager and Maintenance Windows, a capability of AWS Systems Manager, in order to set up or modify a scheduled patching window.

**Patch now** is especially useful when you must apply zero-day updates or install other critical patches on your managed nodes as soon as possible.

### Topics

- [How 'Patch now' works \(p. 1190\)](#)
- [Running 'Patch now' \(p. 1192\)](#)

### How 'Patch now' works

To run **Patch now**, you specify just two required settings:

- Whether to scan for missing patches only, or to scan *and* install patches on your managed nodes
- Which managed nodes to run the operation on

When the **Patch now** operation runs, it determines which patch baseline to use in the same way one is selected for other patching operations. If a managed node is associated with a patch group, the patch baseline specified for that group is used. If the managed node isn't associated with a patch group, the operation uses the patch baseline that is currently set as the default for the operating system type of the managed node. This can be a predefined baseline, or the custom baseline you have set as the default. For more information about patch baseline selection, see [About patch groups \(p. 1163\)](#).

Options you can specify for **Patch now** include choosing when, or whether, to reboot managed nodes after patching, specifying an Amazon Simple Storage Service (Amazon S3) bucket to store log data for the patching operation, and running Systems Manager documents (SSM documents) as lifecycle hooks during patching.

### Concurrency and error thresholds for 'Patch now'

For **Patch now** operations, concurrency and error threshold options are handled by Patch Manager. You don't need to specify how many managed nodes to patch at once, nor how many errors are permitted before the operation fails. Patch Manager applies the concurrency and error threshold settings described in the following tables when you patch on demand.

#### Important

The following thresholds apply to `Scan and install` operations only. For `Scan` operations, Patch Manager attempts to scan up to 1,000 nodes concurrently, and continue scanning until it has encountered up to 1,000 errors.

### Concurrency: Install operations

Total number of managed nodes in the Patch now operation	Number of managed nodes scanned or patched at a time
Fewer than 25	1
25-100	5%

Total number of managed nodes in the Patch now operation	Number of managed nodes scanned or patched at a time
101 to 1,000	8%
More than 1,000	10%

### Error threshold: Install operations

Total number of managed nodes in the Patch now operation	Number of errors permitted before the operation fails
Fewer than 25	1
25-100	5
101 to 1,000	10
More than 1,000	10

### Using 'Patch now' lifecycle hooks

**Patch now** provides you with the ability to run SSM Command documents as lifecycle hooks during an Install patching operation. You can use these hooks for tasks such as shutting down applications before patching or running health checks on your applications after patching or after a reboot.

For more information about using lifecycle hooks, see [About the AWS-RunPatchBaselineWithHooks SSM document \(p. 1146\)](#).

The following table lists the lifecycle hooks available for each of the three **Patch now** reboot options, in addition to sample uses for each hook.

### Lifecycle hooks and sample uses

Reboot option	Hook: Before installation	Hook: After installation	Hook: On exit	Hook: After scheduled reboot
<b>Reboot if needed</b>	Run an SSM document before patching begins.  Example use: Safely shut down applications before the patching process begins.	Run an SSM document at the end of the patching operation and before managed node reboot.  Example use: Run operations such as installing third-party applications before a potential reboot.	Run an SSM document immediately after the patching operation completes.  Example use: Ensure that applications are running as expected after patching.	<i>Not available</i>
<b>Do not reboot my instances</b>	Same as above.	Run an SSM document at the end of the patching operation.	<i>Not available</i>	<i>Not available</i>

Reboot option	Hook: Before installation	Hook: After installation	Hook: On exit	Hook: After scheduled reboot
		Example use: Ensure that applications are running as expected after patching.		
<b>Schedule a reboot time</b>	Same as above.	Same as for <b>Do not reboot my instances</b> .	<i>Not available</i>	Run an SSM document immediately after a scheduled reboot completes.  Example use: Ensure that applications are running as expected after the reboot.

## Running 'Patch now'

Use the following procedure to patch your managed nodes on demand.

### To run 'Patch now'

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Patch Manager**.
3. On either the **AWS Systems Manager Patch Manager** page or the **Patch baselines** page, depending on which one opens, choose **Patch now**.
4. For **Patching operation**, choose one of the following:
  - **Scan**: Patch Manager finds which patches are missing from your managed nodes but doesn't install them. You can view the results in the **Compliance** dashboard or in other tools you use for viewing patch compliance.
  - **Scan and install**: Patch Manager finds which patches are missing from your managed nodes and installs them.
5. Use this step only if you chose **Scan and install** in the previous step. For **Reboot option**, choose one of the following:
  - **Reboot if needed**: After installation, Patch Manager reboots managed nodes only if needed to complete a patch installation.
  - **Don't reboot my instances**: After installation, Patch Manager doesn't reboot managed nodes. You can reboot nodes manually when you choose or manage reboots outside of Patch Manager.
  - **Schedule a reboot time**: Specify the date, time, and UTC time zone for Patch Manager to reboot your managed nodes. After you run the **Patch now** operation, the scheduled reboot is listed as an association in State Manager with the name `AWS-PatchRebootAssociation`.
6. For **Instances to patch**, choose one of the following:
  - **Patch all instances**: Patch Manager runs the specified operation on all managed nodes in your AWS account in the current AWS Region.

- **Patch only the target instances I specify:** You specify which managed nodes to target in the next step.
7. Use this step only if you chose **Patch only the target instances I specify** in the previous step. In the **Target selection** section, identify the nodes on which you want to run this operation by specifying tags, selecting nodes manually, or specifying a resource group.
- Note**  
If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.  
If you choose to target a resource group, note that resource groups that are based on an AWS CloudFormation stack must still be tagged with the default `aws:cloudformation:stack-id` tag. If it has been removed, Patch Manager might be unable to determine which managed nodes belong to the resource group.
8. (Optional) For **Patching log storage**, if you want to create and save logs from this patching operation, select the S3 bucket for storing the logs.
9. (Optional) If you want to run SSM documents as lifecycle hooks during specific points of the patching operation, do the following:

- Choose **Use lifecycle hooks**.
- For each available hook, select the SSM document to run at the specified point of the operation:
  - Before installation
  - After installation
  - On exit
  - After scheduled reboot

**Note**

The default document, `AWS-Noop`, runs no operations.

10. Choose **Patch now**.

The **Association execution summary** page opens. (Patch now uses associations in State Manager, a capability of AWS Systems Manager, for its operations.) In the **Operation summary** area, you can monitor the status of scanning or patching on the managed nodes you specified.

## Working with patch baselines

A patch baseline in Patch Manager, a capability of AWS Systems Manager, defines which patches are approved for installation on your managed nodes. 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 won't be approved or installed, even if it matches all of the criteria in an auto-approval rule. Also, a patch is installed on a managed node only if it applies to the software on the node, even if the patch has otherwise been approved for the managed node.

### Topics

- [Viewing AWS predefined patch baselines \(console\) \(p. 1194\)](#)
- [Working with custom patch baselines \(console\) \(p. 1194\)](#)
- [Setting an existing patch baseline as the default \(console\) \(p. 1202\)](#)

### Related content

- [About patch baselines \(p. 1156\)](#)

## Viewing AWS predefined patch baselines (console)

Patch Manager, a capability of AWS Systems 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 \(p. 1157\)](#).

### To view AWS predefined patch baselines

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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, three predefined patch baselines are provided. The patch baselines `AWS-DefaultPatchBaseline` and `AWS-WindowsPredefinedPatchBaseline-OS` support only operating system updates on the Windows operating system itself. `AWS-DefaultPatchBaseline` is used as the default patch baseline for Windows Server managed nodes unless you specify a different patch baseline. The configuration settings in these two patch baselines are the same. The newer of the two, `AWS-WindowsPredefinedPatchBaseline-OS`, was created to distinguish it from the third predefined patch baseline for Windows Server. That patch baseline, `AWS-WindowsPredefinedPatchBaseline-OS-Applications`, can be used to apply patches to both the Windows Server operating system and supported applications released by Microsoft.

For more information, see [Setting an existing patch baseline as the default \(console\) \(p. 1202\)](#).

4. Choose the **Approval rules** tab and review the patch baseline configuration.
5. If the configuration is acceptable for your managed nodes, you can skip ahead to the procedure [Working with patch groups \(p. 1205\)](#).

-or-

To create your own default patch baseline, continue to the topic [Working with custom patch baselines \(console\) \(p. 1194\)](#).

## Working with custom patch baselines (console)

Patch Manager, a capability of AWS Systems 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, update, and delete your own custom patch baselines. To learn more about patch baselines, see [About predefined and custom patch baselines \(p. 1157\)](#).

### Topics

- [Creating a custom patch baseline \(Linux\) \(p. 1195\)](#)
- [Creating a custom patch baseline \(macOS\) \(p. 1198\)](#)
- [Creating a custom patch baseline \(Windows\) \(p. 1199\)](#)
- [Updating or deleting a custom patch baseline \(console\) \(p. 1202\)](#)

## Creating a custom patch baseline (Linux)

Use the following procedure to create a custom patch baseline for Linux managed nodes in Patch Manager, a capability of AWS Systems Manager.

For information about creating a patch baseline for macOS managed nodes, see [Creating a custom patch baseline \(macOS\) \(p. 1198\)](#). For information about creating a patch baseline for Windows managed nodes, see [Creating a custom patch baseline \(Windows\) \(p. 1199\)](#).

### To create a custom patch baseline for Linux managed nodes

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Patch baselines** tab.
4. Choose **Create patch baseline**.
5. For **Name**, enter a name for your new patch baseline, for example, **MyRHELPatchBaseline**.
6. (Optional) For **Description**, enter a description for this patch baseline.
7. For **Operating system**, choose an operating system, for example, Red Hat Enterprise Linux.
8. 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 [Setting an existing patch baseline as the default \(console\) \(p. 1202\)](#).

9. 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 aren't 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.\* managed nodes. (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 nodes, including minor version upgrades, by using the [InstallOverrideList \(p. 1132\)](#) parameter in the `AWS-RunPatchBaseline` document. For more information, see [About the AWS-RunPatchBaseline SSM document \(p. 1128\)](#).
- **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 aren't 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 360. For most scenarios, we recommend waiting no more than 100 days.
- **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 managed nodes. For more information, see the content for SLES in [How security patches are selected \(p. 1097\)](#).

For more information about working with approval rules in a custom patch baseline, see [About custom baselines \(p. 1159\)](#).

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 \(p. 1161\)](#).

- (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** check box for these patches to be installed on your Linux operating system as well.

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 \(p. 1161\)](#).

- 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's a dependency of another package. It's 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, aren't installed under any circumstances. If a package was installed before it was added to the **Rejected patches** list, it's considered noncompliant with the patch baseline and its status is reported as *InstalledRejected*.

12. (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 in the following format:

```
[main]
name=MyCustomRepository
baseurl=https://my-custom-repository
enabled=1
```

**Tip**

For information about other options available for your yum repository configuration, see [dnf.conf\(5\)](#).

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 \(Linux\) \(p. 1101\)](#).

13. (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 allow 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*

14. Choose **Create patch baseline**.

## Creating a custom patch baseline (macOS)

Use the following procedure to create a custom patch baseline for macOS managed nodes in Patch Manager, a capability of AWS Systems Manager.

For information about creating a patch baseline for Windows Server managed nodes, see [Creating a custom patch baseline \(Windows\) \(p. 1199\)](#). For information about creating a patch baseline for Linux managed nodes, see [Creating a custom patch baseline \(Linux\) \(p. 1195\)](#).

### To create a custom patch baseline for macOS managed nodes

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Patch baselines** tab.
4. Choose **Create patch baseline**.
5. For **Name**, enter a name for your new patch baseline, for example, **MymacOSPatchBaseline**.
6. (Optional) For **Description**, enter a description for this patch baseline.
7. For **Operating system**, choose macOS.
8. If you want to begin using this patch baseline as the default for macOS as soon as you create it, check the box next to **Set this patch baseline as the default patch baseline for macOS instances**.

For information about setting an existing patch baseline as the default, see [Setting an existing patch baseline as the default \(console\) \(p. 1202\)](#).

9. 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 Mojave10.14.1 or Catalina10.15.1. The default selection is All.
  - **Classification:** The package manager or package managers that you want to apply packages during the patching process. You can choose from the following:
    - softwareupdate
    - installer
    - brew
    - brew cask

The default selection is All.

- (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 operating system patches made available in the source repository, in addition to the security-related patches.

For more information about working with approval rules in a custom patch baseline, see [About custom baselines \(p. 1159\)](#).

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 \(p. 1161\)](#).
  - (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** check box for these patches to be installed on your macOS operating system as well.
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 \(p. 1161\)](#).
  - 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's a dependency of another package. It's 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, aren't installed under any circumstances. If a package was installed before it was added to the **Rejected patches** list, it's 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 allow 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 package manager 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=PackageManager,Value=softwareupdate
- Key=Environment,Value=Production

13. Choose **Create patch baseline**.

### [Creating a custom patch baseline \(Windows\)](#)

Use the following procedure to create a custom patch baseline for Windows managed nodes in Patch Manager, a capability of AWS Systems Manager.

For information about creating a patch baseline for Linux managed nodes, see [Creating a custom patch baseline \(Linux\) \(p. 1195\)](#). For information about creating a patch baseline for macOS managed nodes, see [Creating a custom patch baseline \(macOS\) \(p. 1198\)](#).

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 \(console\) \(p. 1235\)](#).

## To create a custom patch baseline (Windows)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Patch baselines** tab.
4. Choose **Create patch baseline**.
5. For **Name**, enter a name for your new patch baseline, for example, **MyWindowsPatchBaseline**.
6. (Optional) For **Description**, enter a description for this patch baseline.
7. For **Operating system**, choose Windows.
8. 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 [Setting an existing patch baseline as the default \(console\) \(p. 1202\)](#).

9. 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 WindowsServer2012. 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 \(console\) \(p. 1235\)](#).

- **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 360. For most scenarios, we recommend waiting no more than 100 days.
  - **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. (Optional) In the **Approval rules for applications** section, use the fields to create one or more auto-approval rules.

### Note

Instead of specifying approval rules, you can specify lists of approved and rejected patches as patch exceptions. See steps 10 and 11.

- **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 360. For most scenarios, we recommend waiting no more than 100 days.
- **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.

11. (Optional) If you want to explicitly approve any patches instead of letting patches be selected according to 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 \(p. 1161\)](#).

- (Optional) For **Approved patches compliance level**, assign a compliance level to the patches in the list.

12. 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 \(p. 1161\)](#).

- 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's a dependency of another package. It's 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, aren't installed under any circumstances. If a package was installed before it was added to the **Rejected patches** list, it's considered noncompliant with the patch baseline and its status is reported as *InstalledRejected*.

13. (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 allow 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

14. Choose **Create patch baseline**.

### Updating or deleting a custom patch baseline (console)

You can update or delete a custom patch baseline that you have created in Patch Manager, a capability of AWS Systems Manager. 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.

### Updating or deleting a patch baseline (console)

Follow these steps to update or delete a patch baseline.

#### To update or delete a patch baseline (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 patch baseline that you want to update or delete, and then do one of the following:
  - To remove the patch baseline from your AWS 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 [Working with custom patch baselines \(console\) \(p. 1194\)](#).

### Setting an existing patch baseline as the default (console)

When you create a custom patch baseline in Patch Manager, a capability of AWS Systems Manager, you can set the baseline as the default for the associated operating system type as soon as you create it. For information, see [Working with custom patch baselines \(console\) \(p. 1194\)](#).

You can also set an existing patch baseline as the default for an operating system type.

### To set a patch baseline as the default

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 isn't 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**.

## Viewing available patches (console)

With Patch Manager, a capability of AWS Systems Manager, you can view all available patches for a specified operating system and, optionally, a specific operating system version.

**Tip**

To generate a list of available patches and save them to a file, you can use the `describe-available-patches` command and specify your preferred `output`.

### To view available patches

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Patches** tab.
4. For **Operating system**, choose the operating system for which you want to view available patches, such as Windows or Amazon Linux.
5. (Optional) For **Product**, choose an OS version, such as WindowsServer2019 or AmazonLinux2018.03.
6. (Optional) To add or remove information columns for your results, choose the configure button (⚙) at the top right of the **Patches** list. (By default, the **Patches** tab displays columns for only some of the available patch metadata.)

For information about the types of metadata you can add to your view, see [Patch](#) in the [AWS Systems Manager API Reference](#).

## Creating a patching configuration (console)

A patching configuration defines a unique patching operation. The configuration specifies the managed nodes 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.

In a patching configuration, 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 managed nodes.

**Note**

Many patching use cases benefit from patching managed nodes on a schedule with a maintenance window, but you can also run a one-time patching operation manually without a maintenance window. For more information, see [Patching managed nodes on demand \(console\) \(p. 1190\)](#).

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 \(p. 706\)](#).

If you plan to add the patching configuration to a maintenance window, you must first configure roles and permissions for Maintenance Windows, a capability of AWS Systems Manager, before beginning this procedure. For more information, see [Setting up Maintenance Windows \(p. 707\)](#).

### To create a patching configuration (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 managed node to patch. Select **Add** to include additional tagged managed nodes.
  - **Select a patch group:** Choose the name of an existing patch group that includes the managed nodes 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](#) AWS Command Line Interface (AWS 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 isn't registered with the patch baseline, choose **Enter instance tag**, enter **Patch Group** as the tag key and the patch group's name as the tag value.

5. • **Select instances manually:** Select the check box next to the name of each managed node you want to patch.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

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're specifying a CRON/Rate expression, see [Reference: Cron and rate expressions for Systems Manager \(p. 1540\)](#) 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 managed nodes 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 managed nodes 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 managed nodes you selected that aren't part of a patch group, Patch Manager instead uses the default patch baseline for the operating system type of the managed node.

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 [Updating or deleting maintenance window resources \(console\) \(p. 730\)](#).

## Working with patch groups

To help you organize your patching efforts, we recommend that you add managed nodes 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 \(p. 1163\)](#).

After you group your managed nodes 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 \(p. 1206\)](#)
- [Task 2: Add managed nodes to a patch group using tags \(p. 1206\)](#)
- [Task 3: Add a patch group to a patch baseline \(p. 1207\)](#)

## Task 1: Add EC2 instances to a patch group using tags

For Amazon Elastic Compute Cloud (Amazon EC2) instances, you can add tags by using the AWS Systems Manager console, the Amazon EC2 console, the AWS Command Line Interface (AWS CLI) command `create-tags`, or the API operation `CreateTags`.

### Important

To apply the `Patch Group` tag to an Amazon EC2 instance, the **Allow tags in instance metadata** option must not be enabled on the instance. Allowing tags in instance metadata prevents tag key names from containing spaces. For information about disabling the setting if you have enabled it, see [Turn off access to tags in instance metadata](#) in the *Amazon EC2 User Guide for Linux Instances*.

### To add EC2 instances to a patch group (AWS Systems Manager console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. In the **Managed instances** list, choose the ID of a managed EC2 instance that you want to configure for patching.

### 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.

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

4. Select the **Tags** tab, then choose **Edit**.
5. In the left column, enter **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**, enter **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.

## Task 2: Add managed nodes to a patch group using tags

For AWS IoT Greengrass core devices and hybrid managed nodes (mi-\*), you can add tags by using the Systems Manager console, the AWS CLI command `add-tags-to-resource`, or the API operation `AddTagsToResource`. You can't add tags for hybrid managed node using the Amazon EC2 console.

### To add managed nodes to a patch group (Systems Manager console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
  2. In the navigation pane, choose **Fleet Manager**.
- or-
- If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.
3. In the **Managed instances** list, choose a managed node that you want to configure for patching.
- Note**  
If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.
4. Choose **View details**.
  5. Select the **Tags** tab, then choose **Edit**.
  6. In the left column, enter **Patch Group**.
  7. In the right column, enter a value that helps you understand which managed nodes will be patched.
  8. Choose **Save**.
  9. Repeat this procedure to add other managed nodes to the same patch group.

### Task 3: Add a patch group to a patch baseline

To associate a specific patch baseline with your managed nodes, 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 \(p. 1163\)](#).

### To add a patch group to a patch baseline (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 nodes in the previous section, then choose **Add**.

## Working with Patch Manager settings

### Topics

- [Integrating Patch Manager with AWS Security Hub \(p. 1207\)](#)

### Integrating Patch Manager with AWS Security Hub

[AWS Security Hub](#) provides you with a comprehensive view of your security state in AWS. Security Hub collects security data from across AWS accounts, AWS services, and supported third-party partner products. With Security Hub, you can check your environment against security industry standards and best practices. Security Hub helps you to analyze your security trends and identify the highest priority security issues.

By using the integration between Patch Manager, a capability of AWS Systems Manager, and Security Hub, you can send findings from Patch Manager to Security Hub. A finding is the observable record of a security check or security-related detection. Security Hub can then include those findings in its analysis of your security posture.

## Contents

- [How Patch Manager sends findings to Security Hub \(p. 1208\)](#)
  - [Types of findings that Patch Manager sends \(p. 1208\)](#)
  - [Latency for sending findings \(p. 1208\)](#)
  - [Retrying when Security Hub isn't available \(p. 1209\)](#)
  - [Updating existing findings in Security Hub \(p. 1209\)](#)
- [Typical finding from Patch Manager \(p. 1209\)](#)
- [Turning on and configuring the integration \(p. 1210\)](#)
- [How to stop sending findings \(p. 1210\)](#)

### [How Patch Manager sends findings to Security Hub](#)

In Security Hub, security issues are tracked as findings. Some findings come from issues that are detected by other AWS services or by third-party partners. Security Hub also has a set of rules that it uses to detect security issues and generate findings.

Patch Manager is one of the Systems Manager capabilities that sends findings to Security Hub. After you perform a patching operation by running a SSM document (`AWS-RunPatchBaseline`, `AWS-RunPatchBaselineAssociation`, or `AWS-RunPatchBaselineWithHooks`), the patching information is sent to Inventory or Compliance, capabilities of AWS Systems Manager, or both. After Inventory, Compliance, or both receive the data, Patch Manager receives a notification. Then, Patch Manager evaluates the data for accuracy, formatting, and compliance. If all conditions are met, Patch Manager forwards the data to Security Hub.

Security Hub provides tools to manage findings from across all of these sources. You can view and filter lists of findings and view details for a finding. For more information, see [Viewing findings](#) in the *AWS Security Hub User Guide*. You can also track the status of an investigation into a finding. For more information, see [Taking action on findings](#) in the *AWS Security Hub User Guide*.

All findings in Security Hub use a standard JSON format called the AWS Security Finding Format (ASFF). The ASFF includes details about the source of the issue, the affected resources, and the current status of the finding. For more information, see [AWS Security Finding Format \(ASFF\)](#) in the *AWS Security Hub User Guide*.

### [Types of findings that Patch Manager sends](#)

Patch Manager sends the findings to Security Hub using the [AWS Security Finding Format \(ASFF\)](#). In ASFF, the `Types` field provides the finding type. Findings from Patch Manager have the following value for `Types`:

- Software and Configuration Checks/Patch Management

Patch Manager sends one finding per noncompliant managed node. The finding is reported with the resource type `AwsEc2Instance` so that findings can be correlated with other Security Hub integrations that report `AwsEc2Instance` resource types. Patch Manager only forwards a finding to Security Hub if the operation discovered the managed node to be noncompliant. The finding includes the Patch Summary results. For more information about compliance definitions, see [Understanding patch compliance state values \(p. 1186\)](#). For more information about `PatchSummary`, see [PatchSummary](#) in the *AWS Security Hub API Reference*.

### [Latency for sending findings](#)

When Patch Manager creates a new finding, it's usually sent to Security Hub within a few seconds to 2 hours. The speed depends on the traffic in the AWS Region being processed at that time.

## Retrying when Security Hub isn't available

If there is a service outage, an AWS Lambda function is run to put the messages back into the main queue after the service is running again. After the messages are in the main queue, the retry is automatic.

If Security Hub isn't available, Patch Manager retries sending the findings until they're received.

## Updating existing findings in Security Hub

Patch Manager doesn't update a finding after it sends the finding to Security Hub.

Any additional patching operations on `AwsEc2Instance` resource types before the managed node has been brought to compliance standards result in new findings being sent to Security Hub.

## Typical finding from Patch Manager

Patch Manager sends findings to Security Hub using the [AWS Security Finding Format \(ASFF\)](#).

Here is an example of a typical finding from Patch Manager.

```
{
 "SchemaVersion": "2018-10-08",
 "Id": "arn:aws:patchmanager:us-east-2:111122223333:instance/i-02573cafefEXAMPLE/document/AWS-RunPatchBaseline/run-command/d710f5bd-04e3-47b4-82f6-df4e0EXAMPLE",
 "ProductArn": "arn:aws:securityhub:us-east-1::product/aws/ssm-patch-manager",
 "GeneratorId": "d710f5bd-04e3-47b4-82f6-df4e0EXAMPLE",
 "AwsAccountId": "111122223333",
 "Types": [
 "Software & Configuration Checks/Patch Management/Compliance"
],
 "CreatedAt": "2021-11-11T22:05:25Z",
 "UpdatedAt": "2021-11-11T22:05:25Z",
 "Severity": {
 "Label": "INFORMATIONAL",
 "Normalized": 0
 },
 "Title": "Systems Manager Patch Summary - Managed Instance Non-Compliant",
 "Description": "This AWS control checks whether each instance that is managed by AWS Systems Manager is in compliance with the rules of the patch baseline that applies to that instance when a compliance Scan runs.",
 "Remediation": {
 "Recommendation": {
 "Text": "For information about bringing instances into patch compliance, see 'Remediating out-of-compliance instances (Patch Manager)'.",
 "Url": "https://docs.aws.amazon.com/systems-manager/latest/userguide/patch-compliance-remediation.html"
 }
 },
 "SourceUrl": "https://us-east-2.console.aws.amazon.com/systems-manager/managed-instances/i-02573cafefEXAMPLE/patch?region=us-east-2",
 "ProductFields": {
 "aws/securityhub/FindingId": "arn:aws:securityhub:us-east-2::product/aws/ssm-patch-manager/arn:aws:patchmanager:us-east-2:111122223333:instance/i-02573cafefEXAMPLE/document/AWS-RunPatchBaseline/run-command/d710f5bd-04e3-47b4-82f6-df4e0EXAMPLE",
 "aws/securityhub/ProductName": "Systems Manager Patch Manager",
 "aws/securityhub/CompanyName": "AWS"
 },
 "Resources": [
 {
 "Type": "AwsEc2Instance",
 "Id": "i-02573cafefEXAMPLE",
 "Partition": "aws",
 "Region": "us-east-2"
 }
]
}
```

```
 },
 "WorkflowState": "NEW",
 "Workflow": {
 "Status": "NEW"
 },
 "RecordState": "ACTIVE",
 "PatchSummary": {
 "Id": "pb-0c10e65780EXAMPLE",
 "InstalledCount": 45,
 "MissingCount": 2,
 "FailedCount": 0,
 "InstalledOtherCount": 396,
 "InstalledRejectedCount": 0,
 "InstalledPendingReboot": 0,
 "OperationStartTime": "2021-11-11T22:05:06Z",
 "OperationEndTime": "2021-11-11T22:05:25Z",
 "RebootOption": "NoReboot",
 "Operation": "SCAN"
 }
 }
```

### Turning on and configuring the integration

To use the Patch Manager integration with Security Hub, you must turn on Security Hub. For information about how to turn on Security Hub, see [Setting up Security Hub](#) in the *AWS Security Hub User Guide*.

The following procedure describes how to integrate Patch Manager and Security Hub when Security Hub is already active but Patch Manager integration is turned off. You only need to complete this procedure if integration was manually turned off.

#### To add Patch Manager to Security Hub integration

1. 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**.

2. Choose the **Settings** tab.
3. Under the **Export to Security Hub** section, to the right of **Patch compliance findings aren't being exported to Security Hub**, choose **Enable**.

#### How to stop sending findings

To stop sending findings to Security Hub, you can use either the Security Hub console or the API.

For more information, see the following topics in the *AWS Security Hub User Guide*:

- [Disabling and enabling the flow of findings from an integration \(console\)](#)
- [Disabling the flow of findings from an integration \(Security Hub API, AWS CLI\)](#)

## Working with Patch Manager (AWS CLI)

The section includes examples of AWS Command Line Interface (AWS CLI) commands that you can use to perform configuration tasks for Patch Manager, a capability of AWS Systems Manager.

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 \(AWS CLI\) \(p. 1238\)](#).

For more information about using the AWS CLI for AWS Systems Manager tasks, see the [AWS Systems Manager section of the AWS CLI Command Reference](#).

### Topics

- [AWS CLI commands for patch baselines \(p. 1211\)](#)
- [AWS CLI commands for patch groups \(p. 1220\)](#)
- [AWS CLI commands for viewing patch summaries and details \(p. 1224\)](#)
- [AWS CLI commands for scanning and patching managed nodes \(p. 1231\)](#)

## AWS CLI commands for patch baselines

### Sample commands for patch baselines

- [Create a patch baseline \(p. 1211\)](#)
- [Create a patch baseline with custom repositories for different OS versions \(p. 1212\)](#)
- [Update a patch baseline \(p. 1213\)](#)
- [Rename a patch baseline \(p. 1214\)](#)
- [Delete a patch baseline \(p. 1215\)](#)
- [List all patch baselines \(p. 1216\)](#)
- [List all AWS-provided patch baselines \(p. 1217\)](#)
- [List my patch baselines \(p. 1217\)](#)
- [Display a patch baseline \(p. 1218\)](#)
- [Get the default patch baseline \(p. 1219\)](#)
- [Set a custom patch baseline as the default \(p. 1219\)](#)
- [Reset an AWS patch baseline as the default \(p. 1219\)](#)
- [Tag a patch baseline \(p. 1220\)](#)
- [List the tags for a patch baseline \(p. 1220\)](#)
- [Remove a tag from a patch baseline \(p. 1220\)](#)

### 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're 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's for a production environment.

#### Linux & macOS

```
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

```
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.

```
{
 "BaselineId": "pb-0c10e65780EXAMPLE"
}
```

## Create a patch baseline with custom repositories for different OS versions

Applies to Linux managed nodes 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 allowed by default on Amazon Linux 2017.09, but it could be adapted to a different source repository that you have configured for a managed node.

### Note

To better demonstrate this more complex command, we're using the `--cli-input-json` option with additional options stored an external JSON file.

1. Create a JSON file with a name like `my-patch-repository.json` and add the following content to it.

```
{
 "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] \nnname=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\nngpgcheck=1 \nngpkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-ga \nenabled=1\nnretries=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-0c10e65780EXAMPLE"
}
```

## 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 \(p. 1161\)](#).

#### Linux & macOS

```
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": [
 {
 "PatchFilterGroup": {
 "PatchFilters": [
 {
 "Values": [
 "Important",
 "Critical"
],
 "Key": "MSRC_SEVERITY"
 },
 {
 "Values": [
 "SecurityUpdates"
],
 "Key": "CLASSIFICATION"
 },
 {
 "Values": [
 "WindowsServer2012R2"
],
 "Key": "PRODUCT"
 }
]
 },
 "ApproveAfterDays": 5
 }
]
},
"ModifiedDate": 1481001494.035,
"CreatedDate": 1480997823.81,
"ApprovedPatches": [
 "KB2124261"
],
"Description": "Windows Server 2012 R2, Important and Critical security updates"
}

```

## Rename a patch baseline

Linux & macOS

```

aws ssm update-patch-baseline \
--baseline-id pb-0c10e65780EXAMPLE \
--name "Windows-Server-2012-R2-Important-and-Critical-Security-Updates"

```

Windows

```

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.

```
{
 "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"
 },
 {
 "BaselineName": "Windows-Server-2012R2",
 "DefaultBaseline": false,
 "BaselineDescription": "Windows Server 2012 R2, Important and Critical security
updates",
 "BaselineId": "pb-0c10e65780EXAMPLE"
 }
]
}
```

Here is another command that lists all patch baselines in an AWS Region.

### Linux & macOS

```
aws ssm describe-patch-baselines \
 --region us-east-2 \
 --filters "Key=OWNER,Values=[All]"
```

### Windows

```
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 & macOS

```
aws ssm describe-patch-baselines \
--region us-east-2 \
--filters "Key=OWNER,Values=[AWS]"
```

Windows

```
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 & macOS

```
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 Amazon Resource Name (ARN). For an AWS-provided patch baseline, you must specify the full ARN. For example, `arn:aws:ssm:us-east-2:075727635805:patchbaseline/pb-0c10e65780EXAMPLE`.

The system returns information like the following.

```
{
 "BaselineId": "pb-0c10e65780EXAMPLE",
 "Name": "Windows-Server-2012R2",
 "PatchGroups": [
 "Web Servers"
],
 "RejectedPatches": [
],
 "GlobalFilters": {
 "PatchFilters": [
]
 },
 "ApprovalRules": {
 "PatchRules": [
 {
 "PatchFilterGroup": {
 "PatchFilters": [
 {
 "Values": [
 "Important",
 "Critical"
],
 "Key": "MSRC_SEVERITY"
 },
 {
 "Values": [
 "SecurityUpdates"
],
 "Key": "CLASSIFICATION"
 },
 {
 "Values": [
 "WindowsServer2012R2"
],
 "Key": "PRODUCT"
 }
]
 },
 "ApproveAfterDays": 5
 }
],
 "ModifiedDate": 1480997823.81,
 "CreatedDate": 1480997823.81,
 "ApprovedPatches": [
],
}
```

```
 "Description":"Windows Server 2012 R2, Important and Critical security updates"
}
```

## 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 & macOS

```
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 & macOS

```
aws ssm register-default-patch-baseline \
--region us-east-2 \
--baseline-id "arn:aws:ssm:us-east-2:123456789012:patchbaseline/
pb-0c10e65780EXAMPLE"
```

Windows

```
aws ssm register-default-patch-baseline ^
--region us-east-2 ^
--baseline-id "arn:aws:ssm:us-east-2:123456789012:patchbaseline/
pb-0c10e65780EXAMPLE"
```

The system returns information like the following.

```
{
 "BaselineId": "pb-0c10e65780EXAMPLE"
}
```

## Tag a patch baseline

Linux & macOS

```
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 & macOS

```
aws ssm list-tags-for-resource \
--resource-type "PatchBaseline" \
--resource-id "pb-0c10e65780EXAMPLE"
```

Windows

```
aws ssm list-tags-for-resource ^
--resource-type "PatchBaseline" ^
--resource-id "pb-0c10e65780EXAMPLE"
```

## Remove a tag from a patch baseline

Linux & macOS

```
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"
```

## AWS CLI commands for patch groups

### Sample commands for patch groups

- [Create a patch group \(p. 1221\)](#)
- [Register a patch group "web servers" with a patch baseline \(p. 1222\)](#)
- [Register a patch group "Backend" with the AWS-provided patch baseline \(p. 1222\)](#)
- [Display patch group registrations \(p. 1223\)](#)

- [Deregister a patch group from a patch baseline \(p. 1223\)](#)

## Create a patch group

To help you organize your patching efforts, we recommend that you add managed nodes 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 \(p. 1163\)](#).

After you group your managed nodes 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.

### Task 1: Add EC2 instances to a patch group using tags

#### Note

When using the Amazon Elastic Compute Cloud (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. If an EC2 instance you expect to see in Patch Manager isn't listed after applying the Patch Group tag, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

Run the following command to add the Patch Group tag to an EC2 instance.

```
aws ec2 create-tags --resources "i-1234567890abcdef0" --tags "Key=Patch Group,Value=GroupValue"
```

### Task 2: Add managed nodes to a patch group using tags

Run the following command to add the Patch Group tag to a managed node.

#### Linux & macOS

```
aws ssm add-tags-to-resource \
--resource-type "ManagedInstance" \
--resource-id "mi-0123456789abcdefg" \
--tags "Key=Patch Group,Value=GroupValue"
```

#### Windows

```
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

Run the following command to associate a Patch Group tag value to the specified patch baseline.

#### Linux & macOS

```
aws ssm register-patch-baseline-for-patch-group \
--baseline-id "pb-0c10e65780EXAMPLE" \
--patch-group "Development"
```

#### Windows

```
aws ssm register-patch-baseline-for-patch-group ^
```

```
--baseline-id "pb-0c10e65780EXAMPLE" ^
--patch-group "Development"
```

The system returns information like the following.

```
{
 "PatchGroup": "Development",
 "BaselineId": "pb-0c10e65780EXAMPLE"
}
```

## Register a patch group "web servers" with a patch baseline

Linux & macOS

```
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"
```

The system returns information like the following.

```
{
 "PatchGroup": "Web Servers",
 "BaselineId": "pb-0c10e65780EXAMPLE"
}
```

## Register a patch group "Backend" with the AWS-provided patch baseline

Linux & macOS

```
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

```
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"
```

The system returns information like the following.

```
{
 "PatchGroup": "Backend",
```

```
 "BaselineId": "arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE"
```

## Display patch group registrations

```
aws ssm describe-patch-groups --region us-east-2
```

The system returns information like the following.

```
{
 "PatchGroupPatchBaselineMappings": [
 {
 "PatchGroup": "Backend",
 "BaselineIdentity": {
 "BaselineName": "AWS-DefaultPatchBaseline",
 "DefaultBaseline": false,
 "BaselineDescription": "Default Patch Baseline Provided by AWS.",
 "BaselineId": "arn:aws:ssm:us-east-2:111122223333:patchbaseline/
pb-0c10e65780EXAMPLE"
 }
 },
 {
 "PatchGroup": "Web Servers",
 "BaselineIdentity": {
 "BaselineName": "Windows-Server-2012R2",
 "DefaultBaseline": true,
 "BaselineDescription": "Windows Server 2012 R2, Important and Critical updates",
 "BaselineId": "pb-0c10e65780EXAMPLE"
 }
 }
]
}
```

## Deregister a patch group from a patch baseline

Linux & macOS

```
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"
```

Windows

```
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"
}
```

## AWS CLI commands for viewing patch summaries and details

### Sample commands for viewing patch summaries and details

- [Get all patches defined by a patch baseline \(p. 1224\)](#)
- [Get all patches for AmazonLinux2018.03 that have a Classification SECURITY and Severity of Critical \(p. 1225\)](#)
- [Get all patches for Windows Server 2012 that have a MSRC severity of Critical \(p. 1226\)](#)
- [Get all available patches \(p. 1226\)](#)
- [Get patch summary states per-managed node \(p. 1227\)](#)
- [Get patch compliance details for a managed node \(p. 1228\)](#)
- [View patching compliance results \(AWS CLI\) \(p. 1229\)](#)

### Get all patches defined by a patch baseline

#### Note

This command is supported for Windows Server patch baselines only.

Linux & macOS

```
aws ssm describe-effective-patches-for-patch-baseline \
--region us-east-2 \
--baseline-id "pb-0c10e65780EXAMPLE"
```

Windows

```
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":{
 "ContentUrl":"https://support.microsoft.com/en-us/kb/2876331",
 "ProductFamily":"Windows",
 "Product":"WindowsServer2012R2",
 "Vendor":"Microsoft",
 "Description":"A security issue has been identified in a Microsoft software product that could affect your system. You can help protect your system by installing this update from Microsoft. For a complete listing of the issues that are included in this update, see the associated Microsoft Knowledge Base article. After you install this update, you may have to restart your system.",
 "Classification":"SecurityUpdates",
 "Title":"Security Update for Windows Server 2012 R2 Preview (KB2876331)",
 "ReleaseDate":1384279200.0,
 "MsrmClassification":"Critical",
 "Language":"All",
 "KbNumber":"KB2876331",
 "MsrmNumber":"MS13-089",
```

```

 "Id":"e74ccc76-85f0-4881-a738-59e9fc9a336d"
 },
},
{
 "PatchStatus": {
 "ApprovalDate": 1428858000.0,
 "DeploymentStatus": "APPROVED"
 },
 "Patch": {
 "ContentUrl": "https://support.microsoft.com/en-us/kb/2919355",
 "ProductFamily": "Windows",
 "Product": "WindowsServer2012R2",
 "Vendor": "Microsoft",
 "Description": "Windows Server 2012 R2 Update is a cumulative set of security updates, critical updates and updates. You must install Windows Server 2012 R2 Update to ensure that your computer can continue to receive future Windows Updates, including security updates. For a complete listing of the issues that are included in this update, see the associated Microsoft Knowledge Base article for more information. After you install this item, you may have to restart your computer.",
 "Classification": "SecurityUpdates",
 "Title": "Windows Server 2012 R2 Update (KB2919355)",
 "ReleaseDate": 1428426000.0,
 "MsrcClassification": "Critical",
 "Language": "All",
 "KbNumber": "KB2919355",
 "MsrcNumber": "MS14-018",
 "Id": "8452bac0-bf53-4fdb-915d-499de08c338b"
 }
}
---output truncated---

```

## Get all patches for AmazonLinux2018.03 that have a Classification SECURITY and Severity of Critical

Linux & macOS

```
aws ssm describe-available-patches \
--region us-east-2 \
--filters Key=PRODUCT,Values=AmazonLinux2018.03 Key=SEVERITY,Values=Critical
```

Windows

```
aws ssm describe-available-patches ^
--region us-east-2 ^
--filters Key=PRODUCT,Values=AmazonLinux2018.03 Key=SEVERITY,Values=Critical
```

The system returns information like the following.

```
{
 "Patches": [
 {
 "AdvisoryIds": ["ALAS-2011-1"],
 "BugzillaIds": ["1234567"],
 "Classification": "SECURITY",
 "CVEIds": ["CVE-2011-3192"],
 "Name": "zziplib",
 "Epoch": "0",
 "Version": "2.71",
 "Release": "1.3.amzn1",
 }
]
}
```

```
 "Arch": "i686",
 "Product": "AmazonLinux2018.03",
 "ReleaseDate": 1590519815,
 "Severity": "CRITICAL"
 }
]
}
---output truncated---
```

## Get all patches for Windows Server 2012 that have a MSRC severity of Critical

Linux & macOS

```
aws ssm describe-available-patches \
--region us-east-2 \
--filters Key=PRODUCT,Values=WindowsServer2012 Key=MSRC_SEVERITY,Values=Critical
```

Windows

```
aws ssm describe-available-patches ^
--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,
 "MsricClassification": "Critical",
 "Language": "All",
 "KbNumber": "KB2727528",
 "MsricNumber": "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":[
 {
 "ContentUrl":https://support.microsoft.com/en-us/kb/2032276",
 "ProductFamily":Windows",
 "Product":WindowsServer2008R2",
 "Vendor":Microsoft",
 "Description":A security issue has been identified that could allow an unauthenticated remote attacker to compromise your system and gain control over it. You can help protect your system by installing this update from Microsoft. After you install this update, you may have to restart your system.,
 "Classification":SecurityUpdates,
 "Title":Security Update for Windows Server 2008 R2 x64 Edition (KB2032276),
 "ReleaseDate":1279040400.0,
 "MsrcClassification":Important,
 "Language":All,
 "KbNumber":KB2032276,
 "MsrcNumber":MS10-043,
 "Id":8692029b-a3a2-4a87-a73b-8ea881b4b4d6
 },
 {
 "ContentUrl":https://support.microsoft.com/en-us/kb/2124261",
 "ProductFamily":Windows,
 "Product":Windows7",
 "Vendor":Microsoft,
 "Description":A security issue has been identified that could allow an unauthenticated remote attacker to compromise your system and gain control over it. You can help protect your system by installing this update from Microsoft. After you install this update, you may have to restart your system.,
 "Classification":SecurityUpdates,
 "Title":Security Update for Windows 7 (KB2124261),
 "ReleaseDate":1284483600.0,
 "MsrcClassification":Important,
 "Language":All,
 "KbNumber":KB2124261,
 "MsrcNumber":MS10-065,
 "Id":12ef1bed-0dd2-4633-b3ac-60888aa8ba33
 }
]
}
---output truncated---
```

## Get patch summary states per-managed node

The per-managed node summary gives you the number of patches in the following states per node: "NotApplicable", "Missing", "Failed", "InstalledOther" and "Installed".

Linux & macOS

```
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-0c10e65780EXAMPLE",
 "SnapshotId": "6d03d6c5-f79d-41d0-8d0e-00a9aEXAMPLE",
 "InstalledCount": 50,
 "InstalledOtherCount": 353,
 "InstalledPendingRebootCount": 0,
 "InstalledRejectedCount": 0,
 "MissingCount": 0,
 "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-0c10e65780EXAMPLE",
 "SnapshotId": "c7e0441b-1aea-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 a managed node

```
aws ssm describe-instance-patches --instance-id i-08ee91c0b17045407
```

The system returns information like the following.

```
{
 "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.P1.28.amzn1",
 "KBId": "dhclient.x86_64",
 "Classification": "Security",
 "Severity": "Important",
 "State": "Installed",
 "InstalledTime": "2019-08-26T11:05:31-07:00"
 },
 ...
---output truncated---

```

## View patching compliance results (AWS CLI)

### To view patch compliance results for a single managed node

Run the following command in the AWS Command Line Interface (AWS CLI) to view patch compliance results for a single managed node.

```
aws ssm describe-instance-patch-states --instance-id instance-id
```

Replace *instance-id* with the ID of the managed node for which you want to view results, in the format **i-02573cafccEXAMPLE** or **mi-0282f7c436EXAMPLE**.

The system returns information like the following.

```
{
 "InstancePatchStates": [
 {
 "InstanceId": "i-02573cafccEXAMPLE",
 "PatchGroup": "mypatchgroup",
 "BaselineId": "pb-0c10e65780EXAMPLE",
 "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
 "CriticalNonCompliantCount": 2,
 "SecurityNonCompliantCount": 2,
 "OtherNonCompliantCount": 1,
 "InstalledCount": 123,
 "InstalledOtherCount": 334,
 "InstalledPendingRebootCount": 0,
 "InstalledRejectedCount": 0,
 "MissingCount": 1,
 "FailedCount": 2,
 "UnreportedNotApplicableCount": 11,
 "NotApplicableCount": 2063,
 "OperationStartTime": "2021-05-03T11:00:56-07:00",
 "OperationEndTime": "2021-05-03T11:01:09-07:00",
 "Operation": "Scan",
 "LastNoRebootInstallOperationTime": "2020-06-14T12:17:41-07:00",
 ...
 }
]
}
```

```
 "RebootOption": "RebootIfNeeded"
]
}
```

### To view a patch count summary for all EC2 instances in a Region

The `describe-instance-patch-states` supports retrieving results for just one managed instance at a time. However, using a custom script with the `describe-instance-patch-states` command, you can generate a more granular report.

For example, if the [jq filter tool](#) is installed on your local machine, you could run the following command to identify which of your EC2 instances in a particular AWS Region have a status of `InstalledPendingReboot`.

```
aws ssm describe-instance-patch-states \
 --instance-ids $(aws ec2 describe-instances --region region | jq
'.Reservations[].Instances[] | .InstanceId' | tr '\n|'' ''') \
 --output text --query 'InstancePatchStates[*].{InstanceId:InstanceId,
InstalledPendingRebootCount:InstalledPendingRebootCount}'
```

**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:

```
aws ssm describe-instance-patch-states \
 --instance-ids $(aws ec2 describe-instances --region us-east-2 | jq
'.Reservations[].Instances[] | .InstanceId' | tr '\n|'' ''') \
 --output text --query 'InstancePatchStates[*].{InstanceId:InstanceId,
InstalledPendingRebootCount:InstalledPendingRebootCount}'
```

The system returns information like the following.

```
1 i-02573cafccEXAMPLE
0 i-0471e04240EXAMPLE
3 i-07782c72faEXAMPLE
6 i-083b678d37EXAMPLE
0 i-03a530a2d4EXAMPLE
1 i-01f68df0d0EXAMPLE
0 i-0a39c0f214EXAMPLE
7 i-0903a5101eEXAMPLE
7 i-03823c2fedEXAMPLE
```

In addition to `InstalledPendingRebootCount`, the list of count types you can search for include the following:

- `CriticalNonCompliantCount`
- `SecurityNonCompliantCount`
- `OtherNonCompliantCount`
- `UnreportedNotApplicableCount`
- `InstalledPendingRebootCount`
- `FailedCount`
- `NotApplicableCount`
- `InstalledRejectedCount`
- `InstalledOtherCount`

- `MissingCount`
- `InstalledCount`

## AWS CLI commands for scanning and patching managed nodes

After running the following commands to scan for patch compliance or install patches, you can use commands in the [AWS CLI commands for viewing patch summaries and details \(p. 1224\)](#) section to view information about patch status and compliance.

### Sample commands

- [Scan managed nodes for patch compliance \(AWS CLI\) \(p. 1231\)](#)
- [Install patches on managed nodes \(AWS CLI\) \(p. 1233\)](#)

## Scan managed nodes for patch compliance (AWS CLI)

### To scan specific managed nodes for patch compliance

Run the following command.

Linux & macOS

```
aws ssm send-command \
 --document-name 'AWS-RunPatchBaseline' \
 --targets Key=InstanceIds,Values='i-02573cafefEXAMPLE,i-0471e04240EXAMPLE' \
 --parameters 'Operation=Scan' \
 --timeout-seconds 600
```

Windows

```
aws ssm send-command ^
 --document-name "AWS-RunPatchBaseline" ^
 --targets Key=InstanceIds,Values="i-02573cafefEXAMPLE,i-0471e04240EXAMPLE" ^
 --parameters "Operation=Scan" ^
 --timeout-seconds 600
```

The system returns information like the following.

```
{
 "Command": {
 "CommandId": "a04ed06c-8545-40f4-87c2-a0babEXAMPLE",
 "DocumentName": "AWS-RunPatchBaseline",
 "DocumentVersion": "$DEFAULT",
 "Comment": "",
 "ExpiresAfter": 1621974475.267,
 "Parameters": {
 "Operation": [
 "Scan"
]
 },
 "InstanceIds": [],
 "Targets": [
 {
 "Key": "InstanceIds",
 "Values": [
 "i-02573cafefEXAMPLE",
 i-0471e04240EXAMPLE"
]
 }
]
 }
}
```

```
 }
],
 "RequestedDateTime": 1621952275.267,
 "Status": "Pending",
 "StatusDetails": "Pending",
 "TimeoutSeconds": 600,
 ---output truncated---
}
```

### To scan managed nodes for patch compliance by patch group tag

Run the following command.

Linux & macOS

```
aws ssm send-command \
--document-name 'AWS-RunPatchBaseline' \
--targets Key='tag:Patch Group',Values='Web servers' \
--parameters 'Operation=Scan' \
--timeout-seconds 600
```

Windows

```
aws ssm send-command ^
--document-name "AWS-RunPatchBaseline" ^
--targets Key="tag:Patch Group",Values="Web servers" ^
--parameters "Operation=Scan" ^
--timeout-seconds 600
```

The system returns information like the following.

```
{
 "Command": {
 "CommandId": "87a448ee-8adc-44e0-b4d1-6b429EXAMPLE",
 "DocumentName": "AWS-RunPatchBaseline",
 "DocumentVersion": "$DEFAULT",
 "Comment": "",
 "ExpiresAfter": 1621974983.128,
 "Parameters": {
 "Operation": [
 "Scan"
]
 },
 "InstanceIds": [],
 "Targets": [
 {
 "Key": "tag:Patch Group",
 "Values": [
 "Web servers"
]
 }
],
 "RequestedDateTime": 1621952783.128,
 "Status": "Pending",
 "StatusDetails": "Pending",
 "TimeoutSeconds": 600,
 ---output truncated---
 }
}
```

```
}
```

## Install patches on managed nodes (AWS CLI)

### To install patches on specific managed nodes

Run the following command.

#### Note

The target managed nodes reboot as needed to complete patch installation. For more information, see [About the AWS-RunPatchBaseline SSM document \(p. 1128\)](#).

#### Linux & macOS

```
aws ssm send-command \
--document-name 'AWS-RunPatchBaseline' \
--targets Key=InstanceIds,Values='i-02573cafefEXAMPLE,i-0471e04240EXAMPLE' \
--parameters 'Operation=Install' \
--timeout-seconds 600
```

#### Windows

```
aws ssm send-command ^
--document-name "AWS-RunPatchBaseline" ^
--targets Key=InstanceIds,Values="i-02573cafefEXAMPLE,i-0471e04240EXAMPLE" ^
--parameters "Operation=Install" ^
--timeout-seconds 600
```

The system returns information like the following.

```
{
 "Command": {
 "CommandId": "5f403234-38c4-439f-a570-93623EXAMPLE",
 "DocumentName": "AWS-RunPatchBaseline",
 "DocumentVersion": "$DEFAULT",
 "Comment": "",
 "ExpiresAfter": 1621975301.791,
 "Parameters": {
 "Operation": [
 "Install"
]
 },
 "InstanceIds": [],
 "Targets": [
 {
 "Key": "InstanceIds",
 "Values": [
 "i-02573cafefEXAMPLE",
 i-0471e04240EXAMPLE"
]
 }
],
 "RequestedDateTime": 1621953101.791,
 "Status": "Pending",
 "StatusDetails": "Pending",
 "TimeoutSeconds": 600,
 ...
 }
}
```

```
}
```

### To install patches on managed nodes in a specific patch group

Run the following command.

Linux & macOS

```
aws ssm send-command \
 --document-name 'AWS-RunPatchBaseline' \
 --targets Key='tag:Patch Group',Values='Web servers' \
 --parameters 'Operation=Install' \
 --timeout-seconds 600
```

Windows

```
aws ssm send-command ^
 --document-name "AWS-RunPatchBaseline" ^
 --targets Key="Patch Group",Values="Web servers" ^
 --parameters "Operation=Install" ^
 --timeout-seconds 600
```

The system returns information like the following.

```
{
 "Command": {
 "CommandId": "fa44b086-7d36-4ad5-ac8d-627ecEXAMPLE",
 "DocumentName": "AWS-RunPatchBaseline",
 "DocumentVersion": "$DEFAULT",
 "Comment": "",
 "ExpiresAfter": 1621975407.865,
 "Parameters": {
 "Operation": [
 "Install"
]
 },
 "InstanceIds": [],
 "Targets": [
 {
 "Key": "tag:Patch Group",
 "Values": [
 "Web servers"
]
 }
],
 "RequestedDateTime": 1621953207.865,
 "Status": "Pending",
 "StatusDetails": "Pending",
 "TimeoutSeconds": 600,
 ---output truncated---
 }
}
```

## AWS Systems ManagerPatch Manager walkthroughs

The walkthroughs in this section demonstrate how to use Patch Manager, a capability of AWS Systems Manager, for several patching scenarios.

## Topics

- [Walkthrough: Create a patch baseline for installing Windows Service Packs \(console\) \(p. 1235\)](#)
- [Walkthrough: Update application dependencies, patch a managed node, and perform an application-specific health check \(p. 1236\)](#)
- [Walkthrough: Patch a server environment \(AWS CLI\) \(p. 1238\)](#)

## 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, a capability of AWS Systems 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 managed nodes.

### To use Patch Manager to install Windows Service Packs (console) (Windows)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 [Setting an existing patch baseline as the default \(console\) \(p. 1202\)](#).

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 **WindowsServer2012**. 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 360. For most scenarios, we recommend waiting no more than 100 days.

- **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 allow 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: Update application dependencies, patch a managed node, and perform an application-specific health check

In many cases, a managed node must be rebooted after it has been patched with the latest software update. However, rebooting a node in production without safeguards in place can cause several problems, such as invoking alarms, recording incorrect metric data, and interrupting data synchronizations.

This walkthrough demonstrates how to avoid problems like these by using the AWS Systems Manager document (SSM document) `AWS-RunPatchBaselineWithHooks` to achieve a complex, multi-step patching operation that accomplishes the following:

1. Prevent new connections to the application
2. Install operating system updates
3. Update the package dependencies of the application
4. Restart the system
5. Perform an application-specific health check

For this example, we have set up our infrastructure this way:

- The virtual machines targeted are registered as managed nodes with Systems Manager.
- `Iptables` is used as a local firewall.
- The application hosted on the managed nodes is running on port 443.
- The application hosted on the managed nodes is a `nodeJS` application.
- The application hosted on the managed nodes is managed by the `pm2` process manager.
- The application already has a specified health check endpoint.
- The application's health check endpoint requires no end user authentication. The endpoint allows for a health check that meets the organization's requirements in establishing availability. (In your

environments, it may be enough to simply ascertain that the nodeJS application is running and able to listen for requests. In other cases, you might want to also verify that a connection to the caching layer or database layer has already been established.)

The examples in this walkthrough are for demonstration purposes only and not meant to be implemented as-is into production environments. Also, keep in mind that the lifecycle hooks feature of Patch Manager, a capability of Systems Manager, with the `AWS-RunPatchBaselineWithHooks` document can support numerous other scenarios. Here are several examples.

- Stop a metrics reporting agent before patching and restarting it after the managed node reboots.
- Detach the managed node from a CRM or PCS cluster before patching and reattach after the node reboots.
- Update third-party software (for example, Java, Tomcat, Adobe applications, and so on) on Windows Server machines after operating system (OS) updates are applied, but before the managed node reboots.

### To update application dependencies, patch a managed node, and perform an application-specific health check

1. Create an SSM document for your preinstallation script with the following contents and name it `NodeJSAppPrePatch`. Replace `your_application` with the name of your application.

This script immediately blocks new incoming requests and provides five seconds for already active ones to complete before beginning the patching operation. For the `sleep` option, specify a number of seconds greater than it usually takes for incoming requests to complete.

```
exit on error
set -e
set up rule to block incoming traffic
iptables -I INPUT -j DROP -p tcp --syn --destination-port 443 || exit 1
wait for current connections to end. Set timeout appropriate to your application's
latency
sleep 5
Stop your application
pm2 stop your_application
```

For information about creating SSM documents, see [Creating SSM documents \(p. 1352\)](#).

2. Create another SSM document with the following content for your postinstall script to update your application dependencies and name it `NodeJSAppPostPatch`. Replace `/your/application/path` with the path to your application.

```
cd /your/application/path
npm update
you can use npm-check-updates if you want to upgrade major versions
```

3. Create another SSM document with the following content for your onExit script to bring your application back up and perform a health check. Name this SSM document `NodeJSAppOnExitPatch`. Replace `your_application` with the name of your application.

```
exit on error
set -e
restart nodeJS application
pm2 start your_application
sleep while your application starts and to allow for a crash
sleep 10
check with pm2 to see if your application is running
pm2 pid your_application
```

```
re-enable incoming connections
iptables -D INPUT -j DROP -p tcp --syn --destination-port
perform health check
/usr/bin/curl -m 10 -vk -A "" http://localhost:443/health-check || exit 1
```

4. Create an association in State Manager, a capability of AWS Systems Manager, to issue the operation by performing the following steps:
  1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
  2. In the navigation pane, choose **State Manager**, and then choose **Create association**.
  3. For **Name**, provide a name to help identify the purpose of the association.
  4. In the **Document** list, choose **AWS-RunPatchBaselineWithHooks**.
  5. For **Operation**, choose **Install**.
  6. (Optional) For **Snapshot Id**, provide a GUID that you generate to help speed up the operation and ensure consistency. The GUID value can be as simple as **00000000-0000-0000-0000-111122223333**.
  7. For **Pre Install Hook Doc Name**, enter **NodeJSAppPrePatch**.
  8. For **Post Install Hook Doc Name**, enter **NodeJSAppPostPatch**.
  9. For **On ExitHook Doc Name**, enter **NodeJSAppOnExitPatch**.
  5. For **Targets**, identify your managed nodes by specifying tags, choosing nodes manually, choosing a resource group, or choosing all managed nodes.
  6. For **Specify schedule**, specify how often to run the association. For managed node patching, once per week is a common cadence.
  7. In the **Rate control** section, choose options to control how the association runs on multiple managed nodes. Ensure that only a portion of managed nodes are updated at a time. Otherwise, all or most of your fleet could be taken offline at once. For more information about using rate controls, see [About targets and rate controls in State Manager associations \(p. 1039\)](#).
  8. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Enter 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 managed node, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

9. Choose **Create Association**.

## Walkthrough: Patch a server environment (AWS CLI)

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 managed nodes. To patch Linux managed nodes, your nodes must be running SSM Agent version 2.0.834.0 or later. For more information, see [Update SSM Agent by using Run Command \(p. 997\)](#).
- Configure roles and permissions for Maintenance Windows, a capability of AWS Systems Manager. For more information, see [Setting up Maintenance Windows \(p. 707\)](#).
- Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

## To configure Patch Manager and patch managed nodes (command line)

- 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're released. That is, we have tagged the patch baseline to indicate that it's for a production environment.

### Note

The `OperatingSystem` parameter and `PatchFilters` vary depending on the operating system of the target managed nodes the patch baseline applies to. For more information, see [OperatingSystem](#) and [PatchFilter](#).

Linux & macOS

```
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]} \
}}" \
--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]} \
}}" ^
--description "Baseline containing all updates approved for production systems"
```

The system returns information like the following.

```
{ \
 "BaselineId": "pb-0c10e65780EXAMPLE"
}
```

- 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 & macOS

```
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"
```

The system returns information like the following.

```
{
 "PatchGroup": "Database Servers",
 "BaselineId": "pb-0c10e65780EXAMPLE"
}
```

#### Linux & macOS

```
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"
```

The system returns information like the following.

```
{
 "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's for a production environment.

#### Linux & macOS

```
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
```

The system returns information like the following.

```
{
 "WindowId": "mw-0c50858d01EXAMPLE"
```

```
}
```

#### Linux & macOS

```
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" ^
--schedule "cron(0 0 22 ? * SAT *)" ^
--duration 2 ^
--cutoff 0 ^
--no-allow-unassociated-targets
```

The system returns information like the following.

```
{
 "WindowId": "mw-9a8b7c6d5eEXAMPLE"
}
```

4. Run the following commands to register the Database and Front-End servers patch groups with their respective maintenance windows.

#### Linux & macOS

```
aws ssm register-target-with-maintenance-window \
--window-id mw-0c50858d01EXAMPLE \
--targets "Key=tag:Patch Group,Values=Database Servers" \
--owner-information "Database Servers" \
--resource-type "INSTANCE"
```

#### Windows

```
aws ssm register-target-with-maintenance-window ^
--window-id mw-0c50858d01EXAMPLE ^
--targets "Key=tag:Patch Group,Values=Database Servers" ^
--owner-information "Database Servers" ^
--resource-type "INSTANCE"
```

The system returns information like the following.

```
{
 "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
}
```

## Linux & macOS

```
aws ssm register-target-with-maintenance-window \
--window-id mw-9a8b7c6d5eEXAMPLE \
--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-9a8b7c6d5eEXAMPLE ^
--targets "Key=tag:Patch Group,Values=Front-End Servers" ^
--owner-information "Front-End Servers" ^
--resource-type "INSTANCE"
```

The system returns information like the following.

```
{
 "WindowTargetId": "faa01c41-1d57-496c-ba77-ff9caEXAMPLE"
}
```

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 & macOS

```
aws ssm register-task-with-maintenance-window \
--window-id mw-0c50858d01EXAMPLE \
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" \
--task-arn "AWS-RunPatchBaseline" \
--service-role-arn "arn:aws:iam::123456789012:role/MW-Role" \
--task-type "RUN_COMMAND" \
--max-concurrency 2 \
--max-errors 1 \
--priority 1 \
--task-invocation-parameters "RunCommand={Parameters={Operation=Install}}"
```

## Windows

```
aws ssm register-task-with-maintenance-window ^
--window-id mw-0c50858d01EXAMPLE ^
--targets "Key=WindowTargetIds,Values=e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE" ^
--task-arn "AWS-RunPatchBaseline" ^
--service-role-arn "arn:aws:iam::123456789012:role/MW-Role" ^
--task-type "RUN_COMMAND" ^
--max-concurrency 2 ^
--max-errors 1 ^
--priority 1 ^
--task-invocation-parameters "RunCommand={Parameters={Operation=Install}}"
```

The system returns information like the following.

```
{
 "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
```

```
}
```

### Linux & macOS

```
aws ssm register-task-with-maintenance-window \
--window-id mw-9a8b7c6d5eEXAMPLE \
--targets "Key=WindowTargetIds,Values=faa01c41-1d57-496c-ba77-ff9caEXAMPLE" \
--task-arn "AWS-RunPatchBaseline" \
--service-role-arn "arn:aws:iam::123456789012:role/MW-Role" \
--task-type "RUN_COMMAND" \
--max-concurrency 2 \
--max-errors 1 \
--priority 1 \
--task-invocation-parameters "RunCommand={Parameters={Operation=Install}}"
```

### Windows

```
aws ssm register-task-with-maintenance-window ^
--window-id mw-9a8b7c6d5eEXAMPLE ^
--targets "Key=WindowTargetIds,Values=faa01c41-1d57-496c-ba77-ff9caEXAMPLE" ^
--task-arn "AWS-RunPatchBaseline" ^
--service-role-arn "arn:aws:iam::123456789012:role/MW-Role" ^
--task-type "RUN_COMMAND" ^
--max-concurrency 2 ^
--max-errors 1 ^
--priority 1 ^
--task-invocation-parameters "RunCommand={Parameters={Operation=Install}}"
```

The system returns information like the following.

```
{
 "WindowTaskId": "8a5c4629-31b0-4edd-8aea-33698EXAMPLE"
}
```

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 managed nodes with patches in the respective patch states.

**Note**

It's expected to see zeroes for the number of managed nodes in the summary until the patch task runs during the first maintenance window.

### Linux & macOS

```
aws ssm describe-patch-group-state \
--patch-group "Database Servers"
```

### Windows

```
aws ssm describe-patch-group-state ^
--patch-group "Database Servers"
```

The system returns information like the following.

```
{
 "Instances": number,
```

```
"InstancesWithFailedPatches": number,
"InstancesWithInstalledOtherPatches": number,
"InstancesWithInstalledPatches": number,
"InstancesWithInstalledPendingRebootPatches": number,
"InstancesWithInstalledRejectedPatches": number,
"InstancesWithMissingPatches": number,
"InstancesWithNotApplicablePatches": number,
"InstancesWithUnreportedNotApplicablePatches": number
}
```

- Run the following command to get patch summary states per-managed node for a patch group. The per-managed node summary includes a number of patches in the respective patch states per managed node for a patch group.

#### Linux & macOS

```
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"
```

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 \(AWS CLI\) \(p. 1210\)](#).

# Troubleshooting Patch Manager

Use the following information to help you troubleshoot problems with Patch Manager, a capability of AWS Systems Manager.

## Topics

- [Errors when running AWS-RunPatchBaseline on Linux \(p. 1245\)](#)
- [Errors when running AWS-RunPatchBaseline on Windows Server \(p. 1248\)](#)
- [Contacting AWS Support \(p. 1252\)](#)

## Errors when running AWS-RunPatchBaseline on Linux

### Topics

- [Issue: 'No such file or directory' error \(p. 1245\)](#)
- [Issue: 'another process has acquired yum lock' error \(p. 1246\)](#)
- [Issue: 'Permission denied / failed to run commands' error \(p. 1246\)](#)
- [Issue: 'Unable to download payload' error \(p. 1246\)](#)
- [Issue: 'unsupported package manager and python version combination' error \(p. 1246\)](#)
- [Issue: Patch Manager isn't applying rules specified to exclude certain packages \(p. 1247\)](#)
- [Issue: Patching fails and Patch Manager reports that the Server Name Indication extension to TLS is not available \(p. 1247\)](#)
- [Issue: Patch Manager reports 'No more mirrors to try' \(p. 1247\)](#)

### Issue: 'No such file or directory' error

**Problem:** When you run AWS-RunPatchBaseline, patching fails with one of the following errors.

```
IOError: [Errno 2] No such file or directory: 'patch-baseline-operations-X.XX.tar.gz'
```

```
Unable to extract tar file: /var/log/amazon/ssm/patch-baseline-operations/patch-baseline-operations-1.75.tar.gz.failed to run commands: exit status 155
```

```
Unable to load and extract the content of payload, abort.failed to run commands: exit status 152
```

**Cause 1:** Two commands to run AWS-RunPatchBaseline were running at the same time on the same managed node. This creates a race condition that results in the temporary file `patch-baseline-operations*` not being created or accessed properly.

**Cause 2:** Insufficient storage space remains under the `/var` directory.

**Solution 1:** Ensure that no maintenance window has two or more Run Command tasks that run AWS-RunPatchBaseline with the same Priority level and that run on the same target IDs. If this is the case, reorder the priority. Run Command is a capability of AWS Systems Manager.

**Solution 2:** Ensure that only one maintenance window at a time is running Run Command tasks that use AWS-RunPatchBaseline on the same targets and on the same schedule. If this is the case, change the schedule.

**Solution 3:** Ensure that only one State Manager association is running AWS-RunPatchBaseline on the same schedule and targeting the same managed nodes. State Manager is a capability of AWS Systems Manager.

**Solution 4:** Free up sufficient storage space under the /var directory for the update packages.

### Issue: 'another process has acquired yum lock' error

**Problem:** When you run AWS-RunPatchBaseline, patching fails with the following error.

```
12/20/2019 21:41:48 root [INFO]: another process has acquired yum lock, waiting 2 s and
retry.
```

**Cause:** The AWS-RunPatchBaseline document has started running on a managed node where it's already running in another operation and has acquired the package manager yum process.

**Solution:** Ensure that no State Manager association, maintenance window tasks, or other configurations that run AWS-RunPatchBaseline on a schedule) are targeting the same managed node around the same time.

### Issue: 'Permission denied / failed to run commands' error

**Problem:** When you run AWS-RunPatchBaseline, patching fails with the following error.

```
sh:
/var/lib/amazon/ssm/instanceid/document/orchestration/commandid/PatchLinux/_script.sh:
Permission denied
failed to run commands: exit status 126
```

**Cause:** /var/lib/amazon/ might be mounted with noexec permissions. This is an issue because SSM Agent downloads payload scripts to /var/lib/amazon/ssm and runs them from that location.

**Solution:** Ensure that you have configured exclusive partitions to /var/log/amazon and /var/lib/amazon, and that they're mounted with exec permissions.

### Issue: 'Unable to download payload' error

**Problem:** When you run AWS-RunPatchBaseline, patching fails with the following error.

```
Unable to download payload: https://s3.DOC-EXAMPLE-BUCKET.region.amazonaws.com/
aws-ssm-region/patchbaselineoperations/linux/payloads/patch-baseline-operations-
X.XX.tar.gz.failed to run commands: exit status 156
```

**Cause:** The managed node doesn't have the required permissions to access the specified Amazon Simple Storage Service (Amazon S3) bucket.

**Solution:** Update your network configuration so that S3 endpoints are reachable. For more details, see information about required access to S3 buckets for Patch Manager in [SSM Agent communications with AWS managed S3 buckets \(p. 138\)](#).

### Issue: 'unsupported package manager and python version combination' error

**Problem:** When you run AWS-RunPatchBaseline, patching fails with the following error.

```
An unsupported package manager and python version combination was found. Apt requires
Python3 to be installed.
```

```
failed to run commands: exit status 1
```

**Cause:** python3 isn't installed on the Debian Server, Raspberry Pi OS, or Ubuntu Server instance.

**Solution:** Install python3 on the server, which is required for Debian Server, Raspberry Pi OS, and Ubuntu Server managed nodes.

## Issue: Patch Manager isn't applying rules specified to exclude certain packages

**Problem:** You have attempted to exclude certain packages by specifying them in the /etc/yum.conf file, in the format exclude=*package-name*, but they aren't excluded during the Patch Manager Install operation.

**Cause:** Patch Manager doesn't incorporate exclusions specified in the /etc/yum.conf file.

**Solution:** To exclude specific packages, create a custom patch baseline and create a rule to exclude the packages you don't want installed.

## Issue: Patching fails and Patch Manager reports that the Server Name Indication extension to TLS is not available

**Problem:** The patching operation issues the following message.

```
/var/log/amazon/ssm/patch-baseline-operations/urllib3/util/ssl_.py:369:
SNIMissingWarning: An HTTPS request has been made, but the SNI (Server Name Indication)
extension
to TLS is not available on this platform. This may cause the server to present an incorrect
TLS
certificate, which can cause validation failures. You can upgrade to a newer version of
Python
to solve this.
For more information, see https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-
warnings
```

**Cause:** This message doesn't indicate an error. Instead, it's a warning that the older version of Python distributed with the operating system doesn't support TLS Server Name Indication. The Systems Manager patch payload script issues this warning when connecting to AWS APIs that support SNI.

**Solution:** To troubleshoot any patching failures when this message is reported, review the contents of the stdout and stderr files. If you haven't configured the patch baseline to store these files in an Amazon S3 bucket or in Amazon CloudWatch Logs, you can locate the files in the following location on your Linux managed node.

```
/var/lib/amazon/ssm/instance-id/document/orchestration/Run-Command-execution-id/awsrunShellScript/PatchLinux
```

## Issue: Patch Manager reports 'No more mirrors to try'

**Problem:** The patching operation issues the following message.

```
[Errno 256] No more mirrors to try.
```

**Cause:** The repositories configured on the managed node are not working correctly. Possible causes for this include:

- The yum cache is corrupted.

- A repository URL can't be reached due to network-related issues.

**Solution:** Patch Manager uses the managed node's default package manager to perform patching operation. Double-check that repositories are configured and operating correctly.

## Errors when running AWS-RunPatchBaseline on Windows Server

### Topics

- [Issue: mismatched product family/product pairs \(p. 1248\)](#)
- [Issue: AWS-RunPatchBaseline output returns an HRESULT \(Windows Server\) \(p. 1248\)](#)
- [Issue: managed node doesn't have access to Windows Update Catalog or WSUS \(p. 1249\)](#)
- [Issue: PatchBaselineOperations PowerShell module is not downloadable \(p. 1250\)](#)
- [Issue: missing patches \(p. 1251\)](#)

### Issue: mismatched product family/product pairs

**Problem:** When you create a patch baseline in the Systems Manager console, you specify a product family and a product. For example, you might choose:

- **Product family: Office**

**Product: Office 2016**

**Cause:** 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 and 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 Command Line Interface (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 is also included in the **Obsolete or mismatched options** sublist if it was specified for a previous patch baseline but has no patches available from Microsoft.

**Solution:** To avoid this issue in the console, always choose options from the **Currently available options** sublists.

You can also view the products that have available patches by using the [describe-patch-properties](#) command in the AWS CLI or the [DescribePatchProperties](#) API command.

### Issue: AWS-RunPatchBaseline output returns an HRESULT (Windows Server)

**Problem:** You received an error like the following.

```
-----ERROR-----
Invoke-PatchBaselineOperation : Exception Details: An error occurred when
attempting to search Windows Update.
Exception Level 1:
Error Message: Exception from HRESULT: 0x80240437
Stack Trace: at WUApiLib.IUpdateSearcher.Search(String criteria)..
(Windows updates)
```

```
11/22/2020 09:17:30 UTC | Info | Searching for Windows Updates.
11/22/2020 09:18:59 UTC | Error | Searching for updates resulted in error: Exception from
HRESULT: 0x80240437
-----ERROR-----
failed to run commands: exit status 4294967295
```

**Cause:** This output indicates that the native Windows Update APIs were unable to run the patching operations.

**Solution:** Check the HRESULT code in the [Microsoft documentation](#) to identify troubleshooting steps for resolving the error.

## Issue: managed node doesn't have access to Windows Update Catalog or WSUS

**Problem:** You received an error like the following.

```
Downloading PatchBaselineOperations PowerShell module from https://s3.amazonaws.com/path_to_module.zip to C:\Windows\TEMP\Amazon.PatchBaselineOperations-1.29.zip.
Extracting PatchBaselineOperations zip file contents to temporary folder.
Verifying SHA 256 of the PatchBaselineOperations PowerShell module files.
Successfully downloaded and installed the PatchBaselineOperations PowerShell module.
Patch Summary for
PatchGroup :
BaselineId :
Baseline : null
SnapshotId :
RebootOption : RebootIfNeeded
OwnerInformation :
OperationType : Scan
OperationStartTime : 1970-01-01T00:00:00.0000000Z
OperationEndTime : 1970-01-01T00:00:00.0000000Z
InstalledCount : -1
InstalledRejectedCount : -1
InstalledPendingRebootCount : -1
InstalledOtherCount : -1
FailedCount : -1
MissingCount : -1
NotApplicableCount : -1
UnreportedNotApplicableCount : -1
EC2AMAZ-VL3099P - PatchBaselineOperations Assessment Results - 2020-12-30T20:59:46.169
-----ERROR-----
```

```
Invoke-PatchBaselineOperation : Exception Details: An error occurred when attempting to
search Windows Update.

Exception Level 1:

Error Message: Exception from HRESULT: 0x80072EE2

Stack Trace: at WUApiLib.IUpdateSearcher.Search(String criteria)
at
Amazon.Patch.Baseline.Operations.PatchNow.Implementations.WindowsUpdateAgent.SearchForUpdates(String
searchCriteria)

At C:\ProgramData\Amazon\SSM\InstanceData\i-02573cafcfEXAMPLE\document\orchestration
\3d2d4864-04b7-4316-84fe-eafff1ea58

e3\PatchWindows_script.ps1:230 char:13
+ $response = Invoke-PatchBaselineOperation -Operation Install -Snapshot ...
+ -----
+ CategoryInfo : OperationStopped:
(Amazon.Patch.Ba...UpdateOperation:InstallWindowsUpdateOperation) [Inv
oke-PatchBaselineOperation], Exception
+ FullyQualifiedErrorId : Exception Level 1:

Error Message: Exception Details: An error occurred when attempting to search Windows
Update.

Exception Level 1:

Error Message: Exception from HRESULT: 0x80072EE2

Stack Trace: at WUApiLib.IUpdateSearcher.Search(String criteria)
at
Amazon.Patch.Baseline.Operations.PatchNow.Implementations.WindowsUpdateAgent.SearchForUpdates(String
searc

---Error truncated---
```

**Cause:** This error could be related to the Windows Update components, or to a lack of connectivity to the Windows Update Catalog or Windows Server Update Services (WSUS).

**Solution:** Confirm that the managed node has connectivity to the [Microsoft Update Catalog](#) through an internet gateway, NAT gateway, or NAT instance. If you're using WSUS, confirm that the managed node has connectivity to the WSUS server in your environment. If connectivity is available to the intended destination, check the Microsoft documentation for other potential causes of HRESULT 0x80072EE2. This might indicate an operating system level issue.

## Issue: PatchBaselineOperations PowerShell module is not downloadable

**Problem:** You received an error like the following.

```
Preparing to download PatchBaselineOperations PowerShell module from S3.

Downloading PatchBaselineOperations PowerShell module from https://s3.amazonaws.com/path_to_module.zip to C:\Windows\TEMP\Amazon.PatchBaselineOperations-1.29.zip.
```

```
-----ERROR-----
C:\ProgramData\Amazon\SSM\InstanceData\i-02573cafcfEXAMPLE\document\orchestration\aaaaaaaa-
bbbb-cccc-dddd-4f6ed6bd5514\

PatchWindows_script.ps1 : An error occurred when executing PatchBaselineOperations: Unable
to connect to the remote server

+ CategoryInfo : NotSpecified: (:) [Write-Error], WriteErrorException
+ FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,_script.ps1

failed to run commands: exit status 4294967295
```

**Solution:** Check the managed node connectivity and permissions to Amazon Simple Storage Service (Amazon S3). The managed node's AWS Identity and Access Management (IAM) role must use the minimum permissions cited in [SSM Agent communications with AWS managed S3 buckets \(p. 138\)](#). The node must communicate with the Amazon S3 endpoint through the Amazon S3 gateway endpoint, NAT gateway, or internet gateway. For more information about the VPC Endpoint requirements for AWS Systems Manager SSM Agent (SSM Agent), see [Step 6: \(Optional\) Create a VPC endpoint \(p. 28\)](#).

## Issue: missing patches

**Problem:** AWS–RunPatchbaseline completed successfully, but there are some missing patches.

The following are some common causes and their solutions.

**Cause 1:** The baseline isn't effective.

**Solution 1:** To check if this is the cause, use the following procedure.

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 history** tab and then select the command whose baseline you want to check.
4. Select the managed node that has missing patches.
5. Select **Step 1 - Output** and find the BaselineId value.
6. Check the assigned [patch baseline configuration \(p. 1159\)](#), that is, the operating system, product name, classification, and severity for the patch baseline.
7. Go to the [Microsoft Update Catalog](#).
8. Search the Microsoft Knowledge Base (KB) article IDs (for example, KB3216916).
9. Verify that the value under **Product** matches that of your managed node and select the corresponding **Title**. A new **Update Details** window will open.
10. In the **Overview** tab, the **classification** and **MSRC severity** must match the patch baseline configuration you found earlier.

**Cause 2:** The patch was replaced.

**Solution 2:** To check if this is true, use the following procedure.

1. Go to the [Microsoft Update Catalog](#).
2. Search the Microsoft Knowledge Base (KB) article IDs (for example, KB3216916).

3. Verify that the value under **Product** matches that of your managed node and select the corresponding **Title**. A new **Update Details** window will open.
4. Go to the **Package Details** tab. Look for an entry under the **This update has been replaced by the following updates:** header.

**Cause 3:** The same patch might have different KB numbers because the WSUS and Window online updates are handled as independent Release Channels by Microsoft.

**Solution 3:** Check the patch eligibility. If the package isn't available under WSUS, install [OS Build 14393.3115](#). If the package is available for all operating system builds, install [OS Builds 18362.1256](#) and [18363.1256](#).

## Contacting AWS Support

If you can't find troubleshooting solutions in this section or in the Systems Manager issues in [AWS re:Post](#), and you have a [Developer, Business, or Enterprise AWS Support plan](#), you can create a technical support case at [AWS Support](#).

Before you contact AWS Support, collect the following items:

- [SSM agent logs \(p. 132\)](#)
- Run Command command ID, maintenance window ID, or Automation execution ID
- For Windows Server managed nodes, also collect the following:
  - %PROGRAMDATA%\Amazon\PatchBaselineOperations\Logs as described on the **Windows** tab of [How patches are installed \(p. 1103\)](#)
  - Windows update logs: For Windows Server 2012 R2 and older, use %windir%\\WindowsUpdate.log. For Windows Server 2016 and newer, first run the PowerShell command [Get-WindowsUpdateLog](#) before using %windir%\\WindowsUpdate.log
- For Linux managed nodes, also collect the following:
  - The contents of the file /var/lib/amazon/ssm/*instance-id*/document/orchestration/[Run-Command-execution-id](#)/awsrunShellScript/PatchLinux

# AWS Systems Manager Distributor

Distributor, a capability of AWS Systems Manager, helps you package and publish software to AWS Systems Manager managed nodes. You can package and publish your own software or use Distributor to find and publish AWS-provided agent software packages, such as [AmazonCloudWatchAgent](#), or third-party packages such as [Trend Micro](#). Publishing a package advertises specific versions of the package's document to managed nodes that you identify using node IDs, AWS account IDs, tags, or an AWS Region. To get started with Distributor, open the [Systems Manager console](#). In the navigation pane, choose **Distributor**.

After you create a package in Distributor, you can install the package in one of the following ways:

- One time by using [AWS Systems Manager Run Command \(p. 991\)](#)
- On a schedule by using [AWS Systems Manager State Manager \(p. 1034\)](#)

### Important

Distributor doesn't guarantee that the third-party packages you install are free of any potential malware. We encourage that you conduct your own additional due diligence to ensure compliance with your internal security controls. Security is a shared responsibility between AWS and you. This is described as the shared responsibility model. To learn more, see the [shared responsibility model](#).

## How can Distributor benefit my organization?

Distributor offers these benefits:

- **One package, many platforms**

When you create a package in Distributor, the system creates an AWS Systems Manager document (SSM document). You can attach .zip files to this document. When you run Distributor, the system processes the instructions in the SSM document and installs the software package in the .zip file on the specified targets. Distributor supports multiple operating systems, including Windows, Ubuntu Server, Debian Server, and Red Hat Enterprise Linux. For more information about supported platforms, see [Supported package platforms and architectures \(p. 1254\)](#).

- **Control package access across groups of managed instances**

You can use Run Command or State Manager to control which of your managed nodes get a package and which version of that package. Run Command and State Manager are capabilities of AWS Systems Manager. Managed nodes can be grouped by instance or device 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 nodes. 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 managed nodes when those machines 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 Systems Manager managed nodes at one time.
- Software developers who create software packages.
- Administrators who are responsible for keeping Systems Manager managed nodes current with the most up-to-date software packages.

## What are the features of Distributor?

- **Deployment of packages to both Windows and Linux instances**

With Distributor, you can deploy software packages to Amazon Elastic Compute Cloud (Amazon EC2) instances and AWS IoT Greengrass core devices for Linux and Windows Server. For a list of supported instance operating system types, see [the section called "Supported package platforms and architectures" \(p. 1254\)](#).

**Note**

Distributor isn't supported on the macOS operating system.

- **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 Systems Manager console, AWS Command Line Interface (AWS CLI), AWS Tools for PowerShell, or the AWS SDK of your choice.

- **IAM access control**

By using AWS Identity and Access Management (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 \(p. 1284\)](#).

## 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 managed nodes require PowerShell scripts (scripts named `install.ps1` and `uninstall.ps1`). Linux-based managed nodes require shell scripts (scripts named `install.sh` and `uninstall.sh`). AWS Systems Manager 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 managed nodes.
- A JSON-formatted manifest file that describes the package contents. The manifest isn't included in the .zip file, but it's stored in the same Amazon Simple Storage Service (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 managed node attributes, such as operating system version or architecture. For information about how to create the manifest, see [Step 2: Create the JSON package manifest \(p. 1263\)](#).

When you choose **Simple** package creation in the Distributor console, Distributor generates the installation and un-installation 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

You can use Distributor to publish packages to the following Systems Manager managed node platforms. A version value must match the exact release version of the operating system Amazon Machine Image (AMI) that you're targeting. For more information about determining this version, see step 4 of [Step 2: Create the JSON package manifest \(p. 1263\)](#).

**Note**

Systems Manager doesn't support all of the following operating systems for AWS IoT Greengrass core devices. For more information, see [Setting up AWS IoT Greengrass core devices](#) in the [AWS IoT Greengrass Version 2 Developer Guide](#).

Platform	Code value in manifest file	Architecture
Windows Server	windows	x86_64 or 386
Debian Server	debian	x86_64 or 386
Ubuntu Server	ubuntu	x86_64 or 386  arm64 (Ubuntu Server 16 and later, A1 instance types)
Red Hat Enterprise Linux (RHEL)	redhat	x86_64 or 386  arm64 (RHEL 7.6 and later, A1 instance types)
CentOS	centos	x86_64 or 386
Amazon Linux and Amazon Linux 2	amazon	x86_64 or 386  arm64 (Amazon Linux 2, A1 instance types)
SUSE Linux Enterprise Server (SLES)	suse	x86_64 or 386
openSUSE	opensuse	x86_64 or 386
openSUSE Leap	opensuseleap	x86_64 or 386
Oracle Linux	oracle	x86_64

#### Topics

- [Setting up Distributor \(p. 1255\)](#)
- [Working with Distributor \(p. 1257\)](#)
- [Auditing and logging Distributor activity \(p. 1284\)](#)
- [Troubleshooting AWS Systems ManagerDistributor \(p. 1285\)](#)

## Setting up Distributor

Before you use Distributor, a capability of AWS Systems Manager, to create, manage, and deploy software packages, follow these steps.

#### Topics

- [Step 1: Complete Distributor prerequisites \(p. 1255\)](#)
- [Step 2: Verify or create an IAM instance profile with Distributor permissions \(p. 1256\)](#)
- [Step 3: Control user access to packages \(p. 1257\)](#)
- [Step 4: Create or choose an Amazon S3 bucket \(p. 1257\)](#)

## Step 1: Complete Distributor prerequisites

Before you use Distributor, a capability of AWS Systems Manager, be sure your environment meets the following requirements.

## Distributor prerequisites

Requirement	Description
SSM Agent	<p>AWS Systems Manager SSM Agent version 2.3.274.0 or later must be installed on the managed nodes on which you want to deploy or from which you want to remove packages.</p> <p>To install or update SSM Agent, see <a href="#">Working with SSM Agent (p. 68)</a>.</p>
AWS CLI	<p>(Optional) To use the AWS Command Line Interface (AWS CLI) instead of the Systems Manager console to create and manage packages, install the newest release of the AWS CLI on your local computer.</p> <p>For more information about how to install or upgrade the CLI, see <a href="#">Installing the AWS Command Line Interface</a> in the <i>AWS Command Line Interface User Guide</i>.</p>
AWS Tools for PowerShell	<p>(Optional) To use the Tools for PowerShell instead of the Systems Manager console to create and manage packages, install the newest release of Tools for PowerShell on your local computer.</p> <p>For more information about how to install or upgrade the Tools for PowerShell, see <a href="#">Setting Up the AWS Tools for Windows PowerShell or AWS Tools for PowerShell Core</a> in the <i>AWS Tools for PowerShell User Guide</i>.</p>

### Note

Systems Manager doesn't support distributing packages to Oracle Linux managed nodes 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 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. This requirement applies to permissions for all Systems Manager capabilities, not just Distributor, which is a capability of AWS Systems Manager.

### Note

When you configure your edge devices to run AWS IoT Greengrass Core software and SSM Agent, you specify an IAM service role that enables Systems Manager to perform actions on it. You don't need to configure managed edge devices with an instance profile.

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 \(p. 21\)](#).

## Step 3: Control user access to packages

Using AWS Identity and Access Management (IAM) policies, you can control who can create, deploy, and manage packages. You also control which Run Command and State Manager API operations they can perform on managed nodes. Like Distributor, both Run Command and State Manager, are capabilities of AWS Systems Manager.

### ARN Format

User-defined packages are associated with document Amazon Resource Names (ARNs) and have the following format.

```
arn:aws:ssm:region-id:account-id:document/document-name
```

The following is an example.

```
arn:aws:ssm:us-west-1:123456789012:document/ExampleDocumentName
```

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 AWS Systems Manager console, you choose an existing Amazon Simple Storage Service (Amazon S3) bucket to which Distributor uploads your software. Distributor is a capability of AWS Systems Manager. In the **Advanced** workflow, you must upload .zip files of your software or assets to an Amazon 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 Amazon 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 Systems Manager store. Because the assets are copied to an internal store, you can delete or repurpose your Amazon 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. Distributor is a capability of AWS Systems Manager. Before you add a package to Distributor:

- Create and zip installable assets.
- (Optional) Create a JSON manifest file for the package. This isn't 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 Amazon Simple Storage Service (Amazon S3) bucket ready to store your installable assets or software. If you're using the **Advanced** package creation process, upload your assets to the Amazon 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 nodes, see [Install or update packages \(p. 1274\)](#).

You can share Distributor packages between AWS accounts. When using a package shared from another account in AWS CLI commands use the package Amazon Resource Name (ARN) instead of the package name.

**Topics**

- [View packages \(p. 1258\)](#)
- [Create a package \(p. 1260\)](#)
- [Edit package permissions \(console\) \(p. 1270\)](#)
- [Edit package tags \(console\) \(p. 1270\)](#)
- [Add a package version to Distributor \(p. 1271\)](#)
- [Install or update packages \(p. 1274\)](#)
- [Uninstall a package \(p. 1280\)](#)
- [Delete a package \(p. 1281\)](#)

## View packages

To view packages that are available for installation, you can use the AWS Systems Manager console or your preferred AWS command line tool. Distributor is a capability of AWS Systems Manager. To access Distributor, open the AWS Systems Manager console and choose **Distributor** in the left navigation pane. You will see all of the packages available to you.

The following section describes how you can view Distributor packages using your preferred command line tool.

### View packages (command line)

This section contains information about how you can use your preferred command line tool to view Distributor packages using the provided commands.

Linux & macOS

#### To view packages using the AWS CLI on Linux

- To view all packages, excluding shared packages, run the following command.

```
aws ssm list-documents \
--filters Key=DocumentType,Values=Package
```

- To view all packages owned by Amazon, run the following command.

```
aws ssm list-documents \
--filters Key=DocumentType,Values=Package Key=Owner,Values=Amazon
```

- To view all packages owned by third parties, run the following command.

```
aws ssm list-documents \
--filters Key=DocumentType,Values=Package Key=Owner,Values=ThirdParty
```

## Windows

### To view packages using the AWS CLI on Windows

- To view all packages, excluding shared packages, run the following command.

```
aws ssm list-documents ^
--filters Key=DocumentType,Values=Package
```

- To view all packages owned by Amazon, run the following command.

```
aws ssm list-documents ^
--filters Key=DocumentType,Values=Package Key=Owner,Values=Amazon
```

- To view all packages owned by third parties, run the following command.

```
aws ssm list-documents ^
--filters Key=DocumentType,Values=Package Key=Owner,Values=ThirdParty
```

## PowerShell

### To view packages using the Tools for PowerShell

- To view all packages, excluding shared packages, run the following command.

```
$filter = New-Object Amazon.SimpleSystemsManagement.Model.DocumentKeyValuesFilter
$filter.Key = "DocumentType"
$filter.Values = "Package"

Get-SSMDocumentList ^
-Filters @($filter)
```

- To view all packages owned by Amazon, run the following command.

```
$typeFilter = New-Object Amazon.SimpleSystemsManagement.Model.DocumentKeyValuesFilter
$typeFilter.Key = "DocumentType"
$typeFilter.Values = "Package"

$ownerFilter = New-Object
Amazon.SimpleSystemsManagement.Model.DocumentKeyValuesFilter
$ownerFilter.Key = "Owner"
$ownerFilter.Values = "Amazon"

Get-SSMDocumentList ^
-Filters @($typeFilter,$ownerFilter)
```

- To view all packages owned by third parties, run the following command.

```
$typeFilter = New-Object Amazon.SimpleSystemsManagement.Model.DocumentKeyValuesFilter
$typeFilter.Key = "DocumentType"
$typeFilter.Values = "Package"
```

```
$ownerFilter = New-Object
 Amazon.SimpleSystemsManagement.Model.DocumentKeyValuesFilter
$ownerFilter.Key = "Owner"
$ownerFilter.Values = "ThirdParty"

Get-SSMDocumentList `
 -Filters @($typeFilter,$ownerFilter)
```

## 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're 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 \(p. 1254\)](#).

When you create a package, the system creates a new [SSM document \(p. 1287\)](#). This document allows you to deploy the package to managed nodes.

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 don't 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 don't zip installable assets in the **Simple** workflow.

### Topics

- [Create a package \(simple\) \(p. 1260\)](#)
- [Create a package \(advanced\) \(p. 1262\)](#)

## 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. Distributor is a capability of AWS Systems Manager. 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 \(p. 1287\)](#). For more information about supported platforms, see [Supported package platforms and architectures \(p. 1254\)](#).

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

Use the **In-place update** option to add new or updated files to an existing package installation without taking the associated application offline.

### To create a package (simple)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 can't 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 navigate to installable software files with .rpm, .msi, or .deb extensions. If the file name contains spaces, the upload fails. 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 isn't correct, choose the correct operating system from the dropdown 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 \(advanced\) \(p. 1262\)](#).
  - 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 \(p. 1263\)](#).
9. For **Platform version**, verify that the operating system platform version shown is either `_any`, a major release version followed by a wildcard (7.\*), or the exact 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 \(p. 1263\)](#).
  10. For **Architecture**, choose the correct processor architecture for each installable file from the dropdown list. For more information about supported processor architectures, see [Supported package platforms and architectures \(p. 1254\)](#).
  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 \(p. 1263\)](#).

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're 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 doesn't 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 can't 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 ManagerDistributor \(p. 1285\)](#).

## 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 Amazon 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 or directory, 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 \(p. 1262\)](#)
- [Step 2: Create the JSON package manifest \(p. 1263\)](#)
- [Step 3: Upload the package and manifest to an Amazon S3 bucket \(p. 1268\)](#)
- [Step 4: Add a package to Distributor \(p. 1268\)](#)

### 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 managed nodes require PowerShell scripts (scripts named `install.ps1` and `uninstall.ps1`). Linux-based managed nodes 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 doesn't take the package application offline while the update is performed. Windows Server-based managed nodes require a PowerShell script (script named

`update.ps1`). Linux-based managed nodes 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 \(p. 1274\)](#).

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.

```
{
 "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.

```
{ "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 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 \(p. 1254\)](#).

The `platform-version` can be the wildcard value, `_any`. Use it to indicate that a .zip file supports any release of the platform. You can also specify a major release version followed by a wildcard so all minor versions are supported, for example `7.*`. If you choose to specify a `platform-version` value for a specific operating system version, be sure that it matches the exact release version of the operating system AMI that you're targeting. The following are suggested resources for getting the correct value of the operating system.

- On a Windows Server-based managed nodes, the release version is available as Windows Management Instrumentation (WMI) data. You can run the following command from a command prompt to get version information, then parse the results for `version`. This command doesn't 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 managed node, 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 doesn't 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 Server, 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"
 }
 }
 },
 "another-platform": {
 "platform-version": {
 "architecture": {
 "file": ".zip-file-name-3.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.

```

{
 "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.

```

{
 "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 managed nodes 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.

```

{
 "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 7.x versions and Amazon Linux, you have two entries in the "packages" section that point to the same .zip file, as shown in the following example.

```

{
 "amazon": {
 "2018.03": {
 "x86_64": {
 "file": "test.zip"
 }
 }
 },
 "redhat": {
 "7.*": {
 "x86_64": {
 "file": "test.zip"
 }
 }
 }
},

```

- 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 `shasum -a 256 file-name.zip` or `openssl dgst -sha256 file-name.zip`. 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.

```

"files": {
 "test-agent-x86.deb.zip": {
 "checksums": {
 "sha256":
 "EXAMPLE2706223c7616ca9fb28863a233b38e5a23a8c326bb4ae241dcEXAMPLE"
 }
 },
 "test-agent-x86_64.deb.zip": {
 "checksums": {
 "sha256":
 "EXAMPLE572a745844618c491045f25ee6aae8a66307ea9bff0e9d1052EXAMPLE"
 }
 },
 "test-agent-x86_64.nano.zip": {
 "checksums": {
 "sha256":
 "EXAMPLE63ccb86e830b63dfef46995af6b32b3c52ce72241b5e80c995EXAMPLE"
 }
 },
 "test-agent-rhel5-x86.nano.zip": {
 "checksums": {

```

```

 "sha256":

 "EXAMPLE13df60aa3219bf117638167e5bae0a55467e947a363fff0a51EXAMPLE"

 }

},

"test-agent-x86.msi.zip": {

 "checksums": {

 "sha256":

 "EXAMPLE12a4abb10315aa6b8a7384cc9b5ca8ad8e9ced8ef1bf0e5478EXAMPLE"

 }

},

"test-agent-x86_64.msi.zip": {

 "checksums": {

 "sha256":

 "EXAMPLE63ccb86e830b63dfef46995af6b32b3c52ce72241b5e80c995EXAMPLE"

 }

},

"test-agent-rhel5-x86.rpm.zip": {

 "checksums": {

 "sha256":

 "EXAMPLE13df60aa3219bf117638167e5bae0a55467e947a363fff0a51EXAMPLE"

 }

},

"test-agent-rhel5-x86_64.rpm.zip": {

 "checksums": {

 "sha256":

 "EXAMPLE7ce8a2c471a23b5c90761a180fd157ec0469e12ed38a7094d1EXAMPLE"

 }

}
}

```

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.

```
{
 "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": "EXAMPLE2b8b9ed71e86f39f5946e837df0d38aacdd38955b4b18ffa6fEXAMPLE"
 }
}
}
```

### 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 Amazon 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 \(p. 1263\)](#) 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. Don't 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 or directory 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, don't 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 or directory 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 Distributor. When you add a package, you're adding a new [SSM document \(p. 1287\)](#). The document allows you to deploy the package to managed nodes.

#### Topics

- [Adding a package \(console\) \(p. 1268\)](#)
- [Adding a package \(AWS CLI\) \(p. 1269\)](#)

#### 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 Amazon S3 bucket \(p. 1268\)](#).

### To add a package to Distributor (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
  2. In the navigation pane, choose **Distributor**.
  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 Amazon S3 bucket" (p. 1268).
  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 Amazon S3 bucket with your .zip files.
- (Optional) If you didn't 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 \(p. 1263\)](#).
9. When you're 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 doesn't 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 Amazon S3 bucket \(p. 1268\)](#).

### 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. `DOC-EXAMPLE-BUCKET` is the URL of the Amazon 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 didn't add your manifest file to the Amazon S3 bucket, the `--content` parameter value is the file path to the JSON manifest file.

```
aws ssm create-document \
--name "package-name" \
--content file://path-to-manifest-file \
--attachments Key="SourceUrl",Values="DOC-EXAMPLE-BUCKET" \
--version-name version-value-from-manifest \
--document-type Package
```

The following is an example.

```
aws ssm create-document \
```

```
--name "ExamplePackage" \
--content file://path-to-manifest-file \
--attachments Key="SourceUrl",Values="https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/
ExamplePackage" \
--version-name 1.0.1 \
--document-type Package
```

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 add a package to Distributor, a capability of AWS Systems Manager, you can edit the package's permissions in the Systems Manager console. You can add other AWS accounts to a package's permissions. Packages can be shared with other accounts in the same AWS Region only. Cross-Region sharing isn't supported. 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.

### Note

You can update the permissions of packages that are shared with 20 or fewer accounts.

### To edit package permissions (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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're finished, choose **Save**.

## Edit package tags (console)

After you have added a package to Distributor, a capability of AWS Systems Manager, you can edit the package's tags in the Systems Manager console. These tags are applied to the package, and aren't connected to tags on the managed node on 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 don't want to add tags, you're ready to install your package or add a new version.

### To edit package tags (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Distributor**.

3. On the **Packages** page, choose the package for which you want to edit tags.
4. On the **Package details** tab, in **Tags**, choose **Edit**.
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're finished adding tags to your package, choose **Save**.

## Add a package version to Distributor

To add a package version, [create a package \(p. 1260\)](#), and then use Distributor to add a package version by adding an entry to the AWS Systems Manager (SSM) document that already exists for older versions. Distributor is a capability of AWS Systems Manager. 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 Amazon Simple Storage Service (Amazon S3) bucket, but must have a URL with a different file name shown at the end. You can use the Systems Manager console or the AWS Command Line Interface (AWS CLI) to add the new version. Uploading an installable file with the exact name as an existing installable file in the Amazon 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 Amazon 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 \(console\) \(p. 1271\)](#)
- [Adding a package version \(AWS CLI\) \(p. 1273\)](#)

### Adding a package version (console)

Before you perform these steps, follow the instructions in [Create a package \(p. 1260\)](#) to create a new package for the version. Then, use the 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)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.

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 can't 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**, navigate to the installable software files that you want to attach to the new version. Installable files from existing versions aren't 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 isn't correct, choose the correct operating system from the dropdown 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 \(advanced\) \(p. 1273\)](#).
  - 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 \(p. 1263\)](#).
10. For **Platform version**, verify that the operating system platform version shown is either `_any`, a major release version followed by a wildcard (`7.*`), or the exact 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 \(p. 1263\)](#).
  11. For **Architecture**, choose the correct processor architecture for each installable file from the dropdown list. For more information about supported architectures, see [Supported package platforms and architectures \(p. 1254\)](#).
  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 \(p. 1263\)](#).

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 doesn't 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 can't 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 ManagerDistributor \(p. 1285\)](#).
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 don't 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 \(p. 1260\)](#), 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)**

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 didn't upload your revised JSON manifest to the Amazon 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 \(p. 1263\)](#).

8. When you're 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 don't 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)

- 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 `DOC-EXAMPLE-BUCKET` with the URL of the JSON manifest that you copied in [Step 3: Upload the package and manifest to an Amazon S3 bucket \(p. 1268\)](#). `S3-bucket-URL-of-package` is the URL of the Amazon S3 bucket where the entire package is stored. Replace `version-name-from-updated-manifest` 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.

```
aws ssm update-document \
--name "document-name" \
--content "S3-bucket-URL-to-manifest-file" \
--attachments Key="SourceUrl",Values="DOC-EXAMPLE-BUCKET" \
--version-name version-name-from-updated-manifest \
--document-version $LATEST
```

The following is an example.

```
aws ssm update-document \
--name ExamplePackage \
--content "https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/ExamplePackage/
manifest.json" \
--attachments Key="SourceUrl",Values="https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/
ExamplePackage" \
--version-name 1.1.1 \
--document-version $LATEST
```

- 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.

```
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 deploy packages to your AWS Systems Manager managed nodes by using Distributor, a capability of AWS Systems Manager. To deploy the packages, use either the AWS Management Console or AWS Command Line Interface (AWS CLI). You can 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. We recommend using State Manager, a capability of AWS Systems Manager, to install packages. Using State Manager helps ensure that your managed nodes are always running the most up-to-date version of your package.

Preference	AWS Systems Manager action	More information
Install or update a package immediately.	Run Command	<ul style="list-style-type: none"> <li><a href="#">Installing or updating a package one time (console) (p. 1275)</a></li> </ul>

Preference	AWS Systems Manager action	More information
		<ul style="list-style-type: none"> <li>• <a href="#">Installing a package one time (AWS CLI) (p. 1278)</a></li> <li>• <a href="#">Updating a package one time (AWS CLI) (p. 1278)</a></li> </ul>
Install or update a package on a schedule, so that the installation always includes the default version.	State Manager	<ul style="list-style-type: none"> <li>• <a href="#">Scheduling a package installation or update (console) (p. 1277)</a></li> <li>• <a href="#">Scheduling a package installation (AWS CLI) (p. 1279)</a></li> <li>• <a href="#">Scheduling a package update (AWS CLI) (p. 1279)</a></li> </ul>
Automatically install a package on new managed nodes that have a specific tag or set of tags. For example, installing the Amazon CloudWatch agent on new instances.	State Manager	<p>One way to do this is to apply tags to new managed nodes, and then specify the tags as targets in your State Manager association. State Manager automatically installs the package in an association on managed nodes that have matching tags. See <a href="#">About targets and rate controls in State Manager associations (p. 1039)</a>.</p>

### Topics

- [Installing or updating a package one time \(console\) \(p. 1275\)](#)
- [Scheduling a package installation or update \(console\) \(p. 1277\)](#)
- [Installing a package one time \(AWS CLI\) \(p. 1278\)](#)
- [Updating a package one time \(AWS CLI\) \(p. 1278\)](#)
- [Scheduling a package installation \(AWS CLI\) \(p. 1279\)](#)
- [Scheduling a package update \(AWS CLI\) \(p. 1279\)](#)

## 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 \(p. 991\)](#), a capability of AWS Systems Manager, to perform the installation.

### To install or update a package one time (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 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 is complete.
  - **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. This option isn't supported for AWS published packages except the AWSEC2Launch-Agent package.
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, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or devices manually, or by specifying a resource group.

**Note**

If you don't see a managed node in the list, see [Troubleshooting managed node availability \(p. 816\)](#).

11. For **Other parameters**:

- For **Comment**, enter information about this command.
- For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

12. For **Rate Control**:

- For **Concurrency**, specify either a number or a percentage of targets on which to run the command at the same time.

**Note**

If you selected targets by specifying tags or resource groups and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 targets after it fails on either a number or a percentage of managed nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

15. When you're 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 a managed node 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 Amazon 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 \(p. 1034\)](#) to install or update.

### To schedule a package installation (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 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 is complete.
  - **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.

#### Note

You can choose managed AWS IoT Greengrass core devices by choosing either **Selecting all managed instances in this account** or **Manually Selecting Instance**.

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 [Creating associations \(p. 1042\)](#). 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 targets. For more information about the results of running associations, see [Creating associations \(p. 1042\)](#) in this guide.

For more information about working with the options in **Advanced options**, **Rate control**, and **Output options**, see [Creating associations \(p. 1042\)](#).

## 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.

```
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're installing a complete package.

The following is an example.

```
aws ssm send-command \
--document-name "AWS-ConfigureAWSPackage" \
--instance-ids "i-0000000000000000" \
--parameters '{"action":["Install"],"installationType":["Uninstall and
reinstall"],"name":["ExamplePackage"]}'
```

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*.

## Updating a package one time (AWS CLI)

You can run **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.

```
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.

```
aws ssm send-command \
```

```
--document-name "AWS-ConfigureAWSPackage" \
--instance-ids "i-02573cafccEXAMPLE" \
--parameters '{"action":["Install"],"installationType":["In-place update"],"name": \
["ExamplePackage"]}'
```

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*.

## 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 managed nodes. 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.

```
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're installing a complete package.

The following is an example.

```
aws ssm create-association \
--name "AWS-ConfigureAWSPackage" \
--parameters '{"action":["Install"],"installationType":["Uninstall and \
reinstall"],"name": ["Test-ConfigureAWSPackage"]}' \
--targets [{"Key": "InstanceIds", "Values": ["i-02573cafccEXAMPLE", \
"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.

```
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.

```
aws ssm create-association \
--name "AWS-ConfigureAWSPackage" \
--parameters '{"action":["Install"],"installationType":["In-place update"],"name":'["Test-ConfigureAWSPackage"]}' \
--targets [{"Key": "InstanceIds", "Values": ["i-02573cafefEXAMPLE", "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 Command Line Interface (AWS CLI) to uninstall Distributor packages from your AWS Systems Manager managed nodes by using Run Command. Distributor and Run Command are capabilities of AWS Systems Manager. 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 \(console\) \(p. 1280\)](#)
- [Uninstalling a package \(AWS CLI\) \(p. 1280\)](#)

## Uninstalling a package (console)

You can use Run Command in the Systems Manager console to uninstall a package one time. Distributor uses [AWS Systems Manager Run Command \(p. 991\)](#) to uninstall packages.

### To uninstall a package (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Run Command**.
3. On the Run Command home page, choose **Run command**.
4. Choose the `AWS-ConfigureAWSPackage` command document.
5. From **Action**, choose **Uninstall**.
6. For **Name**, enter the name of the package that you want to uninstall.
7. For **Targets**, choose how you want to target your managed nodes. You can specify a tag key and values that are shared by the targets. You can also specify targets by choosing attributes, such as an 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 Simple Notification Service (Amazon SNS) notifications. For more information, see [Running Commands from the Console](#) in this guide.
9. When you're 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 nodes by using Run Command.

## To uninstall a package (AWS CLI)

- Run the following command in the AWS CLI.

```
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.

```
aws ssm send-command \
 --document-name "AWS-ConfigureAWSPackage" \
 --instance-ids "i-02573cafefEXAMPLE" \
 --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

This section describes how to delete a package. You can't delete a version of a package, only the entire package.

### Deleting a package (console)

You can use the AWS Systems Manager console to delete a package or a package version from Distributor, a capability of AWS Systems Manager. Deleting a package deletes all versions of a package from Distributor.

#### To delete a package (console)

- Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
- In the navigation pane, choose **Distributor**.
- On the **Distributor** home page, choose the package that you want to delete.
- On the package's details page, choose **Delete package**.
- When you're prompted to confirm the deletion, choose **Delete package**.

### Deleting a package version (console)

You can use the Systems Manager console to delete a package version from Distributor.

#### To delete a package version (console)

- Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
- In the navigation pane, choose **Distributor**.
- On the **Distributor** home page, choose the package that you want to delete a version of.
- On the versions page for the package, choose the version to delete and choose **Delete version**.
- When you're 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 & macOS

### 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.

```
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.

```
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 shouldn't be included in the list.

```
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.

```
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.

```
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 shouldn't be included 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 = New-Object Amazon.SimpleSystemsManagement.Model.DocumentKeyValuesFilter
$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 shouldn't be included in the list.

```
$filter = New-Object Amazon.SimpleSystemsManagement.Model.DocumentKeyValuesFilter
$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 & macOS

### 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 shouldn't 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 shouldn't 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 shouldn't be found.

```
Get-SSMDocumentVersionList ^
-Name "package-name"
```

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

You can use AWS CloudTrail to audit activity related to Distributor, a capability of AWS Systems Manager. For more information about auditing and logging options for Systems Manager, see [Monitoring AWS Systems Manager \(p. 1428\)](#).

### Audit Distributor activity using CloudTrail

CloudTrail captures API calls made in the AWS Systems Manager console, the AWS Command Line Interface (AWS CLI), and the Systems Manager SDK. The information can be viewed in the CloudTrail console or stored in an Amazon Simple Storage Service (Amazon 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. Run Command and State Manager are capabilities of AWS Systems

Manager. For more information about viewing and using CloudTrail logs of Systems Manager activity, see [Logging AWS Systems Manager API calls with AWS CloudTrail \(p. 1442\)](#).

## Troubleshooting AWS Systems ManagerDistributor

The following information can help you troubleshoot problems that might occur when you use Distributor, a capability of AWS Systems Manager.

### Topics

- [Wrong package with the same name is installed \(p. 1285\)](#)
- [Error: Failed to retrieve manifest: Could not find latest version of package \(p. 1285\)](#)
- [Error: Failed to retrieve manifest: Validation exception \(p. 1285\)](#)
- [Package isn't supported \(package is missing install action\) \(p. 1286\)](#)
- [Error: Failed to download manifest : Document with name does not exist \(p. 1286\)](#)

### Wrong package with the same name is installed

**Problem:** You've installed a package, but Distributor installed a different package instead.

**Cause:** During installation, 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're using a version of SSM Agent with 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 \(p. 997\)](#) or [Walkthrough: Automatically update SSM Agent \(CLI\) \(p. 1089\)](#).

### 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're using a version of SSM Agent with 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 \(p. 997\)](#) or [Walkthrough: Automatically update SSM Agent \(CLI\) \(p. 1089\)](#).

## Package isn't supported (package is missing install action)

**Problem:** You received an error like the following.

```
Package is not supported (package is missing install action)
```

**Cause:** The package directory structure is incorrect.

**Solution:** Don't zip a parent directory containing the software and required scripts. Instead, create a .zip file of all the required contents directly in the absolute path. To verify the .zip file was created correctly, unzip the target platform directory and review the directory structure. For example, the install script absolute path should be */ExamplePackage\_targetPlatform/install.sh*.

## Error: Failed to download manifest : Document with name does not exist

**Problem:** You received an error like the following.

```
Failed to download manifest - failed to retrieve package document description:
InvalidDocument: Document with name filename does not exist.
```

**Cause:** Distributor can't find the package by the package name when sharing a Distributor package from another account.

**Solution:** When sharing a package from another account use the package Amazon Resource Name (ARN).

# AWS Systems Manager Shared Resources

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

## Topics

- [AWS Systems Manager documents \(p. 1287\)](#)

## 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 100 pre-configured documents that you can use by specifying parameters at runtime. You can find pre-configured documents in the Systems Manager Documents console by choosing the **Owned by Amazon** tab, or by specifying Amazon for the Owner filter when calling the `ListDocuments` API operation. Documents use JavaScript Object Notation (JSON) or YAML, and they include steps and parameters that you specify. To get started with SSM documents, open the [Systems Manager console](#). In the navigation pane, choose **Documents**.

## How can SSM documents benefit my organization?

SSM documents offers these benefits:

- **Document categories**

To help you find the documents you need, choose a category depending on the type of document you're searching for. To broaden your search, you can choose multiple categories of the same document type. Choosing categories of different document types is not supported. Categories are only supported for documents owned by Amazon.

- **Document versions**

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, AWS Command Line Interface (AWS CLI) commands, or API calls.

- **Customize documents for your needs**

If you want to customize the steps and actions in a document, you can create your own. The system stores the document with your AWS account in the AWS Region you create it in. For more information about how to create an SSM document, see [Creating SSM documents \(p. 1352\)](#).

- **Tag documents**

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 AWS Identity and Access Management (IAM) policy that specifies the tags that a user or group can access. For more information, see [Tagging Systems Manager documents \(p. 1506\)](#).

- **Share documents**

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 Amazon Elastic Compute Cloud (Amazon 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 \(p. 1361\)](#).

## Who should use SSM documents?

- Any AWS customer who wants to use Systems Manager capabilities to improve their operational efficiency at scale, reduce errors associated with manual intervention, and reduce time to resolution of common issues.
- Infrastructure experts who want to automate deployment and configuration tasks.
- Administrators who want to reliably resolve common issues, improve troubleshooting efficiency, and reduce repetitive operations.
- Users who want to automate a task they normally perform manually.

## What are the types of SSM documents?

The following table describes the different types of SSM documents and their uses.

Type	Use with	Details
ApplicationConfiguration	<a href="#">AWS AppConfig</a>	AWS AppConfig, a capability of AWS Systems Manager, enables you to create, manage, and quickly deploy application configurations. You can store configuration data in an SSM document by creating a document that uses the ApplicationConfiguration document type. For more information, see <a href="#">Freeform configurations</a> in the <a href="#">AWS AppConfig User Guide</a> .
ApplicationConfigurationSchema		If you create a configuration in an SSM document, then you must specify a corresponding JSON Schema. The schema uses the ApplicationConfigurationSchema document type and, like a set of rules, defines the allowable properties for each application configuration setting. For more information, see <a href="#">About validators</a> in the <a href="#">AWS AppConfig User Guide</a> .

Type	Use with	Details
Automation runbook	<a href="#">Automation (p. 397)</a> <a href="#">State Manager (p. 1034)</a> <a href="#">Maintenance Windows (p. 706)</a>	<p>Use Automation runbooks when performing common maintenance and deployment tasks such as creating or updating an Amazon Machine Image (AMI). State Manager uses Automation runbooks 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 runbooks to perform common maintenance and deployment tasks based on the specified schedule.</p> <p>All Automation runbooks that are supported for Linux-based operating systems are also supported on EC2 instances for macOS.</p>
Change Calendar document	<a href="#">Change Calendar (p. 695)</a>	<p>Change Calendar, a capability of AWS Systems Manager, uses the <code>ChangeCalendar</code> 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 <a href="#">iCalendar 2.0</a> data in plaintext format.</p> <p>Change Calendar isn't supported on EC2 instances for macOS.</p>

Type	Use with	Details
AWS CloudFormation template	<a href="#">AWS CloudFormation</a>	<p>AWS CloudFormation templates describe the resources that you want to provision in your CloudFormation stacks. By storing CloudFormation templates as Systems Manager documents, you can benefit from Systems Manager document features. These include creating and comparing multiple versions of your template, and sharing your template with other accounts in the same AWS Region.</p> <p>You can create and edit CloudFormation templates and stacks by using Application Manager, a capability of Systems Manager. For more information, see <a href="#">Working with AWS CloudFormation templates and stacks in Application Manager (p. 248)</a></p>

Type	Use with	Details
Command document	<a href="#">Run Command (p. 991)</a> <a href="#">State Manager (p. 1034)</a> <a href="#">Maintenance Windows (p. 706)</a>	<p>Run Command, a capability of AWS Systems Manager, uses Command documents to run commands. State Manager, a capability of AWS Systems 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, a capability of AWS Systems Manager, uses Command documents to apply a configuration based on the specified schedule.</p> <p>Most Command documents are supported on all Linux and Windows Server operating systems supported by Systems Manager. The following Command documents are supported on EC2 instances for macOS:</p> <ul style="list-style-type: none"> <li>• <code>AWS-ConfigureAWSPackage</code></li> <li>• <code>AWS-RunPatchBaseline</code></li> <li>• <code>AWS-RunPatchBaselineAssociation</code></li> <li>• <code>AWS-RunShellScript</code></li> </ul>
Package document	<a href="#">Distributor (p. 1252)</a>	<p>In Distributor, a capability of AWS Systems Manager, 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.</p> <p>Distributor isn't supported on Oracle Linux and macOS managed instances.</p>

Type	Use with	Details
Policy document	<a href="#">State Manager (p. 1034)</a>	<p>Inventory, a capability of AWS Systems Manager, uses the <code>AWS-GatherSoftwareInventory</code> Policy document with a State Manager association to collect inventory data from managed instances. When creating your own SSM documents, Automation runbooks and Command documents are the preferred method for enforcing a policy on a managed instance.</p> <p>Systems Manager Inventory and the <code>AWS-GatherSoftwareInventory</code> Policy document are supported on all operating systems supported by Systems Manager.</p>
Post-incident analysis template	<a href="#">Incident Manager post-incident analysis</a>	<p>Incident Manager uses the post-incident analysis template to create an analysis based on AWS operations management best practices.</p> <p>Use the template to create an analysis that your team can use to identify improvements to your incident response.</p>

Type	Use with	Details
Session document	<a href="#">Session Manager (p. 912)</a>	<p>Session Manager, a capability of AWS Systems 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.</p> <p>Session documents are supported on all Linux and Windows Server operating systems supported by Systems Manager. The following Command documents are supported on EC2 instances for macOS:</p> <ul style="list-style-type: none"> <li>• <code>AWS-PasswordReset</code></li> <li>• <code>AWS-StartInteractiveCommand</code></li> <li>• <code>AWS-StartPortForwardingSession</code></li> <li>• <code>AWS-StartPortForwardingSessionToSocket</code></li> <li>• <code>AWS-StartSSHSession</code></li> </ul>

### SSM document quotas

For information about SSM document quotas, see [Systems Manager service quotas](#) in the *Amazon Web Services General Reference*.

#### Topics

- [SSM document schema features and examples \(p. 1293\)](#)
- [SSM document syntax \(p. 1308\)](#)
- [Systems Manager Command document plugin reference \(p. 1314\)](#)
- [Viewing SSM Command document content \(p. 1351\)](#)
- [Creating SSM documents \(p. 1352\)](#)
- [Deleting custom SSM documents \(p. 1360\)](#)
- [Comparing SSM document versions \(p. 1361\)](#)
- [Sharing SSM documents \(p. 1361\)](#)
- [Searching for SSM documents \(p. 1371\)](#)
- [Running Systems Manager Command documents from remote locations \(p. 1373\)](#)

## SSM document schema features and examples

AWS Systems Manager (SSM) documents use the following schema versions.

- Documents of type `Command` can use schema version 1.2, 2.0, and 2.2. If you use 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

Feature	Details
Document editing	Documents can now be updated. With version 1.2, any update to a document required that you save it with a different name.
Automatic versioning	Any update to a document creates a new version. This isn't a schema version, but a version of the document.
Default version	If you have multiple versions of a document, you can specify which version is the default document.
Sequencing	Plugins or <code>steps</code> in a document run in the order that you specified.
Cross-platform support	Cross-platform support allows you to specify different operating systems for different plugins within the same SSM document. Cross-platform support uses the <code>precondition</code> parameter within a step.

#### Note

You must keep AWS Systems Manager 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 \(p. 997\)](#).

The following table lists the differences between major schema versions.

Version 1.2	Version 2.2 (latest version)	Details
<code>runtimeConfig</code>	<code>mainSteps</code>	In version 2.2, the <code>mainSteps</code> section replaces <code>runtimeConfig</code> . The <code>mainSteps</code> section allows Systems Manager to run steps in sequence.
<code>properties</code>	<code>inputs</code>	In version 2.2, the <code>inputs</code> section replaces the <code>properties</code> section. The <code>inputs</code> section accepts parameters for steps.
<code>commands</code>	<code>runCommand</code>	In version 2.2, the <code>inputs</code> section takes the <code>runCommand</code>

Version 1.2	Version 2.2 (latest version)	Details
		parameter instead of the commands parameter.
id	action	In version 2.2, Action replaces ID. This is just a name change.
not applicable	name	In version 2.2, name is any user-defined name for a step.

### 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 or to validate input parameters you've defined in your SSM document. The precondition parameter supports referencing your SSM document's input parameters, and platformType using values of Linux, MacOS, and Windows. Only the StringEquals operator is supported.

For documents that use schema version 2.2 or later, if precondition isn't specified, each plugin is either run or skipped based on the plugin's compatibility with the operating system. Plugin compatibility with the operating system is evaluated before the precondition. 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 isn't 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 isn't 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

```

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

```
{
```

```

"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 }}"
 }
 }
]
}

```

### Schema version 2.2 example

The following example uses the `aws:runPowerShellScript` plugin to run a PowerShell command on the target instances.

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

```
{
 "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 examples

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. You can also use the `precondition` parameter to validate input parameters you've defined in your SSM document. You can see this in the second of the following examples.

YAML

```

schemaVersion: '2.2'
description: cross-platform sample
mainSteps:
- action: aws:runPowerShellScript
 name: PatchWindows
 precondition:
 StringEquals:
 - platformType
 - Windows
 inputs:
 runCommand:
 - cmd
- action: aws:runShellScript
 name: PatchLinux
 precondition:
 StringEquals:
 - platformType
 - Linux
 inputs:
 runCommand:
 - cmd

```

JSON

```
{
 "schemaVersion": "2.2",
 "description": "cross-platform sample",
 "mainSteps": [
 {
 "action": "aws:runPowerShellScript",
 "name": "PatchWindows",
 "precondition": {
 "StringEquals": [
 "platformType",
 "Windows"
]
 },
 "inputs": {
 "runCommand": [
 "cmd"
]
 }
 }
]
}
```

```

 }
 },
{
 "action": "aws:runShellScript",
 "name": "PatchLinux",
 "precondition": {
 "StringEquals": [
 "platformType",
 "Linux"
]
 },
 "inputs": {
 "runCommand": [
 "cmds"
]
 }
}
]
}

```

#### YAML

```

schemaVersion: '2.2'
parameters:
 action:
 type: String
 allowedValues:
 - Install
 - Uninstall
 confirmed:
 type: String
 allowedValues:
 - True
 - False
mainSteps:
- action: aws:runShellScript
 name: InstallAwsCLI
 precondition:
 StringEquals:
 - "{{ action }}"
 - "Install"
 inputs:
 runCommand:
 - sudo apt install aws-cli
- action: aws:runShellScript
 name: UninstallAwsCLI
 precondition:
 StringEquals:
 - "{{ action }} {{ confirmed }}"
 - "Uninstall True"
 inputs:
 runCommand:
 - sudo apt remove aws-cli

```

#### JSON

```
{
 "schemaVersion": "2.2",
 "parameters": {
 "action": {
 "type": "String",
 "allowedValues": [

```

```

 "Install",
 "Uninstall"
],
},
"confirmed": {
 "type": "String",
 "allowedValues": [
 true,
 false
]
},
"mainSteps": [
{
 "action": "aws:runShellScript",
 "name": "InstallAwsCLI",
 "precondition": {
 "StringEquals": [
 "{{ action }}",
 "Install"
]
 },
 "inputs": {
 "runCommand": [
 "sudo apt install aws-cli"
]
 }
},
{
 "action": "aws:runShellScript",
 "name": "UninstallAwsCLI",
 "precondition": {
 "StringEquals": [
 "{{ action }} {{ confirmed }}",
 "Uninstall True"
]
 },
 "inputs": {
 "runCommand": [
 "sudo apt remove aws-cli"
]
 }
}
]
}

```

### Schema version 2.2 State Manager example

You can use the following SSM document with State Manager, a capability of Systems 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

```

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

```
{
 "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

```

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\nthe 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

```

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-z0-9][-.a-z0-9]{0,62})?:package\/\/[a-zA-Z][a-zA-Z0-9\-_]
{0,39}$|^@[a-zA-Z][a-zA-Z0-9\-_]{0,39}$"
 version:
 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

```
{
 "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-z0-9][-.a-z0-9]{0,62})?:package\/\/[a-zA-Z][a-zA-Z0-9\-_]
{0,39}$|^@[a-zA-Z][a-zA-Z0-9\-_]{0,39}$"
 },
 "version": {
 "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 }}"
 }
]
 }
}

```

## Schema version 1.2

The following example shows the top-level elements of a schema version 1.2 document.

```
{
 "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": "",
 "description": "(Optional) The path to the working directory on your instance.",
 "maxChars": 4096
 },
 "executionTimeout": {
 "type": "String",
 "default": "3600",
 "description": "(Optional) The execution timeout for the command in seconds. The value must be an integer between 1 and 3600. The default value is 3600."
 }
 }
}
```

```

 "description": "(Optional) The time in seconds for a command to complete before
it is considered to have failed. Default is 3600 (1 hour). Maximum is 172800 (48 hours).",
 "allowedPattern": "([1-9][0-9]{0,3})|(1[0-9]{1,4})|(2[0-7][0-9]{1,3})|(28[0-7]
[0-9]{1,2})|(28800)"
 }
},
"runtimeConfig": {
 "aws:runShellScript": {
 "properties": [
 {
 "id": "0.aws:runShellScript",
 "runCommand": "{{ commands }}",
 "workingDirectory": "{{ workingDirectory }}",
 "timeoutSeconds": "{{ executionTimeout }}"
 }
]
 }
}
}

```

## Schema version 0.3

### Top-level elements

The following example shows the top-level elements of a schema version 0.3 Automation runbook 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": ""
 }
 },
 "mainSteps": [
 {
 "name": "myStepName",
 "action": "action-name",
 "maxAttempts": 1,
 "inputs": {
 "Handler": "python-only-handler-name",
 "Runtime": "runtime-name",
 "Attachment": "script-or-zip-name"
 },
 "outputs": {
 "Name": "output-name",
 "Selector": "selector.value",
 "Type": "data-type"
 }
 }
],
 "files": {
 "script-or-zip-name": {
 "checksums": {
 "sha256": "checksum"
 }
 }
 }
}
```

```

 },
 "size": 1234
 }
}

```

### **YAML Automation runbook example**

The following sample shows the contents of an Automation runbook, 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 runbook 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:

Name | Type | Description | Default Value
----- | ----- | ----- | -----
assumeRole | String | (Optional) The ARN of the role that allows Automation to
perform the actions on your behalf. | -
imageId | String | (Optional) The AMI ID to use for launching the instance.
The default value uses the latest Amazon Linux AMI ID available. | {{{
ssm:/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2 }}}
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]['InstanceState']['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 recommend that you use schema version 2.2 or later for Command documents. Automation runbooks use schema version 0.3. Additionally, Automation runbooks 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 about 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. You can also use this field to specify whether a parameter requires a value for a document to run, or if providing a value for the parameter is optional. Required and optional parameters can be seen in the examples throughout this topic.

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 Parameter Store, a capability of AWS Systems Manager, 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. You can reference a Parameter Store parameter using the following format:

```
 {{ssm:parameter-name}}
```

#### YAML

```
AMI:
 type: String
 description: "(Optional) The AMI to use when launching the instance."
 default: "{{ssm:/aws/service/list/ami-windows-latest}}"
```

#### JSON

```
"AMI": {
 "type": "String",
 "description": "(Optional) The AMI to use when launching the instance.",
 "default": "{{ssm:/aws/service/list/ami-windows-latest}}"
}
```

For more information about Parameter Store, see [AWS Systems Manager Parameter Store \(p. 256\)](#).

Type: Structure

The parameters structure accepts the following fields and values:

- **type:** (Required) Allowed values include the following: String, StringList, Integer Boolean, MapList, and StringMap. To view examples of each type, see [SSM document parameter type examples \(p. 1311\)](#) in the next section.

**Note**

To use a number as a parameter value, use String as the parameter type.

- **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 isn't allowed, the execution fails to start.

YAML

```
DirectoryType:
 type: String
 description: "(Required) The directory type to launch."
 default: AwsMad
 allowedValues:
 - AdConnector
 - AwsMad
 - SimpleAd
```

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 doesn't match the allowed pattern, the execution fails to start.

**Note**

Systems Manager performs two validations for allowedPattern. The first validation is performed using the [Java regex library](#) at the API level when you use a document. The second validation is performed on SSM Agent by using the [GO regexp library](#) before processing the document.

YAML

```
InstanceId:
 type: String
 description: "(Required) The instance ID to target."
 allowedPattern: "^[i-][a-z0-9]{8,17}$"
 default: ''
```

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 aren't 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 \(p. 567\)](#).

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 runbooks support scripts for Python 3.6, Python 3.7, Python 3.8, PowerShell Core 6.0, and PowerShell 7.0. For more information about including scripts in Automation runbooks, see [Creating runbooks that run scripts \(p. 545\)](#) and [Walkthrough: Using Document Builder to create a custom runbook \(p. 648\)](#).

When creating an Automation runbook with attachments, you must also 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 Simple Storage Service (Amazon S3) buckets. For more information, see [Attachments in the AWS Systems Manager API Reference](#).

#### **YAML**

```

```

```
files:
 launch.py:
 checksums:
 sha256: 18871b1311b295c43d0f...[truncated]...772da97b67e99d84d342ef4aEXAMPLE
```

JSON

```
"files": {
 "launch.py": {
 "checksums": {
 "sha256": "18871b1311b295c43d0f...
 [truncated]...772da97b67e99d84d342ef4aEXAMPLE"
 }
 }
}
```

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's 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. This is also true when using parameters downstream within inputs for other plugins or AWS Systems Manager Automation actions. For example, you can't reference a `StringList` parameter within the `documentParameters` input of the `aws:runDocument` plugin. The `documentParameters` input accepts a map of strings even if the downstream SSM document parameter type is a `StringList` parameter and matches the parameter you're referencing.

When using parameters with 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`

```

InstanceId:
 type: String
 description: "(Optional) The target EC2 instance ID."
```

#### JSON

```
"InstanceId":{
 "type": "String",
 "description": "(Optional) The target EC2 instance ID."
}
```

#### StringList

A list of String items separated by commas. For example, ["cd ~", "pwd"].

#### 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. Doesn't 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

```

```

```
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
}
```

## Systems Manager Command document plugin reference

This reference describes the plugins that you can specify in an AWS Systems Manager (SSM) Command type document. These plugins can't be used in SSM Automation runbooks, which use Automation actions. For information about AWS Systems Manager Automation actions, see [Systems Manager Automation actions reference \(p. 477\)](#).

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 \(p. 1287\)](#).

### Note

Some of the plugins described here run only on either Windows Server instances or Linux instances. Platform dependencies are noted for each plugin.

The following document plugins are supported on Amazon Elastic Compute Cloud (Amazon EC2) instances for macOS:

- `aws:refreshAssociation`
- `aws:runShellScript`
- `aws:runPowerShellScript`
- `aws:softwareInventory`
- `aws:updateSsmAgent`

### Contents

- [Shared inputs \(p. 1315\)](#)
- [aws:applications \(p. 1317\)](#)
- [aws:cloudWatch \(p. 1318\)](#)
- [aws:configureDocker \(p. 1325\)](#)
- [aws:configurePackage \(p. 1326\)](#)
- [aws:domainJoin \(p. 1328\)](#)
- [aws:downloadContent \(p. 1330\)](#)
- [aws:psModule \(p. 1335\)](#)

- [aws:refreshAssociation \(p. 1337\)](#)
- [aws:runDockerAction \(p. 1338\)](#)
- [aws:runDocument \(p. 1340\)](#)
- [aws:runPowerShellScript \(p. 1341\)](#)
- [aws:runShellScript \(p. 1343\)](#)
- [aws:softwareInventory \(p. 1345\)](#)
- [aws:updateAgent \(p. 1347\)](#)
- [aws:updateSsmAgent \(p. 1349\)](#)

## Shared inputs

With SSM Agent version 3.0.502 and later only, all plugins can use the following inputs:

### **finallyStep**

The last step you want the document to run. If this input is defined for a step, it takes precedence over an `exit` value specified in the `onFailure` or `onSuccess` inputs. In order for a step with this input to run as expected, the step must be the last one defined in the `mainSteps` of your document.

Type: Boolean

Valid values: `true` | `false`

Required: No

### **onFailure**

If you specify this input for a plugin with the `exit` value and the step fails, the step status reflects the failure and the document doesn't run any remaining steps unless a `finallyStep` has been defined. If you specify this input for a plugin with the `successAndExit` value and the step fails, the step status shows successful and the document doesn't run any remaining steps unless a `finallyStep` has been defined.

Type: String

Valid values: `exit` | `successAndExit`

Required: No

### **onSuccess**

If you specify this input for a plugin and the step runs successfully, the document doesn't run any remaining steps unless a `finallyStep` has been defined.

Type: String

Valid values: `exit`

Required: No

### YAML

```

schemaVersion: '2.2'
description: Shared inputs example
parameters:
 customDocumentParameter:
 type: String
```

```
description: Example parameter for a custom Command-type document.
mainSteps:
- action: aws:runDocument
 name: runCustomConfiguration
 inputs:
 documentType: SSMDocument
 documentPath: "yourCustomDocument"
 documentParameters: '"documentParameter":{{customDocumentParameter}}'
 onSuccess: exit
- action: aws:runDocument
 name: ifConfigurationFailure
 inputs:
 documentType: SSMDocument
 documentPath: "yourCustomRepairDocument"
 onFailure: exit
- action: aws:runDocument
 name: finalConfiguration
 inputs:
 documentType: SSMDocument
 documentPath: "yourCustomFinalDocument"
 finallyStep: true
```

JSON

```
{
 "schemaVersion": "2.2",
 "description": "Shared inputs example",
 "parameters": {
 "customDocumentParameter": {
 "type": "String",
 "description": "Example parameter for a custom Command-type document."
 }
 },
 "mainSteps": [
 {
 "action": "aws:runDocument",
 "name": "runCustomConfiguration",
 "inputs": {
 "documentType": "SSMDocument",
 "documentPath": "yourCustomDocument",
 "documentParameters": "\"documentParameter\":{{customDocumentParameter}}",
 "onSuccess": "exit"
 }
 },
 {
 "action": "aws:runDocument",
 "name": "ifConfigurationFailure",
 "inputs": {
 "documentType": "SSMDocument",
 "documentPath": "yourCustomRepairDocument",
 "onFailure": "exit"
 }
 },
 {
 "action": "aws:runDocument",
 "name": "finalConfiguration",
 "inputs": {
 "documentType": "SSMDocument",
 "documentPath": "yourCustomFinalDocument",
 "finallyStep": true
 }
 }
]
}
```

## 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 \(p. 1287\)](#).

### Syntax

#### Schema 2.2

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

```
{
 "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

```

runtimeConfig:
 aws:applications:
 properties:
 - id: 0.aws:applications
 action: "{{ action }}"
 parameters: "{{ parameters }}"
 source: "{{ source }}"
```

```
sourceHash: "{{ sourceHash }}"
```

## 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 Elastic Compute Cloud (Amazon EC2), see [Sending Logs, Events, and Performance Counters to Amazon CloudWatch](#). For more information about documents, see [AWS Systems Manager documents \(p. 1287\)](#).

**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 node logs to CloudWatch Logs \(CloudWatch agent\) \(p. 1429\)](#)
- [Migrate Windows Server node log collection to the CloudWatch agent \(p. 1430\)](#)
- [Collecting metrics and logs from Amazon EC2 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 Amazon 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 turns on 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 AWS Systems Manager SSM Agent or the CloudWatch plugin is stopped, performance counter data isn't 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 }}"
 },
 "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 can't 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 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. The **div**, **div-MV**, **hu**, and **hu-HU** values aren't 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" to the CloudWatch destination defined under "Id": "CloudWatch", 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 | CloudWatch<sup>n</sup> | CloudWatchLogs<sup>n</sup>

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. Don't 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 informational messages uploaded.

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 aren't 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, select **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 select **Create Custom View**.
3. In the **Create Custom View** dialog box, select 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 is displayed on the **Log Groups > Streams for <YourLogStream>** screen in the CloudWatch console.

Required: Yes

#### MetricName

The CloudWatch metric that you want performance data to be included 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**

Turn on or turn off 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](#) 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

Valid values: Seconds | Microseconds | Milliseconds | Bytes | Kilobytes | Megabytes | Gigabytes | Terabytes | Bits | Kilobits | Megabits | Gigabits | Terabits | Percent | Count | Bytes/Second | Kilobytes/Second | Megabytes/Second | Gigabytes/Second | Terabytes/Second | Bits/Second | Kilobits/Second | Megabits/Second | Gigabits/Second | Terabits/Second | Count/Second | None

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 \(p. 1287\)](#).

### Syntax

#### Schema 2.2

##### 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 }}"
```

##### JSON

```
{
 "schemaVersion": "2.2",
 "description": "aws:configureDocker plugin",
 "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 an AWS Systems Manager 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`, `CodeDeployAgent`, and `AWSSupport-EC2Rescue`.

Available AWS packages for Linux operating systems include the following: `AmazonCloudWatchAgent`, `CodeDeployAgent`, and `AWSSupport-EC2Rescue`.

For more information, see [AWS Systems Manager documents \(p. 1287\)](#).

## Syntax

### Schema 2.2

#### 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 }}"
 additionalArguments:
 SSM_parameter_store_arg: "{{ ssm:parameter_store_arg }}"
 SSM_custom_arg: "myValue"
```

## JSON

```
{
 "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 }}",
 "additionalArguments": {
 "SSM_parameter_store_arg": "{{ ssm:parameter_store_arg }}",
 "SSM_custom_arg": "myValue"
 }
 }
 }
]
}
```

## Inputs

### **name**

The name of the AWS package to install or uninstall. Available packages include the following: `AWSVPDriver`, `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 is complete. 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. The `In-place` update option isn't supported for AWS-published packages. `Uninstall` and `reinstall` is the default value.

Type: `Enum`

Valid values: `Uninstall` and `reinstall` | `In-place` update

Required: No

#### additionalArguments

The additional parameters to provide to your install, uninstall, or update scripts. Each parameter must be prefixed with `SSM_`. You can reference a Parameter Store parameter in your additional arguments by using the convention `{ssm:parameter-name}`. To use the additional parameter in your install, uninstall, or update script, you must reference the parameter as an environment variable using the syntax appropriate for the operating system. For example, in PowerShell, you reference the `SSM_arg` argument as `$Env:SSM_arg`. There is no limit to the number of arguments you define, but the additional argument input has a 4096 character limit. This limit includes all of the keys and values you define.

Type: `StringMap`

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 runs on Linux and Windows Server operating systems. For more information about joining EC2 instances, see [Join an EC2 Instance to Your AWS Managed Microsoft AD Directory](#) in the *AWS Directory Service Administration Guide*. For more information about documents, see [AWS Systems Manager documents \(p. 1287\)](#).

### 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
 dnsIpAddresses:
 description: "(Required) The IP addresses of the DNS servers for your directory."
 type: StringList
```

```
mainSteps:
- action: aws:domainJoin
 name: domainJoin
 inputs:
 directoryId: "{{ directoryId }}"
 directoryName: "{{ directoryName }}"
 directoryOU: "{{ directoryOU }}"
 dnsIpAddresses: "{{ dnsIpAddresses }}"
```

JSON

```
{
 "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"
 },
 "dnsIpAddresses": {
 "description": "(Required) The IP addresses of the DNS servers for your
directory.",
 "type": "StringList"
 },
 },
 "mainSteps": [
 {
 "action": "aws:domainJoin",
 "name": "domainJoin",
 "inputs": {
 "directoryId": "{{ directoryId }}",
 "directoryName": "{{ directoryName }}",
 "directoryOU": "{{ directoryOU }}",
 "dnsIpAddresses": "{{ dnsIpAddresses }}"
 }
 }
]
}
```

## Schema 1.2

YAML

```

runtimeConfig:
 aws:domainJoin:
 properties:
 directoryId: "{{ directoryId }}"
 directoryName: "{{ directoryName }}"
 directoryOU: "{{ directoryOU }}"
 dnsIpAddresses: "{{ dnsIpAddresses }}"
```

JSON

```
{
 "runtimeConfig":{
 "aws:domainJoin":{
```

```
 "properties":{
 "directoryId":"{{ directoryId }}",
 "directoryName":"{{ directoryName }}",
 "directoryOU":"{{ directoryOU }}",
 "dnsIpAddresses":"{{ dnsIpAddresses }}"
 }
}
}
```

## 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: StringList

Required: Yes

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

```

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: StringMap
mainSteps:
- action: aws:downloadContent
 name: downloadContent
 inputs:
 sourceType: "{{ sourceType }}"
 sourceInfo: "{{ sourceInfo }}"
```

#### JSON

```
{
 "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": "StringMap"
 }
 },
 "mainSteps": [
 {
 "action": "aws:downloadContent",
 "name": "downloadContent",
 "inputs": {
 "sourceType": "{{ sourceType }}",
 "sourceInfo": "{{ sourceInfo }}"
 }
 }
]
}
```

## Inputs

### sourceType

The download source. Systems Manager supports the following source types for downloading scripts and SSM documents: GitHub, Git, HTTP, 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're using the latest commit in the master branch. If your repository was created after October 1, 2020 the default branch might be named main instead of master. In this case, you will need to specify values for the getOptions parameter.

This parameter uses the following format:

- branch:refs/heads/*branch\_name*

The default is master.

To specify a non-default branch use the following format:

branch:refs/remotes/origin/*branch\_name*

- 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:b7c1ddb94...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 aren't supported for any other fields, nor for any other SSM document plugins.

```
{
 "owner": "TestUser",
 "repository": "GitHubTest",
 "path": "scripts/python/test-script",
 "getOptions": "branch:master",
 "tokenInfo": "{{ssm-secure:secure-string-token}}"
}
```

##### For sourceType Git, you must specify the following:

- repository

The Git repository URL to the file or directory you want to download.

Type: String

Additionally, you can specify the following optional parameters:

- `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're using the latest commit in the master branch.

Type: String

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",
```

- `privateSSHKey`

The SSH key to use when connecting to the repository you specify. You can use the following format to reference a `SecureString` parameter for the value of your SSH key: `{{ssm-secure:your-secure-string-parameter}}`.

Type: String

- `skipHostKeyChecking`

Determines the value of the `StrictHostKeyChecking` option when connecting to the repository you specify. The default value is `false`.

Type: Boolean

- `username`

The username to use when connecting to the repository you specify using HTTP. You can use the following format to reference a `SecureString` parameter for the value of your username: `{{ssm-secure:your-secure-string-parameter}}`.

Type: String

- `password`

The password to use when connecting to the repository you specify using HTTP. You can use the following format to reference a `SecureString` parameter for the value of your password: `{{ssm-secure:your-secure-string-parameter}}`.

Type: String

**For sourceType `HTTP`, you must specify the following:**

- `url`

The URL to the file or directory you want to download.

Type: String

Additionally, you can specify the following optional parameters:

- allowInsecureDownload

Determines whether a download can be performed over a connection that isn't encrypted with Secure Socket Layer (SSL) or Transport Layer Security (TLS). The default value is `false`. We don't recommend performing downloads without encryption. If you choose to do so, you assume all associated risks. Security is a shared responsibility between AWS and you. This is described as the [shared responsibility model](#). To learn more, see the [shared responsibility model](#).

Type: Boolean

- authMethod

Determines whether a username and password are used for authentication when connecting to the `url` you specify. If you specify `Basic` or `Digest`, you must provide values for the `username` and `password` parameters. To use the `Digest` method, SSM Agent version 3.0.1181.0 or later must be installed on your instance. The `Digest` method supports MD5 and SHA256 encryption.

Type: String

Valid values: `None` | `Basic` | `Digest`

- username

The username to use when connecting to the `url` you specify using `Basic` authentication. You can use the following format to reference a `SecureString` parameter for the value of your `username`: `{{ssm-secure:your-secure-string-parameter}}`.

Type: String

- password

The password to use when connecting to the `url` you specify using `Basic` authentication. You can use the following format to reference a `SecureString` parameter for the value of your `password`: `{{ssm-secure:your-secure-string-parameter}}`.

Type: String

**For sourceType S3, specify the following:**

- path: The URL to the file or directory you want to download from Amazon S3.

```
{
 "path": "https://s3.amazonaws.com/doc-example-bucket/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.

```
{
 "name": "Example-RunPowerShellScript:3"
}
```

- name: The ARN for the document in the following format:  
`arn:aws:ssm:region:account_id:document/document_name`

```
{
```

```
 "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 Amazon EC2 instance. This plugin only runs on Windows Server operating systems. For more information, see [AWS Systems Manager documents \(p. 1287\)](#).

### 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.",
 "type": "String"
 }
 },
 "mainSteps": [
 {
 "action": "aws:psModule",
 "name": "psModule",
 "inputs": {
 "source": "{{ source }}"
 }
 }
]
}
```

## Schema 1.2

YAML

```

runtimeConfig:
 aws:psModule:
 properties:
 - runCommand: "{{ commands }}"
 source: "{{ source }}"
 sourceHash: "{{ sourceHash }}"
 workingDirectory: "{{ workingDirectory }}"
 timeoutSeconds: "{{ executionTimeout }}"
```

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

Required: No

### **timeoutSeconds**

The time in seconds for a command to be completed before it's 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 \(p. 1287\)](#).

### Syntax

#### Schema 2.2

##### 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

```
{
 "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 \(p. 1287\)](#).

### Syntax

#### Schema 2.2

##### 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

```
{
 "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

### **container**

The Docker container ID.

Type: String

Required: No

### **image**

The Docker image name.

Type: String

Required: No

### **cmd**

The container command.

Type: String

Required: No

### **memory**

The container memory limit.

Type: String

Required: No

### **cpuShares**

The container CPU shares (relative weight).

Type: String

Required: No

### **volume**

The container volume mounts.

Type: StringList

Required: No

### **env**

The container environment variables.

Type: String

Required: No

**user**

The container user name.

Type: String

Required: No

**publish**

The container published ports.

Type: String

Required: No

## 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 \(p. 1330\)](#) 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. This plugin doesn't support running the AWS-UpdateSSMAgent document or any document that uses the aws:updateSsmAgent plugin.

### Syntax

#### Schema 2.2

##### YAML

```

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 }}
```

##### 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 \(p. 1287\)](#).

## Syntax

### Schema 2.2

#### 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 }}"
]
 }
 }
]
}
```

## Schema 1.2

YAML

```

runtimeConfig:
 aws:runPowerShellScript:
 properties:
 - id: 0.aws:runPowerShellScript
 runCommand: "{{ commands }}"
 workingDirectory: "{{ workingDirectory }}"
 timeoutSeconds: "{{ executionTimeout }}"

```

JSON

```
{
 "runtimeConfig": {
 "aws:runPowerShellScript": {
 "properties": [
 {
 "id": "0.aws:runPowerShellScript",
 "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's 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 \(p. 1287\)](#).

### Syntax

#### Schema 2.2

##### 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

```
{
 "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

```

runtimeConfig:
 aws:runShellScript:
 properties:
 - runCommand: "{{ commands }}"
 workingDirectory: "{{ workingDirectory }}"
 timeoutSeconds: "{{ executionTimeout }}"
```

## 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's 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. When you configure inventory collection, you start by creating an AWS 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 the system returns the following error:

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

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. For more information about collecting inventory, see [AWS Systems Manager Inventory \(p. 848\)](#).

## Syntax

### Schema 2.2

#### 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

```
{
```

```
"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 \(p. 856\)](#).

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 \(p. 856\)](#).

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 \(p. 885\)](#)

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 \(p. 1287\)](#).

### **Syntax**

#### **Schema 2.2**

##### **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

```
{
 "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

```

runtimeConfig:
 aws:updateAgent:
 properties:
 agentName: Ec2Config
 source: https://s3.{Region}.amazonaws.com/aws-ssm-{Region}/manifest.json
 allowDowngrade: "{{ allowDowngrade }}"
 targetVersion: "{{ version }}"
```

##### 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

## aws :updateSsmAgent

Update the SSM Agent to the latest version or specify an older version. This plugin runs on Linux and Windows Server operating systems. For more information, see [Working with SSM Agent \(p. 68\)](#). For more information about documents, see [AWS Systems Manager documents \(p. 1287\)](#).

### Syntax

#### Schema 2.2

##### YAML

```

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

```
{
 "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.",
 "type": "String",
 "allowedValues": [
 "true",
 "false"
]
 }
 },
 "mainSteps": [
 {
 "action": "aws:updateSsmAgent",
 "name": "awsupdateSsmAgent",
 "inputs": {
 "agentName": "amazon-ssm-agent",
 "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/ssm-agent-manifest.json",
 "allowDowngrade": "{{ allowDowngrade }}"
 }
 }
]
}
```

#### Schema 1.2

##### YAML

```

runtimeConfig:
 aws:updateSsmAgent:
 properties:
 - agentName: amazon-ssm-agent
 source: https://s3.{Region}.amazonaws.com/aws-ssm-{Region}/manifest.json
 allowDowngrade: "{{ allowDowngrade }}"
```

##### JSON

```
{
 "runtimeConfig": {
 "aws:updateSsmAgent": {
 "properties": [
 {
 "agentName": "amazon-ssm-agent",
 "source": "https://s3.{Region}.amazonaws.com/aws-ssm-{Region}/manifest.json",
 "allowDowngrade": "{{ allowDowngrade }}"
 }
]
 }
 }
}
```

```
 }
}
```

## Properties

### agentName

amazon-ssm-agent. This is the name of the Systems Manager agent that processes requests and runs commands on the instance.

Type: String

Required: Yes

### allowDowngrade

Allow the SSM Agent to be downgraded to an earlier version. If set to false, the agent can be upgraded to newer versions only (default). If set to true, specify the earlier version.

Type: Boolean

Required: Yes

### source

The location where Systems Manager copies the SSM Agent version to install. You can't change this location.

Type: String

Required: Yes

### targetVersion

A specific version of SSM Agent to install. If not specified, the agent will be updated to the latest version.

Type: String

Required: No

## Viewing SSM Command document content

To preview the required and optional parameters for an AWS Systems Manager (SSM) Command document, in addition to the actions the document runs, you can view the content of the document in the Systems Manager console.

### To view SSM Command document content

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 search box, select **Document type**, and then select **Command**.

4. Choose the name of a document, and then choose the **Content** tab.
5. In the content field, review the available parameters and action steps for the document.

For example, the following image shows that (1) `version` and (2) `allowDowngrade` are optional parameters for the `AWS-UpdateSSMAgent` document, and that the first action run by the document is (3) `aws:updateSsmAgent`.

The screenshot shows the AWS-UpdateSSMAgent SSM document in the AWS Management Console. The 'Content' tab is selected. The document version is 1 (Default). The content is as follows:

```
1: {
2: "schemaVersion": "1.2",
3: "description": "Update the Amazon SSM Agent to the latest version or specified version.",
4: "parameters": {
5: "version": {
6: "default": "",
7: "description": "(Optional) A specific version of the Amazon SSM Agent to install. If not specified, the agent will be updated to the latest version available.",
8: "type": "String"
9: }
10: "allowDowngrade": {
11: "default": "false",
12: "description": "(Optional) Allow the Amazon SSM Agent service to be downgraded to an earlier version. If set to false, the agent will not be downgraded.",
13: "type": "String",
14: "allowedValues": [
15: "true",
16: "false"
17:]
18: }
19: },
20: "runtimeConfig": {
21: "aws:updateSsmAgent": {
22: "properties": [
23: {
24: "agentName": "amazon-ssm-agent",
25: "source": "https://s3-{Region}.amazonaws.com/amazon-ssm-{Region}/ssm-agent-manifest.json"
26: }
27:]
28: }
29: }
30: }
```

Annotations with red circles and numbers 1, 2, and 3 point to the `version` parameter, the `allowDowngrade` parameter, and the `aws:updateSsmAgent` action respectively.

## Creating SSM documents

If the AWS 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. You can also clone SSM documents using the console. Cloning documents copies content from an existing document to a new document that you can modify. 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 \(p. 1308\)](#)
- [SSM document schema features and examples \(p. 1293\)](#)
- [Systems Manager Command document plugin reference \(p. 1314\)](#)
- [Systems Manager Automation actions reference \(p. 477\)](#)
- [Automation system variables \(p. 531\)](#)

- Sample scenarios and custom runbook solutions ([p. 574](#))
- Working with Systems Manager Automation runbooks using the AWS Toolkit for Visual Studio Code
- Walkthrough: Using Document Builder to create a custom runbook ([p. 648](#))
- Creating runbooks that run scripts ([p. 545](#))

AWS pre-defined SSM documents might perform some of the actions you require. You can call these documents by using the `aws:ssm:runDocument`, `aws:ssm: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 & macOS**

```
latestDocVersion=$(aws ssm update-document \
--content file:///path/to/file/documentContent.json \
--name "ExampleDocument" \
--document-format JSON \
--document-version '$LATEST' \
| jq -r '.DocumentDescriptionLatestVersion')

aws ssm update-document-default-version \
--name "ExampleDocument" \
--document-version $latestDocVersion
```

**Windows**

```
latestDocVersion=$(aws ssm update-document ^
--content file://C:/path\to\file\documentContent.json ^
--name "ExampleDocument" ^
--document-format JSON ^
--document-version "$LATEST" ^
| jq -r '.DocumentDescriptionLatestVersion')

aws ssm update-document-default-version ^
--name "ExampleDocument" ^
--document-version $latestDocVersion
```

**PowerShell**

```
$content = Get-Content -Path "C:/path\to\file\documentContent.json" | Out-String
$latestDocVersion = Update-SSMDocument `
-Content $content `
-Name "ExampleDocument" `
-DocumentFormat "JSON" `
-DocumentVersion '$LATEST' `
| Select-Object -ExpandProperty LatestVersion

Update-SSMDocumentDefaultVersion `
-Name "ExampleDocument" `
-DocumentVersion $latestDocVersion
```

## Cloning an SSM document

You can clone AWS Systems Manager documents using the Systems Manager Documents console to create SSM documents. Cloning SSM documents copies content from an existing document to a new document that you can modify.

### To clone an SSM document

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 search box, enter the name of the document you want to clone.
4. Choose the name of the document you want to clone, and then choose **Clone document** in the **Actions** dropdown.
5. Modify the document as you prefer, and then choose **Create document** to save the document.

## Using SSM documents in State Manager Associations

If you create an SSM document for State Manager, a capability of AWS Systems Manager, you must associate the document with your managed instances after you add the document to the system. For more information, see [Creating associations \(p. 1042\)](#).

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 (for example, domain join and remove from domain), the last plugin run will be the final state. State Manager doesn't 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 won't 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 \(console\) \(p. 1355\)](#)
- [Create an SSM document \(command line\) \(p. 1355\)](#)
- [Create an SSM document \(API\) \(p. 1357\)](#)

- [Creating composite documents \(p. 1358\)](#)

## Create an SSM document (console)

After you create the content for your custom SSM document, as described in [Writing SSM document content \(p. 1352\)](#), you can use the Systems Manager console to create an SSM document using your content.

### To create an SSM document (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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. Enter 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 allow 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=AMAZON\_LINUX\_2
- Key=Environment, Value=Production

For more information about tagging Systems Manager resources, see [Tagging Systems Manager resources \(p. 1505\)](#).

9. Choose **Create document** to save the document.

## Create an SSM document (command line)

After you create the content for your custom AWS Systems Manager (SSM) document, as described in [Writing SSM document content \(p. 1352\)](#), you can use the AWS Command Line Interface (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 haven't already. For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

Linux & macOS

```
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 `n
 -Content $json `n
 -Name "document-name" `n
 -DocumentType "Command" `n
 -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 returns a response similar to the following.

```
{
 "DocumentDescription": {
 "CreatedDate": 1.585061751738E9,
 "DefaultVersion": "1",
 "Description": "MyCustomDocument",
 "DocumentFormat": "JSON",
 "DocumentType": "Command",
 "DocumentVersion": "1",
 "Hash": "0d3d879b3ca072e03c12638d0255ebd004d2c65bd318f8354fcde820dEXAMPLE",
 "HashType": "Sha256",
 "LatestVersion": "1",
 "Name": "Example",
 "Owner": "111122223333",
 "Parameters": [
 --truncated--
],
 "PlatformTypes": [
 "Windows",
 "Linux"
],
 "SchemaVersion": "0.3",
 "Status": "Creating",
 "Tags": [
 {
 "Key": "Purpose",
 "Value": "Test"
 }
]
 }
}
```

## Create an SSM document (API)

After you create the content for your custom AWS Systems Manager (SSM) document, as described in [Writing SSM document content \(p. 1352\)](#), you can use your preferred SDK to call the AWS Systems Manager `CreateDocument` API operation 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

```
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

```
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)
 sess := session.Must(session.NewSessionWithOptions(session.Options{
 SharedConfigState: session.SharedConfigEnable}))
}

ssmClient := ssm.New(sess)
createDocRequest, err := ssmClient.CreateDocument(&ssm.CreateDocumentInput{
 Content: &documentContent,
 Name: aws.String("createDocumentApiExample"),
 DocumentType: aws.String("Automation"),
 DocumentFormat: aws.String("YAML"),
})
result := *createDocRequest
fmt.Println(result)
```

```
}
```

Java

```
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.AWS Credentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagement;
import
com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementClientBuilder;
import com.amazonaws.services.simplesystemsmanagement.model.*;

public class createDocumentApiExample {
 public static void main(String[] args) {
 try {
 createDocumentMethod(getDocumentContent());
 }
 catch (IOException e) {
 e.printStackTrace();
 }
 }
 public static String getDocumentContent() throws IOException {
 String filepath = new String("/path/to/file/documentContent.yaml");
 byte[] encoded = Files.readAllBytes(Paths.get(filepath));
 String documentContent = new String(encoded, StandardCharsets.UTF_8);
 return documentContent;
 }

 public static void createDocumentMethod (final String documentContent) {
 AWSSimpleSystemsManagement ssm =
 AWSSimpleSystemsManagementClientBuilder.defaultClient();
 final CreateDocumentRequest createDocRequest = new CreateDocumentRequest()
 .withContent(documentContent)
 .withName("createDocumentApiExample")
 .withDocumentType("Automation")
 .withDocumentFormat("YAML");
 final CreateDocumentResult result = ssm.createDocument(createDocRequest);
 }
}
```

For more information about creating custom document content, see [SSM document syntax \(p. 1308\)](#).

## Creating composite documents

A *composite AWS Systems Manager (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 allowing 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 patches in the allow list.
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 AWS Systems Manager SSM Agent to the latest version.
2. The `aws:runDocument` plugin to run the legacy `AWS-ApplyPatchBaseline` document, which installs all allow listed 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. 1340) 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. 1330) 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.

```
{
 "schemaVersion": "2.2",
 "description": "My composite document for bootstrapping software and installing Docker.",
 "parameters": {},
 "mainSteps": [
 {
 "action": "aws:downloadContent",
 "name": "downloadContent",
 "inputs": {
 "sourceType": "GitHub",
 "sourceInfo": "{\"owner\":\"TestUser1\",\"repository\":\"TestPublic\", \"path\":\"documents/bootstrap/StateManagerBootstrap.yml\"}",
 "destinationPath": "bootstrap"
 }
 },
 {
 "action": "aws:runDocument",
 "name": "runDocument",
 "inputs": {}
 }
]
}
```

```
"inputs": {
 "documentType": "LocalPath",
 "documentPath": "bootstrap",
 "documentParameters": "{}"
},
{
 "action": "aws:runDocument",
 "name": "configureDocker",
 "inputs": {
 "documentType": "SSMDocument",
 "documentPath": "AWS-ConfigureDocker",
 "documentParameters": "{\"action\":\"Install\"}"
 }
}
]
```

## Related topics

- For information about rebooting servers and instances when using Run Command to call scripts, see [Rebooting managed nodes from scripts \(p. 1011\)](#).
- For more information about creating an SSM document, see [Creating SSM documents \(p. 1352\)](#).
- For more information about the plugins you can add to a custom SSM document, see [Systems Manager Command document plugin reference \(p. 1314\)](#).
- If you simply want to run a document from a remote location (without creating a composite document), see [Running Systems Manager Command documents from remote locations \(p. 1373\)](#).

## Deleting custom SSM documents

If you no longer want to use a custom SSM document, you can delete it by using either the AWS Command Line Interface (AWS CLI) or the AWS Systems Manager console.

### To delete an SSM document (AWS CLI)

1. Before you delete the document, we recommend that you disassociate all instances that are associated with the document.

Run the following command to disassociate an instance from a document.

```
aws ssm delete-association --instance-id "123456789012" --name "documentName"
```

There is no output if the command succeeds.

2. Run the following command.

Linux

```
aws ssm delete-document \
 --name "document-name" \
 --document-version "document-version" \
 --version-name "version-name"
```

Windows

```
aws ssm delete-document ^
 --name "document-name" ^
 --document-version "document-version" ^
```

```
--version-name "version-name"
```

#### PowerShell

```
Delete-SSMDocument
-Name "document-name"
-DocumentVersion 'document-version'
-VersionName 'version-name'
```

There is no output if the command succeeds.

#### Important

If the `document-version` or the `version-name` are not provided, all versions of the document are deleted.

### To delete an SSM document (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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. Select the document you want to delete.
4. Select **Delete**. When prompted to delete the document, select **Delete**.

## Comparing SSM document versions

You can compare the differences in content between versions of AWS Systems Manager (SSM) documents in the Systems Manager Documents console. When comparing versions of an SSM document, differences between the content of the versions are highlighted.

### To compare SSM document content (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 whose content you want to compare.
4. On the **Content** tab, select **Compare versions**, and choose the version of the document you want to compare the content to.

## Sharing SSM documents

You can share AWS Systems Manager (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 AWS account 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 \(p. 1362\)](#).

## 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 \(p. 1368\)](#).
- 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 isn't supported.

For more information about Systems Manager service quotas, see [AWS Systems Manager Service Quotas](#).

## Contents

- [Best practices for shared SSM documents \(p. 1362\)](#)
- [Block public sharing for SSM documents \(p. 1363\)](#)
- [Share an SSM document \(p. 1365\)](#)
- [Modify permissions for a shared SSM document \(p. 1368\)](#)
- [Using shared SSM documents \(p. 1369\)](#)

## Best practices for shared SSM documents

Review the following guidelines before you share or use a shared document.

### Remove sensitive information

Review your AWS Systems Manager (SSM) document carefully and remove any sensitive information. For example, verify that the document doesn't 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.

### Block public sharing for documents

Unless your use case requires public sharing to be turned on, we recommend turning on the block public sharing setting for your Systems Manager documents in the **Preferences** section of the Systems Manager Documents console.

### 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 Elastic Compute Cloud (Amazon EC2) console or by calling `ListDocuments` using the AWS Command Line Interface (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 \(p. 18\)](#) and [Customer managed policy examples \(p. 1395\)](#).

#### 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's 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're running a command from the correct Systems Manager document
- The content of the document hasn't changed since it was shared with you.

If the hash doesn't match the specified document or if the content of the shared document has changed, the command returns an `InvalidDocument` exception. The hash can't verify document content from external locations.

## Block public sharing for SSM documents

Unless your use case requires public sharing to be turned on, we recommend turning on the block public sharing setting for your AWS Systems Manager (SSM) documents. Turning on this setting prevents unwanted access to your SSM documents. The block public sharing setting is an account level setting that can differ for each AWS Region. Complete the following tasks to block public sharing for your SSM documents.

### Block public sharing (console)

#### To block public sharing of your SSM documents

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Preferences**, and then choose **Edit** in the **Block public sharing** section.
4. Select the **Block public sharing** check box, and then choose **Save**.

### Block public sharing (command line)

Open the AWS Command Line Interface (AWS CLI) or AWS Tools for Windows PowerShell on your local computer and run the following command to block public sharing of your SSM documents.

Linux & macOS

```
aws ssm update-service-setting \
--setting-id /ssm/documents/console/public-sharing-permission \
--setting-value Disable \
```

```
--region 'The AWS Region you want to block public sharing in'
```

#### Windows

```
aws ssm update-service-setting ^
--setting-id /ssm/documents/console/public-sharing-permission ^
--setting-value Disable ^
--region "The AWS Region you want to block public sharing in"
```

#### PowerShell

```
Update-SSMServiceSetting `
-SettingId /ssm/documents/console/public-sharing-permission `
-SettingValue Disable `
-Region The AWS Region you want to block public sharing in
```

Confirm the setting value was updated using the following command.

#### Linux & macOS

```
aws ssm get-service-setting \
--setting-id /ssm/documents/console/public-sharing-permission \
--region The AWS Region you blocked public sharing in
```

#### Windows

```
aws ssm get-service-setting ^
--setting-id /ssm/documents/console/public-sharing-permission ^
--region "The AWS Region you blocked public sharing in"
```

#### PowerShell

```
Get-SSMServiceSetting `
-SettingId /ssm/documents/console/public-sharing-permission `
-Region The AWS Region you blocked public sharing in
```

## Restricting access to block public sharing with IAM

You can create AWS Identity and Access Management (IAM) policies that restrict users from modifying the block public sharing setting. This prevents users from allowing unwanted access to your SSM documents.

The following is an example of an IAM policy that prevents users from updating the block public sharing setting. To use this example, you must replace the example Amazon Web Services account ID with your own account ID.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Deny",
 "Action": "ssm:UpdateServiceSetting",
 "Resource": "arn:aws:ssm:*:987654321098:servicesetting/ssm/documents/console/
public-sharing-permission"
 }
]
}
```

}

## Share an SSM document

You can share AWS Systems Manager (SSM) documents by using the Systems Manager console. You can also share SSM documents programmatically by calling the `ModifyDocumentPermission` API operation using the AWS Command Line Interface (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)

#### Share a document

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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're 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.

In the following command, replace `region` with your own information. For a list of supported `region` values, see the **Region** column in [Systems Manager service endpoints](#) in the *Amazon Web Services General Reference*.

Linux & macOS

```
aws config

AWS Access Key ID: [your key]
AWS Secret Access Key: [your key]
Default region name: region
Default output format [None]:
```

Windows

```
aws config

AWS Access Key ID: [your key]
AWS Secret Access Key: [your key]
Default region name: region
Default output format [None]:
```

PowerShell

```
Set-AWSCredentials -AccessKey your key -SecretKey your key
Set-DefaultAWSRegion -Region region
```

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 & macOS

```
aws ssm list-documents
```

Windows

```
aws ssm list-documents
```

PowerShell

```
Get-SSMDocumentList
```

3. Use the following command to get a specific document.

Linux & macOS

```
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 & macOS

```
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 & macOS

```
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 & macOS

```
aws ssm modify-document-permission \
--name document name \
--permission-type Share \
--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 & macOS

```
aws ssm modify-document-permission \
--name document name \
--permission-type Share \
--account-ids-to-add 'all'
```

Windows

```
aws ssm modify-document-permission ^
--name document name ^
```

```
--permission-type Share ^
--account-ids-to-add "all"
```

#### PowerShell

```
Edit-SSMDocumentPermission ^
-Name document name ^
-PermissionType Share ^
-AccountIdsToAdd ('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 AWS Systems Manager (SSM) document or delete the SSM document. However, you can't delete a document as long as it's shared. You must stop sharing it first and then delete it.

### Stop sharing a document (console)

#### Stop sharing a document

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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're 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 & macOS

```
aws ssm modify-document-permission \
--name document name \
--permission-type Share \
--account-ids-to-remove 'AWS account ID'
```

#### Windows

```
aws ssm modify-document-permission ^
--name document name ^
--permission-type Share ^
--account-ids-to-remove "AWS account ID"
```

## PowerShell

```
Edit-SSMDocumentPermission `
 -Name document name `
 -PermissionType Share `
 -AccountIdsToRemove AWS account ID
```

## Using shared SSM documents

When you share an AWS Systems Manager (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 Systems Manager console, you don't see the ARN. However, if you want to run a shared SSM document using a method other than the Systems Manager console, you must specify the full ARN of the document for the DocumentName request parameter. You're shown the full ARN for an SSM document when you run the command to list documents.

### Note

You aren't 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 & macOS

```
aws ssm list-documents \
 --filters Key=Owner,Values=Public
```

#### Windows

```
aws ssm list-documents ^
 --filters Key=Owner,Values=Public
```

#### PowerShell

```
$filter = New-Object Amazon.SimpleSystemsManagement.Model.DocumentKeyValuesFilter
$filter.Key = "Owner"
$filter.Values = "Public"

Get-SSMDocumentList `
 -Filters @($filter)
```

### To list private SSM documents that have been shared with you

#### Linux & macOS

```
aws ssm list-documents \
 --filters Key=Owner,Values=Private
```

#### Windows

```
aws ssm list-documents ^
 --filters Key=Owner,Values=Private
```

PowerShell

```
$filter = New-Object Amazon.SimpleSystemsManagement.Model.DocumentKeyValuesFilter
$filter.Key = "Owner"
$filter.Values = "Private"

Get-SSMDocumentList ^
 -Filters @($filter)
```

**To list all SSM documents available to you**

Linux & macOS

```
aws ssm list-documents
```

Windows

```
aws ssm list-documents
```

PowerShell

```
Get-SSMDocumentList
```

**To get information about an SSM document that has been shared with you**

Linux & macOS

```
aws ssm describe-document \
 --name arn:aws:ssm:us-east-2:12345678912:document/documentName
```

Windows

```
aws ssm describe-document ^
 --name arn:aws:ssm:us-east-2:12345678912:document/documentName
```

PowerShell

```
Get-SSMDocumentDescription ^
 -Name arn:aws:ssm:us-east-2:12345678912:document/documentName
```

**To run a shared SSM document**

Linux & macOS

```
aws ssm send-command \
 --document-name arn:aws:ssm:us-east-2:12345678912:document/documentName \
 --instance-ids ID
```

Windows

```
aws ssm send-command ^
 --document-name arn:aws:ssm:us-east-2:12345678912:document/documentName ^
 --instance-ids ID
```

## PowerShell

```
Send-SSMCommand ^
 -DocumentName arn:aws:ssm:us-east-2:12345678912:document/documentName ^
 -InstanceIds ID
```

# Searching for SSM documents

You can search the AWS Systems Manager (SSM) document store for SSM documents by using either free text search or a filter-based search. This section describes how to use both capabilities to find an SSM document.

## Using free text search

The search box on the Systems Manager **Documents** page supports free text search. Free text search compares the search term or terms that you enter against the document name in each SSM document. If you enter a single search term, for example **ansible**, then Systems Manager returns all SSM documents where this term was discovered. If you enter multiple search terms, then Systems Manager searches by using an OR statement. For example, if you specify **ansible** and **linux**, then search returns all documents with *either* keyword in their name.

If you enter a free text search term and choose a search option, such as **Platform type**, then search uses an AND statement and returns all documents with the keyword in their name and the specified platform type.

### Note

Note the following details about free text search.

- Free text search is *not* case sensitive.
- Search terms require a minimum of three characters and have a maximum of 20 characters.
- Free text search accepts up to five search terms.
- If you enter a space between search terms, the system includes the space when searching.
- You can combine free text search with other search options such as **Document type** or **Platform type**.
- The **Document Name Prefix** filter and free text search can't be used together. They're mutually exclusive.

### To search for an SSM document

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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. Enter your search terms in the search box, and press Enter.

## Performing free text document search by using the AWS CLI

### To perform a free text document search by using the CLI

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. To perform free text document search with a single term, run the following command. In this command, replace `search_term` with your own information.

```
aws ssm list-documents --filters Key="SearchKeyword",Values="search_term"
```

Here's an example.

```
aws ssm list-documents --filters Key="SearchKeyword",Values="aws-asg" --region us-east-2
```

To search using multiple terms that create an AND statement, run the following command. In this command, replace `search_term_1` and `search_term_2` with your own information.

```
aws ssm list-documents --filters Key="SearchKeyword",Values="search_term_1","search_term_2","search_term_3" --region us-east-2
```

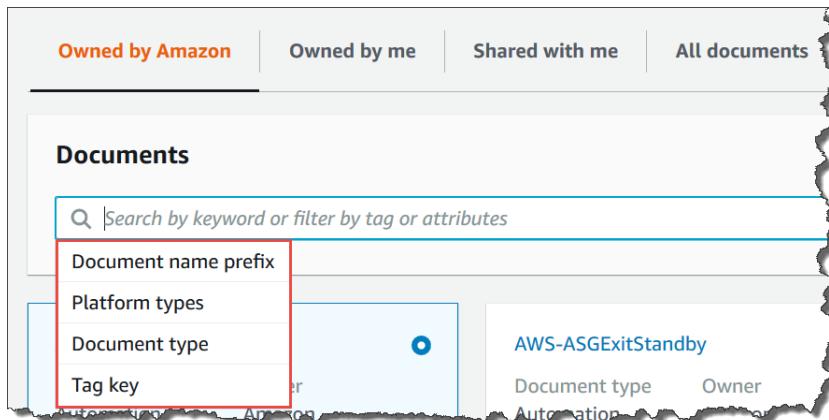
Here's an example.

```
aws ssm list-documents --filters Key="SearchKeyword",Values="aws-asg","aws-ec2","restart" --region us-east-2
```

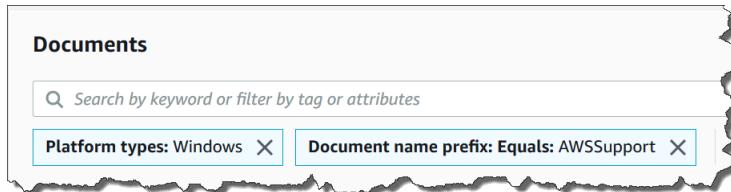
## Using filters

The Systems Manager **Documents** page automatically displays the following filters when you choose the search box.

- Document name prefix
- Platform types
- Document type
- Tag key



You can search for SSM documents by using a single filter. If you want to return a more specific set of SSM documents, you can apply multiple filters. Here is an example of a search that uses the **Platform types** and the **Document name prefix** filters.



If you apply multiple filters, Systems Manager creates different search statements based on the filters you choose:

- If you apply the *same* filter multiple times, for example **Document name prefix**, then Systems Manager searches by using an OR statement. For example, if you specify one filter of **Document name prefix=AWS** and a second filter of **Document name prefix=Lambda**, then search returns all documents with the prefix "AWS" and all documents with the prefix "Lambda".
- If you apply *different* filters, for example **Document name prefix** and **Platform types**, then Systems Manager searches by using an AND statement. For example, if you specify a **Document name prefix=AWS** filter and a **Platform types=Linux** filter, then search returns all documents with the prefix "AWS" that are specific to the Linux platform.

**Note**

Searches that use filters are case sensitive.

## Running Systems Manager Command documents from remote locations

You can run AWS Systems Manager (SSM) documents from remote locations by using the `AWS-RunDocument` pre-defined SSM document. This document supports running SSM documents stored in the following locations:

- GitHub repositories (public and private)
- Amazon S3 buckets
- Systems Manager

While you can also run remote documents by using State Manager or Automation, capabilities of AWS Systems Manager, the following procedure describes only how to run remote SSM documents by using AWS Systems Manager Run Command in the Systems Manager console.

**Note**

`AWS-RunDocument` can be used to run only command-type SSM documents, not other types such as Automation runbooks. The `AWS-RunDocument` uses the `aws:downloadContent` plugin. For more information about the `aws:downloadContent` plugin, see [aws:downloadContent \(p. 1330\)](#).

### Before you begin

Before you run a remote document, you must complete the following tasks.

- Create an SSM Command document and save it in a remote location. For more information, see [Creating SSM documents \(p. 1352\)](#)
- 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 \(p. 279\)](#).

## Run a remote document (console)

### To run a remote document

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 **Document** list, choose **AWS-RunDocument**.
5. In **Command parameters**, for **Source Type**, choose an option.

- If you choose **GitHub**, specify **Source Info** information in the following format:

```
{
 "owner": "owner_name",
 "repository": "repository_name",
 "path": "path_to_document",
 "getOptions": "branch:branch_name",
 "tokenInfo": "{{ssm-secure:secure-string-token}}"
}
```

For example:

```
{
 "owner": "TestUser",
 "repository": "GitHubTestExamples",
 "path": "scripts/python/test-script",
 "getOptions": "branch:exampleBranch",
 "tokenInfo": "{{ssm-secure:my-secure-string-token}}"
}
```

### Note

`getOptions` are 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. The `branch` parameter 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:b8c1ddb94...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/DOC-EXAMPLE-BUCKET/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, enter 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, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

8. For **Other parameters**:

- For **Comment**, enter information about this command.
- For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

9. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different

AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

12. Choose **Run**.

**Note**

For information about rebooting servers and instances when using Run Command to call scripts, see [Rebooting managed nodes from scripts \(p. 1011\)](#).

# Security in AWS Systems Manager

Cloud security at Amazon Web Services 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're 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 AWS 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 \(p. 1377\)](#)
- [Identity and access management for AWS Systems Manager \(p. 1380\)](#)
- [Using service-linked roles for Systems Manager \(p. 1410\)](#)
- [Logging and monitoring in AWS Systems Manager \(p. 1421\)](#)
- [Compliance validation for AWS Systems Manager \(p. 1422\)](#)
- [Resilience in AWS Systems Manager \(p. 1423\)](#)
- [Infrastructure security in AWS Systems Manager \(p. 1423\)](#)
- [Configuration and vulnerability analysis in AWS Systems Manager \(p. 1424\)](#)
- [Security best practices for Systems Manager \(p. 1424\)](#)

## Data protection in AWS Systems Manager

Data protection refers to protecting data while *in transit* (as it travels to and from Systems Manager) and *at rest* (while it's stored in AWS data centers).

The AWS [shared responsibility model](#) applies to data protection in AWS Systems Manager. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the [AWS Security Blog](#).

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). That way 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. We recommend TLS 1.2 or later.
- 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.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or 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 tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

## Data encryption

### Encryption at rest

#### Parameter Store parameters

The types of parameters you can create in Parameter Store, a capability of AWS Systems Manager, include `String`, `StringList`, and `SecureString`.

To encrypt `SecureString` parameter values, Parameter Store uses an AWS KMS key in AWS Key Management Service (AWS KMS). AWS KMS uses either a customer managed key or an AWS managed key to encrypt the parameter value in an AWS managed database.

#### Important

Don't 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 [What is a parameter? \(p. 258\)](#) and [Restricting access to Systems Manager parameters using IAM policies \(p. 260\)](#).

#### 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 User 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 commands in Run Command, a capability of AWS Systems Manager
- Packages in Distributor, a capability of AWS Systems Manager
- Patching operation logs in Patch Manager, a capability of AWS Systems Manager
- Patch Manager patch override lists
- Scripts or Ansible Playbooks to run in a runbook workflow in Automation, a capability of AWS Systems Manager
- Chef InSpec profiles for use with scans in Compliance, a capability of AWS Systems Manager
- AWS CloudTrail logs

- Session history logs in Session Manager, a capability of AWS Systems Manager
- Reports from Explorer, a capability of AWS Systems Manager
- OpsData from OpsCenter, a capability of AWS Systems Manager
- AWS CloudFormation templates for use with Automation workflows
- Compliance data from a resource data sync scan
- Output of requests to create or edit association in State Manager, a capability of AWS Systems Manager, on managed nodes
- Custom Systems Manager documents (SSM documents) that you can run using the AWS managed SSM document `AWS-RunDocument`

### 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 using AWS Key Management Service](#) 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 runbook
- Session Manager session history logs
- Logs from SSM Agent on your managed nodes

## 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 nodes.

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 [AWS service endpoints](#) in the *Amazon Web Services General Reference*.

### Managed instances

AWS provides secure and private connectivity between Amazon Elastic Compute Cloud (Amazon EC2) instances. In addition, we automatically encrypt in-transit traffic between supported instances in the same virtual private cloud (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 an AWS KMS key that has been created in AWS KMS. AWS KMS encryption is available for `Standard_Stream`, `InteractiveCommands`, and `NonInteractiveCommands` session types.

### Run Command access

By default, remote access to your nodes 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 nodes and control traffic between them, your on-premises network, and the internet. For details, see [\(Optional\) Create a VPC endpoint \(p. 28\)](#).

For more information about Amazon Virtual Private Cloud security, see [Internetwork traffic privacy in Amazon VPC](#) 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 \(p. 1380\)](#)
- [Authenticating with identities \(p. 1381\)](#)
- [Managing access using policies \(p. 1382\)](#)
- [How AWS Systems Manager works with IAM \(p. 1384\)](#)
- [AWS Systems Manager identity-based policy examples \(p. 1391\)](#)
- [AWS managed policies for AWS Systems Manager \(p. 1398\)](#)
- [Troubleshooting AWS Systems Manager identity and access \(p. 1408\)](#)

## Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that 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 \(p. 1408\)](#).

**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 \(p. 1384\)](#).

**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 \(p. 1391\)](#).

## 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 [Signing in to the AWS Management Console as an IAM user or root user](#) 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 address or your IAM user name. You can access AWS programmatically using your root user or IAM users 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.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an *identity provider*. For more information about federated users, see [Federated users and roles](#) in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see [Actions, Resources, and Condition Keys for AWS Systems Manager](#) in the *Service Authorization Reference*.
  - **Service role** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform 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.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see [When to create an IAM role \(instead of a user\)](#) in the *IAM User Guide*.

## 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. You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. 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*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

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, group of users, or role. These policies control what actions users and roles 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 \(p. 1390\)](#).

## Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

## Access control lists (ACLs)

Access control lists (ACLs) control 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 AWS Identity and Access Management (IAM) to manage access to AWS 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. 1384)
- [Systems Manager resource-based policies](#) (p. 1388)
- [Authorization based on Systems Manager tags](#) (p. 1388)
- [Systems Manager IAM roles](#) (p. 1389)

## Systems Manager identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources and 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

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The **Action** element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions 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 (SSM 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"]
```

**Note**

AWS AppConfig, a capability of AWS Systems Manager, uses the prefix `appconfig:` before actions.

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 *Service Authorization Reference*.

## Resources

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its [Amazon Resource Name \(ARN\)](#). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (\*) to indicate that the statement applies to all resources.

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

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 in the US East (Ohio) Region, you would 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 (\*).

```
"Resource": "arn:aws:ssm:region:123456789012:maintenancewindow/*"
```

For Parameter Store API operations, 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.

```
"Resource": "arn:aws:ssm:region:123456789012:parameter/Dev/ERP/Oracle/*"
```

Some Systems Manager actions, such as those for creating resources, can't be performed on a specific resource. In those cases, you must use the wildcard (\*).

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

Some Systems Manager API operations 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.

Resource type	ARN format
Application (AWS AppConfig)	arn:aws:appconfig: <b>region</b> : <b>account-id</b> :application/ <b>application-id</b>
Association	arn:aws:ssm: <b>region</b> : <b>account-id</b> :association/ <b>association-id</b>
Automation execution	arn:aws:ssm: <b>region</b> : <b>account-id</b> :automation-execution/ <b>automation-execution-id</b>
Automation definition (with version subresource)	arn:aws:ssm: <b>region</b> : <b>account-id</b> :automation-definition/ <b>automation-definition-id</b> : <b>version-id</b> 
Configuration profile (AWS AppConfig)	arn:aws:appconfig: <b>region</b> : <b>account-id</b> :application/ <b>application-id</b> /configurationprofile/ <b>configurationprofile-id</b>
Contact (Incident Manager)	arn:aws:ssm-contacts: <b>region</b> : <b>account-id</b> :contact/ <b>contact-alias</b>
Deployment strategy (AWS AppConfig)	arn:aws:appconfig: <b>region</b> : <b>account-id</b> :deploymentsstrategy/ <b>deploymentstrategy-id</b>
Document	arn:aws:ssm: <b>region</b> : <b>account-id</b> :document/ <b>document-name</b>
Environment (AWS AppConfig)	arn:aws:appconfig: <b>region</b> : <b>account-id</b> :application/ <b>application-id</b> /environment/ <b>environment-id</b>
Incident	arn:aws:ssm-incidents: <b>region</b> : <b>account-id</b> :incident-record/ <b>response-plan-name</b> / <b>incident-id</b>
Maintenance window	arn:aws:ssm: <b>region</b> : <b>account-id</b> :maintenancewindow/ <b>window-id</b>
Managed node	arn:aws:ssm: <b>region</b> : <b>account-id</b> :managed-instance/ <b>managed-node-id</b>
Managed node inventory	arn:aws:ssm: <b>region</b> : <b>account-id</b> :managed-instance-inventory/ <b>managed-node-id</b>

Resource type	ARN format
OpsItem	arn:aws:ssm: <i>region</i> : <i>account-id</i> :opsitem/ <i>OpsItem-id</i>
Parameter	<p>A one-level parameter:</p> <ul style="list-style-type: none"> <li>• arn:aws:ssm:<i>region</i>:<i>account-id</i>:parameter/<i>parameter-name</i>/</li> </ul> <p>A parameter named with a hierarchical construction:</p> <ul style="list-style-type: none"> <li>• arn:aws:ssm:<i>region</i>:<i>account-id</i>:parameter/<i>parameter-name-root/level-2/level-3/level-4/level-5</i> ②</li> </ul>
Patch baseline	arn:aws:ssm: <i>region</i> : <i>account-id</i> :patchbaseline/ <i>patch-baseline-id</i>
Response plan	arn:aws:ssm-incidents: <i>region</i> : <i>account-id</i> :response-plan/ <i>response-plan-name</i>
Session	arn:aws:ssm: <i>region</i> : <i>account-id</i> :session/ <i>session-id</i> ③
All Systems Manager resources	arn:aws:ssm:*
All Systems Manager resources owned by the specified AWS account in the specified AWS Region	arn:aws:ssm: <i>region</i> : <i>account-id</i> :*

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 allows you to 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 node 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 \(p. 279\)](#).

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 isn't 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 *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see [Actions Defined by AWS Systems Manager](#).

## Condition keys

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on **what resources**, and under **what conditions**.

The Condition element (or Condition *block*) lets you specify conditions in which a statement is in effect. The Condition element is optional. You can create 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*.

AWS supports global condition keys and service-specific 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 *Service Authorization Reference*. 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:

- [Restricting access to root-level commands through SSM Agent \(p. 134\)](#)
- [Restricting Run Command access based on tags \(p. 993\)](#)
- [Restrict session access based on instance tags \(p. 936\)](#)

For information about using the `ssm:Recursive` and `ssm:Overwrite` condition keys, see [Working with parameter hierarchies \(p. 326\)](#).

## Examples

To view examples of Systems Manager identity-based policies, see [AWS Systems Manager identity-based policy examples \(p. 1391\)](#).

## Systems Manager resource-based policies

Other AWS services, such as Amazon Simple Storage Service (Amazon S3), 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 doesn't 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 node
- Maintenance window
- Parameter
- Patch baseline
- OpsItem

For information about tagging Systems Manager resources, see [Tagging Systems Manager resources \(p. 1505\)](#).

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 \(p. 1398\)](#).

## 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 Security Token Service (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 are listed 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 \(p. 1410\)](#).

### 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 are displayed 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 nodes, you must choose a role to allow Systems Manager to access nodes 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 nodes.

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 \(p. 21\)](#).

To access on-premises nodes 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 \(p. 36\)](#).

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 don't 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 automations \(p. 401\)](#).

## 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 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 operations, such as `Get*` and `List*`.
- **AmazonSSMAutomationApproverAccess** – User trust policy that allows 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 Systems Manager Automation service to run activities defined within Automation runbooks. Assign this policy to administrators and trusted power users.
- **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 node.
- **AmazonSSMMaintenanceWindowRole** – Service role policy for Systems Manager Maintenance Windows.
- **AmazonSSMManagedInstanceCore** – Instance trust policy that allows a node to use Systems Manager service core functionality.
- **AmazonSSMServiceRolePolicy** – Service role policy that provides access to AWS resources managed or used by 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.
- **AWSSystemsManagerChangeManagementServicePolicy** – Service policy that provides access to AWS resources managed or used by the Systems Manager change management framework and used by the service-linked role `AWSServiceRoleForSystemsManagerChangeManagement`.
- **AWSSystemsManagerOpsDataSyncServiceRolePolicy** – Service policy that allows the `AWSServiceRoleForSystemsManagerOpsDataSync` service-linked role to create and update OpsItems and OpsData from AWS Security Hub findings.
- **AmazonEC2RoleforSSM** – This policy will be deprecated. In its place, use the **AmazonSSMManagedInstanceCore** policy to allow Systems Manager service core functionality on EC2 instances. For information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#).

### 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 \(p. 36\)](#) and [AssumeRole](#) in the [AWS Security Token Service API Reference](#).

## AWS Systems Manager identity-based policy examples

By default, AWS Identity and Access Management (IAM) users and roles don't have permission to create or modify AWS Systems Manager resources. They also can't perform tasks using the Systems Manager console, AWS Command Line Interface (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 **MyDocument-** in the US East (Ohio) (us-east-2) AWS Region.

```
{
 "Version": "2012-10-17",
 "Statement" : [
 {
 "Effect" : "Allow",
 "Action" : [
 "ssm:DeleteDocument"
],
 "Resource" : [
 "arn:aws:ssm:us-east-2:111122223333:document/MyDocument-*"
]
 }
]
}
```

To learn how to create an IAM identity-based policy using these example JSON Policy documents, see [Creating IAM policies](#) in the *IAM User Guide*.

### Topics

- [Policy best practices \(p. 1391\)](#)
- [Using the Systems Manager console \(p. 1392\)](#)
- [Allow users to view their own permissions \(p. 1393\)](#)
- [Cross-service confused deputy prevention \(p. 1393\)](#)
- [Customer managed policy examples \(p. 1395\)](#)
- [Viewing Systems Manager documents based on tags \(p. 1398\)](#)

## 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:

- **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*.
- **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*.

- **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 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>ListRoles"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "iam:PassedToService": "ssm.amazonaws.com"
 }
 }
 }
]
}
```

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>ListGroupsForUser",
 "iam>ListAttachedUserPolicies",
 "iam>ListUserPolicies",
 "iam GetUser"
],
 "Resource": ["arn:aws:iam::*:user/${aws:username}"]
 },
 {
 "Sid": "NavigateInConsole",
 "Effect": "Allow",
 "Action": [
 "iam:GetGroupPolicy",
 "iam:GetPolicyVersion",
 "iam GetPolicy",
 "iam>ListAttachedGroupPolicies",
 "iam>ListGroupPolicies",
 "iam>ListPolicyVersions",
 "iam>ListPolicies",
 "iam>ListUsers"
],
 "Resource": "*"
 }
]
}
```

## Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that AWS Systems Manager gives another service to the resource. If the `aws:SourceArn` value does not contain the account ID, such as an Amazon S3 bucket Amazon Resource Name (ARN), you must use both global condition context keys to limit permissions. If you use both global condition context keys and the `aws:SourceArn` value contains the account ID, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement. Use `aws:SourceArn` if you want only one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use.

The following sections provide example policies for AWS Systems Manager capabilities.

## Hybrid activation policy example

For service roles used in a hybrid activation, the value of `aws:SourceArn` must be the ARN of the AWS account. Be sure to specify the AWS Region in the ARN where you created your hybrid activation. If you don't know the full ARN of the resource or if you're specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (\*) for the unknown portions of the ARN. For example, `arn:aws:ssm:*:region:123456789012:*`.

The following example demonstrates using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys for Automation to prevent the confused deputy problem in the US East (Ohio) Region (us-east-2).

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "",
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Action": "sts:AssumeRole",
 "Condition": {
 "StringEquals": {
 "aws:SourceAccount": "123456789012"
 },
 "ArnEquals": {
 "aws:SourceArn": "arn:aws:ssm:us-east-2:123456789012:*"
 }
 }
 }
]
}
```

## Resource data sync policy example

Systems Manager Inventory, Explorer, and Compliance enable you to create a resource data sync to centralize storage of your operations data (OpsData) in a central Amazon Simple Storage Service bucket. If you want to encrypt a resource data sync by using AWS Key Management Service (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. The `aws:SourceArn` and `aws:SourceAccount` condition keys in this policy prevent the confused deputy problem. Here is an example policy.

```
{
 "Version": "2012-10-17",
 "Id": "ssm-access-policy",
 "Statement": [
 {
 "Sid": "ssm-access-policy-statement",
 "Action": [
 "kms:GenerateDataKey"
],
 "Effect": "Allow",
 "Principal": {
 "Service": "ssm.amazonaws.com"
 },
 "Resource": "arn:aws:kms:us-east-2:123456789012:key/KMS_key_id",
 "Condition": {
 "StringLike": {
 "aws:SourceAccount": "123456789012"
 },
 }
 }
]
}
```

```
 "ArnLike": {
 "aws:SourceArn": "arn:aws:ssm:*:123456789012:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM"
 }
 }
}
```

#### Note

The ARN in the policy example enables the system to encrypt OpsData from all sources except AWS Security Hub. If you need to encrypt Security Hub data, for example if you use Explorer to collect Security Hub data, then you must attach an additional policy that specifies the following ARN:

```
"aws:SourceArn": "arn:aws:ssm:*:account-id:role/
aws-service-role/opsdatasync.ssm.amazonaws.com/
AWSServiceRoleForSystemsManagerOpsDataSync"
```

## 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 policy examples](#) in the [IAM User Guide](#).

The following examples of user policies grant permissions for various 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 \(p. 1392\)](#).

#### Note

All examples use the US West (Oregon) Region (us-west-2) and contain fictitious account IDs. The account ID shouldn't be specified in the Amazon Resource Name (ARN) for AWS public documents (documents that begin with AWS-\*).

### Examples

- [Example 1: Allow a user to perform Systems Manager operations in a single Region \(p. 1395\)](#)
- [Example 2: Allow a user to list documents for a single Region \(p. 1396\)](#)

### Example 1: Allow a user to perform Systems Manager operations in a single Region

The following example grants permissions to perform Systems Manager operations only in the US East (Ohio) Region (us-east-2).

```
{
 "Version": "2012-10-17",
 "Statement" : [
 {
 "Effect" : "Allow",
 "Action" : [
 "ssm:)"
],
 "Resource" : [
 "arn:aws:ssm:us-east-2:aws-account-ID:)"
]
 }
]
}
```

```
]
}
```

## 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 East (Ohio) Region (us-east-2).

```
{
 "Version": "2012-10-17",
 "Statement" : [
 {
 "Effect" : "Allow",
 "Action" : [
 "ssm>ListDocuments"
],
 "Resource" : [
 "arn:aws:ssm:us-east-2:aws-account-ID:document/Update*"
]
 }
]
}
```

## Example 3: Allow a user to use a specific SSM document to run commands on specific nodes

The following example IAM policy allows a user to do the following in the US East (Ohio) Region (us-east-2):

- List Systems Manager documents (SSM 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 the following entry.

```
arn:aws:ssm:us-east-2:aws-account-ID:document/Systems-Manager-document-name
```

- Send a command to three nodes. The nodes are determined by the following entries in the second Resource section.

```
"arn:aws:ssm:us-east-2:aws-account-ID:instance/i-02573cafcfEXAMPLE",
"arn:aws:ssm:us-east-2:aws-account-ID:instance/i-0471e04240EXAMPLE",
"arn:aws:ssm:us-east-2:aws-account-ID:instance/i-07782c72faEXAMPLE"
```

- View details about a command after it has been sent.
- Start and stop workflows in Automation, a capability of AWS Systems Manager.
- Get information about Automation workflows.

If you want to give a user permission to use this document to send commands on any node for which the user has access (as determined by their user account), you could specify an entry similar to the following in the Resource section and remove the other node entries. The following example uses the US East (Ohio) Region (us-east-2).

```
"arn:aws:ssm: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",
 "Resource": [
 "arn:aws:ssm:us-east-2:aws-account-ID:instance/i-02573cafccEXAMPLE",
 "arn:aws:ssm:us-east-2:aws-account-ID:instance/i-0471e04240EXAMPLE",
 "arn:aws:ssm:us-east-2:aws-account-ID:instance/i-07782c72faEXAMPLE",
 "arn:aws:ssm:us-east-2:aws-account-ID:document/Systems-Manager-document-name"
]
 },
 {
 "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:us-east-2:aws-account-ID:automation-definition/*"
]
 },
 {
 "Action": "ssm>DescribeAutomationExecutions",
 "Effect": "Allow",
 "Resource": [
 "*"
]
 },
 {
 "Action": [
 "ssm>StopAutomationExecution",
 "ssm>GetAutomationExecution"
],
 "Effect": "Allow",
 "Resource": [
 "*"
]
 }
]
]

```

}

## 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 an Systems Manager document, the document must be tagged `Owner=richard-roe` or `owner=richard-roe`. Otherwise they're denied access. The condition tag key `Owner` matches both `Owner` and `owner` because condition key names aren't case-sensitive. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.

## AWS managed policies for AWS Systems Manager

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to [create IAM customer managed policies](#) that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see [AWS managed policies](#) in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the `ViewOnlyAccess` AWS managed policy provides read-only access to many AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see [AWS managed policies for job functions](#) in the *IAM User Guide*.

## AWS managed policy: AmazonSSMSERVICERolePolicy

You can't attach `AmazonSSMSERVICERolePolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows AWS Systems Manager to perform actions on your behalf. For more information, see [Using roles to collect inventory, run maintenance window tasks, and view OpsData: AWSERVICERoleForAmazonSSM \(p. 1410\)](#).

Three Systems Manager capabilities use the service-linked role:

- The Inventory capability requires a service-linked role. The role allows the system to collect Inventory metadata from tags and resource groups.
- The Maintenance Windows capability can optionally use the service-linked role. The role allows the Maintenance Windows service to run maintenance tasks on target managed nodes. 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? \(p. 708\)](#).
- The Explorer capability uses the service-linked role to allow viewing OpsData and OpsItems from multiple accounts. This service-linked role also allows Explorer to create a managed rule when you turn on Security Hub as a data source from Explorer or OpsCenter.

### Permissions details

The `AWSERVICERoleForAmazonSSM` 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>ListTagsForResource`
- `ssm:GetCalendarState`
- `ssm:UpdateServiceSetting [1]`
- `ssm:GetServiceSetting [1]`
- `ec2:DescribeInstanceAttribute`
- `ec2:DescribeInstanceState`
- `ec2:DescribeInstances`
- `lambda:InvokeFunction [2]`
- `states:DescribeExecution [3]`
- `states:StartExecution [3]`
- `resource-groups>ListGroups`
- `resource-groupsListGroupResources`
- `resource-groupsGetGroupQuery`

- tag:GetResources
- config>SelectResourceConfig
- config>DescribeComplianceByConfigRule
- config>DescribeComplianceByResource
- config>DescribeRemediationConfigurations
- config>DescribeConfigurationRecorders
- compute-optimizer>GetEC2InstanceRecommendations
- compute-optimizer>GetEnrollmentStatus
- support>DescribeTrustedAdvisorChecks
- support>DescribeTrustedAdvisorCheckSummaries
- support>DescribeTrustedAdvisorCheckResult
- support>DescribeCases
- iam>PassRole [4]
- cloudformation>DescribeStacks
- cloudformation>ListStackResources
- cloudformation>ListStackInstances [5]
- cloudformation>DescribeStackSetOperation [5]
- cloudformation>DeleteStackSet [5]
- cloudformation>DeleteStackInstances [6]
- events>PutRule [7]
- events>PutTargets [7]
- events>RemoveTargets [8]
- events>DeleteRule [8]
- events>DescribeRule
- securityhub>DescribeHub

[1] The ssm:UpdateServiceSetting and ssm:GetServiceSetting actions are allowed permissions for the following resources only.

```
arn:aws:ssm:***:servicesetting/ssm/opsitem/*
arn:aws:ssm:***:servicesetting/ssm/opsdata/*
```

[2] The lambda:InvokeFunction action is allowed permissions for the following resources only.

```
arn:aws:lambda:***:function:SSM*
arn:aws:lambda:***:function:*:SSM*
```

[3] The states: actions are allowed permissions on the following resources only.

```
arn:aws:states:***:stateMachine:SSM*
arn:aws:states:***:execution:SSM*
```

[4] The iam:PassRole action is allowed permissions by the following condition for the Systems Manager service only.

```
"Condition": {
 "StringEquals": {
```

```
 "iam:PassedToService": [
 "ssm.amazonaws.com"
]
}
```

[5] The `cloudformation>ListStackInstances`, `cloudformationDescribeStackSetOperation`, and `cloudformationDeleteStackSet` actions are allowed permissions on the following resource only.

```
arn:aws:cloudformation::*:stackset/AWS-QuickSetup-SSM*:*
```

[6] The `cloudformationDeleteStackInstances` action is allowed permissions on the following resources only.

```
arn:aws:cloudformation::*:stackset/AWS-QuickSetup-SSM*:*
arn:aws:cloudformation::*:stackset-target/AWS-QuickSetup-SSM*:*
arn:aws:cloudformation::*:type/resource/*
```

[7] The `eventsPutRule` and `eventsPutTargets` actions are allowed permissions by the following condition for the Systems Manager service only.

```
"Condition": {
 "StringEquals": {
 "events:ManagedBy": "ssm.amazonaws.com"
 }
}
```

[8] The `eventsRemoveTargets` and `eventsDeleteRule` actions are allowed permissions on the following resource only.

```
arn:aws:events::rule/SSMExplorerManagedRule
```

### Full **AmazonSSMSERVICERolePolicy** policy

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:CancelCommand",
 "ssm:GetCommandInvocation",
 "ssm>ListCommandInvocations",
 "ssm>ListCommands",
 "ssm:SendCommand",
 "ssm:GetAutomationExecution",
 "ssm:GetParameters",
 "ssm:StartAutomationExecution",
 "ssm>ListTagsForResource",
 "ssm:GetCalendarState"
],
 "Resource": [
 "*"
]
 },
 {
 "Effect": "Allow",
```

```
"Action": [
 "ssm:UpdateServiceSetting",
 "ssm:GetServiceSetting"
],
"Resource": [
 "arn:aws:ssm:*::servicesetting/ssm/opsitem/*",
 "arn:aws:ssm:*::servicesetting/ssm/opsdata/*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "ec2:DescribeInstanceAttribute",
 "ec2:DescribeInstanceState",
 "ec2:DescribeInstances"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "lambda:InvokeFunction"
],
 "Resource": [
 "arn:aws:lambda::*:function:SSM*",
 "arn:aws:lambda::*:function::*:SSM*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "states:DescribeExecution",
 "states:StartExecution"
],
 "Resource": [
 "arn:aws:states::*:stateMachine:SSM*",
 "arn:aws:states::*:execution:SSM*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "resource-groups>ListGroups",
 "resource-groups>ListGroupResources",
 "resource-groups>GetGroupQuery"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "cloudformation>DescribeStacks",
 "cloudformation>ListStackResources"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "tag:GetResources"
]
}
```

```
],
 "Resource": [
 "*"
]
 },
{
 "Effect": "Allow",
 "Action": [
 "config:SelectResourceConfig"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "compute-optimizer:GetEC2InstanceRecommendations",
 "compute-optimizer:GetEnrollmentStatus"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "support:DescribeTrustedAdvisorChecks",
 "support:DescribeTrustedAdvisorCheckSummaries",
 "support:DescribeTrustedAdvisorCheckResult",
 "support:DescribeCases"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "config:DescribeComplianceByConfigRule",
 "config:DescribeComplianceByResource",
 "config:DescribeRemediationConfigurations",
 "config:DescribeConfigurationRecorders"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "iam:PassedToService": [
 "ssm.amazonaws.com"
]
 }
 }
},
{
 "Effect": "Allow",
 "Action": "organizations:DescribeOrganization",
 "Resource": "*"
}
```

```
 "Effect": "Allow",
 "Action": "cloudformation>ListStackSets",
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "cloudformation>ListStackInstances",
 "cloudformation>DescribeStackSetOperation",
 "cloudformation>DeleteStackSet"
],
 "Resource": "arn:aws:cloudformation:*::stackset/AWS-QuickSetup-SSM*::*"
 },
 {
 "Effect": "Allow",
 "Action": "cloudformation>DeleteStackInstances",
 "Resource": [
 "arn:aws:cloudformation:*::stackset/AWS-QuickSetup-SSM*::*",
 "arn:aws:cloudformation:*::stackset-target/AWS-QuickSetup-SSM*::*",
 "arn:aws:cloudformation:*::type/resource/*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "events>PutRule",
 "events>PutTargets"
],
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "events:ManagedBy": "ssm.amazonaws.com"
 }
 }
 },
 {
 "Effect": "Allow",
 "Action": [
 "events>RemoveTargets",
 "events>DeleteRule"
],
 "Resource": [
 "arn:aws:events::*:rule/SSMExplorerManagedRule"
]
 },
 {
 "Effect": "Allow",
 "Action": "events>DescribeRule",
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": "securityhub>DescribeHub",
 "Resource": "*"
 }
]
```

## AmazonSSMReadOnlyAccess

Use the [AmazonSSMReadOnlyAccess](#) AWS managed policy to allow read-only access to AWS Systems Manager. Assigning this policy to an AWS Identity and Access Management (IAM) user allows read-only API operations, such as `Describe*`, `Get*`, and `List*`. View the [policy](#) for the full list of actions supported by this policy.

## Service-level permissions

This policy provides read-only access to Systems Manager. No other service permissions are included in this policy.

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

## AWS managed policy: **AWSSystemsManagerOpsDataSyncServiceRolePolicy**

You can't attach **AWSSystemsManagerOpsDataSyncServiceRolePolicy** to your IAM entities. This policy is attached to a service-linked role that allows Systems Manager to perform actions on your behalf. For more information, see [Using roles to create OpsData and OpsItems for Systems Manager Explorer: AWSServiceRoleForSystemsManagerOpsDataSync \(p. 1415\)](#).

**AWSSystemsManagerOpsDataSyncServiceRolePolicy** allows the [\(p. 1415\)](#) service-linked role to create and update OpsItems and OpsData from AWS Security Hub findings.

### Permissions details

The **AWSServiceRoleForSystemsManagerOpsDataSync** service-linked role permissions policy allows Systems Manager to complete the following actions on all related resources ("Resource": "\*"), except where indicated:

- **ssm:GetOpsItem [1]**
- **ssm:UpdateOpsItem [1]**
- **ssm>CreateOpsItem**
- **ssm:AddTagsToResource [2]**
- **ssm:UpdateServiceSetting [3]**
- **ssm:GetServiceSetting [3]**
- **securityhub:GetFindings**
- **securityhub:GetFindings**
- **securityhub:BatchUpdateFindings [4]**

[1] The **ssm:GetOpsItem** and **ssm:UpdateOpsItem** actions are allowed permissions by the following condition for the Systems Manager service only.

```
"Condition": {
 "StringEquals": {
 "aws:ResourceTag/ExplorerSecurityHubOpsItem": "true"
 }
}
```

```
}
```

[2] The `ssm:AddTagsToResource` action is allowed permissions for the following resource only.

```
arn:aws:ssm:*:*:opsitem/*
```

[3] The `ssm:UpdateServiceSetting` and `ssm:GetServiceSetting` actions are allowed permissions for the following resources only.

```
arn:aws:ssm:*:*:servicesetting/ssm/opsitem/*
arn:aws:ssm:*:*:servicesetting/ssm/opsdata/*
```

[4] The `securityhub:BatchUpdateFindings` are denied permissions by the following condition for the Systems Manager service only.

```
"Condition": {
 "StringEquals": {
 "securityhub:ASFFSyntaxPath/Workflow.Status": "SUPPRESSED"
 },
 "Null": {
 "securityhub:ASFFSyntaxPath/Confidence": false,
 "securityhub:ASFFSyntaxPath/Criticality": false,
 "securityhub:ASFFSyntaxPath/Note": false,
 "securityhub:ASFFSyntaxPath/RelatedFindings": false,
 "securityhub:ASFFSyntaxPath/Types": false,
 "securityhub:ASFFSyntaxPath/UserDefinedFields": false,
 "securityhub:ASFFSyntaxPath/VerificationState": false
 }
}
```

#### Full AWSSystemsManagerOpsDataSyncServiceRolePolicy policy

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetOpsItem",
 "ssm:UpdateOpsItem"
],
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "aws:ResourceTag/ExplorerSecurityHubOpsItem": "true"
 }
 }
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:CreateOpsItem"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:AddTagsToResource"
],
 "Resource": "arn:aws:ssm:*:*:opsitem/*"
 }
]
}
```

```

},
{
 "Effect": "Allow",
 "Action": [
 "ssm:UpdateServiceSetting",
 "ssm:GetServiceSetting"
],
 "Resource": [
 "arn:aws:ssm:*::servicesetting/ssm/opsitem/*",
 "arn:aws:ssm:*::servicesetting/ssm/opsdata/*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "securityhub:GetFindings",
 "securityhub:BatchUpdateFindings"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Deny",
 "Action": "securityhub:BatchUpdateFindings",
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "securityhub:ASFFSyntaxPath/Workflow.Status": "SUPPRESSED"
 },
 "Null": {
 "securityhub:ASFFSyntaxPath/Confidence": false,
 "securityhub:ASFFSyntaxPath/Criticality": false,
 "securityhub:ASFFSyntaxPath/Note": false,
 "securityhub:ASFFSyntaxPath/RelatedFindings": false,
 "securityhub:ASFFSyntaxPath/Types": false,
 "securityhub:ASFFSyntaxPath/UserDefinedFields": false,
 "securityhub:ASFFSyntaxPath/VerificationState": false
 }
 }
}
]
}

```

## Systems Manager updates to AWS managed policies

View details about updates to AWS managed policies for Systems Manager since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Systems Manager [Document history \(p. 1558\)](#) page.

Change	Description	Date
<a href="#">AmazonSSMSERVICERolePolicy</a> – Update to an existing policy.	Systems Manager added new permissions to allow Explorer to create a managed rule when you turn on Security Hub from Explorer or OpsCenter. New permissions were added to check	April 27, 2021

Change	Description	Date
	that config and the compute-optimizer meet the necessary requirements before allowing OpsData.	
<a href="#">AWSSystemsManagerOpsDataSyncPolicy</a> – New policy.	Systems Manager added a new policy to create and update OpsItems and OpsData from Security Hub findings in Explorer and OpsCenter.	April 27, 2021
<a href="#">AmazonSSMServiceRolePolicy</a> – Update to an existing policy.	Systems Manager added new permissions to allow viewing aggregate OpsData and OpsItems details from multiple accounts and AWS Regions in Explorer.	March 24, 2021
Systems Manager started tracking changes	Systems Manager started tracking changes for its AWS managed policies.	March 12, 2021

## 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 AWS Systems Manager and AWS Identity and Access Management (IAM).

### Topics

- [I am not authorized to perform an action in Systems Manager \(p. 1408\)](#)
- [I am not authorized to perform iam:PassRole \(p. 1409\)](#)
- [I want to view my access keys \(p. 1409\)](#)
- [I'm an administrator and want to allow others to access Systems Manager \(p. 1409\)](#)
- [I want to allow people outside of my AWS account to access my Systems Manager resources \(p. 1409\)](#)

## 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 doesn't have `ssm:GetDocument` permissions.

```
User: arn:aws:ssm::123456789012:user/mateojackson isn't 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, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). 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 \(p. 1384\)](#).
- 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, run maintenance window tasks, and view OpsData: [AWSRoleForAmazonSSM \(p. 1410\)](#)
- Using roles to collect AWS account information for Systems Manager Explorer: [AWSRoleForAmazonSSM\\_AccountDiscovery \(p. 1412\)](#)
- Using roles to create OpsData and OpsItems for Systems Manager Explorer: [AWSRoleForSystemsManagerOpsDataSync \(p. 1415\)](#)
- Using roles to create operational insight OpsItems in Systems Manager OpsCenter: [AWSRoleForAmazonSSM\\_OpsInsights \(p. 1418\)](#)

## Using roles to collect inventory, run maintenance window tasks, and view OpsData: [AWSRoleForAmazonSSM](#)

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.

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 roles** 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.

Three Systems Manager capabilities use the service-linked role:

- The Inventory capability requires a service-linked role. The role allows the system to collect Inventory metadata from tags and resource groups.
- Systems Manager Maintenance Windows can optionally use the service-linked role. The role allows the Maintenance Windows service to run maintenance tasks on target nodes. 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? \(p. 708\)](#).
- The Explorer capability uses the service-linked role to allow viewing OpsData and OpsItems from multiple accounts. This service-linked role also allows Explorer to create a managed rule when you allow Security Hub as a data source from Explorer or OpsCenter.

The managed policy that is used to provide permissions for the `AWSServiceRoleForAmazonSSM` role is `AmazonSSMSERVICERolePolicy`. For details about the permissions it grants, see [AWS managed policy: AmazonSSMSERVICERolePolicy \(p. 1399\)](#).

## Creating the `AWSServiceRoleForAmazonSSM` service-linked role for Systems Manager

You can use the IAM console to create a service-linked role with the **EC2** use case. Using commands for IAM in the AWS Command Line Interface (AWS CLI) or using 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 the `AWSServiceRoleForAmazonSSM` service-linked role for Systems Manager

Systems Manager doesn't allow you to edit the `AWSServiceRoleForAmazonSSM` service-linked role. After you create a service-linked role, you can't 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 the `AWSServiceRoleForAmazonSSM` 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 isn't actively monitored or maintained. You can use the IAM console, the AWS 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 `AWSServiceRoleForAmazonSSM` service-linked role can be used by multiple capabilities, ensure that none are using the role before attempting to delete it.

- **Inventory:** If you delete the service-linked role used by the Inventory capability, 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 rely on the role. You must first remove the service-linked role from the tasks before you can delete the role.
- **Explorer:** If you delete the service-linked role used by the Explorer capability, then the cross-account and cross-Region OpsData and OpsItems are no longer viewable.

#### Note

If the Systems Manager service is using the role when you try to delete 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 [Add and delete tags on an individual resource](#).
2. To delete resource groups, see [Delete groups from AWS Resource Groups](#).
3. To delete maintenance window tasks, see [Updating or deregistering maintenance window tasks \(console\) \(p. 731\)](#).

#### To manually delete the `AWSServiceRoleForAmazonSSM` service-linked role using IAM

Use the IAM console, the AWS 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 the Systems Manager `AWSServiceRoleForAmazonSSM` service-linked role

Systems Manager supports using the `AWSServiceRoleForAmazonSSM` service-linked role in all of the AWS Regions where the service is available. For more information, see [AWS Systems Manager endpoints and quotas](#).

## Using roles to collect AWS account information for Systems Manager Explorer: `AWSServiceRoleForAmazonSSM_AccountDiscovery`

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.

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 account discovery

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>ListAWSAccessForOrganization`
- `organizations>ListChildren`
- `organizations>ListParents`
- `organizations>ListDelegatedServicesForAccount`
- `organizations>ListDelegatedAdministrators`

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 the `AWSServiceRoleForAmazonSSM_AccountDiscovery` service-linked role for Systems Manager

You must create a service-linked role. If you create a resource data sync by using Systems Manager in the AWS Management Console, you can create the service-linked role by choosing the **Create role** button. If you want to create a resource data sync programmatically, then you must create the role before you create the resource data sync. You can create the role by using the [CreateServiceLinkedRole](#) API operation.

## Editing the `AWSServiceRoleForAmazonSSM_AccountDiscovery` service-linked role for Systems Manager

Systems Manager doesn't allow you to edit the `AWSServiceRoleForAmazonSSM_AccountDiscovery` service-linked role. After you create a service-linked role, you can't 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 the AWSServiceRoleForAmazonSSM\_AccountDiscovery 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 isn't actively monitored or maintained. However, you must clean up your service-linked role before you can manually delete it.

### Cleaning up the AWSServiceRoleForAmazonSSM\_AccountDiscovery service-linked role

Before you can use IAM to delete the `AWSServiceRoleForAmazonSSM_AccountDiscovery` service-linked role, you must first delete all Explorer resource data syncs. For more information, see [Deleting a Systems Manager Explorer resource data sync \(p. 175\)](#).

#### 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 AWSServiceRoleForAmazonSSM\_AccountDiscovery 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 the Systems ManagerAWSServiceRoleForAmazonSSM\_AccountDiscovery service-linked role

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 doesn't support using service-linked roles in every Region where the service is available. You can use the `AWSServiceRoleForAmazonSSM_AccountDiscovery` role in the following Regions.

AWS Region name	Region identity	Support in Systems Manager
US East (N. Virginia)	us-east-1	Yes
US East (Ohio)	us-east-2	Yes
US West (N. California)	us-west-1	Yes
US West (Oregon)	us-west-2	Yes
Asia Pacific (Mumbai)	ap-south-1	Yes
Asia Pacific (Osaka)	ap-northeast-3	Yes
Asia Pacific (Seoul)	ap-northeast-2	Yes

AWS Region name	Region identity	Support in Systems Manager
Asia Pacific (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes
Asia Pacific (Tokyo)	ap-northeast-1	Yes
Canada (Central)	ca-central-1	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes
South America (São Paulo)	sa-east-1	Yes
AWS GovCloud (US)	us-gov-west-1	No

## Using roles to create OpsData and OpsItems for Systems Manager Explorer: [AWSServiceRoleForSystemsManagerOpsDataSync](#)

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.

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 OpsData sync

Systems Manager uses the service-linked role named [AWSServiceRoleForSystemsManagerOpsDataSync](#) – AWS Systems Manager uses this IAM service role for Systems Manager Explorer to create OpsData and OpsItems.

The [AWSServiceRoleForSystemsManagerOpsDataSync](#) service-linked role trusts the following services to assume the role:

- opsdatasync.ssm.amazonaws.com

The role permissions policy allows Systems Manager to complete the following actions on the specified resources:

- Systems Manager Explorer requires that a service-linked role grant permission to update a security finding when an OpsItem is updated, create and update an OpsItem, and turn off the Security Hub data source when an SSM managed rule is deleted by customers.

The managed policy that is used to provide permissions for the `AWSServiceRoleForSystemsManagerOpsDataSync` role is `AWSSystemsManagerOpsDataSyncServiceRolePolicy`. For details about the permissions it grants, see [AWS managed policy: AWSSystemsManagerOpsDataSyncServiceRolePolicy \(p. 1405\)](#).

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 the `AWSServiceRoleForSystemsManagerOpsDataSync` service-linked role for Systems Manager

You don't need to manually create a service-linked role. When you enable Explorer in the AWS Management Console, Systems Manager creates the service-linked role for you.

### Important

This service-linked role can be displayed in your account if you completed an action in another service that uses the features supported by this role. Also, if you were using the Systems Manager service before January 1, 2017, when it began supporting service-linked roles, then Systems Manager created the `AWSServiceRoleForSystemsManagerOpsDataSync` role in your account. To learn more, see [A new role appeared in my IAM account](#).

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 enable Explorer in the AWS Management Console, Systems Manager creates the service-linked role for you again.

You can also use the IAM console to create a service-linked role with the **AWS Service Role for AWS Systems Manager** use case. In the AWS CLI or the AWS 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*. If you delete this service-linked role, you can use this same process to create the role again.

## Editing the `AWSServiceRoleForSystemsManagerOpsDataSync` service-linked role for Systems Manager

Systems Manager doesn't allow you to edit the `AWSServiceRoleForSystemsManagerOpsDataSync` service-linked role. After you create a service-linked role, you can't 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 the `AWSServiceRoleForSystemsManagerOpsDataSync` 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 isn't actively monitored or

maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

**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.

The procedure for deleting Systems Manager resources used by the `AWSServiceRoleForSystemsManagerOpsDataSync` role depends on if you've configured Explorer or OpsCenter to integrate with Security Hub.

**To delete Systems Manager resources used by the `AWSServiceRoleForSystemsManagerOpsDataSync` role**

- To stop Explorer from creating new OpsItems for Security Hub findings, see [How to stop receiving findings \(p. 177\)](#).
- To stop OpsCenter from creating new OpsItems for Security Hub findings, see [How to stop receiving findings \(p. 229\)](#).

**To manually delete the `AWSServiceRoleForSystemsManagerOpsDataSync` service-linked role using IAM**

Use the IAM console, the AWS CLI, or the AWS API to delete the `AWSServiceRoleForSystemsManagerOpsDataSync` service-linked role. For more information, see [Deleting a service-linked role](#) in the *IAM User Guide*.

## Supported Regions for the Systems Manager`AWSServiceRoleForSystemsManagerOpsDataSync` service-linked role

Systems Manager supports using service-linked roles in all of the Regions where the service is available. For more information, see [AWS Systems Manager endpoints and quotas](#).

Systems Manager doesn't support using service-linked roles in every Region where the service is available. You can use the `AWSServiceRoleForSystemsManagerOpsDataSync` role in the following Regions.

AWS Region name	Region identity	Support in Systems Manager
US East (N. Virginia)	us-east-1	Yes
US East (Ohio)	us-east-2	Yes
US West (N. California)	us-west-1	Yes
US West (Oregon)	us-west-2	Yes
Asia Pacific (Mumbai)	ap-south-1	Yes
Asia Pacific (Osaka)	ap-northeast-3	Yes
Asia Pacific (Seoul)	ap-northeast-2	Yes
Asia Pacific (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes

AWS Region name	Region identity	Support in Systems Manager
Asia Pacific (Tokyo)	ap-northeast-1	Yes
Canada (Central)	ca-central-1	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes
Europe (Stockholm)	eu-north-1	Yes
South America (São Paulo)	sa-east-1	Yes
AWS GovCloud (US)	us-gov-west-1	No

## Using roles to create operational insight OpsItems in Systems Manager OpsCenter: `AWSServiceRoleForAmazonSSM_OpsInsights`

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 operational insight OpsItems

Systems Manager uses the service-linked role named `AWSServiceRoleForAmazonSSM_OpsInsights`. AWS Systems Manager uses this IAM service role to create and update operational insight OpsItems in Systems Manager OpsCenter.

The `AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role trusts the following services to assume the role:

- `opsinsights.ssm.amazonaws.com`

The role permissions policy allows Systems Manager to complete the following actions on the specified resources:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowCreateOpsItem",
 "Effect": "Allow",
 "Action": [
 "ssm:CreateOpsItem",
 "ssm:AddTagsToResource"
],
 "Resource": "*"
 },
 {
 "Sid": "AllowAccessOpsItem",
 "Effect": "Allow",
 "Action": [
 "ssm:UpdateOpsItem",
 "ssm:GetOpsItem"
],
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "aws:ResourceTag/SsmOperationalInsight": "true"
 }
 }
 }
]
}
```

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 the `AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role for Systems Manager

You must create a service-linked role. If you enable operational insights by using Systems Manager in the AWS Management Console, you can create the service-linked role by choosing the **Enable** button.

## Editing the `AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role for Systems Manager

Systems Manager does not allow you to edit the `AWSServiceRoleForAmazonSSM_OpsInsights` 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 the `AWSServiceRoleForAmazonSSM_OpsInsights` 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 the `AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role

Before you can use IAM to delete the `AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role, you must first disable operational insights in Systems Manager OpsCenter. For more information, see [Working with operational insights \(p. 214\)](#).

### Manually delete the `AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role

Use the IAM console, the AWS CLI, or the AWS API to delete the `AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role. For more information, see [Deleting a service-linked role](#) in the *IAM User Guide*.

## Supported Regions for the Systems Manager `AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role

Systems Manager does not support using service-linked roles in every Region where the service is available. You can use the `AWSServiceRoleForAmazonSSM_OpsInsights` role in the following Regions.

Region name	Region identity	Support in Systems Manager
US East (N. Virginia)	us-east-1	Yes
US East (Ohio)	us-east-2	Yes
US West (N. California)	us-west-1	Yes
US West (Oregon)	us-west-2	Yes
Asia Pacific (Mumbai)	ap-south-1	Yes
Asia Pacific (Tokyo)	ap-northeast-1	Yes
Asia Pacific (Seoul)	ap-northeast-2	Yes
Asia Pacific (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes
Asia Pacific (Hong Kong)	ap-east-1	Yes
Canada (Central)	ca-central-1	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes
Europe (Stockholm)	eu-north-1	Yes
Europe (Milan)	eu-south-1	Yes
South America (São Paulo)	sa-east-1	Yes

Region name	Region identity	Support in Systems Manager
Middle East (Bahrain)	me-south-1	Yes
Africa (Cape Town)	af-south-1	Yes
AWS GovCloud (US)	us-gov-west-1	Yes
AWS GovCloud (US)	us-gov-east-1	Yes

## Logging and monitoring in AWS Systems Manager

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS 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 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 \(p. 1442\)](#).

### Amazon CloudWatch alarms

Using Amazon CloudWatch alarms, you watch a single metric over a time period that you specify for your Amazon Elastic Compute Cloud (Amazon EC2) instances and other resources. If the metric exceeds a given threshold, a notification is sent to an Amazon Simple Notification Service (Amazon SNS) topic or AWS Auto Scaling policy. CloudWatch alarms don't invoke actions because they're 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 AWS 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 \(p. 229\)](#).

### Amazon EventBridge

Using Amazon EventBridge, you can configure rules to alert you to changes in Systems Manager resources, and to direct EventBridge to take actions based on the content of those events. EventBridge 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 EventBridge \(p. 1449\)](#).

### Amazon CloudWatch Logs and SSM Agent logs

SSM Agent writes information about executions, scheduled actions, errors, and health statuses to log files on each node. You can view log files by manually connecting to a node. We recommend automatically sending agent log data to a log group in CloudWatch Logs for analysis. For more information, see [Sending node logs to CloudWatch Logs \(CloudWatch agent\) \(p. 1429\)](#) and [Viewing SSM Agent logs \(p. 132\)](#).

## AWS Systems Manager Compliance

You can use Compliance, a capability of AWS Systems Manager, to scan your fleet of managed nodes for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and AWS Regions, and then drill down into specific resources that aren't compliant. By default, Compliance displays current compliance data about patching in Patch Manager, a capability of AWS Systems Manager, and associations in State Manager, a capability of AWS Systems Manager. For more information, see [AWS Systems Manager Compliance \(p. 836\)](#).

## AWS Systems Manager Explorer

Explorer, a capability of AWS Systems Manager, 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 AWS 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 \(p. 157\)](#).

## AWS Systems Manager OpsCenter

OpsCenter, a capability of AWS Systems Manager, 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 runbooks in Automation, a capability of AWS Systems Manager, that you can use to quickly resolve issues. OpsCenter is integrated with Amazon EventBridge. This means you can create EventBridge rules that automatically create OpsItems for any AWS service that publishes events to EventBridge. For more information, see [AWS Systems Manager OpsCenter \(p. 180\)](#).

## 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 Run Command or Maintenance Windows, capabilities of AWS Systems Manager. 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 nodes, you can receive a notification for each copy of the command sent to a specific node. For more information, see [Monitoring Systems Manager status changes using Amazon SNS notifications \(p. 1462\)](#).

## AWS Trusted Advisor and AWS Health Dashboard

Trusted Advisor draws upon best practices learned from serving hundreds of thousands of AWS customers. Trusted Advisor inspects your AWS environment and then makes recommendations when opportunities exist to save money, improve system availability and performance, or help close security gaps. All AWS customers have access to five Trusted Advisor checks. Customers with either an AWS Support Business or Enterprise plan can view all Trusted Advisor checks. For more information, see [AWS Trusted Advisor](#) in the [AWS Support User Guide](#) and the [AWS Health User Guide](#).

### Related content

- [Monitoring AWS Systems Manager \(p. 1428\)](#)

# 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 nodes, see [AWS Systems Manager Compliance \(p. 836\)](#).

Third-party auditors assess the security and compliance of 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 under [Best Practices for Security, Identity, & Compliance](#).

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 such as 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 \(p. 1422\)](#)
- [Shared Responsibility Model](#)
- [Best Practices for Security, Identity, & Compliance](#)

## Security best practices for Systems Manager

AWS 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 \(p. 1424\)](#)
- [Systems Manager monitoring and auditing best practices \(p. 1426\)](#)

## 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 allow 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

In Parameter Store, a capability of AWS Systems Manager, 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` data type. Parameter Store uses an AWS KMS key in AWS Key Management Service (AWS KMS) to encrypt the parameter value. AWS KMS uses either a customer managed key or an AWS managed key when encrypting the parameter value. For maximum security, we recommend using your own KMS key. If you use the AWS managed 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're using customer managed keys 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 customer managed keys.

For more information, see [Restricting access to Systems Manager parameters using IAM policies \(p. 260\)](#) 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 parameters in Systems Manager documents (SSM documents) by defining `allowedValues` and `allowedPattern`. For `allowedValues`, you define an array of values allowed for the parameter. If a user inputs a value that isn't 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 doesn't match the allowed pattern, the execution fails to start.

For more information about `allowedValues` and `allowedPattern`, see [SSM document syntax \(p. 1308\)](#).

### Block public sharing for documents

Unless your use case requires public sharing to be allowed, we recommend turning on the block public sharing setting for your SSM documents in the **Preferences** section of the Systems Manager Documents console.

### 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 don't require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service doesn't leave the Amazon network.

For more information about Amazon VPC security, see [\(Optional\) Create a VPC endpoint \(p. 28\)](#) and [Internetwork traffic privacy in Amazon VPC](#) in the [Amazon VPC User Guide](#).

### Restrict Session Manager users to sessions using interactive commands

Session Manager, a capability of AWS Systems Manager, provides [several methods for starting sessions \(p. 971\)](#) to your managed nodes. 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 \(interactive and noninteractive commands\) \(p. 975\)](#).

### Provide temporary node permissions for Automation workflows

During a workflow in Automation, a capability of AWS Systems Manager, your nodes might need permissions that are needed for that execution only but not for other Systems Manager operations. For example, an Automation workflow might require a node to call a particular API operation 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 nodes within the Automation runbook itself instead of adding the permissions to your IAM instance profile. At the end of the Automation workflow, 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 nodes 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 Amazon Elastic Compute Cloud (Amazon 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 [Automating updates to SSM Agent \(p. 135\)](#).
- **AWS CLI** – The AWS Command Line Interface (AWS CLI) is an open source tool that allows 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 CLI version 2 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 allow you to script operations on your AWS resources from the PowerShell command line. Periodically, as updated versions of the Tools for Windows PowerShell are released, you should update the version that you're 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](#) in the *IAM Policy Simulator User Guide*.
- **Session Manager plugin** – If users in your organization with permissions to use Session Manager want to connect to a node 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 \(p. 964\)](#).
- **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 \(p. 145\)](#).

## 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 [Find resources to tag](#) in the *AWS Resource Groups User Guide*.

Create resource groups for your Systems Manager resources. For more information, see [What are 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. Amazon CloudWatch provides several tools and

services to help you monitor Systems Manager and your other AWS services. For more information, see [Sending node logs to CloudWatch Logs \(CloudWatch agent\) \(p. 1429\)](#) and [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#).

### Use 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 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 \(p. 1442\)](#).

### Turn on AWS Config

AWS Config allows 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 through the Personal Health Dashboard. For more information, see the [AWS Health Documentation](#).

### Related content

- [Best Practices for Security, Identity, & Compliance](#)
- [Getting Started: Follow Security Best Practices as You Configure Your AWS Resources \(AWS Security Blog\)](#)
- [Security best practices in IAM](#)
- [Security best practices in AWS CloudTrail](#)
- [Security Best Practices for Amazon S3](#)
- [Security best practices for AWS Key Management Service](#)

# Monitoring AWS Systems Manager

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Systems Manager and your AWS solutions. You should collect monitoring data from all parts of your AWS solution so that you can debug a multipoint failure if one occurs. But before you start monitoring Systems Manager, 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 performs 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 create 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. Automation is a capability of AWS Systems Manager.

You can also monitor CPU utilization, disk I/O, and network utilization of your managed nodes. When performance falls outside your established baseline, you might need to reconfigure or optimize the node to reduce CPU utilization, improve disk I/O, or reduce network traffic. For more information about monitoring EC2 instances, see [Monitor Amazon EC2 in the Amazon EC2 User Guide for Linux Instances](#).

## Topics

- [Monitoring tools \(p. 1428\)](#)
- [Sending node logs to CloudWatch Logs \(CloudWatch agent\) \(p. 1429\)](#)
- [Sending SSM Agent logs to CloudWatch Logs \(p. 1438\)](#)
- [Monitoring Automation metrics using Amazon CloudWatch \(p. 1440\)](#)
- [Monitoring Run Command metrics using Amazon CloudWatch \(p. 1441\)](#)
- [Logging AWS Systems Manager API calls with AWS CloudTrail \(p. 1442\)](#)
- [Logging Automation action output with CloudWatch Logs \(p. 1445\)](#)
- [Configuring Amazon CloudWatch Logs for Run Command \(p. 1447\)](#)
- [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#)
- [Monitoring Systems Manager status changes using Amazon SNS notifications \(p. 1462\)](#)

## 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 \(p. 1421\)](#).

# Sending node logs to CloudWatch Logs (CloudWatch agent)

You can configure and use the Amazon CloudWatch agent to collect metrics and logs from your nodes instead of using AWS Systems Manager Agent (SSM Agent) for these tasks. The CloudWatch agent allows 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. Parameter Store is a capability of AWS Systems Manager.

## Note

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 [Collecting metrics and logs from Amazon EC2 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 won't 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 node. Manually connecting to a node to view log files and troubleshoot an issue with SSM Agent is time-consuming. For more efficient node 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 node logs to CloudWatch Logs \(CloudWatch agent\)](#) (p. 1429)
- [Migrate Windows Server node log collection to the CloudWatch agent](#) (p. 1430)
- [Collecting metrics and logs from Amazon EC2 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 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 node.
- Access to other CloudWatch features such as metrics and alarms.

For information about monitoring Session Manager activity, see [Auditing session activity](#) (p. 977) and [Logging session activity](#) (p. 978).

# Migrate Windows Server node log collection to the CloudWatch agent

If you're using SSM Agent on supported Windows Server nodes 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, and migrate your configuration settings.

The CloudWatch agent isn't 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 might 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 nodes 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 turn off log collection in SSM Agent and install the CloudWatch agent without a migration process. For more information, see the following topics in the *Amazon CloudWatch User Guide*:

- [Installing the CloudWatch agent](#)
- [Installing the CloudWatch agent on on-premises servers](#)

## Before you begin

Before you begin a migration to the CloudWatch agent for log collection, ensure that the nodes 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 node.
- SSM Agent is configured for monitoring on the node.

## Topics

- [Automatically migrating to the CloudWatch agent \(p. 1431\)](#)
- [Manually migrating to the CloudWatch agent \(p. 1432\)](#)

## Automatically migrating to the CloudWatch agent

For EC2 instances for Windows Server only, you can use the AWS Systems Manager console or the AWS Command Line Interface (AWS CLI) to automatically migrate to the CloudWatch agent as your log collection tool.

### Note

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 [Collecting metrics and logs from Amazon EC2 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 won't be able to use Systems Manager to perform a tool migration.

After the migration succeeds, check your results in CloudWatch to ensure you're receiving the metrics, logs, or Windows event logs you expect. If you're satisfied with the results, you can optionally [Store CloudWatch agent configuration settings in Parameter Store \(p. 1436\)](#). If the migration isn't successful or the results aren't as expected, you can try [Rolling back to log collection with SSM Agent \(p. 1436\)](#).

### 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*.

```
{
 "Id": "CloudWatchIISLogs",
 "FullName": "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
 "Parameters": {
 "AccessKey": "",
 "SecretKey": "",
 "Region": "us-east-1",
 "LogGroup": "Production-Windows-IIS",
 "LogStream": "{hostname}"
 }
}
```

After the migration, this entry maps 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)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Run Command**, and then choose **Run command**.
3. In the **Command document** list, choose `AmazonCloudWatch-MigrateCloudWatchAgent`.
4. For **Status**, choose **Enabled**.
5. In the **Targets** section, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

### Note

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

6. For **Rate control**:
  - For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes still processing the command might also send errors.
7. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

8. 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 \(p. 1462\)](#).

9. 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 nodes you want to update, such as *i-02573cafefEXAMPLE*.

## Manually migrating to the CloudWatch agent

For on-premises Windows Server nodes 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*.

```
{
 "Id": "CloudWatchIISLogs",
 "FullName": "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
 "Parameters": {
 "AccessKey": "",
 "SecretKey": "",
 "Region": "us-east-1",
```

```
"LogGroup": "Production-Windows-IIS",
"LogStream": "{hostname}"
}
}
```

After the migration, this entry maps 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)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Run Command**, and then choose **Run command**.
3. In the **Command document** list, choose AWS-ConfigureAWSPackage.
4. For **Action**, choose **Install**.
5. For **Name**, enter **AmazonCloudWatchAgent**.
6. For **Version**, enter **latest** if it isn't already provided by default.
7. In the **Targets** section, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

8. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

11. Choose **Run**.

## Two: To update config data JSON format

- To update the JSON formatting of the existing config settings for the CloudWatch agent, use **Run Command**, a capability of AWS Systems Manager, or log in to the node directly with an RDP connection to run the following Windows PowerShell commands on the node, one at a time.

```
cd ${Env:ProgramFiles}\\\Amazon\\AmazonCloudWatchAgent
```

```
.\\amazon-cloudwatch-agent-config-wizard.exe --isNonInteractiveWindowsMigration
```

{Env:ProgramFiles} represents the location where the Amazon directory containing the CloudWatch agent can be found, typically C:\\Program Files.

## Three: To configure and start the CloudWatch agent (console)

- Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
- In the navigation pane, choose **Run Command**, and then choose **Run command**.
- In the **Command document** list, choose **AWS-RunPowerShellScript**.
- For **Commands**, enter the following two commands.

```
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 directory containing the CloudWatch agent can be found, typically C:\\Program Files.

- In the **Targets** section, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

- For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes still processing the command might also send errors.

- (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role](#)

- for a hybrid environment (p. 36). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.
8. 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 \(p. 1462\)](#).

9. Choose **Run**.

#### Four: To turn off log collection in SSM Agent (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Run Command**, and then choose **Run command**.
3. In the **Command document** list, choose **AWS-ConfigureCloudWatch**.
4. For **Status**, choose **Disabled**.
5. In the **Targets** section, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

6. For **Status**, choose **Disabled**.

7. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes still processing the command might also send errors.
8. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

9. 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 \(p. 1462\)](#).

10. Choose **Run**.

After completing these steps, check your logs in CloudWatch to verify you are receiving the metrics, logs, or Windows event logs you expect. If the results are satisfactory, you can optionally [Store CloudWatch agent configuration settings in Parameter Store \(p. 1436\)](#). If the migration isn't successful or the results aren't as expected, you can [Rolling back to log collection with SSM Agent \(p. 1436\)](#).

## Store CloudWatch agent configuration settings in Parameter Store

You can store the contents of an CloudWatch agent configuration file in Parameter Store. By maintaining this configuration data in a parameter, multiple nodes can derive their configuration settings from it, and you avoid having to create or manually update configuration files on your nodes. For example, you can use Run Command to write the contents of the parameter to configuration files on multiple nodes, or use State Manager, a capability of AWS Systems Manager, to help avoid configuration drift in the CloudWatch agent configuration settings across a fleet of nodes.

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](#) in the *Amazon CloudWatch User Guide*.

If you ran the wizard but didn't 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 node in the following file.

```
$(Env:ProgramFiles)\Amazon\AmazonCloudWatchAgent\config.json
```

`{Env:ProgramFiles}` represents the location where the Amazon directory containing the CloudWatch agent can be found, typically `C:\Program Files`.

We recommend keeping a backup of the JSON in this file on a location other than the node itself.

For information about creating a parameter, see [Creating Systems Manager parameters \(p. 279\)](#).

For more information about the CloudWatch agent, see [Collecting metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch agent](#) in the *Amazon CloudWatch User Guide*.

## 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 node 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:

```
$(Env:ProgramFiles)\\Amazon\\SSM\\Plugins\\awsCloudWatch\\\nAWS.EC2.Windows.CloudWatch.json
```

`{Env:ProgramFiles}` represents the location where the Amazon directory 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 node itself.

## Two: To uninstall the CloudWatch agent (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Run Command**, and then choose **Run command**.
3. In the **Command document** list, choose `AWS-ConfigureAWSPackage`.
4. For **Action**, choose **Uninstall**.
5. For **Name**, enter `AmazonCloudWatchAgent`.
6. In the **Targets** section, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes still processing the command might also send errors.
8. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

9. 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 \(p. 1462\)](#).

10. Choose **Run**.

## Three: To turn log collection back on in SSM Agent (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Run Command**, and then choose **Run command**.
3. In the **Command document** list, choose `AWS-ConfigureCloudWatch`.
4. For **Status**, choose **Enabled**.
5. For **Properties**, paste the contents of the old config data you saved to the text file.
6. In the **Targets** section, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes still processing the command might also send errors.

8. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

9. 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 \(p. 1462\)](#).

10. Choose **Run**.

## Sending SSM Agent logs to CloudWatch Logs

AWS Systems Manager Agent (SSM Agent) is Amazon software that runs on your EC2 instances, edge devices, and on-premises servers 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 \(p. 68\)](#).

In addition, using the following steps, you can configure SSM Agent to send log data to Amazon CloudWatch Logs.

### Before you begin

Create a log group in CloudWatch Logs. For more information, see [Getting started with CloudWatch Logs in the Amazon CloudWatch Logs User Guide](#).

### To configure SSM Agent to send logs to CloudWatch

1. Log into a node and locate the following file:

**Linux**

On most Linux node types: /etc/amazon/ssm/seelog.xml.template.

On Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS: /snap/amazon-ssm-agent/current/seelog.xml.template

#### macOS

```
/opt/aws/ssm/seelog.xml.template
```

#### Windows

```
%ProgramFiles%\Amazon\SSM\seelog.xml.template
```

2. Change the file name from seelog.xml.template to seelog.xml

##### Note

On Ubuntu Server 20.10 STR & 20.04, 18.04, and 16.04 LTS, the file seelog.xml must be created in the directory /etc/amazon/ssm/. You can create this directory and file by running the following commands.

```
sudo mkdir -p /etc/amazon/ssm
```

```
sudo cp -pr /snap/amazon-ssm-agent/current/* /etc/amazon/ssm
```

```
sudo cp -p /etc/amazon/ssm/seelog.xml.template /etc/amazon/ssm/seelog.xml
```

3. Open the seelog.xml file in a text editor, and locate the following section.

#### Linux and macOS

```
<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>
```

#### 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>
</outputs>
```

4. Edit the file, and add a *custom name* element after the closing </filter> tag. In the following example, the custom name has been specified as cloudwatch\_receiver.

#### Linux and macOS

```
<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>
<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 node.
6. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
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 node ID.

## Monitoring Automation metrics using Amazon CloudWatch

*Metrics* are the fundamental concept in Amazon 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.

Automation is a capability of AWS Systems Manager. Systems Manager publishes metrics about Automation usage to CloudWatch. This allows you to set alarms based on those metrics.

### To view Automation metrics in the CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Choose **SSM**.
4. On the **Metrics** tab, choose **Usage**, and then choose **By AWS Resource**.
5. In the search box near the list of metrics, enter **SSM**.

### To view Automation metrics using the AWS CLI

Open a command prompt, and use the following command.

```
aws cloudwatch list-metrics \
--namespace "AWS/Usage"
```

## Automation metrics

Systems Manager sends the following Automation metrics to CloudWatch.

Metric	Description
ConcurrentAutomationUsage	The number of automations running at the same time in the current AWS account and AWS Region.
QueuedAutomationUsage	The number of automations currently queued that have not started and have a status of Pending.

For more information about working with CloudWatch metrics, see the following topics in the *Amazon CloudWatch User Guide*:

- Metrics
- Using Amazon CloudWatch metrics
- Using Amazon CloudWatch alarms

## Monitoring Run Command metrics using Amazon CloudWatch

Metrics are the fundamental concept in Amazon 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, allowing you to set alarms based on those metrics. Run Command is a capability of AWS Systems Manager. 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 isn't reported for any of those hours. For more information about command status values, see [Understanding command statuses \(p. 1013\)](#).

### To view metrics in the CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose Metrics.
3. In the Alarms by AWS service area, for Services, choose SSM-Run Command.

### To view metrics using the AWS CLI

Open a command prompt, and use the following command.

```
aws cloudwatch list-metrics --namespace "AWS/SSM-RunCommand"
```

To list all available metrics, use the following command.

```
aws cloudwatch list-metrics
```

## Systems Manager Run Command metrics and dimensions

Systems Manager sends Run Command command metrics to CloudWatch one time every minute.

Systems Manager sends the following command metrics to CloudWatch.

**Note**

These metrics use Count as the unit, so Sum and SampleCount are the most useful statistics.

Metric	Description
CommandsDeliveryTimedOut	The number of commands that have a terminal status of Delivery Timed Out.
CommandsFailed	The number of commands that have a terminal status of Failed.
CommandsSucceeded	The number of commands that have a terminal status of Success.

For more information about working with CloudWatch metrics, see the following topics in the *Amazon CloudWatch User Guide*:

- [Metrics](#)
- [Using Amazon CloudWatch metrics](#)
- [Using Amazon CloudWatch alarms](#)

## Logging AWS Systems Manager API calls with AWS CloudTrail

AWS 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 turn on continual 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 activated 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](#) in the *AWS CloudTrail User Guide*.

For an ongoing record of events in your AWS account, including events for Systems Manager, create a trail. A trail allows CloudTrail to deliver log files to an S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS 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:

- [Creating a trail for your AWS account](#)
  - [AWS service integrations with CloudTrail Logs](#)
  - [Configuring Amazon SNS Notifications for CloudTrail](#)
  - [Receiving CloudTrail log files from multiple Regions](#)
- [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 EventBridge \(console\) \(p. 977\)](#).

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 AWS account root user credentials 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 allows 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 operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they aren't displayed in any specific order.

### Example 1: DeleteDocument

The following example shows a CloudTrail log entry that demonstrates the `DeleteDocument` operation 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",
 "sourceIPAddress": "203.0.113.11",
 "userAgent": "example-user-agent-string",
 "requestParameters": {
 "name": "example-Document"
 },
 "responseElements": null,
 "requestID": "86168559-75e9-11e4-8cf8-75d18EXAMPLE",
 "eventId": "832b82d5-d474-44e8-a51d-093ccEXAMPLE",
 "resources": [
 {
 "ARN": "arn:aws:ssm:us-east-2:123456789012:document/example-Document",
 "accountId": "123456789012"
 }
],
 "eventType": "AwsApiCall",
 "recipientAccountId": "123456789012"
}

```

### **Example 2: StartConnection**

The following example shows a CloudTrail log entry for a user who starts an RDP connection using Fleet Manager in the US East (Ohio) Region (us-east-2). The underlying API action is StartConnection.

```

{
 "eventVersion": "1.08",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "AKIAI44QH8DHBEXAMPLE",
 "arn": "arn:aws:sts::123456789012:assumed-role/exampleRole",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "AKIAI44QH8DHBEXAMPLE",
 "arn": "arn:aws:sts::123456789012:assumed-role/exampleRole",
 "accountId": "123456789012",
 "userName": "exampleRole"
 },
 "webIdFederationData": {},
 "attributes": {
 "creationDate": "2021-12-13T14:57:05Z",
 "mfaAuthenticated": "false"
 }
 }
 },
 "eventTime": "2021-12-13T16:50:41Z",
 "eventSource": "ssm-guiconnect.amazonaws.com",
 "eventName": "StartConnection",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "34.230.45.60",
 "userAgent": "example-user-agent-string",
 "requestParameters": {
 "AuthType": "Credentials",
 "Protocol": "RDP",
 "ConnectionType": "SessionManager",
 "InstanceId": "i-02573cafefEXAMPLE"
 }
}

```

```
 },
 "responseElements": {
 "ConnectionArn": "arn:aws:ssm-guiconnect:us-east-2:123456789012:connection/fcb810cd-241f-4aae-9ee4-02d59EXAMPLE",
 "ConnectionKey": "71f9629f-0f9a-4b35-92f2-2d253EXAMPLE",
 "ClientToken": "49af0f92-d637-4d47-9c54-ea51aEXAMPLE",
 "requestId": "d466710f-2adf-4e87-9464-055b2EXAMPLE"
 },
 "requestID": "d466710f-2adf-4e87-9464-055b2EXAMPLE",
 "eventID": "fc514f57-ba19-4e8b-9079-c2913EXAMPLE",
 "readOnly": false,
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "123456789012",
 "eventCategory": "Management"
 }
}
```

## Logging Automation action output with CloudWatch Logs

Automation, a capability of AWS Systems Manager, integrates with Amazon CloudWatch Logs. You can send the output from `aws:executeScript` actions in your runbooks to the log group you specify. Systems Manager doesn't create a log group or any log streams for documents that don't use `aws:executeScript` actions. If the document does use `aws:executeScript`, the output sent to CloudWatch Logs only pertains to those actions. You can use the `aws:executeScript` action output stored in your CloudWatch Logs log group for debugging and troubleshooting purposes. If you choose a log group that is encrypted, the `aws:executeScript` action output is also encrypted. Logging output from `aws:executeScript` actions is an account-level setting.

To send action output to CloudWatch Logs, the IAM user or role that runs the automation must have permissions for the following operations:

- `logs>CreateLogGroup`
- `logs>CreateLogStream`
- `logsDescribeLogGroups`
- `logsDescribeLogStreams`
- `logsPutLogEvents`

### To send action output to CloudWatch Logs (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Automation**.
3. Choose the **Preferences** tab, and then choose **Edit**.
4. Select the check box next to **Send output to CloudWatch Logs**.
5. (Recommended) Select the check box next to **Encrypt log data**. With this option turned on, log data is encrypted using the server-side encryption key specified for the log group. If you don't want to encrypt the log data that is sent to CloudWatch Logs, clear the check box. Clear the check box if encryption isn't allowed on the log group.
6. For **CloudWatch Logs log group**, to specify the existing CloudWatch Logs log group in your AWS account that you want to send action output to, select one of the following:
  - **Send output to the default log group** – If the default log group doesn't exist (`/aws/ssm/automation/executeScript`), Automation creates it for you.

- **Choose from a list of log groups** – Select a log group that has already been created in your account to store action output.
- **Enter a log group name** – Enter the name of a log group in the text box that has already been created in your account to store action output.

7. Choose **Save**.

### To send action output to CloudWatch Logs (command line)

1. Open your preferred command line tool and run the following command to update the action output destination.

Linux & macOS

```
aws ssm update-service-setting \
--setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/automation/
customer-script-log-destination \
--setting-value CloudWatch
```

Windows

```
aws ssm update-service-setting ^
--setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/automation/
customer-script-log-destination ^
--setting-value CloudWatch
```

PowerShell

```
Update-SSMSERVICESETTING `
-SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/automation/
customer-script-log-destination" `
-SettingValue "CloudWatch"
```

There is no output if the command succeeds.

2. Run the following command to specify the log group you want to send action output to.

Linux & macOS

```
aws ssm update-service-setting \
--setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/automation/
customer-script-log-group-name \
--setting-value my-log-group
```

Windows

```
aws ssm update-service-setting ^
--setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/automation/
customer-script-log-group-name ^
--setting-value my-log-group
```

PowerShell

```
Update-SSMSERVICESETTING `
-SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/automation/
customer-script-log-group-name" `
```

```
-SettingValue "my-log-group"
```

There is no output if the command succeeds.

- Run the following command to view the current service settings for Automation action logging preferences in the current AWS account and AWS Region.

Linux & macOS

```
aws ssm get-service-setting \
--setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/automation/
customer-script-log-destination
```

Windows

```
aws ssm get-service-setting ^
--setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/automation/
customer-script-log-destination
```

PowerShell

```
Get-SSMServiceSetting `
-SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/automation/
customer-script-log-destination"
```

The command returns information like the following.

```
{
 "ServiceSetting": {
 "Status": "Customized",
 "LastModifiedDate": 1613758617.036,
 "SettingId": "/ssm/automation/customer-script-log-destination",
 "LastModifiedUser": "arn:aws:sts::123456789012:assumed-role/Administrator/
User_1",
 "SettingValue": "CloudWatch",
 "ARN": "arn:aws:ssm:us-east-2:123456789012:servicesetting/ssm/automation/
customer-script-log-destination"
 }
}
```

## Configuring Amazon CloudWatch Logs for Run Command

When you send a command by using Run Command, a capability of AWS Systems Manager, you can specify where you want to send the command output. By default, Systems Manager returns only the first 48,000 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 node or on-premises hybrid machine to use the AWS Identity and Access Management (IAM) managed policies `AmazonSSMManagedInstanceCore` and

If your node has the `CloudWatchAgentServerPolicy`, then your node requires no additional configuration to send output to CloudWatch Logs. Choose this option if sending commands from the console, or add the `cloud-watch-output-config` section and `CloudWatchOutputEnabled` parameter if using the AWS Command Line Interface (AWS CLI), AWS Tools for Windows PowerShell, or an API operation. 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 \(p. 21\)](#). For information about adding policies to a service role for on-premises servers and virtual machines that you plan to use as managed nodes, see [Create an IAM service role for a hybrid environment \(p. 36\)](#).

For information about updating an existing instance profile, see [Add permissions to a Systems Manager instance profile \(console\) \(p. 23\)](#).

If you're using a custom policy on your nodes, update the policy on each node 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:DescribeLogGroups",
 "Resource": "*"
},
{
 "Effect": "Allow",
 "Action": [
 "logs:CreateLogGroup",
 "logs:CreateLogStream",
 "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, specify the `cloud-watch-output-config` section in your command. This section allows you to specify the `CloudWatchOutputEnabled` parameter, and optionally, the `CloudWatchLogGroupName` parameter. Here is an example.

Linux & macOS

```
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}'
```

## Windows

```
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 is displayed 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 EventBridge

Amazon EventBridge is a serverless event bus service that allows you to connect your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your own applications, software-as-a-service (SaaS) applications, and AWS services and routes that data to targets such as AWS Lambda. You can set up routing rules to determine where to send your data to build application architectures that react in real time to all of your data sources. EventBridge allows you to build event driven architectures, which are loosely coupled and distributed.

EventBridge was formerly called Amazon CloudWatch Events. EventBridge includes new features that allow you to receive events from SaaS partners and your own applications. Existing CloudWatch Events users can access their existing default bus, rules, and events in the new EventBridge console and in the CloudWatch Events console. EventBridge uses the same CloudWatch Events API, so all of your existing CloudWatch Events API usage remains the same.

EventBridge can add events from dozens of AWS services to your rules, and targets from over 20 AWS services.

EventBridge provides support for both AWS Systems Manager events and Systems Manager targets.

### Supported Systems Manager event types

Among the many types of Systems Manager events that EventBridge can detect are:

- A maintenance window being turned off.
- An Automation workflow completing successfully. Automation is a capability of AWS Systems Manager.
- A managed node being out of patch compliance.
- A parameter value being updated.

EventBridge supports events from the following AWS Systems Manager capabilities:

- Automation (Events are emitted on a best effort basis.)
- Change Calendar (Events are emitted on a best effort basis.)
- Compliance
- Inventory (Events are emitted on a best effort basis.)
- Maintenance Windows (Events are emitted on a best effort basis.)
- Parameter Store (Events are emitted on a best effort basis.)
- Run Command (Events are emitted on a best effort basis.)
- State Manager (Events are emitted on a best effort basis.)

For complete details about supported Systems Manager event types, see [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#) and [Amazon EventBridge event examples for Systems Manager \(p. 1452\)](#).

### Supported Systems Manager target types

EventBridge supports the following three Systems Manager capabilities as targets of an event rule:

- Running an Automation workflow
- Running a Run Command Command document (Events are emitted on a best effort basis.)
- Creating an OpsCenter OpsItem

For suggested ways you might use these targets, see [Amazon EventBridge target examples for Systems Manager \(p. 1461\)](#).

For more information about how to get started with EventBridge and set up rules, see [Getting started with Amazon EventBridge](#) in the [Amazon EventBridge User Guide](#). For complete information about working with EventBridge, see the [Amazon EventBridge User Guide](#).

#### Topics

- [Configuring EventBridge for Systems Manager events \(p. 1450\)](#)
- [Amazon EventBridge event examples for Systems Manager \(p. 1452\)](#)
- [Amazon EventBridge target examples for Systems Manager \(p. 1461\)](#)

## Configuring EventBridge for Systems Manager events

You can use Amazon EventBridge to perform a target event when supported AWS Systems Manager status changes, state changes, or other conditions occur. You can create a rule that runs whenever there is a state or status transition, or when there is a transition to one or more states that are of interest.

The following procedure provides general steps for creating an EventBridge rule that engages when a specified event is emitted by Systems Manager. For a list of procedures in this user guide that address specific scenarios, see **Related content** at the end of this topic.

**Note**

When a service in your AWS account emits an event, it always goes to your account's default event bus. To write a rule that responds to events from AWS services in your account, associate it with the default event bus. You can create a rule on a custom event bus that looks for events from AWS services, but this rule only engages when you receive such an event from another account through cross-account event delivery. For more information, see [Sending and receiving Amazon EventBridge events between AWS accounts](#) in the *Amazon EventBridge User Guide*.

## To configure EventBridge for Systems Manager events

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**, and then choose **Create rule**.

-or-

If the EventBridge home page opens first, choose **Create rule**.

3. Enter a name and description for the rule.

A rule can't have the same name as another rule in the same AWS Region and on the same event bus.

4. For **Define pattern**, choose **Event pattern**.
5. For **Event matching pattern**, choose **Pre-defined pattern by service**.
6. For **Service provider**, choose **AWS**.
7. For **Service name**, choose **Systems Manager**.
8. For **Event type**, do one of the following:

- Choose **All Events**.

If you choose **All Events**, all events emitted by Systems Manager will match the rule. Be aware that this option can result in many event target actions.

- Choose the type of Systems Manager event to use for this rule. EventBridge supports events from the following AWS Systems Manager capabilities:
  - Automation
  - Change Calendar
  - Compliance
  - Inventory
  - Maintenance Windows
  - Parameter Store
  - Run Command
  - State Manager

**Note**

For Systems Manager actions that aren't supported by EventBridge, you can choose an AWS API call through CloudTrail to create an event rule that is based on an API call, which are recorded by CloudTrail. For an example, see [Monitoring session activity using Amazon EventBridge \(console\) \(p. 977\)](#).

(Optional) If you want to customize the event pattern, choose **Edit** next to **Event pattern**, make your changes, and choose **Save**.

9. If you chose a Systems Manager capability in step 8, do one of the following:

- For targets to be invoked when any supported event type for the selected capability occurs, choose **Any detail type**.
- For targets to be invoked when only certain event types for the selected capability occur, choose **Specific detail type(s)**, and then choose one or more types from the list.

For complete details about supported detail types, see [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#).

10. If you chose a Systems Manager capability in step 8, choose whether to invoke targets for all or only certain detail types, statuses, or other supported options. The available options depend on the capability you have selected.
11. In the **Select event bus** area, choose the event bus that you want to associate with this rule. If you want this rule to respond to matching events that come from your own AWS account, select **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
12. In the **Select targets** area, choose the AWS service that is to act when an event of the selected type is detected.
13. In the other fields in this section, enter information specific to this target type, if any is needed.
14. For many target types, EventBridge needs permissions to send events to the target. In these cases, EventBridge can create the AWS Identity and Access Management (IAM) role needed for your rule to run:
  - To create an IAM role automatically, choose **Create a new role for this specific resource**.
  - To use an IAM role that you created earlier, choose **Use existing role**.
15. (Optional) Choose **Add target** to add another target for this rule.
16. (Optional) In the **Tags** area, enter one or more tags for the rule. For more information, see [EventBridge tagging in the Amazon EventBridge User Guide](#).
17. Choose **Create**.

## Related content

- [Creating an EventBridge event that uses a runbook \(console\) \(p. 436\)](#)
- [Walkthrough: Using input transformers with Automation \(p. 686\)](#)
- [Remediating compliance issues using EventBridge \(p. 843\)](#)
- [Viewing inventory delete actions in EventBridge \(p. 893\)](#)
- [Configuring EventBridge to automatically create OpsItems for specific events \(p. 203\)](#)
- [Configuring EventBridge for parameters \(p. 276\)](#)
- [Configuring EventBridge for parameter policies \(p. 277\)](#)

# Amazon EventBridge event examples for Systems Manager

The following are examples, in JSON format, of supported EventBridge events for AWS Systems Manager.

## Systems Manager event types

- [AWS Systems Manager Automation Events \(p. 1453\)](#)
- [AWS Systems ManagerChange Calendar Events \(p. 1453\)](#)
- [AWS Systems Manager Compliance Events \(p. 1454\)](#)
- [AWS Systems ManagerMaintenance Windows Events \(p. 1456\)](#)

- [AWS Systems ManagerParameter Store Events \(p. 1458\)](#)
- [AWS Systems ManagerRun Command Events \(p. 1459\)](#)
- [AWS Systems ManagerState Manager Events \(p. 1460\)](#)

## AWS Systems Manager Automation Events

### Automation Step Status-change Notification

```
{
 "version": "0",
 "id": "eeca120b-a321-433e-9635-dab369006a6b",
 "detail-type": "EC2 Automation Step Status-change Notification",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2016-11-29T19:43:35Z",
 "region": "us-east-1",
 "resources": ["arn:aws:ssm:us-east-2:123456789012:automation-
execution/333ba70b-2333-48db-b17e-a5e69c6f4d1c",
 "arn:aws:ssm:us-east-2:123456789012:automation-definition/runcommand1:1"],
 "detail": {
 "ExecutionId": "333ba70b-2333-48db-b17e-a5e69c6f4d1c",
 "Definition": "runcommand1",
 "DefinitionVersion": 1.0,
 "Status": "Success",
 "EndTime": "Nov 29, 2016 7:43:25 PM",
 "StartTime": "Nov 29, 2016 7:43:23 PM",
 "Time": 2630.0,
 "StepName": "runFixedCmds",
 "Action": "aws:runCommand"
 }
}
```

### Automation Execution Status-change Notification

```
{
 "version": "0",
 "id": "d290ece9-1088-4383-9df6-cd5b4ac42b99",
 "detail-type": "EC2 Automation Execution Status-change Notification",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2016-11-29T19:43:35Z",
 "region": "us-east-2",
 "resources": ["arn:aws:ssm:us-east-2:123456789012:automation-
execution/333ba70b-2333-48db-b17e-a5e69c6f4d1c",
 "arn:aws:ssm:us-east-2:123456789012:automation-definition/runcommand1:1"],
 "detail": {
 "ExecutionId": "333ba70b-2333-48db-b17e-a5e69c6f4d1c",
 "Definition": "runcommand1",
 "DefinitionVersion": 1.0,
 "Status": "Success",
 "StartTime": "Nov 29, 2016 7:43:20 PM",
 "EndTime": "Nov 29, 2016 7:43:26 PM",
 "Time": 5753.0,
 "ExecutedBy": "arn:aws:iam::123456789012:user/username"
 }
}
```

## AWS Systems ManagerChange Calendar Events

The following are examples of the events for AWS Systems Manager Change Calendar.

**Note**

State changes for calendars shared from other AWS accounts are not currently supported.

**Calendar OPEN**

```
{
 "version": "0",
 "id": "47a3f03a-f30d-1011-ac9a-du3bdEXAMPLE",
 "detail-type": "Calendar State Change",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2020-09-19T18:00:07Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:document/MyCalendar"
],
 "detail": {
 "state": "OPEN",
 "atTime": "2020-09-19T18:00:07Z",
 "nextTransitionTime": "2020-10-11T18:00:07Z"
 }
}
```

**Calendar CLOSED**

```
{
 "version": "0",
 "id": "f30df03a-1011-ac9a-47a3-f761eEXAMPLE",
 "detail-type": "Calendar State Change",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2020-09-17T21:40:02Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:document/MyCalendar"
],
 "detail": {
 "state": "CLOSED",
 "atTime": "2020-08-17T21:40:00Z",
 "nextTransitionTime": "2020-09-19T18:00:07Z"
 }
}
```

## AWS Systems Manager Compliance Events

The following are examples of the events for AWS Systems Manager Compliance.

**Association Compliant**

```
{
 "version": "0",
 "id": "01234567-0123-0123-0123-012345678901",
 "detail-type": "Configuration Compliance State Change",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2017-07-17T19:03:26Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:managed-instance/i-01234567890abcdef"
],
 "detail": {
 "status": "Compliant",
 "compliance_type": "AWS_SSM_INSTANCE_INFORMATION",
 "compliance_status": "Compliant",
 "compliance_message": "The instance is compliant with the configuration.
 The instance is running the correct operating system version.
 The instance has the correct security group settings.",
 "compliance_severity": "Info",
 "compliance_resource_id": "i-01234567890abcdef",
 "compliance_resource_type": "AWS::SSM::ManagedInstance",
 "compliance_start_time": "2017-07-17T19:03:26Z",
 "compliance_end_time": "2017-07-17T19:03:26Z",
 "compliance_transition_time": "2017-07-17T19:03:26Z",
 "compliance_transition_status": "Compliant",
 "compliance_transition_severity": "Info",
 "compliance_transition_message": "The instance became compliant at 2017-07-17T19:03:26Z."
 }
}
```

```
 "last-runtime": "2017-01-01T10:10:10Z",
 "compliance-status": "compliant",
 "resource-type": "managed-instance",
 "resource-id": "i-01234567890abcdef",
 "compliance-type": "Association"
 }
}
```

### Association Non-Compliant

```
{
 "version": "0",
 "id": "01234567-0123-0123-0123-012345678901",
 "detail-type": "Configuration Compliance State Change",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2017-07-17T19:02:31Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:managed-instance/i-01234567890abcdef"
],
 "detail": {
 "last-runtime": "2017-01-01T10:10:10Z",
 "compliance-status": "non_compliant",
 "resource-type": "managed-instance",
 "resource-id": "i-01234567890abcdef",
 "compliance-type": "Association"
 }
}
```

### Patch Compliant

```
{
 "version": "0",
 "id": "01234567-0123-0123-0123-012345678901",
 "detail-type": "Configuration Compliance State Change",
 "source": "aws.123456789012",
 "account": "123456789012",
 "time": "2017-07-17T19:03:26Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:managed-instance/i-01234567890abcdef"
],
 "detail": {
 "resource-type": "managed-instance",
 "resource-id": "i-01234567890abcdef",
 "compliance-status": "compliant",
 "compliance-type": "Patch",
 "patch-baseline-id": "PB789",
 "severity": "critical"
 }
}
```

### Patch Non-Compliant

```
{
 "version": "0",
 "id": "01234567-0123-0123-0123-012345678901",
 "detail-type": "Configuration Compliance State Change",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2017-07-17T19:02:31Z",
```

```

"region": "us-east-2",
"resources": [
 "arn:aws:ssm:us-east-2:123456789012:managed-instance/i-01234567890abcdef"
],
"detail": {
 "resource-type": "managed-instance",
 "resource-id": "i-01234567890abcdef",
 "compliance-status": "non_compliant",
 "compliance-type": "Patch",
 "patch-baseline-id": "PB789",
 "severity": "critical"
}
}

```

## AWS Systems Manager Maintenance Windows Events

The following are examples of the events for Systems Manager Maintenance Windows.

### Register a Target

The other valid status value is Deregistered.

```

{
 "version": "0",
 "id": "01234567-0123-0123-0123-0123456789ab",
 "detail-type": "Maintenance Window Target Registration Notification",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2016-11-16T00:58:37Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:maintenancewindow/mw-0ed7251d3fcf6e0c2",
 "arn:aws:ssm:us-east-2:123456789012:windowtarget/
e7265f13-3cc5-4f2f-97a9-7d3ca86c32a6"
],
 "detail": {
 "window-target-id": "e7265f13-3cc5-4f2f-97a9-7d3ca86c32a6",
 "window-id": "mw-0ed7251d3fcf6e0c2",
 "status": "REGISTERED"
 }
}

```

### Window Execution Type

The other valid status values are PENDING, IN\_PROGRESS, SUCCESS, FAILED, TIMED\_OUT, and SKIPPED\_OVERLAPPING.

```

{
 "version": "0",
 "id": "01234567-0123-0123-0123-0123456789ab",
 "detail-type": "Maintenance Window Execution State-change Notification",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2016-11-16T01:00:57Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:maintenancewindow/mw-123456789012345678"
],
 "detail": {
 "start-time": "2016-11-16T01:00:56.427Z",
 "end-time": "2016-11-16T01:00:57.070Z",
 "window-id": "mw-0ed7251d3fcf6e0c2",
 "status": "IN_PROGRESS"
 }
}

```

```
 "window-execution-id": "b60fb56e-776c-4e5c-84ee-123456789012",
 "status": "TIMED_OUT"
 }
}
```

### Task Execution Type

The other valid status values are IN\_PROGRESS, SUCCESS, FAILED, and TIMED\_OUT.

```
{
 "version": "0",
 "id": "01234567-0123-0123-0123-0123456789ab",
 "detail-type": "Maintenance Window Task Execution State-change Notification",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2016-11-16T01:00:56Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:maintenancewindow/mw-123456789012345678"
],
 "detail": {
 "start-time": "2016-11-16T01:00:56.759Z",
 "task-execution-id": "6417e808-7f35-4d1a-843f-123456789012",
 "end-time": "2016-11-16T01:00:56.847Z",
 "window-id": "mw-0ed7251d3fcf6e0c2",
 "window-execution-id": "b60fb56e-776c-4e5c-84ee-123456789012",
 "status": "TIMED_OUT"
 }
}
```

### Task Target Processed

The other valid status values are IN\_PROGRESS, SUCCESS, FAILED, and TIMED\_OUT.

```
{
 "version": "0",
 "id": "01234567-0123-0123-0123-0123456789ab",
 "detail-type": "Maintenance Window Task Target Invocation State-change Notification",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2016-11-16T01:00:57Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:maintenancewindow/mw-123456789012345678"
],
 "detail": {
 "start-time": "2016-11-16T01:00:56.427Z",
 "end-time": "2016-11-16T01:00:57.070Z",
 "window-id": "mw-0ed7251d3fcf6e0c2",
 "window-execution-id": "b60fb56e-776c-4e5c-84ee-123456789012",
 "task-execution-id": "6417e808-7f35-4d1a-843f-123456789012",
 "window-target-id": "e7265f13-3cc5-4f2f-97a9-123456789012",
 "status": "TIMED_OUT",
 "owner-information": "Owner"
 }
}
```

### Window State Change

The valid status values are ENABLED and DISABLED.

```
{
```

```
"version": "0",
"id": "01234567-0123-0123-0123-0123456789ab",
"detail-type": "Maintenance Window State-change Notification",
"source": "aws.ssm",
"account": "123456789012",
"time": "2016-11-16T00:58:37Z",
"region": "us-east-2",
"resources": [
 "arn:aws:ssm:us-east-2:123456789012:maintenancewindow/mw-123456789012345678"
],
"detail": {
 "window-id": "mw-123456789012",
 "status": "DISABLED"
}
}
```

## AWS Systems Manager Parameter Store Events

The following are examples of the events for Systems Manager Parameter Store.

### Create Parameter

```
{
 "version": "0",
 "id": "6a7e4feb-b491-4cf7-a9f1-bf3703497718",
 "detail-type": "Parameter Store Change",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2017-05-22T16:43:48Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:parameter/MyExampleParameter"
],
 "detail": {
 "operation": "Create",
 "name": "MyExampleParameter",
 "type": "String",
 "description": "Sample Parameter"
 }
}
```

### Update Parameter

```
{
 "version": "0",
 "id": "9547ef2d-3b7e-4057-b6cb-5fdf09ee7c8f",
 "detail-type": "Parameter Store Change",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2017-05-22T16:44:48Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:parameter/MyExampleParameter"
],
 "detail": {
 "operation": "Update",
 "name": "MyExampleParameter",
 "type": "String",
 "description": "Sample Parameter"
 }
}
```

### Delete Parameter

```
{
 "version": "0",
 "id": "80e9b391-6a9b-413c-839a-453b528053af",
 "detail-type": "Parameter Store Change",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2017-05-22T16:45:48Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2:123456789012:parameter/MyExampleParameter"
],
 "detail": {
 "operation": "Delete",
 "name": "MyExampleParameter",
 "type": "String",
 "description": "Sample Parameter"
 }
}
```

## AWS Systems Manager Run Command Events

### Run Command Status-change Notification

```
{
 "version": "0",
 "id": "51c0891d-0e34-45b1-83d6-95db273d1602",
 "detail-type": "EC2 Command Status-change Notification",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2016-07-10T21:51:32Z",
 "region": "us-east-2",
 "resources": ["arn:aws:ec2:us-east-2:123456789012:instance/i-abcd1111"],
 "detail": {
 "command-id": "e8d3c0e4-71f7-4491-898f-c9b35bee5f3b",
 "document-name": "AWS-RunPowerShellScript",
 "expire-after": "2016-07-14T22:01:30.049Z",
 "parameters": {
 "executionTimeout": ["3600"],
 "commands": ["date"]
 },
 "requested-date-time": "2016-07-10T21:51:30.049Z",
 "status": "Success"
 }
}
```

### Run Command Invocation Status-change Notification

```
{
 "version": "0",
 "id": "4780e1b8-f56b-4de5-95f2-95db273d1602",
 "detail-type": "EC2 Command Invocation Status-change Notification",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2016-07-10T21:51:32Z",
 "region": "us-east-2",
 "resources": ["arn:aws:ec2:us-east-2:123456789012:instance/i-abcd1111"],
 "detail": {
 "command-id": "e8d3c0e4-71f7-4491-898f-c9b35bee5f3b",
 "document-name": "AWS-RunPowerShellScript",
 "instance-id": "i-9bb89e2b",
 "status": "Success"
 }
}
```

```
 "requested-date-time": "2016-07-10T21:51:30.049Z",
 "status": "Success"
 }
}
```

## AWS Systems Manager State Manager Events

### State Manager Association State Change

```
{
 "version": "0",
 "id": "db839caf-6f6c-40af-9a48-25b2ae2b7774",
 "detail-type": "EC2 State Manager Association State Change",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2017-05-16T23:01:10Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ssm:us-east-2::document/AWS-RunPowerShellScript"
],
 "detail": {
 "association-id": "6e37940a-23ba-4ab0-9b96-5d0a1a05464f",
 "document-name": "AWS-RunPowerShellScript",
 "association-version": "1",
 "document-version": "Optional.empty",
 "targets": "[{\\"key\\": \"InstanceIds\", \\"values\\\": [\"i-12345678\"]}]",
 "creation-date": "2017-02-13T17:22:54.458Z",
 "last-successful-execution-date": "2017-05-16T23:00:01Z",
 "last-execution-date": "2017-05-16T23:00:01Z",
 "last-updated-date": "2017-02-13T17:22:54.458Z",
 "status": "Success",
 "association-status-aggregated-count": "{\"Success\":1}",
 "schedule-expression": "cron(0 */30 * * * ? *)",
 "association-cwe-version": "1.0"
 }
}
```

### State Manager Instance Association State Change

```
{
 "version": "0",
 "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
 "detail-type": "EC2 State Manager Instance Association State Change",
 "source": "aws.ssm",
 "account": "123456789012",
 "time": "2017-02-23T15:23:48Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ec2:us-east-2:123456789012:instance/i-12345678",
 "arn:aws:ssm:us-east-2:123456789012:document/my-custom-document"
],
 "detail": {
 "association-id": "34fc7e0-9a14-4984-9989-0e04e3f60bd8",
 "instance-id": "i-12345678",
 "document-name": "my-custom-document",
 "document-version": "1",
 "targets": "[{\\"key\\": \"instanceids\", \\"values\\\": [\"i-12345678\"]}]",
 "creation-date": "2017-02-23T15:23:48Z",
 "last-successful-execution-date": "2017-02-23T16:23:48Z",
 "last-execution-date": "2017-02-23T16:23:48Z",
 "status": "Success",
 "detailed-status": ""
 }
}
```

```
 "error-code": "testErrorCode",
 "execution-summary": "testExecutionSummary",
 "output-url": "sampleurl",
 "instance-association-cwe-version": "1"
}
```

## Amazon EventBridge target examples for Systems Manager

When you specify the target to invoke in an Amazon EventBridge rule, you can choose from over 20 target types and add up to five targets to each rule.

Of the various targets, you can choose from Automation, OpsCenter, and Run Command, which are capabilities of AWS Systems Manager, as target actions when an EventBridge event occurs.

The following are several examples of ways you can use these capabilities as the target of an EventBridge rule.

### Automation examples

You can configure an EventBridge rule to start Automation workflows when events such as the following occur:

- When an Amazon CloudWatch alarm reports that a managed node has failed a status check (`StatusCheckFailed_Instance=1`), run the `AWSSupport-ExecuteEC2Rescue` Automation runbook on the node.
- When an `EC2 Instance State-change Notification` event occurs because a new Amazon Elastic Compute Cloud (Amazon EC2) instance is running, run the `AWS-AttachEBSVolume` Automation runbook on the instance.
- When an Amazon Elastic Block Store (Amazon EBS) volume is created and available, run the `AWS-CreateSnapshot` Automation runbook on the volume.

### OpsCenter examples

You can configure an EventBridge rule to create a new OpsItem when incidents such as the following occur:

- A throttling event for Amazon DynamoDB occurs, or Amazon EBS volume performance has degraded.
- An Amazon EC2 Auto Scaling group fails to launch a node, or a Systems Manager Automation workflow fails.
- An EC2 instance changes state from `Running` to `Stopped`.

### Run Command examples

You can configure an EventBridge rule to run a Systems Manager Command document in Run Command when events such as the following occur:

- When an Auto Scaling group is about to end, a Run Command script could capture the log files from the node before it is ended.
- When a new node is created in an Auto Scaling group, a Run Command target action could turn on the web server role or install software on the node.
- When a managed node is found to be out of compliance, a Run Command target action could update patches on the node by running the `AWS-RunPatchBaseline` document.

# 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 Run Command or Maintenance Windows, which are capabilities of AWS Systems Manager. 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 nodes, you can receive a notification for each copy of the command sent to a specific node. 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 \(p. 1467\)](#).

## Configure Amazon SNS notifications for AWS Systems Manager

Run Command and Maintenance Windows tasks that are registered to a maintenance window can send Amazon SNS notifications for command tasks that enter the following statuses:

- In Progress
- Success
- Failed
- Timed Out
- Canceled

For information about the conditions that cause a command to enter one of these statuses, see [Understanding command statuses \(p. 1013\)](#).

**Note**

Commands sent using Run Command also report Canceling and Pending status. These statuses aren't 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.

Field	Type	Description
eventTime	String	The time that the event was initiated. The timestamp is important because Amazon SNS doesn't guarantee message delivery order. Example: 2016-04-26T13:15:30Z
documentName	String	The name of the SSM document used to run this command.

Field	Type	Description
commandId	String	The ID generated by Run Command after the command was sent.
expiresAfter	Date	If this time is reached and the command hasn't already started executing, it won't run.
outputS3BucketName	String	The Amazon Simple Storage Service (Amazon S3) bucket where the responses to the command execution should be stored.
outputS3KeyPrefix	String	The Amazon S3 directory path inside the bucket where the responses to the command execution should be stored.
requestedDateTime	String	The time and date that the request was sent to this specific node.
instanceIds	StringList	<p>The nodes that were targeted by the command.</p> <p><b>Note</b>            Instance IDs are only included in the summary message if the Run Command task targeted instance IDs directly. Instance IDs aren't included in the summary message if the Run Command task was issued using tag-based targeting.</p>
status	String	Command status for the command.

## Invocation-based Amazon SNS notifications

If you send a command to multiple nodes, Amazon SNS can send messages about each copy or invocation of the command. The messages include the following information.

Field	Type	Description
eventTime	String	The time that the event was initiated. The timestamp is important because Amazon SNS doesn't guarantee message delivery order. Example: 2016-04-26T13:15:30Z

Field	Type	Description
documentName	String	The name of the Systems Manager document (SSM document) used to run this command.
requestedDateTime	String	The time and date that the request was sent to this specific node.
commandId	String	The ID generated by Run Command after the command was sent.
instanceId	String	The instance that was targeted by the command.
status	String	Command status for this invocation.

To set up Amazon SNS notifications when a command changes status, complete the following tasks.

**Note**

If you aren't configuring Amazon SNS notifications for your maintenance window, then you can skip Task 5 later in this topic.

**Topics**

- [Task 1: Create and subscribe to an Amazon SNS topic \(p. 1464\)](#)
- [Task 2: Create an IAM policy for Amazon SNS notifications \(p. 1465\)](#)
- [Task 3: Create an IAM role for Amazon SNS notifications \(p. 1465\)](#)
- [Task 4: Configure user access \(p. 1466\)](#)
- [Task 5: Attach the iam:PassRole policy to your maintenance window role \(p. 1467\)](#)

## 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 Simple Queue Service (Amazon SQS). To get started, we recommend that you start with the email protocol. For information about how to create a topic, see [Creating an Amazon SNS 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 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 initiating 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 is shown, choose it, and then choose **Create Policy**.)
3. Choose the **JSON** tab.
4. Replace the default content with the following.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "sns:Publish"
],
 "Resource": "arn:aws:sns:region:account-id:sns-topic-name"
 }
]
}
```

*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*.

*account-id* represents the 12-digit identifier for your AWS account, in the format `123456789012`.

*sns-topic-name* represents the name of the Amazon SNS topic you want to use for publishing notifications.

5. Choose **Next: Tags**.
6. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this policy.
7. Choose **Next: Review**.
8. On the **Review policy** page, for **Name**, enter a name for the inline policy. For example: `my-sns-publish-permissions`.
9. (Optional) For **Description**, enter a description for the policy.
10. 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 initiate Amazon SNS notifications. In all subsequent procedures, this role is referred to as the Amazon SNS IAM role.

### 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**.
5. In the **Select your use case** section, choose **Systems Manager**, and then choose **Next: Permissions**.
6. 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**.
7. Choose **Next: Tags**.
8. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this role.
9. Choose **Next: Review**.
10. On the **Review** page, for **Role name**, enter a name to identify the role, such as **my-sns-role**.
11. (Optional) Change the default role description to reflect the purpose of this role. For example: **Runs SNS topics on your behalf**.
12. Choose **Create role**. The system returns you to the **Roles** page.
13. Choose the name of the role, and then copy or make a note of the **Role ARN** value. This Amazon Resource Name (ARN) for the role is used when you send a command that is configured to return Amazon SNS notifications.
14. Keep the **Summary** page open.

## Task 4: Configure user access

If your IAM user account, group, or role is assigned administrator permissions, then you have access to Run Command and Maintenance Windows, capabilities of AWS Systems Manager. 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 **Choose a service**, and then choose **IAM**.
6. For **Actions**, in the **Filter actions** text box, enter **PassRole**, and then select the check box next to **PassRole**.
7. For **Resources**, verify that **Specific** is selected, and then choose **Add ARN**.
8. In the **Specify ARN for role** field, paste the Amazon SNS IAM role ARN that you copied at the end of Task 3. The system automatically populates the **Account** and **Role name with path** fields.
9. Choose **Add**.
10. Choose **Review policy**.
11. On the **Review Policy** page, enter 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, attach an `iam:PassRole` policy to the maintenance window service role specified. If you don't intend to configure the registered task for Amazon SNS notifications, then you can skip this task.

The `iam:PassRole` policy allows the Maintenance Windows service role to pass the Amazon SNS IAM 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.

### Note

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? \(p. 708\)](#).

If you need to create a custom service role, see one of the following topics:

- [Control access to maintenance windows \(console\) \(p. 709\)](#)
- [Control access to maintenance windows \(AWS CLI\) \(p. 714\)](#)
- [Control access to maintenance windows \(Tools for Windows PowerShell\) \(p. 720\)](#)

### 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 IAM 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 from the **Role name** list.
5. On the **Permissions** tab, verify that either the `AmazonSSMMaintenanceWindowRole` policy is listed or there is a comparable policy that gives maintenance windows permission to the Systems Manager API. If it is not, choose **Attach policies** to attach it.
6. Choose **Add inline policy**.
7. Choose the **Visual editor** tab.
8. For **Service**, choose **IAM**.
9. For **Actions**, in the **Filter actions** text box, enter `PassRole`, and then select the check box next to `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 Run Command or Maintenance Windows, which are capabilities of AWS Systems Manager.

**Note**

This guide doesn't 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 \(p. 1462\)](#).

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**

```
{
 "commandId": "a8c7e76f-15f1-4c33-9052-0123456789ab",
 "documentName": "AWS-RunPowerShellScript",
 "instanceIds": [
 "i-1234567890abcdef0",
 "i-9876543210abcdef0"
],
 "requestedDateTime": "2019-04-25T17:57:09.17Z",
 "expiresAfter": "2019-04-25T19:07:09.17Z",
 "outputS3BucketName": "DOC-EXAMPLE-BUCKET",
 "outputS3KeyPrefix": "runcommand",
 "status": "InProgress",
 "eventTime": "2019-04-25T17:57:09.236Z"
}
```

**Sample JSON Output for Command summary messages using tag-based targeting**

```
{
 "commandId": "9e92c686-ddc7-4827-b040-0123456789ab",
 "documentName": "AWS-RunPowerShellScript",
 "instanceIds": [],
 "requestedDateTime": "2019-04-25T18:01:03.888Z",
 "expiresAfter": "2019-04-25T19:11:03.888Z",
 "outputS3BucketName": "",
 "outputS3KeyPrefix": "",
 "status": "InProgress",
 "eventTime": "2019-04-25T18:01:05.825Z"
}
```

**Sample JSON Output for Invocation messages**

```
{
 "commandId": "ceb96b84-16aa-4540-91e3-925a9a278b8c",
 "documentName": "AWS-RunPowerShellScript",
 "instanceId": "i-1234567890abcdef0",
 "requestedDateTime": "2019-04-25T18:06:05.032Z",
 "status": "InProgress",
 "eventTime": "2019-04-25T18:06:05.099Z"
}
```

## 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 command through Run Command, a capability of AWS Systems Manager, 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)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

#### Note

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Other parameters**:
  - For **Comment**, enter information about this command.
  - For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.
8. For **Rate control**:
  - For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

#### Note

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

10. In the **SNS Notifications** section, choose **Enable SNS notifications**.
11. For **IAM role**, choose the Amazon SNS IAM role ARN you created in Task 3 in [Monitoring Systems Manager status changes using Amazon SNS notifications \(p. 1462\)](#).

12. For **SNS topic**, enter the Amazon SNS topic ARN to be used.
13. For **Event notifications**, choose the events for which you want to receive notifications.
14. For **Change notifications**, choose to receive notifications for the command summary only (**Command status changes**) or for each copy of a command sent to multiple nodes (**Command status on each instance changes**).
15. Choose **Run**.
16. Check your email for a message from Amazon SNS and open the email message. Amazon SNS can take several minutes to send the email message.

## 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 node IDs.

```
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"}'
```

Following is an example.

```
aws ssm send-command --instance-ids "i-02573cafefEXAMPLE, 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"}'
```

Following is an 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 message. Amazon SNS can take several minutes to send the email message.

For more information, see [send-command](#) in the *AWS CLI Command Reference*.

## Use a maintenance window to send a command that returns status notifications

The following procedures show how to register a Run Command task with your maintenance window using the AWS Systems Manager console or the AWS Command Line Interface (AWS CLI). Run Command is a capability of AWS Systems Manager. 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 \(console\) \(p. 724\)](#) for steps on how to create a maintenance window and register targets.

To receive notifications from the Amazon Simple Notification Service (Amazon SNS) service, 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 \(p. 1467\)](#).

## 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)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 Simple Notification Service (Amazon SNS) notifications.
  4. Choose **Actions** and then choose **Register Run command task**.
  5. (Optional) In the **Name** field, enter a name for the task.
  6. (Optional) In the **Description** field, enter a description.
  7. For **Command document**, choose a Command document.
  8. 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. Tasks that have the same priority are scheduled in parallel.
  9. In the **Targets** section, select a registered target group or select unregistered targets.
  10. For **Rate control**:
    - For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

#### Note

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then

restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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**

Add `iam:PassRole` permissions to the Maintenance Windows role to allow 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 \(p. 1462\)](#).

12. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Enter 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 managed node, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, verify that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

13. In the **SNS notifications** section, do the following:

- Choose **Enable SNS Notifications**.
- For **IAM role**, choose the Amazon SNS IAM role Amazon Resource Name (ARN) you created in Task 3 in [Monitoring Systems Manager status changes using Amazon SNS notifications \(p. 1462\)](#) to initiate Amazon SNS.
- For **SNS topic**, enter the Amazon SNS topic ARN to be used.
- For **Event type**, choose the events for which you want to receive notifications.
- For **Notification type**, choose to receive notifications for each copy of a command sent to multiple nodes (invocations) or the command summary.

14. In the **Parameters** section, enter the required parameters based on the Command document you chose.

15. Choose **Register Run command task**.

16. After the next time your maintenance window runs, check your email for a message from Amazon SNS and open the email message. Amazon SNS can take a few minutes to send the email message.

## 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.

```
{
 "Name": "Name",
 "Description": "Description",
 "WindowId": "mw-0c50858d01EXAMPLE",
 "ServiceRoleArn": "arn:aws:iam::account-id:role/MaintenanceWindowIAMRole",
 "MaxConcurrency": "1",
 "MaxErrors": "1",
 "Priority": 3,
 "Targets": [
 {
 "Key": "WindowTargetIds",
 "Values": [
 "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
]
 }
],
 "TaskType": "RUN_COMMAND",
 "TaskArn": "CommandDocumentName",
 "TaskInvocationParameters": {
 "RunCommand": {
 "Comment": "Comment",
 "TimeoutSeconds": 3600,
 "NotificationConfig": {
 "NotificationArn": "arn:aws:sns:region:account-id:SNSTopicName",
 "NotificationEvents": [
 "All"
],
 "NotificationType": "Command"
 },
 "ServiceRoleArn": "arn:aws:iam::account-id:role/SNSIAMRole"
 }
 }
}
```

3. Replace the example values with information about your own resources.

You can also restore options we've 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](#) in the *AWS CLI Command Reference*.

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's required in this command.

If successful, the command returns information similar to the following.

```
{
 "WindowTaskId": "j218d5b5c-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 message. Amazon SNS can take a few minutes to send the email message.

For more information about registering tasks for a maintenance window from the command line, see [Register tasks with the maintenance window \(p. 739\)](#).

# Product and service integrations with Systems Manager

By default, AWS Systems Manager integrates with AWS services and other products and services. The following information can help you configure Systems Manager to integrate with the products and services you use.

- [Integration with AWS services \(p. 1475\)](#)
- [Integration with other products and services \(p. 1492\)](#)
- [Integration examples from the community \(p. 1502\)](#)

## Integration with AWS services

Through the use of Systems Manager Command documents (SSM documents) and Automation runbooks, you can use AWS Systems Manager to integrate with AWS services. For more information about these resources, see [AWS Systems Manager documents \(p. 1287\)](#).

Systems Manager is integrated with the following AWS services.

### Compute

#### Amazon Elastic Compute Cloud (Amazon EC2)

[Amazon EC2](#) provides scalable computing capacity in the AWS Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.

Systems Manager allows you to perform several tasks on EC2 instances. For example you can launch, configure, manage, maintain, troubleshoot, and securely connect to your EC2 instances. You can also use Systems Manager to deploy software, determine compliance status, and gather inventory from your EC2 instances.

#### Learn more

- [Managed nodes \(p. 804\)](#)
- [AWS Systems Manager State Manager \(p. 1034\)](#)
- [AWS Systems Manager Run Command \(p. 991\)](#)
- [AWS Systems Manager Patch Manager \(p. 1093\)](#)
- [AWS Systems Manager Session Manager \(p. 912\)](#)
- [AWS Systems Manager Distributor \(p. 1252\)](#)
- [AWS Systems Manager Compliance \(p. 836\)](#)

	<ul style="list-style-type: none"> <li><a href="#">AWS Systems Manager Inventory (p. 848)</a></li> </ul>
Amazon EC2 Auto Scaling	<p><a href="#">Auto Scaling</a> helps you ensure that you have the correct number of EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups.</p> <p>Systems Manager allows you to automate common procedures like patching the Amazon Machine Image (AMI) used in your Auto Scaling template for your Auto Scaling group.</p> <p><b>Learn more</b></p> <p><a href="#">Walkthrough: Patch an AMI and update an Auto Scaling group (p. 671)</a></p>
Amazon Elastic Container Service (Amazon ECS)	<p><a href="#">Amazon ECS</a> is a highly scalable, fast, container management service that allows you to run, stop, and manage Docker containers on a cluster.</p> <p>Systems Manager allows you to manage container instances remotely and inject sensitive data into your containers by storing your sensitive data in parameters in Parameter Store, a capability of Systems Manager, and then referencing them in your container definition.</p> <p><b>Learn more</b></p> <ul style="list-style-type: none"> <li><a href="#">Manage container instances remotely using AWS Systems Manager</a></li> <li><a href="#">Specifying sensitive data using Systems Manager Parameter Store</a></li> </ul>
AWS Lambda	<p><a href="#">Lambda</a> is a compute service that allows you to run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.</p> <p>Systems Manager allows you to use Lambda functions within Automation runbook content by using the <code>aws:invokeLambdaFunction</code> action.</p> <p><b>Learn more</b></p> <p><a href="#">Walkthrough: Simplify AMI patching using Automation, AWS Lambda, and Parameter Store (p. 665)</a></p>

## Internet of Things (IoT)

AWS IoT Greengrass core devices	<a href="#">AWS IoT Greengrass</a> is an open source IoT edge runtime and cloud service that helps you build,
---------------------------------	---------------------------------------------------------------------------------------------------------------

	<p>deploy and manage IoT applications on your devices. Systems Manager offers native support for AWS IoT Greengrass core devices.</p> <p><b>Learn more</b></p> <p><a href="#">Setting up AWS Systems Manager for edge devices (p. 52)</a></p>
AWS IoT core devices	<p><b>AWS IoT</b> provides the cloud services that connect your IoT devices to other devices and AWS cloud services. AWS IoT provides device software that can help you integrate your IoT devices into AWS IoT-based solutions. If your devices can connect to AWS IoT, AWS IoT can connect them to the cloud services that AWS provides. Systems Manager supports AWS IoT core devices as long as those devices are configured as <i>managed nodes</i> in a hybrid environment.</p> <p><b>Learn more</b></p> <p><a href="#">Setting up AWS Systems Manager for hybrid environments (p. 34)</a></p>

## Storage

Amazon Simple Storage Service (Amazon S3)	<p><b>Amazon S3</b> is storage for the Internet. It's designed to make web-scale computing easier for developers. Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web.</p> <p>Systems Manager allows you to run remote scripts and SSM documents that are stored in Amazon S3. Distributor, a capability of AWS Systems Manager, uses Amazon S3 to store packages. You can also send output to Amazon S3 for Run Command and Session Manager, capabilities of AWS Systems Manager.</p> <p><b>Learn more</b></p> <ul style="list-style-type: none"><li>• <a href="#">Running scripts from Amazon S3 (p. 1485)</a></li><li>• <a href="#">Running Systems Manager Command documents from remote locations (p. 1373)</a></li><li>• <a href="#">AWS Systems Manager Distributor (p. 1252)</a></li><li>• <a href="#">Logging session data using Amazon S3 (console) (p. 979)</a></li></ul>
-------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Developer Tools

AWS CodeBuild	<p><a href="#">CodeBuild</a> is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy. CodeBuild eliminates the need to provision, manage, and scale your own build servers.</p> <p>Parameter Store allows you to store sensitive information for your build specifications and projects.</p> <p><b>Learn more</b></p> <ul style="list-style-type: none"><li>• <a href="#">Build specification reference for CodeBuild</a></li><li>• <a href="#">Create a build project in AWS CodeBuild</a></li></ul>
---------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Security, Identity, and Compliance

AWS Identity and Access Management (IAM)	<p><a href="#">IAM</a> is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.</p> <p>Systems Manager allows you to control access to services using IAM.</p> <p><b>Learn more</b></p> <ul style="list-style-type: none"><li>• <a href="#">How AWS Systems Manager works with IAM (p. 1384)</a></li><li>• <a href="#">Actions, resources, and condition keys for AWS Systems Manager</a></li><li>• <a href="#">Create non-Admin IAM users and groups for Systems Manager (p. 18)</a></li><li>• <a href="#">Create an IAM instance profile for Systems Manager (p. 21)</a></li></ul>
AWS Secrets Manager	<p><a href="#">Secrets Manager</a> provides easier management of secrets. Secrets can be database credentials, passwords, third-party API keys, and even arbitrary text.</p> <p>Parameter Store allows you to retrieve Secrets Manager secrets when using other AWS services that already support references to Parameter Store parameters.</p> <p><b>Learn more</b></p> <p><a href="#">Referencing AWS Secrets Manager secrets from Parameter Store parameters (p. 1488)</a></p>

## AWS Security Hub

[Security Hub](#) gives you a comprehensive view of your high-priority security alerts and compliance status across AWS accounts. Security Hub aggregates, organizes, and prioritizes your security alerts, or findings, from multiple AWS services.

When you turn on integration between Security Hub and Patch Manager, a capability of AWS Systems Manager, Security Hub monitors the patching status of your fleets from a security standpoint. Patch compliance details are automatically exported to Security Hub. This allows you to use a single view to centrally monitor your patch compliance status and to track other security findings. You can receive alerts when nodes in your fleet go out of patch compliance and review patch compliance findings in the Security Hub console.

You can also integrate Security Hub with Explorer and OpsCenter, capabilities of AWS Systems Manager. Integration with Security Hub allows you to receive findings from Security Hub in Explorer and OpsCenter. Security Hub findings provide security information that you can use in Explorer and OpsCenter to aggregate and take action on your security, performance, and operational issues in AWS Systems Manager.

There is a charge to use Security Hub. For more information, see [Security Hub pricing](#).

### Learn more

- [Receiving findings from AWS Security Hub in Explorer \(p. 175\)](#)
- [Receiving findings from AWS Security Hub in OpsCenter \(p. 227\)](#)
- [Integrating Patch Manager with AWS Security Hub \(p. 1207\)](#)

## Cryptography and PKI

### AWS Key Management Service (AWS KMS)

[AWS KMS](#) is a managed service that allows you to create and control customer managed keys, the encryption keys used to encrypt your data.

Systems Manager allows you to use AWS KMS to create SecureString parameters and encrypt Session Manager session data.

**Learn more**

- [How AWS Systems Manager Parameter Store uses AWS KMS](#)
- [Turn on KMS key encryption of session data \(console\) \(p. 945\)](#)

## Management and Governance

AWS CloudFormation	<p><a href="#">AWS CloudFormation</a> is a service that helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS.</p> <p>Parameter Store is a source for dynamic references. Dynamic references provide a compact, powerful way for you to specify external values that are stored and managed in other services in your AWS CloudFormation stack templates.</p> <p><b>Learn more</b></p> <p><a href="#">Using dynamic references to specify template values</a></p>
AWS CloudTrail	<p><a href="#">CloudTrail</a> is an AWS service that helps you authorize governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail. Events include actions taken in the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDKs and APIs.</p> <p>Systems Manager integrates with CloudTrail, allowing you to capture all API calls for Systems Manager as events, including calls from the Systems Manager console and from code calls to the Systems Manager APIs.</p> <p><b>Learn more</b></p> <p><a href="#">Logging AWS Systems Manager API calls with AWS CloudTrail (p. 1442)</a></p>
Amazon CloudWatch Logs	<p><a href="#">Amazon CloudWatch Logs</a> allows you to centralize the logs from all of your systems, applications, and AWS services that you use. You can then view them, search them for specific error codes or patterns, filter them based on specific fields, or archive them securely for future analysis.</p>

	<p>Systems Manager supports sending logs for the SSM Agent, Run Command, and Session Manager to CloudWatch Logs.</p> <p><b>Learn more</b></p> <ul style="list-style-type: none"><li>• <a href="#">Sending node logs to CloudWatch Logs (CloudWatch agent) (p. 1429)</a></li><li>• <a href="#">Configuring Amazon CloudWatch Logs for Run Command (p. 1447)</a></li><li>• <a href="#">Logging session data using Amazon CloudWatch Logs (console) (p. 981)</a></li></ul>
Amazon EventBridge	<p><a href="#">EventBridge</a> delivers a near real-time stream of system events that describes changes in Amazon Web Services 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. EventBridge becomes aware of operational changes as they occur. EventBridge responds to these operational changes and takes corrective action as necessary. These actions include sending messages to respond to the environment, activating functions, and capturing state information.</p> <p>Systems Manager has multiple events that are supported by EventBridge allowing you to take actions based on the content of those events.</p> <p><b>Learn more</b></p> <p><a href="#">Monitoring Systems Manager events with Amazon EventBridge (p. 1449)</a></p> <p><b>Note</b> Amazon EventBridge is the preferred way to manage your events. CloudWatch Events and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge are reflected in each console. For more information, see the <a href="#">Amazon EventBridge User Guide</a>.</p>

## 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. This allows you to see how the configurations and relationships change over time.

Systems Manager is integrated with AWS Config, providing multiple rules that help you gain visibility into your EC2 instances. These rules help you identify which EC2 instances are managed by Systems Manager, operating system configurations, system-level updates, installed applications, network configurations, and more.

### **Learn more**

- [AWS Config supported resource types](#)
- [Recording software configuration for managed instances](#)
- [Viewing inventory history and change tracking \(p. 895\)](#)

## AWS Trusted Advisor

[Trusted Advisor](#) is an online tool that provides real-time guidance to help you provision your resources following AWS best practices.

Systems Manager hosts Trusted Advisor and you can view Trusted Advisor data in Explorer.

### **Learn more**

- [AWS Systems Manager Explorer \(p. 157\)](#)
- [Getting started with AWS Trusted Advisor](#)

## AWS Organizations

[Organizations](#) is an account management service that allows you to consolidate multiple AWS accounts into an organization that you create and centrally manage. Organizations includes account management and consolidated billing capabilities that allow you to better meet the budgetary, security, and compliance needs of your business.

Integration between [Change Manager \(p. 352\)](#), a capability of AWS Systems Manager, with Organizations makes it possible to use a delegated administrator account to manage change requests, change templates, and approvals for your entire organization through this single account.

Organizations integration with [Inventory \(p. 848\)](#), a capability of AWS Systems Manager, and [Explorer \(p. 157\)](#) allows you to aggregate inventory and operations data (OpsData) from multiple AWS Regions and AWS accounts.

Integration between Quick Setup , a capability of AWS Systems Manager, and Organizations automates common service setup tasks, and deploys service configurations based on best practices across your organizational units (OUs).

## Networking and Content Delivery

### AWS PrivateLink

[AWS PrivateLink](#) allows you to privately connect your virtual private cloud (VPC) to supported AWS services and VPC endpoint services without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection.

Systems Manager supports managed nodes connecting to Systems Manager APIs using AWS PrivateLink. This improves the security posture of your managed nodes because AWS PrivateLink restricts all network traffic between your managed nodes, Systems Manager, and Amazon EC2 to the Amazon network. This means that managed nodes aren't required to have access to the internet.

[Learn more](#)

[\(Optional\) Create a VPC endpoint \(p. 28\)](#)

## Analytics

Amazon Athena	<p><a href="#">Athena</a> is an interactive query service that allows you to analyze data directly in Amazon Simple Storage Service (Amazon S3) using standard SQL. With a few actions in the AWS Management Console, you can point Athena at your data stored in Amazon S3 and begin using standard SQL to run one-time queries and get results in seconds.</p> <p>Systems Manager Inventory integrates with Athena to help you query inventory data from multiple AWS Regions and AWS accounts. Athena integration uses resource data sync so that you can view inventory data from all managed nodes on the <a href="#">Detailed View</a> page in the Systems Manager Inventory console.</p> <p><b>Learn more</b></p> <ul style="list-style-type: none"><li>• <a href="#">Querying inventory data from multiple Regions and accounts (p. 870)</a></li><li>• <a href="#">Walkthrough: Use resource data sync to aggregate inventory data (p. 903)</a></li></ul>
AWS Glue	<p><a href="#">AWS Glue</a> is a fully managed ETL (extract, transform, and load) service that makes it simple and cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores and data streams.</p> <p>Systems Manager uses AWS Glue to crawl the Inventory data in your S3 bucket.</p> <p><b>Learn more</b></p> <p><a href="#">Querying inventory data from multiple Regions and accounts (p. 870)</a></p>
Amazon QuickSight	<p><a href="#">Amazon QuickSight</a> is a business analytics service you can use to build visualizations, perform one-time analysis, and get business insights from your data. It can automatically discover AWS data sources and also works with your data sources.</p> <p>Systems Manager resource data sync sends inventory data collected from all of your managed nodes to a single Amazon S3 bucket. You can use Amazon QuickSight to query and analyze the aggregated data.</p> <p><b>Learn more</b></p> <ul style="list-style-type: none"><li>• <a href="#">Configuring resource data sync for Inventory (p. 858)</a></li></ul>

- [Walkthrough: Use resource data sync to aggregate inventory data \(p. 903\)](#)

## Application Integration

Amazon Simple Notification Service (Amazon SNS)

[Amazon SNS](#) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.

Systems Manager generates statuses for multiple services that can be captured by Amazon SNS notifications.

### Learn more

- [Monitoring Systems Manager status changes using Amazon SNS notifications \(p. 1462\)](#)
- [Setting up notifications or trigger actions based on Parameter Store events \(p. 275\)](#)

## AWS Management Console

AWS Resource Groups

[Resource Groups](#) organize your AWS resources. Resource groups make it easier to manage, monitor, and automate tasks on large numbers of resources at one time.

Systems Manager resource types like managed nodes, SSM documents, maintenance windows, Parameter Store parameters, and patch baselines can be added to resource groups.

### Learn more

[What are AWS Resource Groups?](#)

### Topics

- [Running scripts from Amazon S3 \(p. 1485\)](#)
- [Referencing AWS Secrets Manager secrets from Parameter Store parameters \(p. 1488\)](#)

## Running scripts from Amazon S3

This section describes how to download and run scripts from Amazon Simple Storage Service (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, AWS Systems Manager also runs any referenced scripts that are included in the directory.

Note the following important details about running scripts from Amazon S3:

- Systems Manager doesn't verify that your script is capable of running on a node. Before you download and run the script, verify that the required software is installed on the node. Or, you can create a composite document that installs the software by using either Run Command or State Manager, capabilities of AWS Systems 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 Amazon Elastic Compute Cloud (Amazon 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*.

## Run shell scripts from Amazon S3

This following information includes procedures to help you run scripts from Amazon Simple Storage Service (Amazon S3) by using either the AWS Systems Manager console or the AWS Command Line Interface (AWS CLI). Though shell scripts are used in the examples, other types of scripts can be substituted.

### Run a shell script from Amazon S3 (console)

#### Run a shell script from Amazon S3

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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-RunRemoteScript**.
  5. In **Command parameters**, do the following:
    - In **Source Type**, select **S3**.
    - In the **Source Info** text box, enter the required information to access the source in the following format. Replace each `example resource placeholder` with your own information.

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

The following is an example.

```
{"path": "https://s3.amazonaws.com/doc-example-bucket/scripts/shell/helloWorld.sh"}
```

- In the **Command Line** field, enter parameters for the script execution. Here is an example.

```
helloWorld.sh argument-1 argument-2
```

- (Optional) In the **Working Directory** field, enter the name of a directory on the node 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, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Other parameters**:

- For **Comment**, enter information about this command.
- For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

11. Choose **Run**.

## Run a shell script from Amazon S3 (command line)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command. Replace each *example resource placeholder* with your own information.

Linux & macOS

```
aws ssm send-command \
--document-name "AWS-RunRemoteScript" \
--targets "Key=InstanceIds,Values=instance ID" \
--parameters '[{"sourceType":["S3"],"sourceInfo":[{"path":"https://s3.aws-api-domain/script path"}],"commandLine":["script name and arguments"]}'
```

#### Windows

```
aws ssm send-command ^
--document-name "AWS-RunRemoteScript" ^
--targets "Key=InstanceIds,Values=instance ID" ^
--parameters "sourceType="S3",sourceInfo='{\"path\": \"https://s3.amazonaws.com/script path\"}', "commandLine"="script name and arguments"
```

#### PowerShell

```
Send-SSMCommand `-
-DocumentName "AWS-RunRemoteScript" `-
-Targets "Key=InstanceIds,Values=instance ID" `-
-Parameter @{ sourceType="S3";sourceInfo='{"path": "https://s3.amazonaws.com/script path"}'; "commandLine"="script name and arguments"}
```

## Referencing AWS Secrets Manager secrets from Parameter Store parameters

AWS Secrets Manager helps you organize and manage important configuration data such as credentials, passwords, and license keys. Parameter Store, a capability of AWS Systems Manager, is 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 Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS), 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 more information about Secrets Manager, see [What Is AWS Secrets Manager?](#) in the *AWS Secrets Manager User Guide*.

## 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 operations. Modification operations and advance querying API operations, such as [DescribeParameters](#) and [GetParametersByPath](#), aren't supported for Secrets Manager.
- You can use the AWS Command Line Interface (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 name must begin with the following reserved path: /aws/reference/secretsmanager/[secret\\_ID\\_in\\_Secrets\\_Manager](#).

Here is an example: /aws/reference/secretsmanager/CFCreds1

- Parameter Store honors AWS Identity and Access Management (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 uses 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 permissions 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*.

### 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.

### To reference a Secrets Manager secret by using Parameter Store

1. Create a secret in Secrets Manager. For more information, see [Create and manage 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 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 names that correctly reference the Secrets Manager secrets, CFCreds1 and DBPass, using Parameter Store.
  - /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 accessed from Parameter Store by using the GetParameter API.

```
/**
 * Initialize 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();
return client;
}

/**
 * Helper method to retrieve SSM Parameter's value
 * @param parameterName identifier of the SSM Parameter
 * @return decrypted parameter value
 */
public GetParameterResult getParameter(String parameterName) {
 GetParameterRequest request = new GetParameterRequest();
 request.setName(parameterName);
 request.setWithDecryption(true);
 return ssm.newGetParameterCall().call(request).getParameter().getValue();
}
```

Here are some AWS CLI examples. Use the `aws secretsmanager list-secrets` command to find the names of your secrets.

#### AWS CLI Example 1: Reference by using the name of the secret

Linux & macOS

```
aws ssm get-parameter \
--name /aws/reference/secretsmanager/s1-secret \
--with-decryption
```

Windows

```
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",
 "Type": "SecureString",
 "Value": "Fl*MEishm!a1875",
 "Version": 0,
 "SourceResult": {
 "CreatedDate": 1526334434.743,
 "Name": "s1-secret",
 "VersionId": "aaabbccc-1111-222-333-123456789",
 "SecretString": "Fl*MEishm!a1875",
 "VersionStages": ["AWSCURRENT"],
 "ARN": "arn:aws:secretsmanager:us-east-2:123456789012:secret:s1-
secret-E18LRP"
 }
 },
 "LastModifiedDate": 2018-05-14T21:47:14.743Z,
 "ARN": "arn:aws:secretsmanager:us-east-2:123456789012:secret:s1-secret-
E18LRP",
}
}
```

#### AWS CLI Example 2: Reference that includes the version ID

Linux & macOS

```
aws ssm get-parameter \
--name /aws/reference/secretsmanager/s1-secret:11111-aaa-bbb-ccc-123456789 \
--with-decryption
```

Windows

```
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",
 "Type": "SecureString",
 "Value": "Fl*MEishm!a1875",
 "Version": 0,
 "SourceResult": {
 {
 \"CreatedDate\": 1526334434.743,
 \"Name\": \"s1-secret\",
 \"VersionId\": \"11111-aaa-bbb-ccc-123456789\",
 \"SecretString\": \"Fl*MEishm!a1875\",
 \"VersionStages\": [\"AWSCURRENT\"],
 \"ARN\": \"arn:aws:secretsmanager:us-east-2:123456789012:secret:s1-
secret-E18LRP\""
 }
 },
 "Selector": ":11111-aaa-bbb-ccc-123456789"
 },
 "LastModifiedDate": 2018-05-14T21:47:14.743Z,
 "ARN": "arn:aws:secretsmanager:us-east-2:123456789012:secret:s1-secret-
E18LRP",
}
```

**AWS CLI Example 3: Reference that includes the version stage**

Linux & macOS

```
aws ssm get-parameter \
--name /aws/reference/secretsmanager/s1-secret:AWSCURRENT \
--with-decryption
```

Windows

```
aws ssm get-parameter ^
--name /aws/reference/secretsmanager/s1-secret:AWSCURRENT ^
--with-decryption
```

The command returns information like the following.

```
{
 "Parameter": {
 "Name": "/aws/reference/secretsmanager/s1-secret",
```

```
"Type": "SecureString",
"Value": "Fl*MEishm!a1875",
"Version": 0,
"SourceResult":
 "{\n \"CreatedDate\": 1526334434.743,\n \"Name\": \"s1-secret\",\n \"VersionId\": \"11111-aaa-bbb-ccc-123456789\", \n \"SecretString\": \"Fl*MEishm!a1875\", \n \"VersionStages\": [\"AWSCURRENT\"],\n \"ARN\": \"arn:aws:secretsmanager:us-east-2:123456789012:secret:s1-\nsecret-E18LRP\"\n }"
 "Selector": ":AWSCURRENT"
}
"LastModifiedDate": 2018-05-14T21:47:14.743Z,
"ARN": "arn:aws:secretsmanager:us-east-2:123456789012:secret:s1-secret-
E18LRP",
}
```

## Integration with other products and services

AWS Systems Manager has built-in integration for the products and services shown in the following table.

Ansible	<p><a href="#">Ansible</a> is an IT automation platform that makes your applications and systems easier to deploy.</p> <p>Systems Manager provides the Systems Manager document (SSM document) <a href="#">AWS-ApplyAnsiblePlaybooks</a> which allows you to create State Manager associations that run Ansible playbooks.</p> <p><a href="#">Learn more</a></p> <p><a href="#">Walkthrough: Creating associations that run Ansible playbooks (p. 1076)</a></p>
Chef	<p><a href="#">Chef</a> is an IT automation tool that makes your applications and systems easier to deploy.</p> <p>Systems Manager provides the <a href="#">AWS-ApplyChefRecipes</a> SSM document, which allows you to create associations in State Manager, a capability of AWS Systems Manager, that run Chef recipes.</p> <p><a href="#">Learn more</a></p> <p><a href="#">Walkthrough: Creating associations that run Chef recipes (p. 1081)</a></p> <p>Systems Manager also integrates with <a href="#">Chef InSpec</a> profiles, allowing you to run compliance scans and view compliant and noncompliant nodes.</p> <p><a href="#">Learn more</a></p>

	<p><a href="#">Using Chef InSpec profiles with Systems Manager Compliance (p. 1498)</a></p>
GitHub	<p><a href="#">GitHub</a> provides hosting for software development version control and collaboration.</p> <p>Systems Manager provides the SSM document <code>AWS-RunDocument</code>, which allows you to run other SSM documents stored in GitHub, and the SSM document <code>AWS-RunRemoteScript</code>, which allows you to run scripts stored in GitHub.</p> <p><b>Learn more</b></p> <ul style="list-style-type: none"><li>• <a href="#">Running Systems Manager Command documents from remote locations (p. 1373)</a></li><li>• <a href="#">Running scripts from GitHub (p. 1493)</a></li></ul>
Jenkins	<p><a href="#">Jenkins</a> is an open-source automation server that allows developers to reliably build, test, and deploy their software.</p> <p>Automation, a capability of Systems Manager, can be used as a post-build step to pre-install application releases into Amazon Machine Images (AMIs).</p> <p><b>Learn more</b></p> <p><a href="#">Walkthrough: Using Automation with Jenkins (p. 687)</a></p>

## Topics

- [Running scripts from GitHub \(p. 1493\)](#)

## Running scripts from GitHub

This topic describes how to use the pre-defined Systems Manager document (SSM document) `AWS-RunRemoteScript` to download scripts from GitHub, including Ansible Playbooks, Python, Ruby, and PowerShell scripts. By using this SSM document, you no longer need to manually port scripts into Amazon Elastic Compute Cloud (Amazon EC2) or wrap them in SSM documents. AWS Systems Manager integration with GitHub promotes *infrastructure as code*, which reduces the time it takes to manage nodes while standardizing configurations across your fleet.

You can also create custom SSM documents that allow you to download and run scripts or other SSM documents from remote locations. For more information, see [Creating composite documents \(p. 1358\)](#).

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 that are included in the directory.

Note the following important details about running scripts from GitHub.

- Systems Manager doesn't verify that your script is capable of running on a node. Before you download and run the script, verify that the required software is installed on the node. Or, you can create a composite document that installs the software by using either Run Command or State Manager, capabilities of AWS Systems Manager, and then downloads and runs the script.

- You're responsible for ensuring that all GitHub requirements are met. This includes refreshing your access token, as needed. 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 \(p. 1494\)](#)
- [Run Python scripts from GitHub \(p. 1496\)](#)

## 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 Command Line Interface (AWS CLI).

### Before you begin

If you plan to run a script stored in a private GitHub repository, create an AWS 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 \(p. 279\)](#).

### Run an Ansible Playbook from GitHub (console)

#### Run an Ansible Playbook from GitHub

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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-RunRemoteScript**.
  5. In **Command parameters**, do the following:
    - In **Source Type**, select **GitHub**.
    - In the **Source Info** box, enter the required information to access the source in the following format.

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

This example downloads a file named `webserver.yml`.

```
{
 "owner": "TestUser1",
 "repository": "GitHubPrivateTest",
 "getOptions": "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:

```
"getOptions": "commitID:b8c1dddb94...b76d3bEXAMPLE",
```

- In the **Command Line** field, enter parameters for the script execution. Here is an example.

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

- (Optional) In the **Working Directory** field, enter the name of a directory on the node 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, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

**Note**

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Other parameters**:

- For **Comment**, enter information about this command.
- For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

11. Choose **Run**.

## Run an Ansible Playbook from GitHub by using the AWS CLI

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to download and run a script from GitHub.

```
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 command to run on a local Linux machine.

```
aws ssm send-command \
--document-name "AWS-RunRemoteScript" \
--instance-ids "i-02573cafefEXAMPLE" \
--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 AWS Systems Manager console or the AWS Command Line Interface (AWS CLI).

### Run a Python script from GitHub (console)

#### Run a Python script from GitHub

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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-RunRemoteScript**.
5. For **Command parameters**, do the following:
  - In **Source Type**, select **GitHub**.
  - In the **Source Info** box, enter the required information to access the source in the following format:

```
{
 "owner": "owner_name",
 "repository": "repository_name",
 "getOptions": "branch:branch_name",
```

```
 "path": "path_to_document",
 "tokenInfo": "{{ssm-secure:SecureString_parameter_name }
```

The following example downloads a directory of scripts named *complex-script*.

```
{
 "owner": "TestUser1",
 "repository": "SSMTestDocsRepo",
 "getOptions": "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:

```
"getOptions": "commitID:bbc1ddb94...b76d3bEXAMPLE",
```

- For **Command Line**, enter parameters for the script execution. Here is an example.

```
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**, enter the name of a directory on the node 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, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

#### Note

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

7. For **Other parameters**:

- For **Comment**, enter information about this command.
- For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

#### Note

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node 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 \(p. 1462\)](#).

11. Choose **Run**.

## Run a Python script from GitHub by using the AWS CLI

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).

2. Run the following command to download and run a script from GitHub.

```
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.

```
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids "i-02573cafefEXAMPLE" --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 called `complex-script`. The `commandLine` entry runs `mainFile.py`, which can then run other scripts in the `complex-script` directory.

## Using Chef InSpec profiles with Systems Manager Compliance

AWS Systems Manager integrates with [Chef InSpec](#). InSpec is an open-source testing framework that allows you to create human-readable profiles to store in GitHub or Amazon Simple Storage Service (Amazon S3). Then you can use Systems Manager to run compliance scans and view compliant and noncompliant nodes. 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 nodes with Compliance, a capability of AWS Systems Manager:

- 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 Amazon Elastic Compute Cloud (Amazon EC2) instances and on-premises servers or virtual machines (VMs) that you manage with Systems Manager. The following sample Chef InSpec profile checks if port 22 is open.

```
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 nodes. You can author and store profiles in GitHub or Amazon S3.

## How it works

Here is how the process of using InSpec profiles with 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 Systems Manager document (SSM document) [AWS-RunInspecChecks](#). You can begin a Compliance scan by using Run Command, a capability of AWS Systems Manager, for on-demand scans, or you can schedule regular Compliance scans by using State Manager, a capability of AWS Systems Manager.
4. Identify noncompliant nodes by using the Compliance API or the Compliance console.

### Note

Note the following information.

- Chef uses a client on your nodes to process the profile. You don't need to install the client. When Systems Manager runs the SSM document [AWS-RunInspecChecks](#), 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 the SSM document [AWS-RunInspecChecks](#), as described in this topic, assigns a compliance entry of type `Custom:Inspec` to each targeted node. To assign this compliance type, the document calls the [PutComplianceItems](#) API operation.

## 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 Command Line Interface (AWS CLI). The console procedure shows how to configure State Manager to run the scan. The AWS CLI procedure shows how to configure Run Command to run the scan.

## Running an InSpec compliance scan with State Manager (console)

### To run an InSpec compliance scan with State Manager by using the AWS Systems Manager console

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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>.

```
{"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 \(p. 1493\)](#).

8. In the **Targets** section, choose the managed nodes on which you want to run this operation by specifying tags, selecting instances or edge devices manually, or specifying a resource group.

#### Note

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

9. In the **Specify schedule** section, use the schedule builder options to create a schedule that specifies when you want the Compliance scan to run.

10. For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of managed nodes on which to run the command at the same time.

#### Note

If you selected targets by specifying tags applied to managed nodes or by specifying AWS resource groups, and you aren't certain how many managed nodes are targeted, then restrict the number of targets 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 managed nodes after it fails on either a number or a percentage of nodes. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Managed nodes 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. Enter 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 (for EC2 instances) or IAM service role (on-premises machines) 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 \(p. 21\)](#) or [Create an IAM service role for a hybrid environment \(p. 36\)](#). In addition, if the specified S3 bucket is in a different AWS account, make sure that the instance profile or IAM service role associated with the managed node has the necessary permissions to write to that bucket.

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 nodes where the **Compliance Type** column is **Custom:Inspec**.
15. Choose a node ID to view the details of noncompliant statuses.

## Running an InSpec compliance scan with Run Command (AWS CLI)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.  
For information, see [Install or upgrade AWS command line tools \(p. 60\)](#).
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/DOC-EXAMPLE-
BUCKET/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/DOC-EXAMPLE-BUCKET/InSpec/PortCheck.yml"}]}'
```

3. Run the following command to view a summary of the Compliance scan.

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

4. Run the following command to see details of a node that isn't compliant.

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

## Integration examples from the community

The following sections provide links to blog posts, articles, and community-provided examples related to AWS Systems Manager integration with AWS and other products and services.

### Note

These links are provided for informational purposes only, and shouldn't be considered either a comprehensive list or an endorsement of the content of the examples. AWS isn't responsible for the content or accuracy of this content.

## Blog posts

### Application Management

- [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, a capability of Systems Manager.

*Published May 2020*

- [Keep Your Secrets Safe with AWS Systems Manager Parameter Store and Node](#)

Amir Boroumand walks through how to save and retrieve a password using the Parameter Store with Node.

*Published October 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.

*Published August 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.

*Published March 2019*

- [Using AWS Systems ManagerParameter Store SecureString parameters in AWS CloudFormation templates](#)

Learn how to use plaintext and `SecureString` parameters in your AWS CloudFormation templates through the use of dynamic references to fetch parameter values.

*Published October 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.

*Published July 2018*

- [Integrating AWS CloudFormation with AWS Systems ManagerParameter 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.

*Published December 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.

*Published August 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 Amazon Elastic Compute Cloud (Amazon 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.

*Published June 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.

*Published March 2017*

- [Secrets in AWS](#)

Stephen Price describes how you can use Parameter Store to manage and use secrets in your favorite programming language to handle secrets for cloud-based architectures, such as microservices or containerized applications.

*Published March 2017*

- [Using Parameter Store with AWS CodePipeline](#)

Trey McElhattan demonstrates how to effectively use Parameter Store as part of a continuous delivery pipeline using AWS CodePipeline.

*Published March 2017*

## Actions and Change

- [Providing temporary instance permissions with AWS Systems Manager Automations](#)

Learn how to provide temporary permissions to Amazon EC2 instances within your Automation runbooks. This helps you to avoid modifying instance profiles that are attached to instances long term and only contain the core permissions required for Systems Manager functionality.

*Published December 2019*

- [Automating the Cloud: AWS Security Done Efficiently](#)

Josh Frantz, Lead Security Consultant at Rapid7, shows how to automate common tasks so you can more efficiently secure your AWS environment and focus on solving important, engaging, and difficult issues.

*Published August 2019*

- [Managing AWS resources across multiple accounts and Regions using AWS Systems Manager Automation](#)

Learn how to manage your resources across multiple AWS accounts and AWS Regions using Systems Manager Automation.

*Published January 2019*

- [Onica Demonstrates Uses for New AWS Systems Manager Automation Actions](#)

Onica uses real world examples representing some problems they see in the field while assisting AWS customers to show how Automation actions can be used to solve these problems.

*Published August 2018*

## Instances and Nodes

- [How to patch Amazon EC2 Windows instances in private subnets using AWS Systems Manager](#)

Learn how to patch Amazon EC2 Windows instances in private subnets without internet connectivity.

*Published December 2018*

- [Centralized multi-account and multi-Region patching with AWS Systems Manager Automation](#)

Learn how to use Systems Manager Automation to patch your managed nodes across multiple AWS accounts and AWS Regions.

*Published November 2018*

- [Scalable cross-platform patching with AWS Systems Manager](#)

Learn how to patch Amazon EC2 instances at scale with Systems Manager.

*Published April 2018*

## Shared Resources

- [Writing your own AWS Systems Manager documents](#)

Learn how to author your own Systems Manager documents.

*Published May 2018*

# Tagging 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 allow 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 nodes that help you track or target nodes 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 nodes in the group or reviewing the status of those nodes.

You can use the tags applied to your AWS Systems Manager resources in various operations. For example, you can target only managed nodes that are tagged with a specified tag key-value pair when you [run a command \(p. 995\)](#) or [assign targets to a maintenance window \(p. 726\)](#). You can also [restrict access to your resources \(p. 1391\)](#) 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 Resource Groups 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 [What are AWS Resource Groups?](#) in the *AWS Resource Groups User Guide*.

The remainder of this chapter describes how to add and remove tags from Systems Manager resources.

## Topics

- [Taggable Systems Manager resources \(p. 1505\)](#)
- [Tagging Systems Manager documents \(p. 1506\)](#)
- [Tagging maintenance windows \(p. 1510\)](#)
- [Tagging managed nodes \(p. 1515\)](#)
- [Tagging OpsItems \(p. 1519\)](#)
- [Tagging automations \(p. 1522\)](#)
- [Tagging Systems Manager parameters \(p. 1525\)](#)
- [Tagging patch baselines \(p. 1529\)](#)

## Taggable Systems Manager resources

You can apply tags to the following AWS Systems Manager resource types:

- Automations
- Documents
- Maintenance windows
- Managed nodes
- OpsItems
- OpsMetadata

- Parameters
- Patch baselines

You can add each of these types, except OpsItems and OpsMetadata, 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 nodes and then run a maintenance window task that targets only nodes with that key-value pair.

You can also restrict user access to these resource types by creating AWS Identity and Access Management (IAM) policies that specify the tags that a user can access and attaching the policy to user accounts or groups. The following are several examples of restricting resource access using tags.

- You can apply a tag to a set of custom Systems Manager documents (SSM 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're 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 makes it possible, for example, to 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 Command Line Interface (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 (SSM documents).

### Topics

- [Creating documents with tags \(p. 1506\)](#)
- [Adding tags to existing documents \(p. 1507\)](#)
- [Removing tags from SSM documents \(p. 1509\)](#)

## Creating documents with tags

You can add tags to custom SSM documents at the time you create them.

For information, see the following topics:

- [Create an SSM document \(console\) \(p. 1355\)](#)
- [Create an SSM document \(command line\) \(p. 1355\)](#)

## Adding tags to existing documents

You can add tags to custom SSM documents that you own by using the Systems Manager console or the command line.

### Topics

- [Adding tags to an existing SSM document \(console\) \(p. 1507\)](#)
- [Adding tags to an existing SSM document \(command line\) \(p. 1507\)](#)

## Adding tags to an existing SSM document (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 SSM 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 & macOS

```
aws ssm list-documents
```

Windows

```
aws ssm list-documents
```

PowerShell

```
Get-SSMDocumentList
```

Note the name of a document that you want to tag.

2. Run the following command to tag a document.

Linux & macOS

```
aws ssm add-tags-to-resource \
--resource-type "Document" \
--resource-id "document-name" \
```

```
--tags "Key=tag-key,Value=tag-value"
```

### Windows

```
aws ssm add-tags-to-resource ^
--resource-type "Document" ^
--resource-id "document-name" ^
--tags "Key=tag-key,Value=tag-value"
```

### PowerShell

```
$tag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
```

```
$tag.Key = "tag-key"
```

```
$tag.Value = "tag-value"
```

```
Add-SMResourceTag `^
-ResourceType "Document" `^
-ResourceId "document-name" `^
-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*.

**document-name** is the name of the SSM document you want to tag.

If successful, the command has no output.

3. Run the following command to verify the document tags.

### Linux & macOS

```
aws ssm list-tags-for-resource \
--resource-type "Document" \
--resource-id "document-name"
```

### Windows

```
aws ssm list-tags-for-resource ^
--resource-type "Document" ^
--resource-id "document-name"
```

### PowerShell

```
Get-SMResourceTag `^
-ResourceType "Document" `^
-ResourceId "document-name"
```

## Removing tags from SSM documents

You can use the Systems Manager console or the command line to remove tags from SSM documents.

### Topics

- [Removing tags from SSM documents \(console\) \(p. 1509\)](#)
- [Removing tags from SSM documents \(command line\) \(p. 1509\)](#)

## Removing tags from SSM documents (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 SSM documents (command line)

1. Using your preferred command line tool, run the following command to list the documents in your account.

Linux & macOS

```
aws ssm list-documents
```

Windows

```
aws ssm list-documents
```

PowerShell

```
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 & macOS

```
aws ssm remove-tags-from-resource \
--resource-type "Document" \
--resource-id "document-name" \
--tag-key "tag-key"
```

## Windows

```
aws ssm remove-tags-from-resource ^
--resource-type "Document" ^
--resource-id "document-name" ^
--tag-key "tag-key"
```

## PowerShell

```
Remove-SSMResourceTag `^
-ResourceId "document-name" `^
-ResourceType "Document" `^
-TagKey "tag-key" `^
-Force
```

*document-name* is the name of the SSM document from which you want to remove tags.

*tag-key* is the name of a key assigned to the document. For example, *Environment* or *Quarter*.

If successful, the command has no output.

3. Run the following command to verify the document tags.

## Linux & macOS

```
aws ssm list-tags-for-resource \
--resource-type "Document" \
--resource-id "document-name"
```

## Windows

```
aws ssm list-tags-for-resource ^
--resource-type "Document" ^
--resource-id "document-name"
```

## PowerShell

```
Get-SSMResourceTag `^
-ResourceType "Document" `^
-ResourceId "document-name"
```

# Tagging maintenance windows

The topics in this section describe how to work with tags on maintenance windows.

## Topics

- [Creating maintenance windows with tags \(p. 1511\)](#)
- [Adding tags to existing maintenance windows \(p. 1511\)](#)
- [Removing tags from maintenance windows \(p. 1513\)](#)

## 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 \(console\) \(p. 724\)](#)
- [Tutorial: Create and configure a maintenance window \(AWS CLI\) \(p. 733\)](#)

## Adding tags to existing maintenance windows

You can add tags to maintenance windows that you own by using the AWS Systems Manager console or the command line.

### Topics

- [Adding tags to an existing maintenance window \(console\) \(p. 1511\)](#)
- [Adding tags to an existing maintenance window \(AWS CLI\) \(p. 1511\)](#)
- [Tag a maintenance window \(AWS Tools for PowerShell\) \(p. 1512\)](#)

## Adding tags to an existing maintenance window (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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.

```
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 & macOS

```
aws ssm add-tags-to-resource \
--resource-type "MaintenanceWindow" \
--resource-id "window-id" \
```

```
--tags "Key=tag-key,Value=tag-value"
```

### Windows

```
aws ssm add-tags-to-resource ^
--resource-type "MaintenanceWindow" ^
--resource-id "window-id" ^
--tags "Key=tag-key,Value=tag-value"
```

If successful, the command has no output.

**window-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*.

- Run the following command to verify the maintenance window tags.

### Linux & macOS

```
aws ssm list-tags-for-resource \
--resource-type "MaintenanceWindow" \
--resource-id "window-id"
```

### Windows

```
aws ssm list-tags-for-resource ^
--resource-type "MaintenanceWindow" ^
--resource-id "window-id"
```

## Tag a maintenance window (AWS Tools for PowerShell)

- Run the following command to list maintenance windows that you can tag.

```
Get-SSMMaintenanceWindow
```

- Run the following commands to tag a maintenance window.

```
$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.

```
Get-SSMResourceTag `
 -ResourceType "MaintenanceWindow" `
 -ResourceId "window-id"
```

## 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 \(console\) \(p. 1513\)](#)
- [Removing tags from maintenance windows \(command line\) \(p. 1513\)](#)

## Removing tags from maintenance windows (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 & macOS

```
aws ssm describe-maintenance-windows
```

Windows

```
aws ssm describe-maintenance-windows
```

PowerShell

```
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 & macOS

```
aws ssm remove-tags-from-resource \
--resource-type "MaintenanceWindow" \
--resource-id "window-id" \
--tag-key "tag-key"
```

Windows

```
aws ssm remove-tags-from-resource ^
--resource-type "MaintenanceWindow" ^
--resource-id "window-id" ^
--tag-key "tag-key"
```

PowerShell

```
Remove-SSMResourceTag `
- ResourceType "MaintenanceWindow" `
- ResourceId "window-id" `
- TagKey "tag-key"
```

**window-id** is the ID of the maintenance window from which you want to remove a tag, such as mw-0c50858d01EXAMPLE.

**tag-key** is the name of a key assigned to the maintenance window. For example, *Environment* or *Quarter*.

If successful, the command has no output.

3. Run the following command to verify the maintenance window tags.

Linux & macOS

```
aws ssm list-tags-for-resource \
--resource-type "MaintenanceWindow" \
--resource-id "window-id"
```

Windows

```
aws ssm list-tags-for-resource ^
--resource-type "MaintenanceWindow" ^
--resource-id "window-id"
```

PowerShell

```
Get-SSMResourceTag `
- ResourceType "MaintenanceWindow" `
- ResourceId "window-id"
```

# Tagging managed nodes

The topics in this section describe how to work with tags on managed nodes.

A managed node is any machine configured for AWS Systems Manager. This includes Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and on-premises servers or virtual machines (VMs) in a hybrid environment that are configured for Systems Manager.

The instructions in this topic are applicable to any machine that is managed using Systems Manager.

## Topics

- [Creating or activating managed nodes with tags \(p. 1515\)](#)
- [Adding tags to existing managed nodes \(p. 1515\)](#)
- [Removing tags from managed nodes \(p. 1517\)](#)

## Creating or activating managed nodes 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 [Tag 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 \(p. 41\)](#).

## Adding tags to existing managed nodes

You can add tags to managed nodes by using the Systems Manager console or the command line.

## Topics

- [Adding tags to an existing managed node \(console\) \(p. 1515\)](#)
- [Adding tags to an existing managed node \(command line\) \(p. 1516\)](#)

## Adding tags to an existing managed node (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

3. Choose the ID of the managed node to add tags to, and then choose the **Tags** tab.

### Note

If a managed node you expect to see isn't listed, see [Troubleshooting managed node availability \(p. 816\)](#) for troubleshooting tips.

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 node (command line)

### To add tags to an existing managed node (command line)

1. Using your preferred command line tool, run the following command to view the list of managed nodes that you can tag.

Linux & macOS

```
aws ssm describe-instance-information
```

Windows

```
aws ssm describe-instance-information
```

PowerShell

```
Get-SSMInstanceInformation
```

Note the ID of a managed node 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-02573cafefEXAMPLE`.

2. Run the following command to tag a managed node.

Linux & macOS

```
aws ssm add-tags-to-resource \
--resource-type "ManagedInstance" \
--resource-id "instance-id" \
--tags "Key=tag-key,Value=tag-value"
```

Windows

```
aws ssm add-tags-to-resource ^
--resource-type "ManagedInstance" ^
--resource-id "instance-id" ^
--tags "Key=tag-key,Value=tag-value"
```

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 node you want to tag.

If successful, the command has no output.

- Run the following command to verify the managed node tags.

Linux & macOS

```
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"
```

PowerShell

```
Get-SSMResourceTag ^
-ResourceType "ManagedInstance" ^
-ResourceId "instance-id"
```

## Removing tags from managed nodes

You can use the Systems Manager console or the command line to remove tags from managed nodes.

### Topics

- [Removing tags from managed nodes \(console\) \(p. 1517\)](#)
- [Removing tags from managed nodes \(command line\) \(p. 1518\)](#)

## Removing tags from managed nodes (console)

- Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
- In the navigation pane, choose **Fleet Manager**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Fleet Manager** in the navigation pane.

- Choose the name of the managed node to remove tags from, and then choose the **Tags** tab.
- In the **Tags** section, choose **Edit**, and then choose **Remove** next to the tag pair you no longer need.
- Choose **Save**.

## Removing tags from managed nodes (command line)

1. Using your preferred command line tool, run the following command to list the managed nodes in your account.

Linux & macOS

```
aws ssm describe-instance-information
```

Windows

```
aws ssm describe-instance-information
```

PowerShell

```
Get-SSMInstanceInformation
```

Note the name of a managed node from which you want to remove tags.

2. Run the following command to remove tags from a managed node.

Linux & macOS

```
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 `
-ResourceId "instance-id" `
-ResourceType "ManagedInstance" `
-TagKey "tag-key" `
-Force
```

*instance-id* is the name of the managed node from which you want to remove tags.

*tag-key* is the name of a key assigned to the managed node. For example, *Environment* or *Quarter*.

If successful, the command has no output.

3. Run the following command to verify the managed node tags.

Linux & macOS

```
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"
```

#### PowerShell

```
Get-SSMResourceTag `^
- ResourceType "ManagedInstance" `^
- ResourceId "instance-id"
```

## Tagging OpsItems

The topics in this section describe how to work with tags on OpsItems.

### Topics

- [Creating OpsItems with tags \(p. 1519\)](#)
- [Adding tags to existing OpsItems \(p. 1519\)](#)
- [Removing tags from Systems Manager OpsItems \(p. 1521\)](#)

## Creating OpsItems with tags

You can add tags to custom AWS Systems Manager OpsItems at the time you create them if you use a command line tool.

For information, see the following topic:

- [Creating OpsItems by using the AWS CLI \(p. 206\)](#)

## Adding tags to existing OpsItems

You can add tags to OpsItems by using a command line tool.

### Topics

- [Adding tags to an existing OpsItem \(command line\) \(p. 1519\)](#)

## 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 & macOS

```
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 & macOS

```
aws ssm add-tags-to-resource \
--resource-type "OpsItem" \
--resource-id "ops-item-id" \
--tags "Key=tag-key,Value=tag-value"
```

Windows

```
aws ssm add-tags-to-resource ^
--resource-type "OpsItem" ^
--resource-id "ops-item-id" ^
--tags "Key=tag-key,Value=tag-value"
```

PowerShell

```
$tag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
```

```
$tag.Key = "tag-key"
```

```
$tag.Value = "tag-value"
```

```
Add-SSMResourceTag `
-ResourceType "OpsItem" `
-ResourceId "ops-item-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*.

*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 & macOS

```
aws ssm list-tags-for-resource \
--resource-type "OpsItem" \
--resource-id "ops-item-id"
```

Windows

```
aws ssm list-tags-for-resource ^
--resource-type "OpsItem" ^
--resource-id "ops-item-id"
```

PowerShell

```
Get-SSMResourceTag `
-ResourceType "OpsItem" `
-ResourceId "ops-item-id"
```

## 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 \(command line\) \(p. 1521\)](#)

## 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 & macOS

```
aws ssm describe-ops-items
```

Windows

```
aws ssm describe-ops-items
```

PowerShell

```
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 & macOS

```
aws ssm remove-tags-from-resource \
--resource-type "OpsItem" \
--resource-id "ops-item-id" \
```

```
--tag-key "tag-key"
```

#### Windows

```
aws ssm remove-tags-from-resource ^
--resource-type "OpsItem" ^
--resource-id "ops-item-id" ^
--tag-key "tag-key"
```

#### PowerShell

```
Remove-SSMResourceTag `-
-ResourceId "ops-item-id" `-
-ResourceType "OpsItem" `-
-TagKey "tag-key" `-
-Force
```

*ops-item-id* is the ID of the OpsItem from which you want to remove tags.

*tag-key* is the name of a key assigned to the OpsItem. For example, *Environment* or *Quarter*.

If successful, the command has no output.

- Run the following command to verify the OpsItem tags.

#### Linux & macOS

```
aws ssm list-tags-for-resource \
--resource-type "OpsItem" \
--resource-id "ops-item-id"
```

#### Windows

```
aws ssm list-tags-for-resource ^
--resource-type "OpsItem" ^
--resource-id "ops-item-id"
```

#### PowerShell

```
Get-SSMResourceTag `-
-ResourceType "OpsItem" `-
-ResourceId "ops-item-id"
```

## Tagging automations

The topics in this section describe how to work with tags on automations. You can add a maximum of five tags to AWS Systems Manager automations. You can add tags to automations at the time you initiate them from either the console or the command line, or after they've run by using the command line.

### Adding tags to automations (console)

- Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
- 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 Automation runbook you want to run.
4. Select **Execute automation**.
5. In the **Tags** section, choose **Edit**, and then add one or more key-value tag pairs.
6. Choose **Save**.

## Adding tags to automations (command line)

Using your preferred command line tool, run the following command to add tags to an automation when it starts.

Linux & macOS

```
aws ssm start-automation-execution \
--document-name DocumentName \
--parameters ParametersRequiredByDocument \
--tags "Key=ExampleKey,Value=ExampleValue"
```

Windows

```
aws ssm start-automation-execution ^
--document-name DocumentName ^
--parameters ParametersRequiredByDocument ^
--tags "Key=ExampleKey,Value=ExampleValue"
```

PowerShell

```
$exampleTag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$exampleTag.Key = "ExampleKey"
$exampleTag.Value = "ExampleValue"

Start-SMAutomationExecution ^
-DocumentName DocumentName ^
-Parameter ParametersRequiredByDocument
-Tag $exampleTag
```

1. You can also tag automations after they run by using your preferred command line tool. Run the following command to add tags to an automation.

Linux & macOS

```
aws ssm add-tags-to-resource \
--resource-type "Automation" \
--resource-id "automation-execution-id" \
--tags "Key=ExampleKey,Value=ExampleValue"
```

Windows

```
aws ssm add-tags-to-resource ^
--resource-type "Automation" ^
--resource-id "automation-execution-id" ^
```

```
--tags "Key=ExampleKey,Value=ExampleValue"
```

#### PowerShell

```
$exampleTag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$exampleTag.Key = "ExampleKey"
$exampleTag.Value = "ExampleValue"

Add-SSMResourceTag ^
 -ResourceType "Automation" ^
 -ResourceId "automation-execution-id" ^
 -Tag $exampleTag ^
 -Force
```

*ExampleKey* is the name of a custom key you specify. For example, *Region* or *Quarter*.

*ExampleValue* is the custom content for the value you want to associate with that key. For example, *West* or *Q321*.

*automation-execution-id* is the ID of the automation you want to tag.

If successful, the command has no output.

2. Run the following command to verify the automation's tags.

#### Linux & macOS

```
aws ssm list-tags-for-resource \
 --resource-type "Automation" \
 --resource-id "automation-execution-id"
```

#### Windows

```
aws ssm list-tags-for-resource ^
 --resource-type "Automation" ^
 --resource-id "automation-execution-id"
```

#### PowerShell

```
Get-SSMResourceTag ^
 -ResourceType "Automation" ^
 -ResourceId "automation-execution-id"
```

## Removing tags from automations

You can use a command line tool to remove tags from an automation.

### Removing tags from automations (command line)

1. Using your preferred command line tool, run the following command to remove a tag from an automation.

#### Linux & macOS

```
aws ssm remove-tags-from-resource \
```

```
--resource-type "Automation" \
--resource-id "automation-execution-id" \
--tag-key "tag-key"
```

#### Windows

```
aws ssm remove-tags-from-resource ^
--resource-type "Automation" ^
--resource-id "automation-execution-id" ^
--tag-key "tag-key"
```

#### PowerShell

```
Remove-SSMResourceTag `^
-ResourceId "automation-execution-id" `^
-ResourceType "Automation" `^
-TagKey "tag-key" `^
-Force
```

- Run the following command to verify the automation's tags.

#### Linux & macOS

```
aws ssm list-tags-for-resource \
--resource-type "Automation" \
--resource-id "automation-execution-id"
```

#### Windows

```
aws ssm list-tags-for-resource ^
--resource-type "Automation" ^
--resource-id "automation-execution-id"
```

#### PowerShell

```
Get-SSMResourceTag `^
-ResourceType "Automation" `^
-ResourceId "automation-execution-id"
```

## Tagging Systems Manager parameters

The topics in this section describe how to work with tags on AWS Systems Manager parameters (SSM parameters).

### Topics

- [Creating parameters with tags \(p. 1525\)](#)
- [Adding tags to existing parameters \(p. 1526\)](#)
- [Removing tags from SSM parameters \(p. 1527\)](#)

## Creating parameters with tags

You can add tags to SSM parameters at the time you create them.

For information, see the following topics:

- [Create a Systems Manager parameter \(console\) \(p. 281\)](#)
- [Create a Systems Manager parameter \(AWS CLI\) \(p. 282\)](#)
- [Create a Systems Manager parameter \(Tools for Windows PowerShell\) \(p. 292\)](#)

## Adding tags to existing parameters

You can add tags to custom SSM parameters that you own by using the Systems Manager console or the command line.

### Topics

- [Adding tags to an existing parameter \(console\) \(p. 1526\)](#)
- [Adding tags to an existing parameter \(AWS CLI\) \(p. 1526\)](#)
- [Adding tags to an existing parameter \(AWS Tools for PowerShell\) \(p. 1527\)](#)

## Adding tags to an existing parameter (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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.

```
aws ssm describe-parameters
```

Note the name of a parameter that you want to tag.

2. Run the following command to tag a parameter.

```
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.

```
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.

```
Get-SSMParameterList
```

2. Run the following commands to tag a parameter.

```
$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.

```
Get-SSMResourceTag `~
 -ResourceType "Parameter" `~
 -ResourceId "parameter-name"
```

## Removing tags from SSM parameters

You can use the Systems Manager console or the command line to remove tags from SSM parameters.

### Topics

- [Removing tags from SSM parameters \(console\) \(p. 1527\)](#)
- [Removing tags from SSM parameters \(command line\) \(p. 1528\)](#)

## Removing tags from SSM parameters (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.

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 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 SSM parameters (command line)

1. Using your preferred command line tool, run the following command to list the parameters in your account.

Linux & macOS

```
aws ssm describe-parameters
```

Windows

```
aws ssm describe-parameters
```

PowerShell

```
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 & macOS

```
aws ssm remove-tags-from-resource \
--resource-type "Parameter" \
--resource-id "parameter-name" \
--tag-key "tag-key"
```

Windows

```
aws ssm remove-tags-from-resource ^
--resource-type "Parameter" ^
--resource-id "parameter-name" ^
--tag-key "tag-key"
```

PowerShell

```
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 & macOS

```
aws ssm list-tags-for-resource \
--resource-type "Parameter" \
--resource-id "parameter-name"
```

Windows

```
aws ssm list-tags-for-resource ^
--resource-type "Parameter" ^
--resource-id "parameter-name"
```

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 \(p. 1529\)](#)
- [Adding tags to existing patch baselines \(p. 1529\)](#)
- [Removing tags from patch baselines \(p. 1531\)](#)

## Creating patch baselines with tags

You can add tags to AWS Systems Manager patch baselines at the time you create them.

For information, see the following topics:

- [Working with custom patch baselines \(console\) \(p. 1194\)](#)
- [Create a patch baseline \(p. 1211\)](#)
- [Create a patch baseline with custom repositories for different OS versions \(p. 1212\)](#)

## 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 \(console\) \(p. 1530\)](#)

- [Adding tags to an existing patch baseline \(AWS CLI\) \(p. 1530\)](#)
- [Tag a patch baseline \(AWS Tools for PowerShell\) \(p. 1531\)](#)

## Adding tags to an existing patch baseline (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 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 patch baseline (AWS CLI)

1. Using your preferred command line tool, run the following command to view the list of patch baselines that you can tag.

```
aws ssm describe-patch-baselines --filters "Key=OWNER,Values=[Self]"
```

Note the ID of a patch baseline that you want to tag.

2. Run the following command to tag a patch baseline.

Linux & macOS

```
aws ssm add-tags-to-resource \
--resource-type "PatchBaseline" \
--resource-id "baseline-id" \
--tags "Key=tag-key,Value=tag-value"
```

Windows

```
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.

### Linux & macOS

```
aws ssm list-tags-for-resource \
--resource-type "PatchBaseline" \
--resource-id "baseline-id"
```

### Windows

```
aws ssm list-tags-for-resource ^
--resource-type "PatchBaseline" ^
--resource-id "patchbaseline-id"
```

## Tag a patch baseline (AWS Tools for PowerShell)

1. Run the following command to list patch baseline that you can tag.

```
Get-SSMPatchBaseline
```

2. Run the following commands to tag a patch baseline.

```
$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 patch baseline 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.

```
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 a patch baseline.

### Topics

- [Removing tags from patch baseline \(console\) \(p. 1532\)](#)

- [Removing tags from patch baselines \(command line\) \(p. 1532\)](#)

## Removing tags from patch baseline (console)

1. Open the AWS Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
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 & macOS

```
aws ssm describe-patch-baselines
```

Windows

```
aws ssm describe-patch-baselines
```

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 & macOS

```
aws ssm remove-tags-from-resource \
--resource-type "PatchBaseline" \
--resource-id "baseline-id" \
--tag-key "tag-key"
```

Windows

```
aws ssm remove-tags-from-resource ^
--resource-type "PatchBaseline" ^
--resource-id "baseline-id" ^
--tag-key "tag-key"
```

## 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.

- Run the following command to verify the patch baseline tags.

## Linux & macOS

```
aws ssm list-tags-for-resource \
--resource-type "PatchBaseline" \
--resource-id "baseline-id"
```

## Windows

```
aws ssm list-tags-for-resource ^
--resource-type "PatchBaseline" ^
--resource-id "baseline-id"
```

## PowerShell

```
Get-SSMResourceTag `^
-ResourceType "PatchBaseline" `^
-ResourceId "baseline-id"
```

# AWS Systems Manager reference

The following information and topics can help you better implement AWS Systems Manager solutions.

## Principal

In AWS Identity and Access Management (IAM), you can grant or deny a service access to resources using the Principal policy element. The Principal policy element value for Systems Manager is `ssm.amazonaws.com`.

## Supported AWS Regions and endpoints

See [Systems Manager service endpoints](#) in the *Amazon Web Services General Reference*.

## Service Quotas

See [Systems Manager service quotas](#) in the *Amazon Web Services General Reference*.

## API Reference

See [AWS Systems Manager API Reference](#).

## AWS CLI Command Reference

See [AWS Systems Manager section of the AWS CLI Command Reference](#).

## AWS Tools for PowerShell Cmdlet Reference

See [AWS Systems Manager section of the AWS Tools for PowerShell Cmdlet Reference](#).

## SSM Agent Repository on GitHub

See [aws/amazon-ssm-agent](#).

## Ask a Question

Systems Manager issues in [AWS re:Post](#)

## AWS News Blog

[Management Tools](#)

## More reference topics

- [Reference: Amazon EventBridge event patterns and types for Systems Manager \(p. 1534\)](#)
- [Reference: Cron and rate expressions for Systems Manager \(p. 1540\)](#)
- [Reference: ec2messages, ssmmessages, and other API operations \(p. 1546\)](#)
- [Reference: Creating formatted date and time strings for Systems Manager \(p. 1547\)](#)

## Reference: Amazon EventBridge event patterns and types for Systems Manager

### Note

Amazon EventBridge is the preferred way to manage your events. CloudWatch Events and EventBridge are the same underlying service and API, but EventBridge provides more features.

Changes you make in either CloudWatch or EventBridge are reflected in each console. For more information, see the [Amazon EventBridge User Guide](#).

Using Amazon EventBridge, you can create *rules* that match incoming *events* and route them to *targets* for processing.

An event indicates a change in an environment in your own applications, software as a service (SaaS) applications, or an AWS service. Events are produced on a best effort basis. After an event type that is specified in a rule is detected, EventBridge routes it to a specified target for processing. Targets can include Amazon Elastic Compute Cloud (Amazon EC2) instances, AWS Lambda functions, Amazon Kinesis streams, Amazon Elastic Container Service (Amazon ECS) tasks, AWS Step Functions state machines, Amazon Simple Notification Service (Amazon SNS) topics, Amazon Simple Queue Service (Amazon SQS) queues, built-in targets and many more.

For information about creating EventBridge rules, see the following topics:

- [Monitoring Systems Manager events with Amazon EventBridge \(p. 1449\)](#)
- [Amazon EventBridge event examples for Systems Manager \(p. 1452\)](#)
- [Getting started with Amazon EventBridge in the Amazon EventBridge User Guide](#)

The remainder of this topic describes the types of Systems Manager events that you can include in your EventBridge rules.

## Event type: Automation

Event type name	Description of events you can add to a rule
EC2 Automation Execution Status-change Notification	The overall status of an Automation workflow changes. You can add one or more of the following status changes to an event rule: <ul style="list-style-type: none"><li>• Approved</li><li>• Canceled</li><li>• Failed</li><li>• PendingApproval</li><li>• PendingChangeCalendarOverride</li><li>• Rejected</li><li>• Scheduled</li><li>• Success</li><li>• TimedOut</li></ul>
EC2 Automation Step Status-change Notification	The status of a specific step in an Automation workflow changes. You can add one or more of the following status changes to an event rule: <ul style="list-style-type: none"><li>• Canceled</li><li>• Failed</li><li>• Success</li><li>• TimedOut</li></ul>

## Event type: Change Calendar

Event type name	Description of events you can add to a rule
Calendar State Change	<p>The state of a Change Calendar changes. You can add one or both of the following state changes to an event rule:</p> <ul style="list-style-type: none"><li>• OPEN</li><li>• CLOSED</li></ul> <p>State changes for calendars shared from other AWS accounts aren't supported.</p>

## Event type: Configuration Compliance

Event type name	Description of events you can add to a rule
Configuration Compliance State Change	<p>The state of a managed node changes, for either association compliance or patch compliance. You can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"><li>• compliant</li><li>• non_compliant</li></ul>

## Event type: Inventory

Event type name	Description of events you can add to a rule
Inventory Resource State Change	<p>The deletion of custom inventory and a <a href="#">PutInventory</a> call that uses an old schema version. You can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"><li>• Custom inventory type deleted event on a specific node. EventBridge sends one event per node per custom InventoryType.</li><li>• Custom inventory type deleted event for all nodes.</li><li>• PutInventory call with old schema version event. EventBridge sends this event when the schema version is less than the current schema. This event applies to all inventory types.</li></ul> <p>For more information, see <a href="#">About EventBridge monitoring of Inventory events (p. 865)</a>.</p>

## Event type: State Manager

Event type name	Description of events you can add to a rule
EC2 State Manager Association State Change	<p>The <i>overall</i> state of an Association changes as it's being applied. You can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"> <li>• Failed</li> <li>• Pending</li> <li>• Success</li> </ul>
EC2 State Manager Instance Association State Change	<p>The state of a <i>single</i> managed instance that is targeted by an Association changes. You can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"> <li>• Failed</li> <li>• Pending</li> <li>• Success</li> </ul>

## Event type: Maintenance Window

Event type name	Description of events you can add to a rule
Maintenance Window Status-change Notification	<p>The overall status of one or more maintenance windows changes. You can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"> <li>• DISABLED</li> <li>• ENABLED</li> </ul>
Maintenance Window Target Registration Notification	<p>The status of one or more maintenance window targets changes. You can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"> <li>• Deregistered</li> <li>• Registered</li> <li>• Updated</li> </ul>
Maintenance Window Execution State-change Notification	<p>The overall status of a maintenance window changes while it's running. You can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"> <li>• CANCELLED</li> <li>• CANCELLING</li> <li>• FAILED</li> <li>• IN_PROGRESS</li> <li>• PENDING</li> <li>• SKIPPED_OVERLAPPING</li> <li>• SUCCESS</li> </ul>

Event type name	Description of events you can add to a rule
	<ul style="list-style-type: none"> <li>TIMED_OUT</li> </ul>
Maintenance Window Task Execution State-change Notification	<p>The state of a task in a maintenance window changes while it's running. You can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"> <li>CANCELLED</li> <li>CANCELLING</li> <li>FAILED</li> <li>IN_PROGRESS</li> <li>SUCCESS</li> <li>TIMED_OUT</li> </ul>
Maintenance Window Task Target Invocation State-change Notification	<p>The state of a maintenance window task on a specific target changes.</p> <p>This notification is fully supported only for Run Command tasks. For this type of task, you can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"> <li>CANCELLED</li> <li>CANCELLING</li> <li>FAILED</li> <li>IN_PROGRESS</li> <li>SUCCESS</li> <li>TIMED_OUT</li> </ul> <p>For Automation, AWS Lambda, and AWS Step Functions tasks, EventBridge reports only the states IN_PROGRESS and COMPLETE. COMPLETE is reported whether the task is successful or not.</p>
Maintenance Window Task Registration Notification	<p>The state of one or more maintenance window tasks changes. You can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"> <li>Deregistered</li> <li>REGISTERED</li> <li>UPDATED</li> </ul>

## Event type: Parameter Store

Event type name	Description of events you can add to a rule
Parameter Store Change	<p>The state of a parameter changes. You can add one or more of the following state changes to an event rule:</p> <ul style="list-style-type: none"> <li>Create</li> </ul>

Event type name	Description of events you can add to a rule
	<ul style="list-style-type: none"> <li>• Update</li> <li>• Delete</li> <li>• LabelParameterVersion</li> </ul> <p data-bbox="910 411 1480 481">For more information, see <a href="#">Configuring EventBridge for parameters (p. 276)</a>.</p>
Parameter Store Policy Action	<p data-bbox="910 496 1480 587">A condition of an advanced parameter policy change is met. You can add one or more of the following status changes to an event rule:</p> <ul style="list-style-type: none"> <li>• Expiration</li> <li>• ExpirationNotification</li> <li>• NoChangeNotification</li> </ul> <p data-bbox="910 760 1480 830">For more information, see <a href="#">Configuring EventBridge for parameter policies (p. 277)</a>.</p>

## Event type: Run Command

Event type name	Description of events you can add to a rule
EC2 Command Invocation Status-change Notification	<p data-bbox="910 1058 1480 1170">The status of a command sent to an individual managed instance changes. You can add one or more of the following status changes to an event rule:</p> <ul style="list-style-type: none"> <li>• Success</li> <li>• InProgress</li> <li>• TimedOut</li> <li>• Canceled</li> <li>• Failed</li> </ul>
EC2 Command Status-change Notification	<p data-bbox="910 1389 1480 1480">The overall status of a command changes. You can add one or more of the following status changes to an event rule:</p> <ul style="list-style-type: none"> <li>• Success</li> <li>• InProgress</li> <li>• TimedOut</li> <li>• Canceled</li> <li>• Failed</li> </ul>

# 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. State Manager is a capability of AWS Systems Manager. You can specify a schedule as either a time-based entry, called a *cron expression*, or a frequency-based entry, called a *rate expression*.

When you create an association or a maintenance window, you can specify a timestamp in Coordinated Universal Time (UTC) format so that it runs once at the specified time. Associations and maintenance windows also support *schedule offsets* for cron expressions only. A schedule offset is the number of days to wait after the date and time specified by a cron expression before running the association or maintenance window. For example, the following cron expression schedules an association or maintenance window to run the third Tuesday of every month at 11:30 PM.

```
cron(30 23 ? * TUE#3 *)
```

If the schedule offset is 2, the maintenance window won't run until 11:30 PM two days later.

## Note

If you create an association or a maintenance window with a cron expression that targets a day that has already passed in the current period, but add a schedule offset date that falls in the future, the association or maintenance window won't run in the period. It will go into effect in the following period. For example, if you specify a cron expression that would have run a maintenance window yesterday and add a schedule offset of two days, the maintenance window won't run tomorrow.

When you create either an association or maintenance window programmatically or by using a command line tool such as the AWS Command Line Interface (AWS CLI), specify a schedule parameter with a valid cron or rate expression (or timestamp 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 timestamp. The following examples use the AWS CLI on a local Linux machine.

```
aws ssm create-maintenance-window \
--name "My-Cron-Maintenance-Window" \
--allow-unassociated-targets \
--schedule "cron(0 16 ? * TUE *)" \
--schedule-timezone "America/Los_Angeles" \
--start-date 2021-01-01T00:00:00-08:00 \
--end-date 2021-06-30T00:00:00-08:00 \
--duration 4 \
--cutoff 1
```

```
aws ssm create-maintenance-window \
--name "My-Cron-Offset-Maintenance-Window" \
--allow-unassociated-targets \
--schedule "cron(30 23 ? * TUE#3 *)" \
--duration 4 \
--cutoff 1 \
--schedule-offset 2
```

```
aws ssm create-maintenance-window \
--name "My-Rate-Maintenance-Window" \
--allow-unassociated-targets \
--schedule "rate(7 days)" \
--duration 4 \
--schedule-timezone "America/Los_Angeles" \
--cutoff 1
```

```
aws ssm create-maintenance-window \
--name "My-TimeStamp-Maintenance-Window" \
--allow-unassociated-targets \
--schedule "at(2021-07-07T13:15:30)" \
--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. The following examples use the AWS CLI on a local Linux machine.

```
aws ssm create-association \
--association-name "My-Cron-Association" \
--schedule-expression "cron(0 2 ? * SUN *)" \
--targets Key=tag:ServerRole,Values=WebServer \
--name AWS-UpdateSSMAgent
```

```
aws ssm create-association \
--association-name "My-Rate-Association" \
--schedule-expression "rate(7 days)" \
--targets Key=tag:ServerRole,Values=WebServer \
--name AWS-UpdateSSMAgent
```

```
aws ssm create-association \
--association-name "My-Rate-Association" \
--schedule-expression "at(2020-07-07T15:55:00)" \
--targets Key=tag:ServerRole,Values=WebServer \
--name AWS-UpdateSSMAgent \
--apply-only-at-cron-interval
```

### Note

By default, when you create a new association, the system runs it immediately after it's created and then according to the schedule you specified. Specify `--apply-only-at-cron-interval` so that the association doesn't run immediately after you create it. This parameter isn't supported for rate expressions.

### Topics

- [General information about cron and rate expressions \(p. 1541\)](#)
- [Cron and rate expressions for associations \(p. 1544\)](#)
- [Cron and rate expressions for maintenance windows \(p. 1545\)](#)

## 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

Minutes	Hours	Day of month	Month	Day of week	Year	Meaning
0	10	*	*	?	*	Run at 10:00 am (UTC) every day
15	12	*	*	?	*	Run at 12:15 PM (UTC) every day
0	18	?	*	MON-FRI	*	Run at 6:00 PM (UTC) every Monday through Friday
0	8	1	*	?	*	Run at 8:00 AM (UTC) every 1st day of the month

## Supported values

The following table shows supported values for required cron entries.

### Note

Cron expressions for associations don't support all these values. For information, see [Cron and rate expressions for associations \(p. 1544\)](#).

## Supported values for cron expressions

Field	Values	Wildcards
Minutes	0-59	, - * /
Hours	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /
Day-of-week	1-7 or SUN-SAT	, - * ? / L
Year	1970-2199	, - * /

### Note

You can't specify a value in the Day-of-month and in the Day-of-week fields in the same cron expression. If you specify a value in one of the fields, use a ? (question mark) in the other field.

## Wildcards

The following table shows the wildcard values that cron expressions support.

## Supported wildcards for cron expressions

Wildcard	Description
,	The , (comma) wildcard includes additional values. In the Month field, JAN,FEB,MAR would include January, February, and March.
-	The - (dash) wildcard specifies ranges. In the Day field, 1-15 would include days 1 through 15 of the specified month.
*	The * (asterisk) wildcard includes all values in the field. In the Hours field, * would include every hour.
/	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.
?	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.
L	The L wildcard in the Day-of-month or Day-of-week fields specifies the last day of the month or week.
W	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.
#	The # wildcard in the day-of-week field followed by a number between one and five specifies a given day of the month. 5#3 specifies the 3rd Friday of the month.

### Note

Cron expressions that lead to rates faster than five (5) minute aren't supported. Support for specifying both a day-of-week and a day-of-month value isn't complete. Use the question mark (?) character in one of these fields.

For more information about cron expressions, see [CRON expression](#) at the [Wikipedia website](#).

## Rate expressions

Rate expressions have the following two required fields. Fields are separated by spaces.

### Required fields for rate expressions

Field	Values
Value	positive number, such as 1 or 15

Field	Values
Unit	minute
	minutes
	hour
	hours
	day
	days

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 hours)` and `rate(5 hour)` aren't 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 information:

- Associations support the following cron expressions: every 1/2, 1, 2, 4, 8, or 12 hours; every day, every week, every *n*th day, or the last *x* day of the month at a specific time.
- Associations 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 be 0 (zero). For example: `cron(0 */30 * * * ? *)`

Associations support cron expressions that include a day of the week and the number sign (#) to designate the *n*th day of a month to run an association. Here is an example that runs a cron schedule on the third Tuesday of every month at 23:30 UTC:

```
cron(30 23 ? * TUE#3 *)
```

Here is an example that runs on the second Thursday of every month at midnight UTC:

```
cron(0 0 ? * THU#2 *)
```

Associations also support the (L) sign to indicate the last *X* day of the month. Here is an example that runs a cron schedule on the last Tuesday of every month at midnight UTC:

```
cron(0 0 ? * 3L *)
```

To further control when an association runs, for example if you want to run an association two days after patch Tuesday, you can specify an offset. An *offset* defines how many days to wait after the scheduled day to run an association. For example, if you specified a cron schedule of `cron(0 0 ? * THU#2 *)`, you could specify the number 3 in the **Schedule offset** field to run the association each Sunday after the second Thursday of the month.

To use offsets, you must either choose the **Apply association only at the next specified Cron interval** option in the console or you must specify the `useApplyOnlyAtCronInterval` parameter from the command line. This option tells State Manager not to run an association immediately after you create it.

### Note

For an association that collects metadata for Inventory, a capability of AWS Systems Manager, we recommend using a rate expression.

The following table presents cron examples for associations using the required six fields.

#### Cron examples for associations

Example	Details
cron(0/30 * * * ? *)	Every 30 minutes
cron(0 0/1 * * ? *)	Every hour
cron(0 0/2 * * ? *)	Every 2 hours
cron(0 0/4 * * ? *)	Every 4 hours
cron(0 0/8 * * ? *)	Every 8 hours
cron(0 0/12 * * ? *)	Every 12 hours
cron(15 13 ? * * *)	Every day at 1:15 PM
cron(15 13 ? * MON *)	Every Monday at 1:15 PM
cron(30 23 ? * TUE#3 *)	The third Tuesday of every month at 11:30 PM

Here are some rate examples for associations.

#### Rate examples for associations

Example	Details
rate(30 minutes)	Every 30 minutes
rate(1 hour)	Every hour
rate(5 hours)	Every 5 hours
rate(15 days)	Every 15 days

## 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.

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

Example	Details
cron(0 2 ? * THU#3 *)	02:00 AM the third Thursday of every month
cron(15 10 ? * * *)	10:15 AM every day
cron(15 10 ? * MON-FRI *)	10:15 AM every Monday, Tuesday, Wednesday, Thursday and Friday
cron(0 2 L * ? *)	02:00 AM on the last day of every month
cron(15 10 ? * 6L *)	10:15 AM on the last Friday of every month

The following table provides rate examples for maintenance windows.

### Rate examples for maintenance windows

Example	Details
rate(30 minutes)	Every 30 minutes
rate(1 hour)	Every hour
rate(5 hours)	Every 5 hours
rate(25 days)	Every 25 days

## Reference: ec2messages, ssmmessages, and other API operations

If you monitor API operations, you might see calls to the following:

- `ec2messages:AcknowledgeMessage`
- `ec2messages:DeleteMessage`
- `ec2messages:FailMessage`
- `ec2messages:GetEndpoint`
- `ec2messages:GetMessages`
- `ec2messages:SendReply`
- `ssmmessages>CreateControlChannel`
- `ssmmessages>CreateDataChannel`
- `ssmmessages:OpenControlChannel`
- `ssmmessages:OpenDataChannel`
- `ssm:DescribeInstanceProperties`
- `ssm:DescribeDocumentParameters`
- `ssm>ListInstanceAssociations`
- `ssm:RegisterManagedInstance`
- `ssm:UpdateInstanceInformation`

These are special operations used AWS Systems Manager.

### **ec2messages API operations**

**ec2messages : \* API operations** are made to the Amazon Message Delivery Service endpoint. Systems Manager uses this endpoint for API operations from Systems Manager Agent (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 operations**

Systems Manager uses the **ssmmessages** endpoint for API operations from SSM Agent to Session Manager, a capability of AWS Systems Manager, in the cloud. This endpoint is required to create and delete session channels with the Session Manager service in the cloud. For more information, see [Actions, resources, and condition keys for Amazon Session Manager Message Gateway Service](#).

### **Instance-related API operations**

**UpdateInstanceInformation:** SSM Agent calls the Systems Manager service in the cloud every 5 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 runs this API operation to see if a new State Manager association is available. This API operation is required for State Manager, a capability of AWS Systems Manager, to function.

**DescribeInstanceProperties** and **DescribeDocumentParameters:** Systems Manager runs these API operations to render specific nodes in the Amazon EC2 console. Results of the **DescribeInstanceProperties** operation are displayed in the Fleet Manager node. Results of the **DescribeDocumentParameters** operation are displayed in the **Documents** node.

**ssm:RegisterManagedInstance:** SSM Agent runs this API operation to register an on-premises server or virtual machine (VM) with Systems Manager as a managed instance using an activation code and ID, or to register AWS IoT Greengrass Version 2 credentials.

## Reference: Creating formatted date and time strings for Systems Manager

AWS Systems Manager API operations accept filters to limit the number of results returned by a request. Some of these API operations accept filters that require a formatted string to represent a specific date and time. For example, the **DescribeSessions** API operation accepts the **InvokedAfter** and **InvokedBefore** keys as some valid values for a **SessionFilter** object. Another example is the **DescribeAutomationExecutions** API operation, which accepts the **StartTimeBefore** and **StartTimeAfter** keys as some 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 aren't limited to filters. There are also API operations 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** operation. 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 operations.

## 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 Coordinated Universal 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

```
date '+%Y-%m-%dT%H:%M:%SZ'
```

### PowerShell

```
(Get-Date).ToString("yyyy-MM-ddTH:mm:ssZ")
```

When working with Systems Manager API operations, 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 Command Line Interface (AWS CLI).

### AWS CLI

- Retrieve the last week of command history for an SSM document.

```
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.

```
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.

```
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.

```
$lastWeekStamp = (Get-Date).AddDays(-7).ToString("yyyy-MM-ddTH:mm:ssZ")

$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 \(p. 397\)](#)

- Create self-service Automation runbooks for infrastructure.
- Use Automation, a capability of AWS Systems Manager, to simplify creating Amazon Machine Images (AMIs) from the AWS Marketplace or custom AMIs, using public Systems Manager documents (SSM documents) or by authoring your own workflows.
- [Build and maintain AMIs \(p. 604\)](#) using the AWS-UpdateLinuxAmi and AWS-UpdateWindowsAmi Automation runbooks, or using custom Automation runbooks that you create.

### [Inventory \(p. 848\)](#)

- Use Inventory, a capability of AWS Systems Manager, with AWS Config to audit your application configurations over time.

### [Maintenance Windows \(p. 706\)](#)

- Define a schedule to perform potentially disruptive actions on your nodes such as operating system (OS) patching, driver updates, or software installations.
- For information about the differences between State Manager and Maintenance Windows, capabilities of AWS Systems Manager, see [Choosing between State Manager and Maintenance Windows \(p. 1554\)](#).

### [Parameter Store \(p. 256\)](#)

- Use Parameter Store, a capability of AWS Systems Manager, to centrally manage global configuration settings.
- [How AWS Systems Manager Parameter Store uses AWS KMS](#).
- [Reference AWS Secrets Manager secrets from Parameter Store parameters \(p. 1488\)](#).

### [Patch Manager \(p. 1093\)](#)

- Use Patch Manager, a capability of AWS Systems Manager, to roll out patches at scale and increase fleet compliance visibility across your nodes.
- [Integrate Patch Manager with AWS Security Hub \(p. 1207\)](#) to receive alerts when nodes in your fleet go out of compliance and monitor the patching status of your fleets from a security point of view. There is a charge to use Security Hub. For more information, see [Pricing](#).

### [Run Command \(p. 991\)](#)

- [Manage Instances at Scale without SSH Access Using EC2 Run Command](#).
- Audit all API calls made by or on behalf of Run Command, a capability of AWS Systems Manager, using AWS CloudTrail.

- When you send a command using Run Command, don't include sensitive information formatted as plaintext, such as passwords, configuration data, or other secrets. All Systems Manager API activity in your account is logged in an Amazon S3 bucket for AWS CloudTrail logs. This means that any user with access to S3 bucket can view the plaintext values of those secrets. For this reason, we recommend creating and using `SecureString` parameters to encrypt sensitive data you use in your Systems Manager operations.

For more information, see [Restricting access to Systems Manager parameters using IAM policies \(p. 260\)](#).

**Note**

By default, the log files delivered by CloudTrail to your bucket are encrypted by Amazon [server-side encryption with Amazon S3-managed encryption keys \(SSE-S3\)](#). To provide a security layer that is directly manageable, you can instead use [server-side encryption with AWS KMS-managed keys \(SSE-KMS\)](#) for your CloudTrail log files.

For more information, see [Encrypting CloudTrail log files with AWS KMS-Managed Keys \(SSE-KMS\) in the AWS CloudTrail User Guide](#).

- Use the targets and rate control features in Run Command to perform a staged command operation (p. 1003).
- Use fine-grained access permissions for Run Command (and all Systems Manager capabilities) by using AWS Identity and Access Management (IAM) policies (p. 1395).

## Session Manager (p. 912)

- Audit session activity in your AWS account using AWS CloudTrail (p. 977).
- Log session data in your AWS account using Amazon CloudWatch Logs or Amazon S3 (p. 978).
- Control user session access to instances (p. 926).
- Restrict access to commands in a session (p. 952).
- Turn off or turn on `ssm-user` account administrative permissions (p. 958).

## State Manager (p. 1034)

- Update SSM Agent at least once a month using the pre-configured `AWS-UpdateSSMAgent` document (p. 1089).
- (Windows) Upload the PowerShell or DSC module to Amazon Simple Storage Service (Amazon S3), and use `AWS-InstallPowerShellModule`.
- Use tags to create application groups for your nodes. And then target nodes using the `Targets` parameter instead of specifying individual node IDs.
- Automatically remediate findings generated by Amazon Inspector by using Systems Manager.
- Use a centralized configuration repository for your SSM documents, and share documents across your organization (p. 1361).
- For information about the differences between State Manager and Maintenance Windows, see [Choosing between State Manager and Maintenance Windows \(p. 1554\)](#).

## Managed nodes (p. 804)

- Systems Manager requires accurate time references to perform its operations. If your node's date and time aren't set correctly, they might not match the signature date of your API requests. This might lead to errors or incomplete functionality. For example, nodes with incorrect time settings won't be included in your lists of managed nodes.

For information about setting the time on your nodes, see the following topics:

- [Set the time for your Linux instance](#)

- [Set the time for a Windows instance](#)

#### Related content

[Security best practices for Systems Manager \(p. 1424\)](#)

## Deleting Systems Manager resources and artifacts

As a best practice, we recommend that you delete Systems Manager resources and artifacts if you no longer need to view data about those resources or use the artifacts in any way. The following table lists each Systems Manager capability or artifact and a link to more information about deleting the resources or artifacts created by Systems Manager.

Capability or artifact	Details
Application Manager	You can't delete an application in Application Manager, but you can remove an application from the service by deleting the underlying <a href="#">tags</a> , <a href="#">Resource Groups</a> , or <a href="#">AWS CloudFormation stacks</a> .
Automation	If you create AWS resources by using Systems Manager Automation, you must manually delete those resources by using the corresponding AWS Management Console. If you created a custom runbook, you can delete the underlying SSM document. For more information, see <a href="#">Deleting custom SSM documents (p. 1360)</a> .
Change Calendar	You can delete a change calendar and a change calendar event. For more information, see <a href="#">Deleting a change calendar (p. 703)</a> and <a href="#">Deleting a Change Calendar event (p. 700)</a> .
Change Manager	You can delete a change template. For more information, see <a href="#">Deleting change templates (p. 383)</a> .
Compliance	Systems Manager Compliance automatically displays compliance data about Patch Manager patching and State Manager associations. You can't delete this data. If you configured a resource data sync to centralize compliance data in an Amazon S3 bucket, you can delete the sync. For more information, see <a href="#">Deleting a resource data sync for Compliance (p. 842)</a> .
Distributor	You can delete packages in Distributor. For more information, see <a href="#">Delete a package (p. 1281)</a> .
Explorer	You can disconnect from the sources from which Explorer collects OpsData. For more information, see <a href="#">Editing Systems Manager Explorer data sources (p. 172)</a> .  You can also delete a resource data sync used by Explorer to aggregate OpsData and OpsItems

Capability or artifact	Details
	from multiple AWS Regions and accounts to a single Amazon Simple Storage Service (Amazon S3) bucket. For more information, see <a href="#">Deleting a Systems Manager Explorer resource data sync (p. 175)</a> . For information about deleting an Amazon S3 bucket, see <a href="#">Deleting a bucket</a> in the <a href="#">Amazon Simple Email Service Developer Guide</a> .
Fleet Manager	You can't delete a managed node by using Fleet Manager. You must use Amazon Elastic Compute Cloud (Amazon EC2). For more information, see <a href="#">Terminate your instance (Linux)</a> and <a href="#">Terminate your instance (Windows)</a> .
Inventory	<p>You can stop Inventory data collection by deleting the State Manager associations that define the schedule and the resources from which to collect metadata. For more information, see <a href="#">Stopping data collection and deleting inventory data (p. 896)</a>.</p> <p>If you no longer want to use AWS Systems Manager Inventory to view metadata about your AWS resources, then we also recommend deleting resource data syncs used for inventory data collection. For more information, see <a href="#">Deleting an Inventory resource data sync (p. 897)</a>.</p>
Maintenance Windows	You can delete a maintenance window, a maintenance window target, and a maintenance window task. For more information, see <a href="#">Updating or deleting maintenance window resources (console) (p. 730)</a> .
OpsCenter	You can't manually delete OpsItems from OpsCenter, but you can set the status of one or more OpsItems to <b>Resolved</b> . Systems Manager automatically archives OpsItems after 36 months. For information about changing the status of an OpsItem see, <a href="#">Editing OpsItems (p. 211)</a> . For information about bulk resolving OpsItems by using OpsCenter Operational Insights, see <a href="#">Resolving duplicate OpsItems based on insights (p. 216)</a> .
Parameter Store	You can delete a parameter that you have created. For more information, see <a href="#">Deleting Systems Manager parameters (p. 295)</a> .
Patch Manager	You can delete a custom patch baseline. For more information, see <a href="#">Updating or deleting a custom patch baseline (console) (p. 1202)</a> .

Capability or artifact	Details
Quick Setup	You can delete associations created by Quick Setup. The associations are stored and processed by State Manager. For more information, see <a href="#">Deleting an association (p. 1058)</a> .
Run Command	After a command finishes processing, information about it is stored in the <b>Command history</b> tab. You can't delete information from the <b>Command history</b> tab.
Service-linked role	Systems Manager automatically creates service-linked roles <a href="#">for some capabilities (p. 1410)</a> . You can delete these roles. For more information, see <a href="#">Deleting the AWSServiceRoleForAmazonSSM service-linked role for Systems Manager (p. 1411)</a> .
Session Manager	Session Manager doesn't retain data about your resources after you terminate a session. To terminate a session, see <a href="#">End a session (p. 975)</a> .
SSM Agent	You can manually uninstall SSM Agent from your nodes. For more information, see the following topics. <ul style="list-style-type: none"> <li>• Linux: <a href="#">Uninstalling SSM Agent from Linux instances (p. 120)</a></li> <li>• macOS: <a href="#">Uninstall SSM Agent from macOS instances (p. 122)</a></li> <li>• Windows Server: Open <b>Control panel</b> and then choose <b>Add/remove programs</b>.</li> </ul>
State Manager	You can delete an association. For more information, see <a href="#">Deleting an association (p. 1058)</a> .
Systems Manager document service	You can't delete runbooks provided by AWS or AWS Support, but you can delete custom runbooks. For more information, see <a href="#">Deleting custom SSM documents (p. 1360)</a> .

## Choosing between State Manager and Maintenance Windows

State Manager and Maintenance Windows, both capabilities of AWS Systems Manager, can perform some similar types of updates on your managed nodes. Which one you choose depends on whether you need to automate system compliance or perform high-priority, time-sensitive tasks during periods you specify.

### State Manager and Maintenance Windows: Key use cases

State Manager, a capability of AWS Systems Manager, sets and maintains the desired state configuration for managed nodes and AWS resources within your AWS account. You can define combinations of

configurations and targets as association objects. State Manager is the recommended capability if you want to maintain all managed nodes in your account in a consistent state, use Amazon EC2 Auto Scaling to generate new nodes, or have strict compliance reporting requirements for the managed nodes in your account.

The main use cases for State Manager are as follows:

- **Auto Scaling scenarios:** State Manager can monitor all new nodes launched within an account either manually or through Auto Scaling groups. If there are any associations in the account targeting that new node (through tags or all nodes), then that particular association is automatically applied to the new node.
- **Compliance reporting:** State Manager can drive compliance reporting of desired states for resources in your account.
- **Supporting all nodes:** State Manager can target all nodes within a given account.

A **maintenance window** takes one or more actions on AWS resources within a given time window. You can define a single maintenance window with start and end times. You can specify multiple tasks to run within this maintenance window. Use Maintenance Windows, a capability of AWS Systems Manager, if your high priority operations include patching your managed nodes, running multiple types of tasks on your nodes during an update period, or controlling when update operations can be run on your nodes.

The main use cases for Maintenance Windows are as follows:

- **Running multiple documents:** Maintenance windows can run multiple tasks. Each task can use a different document type. As a result, you can build complex workflows using different tasks within a single maintenance window.
- **Patching:** A maintenance window can provide patching support for all managed nodes within an account tagged with a specific tag or resource group. Because patching usually involves bringing down nodes (for example, removing nodes from a load balancer), patching, and post processing (putting nodes back into production), patching can be achieved as a series of tasks within a given patch time window.
- **Window actions:** Maintenance windows can make one or more sets of actions start within a specific time window. Maintenance windows won't start outside of that window. Actions already started continue until finished, even if they finish outside of the time window.

The following table compares the main features of State Manager and Maintenance Windows.

Feature	State Manager	Maintenance Windows
<b>AWS CloudFormation integration</b>	AWS CloudFormation templates support State Manager associations.	AWS CloudFormation templates support maintenance windows, window targets, and window tasks.
<b>Compliance</b>	Every State Manager association reports compliance with respect to the desired state of the targeted resource. You can use the Compliance Dashboard to aggregate and view the reported compliance.	Not applicable.
<b>Configuration Management integration</b>	State Manager supports external desired state solutions such as Microsoft PowerShell Desired State Configuration	Not applicable.

Feature	State Manager	Maintenance Windows
	(DSC), Ansible playbooks, and Chef recipes. You can use State Manager associations to test that the Configuration Management solutions work and to apply their configuration changes to your nodes when you're ready.	
<b>Documents</b>	State Manager configurations can be defined as Policy documents (for gathering inventory information), Automation runbooks, for AWS resources such as Amazon Simple Storage Service (Amazon S3) buckets, or Systems Manager Command documents (SSM documents) for managed nodes.	Maintenance Windows configurations can be defined as automation documents (multi-step actions with optional approval workflows) or SSM documents (desired state for managed nodes).
<b>Monitoring</b>	State Manager monitors changes in the configuration, association, or state of a node (for example, new nodes coming online). When State Manager detects these changes, the given association is re-applied to the nodes originally targeted with that association.	Not applicable.
<b>Priorities within tasks</b>	Not applicable.	Tasks within a maintenance window can be assigned a priority. All tasks with the same priority are run in parallel. Tasks with lower priorities are run after tasks with higher priorities reach a final state. There is no way to conditionally run tasks. After a higher priority task reaches its final state, the next priority task runs, regardless of the state of the previous task.

Feature	State Manager	Maintenance Windows
<b>Safety controls</b>	State Manager supports two safety controls when deploying configurations across a large fleet. You can use maximum concurrency to define how many concurrent nodes or resources should have the configuration applied. You can define a maximum error rate which can be used to pause the State Manager association if a certain number or percentage of errors occur across the fleet.	Maintenance windows support two safety controls when deploying configurations across a large fleet. You can use maximum concurrency to define how many concurrent nodes or resources should have the configuration applied. You can define a maximum error rate which can be used to pause the actions in a maintenance window if a certain number or percentage of errors occur across the fleet.
<b>Scheduling</b>	You can run State Manager associations on demand, at a particular cron interval, at a given rate, or once when they're created. This is useful if you want to maintain the desired state of your resources in a consistent and timely manner.	Maintenance windows support several scheduling options including <code>at</code> expressions (for example, <code>"at(2021-07-07T13:15:30)"</code> ), cron and rate expressions, cron with offsets, and start and end times for when maintenance windows should run, and cutoff times to specify when to stop scheduling within a given time window.
<b>Targeting</b>	State Manager associations can target one or more nodes by using node ID, tag, or resource group. State Manager can target all managed nodes within a given account.	Maintenance windows can target one or more nodes using node IDs, tags, or resource groups.
<b>Tasks within maintenance windows</b>	Not applicable.	<p>Maintenance windows can support one or more tasks where each task targets a specific Automation runbook or Command document action. All tasks within a maintenance window run in parallel unless different priorities are set for different tasks.</p> <p>Overall, maintenance windows support four task types:</p> <ul style="list-style-type: none"> <li>• AWS Systems Manager Run Command commands</li> <li>• AWS Systems Manager Automation workflows</li> <li>• AWS Lambda functions</li> <li>• AWS Step Functions tasks</li> </ul>

# 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. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager capabilities and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For information, see [Automating updates to SSM Agent \(p. 135\)](#). Subscribe to the [SSM Agent Release Notes](#) page on GitHub to get notifications about SSM Agent updates.

- **API version:** 2014-11-06

update-history-change	update-history-description	update-history-date
<a href="#">Port forwarding to remote hosts support for Session Manager (p. 1558)</a>	Session Manager now supports port forwarding sessions to remote hosts. The remote host isn't required to be managed by Systems Manager. For more information, see <a href="#">Starting a session (port forwarding to remote host)</a> .	May 25, 2022
<a href="#">Updated content: Instructions for manually installing SSM Agent on Amazon EC2 Linux instances (p. 1558)</a>	In response to customer feedback, we have overhauled the topics that provide instructions for manually installing SSM Agent on Amazon EC2 instances. These topics now provide commands using globally available files that you can copy and paste for quick installation on EC2 instances in any AWS Region. These topics also provide information to help you creating installation commands that use files available in your own working Region. The latter approach is recommended when you are installing the agent on multiple instances using a script or template. For more information, see the instructions for your Linux operating system in the section <a href="#">Manually installing SSM Agent on EC2 instances for Linux</a> .	May 9, 2022

New topic: Amazon Machine Images (AMIs) with SSM Agent preinstalled (p. 1558)	In response to customer feedback, we have centralized information about which AWS managed AMIs include SSM Agent preinstalled. This topic also provides instructions for how to verify that an Amazon EC2 instance created from these AMIs was successfully installed and is running. For rare cases where the agent might not install successfully, or install but not start, we also provide information about starting or manually installing the agent on these instances. For details, see <a href="#">Amazon Machine Images (AMIs) with SSM Agent preinstalled</a> .	May 8, 2022
New State Manager section (p. 1558)	Added a new section that describes the details of when State Manager runs associations. For more information, see <a href="#">About association scheduling</a> .	April 27, 2022
Patch Manager now supports Rocky Linux (p. 1558)	You can now use Patch Manager to patch Rocky Linux nodes. Many of the rules that apply to RHEL 8 for patching also apply to Rocky Linux. Rocky Linux 8 uses the new <code>AWS-DefaultRockyLinux PatchBaseline</code> . For more information, see the following topics: <ul style="list-style-type: none"><li>• <a href="#">How security patches are selected</a></li><li>• <a href="#">How patches are installed</a></li><li>• <a href="#">How patch baseline rules work on RHEL, CentOS Stream, and Rocky Linux</a>.</li></ul>	April 14, 2022

Patch Manager now supports CentOS Stream 8 (p. 1558)

You can now use Patch Manager to patch CentOS Stream 8 instances and Red Hat Enterprise Linux (RHEL) 4.4-4.5 instances. Many of the rules that apply to RHEL 8 for patching also apply to CentOS Stream 8. CentOS Stream 8 uses the `AWS-DefaultCentOSPatchBaseline`. For more information, see the following topics:

- [How security patches are selected](#)
- [How patches are installed](#)
- [How patch baseline rules work on RHEL and CentOS Stream](#)

April 4, 2022

Create an assume role for Change Manager (p. 1558)

A new section clarifies the requirements for creating and implementing an *assume role* for Change Manager. An assume role is an AWS Identity and Access Management (IAM) service role that enables Change Manager to securely run the runbook workflows specified in an approved change request on your behalf. The role grants AWS Systems Manager (AWS STS) `AssumeRole` trust to Change Manager. For information, see [Configuring roles and permissions for Change Manager](#).

March 18, 2022

Approve or reject Change Manager change requests in bulk (p. 1558)

In the Systems Manager console, you can now select multiple change requests to approve or reject in a single operation. For information, see [Reviewing and approving or rejecting change requests \(console\)](#).

March 8, 2022

Support for Rocky Linux and Windows Server 2022 managed nodes (p. 1558)	Systems Manager supports Rocky Linux and Windows Server 2022 managed nodes, including edge devices and hybrid machines located on-premises or with other cloud providers. To use Systems Manager with these operating systems, you must complete all required Systems Manager set up procedures, including procedures for hybrid environments or edge devices, if applicable. For more information, see <a href="#">Setting up Systems Manager</a> . For Rocky Linux machines, you must also manually install SSM Agent. For more information, see <a href="#">Manually install SSM Agent on Rocky Linux instances</a> . For Windows Server 2022 Amazon Elastic Compute Cloud (Amazon EC2) instances, SSM Agent is installed by default.	March 1, 2022
Allow Automation to adapt to your concurrency needs and view Automation usage metrics (p. 1558)	You can now allow Automation to automatically adjust your concurrent automation quota, and view Automation usage metrics that are published to CloudWatch. For more information about adaptive concurrency, see <a href="#">Allowing Automation to adapt to your concurrency needs</a> . For more information about how to view Automation usage metrics, see <a href="#">Monitoring Automation metrics using Amazon CloudWatch</a> .	January 27, 2022
Allow Automation to adapt to your concurrency needs and view Automation usage metrics (p. 1558)	You can now allow Automation to automatically adjust your concurrent automation quota, and view Automation usage metrics that are published to CloudWatch. For more information about adaptive concurrency, see <a href="#">Allowing Automation to adapt to your concurrency needs</a> . For more information about how to view Automation usage metrics, see <a href="#">Monitoring Automation metrics using Amazon CloudWatch</a> .	January 27, 2022

Systems Manager documents organized by categories (p. 1558)	Amazon owned Systems Manager documents are now organized by type and categories to help you find the documents you need.	January 13, 2022
Create and invoke integrations for Automation (p. 1558)	You can now send messages using webhooks during an automation by creating an integration. Integrations can be invoked during an automation using the new <code>aws:invokeWebhook</code> action in your runbook. For more information about creating integrations, see <a href="#">Creating integrations for Automation</a> . To learn more about the <code>aws:invokeWebhook</code> action, see <a href="#">aws:invokeWebhook – Invoke an Automation webhook integration</a> .	January 13, 2022
Capabilities not available in new AWS Region (p. 1558)	The following Systems Manager capabilities currently aren't available in the new Asia Pacific (Jakarta) Region.	December 13, 2021
View resource cost details for an application (p. 1558)	<ul style="list-style-type: none"><li>• Application Manager</li><li>• Change Calendar</li><li>• Change Manager</li><li>• Explorer</li><li>• Fleet Manager</li><li>• Incident Manager</li><li>• Quick Setup</li></ul> <p>Application Manager is integrated with AWS Billing and Cost Management through the <a href="#">Cost Explorer</a> widget. After you enable Cost Explorer in the Billing and Cost Management console, the Cost Explorer widget in Application Manager shows cost data for a specific non-container application or application component. You can use filters in the widget to view cost data according to different time periods, granularities, and cost types in either a bar or line chart. For more information, see <a href="#">Viewing overview information about an application</a>.</p>	December 7, 2021

Manage processes using Fleet Manager (p. 1558)	You can now use Fleet Manager to manage processes on your nodes. For more information, see <a href="#">Working with processes</a> .	December 6, 2021
Terminology change: managed instances are now managed nodes (p. 1558)	With support for AWS IoT Greengrass core devices, the phrase <i>managed instance</i> has been changed to <i>managed node</i> in most of the Systems Manager documentation. The Systems Manager console, API calls, error messages, and SSM documents still use the term instance.	November 29, 2021
Support for edge devices (p. 1558)	<p>Systems Manager supports the following edge device configurations.</p> <ul style="list-style-type: none"><li><b>AWS IoT Greengrass:</b> Systems Manager now supports any device that is configured for AWS IoT Greengrass and runs the AWS IoT Greengrass Core software. To onboard your AWS IoT Greengrass core devices, you must create an AWS Identity and Access Management (IAM) service role. You must also use the AWS IoT Greengrass console to deploy SSM Agent as a AWS IoT Greengrass component on your devices. For more information, see <a href="#">Setting up AWS Systems Manager for edge devices</a>.</li><li><b>Edge devices in a hybrid environment:</b> Systems Manager also supports AWS IoT Core devices and non-AWS IoT devices after you configure them as on-premises machines. To onboard your devices, you must create an IAM service role, create a managed-node activation for a hybrid environment, and manually install SSM Agent on your devices. For more information, see <a href="#">Setting up AWS Systems Manager for hybrid environments</a></li></ul>	November 29, 2021

<a href="#">Connect to managed instances using Remote Desktop (p. 1558)</a>	You can now use Fleet Manager to connect to managed Windows instances using the Remote Desktop Protocol (RDP). These Remote Desktop sessions powered by NICE DCV provide secure connections to your instances directly from your browser. For more information, see <a href="#">Connect using Remote Desktop</a> .	November 23, 2021
<a href="#">Specify a maximum session duration and provide reasons for sessions (p. 1558)</a>	You can now specify a maximum session duration for all Session Manager sessions in an AWS Region in your AWS account. When a session reaches reaches the duration you specify, it's terminated. You can now also optionally add a reason when starting a session. For more information, see <a href="#">Specify maximum session duration</a> .	November 16, 2021
<a href="#">Patch Manager now supports the Raspberry Pi OS operating system (p. 1558)</a>	You can now use Patch Manager to patch Raspberry Pi OS instances. Patch Manager supports patching Raspberry Pi OS 9 (Stretch) and 10 (Buster). Because the Raspberry Pi OS is Debian-based OS, many of the same patching rules apply to it as to Debian Server. For more information, see the following topics: <ul style="list-style-type: none"><li>• <a href="#">How security patches are selected (p. 1097)</a></li><li>• <a href="#">How patches are installed (p. 1103)</a></li><li>• <a href="#">How patch baseline rules work on Debian Server and Raspberry Pi OS (p. 1115)</a></li></ul>	November 16, 2021
<a href="#">Access the Red Hat Knowledgebase portal (p. 1558)</a>	Use Fleet Manager to access the RHEL Knowledgebase portal to find solutions, articles, documentation, and videos about using Red Hat products. For more information, see <a href="#">Accessing the Red Hat Knowledgebase portal</a> .	November 3, 2021

Bulk edit OpsItems (p. 1558)	OpsCenter now supports bulk editing OpsItems. You can select multiple OpsItems and edit one of the following fields: <b>Status</b> , <b>Priority</b> , <b>Severity</b> , <b>Category</b> . For more information, see <a href="#">Editing OpsItems</a> .	October 15, 2021
Create input parameters that populate AWS resources (p. 1558)	You can now create input parameters in Automation runbooks that populate AWS resources in AWS Management Console. For information, see <a href="#">Creating input parameters that populate AWS resources</a> .	October 14, 2021
New task invocation cutoff option for maintenance windows (p. 1558)	You can now choose to block any new task invocations from starting after the cutoff time specified for a maintenance window is reached. For information, see <a href="#">Assign tasks to a maintenance window (console)</a> .	October 13, 2021
Patch Manager support for macOS 11.3.1 and 11.4 (Big Sur) (p. 1558)	Amazon Elastic Compute Cloud (Amazon EC2) instances for macOS 11.3.1 and 11.4 (Big Sur) can now be patched using Patch Manager. This is in addition to existing support for macOS 10.14.x (Mojave) and 10.15.x (Catalina). For information about working with Patch Manager, see <a href="#">AWS Systems Manager Patch Manager</a> .	October 1, 2021

<a href="#">Application insights in Application Manager (p. 1558)</a>	Application Manager integrates with Amazon CloudWatch Application Insights. Application Insights identifies and sets up key metrics, logs, and alarms across your application resources and technology stack. Application Insights continuously monitors metrics and logs to detect and correlate anomalies and errors. When the system detects errors or anomalies, Application Insights generates CloudWatch Events that you can use to set up notifications or take actions. You can enable and view Application Insights on the <b>Overview</b> and <b>Monitoring</b> tabs in Application Manager. For more information about Application Insights, see <a href="#">What is Amazon CloudWatch Application Insights</a> in the <i>Amazon CloudWatch User Guide</i> .	September 21, 2021
<a href="#">Import events from other calendars into Change Calendar (p. 1558)</a>	You can now import the events from a third-party calendar into a calendar in Change Calendar. Previously, each event had to be entered manually into a calendar. After you export a calendar from a supported third-party calendar provider to an iCalendar (.ics) file, import it into Change Calendar, and its events are included in the rules for your open or closed calendar in Systems Manager. Supported providers include iCloud Calendar, Google Calendar, and Microsoft Outlook. For more information, see <a href="#">Importing and managing events from third-party calendars</a> .	September 8, 2021

New tagging and runbook features in Application Manager (p. 1558)	Tagging enhancements include the ability to add tags to or delete tags from a specific resource or all resources in an Application Manager application. Runbook enhancements include the ability to view a filtered list of runbooks for a specific resource type or initiate a runbook on all resources of the same type. For more information, see <a href="#">Working with tags in Application Manager</a> and <a href="#">Working with runbooks in Application Manager</a> .	August 31, 2021
New example: Create a change request using the AWS CLI (p. 1558)	An example of creating a change request with the AWS CLI has been added to the Change Manager chapter. The example uses the sample AWS-HelloWorldChangeTemplate change template and AWS-HelloWorld runbook: <ul style="list-style-type: none"><li>• <a href="#">Creating change requests (AWS CLI)</a></li></ul>	August 20, 2021
New section: Use parameters in Amazon EKS (p. 1558)	A new section has been added to the Parameter Store chapter. This topic is a walkthrough on how to use your parameters in Amazon EKS clusters. For more information, see <a href="#">Use Parameter Store parameters in Amazon Elastic Kubernetes Service</a> .	August 19, 2021
Updated Patch Manager lifecycle hooks (p. 1558)	Patch Manager now provides a lifecycle hook—the ability to run a Systems Manager Command document—for an additional point during a <b>Patch now</b> patching operation. If you schedule instance reboots after running <b>Patch now</b> , you can specify a lifecycle hook to run after the reboot completes. For more information, see <a href="#">Using 'Patch now' lifecycle hooks</a> and <a href="#">About the AWS-RunPatchBaselineWithHooks SSM document</a> .	August 9, 2021

<a href="#">OpsCenter operational insights (p. 1558)</a>	OpsCenter automatically analyzes OpsItems in your account and generates <i>insights</i> . An insight includes information to help you understand how many duplicate OpsItems are in your account and which sources are creating them. Insights also provide recommended best practices and Automation runbooks to help you resolve duplicate OpsItems. For more information, see <a href="#">Working with operational insights</a> .	July 13, 2021
<a href="#">View stopped instances in Fleet Manager (p. 1558)</a>	You can now view which instances are running and which instances are stopped from the Fleet Manager console. For more information, see <a href="#">AWS Systems Manager Fleet Manager</a> .	July 12, 2021
<a href="#">New topic: Authoring Automation runbooks (p. 1558)</a>	A new topic, <a href="#">Authoring Automation runbooks</a> , provides guidance and narrative examples of how to author content for custom Automation runbooks.	July 8, 2021
<a href="#">AWS CloudFormation stack and template creation in Application Manager (p. 1558)</a>	Application Manager helps you provision and manage resources for your applications by integrating with <a href="#">CloudFormation</a> . You can create, edit, and delete AWS CloudFormation templates and stacks in Application Manager. Application Manager also includes a template library where you can clone, create, and store templates. Application Manager and CloudFormation display the same information about the current status of a stack. Templates and template updates are stored in Systems Manager until you provision the stack, at which time the changes are also displayed in CloudFormation. For more information, see <a href="#">Working with AWS CloudFormation Stacks in Application Manager</a> .	July 8, 2021

New topic: Automatically rotate private keys for SSM Agent on hybrid instances (p. 1558)	A new topic, <a href="#">Setting up private key auto rotation</a> , provides instructions on how to strengthen your security posture by configuring SSM Agent to rotate the hybrid environment private key automatically.	June 15, 2021
Session Manager plugin for the AWS CLI version 1.2.205.0 (p. 1558)	A new version of the Session Manager plugin for the AWS CLI has been released. For more information, see <a href="#">Session Manager plugin latest version and release history</a> .	June 10, 2021
New IAM service-linked role (p. 1558)	When you enable OpsCenter operational insights, Systems Manager creates a new AWS Identity and Access Management (IAM) service-linked role called <code>AWSSSMOpsInsightsServiceRolePolicy</code> . For more information about this role, see <a href="#">Using roles to create operational insight OpsItems in Systems Manager OpsCenter: AWSSSMOpsInsightsServiceRolePolicy</a> .	June 9, 2021
New Patch Manager troubleshooting content for Linux (p. 1558)	A new topic, <a href="#">Errors when running AWS-RunPatchBaseline on Linux</a> , provides descriptions and solutions for several issues that might be encountered when patching managed instances with Linux operating systems.	June 8, 2021

Improved support for maintenance window tasks that don't require specified targets (console) (p. 1558)	You can now create maintenance window tasks in the console without having to specify a target in the task if one isn't required. Previously, this option was available only when using the AWS CLI or API. This option applies to Automation, AWS Lambda, and AWS Step Functions task types. For example, if you create an Automation task and the resources to update are specified in the Automation document parameters, you no longer need to specify a target in the task itself. For more information, see <a href="#">Registering maintenance window tasks without targets</a> , <a href="#">Assign tasks to a maintenance window (console)</a> , and <a href="#">Running automations with triggers using a maintenance window</a> .	May 28, 2021
Automation runbook reference relocated (p. 1558)	The Automation runbook reference has been moved to a new location. For more information, see <a href="#">Systems Manager Automation runbook reference</a> .	May 10, 2021
AWS Systems Manager Incident Manager launch (p. 1558)	Incident Manager is an incident management console designed to help users mitigate and recover from incidents affecting their AWS hosted applications. For more information, see the <a href="#">Incident Manager User Guide</a> .	May 10, 2021
State Manager supports Change Calendar (p. 1558)	You can now specify Change Calendar names or Amazon Resource Names (ARNs) when you create or update a State Manager association. State Manager applies associations only when the change calendar is open, not when it's closed. For more information, see <a href="#">Creating associations</a> and <a href="#">Editing and creating a new version of an association</a> .	May 6, 2021

Clone Systems Manager documents (p. 1558)	Using the Systems Manager Documents console, you can now copy content from an existing document to a new document that you can modify. To learn more, see <a href="#">Cloning an SSM document</a> .	May 4, 2021
Integrate Security Hub with Explorer and OpsCenter (p. 1558)	You can now integrate Explorer and OpsCenter with AWS Security Hub. Security Hub provides a comprehensive view of your security state in AWS and helps you check your environment against security industry standards and best practices. When integrated with Explorer, you can view security findings in the Security Hub widget on the Explorer dashboard. When integrated with OpsCenter, you can create OpsItems for Security Hub findings. For more information, see <a href="#">Receiving findings from AWS Security Hub in Explorer</a> and <a href="#">Receiving findings from AWS Security Hub in OpsCenter</a> .	April 27, 2021
New topic: Document conventions (p. 1558)	We've added a new topic to help users understand the common typographical conventions for the <i>AWS Systems Manager User Guide</i> . For more information, see <a href="#">Document conventions</a> .	April 21, 2021
Updated topic: About patching applications released by Microsoft on Windows Server (p. 1558)	The topic <a href="#">About patching applications released by Microsoft on Windows Server</a> now clarifies that, in order for Patch Manager to be able to patch applications released by Microsoft on your Windows Server managed instances, the Windows update option <b>Give me updates for other Microsoft products when I update Windows</b> must be allowed on the instance.	April 12, 2021

<a href="#">Automation runbook reference reorganization (p. 1558)</a>	To help you find the runbooks you need and navigate the reference more efficiently, we reorganized the content in the Automation runbook reference by the relevant AWS service. To view these changes, see <a href="#">Systems Manager Automation runbook reference</a> .	April 12, 2021
<a href="#">Patch Manager: Generate .csv patch compliance reports (p. 1558)</a>	Patch Manager now supports the ability to generate patch compliance reports for your instances and save the report in an Amazon S3 bucket of your choice, in .csv format. Then, using a tool like <a href="#">Amazon QuickSight</a> , you can analyze the patch compliance report data. You can generate a patch compliance report for a single instance, or for all instances in your AWS account. You can generate a one-time report on demand, or set up a schedule for reports to be created automatically. You can also specify an Amazon Simple Notification Service topic to provide notifications when a report is generated. For more information, see <a href="#">Generating CSV patch compliance reports</a> .	April 9, 2021
<a href="#">Delete Parameter Store parameter labels (p. 1558)</a>	You can now delete Parameter Store parameter labels by using either the Systems Manager console or the AWS CLI. For more information, see <a href="#">Working with parameter labels</a> .	April 6, 2021
<a href="#">Schedule instance reboots when using Patch Now (p. 1558)</a>	Patch Manager now supports scheduling a time for your instances to reboot after patches are installed using the Patch Now feature. This is in addition to existing options to reboot instances only if needed to complete a patch installation or to skip all rebooting after the patching operation. For information, see <a href="#">Patching instances on demand</a> .	April 1, 2021

New topic: Discover public parameters (p. 1558)	Parameter Store public parameters can now be found using the AWS CLI or Systems Manager console. For more information, see <a href="#">Finding public parameters</a> .	April 1, 2021
Patch now updates: Store logs in S3 & and run lifecycle hooks (p. 1558)	When you run the Patch Manager <b>Patch now</b> operation, you can choose an S3 bucket in which to automatically store patching logs. In addition, you can choose to run Systems Manager Command documents (SSM documents) as lifecycle hooks at three points during the operation: <i>Before installation</i> , <i>After installation</i> , and <i>On exit</i> . For more information, see <a href="#">Patching instances on demand</a> .	March 31, 2021
Systems Manager now reports changes to its AWS managed policies (p. 1558)	Beginning March 24, 2021, changes to managed policies are reported in the topic <a href="#">Systems Manager updates to AWS managed policies</a> . The first change listed is the addition of support for the Explorer capability to report OpsData and OpsItems from multiple accounts and Regions.	March 24, 2021
Explorer automatically allows all OpsData sources for resource data syncs based on accounts in AWS Organizations (p. 1558)	When you create a resource data sync, if you choose one of the AWS Organizations options, Systems Manager automatically allows all OpsData sources in the selected AWS Regions for all AWS accounts in your organization (or in the selected organization units). This means, for example, that even if you haven't allowed Explorer in an AWS Region, if you select an AWS Organizations option for your resource data sync, then Systems Manager automatically collects OpsData from that Region. For more information, see <a href="#">About multiple account and Region resource data syncs</a> .	March 24, 2021

Systems Manager Automation provides a new system variable for your runbooks (p. 1558)	With the new <code>global:AWS_PARTITION</code> system variable, you can specify the AWS partition a resource is located in when authoring your runbooks. For more information, see <a href="#">Automation system variables</a> .	March 18, 2021
Allow multiple levels of approval for Change Manager change requests (p. 1558)	When you create a Change Manager change template, you can now require that more than one level of approvers grant permission for a change request to run. For example, you might require technical reviewers to approve a change request created from a change template first, and then require a second level of approvals from one or more managers. For more information, see <a href="#">Creating change templates</a> .	March 4, 2021
Patch Manager now supports Oracle Linux 8.x (p. 1558)	You can now use Patch Manager to patch Oracle Linux 8.x instances, through version 8.3. For more information, see the following topics: <ul style="list-style-type: none"><li>• <a href="#">How security patches are selected</a></li><li>• <a href="#">How patches are installed</a></li><li>• <a href="#">How patch baseline rules work on Oracle Linux</a></li></ul>	March 1, 2021
OpsCenter displays other OpsItems for a selected resource (p. 1558)	To help you investigate issues and provide context for a problem, you can view a list of OpsItems for a specific AWS resource. The list displays the status, severity, and title of each OpsItem. The list also includes deep links to each OpsItem. For more information, see <a href="#">Viewing other OpsItems for a specific resource</a> .	March 1, 2021
Define patching preferences at runtime (p. 1558)	You can now define patching preferences at runtime using the baseline override feature. For more information, see <a href="#">Using the BaselineOverride parameter</a> .	February 25, 2021

New Systems Manager document type (p. 1558)	AWS CloudFormation templates can now be stored as Systems Manager documents. Storing CloudFormation templates as Systems Manager documents allows you to benefit from Systems Manager document features like versioning, comparing version content, and sharing with accounts. For more information, see <a href="#">AWS Systems Manager documents</a> .	February 9, 2021
Patch instances using optional hooks (p. 1558)	The new SSM document <code>AWS-RunPatchBaselineWithHooks</code> provides hooks you can use to run SSM documents at three points during the instance patching cycle. For information about <code>AWS-RunPatchBaselineWithHooks</code> , see <a href="#">About the AWS-RunPatchBaselineWithHooks SSM document</a> . For a sample walkthrough of a patching operation that uses all three hooks, see <a href="#">Walkthrough: Update application dependencies, patch an instance, and perform an application-specific health check</a> .	February 2, 2021
New topic: Validating on-premises servers and virtual machines using a hardware fingerprint (p. 1558)	SSM Agent verifies the identity of on-premises servers and virtual machines and VMs that you register with the service by using a computed <i>fingerprint</i> . The fingerprint is an opaque string, stored in the <i>Vault</i> that the agent passes to certain Systems Manager APIs. For information about the hardware fingerprint and instructions for configuring a <i>similarity threshold</i> to assist in machine verification, see <a href="#">Validating on-premises servers and virtual machines using a hardware fingerprint</a> .	January 25, 2021

New topic: SSM Agent technical reference (p. 1558)	The topic <a href="#">SSM Agent technical reference</a> brings together information to help you implement AWS Systems Manager SSM Agent and understand how the agent works. This topic includes an all-new section, <a href="#">SSM Agent rolling updates by AWS Regions</a> .	January 21, 2021
<a href="#">SSM Agent on Windows Server 2008 (p. 1558)</a>	As of January 14, 2020, Windows Server 2008 is no longer supported for feature or security updates from Microsoft. Windows Server 2008 AMIs do include SSM Agent, but the agent is no longer updated for this operating system.	January 5, 2021
<a href="#">Improved support for maintenance window tasks that don't require specified targets (AWS CLI and API only) (p. 1558)</a>	You can now create maintenance window tasks without having specify a target in the task if one isn't required (AWS CLI and API only). This applies to Automation, AWS Lambda and AWS Step Functions task types. For example, if you create an Automation task and the resources to update are specified in the Automation runbook parameters, you no longer need to specify a target in the task itself. For more information, see <a href="#">Registering maintenance window tasks without targets</a> and <a href="#">Running automations with triggers using a maintenance window</a> .	December 23, 2020
<a href="#">New Automation features (p. 1558)</a>	A new shared property has been added to Systems Manager Automation runbooks. The <code>onCancel</code> property allows you to specify which step the automation should go to in the event that a user cancels the automation. For more information, see <a href="#">Properties shared by all actions</a> .	December 21, 2020
<a href="#">New topic: Working with associations using IAM (p. 1558)</a>	A new topic has been added to the Systems Manager State Manager chapter that describes the best practices for creating associations using IAM. For more information, see <a href="#">Working with associations using IAM</a> .	December 18, 2020

[State Manager now supports multi-regions and multi-accounts \(p. 1558\)](#) Associations can now be created or updated with multiple regions or accounts. For more information, see [Creating associations](#). December 15, 2020

[New capability: Fleet Manager \(p. 1558\)](#) Fleet Manager, a capability of AWS Systems Manager, is a unified user interface (UI) experience that helps you remotely manage your server fleet running on AWS, or on-premises. With Fleet Manager, you can view the health and performance status of your entire server fleet from one console. You can also gather data from individual instances to perform common troubleshooting and management tasks from the console. For information, see [AWS Systems ManagerFleet Manager](#). December 15, 2020

[New capability: Change Manager \(p. 1558\)](#) Amazon Web Services has released Change Manager, an enterprise change management framework for requesting, approving, implementing, and reporting on operational changes to your application configuration and infrastructure. From a single *delegated administrator account*, if you use AWS Organizations, you can manage changes across multiple AWS accounts in multiple AWS Regions. Alternatively, using a *local account*, you can manage changes for a single AWS account. Use Change Manager for managing changes to both AWS resources and on-premises resources. For information, see [AWS Systems ManagerChange Manager](#). December 15, 2020

New capability: Application Manager (p. 1558)	Application Manager helps you investigate and remediate issues with your AWS resources in the context of your applications. Application Manager aggregates operations information from multiple AWS services and Systems Manager capabilities to a single AWS Management console. For information, see <a href="#">AWS Systems Manager Application Manager</a> .	December 15, 2020
AWS Systems Manager supports Amazon EC2 instances for macOS (p. 1558)	In tandem with the release of Amazon Elastic Compute Cloud (Amazon EC2) support for macOS instances, Systems Manager now supports many operations on EC2 instances for macOS. Supported versions include macOS 10.14.x (Mojave) and 10.15.x (Catalina). For more information, see the following topics. <ul style="list-style-type: none"> <li>• For information about installing SSM Agent on EC2 instances for macOS, see <a href="#">Installing and configuring SSM Agent on EC2 instances for macOS..</a></li> <li>• For information about patching EC2 instances for macOS, see <a href="#">How patches are installed</a>, and <a href="#">Creating a custom patch baseline (macOS)</a>.</li> <li>• For general information about support for EC2 instances for macOS, see <a href="#">Amazon EC2 Mac instances</a> in the <a href="#">Amazon EC2 User Guide for Linux Instances</a>.</li> </ul>	November 30, 2020
Maintenance window pseudo parameters: New resource type supported for {{TARGET_ID}} and {{RESOURCE_ID}} (p. 1558)	An additional resource type is now available for use with the pseudo parameters {{TARGET_ID}} and {{RESOURCE_ID}}. You can now use the resource type <code>AWS::RDS::DBCluster</code> with both these pseudo parameters. For information about maintenance window pseudo parameters, see <a href="#">About pseudo parameters</a> .	November 27, 2020

<a href="#">Session Manager plugin for the AWS CLI version 1.2.30.0 (p. 1558)</a>	A new version of the Session Manager plugin for the AWS CLI has been released. For more information, see <a href="#">Session Manager plugin latest version and release history</a> .	November 24, 2020
<a href="#">New topic: Comparing SSM document versions (p. 1558)</a>	You can now compare the differences in content between versions of SSM documents in the Systems Manager Documents console. For more information, see <a href="#">Comparing SSM document versions</a> .	November 24, 2020
<a href="#">Systems Manager now supports VPC endpoint policies (p. 1558)</a>	You can now create policies for VPC interface endpoints for Systems Manager. For more information, see <a href="#">Create an interface VPC endpoint policy</a> .	November 18, 2020
<a href="#">New topic: Specify an idle session timeout value (p. 1558)</a>	You can now specify the amount of time to allow a user to be inactive before a session ends with Session Manager. For more information, see <a href="#">Specify an idle session timeout value</a> .	November 18, 2020
<a href="#">New Session Manager logging feature (p. 1558)</a>	You can now send a continual stream of JSON-formatted session data logs to Amazon CloudWatch Logs. For more information, see <a href="#">Streaming session data using Amazon CloudWatch Logs</a> .	November 18, 2020
<a href="#">New topic: Verify the signature of the SSM Agent (p. 1558)</a>	You can now verify the cryptographic signature of the installer package for the SSM Agent on Linux instances. For more information, see <a href="#">SSM document schemas and features</a> .	November 17, 2020
<a href="#">New topic: Understanding automation statuses (p. 1558)</a>	A new topic has been added to the Systems Manager Automation chapter that describes the statuses for actions and automations. For more information, see <a href="#">Understanding automation statuses</a> .	November 17, 2020
<a href="#">New source types for the aws:downloadContent plugin (p. 1558)</a>	Git and HTTP are now supported as source types for the aws:downloadContent plugin. For more information, see <a href="#">aws:downloadContent</a> .	November 17, 2020

New Systems Manager document (SSM document) schema feature (p. 1558)	In SSM documents with schema version 2.2 or later, the <code>precondition</code> parameter now supports referencing your document's input parameters. For more information, see <a href="#">SSM document schemas and features</a> .	November 17, 2020
New data source in Explorer: AWS Config (p. 1558)	Explorer now displays information about AWS Config compliance, including an overall summary of compliant and non-compliant AWS Config rules, the number of compliant and non-compliant resources, and specific details about each (when you drill down into a non-compliant rule or resource). For more information, see <a href="#">Editing Systems Manager Explorer data sources</a> .	November 11, 2020
New topic: Running Auto Scaling groups with associations (p. 1558)	A new section has been added to State Manager that describes the best practices for creating associations to run Auto Scaling groups. For more information, see <a href="#">Running Auto Scaling groups with associations</a> .	November 10, 2020
Quick Setup now supports targeting a resource group (p. 1558)	Quick Setup now supports choosing a resource group as a target for the local setup type. For more information, see <a href="#">Choosing Targets for Quick Setup</a> .	November 5, 2020
Patch Manager adds support for Debian Server 10 LTS, Oracle Linux 7.9 LTS, and Ubuntu Server 20.10 STR (p. 1558)	You can now use Patch Manager to patch Debian Server 10 LTS, Oracle Linux 7.9 LTS, and Ubuntu Server 20.10 STR instances. For more information, see the following topics: <ul style="list-style-type: none"><li>• <a href="#">Patch Manager prerequisites</a></li><li>• <a href="#">How security patches are selected</a></li><li>• <a href="#">How patches are installed</a></li><li>• <a href="#">How patch baseline rules work on Debian Server</a></li><li>• <a href="#">How patch baseline rules work on Oracle Linux</a></li><li>• <a href="#">How patch baseline rules work on Ubuntu Server</a></li></ul>	November 4, 2020

New EventBridge support for AWS Systems Manager Change Calendar (p. 1558)	Amazon EventBridge now provides support for Change Calendar events in event rules. When the state of a calendar changes, EventBridge can initiate the target action you defined in an EventBridge rule. For information about working with EventBridge and Systems Manager events, see the following topics.	November 4, 2020
Configure CloudWatch to create OpsItems from alarms (p. 1558)	You can configure Amazon CloudWatch to automatically create an OpsItem in Systems Manager OpsCenter when an alarm enters the ALARM state. Doing so allows you to quickly diagnose and remediate issues with AWS resources from a single console. For more information, see <a href="#">Configuring CloudWatch to create OpsItems from alarms</a> .	November 4, 2020
Support for Ubuntu Server 20.10 (p. 1558)	AWS Systems Manager now supports Ubuntu Server 20.10 short-term release (STR). For more information, see the following topics:	October 22, 2020

New topic: Allow configurable shell profiles (p. 1558)	You can now allow configurable shell profiles with Session Manager. By allowing configurable shell profiles, you can customize preferences within sessions such as shell preferences, environment variables, working directories, and running multiple commands when a session is started. For more information, see <a href="#">Allow configurable shell profiles</a> .	October 21, 2020
Patch compliance results now report which CVEs are resolved by which patches (p. 1558)	For most supported Linux systems, when you view patch compliance results for your managed instances, the details you can view now report which Common Vulnerabilities and Exposure (CVE) bulletin issues are resolved by which available patches. This information can help you determine how urgently you need to install a missing or failed patch. For more information, see <a href="#">Viewing patch compliance results</a> .	October 20, 2020
Expanded support for Linux patch metadata (p. 1558)	You can now view many details about available Linux patches in Patch Manager. You can choose to view patch data such as architecture, epoch, version, CVE ID, Advisory ID, Bugzilla ID, repository, and more. In addition, the <a href="#">DescribeAvailablePatches</a> API operation has been updated to support Linux operating systems and filtering according to these newly available patch metadata types. For more information, see the following topics: <ul style="list-style-type: none"><li>• <a href="#">Viewing available patches</a></li><li>• <a href="#">DescribeAvailablePatches</a> and <a href="#">Patch</a> in the <i>AWS Systems Manager API Reference</i></li><li>• <a href="#">describe-available-patches</a> in the <i>AWS Systems Manager section of the AWS CLI Command Reference</i></li></ul>	October 16, 2020

<a href="#">Session Manager plugin for the AWS CLI version 1.2.7.0 (p. 1558)</a>	A new version of the Session Manager plugin for the AWS CLI has been released. For more information, see <a href="#">Session Manager plugin latest version and release history</a> .	October 15, 2020
<a href="#">New topic: Session document schema (p. 1558)</a>	The new topic <a href="#">Session document schema</a> describes the schema elements for a Session document. This information can help you create custom Session documents where you specify preferences for the types of sessions you use with Session Manager.	October 15, 2020
<a href="#">New topic: Free text search for SSM documents (p. 1558)</a>	The search box on the Systems Manager <a href="#">Documents</a> page now supports free text search. Free text search compares the search term or terms that you enter against the document name in each SSM document. For more information, see <a href="#">Using free text search</a> .	October 15, 2020
<a href="#">New topic: Troubleshooting Amazon EC2 managed instance availability (p. 1558)</a>	The new topic <a href="#">Troubleshooting Amazon EC2 managed instance availability</a> helps you investigate why an Amazon EC2 instance that you have confirmed is running isn't available in lists of available managed instances in Systems Manager.	October 6, 2020
<a href="#">Parameter Store chapter reorganization (p. 1558)</a>	To help you find the information you need more efficiently, we reorganized content in the Parameter Store chapter of the <i>AWS Systems Manager User Guide</i> . Most content is now organized in the sections <a href="#">Setting up Parameter Store</a> and <a href="#">Working with Parameter Store</a> . In addition, the topic <a href="#">AWS Systems ManagerParameter Store</a> has been expanded to include the following sections:	October 1, 2020
	<ul style="list-style-type: none"> <li>• How can Parameter Store benefit my organization?</li> <li>• Who should use Parameter Store?</li> <li>• What are the features of Parameter Store?</li> <li>• What is a parameter?</li> </ul>	

New patch compliance-related topics (p. 1558)	The following topics have been added to help you identify managed instances that are out of patch compliance, understand the different types of patch compliance scans, and take the appropriate steps to bring your instances into compliance. <ul style="list-style-type: none"><li>• <a href="#">Identifying out-of-compliance instances</a></li><li>• <a href="#">Patching out-of-compliance instances</a></li><li>• <a href="#">Viewing patch compliance results</a></li></ul>	September 24, 2020
SSM Agent version 3.0 (p. 1558)	Systems Manager launched a new version of SSM Agent. For more information, see <a href="#">SSM Agent version 3</a> .	September 21, 2020
New and updated topics: Amazon EventBridge replaces CloudWatch Events for event management (p. 1558)	CloudWatch Events and EventBridge are the same underlying service and API, but EventBridge provides more features and is now the preferred way to manage your events in AWS. (Changes you make in either CloudWatch or EventBridge are reflected in each console.) References to CloudWatch Events and existing procedures throughout the <i>AWS Systems Manager User Guide</i> have been updated to reflect EventBridge support. In addition, the following new topics have been added. <ul style="list-style-type: none"><li>• <a href="#">Monitoring Systems Manager events</a></li><li>• <a href="#">Configuring EventBridge for Systems Manager events</a></li><li>• <a href="#">Systems Manager target type examples</a></li><li>• <a href="#">Reference: Amazon EventBridge event patterns and types for Systems Manager</a></li></ul>	September 18, 2020
OpsItem archiving (p. 1558)	OpsCenter automatically archives OpsItems after 36 months, regardless of status.	September 17, 2020

[Integrating AWS Security Hub and Patch Manager \(p. 1558\)](#)

You can now integrate Patch Manager with AWS Security Hub. Security Hub provides a comprehensive view of your security state in AWS and helps you check your environment against security industry standards and best practices. When integrated with Patch Manager, Security Hub monitors the patching status of your fleets from a security point of view. For more information, see [Integrating Patch Manager with AWS Security Hub](#).

September 17, 2020

[Maintenance window pseudo parameters: New resource types supported for {{TARGET\\_ID}} and {{RESOURCE\\_ID}} \(p. 1558\)](#)

When you register a maintenance window 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 {{TARGET\_ID}} and {{RESOURCE\_ID}}. When the maintenance window task runs, it passes the correct values instead of the pseudo parameter placeholders. Two additional resource types are now available for use with the pseudo parameters {{TARGET\_ID}} and {{RESOURCE\_ID}}. You can now use the resources types AWS::RDS::DBInstance and AWS::SSM::ManagedInstance with both these pseudo parameters. For information about maintenance window pseudo parameters, see [About pseudo parameters](#).

September 14, 2020

Patch instances on demand with new 'Patch now' option (p. 1558)	You can now use the Systems Manager console to patch instances, or scan for missing patches, at any time. You can do this without having to create or modify a schedule, or specify full patching configuration options to accommodate an immediate patching need. You need only specify whether to scan or install patches and identify the target instances for the operation. Patch Manager automatically applies the current default patch baseline for your instance types and applies best practice options for how many instances are patched at once, and how many errors are permitted before the operation fails. For more information, see <a href="#">Patching instances on demand</a> .	September 9, 2020
New topic: Checking SSM Agent status and starting the agent (p. 1558)	The new topic <a href="#">Checking SSM Agent status and starting the agent</a> provides commands to check whether SSM Agent is running on each supporting operating system. It also provides the commands to start the agent if it isn't running.	September 7, 2020
Patch Manager now supports Ubuntu Server 20.04 LTS (p. 1558)	You can now use Patch Manager to patch Ubuntu Server 20.04 LTS instances. For more information, see the following topics: <ul style="list-style-type: none"><li>• <a href="#">How security patches are selected</a></li><li>• <a href="#">How patches are installed</a></li><li>• <a href="#">How patch baseline rules work on Ubuntu Server</a></li></ul>	August 31, 2020
New topic for Use cases and best practices (p. 1558)	We've added a new topic to help users quickly understand the differences between Maintenance Windows and State Manager. For more information, see <a href="#">Choosing between State Manager and Maintenance Windows</a> .	August 28, 2020

New OpsCenter features (p. 1558)	OpsCenter include new features to help you quickly locate and run Automation runbooks to remediate issues. For more information, see <a href="#">Automation runbook features in OpsCenter</a> .	August 19, 2020
New data source in Explorer: AWS Support cases (p. 1558)	Explorer now displays information about AWS Support cases. You must have either an Enterprise or Business account set up with AWS Support. For more information, see <a href="#">Editing Systems Manager Explorer data sources</a> .	August 13, 2020
Distributor now provides a third-party package from Trend Micro. (p. 1558)	Distributor now includes a third-party package from Trend Micro. You can use Distributor to install the Trend Micro Cloud One agent on your managed instances. Trend Micro Cloud One helps you secure your workloads in the cloud. For more information, see <a href="#">AWSDistributor</a> .	August 12, 2020
The aws : configurePackage document plugin now includes the additionalArguments parameter. (p. 1558)	The Systems Manager Command document plugin aws : configurePackage now supports providing additional parameters to your scripts (install, uninstall, and update) with the new additionalArguments parameter. For more information, see the topic <a href="#">aws : configurePackage</a> .	August 11, 2020
AppConfig content moved into a separate user guide (p. 1558)	Information about AWS AppConfig has been moved into a separate user guide. For more information, see <a href="#">What Is AWS AppConfig?</a> AppConfig also has a separate <a href="#">documentation landing page</a> with links to the user guide, the AppConfig API reference, and a new AppConfig workshop.	August 3, 2020
Quick Setup now supports AWS Organizations (p. 1558)	Quick Setup now supports AWS Organizations allowing you to quickly configure required security roles and commonly used Systems Manager capabilities across multiple accounts and Regions. For more information, see <a href="#">AWS Systems ManagerQuick Setup</a> .	July 23, 2020

New data source in Explorer: association compliance (p. 1558)	Explorer now displays association compliance data from State Manager. For more information, see <a href="#">Editing Systems Manager Explorer data sources</a> .	July 23, 2020
New Systems Manager Command document to turn on and turn off Kernel Live Patching (p. 1558)	The document <a href="#">AWS-ConfigureKernelLivePatching</a> is now available to use with Run Command when you want to turn on or turn off Kernel Live Patching on Amazon Linux 2 instances. This document replaces the need for creating your own custom Command documents for these tasks. For more information, see <a href="#">Use Kernel Live Patching on Amazon Linux 2 instances</a>	July 22, 2020
Updated Automation quotas (p. 1558)	Service quotas for Automation have been updated including a separate queue for rate control automations. For more information, see <a href="#">AWS Systems Manager Automation</a> .	July 20, 2020
Specify the number of schedule offset days for a maintenance window using the console (p. 1558)	Using the Systems Manager console, you can now specify a number of days to wait after the date and time specified by a CRON expression before running a maintenance window. (Previously, this option was available only when using an AWS SDK or a command line tool.) For example, if your CRON expression schedules a maintenance window to run on the third Tuesday of every month at 11:30 PM – cron(0 30 23 ? * TUE#3 *) – and you specify a schedule offset of 2, the window won't run until two days later at 11:30 PM. For more information, see <a href="#">Cron and rate expressions for Systems Manager</a> and <a href="#">Specify the number of schedule offset days for a maintenance window</a> .	July 17, 2020

<a href="#">Update PowerShell using Run Command (p. 1558)</a>	To help you update PowerShell to version 5.1 on your Windows Server 2012 and 2012 R2 instances, we added a walkthrough to the AWS Systems Manager User Guide. For more information, see <a href="#">Update PowerShell using Run Command</a> .	June 30, 2020
<a href="#">Patch Manager now supports CentOS 8.0 and 8.1 (p. 1558)</a>	You can now use Patch Manager to patch CentOS 8.0 and 8.1 instances. For more information, see the the following topics: <ul style="list-style-type: none"><li>• <a href="#">How security patches are selected</a></li><li>• <a href="#">How patches are installed</a></li><li>• <a href="#">How patch baseline rules work on CentOS</a></li><li>• <a href="#">Manually install SSM Agent on CentOS instances</a></li><li>• <a href="#">Install SSM Agent for a hybrid environment (Linux)</a></li></ul>	June 27, 2020

[AppConfig integrates with AWS CodePipeline \(p. 1558\)](#)

AppConfig is an integrated deploy action for AWS CodePipeline (CodePipeline). CodePipeline is a fully managed continual delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define. The integration of AppConfig with CodePipeline offers the following benefits. For more information, see [AppConfig integration with CodePipeline](#).

June 25, 2020

- Customers who use CodePipeline to manage orchestration now have a lightweight means of deploying configuration changes to their applications without having to deploy their entire codebase.
- Customers who want to use AppConfig to manage configuration deployments but are limited because AppConfig doesn't support their current code or configuration store, now have additional options. CodePipeline supports AWS CodeCommit, GitHub, and BitBucket (to name a few).

[New chapter: Product and service integrations \(p. 1558\)](#)

To help you understand how Systems Manager integrates with AWS services and other products and services, a new chapter has been added to the AWS Systems Manager User Guide. For more information, see [Product and service integrations with Systems Manager](#).

June 23, 2020

[Automation chapter reorganization \(p. 1558\)](#)

To help you find what you need, we reorganized topics in the Automation chapter of the *AWS Systems Manager User Guide*. For example, the Automation actions and Automation runbooks references are now top-level sections in the chapter. For more information, see [AWS Systems Manager Automation](#).

June 23, 2020

[Specify the number of schedule offset days for a maintenance window \(p. 1558\)](#)

Using a command line tool or AWS SDK, you can now specify a number of days to wait after the date and time specified by a CRON expression before running a maintenance window. For example, if your CRON expression schedules a maintenance window to run on the third Tuesday of every month at 11:30 PM – `cron(0 30 23 ? * TUE#3 *)` – and you specify a schedule offset of 2, the window won't run until two days later at 11:30 PM. For more information, see [Cron and rate expressions for Systems Manager](#) and [Specify the number of schedule offset days for a maintenance window](#).

June 19, 2020

[Patch Manager support for Kernel Live Patching on Amazon Linux 2 instances \(p. 1558\)](#)

Kernel Live Patching for Amazon Linux 2 allows you to apply security vulnerability and critical bug patches to a running Linux kernel, without reboots or disruptions to running applications. You can now allow the feature and apply kernel live patches using Patch Manager. For information, see [Use Kernel Live Patching on Amazon Linux 2 instances](#).

June 16, 2020

[Patch Manager increases Oracle Linux version support \(p. 1558\)](#)

Previously, Patch Manager supported only version 7.6 of Oracle Linux. As listed in [Patch Manager prerequisites](#), support now covers versions 7.5-7.8.

June 16, 2020

Sample scenario for using the <code>InstallOverrideList</code> parameter in patching operations (p. 1558)	The new topic <a href="#">Sample scenario for using the <code>InstallOverrideList</code> parameter</a> describes a strategy for using the <code>InstallOverrideList</code> 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.	June 11, 2020
Predefined deployment strategies for AppConfig (p. 1558)	AppConfig now offers predefined deployment strategies. For more information, see <a href="#">Creating a deployment strategy</a> .	June 10, 2020
Patch Manager now supports Red Hat Enterprise Linux (RHEL) 7.8-8.2 (p. 1558)	You can now use Patch Manager to patch RHEL 7.8 - 8.2 instances. For more information, see the the following topics: <ul style="list-style-type: none"><li>• <a href="#">How security patches are selected</a></li><li>• <a href="#">How patches are installed</a></li><li>• <a href="#">How patch baseline rules work on RHEL</a></li><li>• <a href="#">Manually install SSM Agent on Red Hat Enterprise Linux instances</a></li><li>• <a href="#">Install SSM Agent for a hybrid environment (Linux)</a></li></ul>	June 9, 2020
Explorer supports delegated administration (p. 1558)	If you aggregate Explorer data from multiple AWS Regions and AWS 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 management account to administer resource data syncs in Explorer. For more information, see <a href="#">Configuring a Delegated Administrator</a> .	June 3, 2020

Apply State Manager association only at the next specified Cron interval (p. 1558)	If you don't want a State Manager association to run immediately after you create it, you can choose the <b>Apply association only at the next specified Cron interval</b> option in the Systems Manager console. For more information, see <a href="#">Creating associations</a> .	June 3, 2020
New data source in Explorer: AWS Compute Optimizer (p. 1558)	Explorer now displays data from AWS Compute Optimizer. This includes a count of <b>Under provisioned</b> and <b>Over provisioned</b> 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 <a href="#">Setting up related services</a> .	May 26, 2020
New chapter: Tagging Systems Manager Resources (p. 1558)	The new chapter <a href="#">Tagging Systems Manager resources</a> 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: <ul style="list-style-type: none"><li>• Documents</li><li>• Maintenance windows</li><li>• Managed instances</li><li>• OpsItems</li><li>• Parameters</li><li>• Patch baselines</li></ul>	May 25, 2020
Install Windows Service Packs and Linux minor version upgrades using Patch Manager (p. 1558)	The new topic <a href="#">Walkthrough: Create a patch baseline for installing Windows Service Packs (console)</a> demonstrates how you can create a patch baseline devoted exclusively to installing Windows Service Packs. The topic <a href="#">Create a custom patch baseline (Linux)</a> has been updated with information about including minor version upgrades for Linux operating systems in patch baselines.	May 21, 2020

<a href="#">Parameter Store chapter reorganization (p. 1558)</a>	All topics that deal with configuring or setting options for Parameter Store operations have been consolidated into the <a href="#">Setting up Parameter Store</a> section. This includes the topics <a href="#">Managing parameter tiers</a> and <a href="#">Increasing Parameter Store throughput</a> , which have been relocated from other parts of the chapter.	May 18, 2020
<a href="#">New topic for creating date and time strings for interacting with Systems Manager API operations. (p. 1558)</a>	The new topic <a href="#">Creating formatted date and time strings for Systems Manager</a> describes how to create formatted date and time strings for interacting with Systems Manager API operations.	May 13, 2020
<a href="#">About permissions for encrypting SecureString parameters (p. 1558)</a>	The new topic <a href="#">Restricting access to Systems Manager parameters using IAM policies</a> explains the difference between encrypting your SecureString parameters using an AWS KMS key and using the AWS managed key provided by AWS.	May 13, 2020
<a href="#">Patch Manager now supports the Debian Server and Oracle Linux 7.6 operating systems (p. 1558)</a>	You can now use Patch Manager to patch Debian Server and Oracle Linux instances. Patch Manager supports patching Debian Server 8.x and 9.x and Oracle Linux 7.6 versions. For more information, see the following topics:	May 7, 2020
	<ul style="list-style-type: none"> <li>• <a href="#">How security patches are selected</a></li> <li>• <a href="#">How patches are installed</a></li> <li>• <a href="#">How patch baseline rules work on Debian Server</a></li> <li>• <a href="#">How patch baseline rules work on Oracle Linux</a></li> </ul>	
<a href="#">Create State Manager associations that target AWS Resource Groups (p. 1558)</a>	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 <a href="#">About targets and rate controls in State Manager associations</a>	May 7, 2020

New aws : ec2 : image data type in Parameter Store to validate AMI IDs (p. 1558)	When you create a String parameter, you can now specify a <i>data type</i> 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 means you don't have to update 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 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 <a href="#">Native parameter support for Amazon Machine Image IDs</a> .	May 5, 2020
Managing exit codes in Run Command commands (p. 1558)	Run Command enables you to 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 <a href="#">Managing exit codes in Run Command commands</a> .	May 5, 2020

New public parameters released for availability zones and local zones (p. 1558)	Public parameters have been released to make information about AWS <i>availability zones</i> and <i>local zones</i> available programmatically. These are in addition to existing global infrastructure public parameters for AWS services and AWS Regions. For more information, see <a href="#">Calling public parameters for AWS services, Regions, endpoints, Availability Zones, local zones, and Wavelength Zones</a> .	May 4, 2020
New data source in Explorer: AWS Trusted Advisor (p. 1558)	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 quotas. For more information, see the details for setting up Trusted Advisor in <a href="#">Setting up related services</a> .	May 4, 2020

[Create State Manager associations that run Chef recipes \(p. 1558\)](#)

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.

March 19, 2020

[Synchronize inventory data from multiple AWS accounts to a central Amazon S3 bucket \(p. 1558\)](#)

For more information, see [Creating associations that run Chef recipes](#)

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 [Creating an Inventory resource data sync for multiple accounts defined in AWS Organizations](#).

March 16, 2020

<a href="#">Store AppConfig configurations in Amazon S3 (p. 1558)</a>	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 <a href="#">About configurations stored in Amazon S3</a> .	March 13, 2020
<a href="#">SSM Agent installed by default on Amazon ECS-optimized AMIs (p. 1558)</a>	SSM Agent is now installed by default on Amazon ECS-Optimized AMIs. For more information, see <a href="#">Working with SSM Agent</a> .	February 25, 2020
<a href="#">Create AppConfig configurations in the console (p. 1558)</a>	AppConfig now allows you to create an application configuration in the console at the time you create a configuration profile. For more information, see <a href="#">Creating a configuration and a configuration profile</a> .	February 13, 2020
<a href="#">Auto-approve only patches released up to a specified date (p. 1558)</a>	In addition to the option for automatically approving patches for installation a specified number of days after they're 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 <a href="#">About custom baselines</a> and <a href="#">Working with custom patch baselines (console)</a> .	February 12, 2020

[Use the {{RESOURCE\\_ID}} pseudo parameter in maintenance window tasks \(p. 1558\)](#)

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](#)

[Quickly rerun commands \(p. 1558\)](#)

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 allows 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. For more information, see [Rerunning commands](#).

February 6, 2020

February 5, 2020

Reverting from the advanced-instances tier to the standard-instances tier (p. 1558)	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 <a href="#">Reverting from the advanced-instances tier to the standard-instances tier</a> .	January 16, 2020
New option to skip instance reboots after patch installation (p. 1558)	Previously, managed instances were always rebooted after Patch Manager installed patches on them. A new <code>RebootOption</code> parameter in the SSM document <code>AWS-RunPatchBaseline</code> allows you to specify whether or not you want your instances to reboot automatically after new patches are installed. For more information, see <a href="#">Parameter name: RebootOption</a> in the topic <a href="#">About the SSM document AWS-RunPatchBaseline</a> .	January 15, 2020
New topic: 'Running PowerShell scripts on Linux instances' (p. 1558)	A new topic that describes how to use Run Command to run PowerShell scripts on Linux instances. For more information, see <a href="#">Running PowerShell scripts on Linux instances</a> .	January 10, 2020
Updates to 'configure SSM Agent to use a proxy' (p. 1558)	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 <a href="#">Configure SSM Agent to use a proxy</a> .	January 9, 2020

New "Security" chapter outlines practices for securing Systems Manager resources (p. 1558)	A new <a href="#">Security</a> chapter in the <i>AWS Systems Manager User Guide</i> helps you understand how to apply the <a href="#">shared responsibility model</a> 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.	December 24, 2019
<b>Note</b> As part of this update, the user guide chapter "Authentication and Access Control" has been replaced by a new, simpler section, <a href="#">Identity and access management for AWS Systems Manager</a> .		
New sample custom Automation runbooks (p. 1558)	A set of sample custom Automation runbooks 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 runbooks. For more information, see <a href="#">Custom Automation runbook samples</a> . You can also view Amazon managed Automation runbook content in the Systems Manager console. For more information, see <a href="#">Systems Manager Automation runbook reference</a> .	December 23, 2019
Support for the Oracle Linux (p. 1558)	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 <a href="#">Oracle Linux</a> . For information about installing SSM Agent on Oracle Linux servers in a hybrid environment, see <a href="#">Step 6: Install SSM Agent for a hybrid environment (Linux)</a> .	December 19, 2019

Add a script from another Automation runbook to your workflow (p. 1558)	When you create an Automation runbook 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 <code>aws : executeScript</code> action type.) You can add scripts from Automation runbooks that you own or that are shared with you from another AWS account. For details, see <a href="#">Creating an Automation runbook that runs scripts (command line)</a> .	December 19, 2019
Launch Session Manager sessions from the Amazon EC2 console (p. 1558)	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.	December 18, 2019
CloudWatch support for Run Command metrics and alarms (p. 1558)	<ul style="list-style-type: none"><li>• <a href="#">Quickstart default IAM policies for Session Manager</a></li><li>• <a href="#">Starting a session (Amazon EC2 console)</a></li></ul> AWS Systems Manager now publishes metrics about the status of Run Command commands to CloudWatch, allowing you to set alarms based on those metrics. The terminal status values for commands for which you can track metrics include Success, Failed, and Delivery Timed Out. For more information, see <a href="#">Monitoring Run Command metrics using Amazon CloudWatch</a> .	December 17, 2019

[New Systems Manager capability: Change Calendar \(p. 1558\)](#)

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 runbooks or AWS Lambda functions) to resources. A change calendar is a new Systems Manager document type that stores [iCalendar 2.0](#) data in plaintext format. For more information, see [AWS Systems Manager Change Calendar](#).

December 11, 2019

[New Systems Manager capability: AWS AppConfig \(p. 1558\)](#)

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 starts an alarm, AppConfig rolls back the change to minimize impact for your application users. For more information, see [AWS AppConfig](#).

November 25, 2019

New Systems Manager capability: Systems Manager Explorer (p. 1558)

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 AWS 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 haven't 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 \(p. 1558\)](#)

The tools for searching for parameters now make it easier to find parameters when you have large number of them in your account or when you don't remember the exact name of a parameter. With the search tool, you filter by contains. Previously, the search tools supported searching for parameter names only by equals and begins-with. For more information, see [Searching for Systems Manager parameters](#).

November 15, 2019

[New console-based Document Builder for Automation | Support for running scripts in Automation steps \(p. 1558\)](#)

You can now use Systems Manager Automation to build and share standardized operational playbooks to ensure consistency across users, AWS accounts, and AWS Regions. With this ability to run scripts and add inline documentation to your Automation runbooks using Markdown, you can reduce errors and eliminate manual steps such as navigating written procedures in wikis and running terminal commands.

November 14, 2019

For more information, see the following topics.

- [Walkthrough: Using Document Builder to create a custom Automation runbook](#)
- [Amazon managed Automation runbooks that run scripts](#)
- [`aws:executeScript` \(Automation actions reference\)](#)
- [Creating Automation runbooks that run scripts](#)
- [Creating Automation runbooks using Document Builder](#)
- [New Automation Features In Systems Manager on the AWS News Blog](#)

[Perform an in-place package update using Distributor \(p. 1558\)](#)

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.

November 11, 2019

- [Create a package](#)
- [Install or update packages](#)

[New SSM Agent auto update feature \(p. 1558\)](#)

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 \(p. 1558\)](#)

A second method for controlling user access to session actions is now available. With this new method, you create IAM access policies using AWS-supplied session tags instead of using the `{aws:username}` 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

[New SSM Command document to apply Ansible Playbooks \(p. 1558\)](#)

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

September 24, 2019

[Port forwarding support for Session Manager \(p. 1558\)](#)

For more information, see [Creating associations that run Ansible playbooks](#)

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 ManagerSession Manager](#) on the [AWS News Blog](#)
- [Starting a session \(port forwarding\)](#)

August 29, 2019

Specify a default parameter tier or automate tier selection (p. 1558)	You can now specify a default parameter tier to use for requests to create or update a parameter that don't 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 <a href="#">Specifying a default parameter tier</a> .	August 27, 2019
Working with associations section updated with CLI and PowerShell procedures (p. 1558)	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, <a href="#">Working with associations in Systems Manager</a> .	August 26, 2019
Working with Automation executions section updated with CLI and PowerShell procedures (p. 1558)	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, <a href="#">Working with Automation executions</a> .	August 20, 2019

[OpsCenter integrates with application insights \(p. 1558\)](#)

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 [Set up, configure, and manage your application for monitoring in the Amazon CloudWatch User Guide](#).

August 7, 2019

[New console feature: AWS Systems Manager Quick Setup \(p. 1558\)](#)

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.

August 7, 2019

For more information, see [AWS Systems Manager Quick Setup](#).

[Register a resource group as a maintenance window target \(p. 1558\)](#)

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](#)

[Simplified package creation and versioning with AWS Systems ManagerDistributor \(p. 1558\)](#)

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 23, 2019

July 22, 2019

July 18, 2019

[New document categories pane for Systems Manager Automation \(p. 1558\)](#)

Systems Manager includes a new Document categories pane when you run an Automation in the console. Use this pane to filter Automation runbooks based on their purpose.

Verify user permissions to access the default Session Manager configuration document (p. 1558)	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 <a href="#">Enforce document permission check for default CLI scenario</a> .	July 9, 2019
Support for starting Session Manager sessions using operating system user credentials (p. 1558)	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 <a href="#">Turn on Run As support for Linux instances</a> .	July 9, 2019
Support for starting Session Manager sessions using SSH (p. 1558)	You can now use the AWS CLI to start an SSH session on a managed instance using Session Manager. For information about allowing SSH sessions with Session Manager, see <a href="#">(Optional) Turn on SSH Session Manager sessions</a> . For information about starting an SSH session using Session Manager, see <a href="#">Starting a session (SSH)</a> .	July 9, 2019
Support for changing passwords on managed instances (p. 1558)	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 <a href="#">Resetting passwords on managed instances</a> .	July 9, 2019

<a href="#">Revisions to "What is AWS Systems Manager?" (p. 1558)</a>	The introductory content in <a href="#">What is AWS Systems Manager?</a> 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.	June 10, 2019
<a href="#">New Systems Manager capability: OpsCenter (p. 1558)</a>	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 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 <a href="#">AWS Systems ManagerOpsCenter</a> .	June 6, 2019
<a href="#">Changes to Systems Manager left navigation pane in the AWS Management Console (p. 1558)</a>	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.	June 6, 2019

Revised tutorial for creating and configuring a maintenance window using the AWS CLI (p. 1558)	<p>Tutorial: <a href="#">Create and configure a maintenance window (AWS CLI)</a> 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 <code>{{TARGET_ID}}</code>. For additional information and examples, see the following topics:</p> <ul style="list-style-type: none"><li>• <a href="#">Examples: Register targets with a maintenance window</a></li><li>• <a href="#">Examples: Register tasks with a maintenance window</a></li><li>• <a href="#">About register-task-with-maintenance-windows options</a></li><li>• <a href="#">About pseudo parameters</a></li></ul>	May 31, 2019
Notifications about SSM Agent updates (p. 1558)	<p>To be notified about SSM Agent updates, subscribe to the <a href="#">SSM Agent Release Notes</a> page on GitHub.</p>	May 24, 2019
Receive notifications or trigger actions based on changes in Parameter Store (p. 1558)	<p>The topic <a href="#">Set up notifications or trigger actions based on Parameter Store events</a> now helps you set up Amazon EventBridge rules to respond to changes in Parameter Store. You can receive notifications or trigger other actions when any of the following occur:</p> <ul style="list-style-type: none"><li>• A parameter is created, updated, or deleted.</li><li>• A parameter label version is created, updated, or deleted.</li><li>• A parameter expires, is going to expire, or hasn't changed in a specified period of time.</li></ul>	May 22, 2019

Major revisions to setting up and getting started content (p. 1558)

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.

May 15, 2019

- [Setting up AWS Systems Manager](#)
- [Setting up AWS Systems Manager for hybrid environments](#)
- [Getting started with AWS Systems Manager](#)

<a href="#">Include patches for applications released by Microsoft in patch baselines (Windows) (p. 1558)</a>	<p>Patch Manager now supports patch updates for applications released by Microsoft 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 <code>AWS-WindowsPredefinedPatchBaseline-OS</code> applies to operating system patches only. <code>AWS-WindowsPredefinedPatchBaseline-OS-Applications</code> applies to both the Windows Server operating system and applications released by Microsoft on Windows. For information about creating a custom patch baseline that includes patches for applications released by Microsoft, see the first procedure in <a href="#">Create a custom patch baseline</a>. Also, as part of this update, the names of AWS-provided predefined patch baselines are being changed. For more information, see <a href="#">Predefined baselines</a>.</p>	May 7, 2019
<a href="#">Examples for registering maintenance window targets using the AWS CLI (p. 1558)</a>	<p>The new topic <a href="#">Examples: Register targets with a maintenance window</a> 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.</p>	May 3, 2019

Updates to patch group topics (p. 1558)

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 \(p. 1558\)](#)

Parameter Store offers the following new features:

April 25, 2019

- **Advanced parameters:**

Parameter Store now allows 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 quota for the parameter value, a higher quota for the number of parameters you can create per AWS account and AWS 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 allowing 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 quota

to a maximum of 1,000 transactions per second. For more information, see [Increasing Parameter Store throughput](#).

Updates to the Automation section (p. 1558)	The Automation section has been updated for improved discoverability. In addition, three new topics have been added to the Automation section: <ul style="list-style-type: none"><li>• <a href="#">Running an automation manually (p. 411)</a></li><li>• <a href="#">Running an automation with approvers (p. 417)</a></li><li>• <a href="#">Running automations based on triggers (p. 435)</a></li></ul>	April 17, 2019
Encrypt session data using an AWS KMS key (p. 1558)	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 an AWS KMS key that has been created in AWS Key Management Service. You can use a KMS 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 KMS key to encrypt session data, see <a href="#">Turn on AWS KMS key encryption of session data (console)</a> , <a href="#">Create Session Manager preferences (AWS CLI)</a> , or <a href="#">Update Session Manager preferences (AWS CLI)</a> .	April 4, 2019
Configuring Amazon SNS notifications for AWS Systems Manager (p. 1558)	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 <a href="#">Configuring Amazon SNS notifications for AWS Systems Manager</a> .	March 6, 2019

Advanced instances for servers and VMs in hybrid environments (p. 1558)	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 allows 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 allow 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 allowing advanced instances, see <a href="#">Using the advanced-instances tier</a> .	March 4, 2019
Create State Manager associations that use shared SSM documents (p. 1558)	You can create State Manager associations that use SSM Command and Automation runbooks 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 aren't in the same account. For information about sharing SSM documents, see <a href="#">AWS Systems Manager Documents</a> . For information about creating a State Manager association, see <a href="#">Create an association</a> .	February 28, 2019
View lists of Systems Manager events supported for Amazon EventBridge rules (p. 1558)	The new topic <a href="#">Monitoring Systems Manager events with Amazon EventBridge</a> provides a summary of the various events emitted by Systems Manager for which you can set up event monitoring rules in EventBridge.	February 25, 2019

<a href="#">Add tags when you create Systems Manager resources (p. 1558)</a>	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.	February 24, 2019
<a href="#">Automatic IAM role creation for Systems Manager Inventory (p. 1558)</a>	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 <b>Inventory Detail View</b> 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 <b>Inventory Detail View</b> page, Systems Manager automatically creates the <code>Amazon-GlueServicePolicyForSSM</code> role and assigns the <code>Amazon-GlueServicePolicyForSSM-{S3 bucket name}</code> policy and the <code>AWSGlueServiceRole</code> policy to it. For more information, see <a href="#">Querying inventory data from multiple Regions and accounts</a> .	February 14, 2019
<a href="#">Maintenance Windows walkthroughs to update SSM Agent (p. 1558)</a>	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 <a href="#">Maintenance Windows walkthroughs</a> .	February 11, 2019
<a href="#">Using Parameter Store public parameters (p. 1558)</a>	Added short section describing Parameter Store public parameters. For more information, see <a href="#">Using Systems Manager public parameters</a> .	January 31, 2019

Use the AWS CLI to create Session Manager preferences (p. 1558)	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 <a href="#">Use the AWS CLI to create Session Manager preferences</a> .	January 22, 2019
Executing Systems Manager automation workflows by using State Manager (p. 1558)	AWS Systems Manager State Manager now supports creating associations that use SSM Automation runbooks. 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 runbooks, you can now create associations that target different types of AWS resources. For more information, see <a href="#">Executing Systems Manager Automation workflows by using State Manager</a> .	January 22, 2019
Reference updates for cron and rate expressions and maintenance window scheduling options (p. 1558)	The reference topic <a href="#">Cron and rate expressions for Systems Manager</a> 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 <a href="#">Maintenance Windows scheduling and active period options</a> explains how the various schedule-related options for maintenance windows (Start date, End date, Time zone, Schedule frequency) relate to one another.	December 6, 2018
Updates to the Systems Manager prerequisites topic (p. 1558)	The <a href="#">Systems Manager prerequisites</a> 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.	December 4, 2018

<a href="#">Turn on SSM Agent debug logging (p. 1558)</a>	You can turn on SSM Agent debug logging by editing the <code>seelog.xml.template</code> file on the managed instance. For more information, see <a href="#">Turn on SSM Agent debug logging</a> .	November 30, 2018
<a href="#">Support for ARM64 processor architectures (p. 1558)</a>	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 <a href="#">Amazon Linux 2</a> , <a href="#">RHEL</a> , and <a href="#">Ubuntu Server 18.04 and 16.04 LTS with Snap packages</a> . For more information about the A1 instance type, see <a href="#">General purpose instances</a> in the <a href="#">Amazon EC2 User Guide for Linux Instances</a> .	November 26, 2018
<a href="#">Create and deploy packages by using AWS Systems ManagerDistributor (p. 1558)</a>	Using AWS Systems Manager Distributor, you package your own software—or find AWS-provided agent software packages, such as <code>AmazonCloudWatchAgent</code> —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 <a href="#">AWS Systems ManagerDistributor</a> .	November 20, 2018

Concurrently run AWS Systems Manager Automation workflows across multiple AWS Regions and AWS accounts from a central account (p. 1558)	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 <a href="#">Executing Automation workflows in multiple AWS Regions and AWS accounts</a> .	November 19, 2018
Query inventory data from multiple AWS Regions and AWS accounts (p. 1558)	Systems Manager Inventory integrates with Amazon Athena to help you query inventory data from multiple AWS Regions and AWS accounts. Athena integration uses resource data sync so that you can view inventory data from all of your managed instances on the <b>Inventory Detail View</b> page in the AWS Systems Manager console. For more information see <a href="#">Querying Inventory data from multiple Regions and accounts</a> .	November 15, 2018

Create State Manager associations that run MOF files (p. 1558)	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 allows 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 <a href="#">Creating associations that run MOF files</a> .	November 15, 2018
Restrict administrative access in Session Manager sessions (p. 1558)	Session Manager sessions are launched using the credentials of a user account that is created with default root or administrator permissions called <code>ssm-user</code> . Information about restricting administrative control for this account is now available in the topic <a href="#">Turn on or turn off ssm-user account administrative permissions</a> .	November 13, 2018
YAML examples in Automation actions reference (p. 1558)	The <a href="#">Automations actions reference</a> now includes a YAML sample for each action that already includes a JSON sample.	October 31, 2018

<a href="#">Assign compliance severity levels to associations (p. 1558)</a>	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 <a href="#">Create an association (console)</a> .	October 26, 2018
<a href="#">Use targets and rate controls with Automation and State Manager (p. 1558)</a>	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 <a href="#">Using targets and rate controls to run Automation workflows on a fleet</a> and <a href="#">Using targets and rate controls with State Manager associations</a> .	October 23, 2018
<a href="#">Specify active time ranges and international time zones for maintenance windows (p. 1558)</a>	You can also specify dates that a maintenance window shouldn't 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 <a href="#">Create a maintenance window (console)</a> and <a href="#">Update a maintenance window (AWS CLI)</a> .	October 9, 2018
<a href="#">Maintain a custom list of patches for your patch baseline in an S3 bucket (p. 1558)</a>	With the new 'InstallOverrideList' parameter in the SSM command document <code>AWS-RunPatchBaseline</code> , you can 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 <a href="#">Parameter name: InstallOverrideList</a> .	October 5, 2018

<a href="#">Expanded control over whether patch dependencies are installed (p. 1558)</a>	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 <a href="#">Create a patch baseline</a> .	October 5, 2018
<a href="#">Create dynamic automation workflows with conditional branching (p. 1558)</a>	The <code>aws:branch</code> Automation action allows 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 runbook based on the results of that evaluation. For more information, see <a href="#">Creating dynamic Automation workflows with conditional branching</a> .	September 26, 2018
<a href="#">Use the AWS CLI to update Session Manager preferences (p. 1558)</a>	Instructions for using the CLI to update Session Manager preferences, such as CloudWatch Logs and S3 bucket logging options, have been added to the <i>AWS Systems Manager User Guide</i> . For information, see <a href="#">Use the AWS CLI to update Session Manager preferences</a> .	September 25, 2018
<a href="#">Set up patching options with the new 'Configure patching' page (p. 1558)</a>	Patch Manager has been updated with a new system for setting up patching configurations. On the <a href="#">Configure patching</a> 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 <a href="#">About patching configurations</a> and <a href="#">Create a patching configuration</a> .	September 22, 2018
<a href="#">Updated SSM Agent requirement for Session Manager (p. 1558)</a>	Session Manager now requires SSM Agent version 2.3.68.0 or later. For more information about Session Manager prerequisites, see <a href="#">Complete Session Manager prerequisites</a> .	September 17, 2018

Manage instances without opening inbound ports or maintaining bastion hosts using Session Manager (p. 1558)	Using Session Manager, a fully managed capability of AWS Systems Manager, you can 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 allows you 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 <a href="#">Learn more about Session Manager</a> .	September 11, 2018
Invoking other AWS services from a Systems Manager Automation workflow (p. 1558)	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 runbooks. For more information, see <a href="#">Invoking other AWS services from a Systems Manager Automation workflow</a> .	August 28, 2018
Use Systems Manager-specific condition keys in IAM policies (p. 1558)	The topic <a href="#">Specifying conditions in a policy</a> 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.	August 18, 2018

Aggregate inventory data with groups to see which instances are and aren't configured to collect an inventory type (p. 1558)	Groups allow 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 more information, see <a href="#">Aggregating Inventory data</a> .	August 16, 2018
View history and change tracking for Inventory and Configuration Compliance (p. 1558)	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 <a href="#">Viewing Inventory history and change tracking</a> .	August 9, 2018
Systems Manager service-linked role extends support for maintenance window tasks (p. 1558)	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 <a href="#">Should I use a service-linked role or a custom service role to run maintenance window tasks?</a>	August 2, 2018

Parameter Store integrates with Secrets Manager (p. 1558)	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 <a href="#">Referencing AWS Secrets Manager secrets from Parameter Store parameters</a> .	July 26, 2018
Attach labels to Parameter Store parameters (p. 1558)	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 <a href="#">Labeling parameters</a> .	July 26, 2018
Create dynamic Automation workflows (p. 1558)	By default, the steps (or actions) that you define in the mainSteps section of an Automation runbook run in sequential order. After one action is complete, the next action specified in the mainSteps section begins. With this release, you can now create Automation workflows that perform <i>conditional branching</i> . This means that you can create Automation workflows that dynamically respond to condition changes and jump to a specified step. For information, see <a href="#">Creating dynamic Automation workflows</a> .	July 18, 2018

<a href="#">SSM Agent now pre-installed on Ubuntu Server 16.04 AMIs using Snap (p. 1558)</a>	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 <a href="#">About SSM Agent installations on 64-bit Ubuntu Server 16.04 instances</a> .	July 7, 2018
<a href="#">Review minimum S3 permissions required by SSM Agent (p. 1558)</a>	The new topic <a href="#">Minimum S3 bucket permissions for SSM Agent</a> 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.	July 5, 2018
<a href="#">View complete execution history for a specific State Manager association ID (p. 1558)</a>	The new topic <a href="#">Viewing association histories</a> describes how to view all executions for a specific association ID and then view execution details for one or more resources.	July 2, 2018
<a href="#">Patch Manager introduces support for Amazon Linux 2 (p. 1558)</a>	You can now use Patch Manager to apply patches to Amazon Linux 2 instances. For general information about Patch Manager operating system support, see <a href="#">Patch Manager prerequisites</a> . For information about the supported key-value pairs for Amazon Linux 2 when defining a patch filter, see <a href="#">PatchFilter</a> in the <i>AWS Systems Manager API Reference</i> .	June 26, 2018
<a href="#">Send command output to Amazon CloudWatch Logs (p. 1558)</a>	The new topic <a href="#">Configuring Amazon CloudWatch Logs for Run Command</a> describes how to send Run Command output to CloudWatch Logs.	June 18, 2018

Quickly create or delete resource data sync for Inventory by using AWS CloudFormation (p. 1558)	You can use AWS CloudFormation to create or delete a resource data sync for Systems Manager Inventory. To use AWS CloudFormation, add the <a href="#">AWS::SSM::ResourceDataSync</a> resource to your AWS CloudFormation template. For more information, see <a href="#">Working with AWS CloudFormation Templates</a> in the <i>AWS CloudFormation User Guide</i> . You can also manually create a resource data sync for Inventory as described in <a href="#">Configuring resource data sync for Inventory</a> .	June 11, 2018
AWS Systems Manager User Guide update notifications now available through RSS (p. 1558)	The HTML version of the Systems Manager User Guide now supports an RSS feed of updates that are documented in the <a href="#">Systems Manager Documentation update history</a> page. The RSS feed includes updates made in June, 2018, and later. Previously announced updates are still available in the <a href="#">Systems Manager documentation update history</a> page. Use the RSS button in the top menu panel to subscribe to the feed.	June 6, 2018
Specify an exit code in scripts to reboot managed instances (p. 1558)	The new topic <a href="#">Rebooting managed instances from scripts</a> describes how to instruct Systems Manager to reboot managed instances by specifying an exit code in scripts that you run with Run Command.	June 3, 2018
Create an event in Amazon EventBridge whenever custom inventory is deleted (p. 1558)	The new topic <a href="#">Viewing inventory delete actions in EventBridge</a> describes how to configure Amazon EventBridge to create an event anytime a user deletes custom Inventory.	June 1, 2018

## Updates prior to June 2018

The following table describes important changes in each release of the *AWS Systems Manager User Guide* before June 2018.

Change	Description	Release date
Inventory all managed instances in your AWS account	<p>You can inventory all managed instances in your AWS account by creating a global inventory association. For more information, see <a href="#">Inventory all managed nodes in your AWS account (p. 867)</a>.</p> <p><b>Note</b> 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 <a href="#">Update SSM Agent by using Run Command (p. 997)</a>.</p>	May 3, 2018
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 <a href="#">Running commands using the document version parameter (p. 1002)</a> 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 <a href="#">Deleting custom inventory (p. 887)</a> 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 turning off or deleting a custom inventory type. This new feature uses the <a href="#">DeleteInventory</a> API operation.	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 <a href="#">Subscribing to SSM Agent notifications (p. 137)</a> .	April 9, 2018
CentOS patching support	Systems Manager now supports patching CentOS instances. For information about supported CentOS versions, see <a href="#">Patch Manager prerequisites (p. 1094)</a> . For more information about how patching works, see <a href="#">How Patch Manager operations work (p. 1096)</a> .	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, <a href="#">AWS Systems Manager reference (p. 1534)</a> . Additional content will be added to this section as it becomes available.	March 15, 2018
New topic	The new topic <a href="#">About package name formats for approved and rejected patch lists (p. 1161)</a> 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 <a href="#">Chef InSpec</a> . InSpec is an open-source, runtime framework that allows 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

Change	Description	Release date
	information, see <a href="#">Using Chef InSpec profiles with Systems Manager Compliance</a> (p. 1498).	
New topic	The new topic <a href="#">Using service-linked roles for Systems Manager</a> (p. 1410) 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.	February 27, 2018
New and updated topics	<p>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:</p> <ul style="list-style-type: none"> <li>• <a href="#">How to specify an alternative patch source repository (Linux)</a> (p. 1101)</li> <li>• <a href="#">Working with custom patch baselines (console)</a> (p. 1194)</li> <li>• <a href="#">Create a patch baseline with custom repositories for different OS versions</a> (p. 1212)</li> </ul> <p>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:</p> <ul style="list-style-type: none"> <li>• <a href="#">How security patches are selected</a> (p. 1097)</li> <li>• <a href="#">How patches are installed</a> (p. 1103)</li> <li>• <a href="#">How patch baseline rules work on SUSE Linux Enterprise Server</a> (p. 1121)</li> </ul>	February 6, 2018
New topic	The new topic <a href="#">Upgrading the Python requests module on Amazon Linux instances that use a proxy server</a> (p. 119) provides instructions for ensuring that instances created using an Amazon Linux AMI have been updated with a current version of the Python <code>requests</code> module. This requirement is to ensure compatibility with Patch Manager.	January 12, 2018
New topic	The new topic <a href="#">About SSM documents for patching managed nodes</a> (p. 1124) describes the seven SSM documents available to help you keep your managed instances patched with the latest security-related updates.	January 10, 2018

Change	Description	Release date
Important updates regarding Linux support	Updated various topics with the following information: <ul style="list-style-type: none"> <li>SSM Agent is installed, by default, on Amazon Linux <i>base AMIs</i> dated 2017.09 and later.</li> <li>Manually install SSM Agent on other versions of Linux, including non-base images like <i>Amazon ECS-Optimized AMIs</i>.</li> </ul>	January 9, 2018
New topic	A new topic, <a href="#">About the AWS-RunPatchBaseline SSM document (p. 1128)</a> , 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 AWS-RunPatchBaseline document, Operation and Snapshot ID.	January 5, 2018
New topics	A new section, <a href="#">How Patch Manager operations work (p. 1096)</a> , 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	January 2, 2018
Retitled and moved the Systems Manager Automation Actions Reference	Based on customer feedback, the Automation actions reference is now called the Systems Manager Automation runbook reference. Furthermore, we moved the reference into the Shared Resources > Documents node so it is closer to the <a href="#">Systems Manager Command document plugin reference (p. 1314)</a> . For more information, see <a href="#">Systems Manager Automation actions reference (p. 477)</a> .	December 20, 2017
New Monitoring chapter and content	A new chapter, <a href="#">Monitoring AWS Systems Manager (p. 1428)</a> , provides instructions for sending metrics and log data to Amazon CloudWatch Logs. A new topic, <a href="#">Sending node logs to CloudWatch Logs (CloudWatch agent) (p. 1429)</a> , provides instructions for migrating on-instance monitoring tasks, on 64-bit Windows Server instances only, from SSM Agent to the CloudWatch agent.	December 14, 2017
New chapter	A new chapter, <a href="#">Identity and access management for AWS Systems Manager (p. 1380)</a> , provides comprehensive information about using <a href="#">AWS Identity and Access Management (IAM)</a> 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.	December 11, 2017
Changes to the left navigation	We changed the headings in the left navigation of this user guide to match the headings in the new <a href="#">AWS Systems Manager console</a> .	December 8, 2017

Change	Description	Release date
Multiple changes for re:Invent 2017	<ul style="list-style-type: none"> <li>• <b>Official launch of AWS Systems Manager:</b> AWS Systems Manager (formerly Amazon EC2 Systems Manager) is a unified interface that allows you to centralize operational data and automate tasks across your AWS resources. You can access the new AWS Systems Manager console <a href="#">here</a>. For more information, see <a href="#">What is AWS Systems Manager? (p. 1)</a></li> <li>• <b>YAML Support:</b> You can create SSM documents in YAML. For more information, see <a href="#">AWS Systems Manager documents (p. 1287)</a>.</li> </ul>	November 29, 2017
Using Run Command to Take VSS-Enabled Snapshots of EBS Volumes	<p>Using Run Command, you can take application-consistent snapshots of all <a href="#">Amazon Elastic Block Store (Amazon EBS)</a> volumes attached to your Amazon EC2 Windows instances. The snapshot process uses the Windows <a href="#">Volume Shadow Copy Service (VSS)</a> 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 <a href="#">Take Microsoft VSS-Enabled Snapshots Using AWS Systems Manager</a> in the <a href="#">Amazon EC2 User Guide for Windows Instances</a>.</p>	November 20, 2017
Enhanced Systems Manager Security Available By Using VPC Endpoints	<p>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 allows 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 <a href="#">(Optional) Create a VPC endpoint (p. 28)</a>.</p>	November 7, 2017

Change	Description	Release date
Inventory Support for Files, Services, Windows Roles, and the Windows Registry	<p>SSM Inventory now supports gathering the following information from your managed instances.</p> <ul style="list-style-type: none"> <li>• <b>Files:</b> Name, size, version, installed date, modification and last accessed times, and so on.</li> <li>• <b>Services:</b> Name, display name, status, dependent services, service type, start type, and so on.</li> <li>• <b>Windows Registry:</b> Registry key path, value name, value type, and value.</li> <li>• <b>Windows roles:</b> Name, display name, path, feature type, installed state, and so on.</li> </ul> <p>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 <a href="#">Walkthrough: Automatically update SSM Agent (CLI) (p. 1089)</a>.</p> <p>For more information Inventory, see <a href="#">Learn more about Systems Manager Inventory (p. 851)</a>.</p>	November 6, 2017
Updates to Automation documentation	Fixed several issues in the information about setting up and configuring access for Systems Manager Automation. For more information, see <a href="#">Setting up Automation (p. 401)</a> .	October 31, 2017

Change	Description	Release date
GitHub and Amazon S3 Integration	<p><b>Run remote scripts:</b> 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 <i>infrastructure as code</i> 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 <a href="#">Running scripts from GitHub (p. 1493)</a> and <a href="#">Running scripts from Amazon S3 (p. 1485)</a>.</p> <p><b>Create composite SSM documents:</b> 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 <i>composite</i> documents. Composite documents allow 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 <a href="#">Creating composite documents (p. 1358)</a> and <a href="#">Running Systems Manager Command documents from remote locations (p. 1373)</a>.</p> <p><b>SSM document plugin reference:</b> 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 <a href="#">Systems Manager Command document plugin reference (p. 1314)</a>.</p>	October 26, 2017
Support for Parameter Versions in Parameter Store	<p>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.</p> <p>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 <a href="#">Working with parameter versions (p. 338)</a>.</p>	October 24, 2017

Change	Description	Release date
Support for Tagging Systems Manager Documents	<p>You can now use the <a href="#">AddTagsToResource</a> 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 <a href="#">Tagging Systems Manager documents (p. 1506)</a>.</p>	October 3, 2017
Various Documentation Updates to Fix Errors or Update Content Based on Feedback	<ul style="list-style-type: none"> <li>• Updated <a href="#">Setting up AWS Systems Manager for hybrid environments (p. 34)</a> with information for Raspbian Linux.</li> <li>• Updated <a href="#">Setting up AWS Systems Manager for EC2 instances (p. 16)</a> 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 <a href="#">Windows Management Framework 3.0</a>.</li> </ul>	October 2, 2017
Troubleshoot Unreachable Windows Instances by Using the EC2Rescue Automation Workflow	<p>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 <a href="#">AWSSupport-ExecuteEC2Rescue</a> document. The <a href="#">AWSSupport-ExecuteEC2Rescue</a> 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 <a href="#">Walkthrough: Run the EC2Rescue tool on unreachable instances (p. 677)</a>.</p>	September 29, 2017
SSM Agent Installed By Default on Amazon Linux	<p>SSM Agent is installed, by default, on Amazon Linux AMIs dated 2017.09 and later. Manually install SSM Agent on other versions of Linux, as described in <a href="#">Working with SSM Agent on EC2 instances for Linux (p. 76)</a>.</p>	September 27, 2017
Run Command Enhancements	<p>Run Command includes the following enhancements.</p> <ul style="list-style-type: none"> <li>• 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 <a href="#">Restricting Run Command access based on tags (p. 993)</a>.</li> <li>• 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 <a href="#">Using targets and rate controls to send commands to a fleet (p. 1003)</a>.</li> </ul>	September 12, 2017
Systems Manager Supported on Raspbian	<p>Systems Manager can now run on Raspbian Jessie and Raspbian Stretch devices, including Raspberry Pi (32-Bit).</p>	September 7, 2017

Change	Description	Release date
Automatically Send SSM Agent Logs to Amazon CloudWatch Logs	You can now make a simple configuration change on your instances to have SSM Agent send log files to CloudWatch. For more information, see <a href="#">Sending SSM Agent logs to CloudWatch Logs (p. 1438)</a> .	September 7, 2017
Encrypt resource data sync	With Systems Manager resource data sync, you can 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 <a href="#">Walkthrough: Use resource data sync to aggregate inventory data (p. 903)</a> .	September 1, 2017
New State Manager Walkthroughs	<p>Added two new walkthroughs to the State Manager documentation:</p> <ul style="list-style-type: none"> <li><a href="#">Walkthrough: Automatically update SSM Agent (CLI) (p. 1089)</a></li> <li><a href="#">Walkthrough: Automatically update PV drivers on EC2 instances for Windows Server (console) (p. 1092)</a></li> </ul>	August 31, 2017
Systems Manager Configuration Compliance	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 AWS 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 <a href="#">AWS Systems Manager Compliance (p. 836)</a> .	August 28, 2017
New Automation Action: <code>aws:executeAutomation</code>	Runs a secondary Automation workflow by calling a secondary Automation runbook. With this action, you can create Automation runbooks for your most common workflows, and reference those documents during an Automation execution. This action can simplify your Automation runbooks by removing the need to duplicate steps across similar runbooks. For more information, see <a href="#">aws:executeAutomation – Run another automation (p. 505)</a> .	August 22, 2017
Automation as the Target of a CloudWatch Event	You can start an Automation workflow by specifying an Automation runbook 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 <a href="#">Running automations with triggers using EventBridge (p. 436)</a> .	August 21, 2017
State Manager Association Versioning and General Updates	You can now create different State Manager association versions. There is a quota 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 <a href="#">AWS Systems Manager State Manager (p. 1034)</a> .	August 21, 2017

Change	Description	Release date
Changes to Maintenance Windows	<p>Maintenance Windows include the following changes or enhancements:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• You can edit the targets of a maintenance window, specify a target name, description, and owner.</li> <li>• You can edit tasks in a maintenance window, including specifying a new SSM document for Run Command and Automation tasks.</li> <li>• All Run Command parameters are now supported, including DocumentHash, DocumentHashType, TimeoutSeconds, Comment, and NotificationConfig.</li> <li>• You can now use a <code>safe</code> flag when you attempt to deregister a target. If turned on, the system returns an error if the target is referenced by any task.</li> </ul> <p>For more information, see <a href="#">AWS Systems Manager Maintenance Windows (p. 706)</a>.</p>	August 16, 2017
New Automation Action: <code>aws:approve</code>	<p>This new action for Automation runbooks 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.</p> <p>For more information, see <a href="#">Systems Manager Automation actions reference (p. 477)</a>.</p>	August 10, 2017
Automation Assume Role No Longer Required	<p>Automation previously required that you specify a service role (or <i>assume role</i>) 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.</p> <p>However, the following situations still require that you specify a service role for Automation:</p> <ul style="list-style-type: none"> <li>• When you want to restrict a user's permissions on a resource, but you want the user to run an Automation workflow that requires elevated permissions. In this scenario, you can create a service role with elevated permissions and allow the user to run the workflow.</li> <li>• Operations that you expect to run longer than 12 hours require a service role.</li> </ul> <p>For more information, see <a href="#">Setting up Automation (p. 401)</a>.</p>	August 3, 2017

Change	Description	Release date
Configuration Compliance	<p>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 AWS Regions, and then drill down into specific resources that aren't compliant. For more information, see <a href="#">AWS Systems Manager Compliance (p. 836)</a>.</p>	August 8, 2017
SSM Document Enhancements	<p>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 allows 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.</p> <p>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 <code>aws:runRunShellScript</code> plugin multiple times.</p> <p>For more information about schema version 2.2 changes, see <a href="#">AWS Systems Manager documents</a>. For more information about SSM plugins, see <a href="#">Systems Manager Command document plugin reference</a>.</p>	July 12, 2017
Linux Patching	<p>Patch Manager can now patch the following Linux distributions:</p> <p><b>64-bit and 32-bit systems</b></p> <ul style="list-style-type: none"> <li>Amazon Linux 2014.03, 2014.09, or later</li> <li>Ubuntu Server 16.04 LTS, 14.04 LTS, or 12.04 LTS</li> <li>Red Hat Enterprise Linux (RHEL) 6.5 or later</li> </ul> <p><b>64-bit systems only</b></p> <ul style="list-style-type: none"> <li>Amazon Linux 2015.03, 2015.09, or later</li> <li>Red Hat Enterprise Linux (RHEL) 7.x or later</li> </ul> <p>For more information, see <a href="#">AWS Systems Manager Patch Manager (p. 1093)</a>.</p> <p><b>Note</b></p> <ul style="list-style-type: none"> <li>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 <i>Example: Update SSM Agent in Running commands from the console (p. 996)</i>.</li> <li>The <code>AWS-ApplyPatchBaseline</code> SSM document is being replaced by the <code>AWS-RunPatchBaseline</code> document.</li> </ul>	July 6, 2017

Change	Description	Release date
Resource data sync	<p>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 <a href="#">Configuring resource data sync for Inventory (p. 858)</a>. For an example of how to work with resource data sync, see <a href="#">Walkthrough: Use resource data sync to aggregate inventory data (p. 903)</a>.</p>	June 29, 2017
Systems Manager Parameter Hierarchies	<p>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:</p> <p>/Environment/Type of computer/Application/Data</p> <pre data-bbox="592 910 1307 952">/Dev/DBServer/MySQL/db-string13</pre> <p>For more information, see <a href="#">Working with parameter hierarchies (p. 326)</a>. For an example of how to work with parameter hierarchies, see <a href="#">Working with parameter hierarchies (p. 326)</a>.</p>	June 22, 2017
SSM Agent Support for SUSE Linux Enterprise Server	<p>You can install SSM Agent on 64-bit SUSE Linux Enterprise Server (SLES). For more information, see <a href="#">Working with SSM Agent on EC2 instances for Linux (p. 76)</a>.</p>	June 14, 2017

# Document conventions

The following are the common typographical conventions for the *AWS Systems Manager User Guide*. For conventions common to all AWS documentation, see [Document conventions](#) in the *Amazon Web Services General Reference*.

## Differentiated examples for local operating systems or command line languages

Formatting: Tabs to present commands based on a user's local operating system type. The backslash (\ ) character is used to break long commands into multiple lines on Linux and macOS machines. On Windows Server, line breaks are made using the caret (^) character.

Example:

Linux & macOS

```
aws ssm update-service-setting \
 --setting-id arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier \
 --setting-value advanced
```

Windows

```
aws ssm update-service-setting ^
 --setting-id arn:aws:ssm:region:aws-account-id:servicesetting/ssm/managed-
instance/activation-tier ^
 --setting-value advanced
```

# AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.