aws

User Guide

AWS Systems Manager



Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Systems Manager: User Guide

Copyright © 2025 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 can Systems Manager benefit my operations? 1
Who should use Systems Manager? 2
What are the main features of Systems Manager? 3
Supported AWS Regions 3
Accessing Systems Manager 4
Systems Manager service name history 5
Supported operating systems and machine types 6
Supported operating systems for Systems Manager 6
Supported machine types in hybrid and multicloud environments 10
What is the unified console? 11
Setting up managed nodes for AWS Systems Manager 16
Managing EC2 instances with Systems Manager 16
Configure instance permissions required for Systems Manager 17
Improve the security of EC2 instances by using VPC endpoints for Systems Manager 28
Managing nodes in hybrid and multicloud environments with Systems Manager 32
Create the IAM service role required for Systems Manager in hybrid and multicloud environments 35
Create a hybrid activation to register nodes with Systems Manager 43
Install SSM Agent on hybrid Linux nodes 49
Install SSM Agent on hybrid Windows Server nodes 55
Managing edge devices with Systems Manager 60
Create an IAM service role for your edge devices 61
Configure your edge devices for AWS IoT Greengrass 67
Update the AWS IoT Greengrass token exchange role and install SSM Agent on your edge devices 67
Creating an AWS Organizations delegated administrator for Systems Manager 68
Using a delegated administrator with Change Manager 69
Using a delegated administrator with Explorer 69
Using a delegated administrator with OpsCenter 70
Using a delegated administrator with Quick Setup 70
General setup 70
Sign up for an AWS account 70
Create a user with administrative access 71

Setting up AWS Systems Manager 73
Setting up Systems Manager console access 73
Systems Manager onboarding policy 73
AWS Systems Manager console operator policy 79
AWS Systems Manager console operator read-only policy 83
Setting up Systems Manager unified console for an organization 85
Prerequisites 86
Unified console resources 86
Set up the unified console 89
Setting up Systems Manager unified console for a single account and Region 91
Unified console resources 91
Set up the unified console 93
Disabling the Systems Manager unified console 94
Performing node management tasks 96
Reviewing node insights 96
Adding or removing widgets 99
Rearranging widgets 100
Exploring nodes 100
Exploring nodes using console filters 101
Exploring nodes using text prompts in Amazon Q 107
Viewing individual node details and taking action on a node 113
Downloading or exporting a managed node report 115
Managing node report content and appearance 119
Just-in-time node access using Systems Manager 119
Setting up just-in-time access with Systems Manager 121
Start a just-in-time node access session 169
Managing just-in-time access requests 171
Moving to just-in-time node access from Session Manager 172
Disabling just-in-time access with Systems Manager 177
FAQ 177
Diagnosing and remediating 179
Diagnosing and remediating failed deployments 180
Diagnosing and remediating drifted configurations 181
Diagnosing and remediating unmanaged Amazon EC2 instances 182
Remediation impact types of runbook actions 189
Viewing execution history details for remediations 192

Adjusting Systems Manager settings ... 193
Account setup settings ... 193
Organizational setup settings ... 193
Feature configurations ... 194
Diagnose and remediate settings ... 197
Working with Amazon S3 buckets and bucket policies for Systems Manager ... 200
Using AWS Systems Manager tools ... 209
Node tools ... 209
Change Management tools ... 211
Application tools ... 213
Operations tools ... 213
Perform a node task with Systems Manager ... 215
Prerequisites ... 215
Launch an instance using an AMI with SSM Agent preinstalled ... 215
Connect to your managed instance using Systems Manager ... 216
Clean up your instance ... 217
Node tools ... 217
Compliance ... 218
Distributor ... 237
Fleet Manager ... 290
Hybrid Activations ... 376
Inventory ... 377
Patch Manager ... 472
Run Command ... 760
Session Manager ... 827
State Manager ... 974
Change management tools ... 1072
Automation ... 1072
Change Calendar ... 1548
Change Manager ... 1567
Documents ... 1650
Maintenance Windows ... 1792
Quick Setup ... 1922
Application tools ... 213
AWS AppConfig ... 1971
Application Manager ... 1971

Parameter Store ... 2017
Operations tools ... 2173
Incident Manager ... 2173
Explorer ... 2174
OpsCenter ... 2214
CloudWatch Dashboards ... 2299
Working with SSM Agent ... 2300
Learn technical details about the SSM Agent ... 2300
SSM Agent version 3.2.x.x credential behavior ... 2301
SSM Agent credentials precedence ... 2301
Configuring SSM Agent for use with the Federal Information Processing Standard (FIPS) 2303
About the local ssm-user account ... 2305
SSM Agent and the Instance Metadata Service (IMDS) ... 2306
Keeping SSM Agent up-to-date ... 2306
Ensuring that the SSM Agent installation directory is not modified, moved, or deleted ... 2306
SSM Agent rolling updates by AWS Regions ... 2307
SSM Agent communications with AWS managed S3 buckets ... 2307
SSM Agent on GitHub ... 2314
Understanding SSM Agent hibernation ... 2314
Find AMIs with the SSM Agent preinstalled ... 2317
Verify the status of SSM Agent ... 2318
Working with SSM Agent on EC2 instances for Linux ... 2322
Verifying the signature of SSM Agent ... 2322
Manually installing and uninstalling SSM Agent on EC2 instances for Linux ... 2334
Configuring SSM Agent to use a proxy on Linux nodes ... 2368
Working with SSM Agent on EC2 instances for macOS ... 2373
Manually installing and uninstalling SSM Agent on EC2 instances for macOS ... 2374
Working with SSM Agent on EC2 instances for Windows Server ... 2376
Manually installing and uninstalling SSM Agent on EC2 instances for Windows Server ... 2377
Configure SSM Agent to use a proxy for Windows Server instances ... 2379
Checking SSM Agent status and starting the agent ... 2383
Checking the SSM Agent version number ... 2384
Viewing SSM Agent logs ... 2388
Allowing SSM Agent debug logging ... 2389
Restricting access to root-level commands through SSM Agent ... 2391
Automating updates to SSM Agent ... 2392

Automatically updating SSM Agent ... 2394
Subscribing to SSM Agent notifications ... 2395
Troubleshooting SSM Agent ... 2396
SSM Agent is out of date ... 2397
Troubleshoot issues using SSM Agent log files ... 2397
Agent log files don't rotate (Windows) ... 2398
Unable to connect to SSM endpoints ... 2398
Verify your VPC configuration ... 2399
Verify your VPC DNS-related attributes ... 2400
Verify ingress rules on endpoint security groups ... 2401
Use `ssm-cli` to troubleshoot managed node availability ... 2401
Security ... 2403
Data protection ... 2404
Data encryption ... 2405
Internetwork traffic privacy ... 2407
Data perimeters ... 2408
AWS service-owned resources accessed by Systems Manager ... 2408
Data perimeter policy considerations ... 2409
Identity and access management ... 2410
Audience ... 2410
Authenticating with identities ... 2411
Managing access using policies ... 2414
How AWS Systems Manager works with IAM ... 2418
Identity-based policy examples ... 2428
AWS managed policies ... 2440
Troubleshooting ... 2500
Using service-linked roles ... 2501
Inventory and Explorer data role ... 2503
OpsCenter and Explorer account discovery role ... 2506
OpsData and OpsItems creation role ... 2509
Operational insights creation role ... 2512
Quick Setup deployment health-check role ... 2516
Export OpsData service role ... 2519
Just-in-time node access service role ... 2521
Just-in-time node access request notifications service role ... 2525
Logging and monitoring ... 2528

Compliance validation ... 2531
Resilience ... 2532
Infrastructure security ... 2532
Configuration and vulnerability analysis ... 2533
Security best practices ... 2533
Systems Manager preventative security best practices ... 2533
SSM Agent installation best practices ... 2537
Systems Manager monitoring and auditing best practices ... 2538
Code examples ... 2540
Basics ... 2547
Hello Systems Manager ... 2551
Learn the basics ... 2555
Actions ... 2606
Logging and monitoring ... 2941
Monitoring tools ... 2942
Sending node logs to unified CloudWatch Logs (CloudWatch agent) ... 2942
Migrate Windows Server node log collection to the CloudWatch agent ... 2944
Store CloudWatch agent configuration settings in Parameter Store ... 2954
Rolling back to log collection with SSM Agent ... 2955
Sending SSM Agent logs to CloudWatch Logs ... 2958
Monitoring your change request events ... 2961
Monitoring your automations ... 2964
Automation metrics ... 2965
Monitoring Run Command metrics using Amazon CloudWatch ... 2965
Systems Manager Run Command metrics and dimensions ... 2966
Logging AWS Systems Manager API calls with AWS CloudTrail ... 2967
Systems Manager data events in CloudTrail ... 2969
Systems Manager management events in CloudTrail ... 2970
Systems Manager event examples ... 2971
Logging Automation action output with CloudWatch Logs ... 2976
Configuring Amazon CloudWatch Logs for Run Command ... 2980
Specifying CloudWatch Logs when you send commands ... 2981
Viewing command output in CloudWatch Logs ... 2982
Monitoring with Amazon EventBridge ... 2982
Configuring EventBridge for Systems Manager events ... 2984
Amazon EventBridge event examples for Systems Manager ... 2987

Sample scenarios: Systems Manager targets in Amazon EventBridge rules ... 3003
Monitoring Systems Manager status changes using Amazon SNS notifications ... 3005
Configure Amazon SNS notifications for AWS Systems Manager ... 3006
Example Amazon SNS notifications for AWS Systems Manager ... 3017
Use Run Command to send a command that returns status notifications ... 3019
Use a maintenance window to send a command that returns status notifications ... 3022
Product and service integrations ... 3028
Integration with AWS services ... 3028
Compute ... 3028
Internet of Things (IoT) ... 3031
Storage ... 3032
Developer Tools ... 3033
Security, Identity, and Compliance ... 3034
Cryptography and PKI ... 3037
Management and Governance ... 3037
Networking and Content Delivery ... 3042
Analytics ... 3043
Application Integration ... 3045
AWS Management Console ... 3045
Running scripts from Amazon S3 ... 3046
Referencing AWS Secrets Manager secrets from Parameter Store parameters ... 3050
AWS KMS encryption for Parameter Store SecureString parameters ... 3056
Use AWS Secrets Manager secrets in Amazon Elastic Kubernetes Service ... 3070
Using Parameter Store parameters in AWS Lambda functions ... 3089
Integration with other products and services ... 3111
Running scripts from GitHub ... 3115
Using Chef InSpec profiles with Systems Manager Compliance ... 3123
Integrating with ServiceNow ... 3128
AWS Systems Manager reference ... 3130
Working with AWS SDKs ... 3131
Amazon S3 buckets for patching operations ... 3132
Buckets containing SSM Command documents for patching operations (Linux and Windows Server) ... 3133
Buckets containing SSM Command documents for patching operations (macOS) ... 3136
Buckets containing AWS managed patch baseline snapshots ... 3139
EventBridge event patterns and types for Systems Manager ... 3141

Event type: Automation ... 3142
Event type: Change Calendar ... 3143
Event type: Change Manager ... 3144
Event type: Configuration Compliance ... 3144
Event type: Inventory ... 3144
Event type: Maintenance Window ... 3145
Event type: OpsCenter ... 3148
Event type: Parameter Store ... 3148
Event type: Run Command ... 3149
Event type: State Manager ... 3150
Cron and rate expressions ... 3150
General information about cron and rate expressions ... 3151
Cron and rate expressions for associations ... 3156
Cron and rate expressions for maintenance windows ... 3159
ec2messages, ssmmessages, and other API operations ... 3161
Agent-related API operations (ssmmessages and ec2messages endpoints) ... 3162
ssm:* namespace instance-related API operations ... 3164
ssm:* namespace other API operations ... 3165
Date and time string formats for Systems Manager ... 3166
Formatting date and time strings for Systems Manager ... 3166
Creating custom date and time strings for Systems Manager ... 3167
Use cases and best practices ... 3170
Deleting Systems Manager resources and artifacts ... 3173
Choosing between State Manager and Maintenance Windows ... 3178
State Manager and Maintenance Windows: Key use cases ... 3178
Related information ... 3185
Document history ... 3187
Updates prior to June 2018 ... 3388
Document conventions ... 3407

What is AWS Systems Manager?

AWS Systems Manager helps you centrally view, manage, and operate nodes at scale in AWS, on-premises, and multicloud environments. With the launch of a unified console experience, Systems Manager consolidates various tools to help you complete common node tasks across AWS accounts and AWS Regions.

To use Systems Manager, nodes must be managed, which means SSM Agent is installed on the machine and the agent can communicate with the Systems Manager service. To help you identify why nodes aren't reporting as *managed*, Systems Manager offers a one-click agent issue diagnosis and remediation runbook that you can configure to run automatically according to a schedule you define. This feature helps identify why nodes can't connect to Systems Manager, including networking misconfigurations. This feature also provides recommended runbooks for remediating networking issues and other problems preventing nodes from being configured as managed nodes.

The unified console experience also includes a dashboard that provides a high-level overview of your nodes. You can drill down for more specific node insights such as which nodes are running outdated operating system (OS) software. You can also use filters for granular views based on instance metadata like OSs and OS versions, AWS Regions, AWS accounts, and SSM Agent versions. These filters help you retrieve relevant information at a specific account level or application level across your entire organization.

Topics

- How can Systems Manager benefit my operations?
- Who should use Systems Manager?
- What are the main features of Systems Manager?
- Supported AWS Regions
- Accessing Systems Manager
- Systems Manager service name history
- Supported operating systems and machine types
- What is the unified console?

How can Systems Manager benefit my operations?

Benefits of Systems Manager include the following:

- **Enhance visibility across your entire infrastructure**

  Systems Manager provides a centralized view of nodes across your organization's accounts and Regions. Quickly access instance information such as ID, name, OS details, and installed agents. Use Amazon Q Developer to query instance metadata using natural language, helping you identify issues and take action faster.

- **Boost operational efficiency with automation**

  Automate common operational tasks and reduce time and effort required to maintain your systems. Systems Manager provides safe and secure remote management of your nodes at scale without logging into your servers. You no longer need to use bastion hosts, SSH, or remote PowerShell. Systems Manager also provides a simple way of automating common administrative tasks across groups of nodes such as registry edits, user management, and software and patch installations.

- **Simplify node management at scale in any environment**

  Systems Manager helps you manage nodes across AWS, on-premises, and multicloud environments. Schedule automated diagnoses to identify SSM Agent issues and remediate them with one-click runbooks. After your nodes are configured as *managed* nodes, you can execute critical operational tasks such as applying security patches, initiating logged sessions, and running commands remotely.

Who should use Systems Manager?

Systems Manager is used by IT operations managers and operators, DevOps engineers, security and compliance managers, and IT directors and CIOs. Broadly speaking, Systems Manager is appropriate for the following:

- Organizations that want to improve the management and security of their nodes at scale.
- Organizations that want to increase visibility and operational agility when managing their infrastructure.
- Organizations that want to increase operational efficiency at scale.

What are the main features of Systems Manager?

The primary features of Systems Manager are shared between the unified console and the individual tools Systems Manager provides to help you manage nodes at scale.

Unified console

The unified console provides a centralized experience to view and manage your nodes. This console leverages several Systems Manager tools and more to provide you with the following:

- Centralized views of your nodes
- Detailed node insights
- Automated diagnosis and remediation of common node issues

For more information about the unified console, see What is the unified console?.

Tools

Tools consist of the individual capabilities of Systems Manager and their features such as Run Command, Session Manager, Automation, and Parameter Store. With Systems Manager tools you can do the following:

- Just-in-time access node access
- Patch nodes at scale
- Securely connect to nodes without opening inbound ports
- Run commands remotely on nodes
- Securely store data referenced by applications
- Automate common systems administration tasks

For more information about Systems Manager tools, see Using AWS Systems Manager tools.

Supported AWS Regions

For a list of AWS Regions that support Systems Manager tools, see Systems Manager service endpoints in the *Amazon Web Services General Reference*.

The unified Systems Manager console, released on November 21, 2024, is available in the following AWS Regions:

- US East (N. Virginia) Region
- US East (Ohio) Region
- US West (N. California) Region
- US West (Oregon) Region
- Canada (Central) Region
- South America (São Paulo) Region
- Asia Pacific (Mumbai) Region
- Asia Pacific (Tokyo) Region
- Asia Pacific (Seoul) Region
- Asia Pacific (Singapore) Region
- Asia Pacific (Sydney) Region
- Europe (Frankfurt) Region
- Europe (Stockholm) Region
- Europe (Ireland) Region
- Europe (London) Region
- Europe (Paris) Region

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 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 grant programmatic access to Systems Manager. For information about the AWS SDKs, including how to download and install them, see Tools for Amazon Web Services.

Systems Manager 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`

Supported operating systems and machine types

Before working with Systems Manager, verify that your operating system (OS), OS version, and machine type are supported as managed nodes.

Topics

- Supported operating systems for Systems Manager
- Supported machine types in hybrid and multicloud environments

Supported operating systems for Systems Manager

The following sections list the OSs and OS versions supported by Systems Manager.

> **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 Managing edge devices with Systems Manager.

> **Important**
>
> Patch Manager, a tool in Systems Manager, might not support all the OS versions listed in this topic. For a list of OS versions supported by Patch Manager, see Patch Manager prerequisites.

Operating system types

- Linux
- macOS (Amazon EC2 instances only)
- Windows Server

Select an OS platform to see the supported major and minor versions.

Linux

AlmaLinux

Versions	x86	x86_64	ARM64
8.3–8.10		✓	✓
9.*x*		✓	✓

Amazon Linux 2

Versions	x86	x86_64	ARM64
2.0 and all later versions		✓	✓

Amazon Linux 2023

Versions	x86	x86_64	ARM64
2023.0.20230315.0 and all later versions		✓	✓

Bottlerocket

Versions	x86_64	ARM64
1.0.0 and all later versions	✓	✓

Debian Server

Versions	x86	x86_64	ARM64
Bullseye (11)		✓	✓
Bookworm (12)		✓	✓

Oracle Linux

Versions	x86	x86_64	ARM64
7.5–7.8		✓	
8.*x*		✓	
9.*x*		✓	

Red Hat Enterprise Linux (RHEL)

Versions	x86	x86_64	ARM64
7.0–7.5		✓	
7.6–8.*x*		✓	✓
9.*x*		✓	✓
10.*x*		✓	✓

Rocky Linux

Versions	x86	x86_64	ARM64
8.*x*		✓	✓
9.*x*		✓	✓

Ubuntu Server

Versions	x86	x86_64	ARM64
16.04 LTS and 18.04 LTS		✓	✓
20.04 LTS		✓	✓
22.04 LTS		✓	✓
24.04 LTS		✓	✓
25.04		✓	✓

macOS (Amazon EC2 instances only)

Version	x86	x86_64	Mac with Apple silicon
13.*x* (Ventura)		✓	✓
14.*x* (Sonoma)		✓	✓
15.*x* (Sequoia)		✓	✓

Note

macOS is not supported in all AWS Regions. For more information about Amazon EC2 support for macOS, see Amazon EC2 Mac instances in the *Amazon EC2 User Guide*.

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
2012 and 2012 R2[2]		✓	
2016		✓	
2019		✓	
2022		✓	
2025		✓	

[1] **Windows Server 2012 and 2012 R2 support**: Windows Server 2012 and 2012 R2 reached end of support on October 10, 2023. To use SSM Agent with these versions, we recommend using Extended Security Updates (ESUs) from Microsoft. For more information, see Windows Server 2012 and 2012 R2 reaching end of support on the Microsoft website.

Supported machine types in hybrid and multicloud environments

Systems Manager supports a number of machine types as *managed nodes*. A managed node is any machine configured to work with Systems Manager.

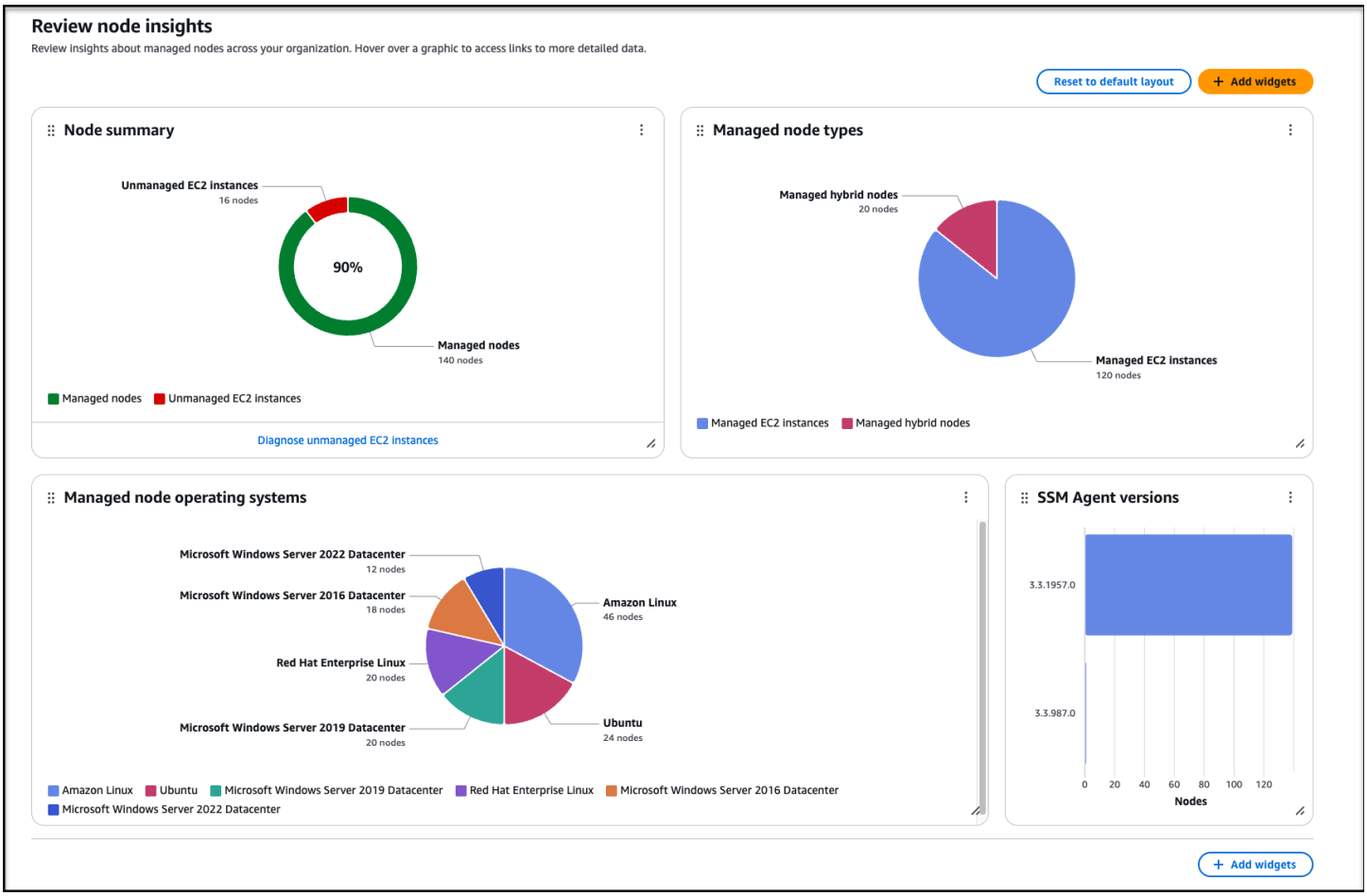
This user guide uses the term *hybrid and multicloud* to refer to an environment that contains any combination of the following machine types:

- Amazon Elastic Compute Cloud (Amazon EC2) instances
- Servers on your own premises (on-premises servers)
- AWS IoT Greengrass core devices
- AWS IoT and non-AWS edge devices
- Virtual machines (VMs), including VMs in other cloud environments

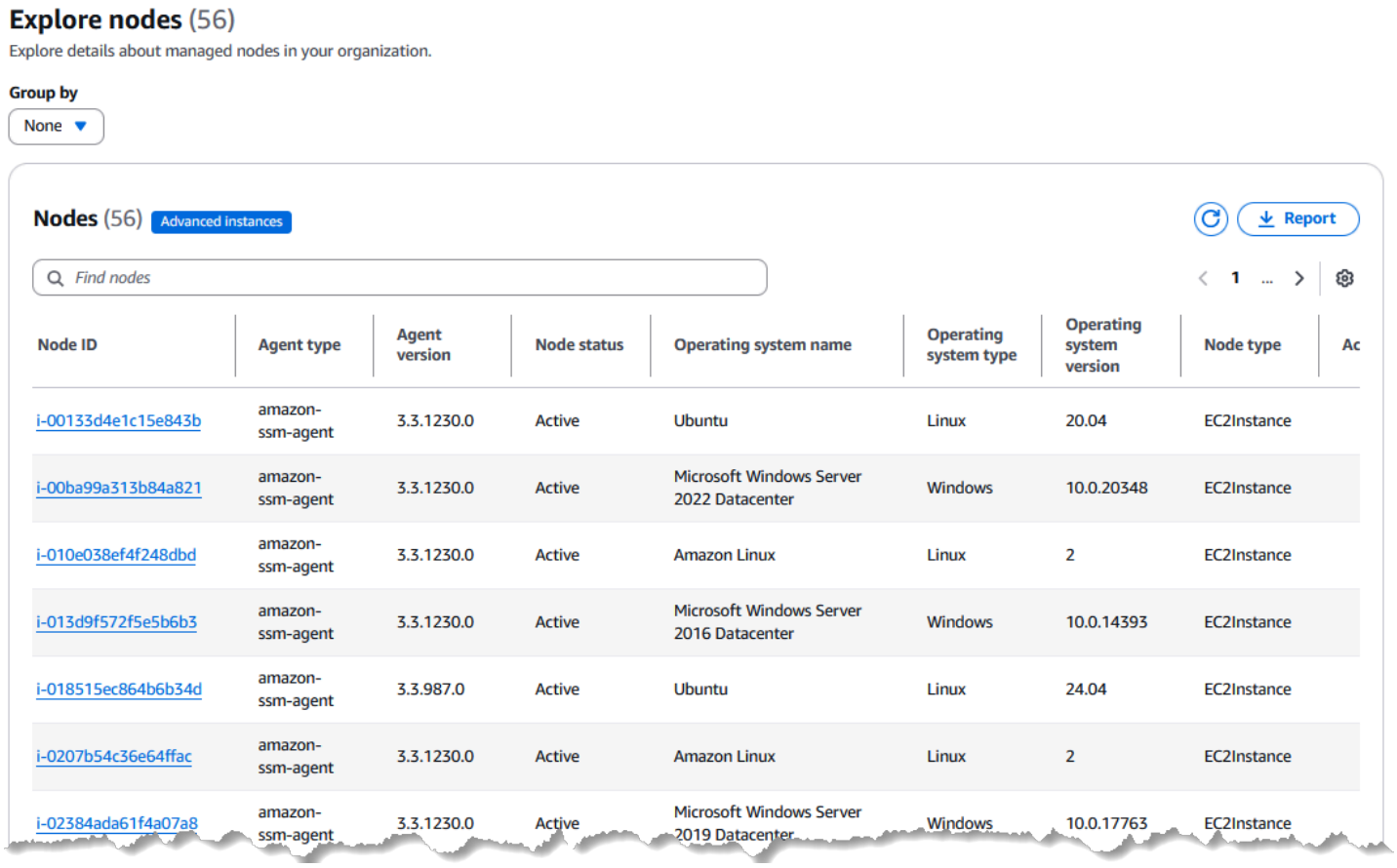
For information about AWS support for hybrid and multicloud environments, see AWS Solutions for Hybrid and Multicloud.

What is the unified console?

The unified Systems Manager console is a consolidated experience that combines various tools to help you complete common node tasks across multiple AWS accounts and AWS Regions in an AWS Organizations organization, or a single account and Region. Nodes can be EC2 instances, hybrid servers, or servers running in a multicloud environment. In the unified console, you're provided with detailed insights to your nodes. You can generate reports for your nodes, diagnose and remediate common issues that prevent nodes from reporting as managed by Systems Manager, like connectivity issues.

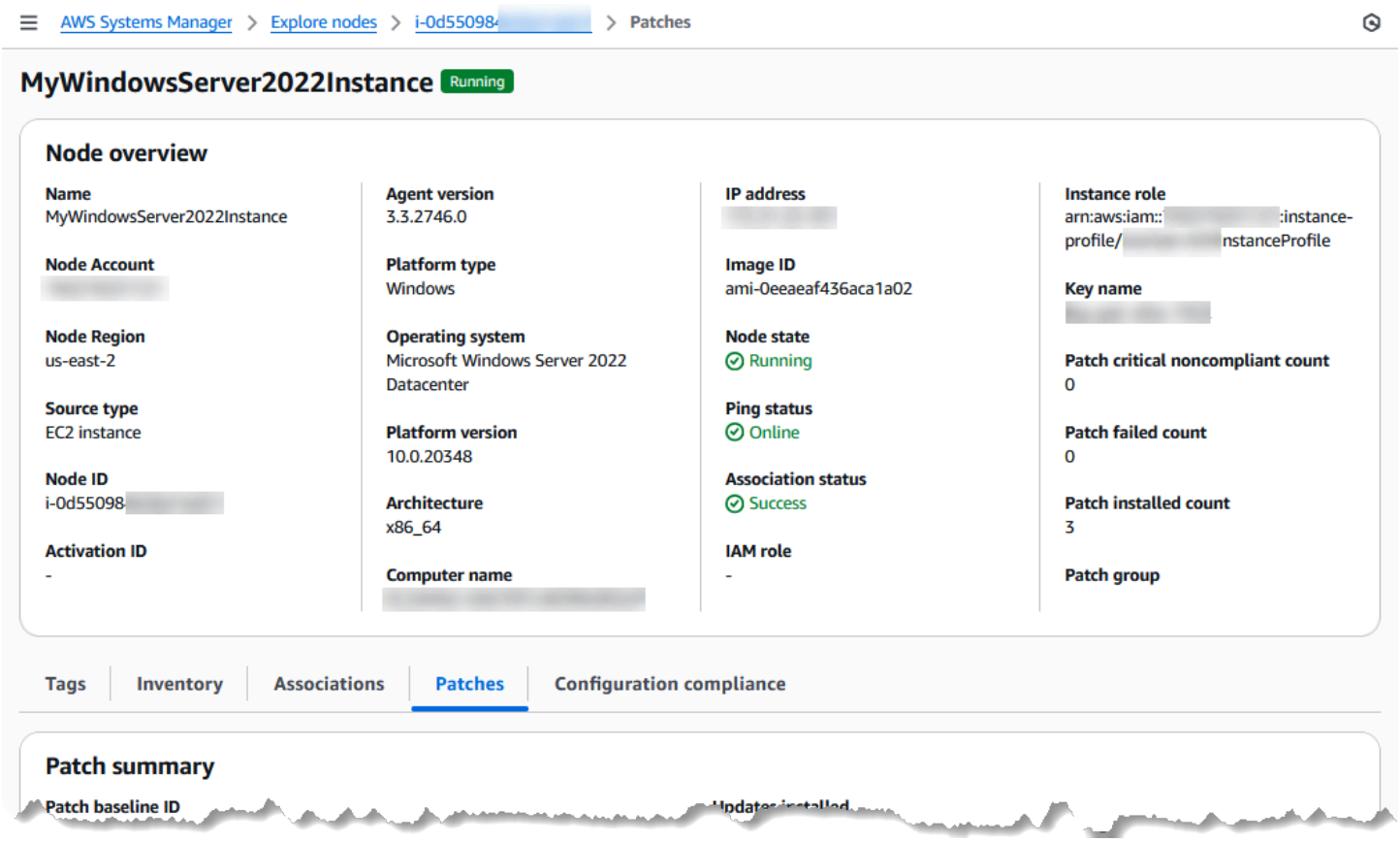


In addition to summaries about your nodes on the **Review node insights** page, you can view specific details about a node from the **Explore nodes** page.



Node details tabs

When you select a node on the **Explore nodes** page, the node detail page provides a comprehensive overview of node details and additional information on a series of tabs:



Tags

(Optional) Manage resource tags to group and filter the managed node with other resources. Tags consist of a case-senstive key-value pair and are used to categorize resources in different ways, such as by purpose, owner, or environment.

Inventory

Displays metadata about the managed node, which you can view according to over 10 different inventory types. For example, when you select the type `AWS:Application`, the inventory filter results provide details about applications installed on the node, such as **Name**, **Version**, **Architecture**, and more. For more information about Inventory, see the section called "Inventory".

Associations

An association is a resource type in State Manager that defines the target state for a managed node and maintains all managed nodes in your account in a consistent state. The association can define the commands, scripts, or policies to apply to which managed instances, and how often the association should run to ensure the nodes are match the defined configuration for

the node. An association can drive compliance reporting of required states for resources in your account. For more information about State Manager and assocations, see the section called "State Manager".

Patches

Displays metadata about the managed node, such as which patch baseline is assigned to the node and the total number of updates for packages that have been updated successfully, failed, or still required for installation. The **Patches** tab also reports details about patches available for the node based on the configuration requirements in the patch baseline, including the package **Name**, such as `libblockdev-crypto.x86_64`; **Classification** (such as `Bugfix` or `Security`); **Description** (showing the full patch title, such as `coreutils.x86_64:0:8.32-36.el9` and `java-11-amazon-corretto-headless-1:11.0.15+9-1.amzn2.x86_64`; and **Patch State**, such as `Installed`, `Installed_Pending_Reboot`, `Missing`, and `Failed`.

> **Note**
>
> Patch *states* do not indicate whether or not a managed node is *compliant*. Patch compliance is not innately tied to patch states, nor is it defined by AWS, by operating system (OS) vendors, or by third parties such as security consulting firms. Instead, you define what patch compliance means for managed nodes in your organization or account in a *patch baseline*. For more information, see the section called "What is compliance in Patch Manager?" and the section called "Predefined and custom patch baselines".

Configuration compliance

Reports patch compliance and configuration inconsistencies on the node (whether the state of a package on the managed node is `Compliant` or `Non-compliant` according to the definition of Compliant as defined in either a State Manager association or a Patch Manager patch baseline). You can filter configuration compliance results according to a package **ID**, **Compliance status**, **Compliance type** (`Association` or `Patch`), `Severity`, and different execution details. For related information, see the section called "Compliance".

Whether you have nodes in multiple accounts and Regions in an organization, or nodes in a single account and Region, we recommend using the unified console. To learn about the node tasks you

can perform now using the unified console, see Performing node management tasks with AWS Systems Manager.

For more information about setting up your nodes for Systems Manager, see Setting up managed nodes for AWS Systems Manager. After you've set up your nodes, you can set up Systems Manager and the unified console. To learn more about setting up Systems Manager, see Setting up AWS Systems Manager.

Setting up managed nodes for AWS Systems Manager

Complete the tasks in this section to set up and configure roles, user accounts, permissions, and initial resources for using AWS Systems Manager tools. 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 use with Systems Manager in a hybrid and multicloud environment.

> **Note**
>
> If you plan to use Amazon EC2 instances *and* your own computing resources in a hybrid and multicloud environment, follow the steps in Managing EC2 instances with Systems Manager. That topic presents steps in the best order for completing Systems Manager setup for EC2 instances and non-EC2 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 tools.

Topics

- Managing EC2 instances with Systems Manager
- Managing nodes in hybrid and multicloud environments with Systems Manager
- Managing edge devices with Systems Manager
- Creating an AWS Organizations delegated administrator for Systems Manager
- General setup for AWS Systems Manager

Managing EC2 instances with Systems Manager

Complete the tasks in this section to set up and configure roles, 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 Managing nodes in hybrid and multicloud environments with Systems Manager. If you plan to use both Amazon EC2 instances *and* non-EC2 machines in a hybrid and multicloud 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 tools.

Contents

- Configure instance permissions required for Systems Manager
- Improve the security of EC2 instances by using VPC endpoints for Systems Manager

Configure instance permissions required for Systems Manager

By default, AWS Systems Manager doesn't have permission to perform actions on your instances. You can provide instance permissions at the account level using an AWS Identity and Access Management (IAM) role, or at the instance level using an instance profile. If your use case allows, we recommend granting access at the account level using the Default Host Management Configuration.

> **Note**
>
> You can skip this step and allow Systems Manager to apply the required permissions to your instances for you when setting up the unified console. For more information, see Setting up AWS Systems Manager.

Recommended configuration for EC2 instance permissions

Default Host Management Configuration allows Systems Manager to manage your Amazon EC2 instances automatically. After you've turned on this setting, all instances using Instance

Metadata Service Version 2 (IMDSv2) in the AWS Region and AWS account with SSM Agent version 3.2.582.0 or later installed automatically become managed instances. Default Host Management Configuration doesn't support Instance Metadata Service Version 1. For information about transitioning to IMDSv2, see Transition to using Instance Metadata Service Version 2 in the *Amazon EC2 User Guide*. For information about checking the version of the SSM Agent installed on your instance, see Checking the SSM Agent version number. For information about updating the SSM Agent, see Automatically updating SSM Agent. Benefits of managed instances include the following:

- Connect to your instances securely using Session Manager.
- Perform automated patch scans using Patch Manager.
- View detailed information about your instances using Systems Manager Inventory.
- Track and manage instances using Fleet Manager.
- Keep the SSM Agent up to date automatically.

Fleet Manager, Inventory, Patch Manager, and Session Manager are tools in AWS Systems Manager.

Default Host Management Configuration allows instance management without the use of instance profiles and ensures that Systems Manager has permissions to manage all instances in the Region and account. If the permissions provided aren't sufficient for your use case, you can also add policies to the default IAM role created by the Default Host Management Configuration. Alternatively, if you don't need permissions for all of the capabilities provided by the default IAM role, you can create your own custom role and policies. Any changes made to the IAM role you choose for Default Host Management Configuration applies to all managed Amazon EC2 instances in the Region and account. For more information about the policy used by Default Host Management Configuration, see AWS managed policy: AmazonSSMManagedEC2InstanceDefaultPolicy. For more information about the Default Host Management Configuration, see Managing EC2 instances automatically with Default Host Management Configuration.

Important

Instances registered using Default Host Management Configuration store registration information locally in the `/lib/amazon/ssm` or `C:\ProgramData\Amazon` directories. Removing these directories or their files will prevent the instance from acquiring the necessary credentials to connect to Systems Manager using Default Host Management

Configuration. In these cases, you must use an instance profile to provide the required permissions to your instance, or recreate the instance.

Note

This procedure is intended to be performed only by administrators. Implement least privilege access when allowing individuals to configure or modify the Default Host Management Configuration. You must turn on the Default Host Management Configuration in each AWS Region you wish to automatically manage your Amazon EC2 instances.

To turn on the Default Host Management Configuration setting

You can turn on the Default Host Management Configuration from the Fleet Manager console. To successfully complete this procedure using either the AWS Management Console or your preferred command line tool, you must have permissions for the GetServiceSetting, ResetServiceSetting, and UpdateServiceSetting API operations. Additionally, you must have permissions for the `iam:PassRole` permission for the `AWSSystemsManagerDefaultEC2InstanceManagementRole` IAM role. The following is an example policy. Replace each *`example resource placeholder`* with your own information.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetServiceSetting",
                "ssm:ResetServiceSetting",
                "ssm:UpdateServiceSetting"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:servicesetting/ssm/
managed-instance/default-ec2-instance-management-role"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/
AWSSystemsManagerDefaultEC2InstanceManagementRole",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ssm.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Before you begin, if you have instance profiles attached to your Amazon EC2 instances, remove any permissions that allow the `ssm:UpdateInstanceInformation` operation. The SSM Agent attempts to use instance profile permissions before using the Default Host Management Configuration permissions. If you allow the `ssm:UpdateInstanceInformation` operation in your instance profiles, the instance will not use the Default Host Management Configuration permissions.

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Fleet Manager**.
3. Choose **Configure Default Host Management Configuration** under the **Account management** dropdown.
4. Turn on **Enable Default Host Management Configuration**.
5. Choose the IAM role used to enable Systems Manager tools for your instances. We recommend using the default role provided by Default Host Management Configuration. It contains the minimum set of permissions necessary to manage your Amazon EC2 instances using Systems Manager. If you prefer to use a custom role, the role's trust policy must allow Systems Manager as a trusted entity.
6. Choose **Configure** to complete setup.

After turning on the Default Host Management Configuration, it might take up 30 minutes for your instances to use the credentials of the role you chose. You must turn on the Default Host

Management Configuration in each Region you wish to automatically manage your Amazon EC2 instances.

Alternative configuration for EC2 instance permissions

You can grant access at the individual instance level 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 tool in 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 3) and Step 4. For more information, see AWS Systems Manager Quick Setup.

Note the following details about creating an IAM instance profile:

- If you're configuring non-EC2 machines in a hybrid and multicloud 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 the IAM service role required for Systems Manager in hybrid and multicloud environments.
- 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.

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. For **Trusted entity type**, choose **AWS service**.

4. Immediately under **Use case**, choose **EC2**, and then choose **Next**.
5. On the **Add permissions** page, do the following:

   - Use the **Search** field to locate the **AmazonSSMManagedInstanceCore** policy. Select the check box next to its name, as shown in the following illustration.



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

   - If you created a custom S3 bucket policy in the previous procedure, [(Optional) Create a custom policy for S3 bucket access](), 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**, update the description for this instance profile.
9. (Optional) For **Tags**, 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:
   - Use the **Search** field to locate the **AmazonSSMManagedInstanceCore** policy. Select the check box next to its name.
   - If you have created a custom S3 bucket policy, search for it and select the check box next to its name. For information about custom S3 bucket policies for an instance profile, see (Optional) Create a custom policy for S3 bucket access.
   - 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.
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*.

(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. You can attach this policy to the default IAM role created by the Default Host Management Configuration, or an instance profile you created in the previous procedure.

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.

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

   JSON

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

```
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::aws-ssm-us-east-2/*",
                "arn:aws:s3:::aws-windows-downloads-us-east-2/*",
                "arn:aws:s3:::amazon-ssm-us-east-2/*",
                "arn:aws:s3:::amazon-ssm-packages-us-east-2/*",
                "arn:aws:s3:::us-east-2-birdwatcher-prod/*",
                "arn:aws:s3:::aws-ssm-distributor-file-us-east-2/*",
                "arn:aws:s3:::aws-ssm-document-attachments-us-east-2/*",
                "arn:aws:s3:::patch-baseline-snapshot-us-east-2/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetEncryptionConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

Note

The first `Statement` element is required only if you're using a VPC endpoint.
The second `Statement` element is required only if you're using an S3 bucket that you created to use in your Systems Manager operations.
The `PutObjectAcl` access control list permission is required only if you plan to support cross-account access to S3 buckets in other accounts.
The `GetEncryptionConfiguration` element is required if your S3 bucket is configured to use encryption.

If your S3 bucket is configured to use encryption, then the S3 bucket root (for example, `arn:aws:s3:::amzn-s3-demo-bucket`) must be listed in the **Resource** section. Your 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 *`amzn-s3-demo-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:::amzn-s3-demo-bucket1/*",
> "arn:aws:s3:::amzn-s3-demo-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**.

Additional policy considerations for managed instances

This section describes some of the policies you can add to the default IAM role created by the Default Host Management Configuration, or your instance profiles 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. Depending on your operations plan, you might need permissions represented in one or more of the other policies.

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 default IAM role created by the Default Host Management Configuration or 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 IAM policies with permissions for additional Systems Manager tools, see the following resources:

- Restricting access to Parameter Store parameters using IAM policies
- Setting up Automation
- Step 2: Verify or add instance permissions for Session Manager

Attach the Systems Manager instance profile to an instance (console)

The following procedure describes how to attach an IAM instance profile to an Amazon EC2 instance using the Amazon EC2 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 Alternative configuration for EC2 instance permissions.
6. Choose **Update IAM role**.

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*
- Attach an IAM role to an instance in the *Amazon EC2 User Guide*

Continue to Improve the security of EC2 instances by using VPC endpoints for Systems Manager.

Improve the security of EC2 instances by using VPC endpoints for Systems Manager

You can improve the security posture of your managed nodes (including non-EC2 machines in a hybrid and multicloud 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.

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
- Creating VPC endpoints for Systems Manager
- Create an interface VPC endpoint policy

VPC endpoint restrictions and limitations

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

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.

DNS resolution

If you use a custom DNS server, you must add a conditional forwarder for any queries to the `amazonaws.com` domain to the Amazon DNS server for your VPC.

S3 buckets

Your VPC endpoint policy must allow access to at least the Amazon S3 buckets listed in SSM Agent communications with AWS managed S3 buckets.

> **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.

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 and multicloud environment

For information about configuring DNS to work with AWS PrivateLink endpoints in hybrid and multicloud 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 endpoints for AWS Systems Manager. This topic links to procedures in the *Amazon VPC User Guide*.

> **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*.

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. Beginning with version 3.3.40.0 of SSM Agent, Systems Manager began using the `ssmmessages:*` endpoint (Amazon Message Gateway Service) whenever available instead of the `ec2messages:*` endpoint (Amazon Message Delivery 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*.s3** – Systems Manager uses this endpoint to update SSM Agent. Systems Manager also uses this endpoint if, optionally, you choose to retrieve scripts or other files stored in buckets or upload output logs to a bucket. 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*.
- **com.amazonaws.*region*.ssmmessages** – This endpoint is required for SSM Agent to communicate with the Systems Manager service, for Run Command, and if you're connecting to

your instances through a secure data channel using Session Manager. For more information, see AWS Systems Manager Session Manager and Reference: ec2messages, ssmmessages, and other API operations.

- (Optional) **com.amazonaws.*region*.kms** – Create this endpoint if you want to use AWS Key Management Service (AWS KMS) encryption for Session Manager or Parameter Store parameters.
- (Optional) **com.amazonaws.*region*.logs** – Create this endpoint if you want to use Amazon CloudWatch Logs (CloudWatch Logs) for Session Manager, Run Command, or SSM Agent logs.

For information about the AWS managed S3 buckets that SSM Agent must be able to access, see SSM Agent communications with AWS managed S3 buckets. If you're using a virtual private cloud (VPC) endpoint in your Systems Manager operations, you must provide explicit permission in an EC2 instance profile for Systems Manager, or in a service role for non-EC2 managed nodes in a hybrid and multicloud environment.

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*.

Managing nodes in hybrid and multicloud environments with Systems Manager

You can use AWS Systems Manager to manage both Amazon Elastic Compute Cloud (EC2) instances and a number of non-EC2 machine types. This section describes the setup tasks that account and system administrators perform to manage non-EC2 machines using Systems Manager in a *hybrid and multicloud environment*. 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 non-EC2 machines.

Any machine that has been configured for use with Systems Manager is called a *managed node*.

Note

- You can register edge devices as managed nodes using the same hybrid-activation steps used for other non-EC2 machines. These types of edge devices include both AWS IoT devices and devices other than AWS IoT devices. Use the process described in this section to set up these types of edge devices.

  Systems Manager also supports edge devices that use AWS IoT Greengrass Core software. The setup process and requirements for AWS IoT Greengrass core devices are different from those for AWS IoT and edge devices other than AWS edge devices. For information about registering AWS IoT Greengrass devices for use with Systems Manager, see Managing edge devices with Systems Manager.
- Non-EC2 macOS machines aren't supported for Systems Manager hybrid and multicloud 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 non-EC2 machines in hybrid and multicloud environment, follow the steps in Managing EC2 instances with Systems Manager first.

After configuring your hybrid and multicloud environment for Systems Manager, you can do the following:

- Create a consistent and secure way to remotely manage your hybrid and multicloud 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 of the operations performed on your machines by viewing the API activity recorded in AWS CloudTrail.

  For information about using CloudTrail to monitor Systems Manager actions, see Logging AWS Systems Manager API calls with AWS CloudTrail.
- 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.

About managed nodes

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

A managed node is any machine configured for Systems Manager. Previously, managed nodes were all referred to as managed instances. The term *instance* now refers to EC2 instances only. The deregister-managed-instance command was named before this terminology change.

For more information, see Working with managed nodes.

About instance tiers

Systems Manager offers a standard-instances tier and an advanced-instances tier for non-EC2 managed nodes in your hybrid and multicloud environment. The standard-instances tier allows you to register a maximum of 1,000 hybrid-activated machines per AWS account per AWS Region. If you need to register more than 1,000 non-EC2 machines in a single account and Region, then use the advanced-instances tier. Advanced instances also allow you to connect to your non-EC2 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.

Topics

- Create the IAM service role required for Systems Manager in hybrid and multicloud environments
- Create a hybrid activation to register nodes with Systems Manager
- Install SSM Agent on hybrid Linux nodes
- Install SSM Agent on hybrid Windows Server nodes

Create the IAM service role required for Systems Manager in hybrid and multicloud environments

Non-EC2 (Amazon Elastic Compute Cloud) machines in a hybrid and multicloud environment require an AWS Identity and Access Management (IAM) service 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 and multicloud environment once for each AWS account. However, you might choose to create multiple service roles for different hybrid activations if machines in your hybrid and multicloud 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.

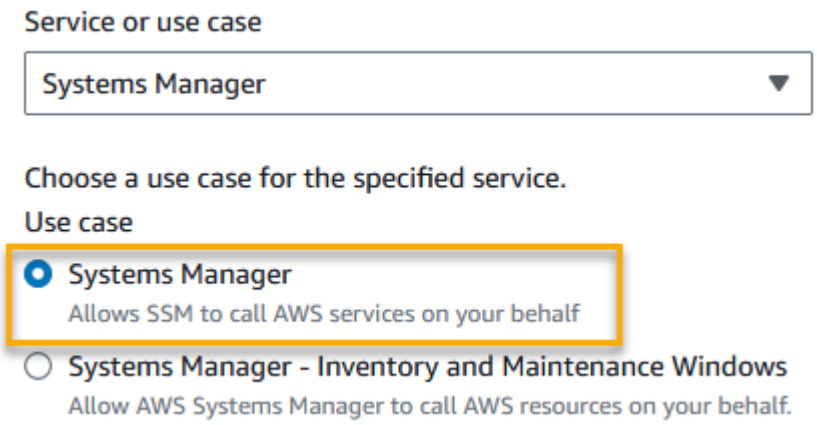
Using the AWS Management Console to create an IAM service role for Systems Manager hybrid activations

Use the following procedure to create a service role for hybrid activation. 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 Systems Manager tools 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. For **Select trusted entity**, make the following choices:
    1. For **Trusted entity type**, choose **AWS service**.
    2. For **Use cases for other AWS services**, choose **Systems Manager**.
    3. Choose **Systems Manager**.

    The following image highlights the location of the Systems Manager option.



4. Choose **Next**.
5. On the **Add permissions** page, do the following:

   - Use the **Search** field to locate the **AmazonSSMManagedInstanceCore** policy. Select the check box next to its name, as shown in the following illustration.



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

   - If you created a custom S3 bucket policy in the procedure [(Optional) Create a custom policy for S3 bucket access](), search for it and select the check box next to its name.
   - If you plan to join non-EC2 machines 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 managed node, 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 IAM server role, such as **`SSMServerRole`**.

> **Note**
>
> Make a note of the role name. You will choose this role when you register new machines that you want to manage by using Systems Manager.

8. (Optional) For **Description**, update the description for this IAM server role.
9. (Optional) For **Tags**, add one or more tag-key value pairs to organize, track, or control access for this role.
10. Choose **Create role**. The system returns you to the **Roles** page.

Using the AWS CLI to create an IAM service role for Systems Manager hybrid activations

Use the following procedure to create a service role for hybrid activation. 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 non-EC2 machines in a hybrid and multicloud environment to be able to access other tools 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 S3 bucket. Before proceeding, follow the steps in Create a custom S3 bucket policy for an instance profile. 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 and multicloud environment (AWS CLI)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

For information, see Installing or updating the latest version of the AWS CLI.

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. Replace the *`placeholder values`* for account ID and Region with your own information.

   JSON

```
{
   "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-1:111122223333:*"
            }
         }
      }
   ]
}
```

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 node permission to run commands using Systems Manager.

> **Note**
>
> The policies you add for a service profile for managed nodes in a hybrid and multicloud 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 Configure instance permissions required for Systems Manager.

(Required) Run the following command to allow a managed node 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 *amzn-s3-demo-bucket* 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/amzn-s3-demo-bucket
```

Windows

```
aws iam attach-role-policy ^
    --role-name SSMServiceRole ^
    --policy-arn arn:aws:iam::account-id:policy/amzn-s3-demo-bucket
```

(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 node. Your service role needs this policy only if you join your nodes 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 nodes. This command makes it possible to read information on a node 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 and multicloud environment (AWS Tools for Windows PowerShell)

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

   For information, see Installing the AWS Tools for PowerShell.

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.

   JSON

```
{
    "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-1:123456789012:*"
                }
            }
        }
    ]
```

```
}
```

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 node permission to run commands using Systems Manager.

> **Note**
>
> The policies you add for a service profile for managed nodes in a hybrid and multicloud 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 Configure instance permissions required for Systems Manager.

(Required) Run the following command to allow a managed node 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 node. Your server role needs this policy only if you join your nodes 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 nodes. This command makes it possible to read information on a node 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 Create a hybrid activation to register nodes with Systems Manager.

Create a hybrid activation to register nodes with Systems Manager

To set up machines other than Amazon Elastic Compute Cloud (EC2) instances as managed nodes for a hybrid and multicloud environment, you create and apply a *hybrid activation*. After you successfully complete the activation, you *immediately* receive an Activation Code and Activation ID at the top of the console page. You specify this Code and ID combination when you install AWS Systems Manager SSM Agent on non-EC2 machines for your hybrid and multicloud environment. The Code and ID provide secure access to the Systems Manager service from your managed nodes.

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 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 node until you explicitly deregister it. You can deregister a non-EC2 managed node in the following ways:

- Use the **Managed nodes** tab in Fleet Manager in the Systems Manager console
- Use the AWS CLI command deregister-managed-instance
- Use the API action DeregisterManagedInstance.

For more information, see the following topics

- Deregister and reregister a managed node (Linux)
- Deregister and reregister a managed node (Windows Server)

About managed nodes

A managed node is any machine configured for AWS Systems Manager. AWS Systems Manager supports Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, and on-premises servers or VMs, including VMs in other cloud environments. Previously, managed nodes were all referred to as managed instances. The term *instance* now refers to EC2 instances only. The deregister-managed-instance command was named before this terminology change.

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 in the US East (Ohio) Region 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 managed nodes 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 node ID and listed in the Systems Manager console with an ID that is prefixed with "mi-".

> **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 and multicloud environment.

Topics

- Using the AWS Management Console to create an activation for registering managed nodes with Systems Manager
- Using the command line to create an activation for registering managed nodes with Systems Manager

Using the AWS Management Console to create an activation for registering managed nodes with Systems Manager

To create a managed-node activation

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

-or-

If you are accessing **Hybrid Activations** for the first time in the current AWS Region, choose **Create an Activation**.

4. (Optional) For **Activation description**, enter a description for this activation. We recommend entering a description if you plan to activate large numbers of servers and VMs.
5. For **Instance limit**, specify the total number of nodes that you want to register with AWS as part of this activation. The default value is 1 instance.
6. For **IAM role**, choose a service role option that allows your servers and VMs to communicate with AWS Systems Manager in the cloud:

   - **Option 1**: Choose **Use the default role created by the system** to use a role and managed policy provided by AWS.
   - **Option 2**: 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 Create the IAM service role required for Systems Manager in hybrid and multicloud environments.
7. For **Activation expiry date**, 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 nodes after the expiry date, you must create a new activation. The expiry date has no impact on registered and running nodes.

8. (Optional) For **Default instance name** field, specify an identifying name value to be displayed for all managed nodes associated with this activation.

9. Choose **Create activation**. Systems Manager immediately returns the Activation Code and ID to the console.

Using the command line to create an activation for registering managed nodes with Systems Manager

The following procedure describes how to use the AWS Command Line Interface (AWS CLI) (on Linux or Windows Server) or AWS Tools for PowerShell to create a managed node 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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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 Create the IAM service role required for Systems Manager in hybrid and multicloud environments.
> - 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.

Install SSM Agent on hybrid Linux nodes

This topic describes how to install AWS Systems Manager SSM Agent on non-EC2 (Amazon Elastic Compute Cloud) Linux machines in a hybrid and multicloud environment. For information about installing SSM Agent on EC2 instances for Linux, see Manually installing and uninstalling SSM Agent on EC2 instances for Linux.

Before you begin, locate the Activation Code and Activation ID that were generated during the hybrid activation process, as described in Create a hybrid activation to register nodes with Systems Manager. You specify the Code and ID in the following procedure.

To install SSM Agent on non-EC2 machines in a hybrid and multicloud environment

1. Log on to a server or VM in your hybrid and multicloud 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=http://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 during the hybrid activation process and with the identifier of the AWS Region you want to download SSM Agent from, then press `Enter`.

   Important

   Note the following important details:

   - Using `ssm-setup-cli` for non-EC2 installations maximizes the security of your Systems Manager installation and configuration.
   - `sudo` isn't necessary if you're a root user.
   - Download `ssm-setup-cli` from the same AWS Region as where your hybrid activation was created.
   - `ssm-setup-cli` supports a `manifest-url` option that determines the source where the agent is downloaded from. Don't specify a value for this option unless required by your organization.
   - When registering instances, only use the provided download link provided for `ssm-setup-cli`. `ssm-setup-cli` shouldn't be stored separately for future use.
   - You can use the script provided here to validate the signature of `ssm-setup-cli`.

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*.

Additionally, `ssm-setup-cli` includes the following options:

- `version` - Valid values are `latest` and `stable`.
- `downgrade` - Allows the SSM Agent to be downgraded to an earlier version. Specify `true` to install an earlier version of the agent.
- `skip-signature-validation` - Skips the signature validation during the download and installation of the agent.

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

```
mkdir /tmp/ssm
curl https://amazon-ssm-region.s3.region.amazonaws.com/latest/linux_amd64/ssm-setup-cli
 -o /tmp/ssm/ssm-setup-cli
sudo chmod +x /tmp/ssm/ssm-setup-cli
sudo /tmp/ssm/ssm-setup-cli -register -activation-code "activation-code" -activation-id
 "activation-id" -region "region"
```

RHEL 8.x

```
mkdir /tmp/ssm
curl https://amazon-ssm-region.s3.region.amazonaws.com/latest/linux_amd64/ssm-setup-cli
 -o /tmp/ssm/ssm-setup-cli
sudo chmod +x /tmp/ssm/ssm-setup-cli
sudo /tmp/ssm/ssm-setup-cli -register -activation-code "activation-code" -activation-id
 "activation-id" -region "region"
```

Debian Server

```
mkdir /tmp/ssm
curl https://amazon-ssm-region.s3.region.amazonaws.com/latest/debian_amd64/ssm-setup-
cli -o /tmp/ssm/ssm-setup-cli
sudo chmod +x /tmp/ssm/ssm-setup-cli
```

```
sudo /tmp/ssm/ssm-setup-cli -register -activation-code "activation-code" -activation-id
 "activation-id" -region "region"
```

Ubuntu Server

- **Using .deb packages**

```
mkdir /tmp/ssm
curl https://amazon-ssm-region.s3.region.amazonaws.com/latest/debian_amd64/ssm-setup-
cli -o /tmp/ssm/ssm-setup-cli
sudo chmod +x /tmp/ssm/ssm-setup-cli
sudo /tmp/ssm/ssm-setup-cli -register -activation-code "activation-code" -activation-
id "activation-id" -region "region"
```

- **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 20.04, 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 managed nodes.

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

The command downloads and installs SSM Agent onto the hybrid-activated machine in your hybrid and multicloud environment. The command stops SSM Agent, and then registers the machine with the Systems Manager service. The machine is now a managed node. Amazon EC2 instances configured for Systems Manager are also managed nodes. In the Systems Manager console, however, your hybrid-activated nodes are distinguished from Amazon EC2 instances with the prefix "mi-".

Continue to Install SSM Agent on hybrid Windows Server nodes.

Setting up private key auto rotation

To strengthen your security posture, you can configure AWS Systems Manager Agent (SSM Agent) to automatically rotate the private key for your hybrid and multicloud environment. 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 private key for a hybrid and multicloud environment

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 node (Linux)

You can deregister a hybrid-activated managed node 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"
```

To remove the remaining registration information for the agent, remove the `IdentityConsumptionOrder` key in the `amazon-ssm-agent.json` file. Then, depending on your installation type, run one of the following commands.

On Ubuntu Server nodes where SSM Agent was installed using Snap packages:

```
sudo /snap/amazon-ssm-agent/current/amazon-ssm-agent -register -clear
```

On all other Linux installations:

```
amazon-ssm-agent -register -clear
```

> **Note**
>
> You can reregister an on-premises server, edge device, or VM using the same activation code and ID as long as you haven't reached the instance limit for the designated activation code and ID. You can verify the instance limit for an activation code and ID by calling the describe-activations API using the AWS CLI. After you run the command, verify that the value of `RegistrationCount` doesn't exceed `RegistrationLimit`. If it does, you must use a different activation code and ID.

To reregister a managed node on a non-EC2 Linux machine

1. Connect to your machine.
2. Run the following command. Be sure to replace the placeholder values with the Activation Code and Activation ID generated when you created a managed-node activation, and with the identifier of the Region you want to download the SSM Agent from.

   ```
   echo "yes" | sudo /tmp/ssm/ssm-setup-cli -register -activation-code "activation-code" -activation-id "activation-id" -region "region
   ```

Troubleshooting SSM Agent installation on non-EC2 Linux machines

Use the following information to help you troubleshoot problems installing SSM Agent on hybrid-activated Linux machines in a hybrid and multicloud environment.

You receive DeliveryTimedOut error

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

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

Unable to load node 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
```

Install SSM Agent on hybrid Windows Server nodes

This topic describes how to install AWS Systems Manager SSM Agent on Windows Server machines in a hybrid and multicloud environment. For information about installing SSM Agent on EC2 instances for Windows Server, see Manually installing and uninstalling SSM Agent on EC2 instances for Windows Server.

Before you begin, locate the Activation Code and Activation ID that were generated during the hybrid activation process, as described in Create a hybrid activation to register nodes with Systems Manager. You specify the Code and ID in the following procedure.

To install SSM Agent on non-EC2 Windows Server machines in a hybrid and multicloud environment

1. Log on to a server or VM in your hybrid and multicloud 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=http://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 each *`example resource placeholder`* with your own information. For example, the Activation Code and Activation ID generated when you create a hybrid activation, and with the identifier of the AWS Region you want to download SSM Agent from.

   Important

   Note the following important details:

   - Using `ssm-setup-cli` for non-EC2 installations maximizes the security of your Systems Manager installation and configuration.
   - `ssm-setup-cli` supports a `manifest-url` option that determines the source where the agent is downloaded from. Don't specify a value for this option unless required by your organization.
   - You can use the script provided here to validate the signature of `ssm-setup-cli`.
   - When registering instances, only use the provided download link provided for `ssm-setup-cli`. `ssm-setup-cli` shouldn't be stored separately for future use.

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*.

Additionally, ssm-setup-cli includes the following options:

- version - Valid values are latest and stable.
- downgrade - Reverts the agent to an earlier version.
- skip-signature-validation - Skips the signature validation during the download and installation of the agent.

64-bit

```
[System.Net.ServicePointManager]::SecurityProtocol = 'TLS12'
$code = "activation-code"
$id = "activation-id"
$region = "us-east-1"
$dir = $env:TEMP + "\ssm"
New-Item -ItemType directory -Path $dir -Force
cd $dir
(New-Object System.Net.WebClient).DownloadFile("https://amazon-ssm-$region.s3.
$region.amazonaws.com/latest/windows_amd64/ssm-setup-cli.exe", $dir + "\ssm-
setup-cli.exe")
./ssm-setup-cli.exe -register -activation-code="$code" -activation-id="$id" -
region="$region"
Get-Content ($env:ProgramData + "\Amazon\SSM\InstanceData\registration")
Get-Service -Name "AmazonSSMAgent"
```

5. Press Enter.

> **Note**
>
> If the command fails, verify that you are running the latest version of AWS Tools for PowerShell.

The command does the following:

- Downloads and installs SSM Agent onto the machine.
- Registers the machine 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 machine is now a *managed node*. These managed nodes are now identified with the prefix "mi-". You can view managed nodes on the **Managed node** page in Fleet Manager, by using the AWS CLI command describe-instance-information, or by using the API command DescribeInstanceInformation.

Setting up private key auto rotation

To strengthen your security posture, you can configure AWS Systems Manager Agent (SSM Agent) to automatically rotate the private key for a hybrid and multicloud environment. 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 private key for a hybrid and multicloud environment

1. Navigate to `/etc/amazon/ssm/` on a Linux machine or `C:\Program Files\Amazon\SSM` for a Windows Server 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 node (Windows Server)

You can deregister a managed node 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"
```

To remove the remaining registration information for the agent, remove the `IdentityConsumptionOrder` key in the `amazon-ssm-agent.json` file. Then run the following command:

```
amazon-ssm-agent -register -clear
```

> **Note**
>
> You can reregister an on-premises server, edge device, or VM using the same activation code and ID as long as you haven't reached the instance limit for the designated activation code and ID. You can verify the instance limit for an activation code and ID by calling the describe-activations API using the AWS CLI. After you run the command, verify that the value of `RegistrationCount` doesn't exceed `RegistrationLimit`. If it does, you must use a different activation code and ID.

To reregister a managed node on a Windows Server hybrid machine

1. Connect to your machine.
2. Run the following command. Be sure to replace the placeholder values with the Activation Code and Activation ID generated when you created a hybrid activation, and with the identifier of the Region you want to download the SSM Agent from.

```
$dir = $env:TEMP + "\ssm"
cd $dir
Start-Process ./ssm-setup-cli.exe -ArgumentList @(
    "-register",
    "-activation-code=$code",
    "-activation-id=$id",
    "-region=$region"
) -Wait -NoNewWindow
```

Managing edge devices with Systems Manager

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 tools 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 using a hybrid activation. For more information, see Managing nodes in hybrid and multicloud environments with Systems Manager.
- 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.

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 for Systems Manager.
- 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.

> **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*.

Create an IAM service role for your 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 non-EC2 nodes for a hybrid and multicloud 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.

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 S3 bucket that you create as part of your Systems Manager operations, such as for storing output for Run Command commands or Session Manager sessions to an S3 bucket. Before proceeding, follow the steps in Create a custom S3 bucket policy for an instance profile. 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 Installing or updating the latest version of the AWS CLI.

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.

JSON

```
{
    "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. Replace each *example resource placeholder* with your own information.

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 IAM policies used in the following commands, see Configure instance permissions required for Systems Manager.

(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 (Tools for Windows PowerShell), if you haven't already.

   For information, see Installing the AWS Tools for PowerShell.

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.

JSON

```
{
    "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 Configure instance permissions required for Systems Manager.

(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
```

Configure your edge devices for 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*.

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. For information about these processes, see Install the AWS Systems Manager Agent in the *AWS IoT Greengrass Version 2 Developer Guide*.

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 tools 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.

Creating an AWS Organizations delegated administrator for Systems Manager

When you set up an organization in AWS Organizations, you assign a management account to perform all administrative tasks for all AWS services. The management account user can assign a *delegated administrator account* only for Systems Manager to perform administrative tasks for Change Manager, Explorer, and OpsCenter. AWS Organizations is an account management service that you can use to create an organization and assign AWS accounts to manage these accounts centrally. For information about AWS Organizations, see AWS Organizations in the *AWS Organizations User Guide*.

Change Manager, Explorer, and OpsCenter, tools in AWS Systems Manager, work with AWS Organizations to perform tasks on all member accounts of your organization. You can assign only one delegated administrator for all Systems Manager tools. The delegated administrator account must be a member of the organization to which it's assigned.

Topics

- Using a delegated administrator with Change Manager
- Using a delegated administrator with Explorer
- Using a delegated administrator with OpsCenter
- Using a delegated administrator with Quick Setup

Using a delegated administrator with Change Manager

Change Manager is an enterprise change management framework for requesting, approving, implementing, and reporting on operational changes to your application configuration and infrastructure.

If you use Change Manager across an organization, assign a delegated administrator account to manage change templates, approvals, and reporting for all member accounts. Using Quick Setup, you can set up Change Manager to use with an organization and select the delegated administrator account. If you use Change Manager with a single AWS account, the delegated administrator account isn't required.

By default, Change Manager displays all change-related tasks in the delegated administrator account. For instructions on configuring a delegated administrator while setting up Change Manager for an organization, see Setting up Change Manager for an organization (management account).

> **Important**
>
> If you use Change Manager across an organization, we recommend always making changes from the delegated administrator account. Although you can make changes from other accounts in the organization, those changes won't be reported in or viewable from the delegated administrator account.

Using a delegated administrator with Explorer

Explorer is a customizable operations dashboard that reports aggregated view of operations data (OpsData) for your AWS accounts, across AWS Regions.

You can configure a delegated administrator account for Systems Manager to aggregate Explorer data from multiple Regions and accounts by using resource data sync with AWS Organizations. A delegated administrator can search, filter, and aggregate Explorer data using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or AWS Tools for Windows PowerShell.

When you use a delegated administrator account for Explorer, you limit the number of administrators who can create or delete multi-account and Region resource data syncs to an individual AWS account.

You can synchronize operations data across all AWS accounts in your organization by using Explorer. For information on how to assign a delegated administrator from Explorer, see Configuring a delegated administrator for Explorer.

Using a delegated administrator with OpsCenter

OpsCenter provides a central location where operations engineers and IT professionals can manage operational work items (OpsItems) related to AWS resources. If you want to use OpsCenter to manage OpsItems centrally across accounts, you must set up the organization in AWS Organizations.

Using Quick Setup for OpsCenter, you can assign a delegated administrator account and configure OpsCenter to manage OpsItems centrally. For more information, see (Optional) Configure OpsCenter to manage OpsItems across accounts by using Quick Setup.

Using a delegated administrator with Quick Setup

Quick Setup is a tool in Systems Manager that helps you to quickly configure frequently used AWS services and features with recommended best practices. You can configure a delegated administrator account for Quick Setup to help you deploy and manage configurations across accounts and Regions using AWS Organizations. A delegated administrator for Quick Setup can create, update, view, and delete configuration manager resources in your organization. Systems Manager registers a delegated administrator for Quick Setup as part of the setup process for the integrated console experience. For more information, see Setting up Systems Manager unified console for an organization.

General setup for AWS Systems Manager

If you haven't already done so, sign up for an AWS account and create an administrative user.

Sign up for an AWS account

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 or text message and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Setting up AWS Systems Manager

The following topics describe how to set up the unified AWS Systems Manager console for AWS Organizations organizations and single AWS accounts.

Topics

- Setting up Systems Manager console access
- Setting up Systems Manager unified console for an organization
- Setting up Systems Manager unified console for a single account and Region
- Disabling the Systems Manager unified console

Setting up Systems Manager console access

To use AWS Systems Manager in the AWS Management Console, you must have the correct permissions configured.

For more information about how to create AWS Identity and Access Management policies and attach them to IAM identities, see Create IAM policies in the *IAM User Guide*

Systems Manager onboarding policy

You can create an IAM policy like the one shown in the following example, and attach the policy to your IAM identities. This policy grants full access to onboard to and configure Systems Manager.

Permissions details

This policy includes the following permissions.

- `ssm-quicksetup` – Allows principals to access all AWS Systems Manager Quick Setup actions.
- `ssm` – Allows principals access to Systems Manager Automation and Resource Explorer.
- `organizations` – Allows principals to read an organization's structure in AWS Organizations, and manage delegated administrators when they are onboarding to Systems Manager as an organization.
- `cloudformation` – Allows principals to manage their Quick Setup stacks.
- `iam` – Allows principals to manage IAM roles and policies that are required for Systems Manager onboarding.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },

     {
       "Sid": "SsmReadOnly",
       "Effect": "Allow",
       "Action": [
         "ssm:DescribeAutomationExecutions",
         "ssm:GetAutomationExecution",
         "ssm:ListAssociations",
         "ssm:DescribeAssociation",
         "ssm:ListDocuments",
         "ssm:ListResourceDataSync",
         "ssm:DescribePatchBaselines",
         "ssm:GetPatchBaseline",
         "ssm:DescribeMaintenanceWindows",
         "ssm:DescribeMaintenanceWindowTasks"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmDocument",
       "Effect": "Allow",
       "Action": [
         "ssm:GetDocument",
         "ssm:DescribeDocument"
       ],
       "Resource": [
         "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
         "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
       ]
     },
```

```
    {
      "Sid": "SsmEnableExplorer",
      "Effect": "Allow",
      "Action": "ssm:StartAutomationExecution",
      "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
    },
    {
      "Sid": "SsmExplorerRds",
      "Effect": "Allow",
      "Action": [
        "ssm:GetOpsSummary",
        "ssm:CreateResourceDataSync",
        "ssm:UpdateResourceDataSync"
      ],
      "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
    },
    {
      "Sid": "OrgsReadOnly",
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListRoots",
        "organizations:ListParents",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:DescribeOrganizationalUnit",
        "organizations:ListAWSServiceAccessForOrganization"
      ],
      "Resource": "*"
    },
    {
      "Sid": "OrgsAdministration",
      "Effect": "Allow",
      "Action": [
        "organizations:EnableAWSServiceAccess",
        "organizations:RegisterDelegatedAdministrator",
        "organizations:DeregisterDelegatedAdministrator"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "organizations:ServicePrincipal": [
            "ssm.amazonaws.com",
```

```
            "ssm-quicksetup.amazonaws.com",
            "member.org.stacksets.cloudformation.amazonaws.com",
            "resource-explorer-2.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "CfnReadOnly",
      "Effect": "Allow",
      "Action": [
        "cloudformation:ListStacks",
        "cloudformation:DescribeStacks",
        "cloudformation:ListStackSets",
        "cloudformation:DescribeOrganizationsAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "OrgCfnAccess",
      "Effect": "Allow",
      "Action": [
        "cloudformation:ActivateOrganizationsAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CfnStackActions",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStackResources",
        "cloudformation:DescribeStackEvents",
        "cloudformation:GetTemplate",
        "cloudformation:RollbackStack",
        "cloudformation:TagResource",
        "cloudformation:UntagResource",
        "cloudformation:UpdateStack"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
        "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
        "arn:aws:cloudformation:*:*:type/resource/*"
```

```
      ]
    },
    {
      "Sid": "CfnStackSetActions",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStackInstances",
        "cloudformation:CreateStackSet",
        "cloudformation:DeleteStackInstances",
        "cloudformation:DeleteStackSet",
        "cloudformation:DescribeStackInstance",
        "cloudformation:DetectStackSetDrift",
        "cloudformation:ListStackInstanceResourceDrifts",
        "cloudformation:DescribeStackSet",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:ListStackInstances",
        "cloudformation:ListStackSetOperations",
        "cloudformation:ListStackSetOperationResults",
        "cloudformation:TagResource",
        "cloudformation:UntagResource",
        "cloudformation:UpdateStackSet"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
        "arn:aws:cloudformation:*:*:type/resource/*",
        "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
      ]
    },
    {
      "Sid": "ValidationReadonlyActions",
      "Effect": "Allow",
      "Action": [
        "iam:ListRoles",
        "iam:GetRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IamRolesMgmt",
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:GetRole",
```

```
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:GetRolePolicy",
        "iam:ListRolePolicies"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWS-QuickSetup-*",
        "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
      ]
    },
    {
      "Sid": "IamPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWS-QuickSetup-*",
        "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "ssm.amazonaws.com",
            "ssm-quicksetup.amazonaws.com",
            "cloudformation.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "IamRolesPoliciesMgmt",
      "Effect": "Allow",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWS-QuickSetup-*",
        "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
      ],
      "Condition": {
        "ArnEquals": {
          "iam:PolicyARN": [
```

```
            "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
            "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
          ]
        }
      }
    },
    {
      "Sid": "CfnStackSetsSLR",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
        "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
        "arn:aws:iam::*:role/aws-service-role/
accountdiscovery.ssm.amazonaws.com/AWSServiceRoleForAmazonSSM_AccountDiscovery",
        "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
        "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
      ]
    }
  ]
}
```

AWS Systems Manager console operator policy

You can create an IAM policy like the one shown in the following example, and attach the policy to your IAM identities. This policy grants full access to operate Systems Manager, and allow Systems Manager to run Automation documents for diagnosis and remediation.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to access all Systems Manager APIs.
- `ssm-quicksetup` – Allows principals to manage their Quick Setup configurations.

- `ec2` – Allows Systems Manager to determine your enabled AWS Regions and Amazon EC2 instance status.
- `cloudformation` – Allows principals to read their Quick Setup stacks.
- `organizations` – Allows principals to read an organization's structure in AWS Organizations, and manage delegated administrators when they are onboarding to Systems Manager as an organization.
- `s3` – Allows principals to list and get objects in an Amazon S3 bucket for diagnosis, which is created during the Systems Manager onboarding process.
- `iam:PassRole` – Allows principals to pass roles to be assumed to Systems Manager when they run automations for diagnosis and remediation of unmanaged nodes.
- `iam:GetRole` – Allows principals to get specific role information for Quick Setup roles when they are working in Systems Manager.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:*",
        "ssm-quicksetup:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowEC2DescribeActions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:DescribeRegions"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CfnAccess",
```

```
      "Effect": "Allow",
      "Action": [
        "cloudformation:ListStacks",
        "cloudformation:ListStackSets",
        "cloudformation:ListStackInstances",
        "cloudformation:ListStackSetOperations",
        "cloudformation:ListStackSetOperationResults",
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackSet",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:DescribeOrganizationsAccess",
        "cloudformation:DescribeStackInstance",
        "cloudformation:DetectStackSetDrift",
        "cloudformation:ListStackInstanceResourceDrifts"
      ],
      "Resource": "*"
    },
    {
      "Sid": "OrgsReadOnly",
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListRoots",
        "organizations:ListParents",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:DescribeOrganizationalUnit",
        "organizations:ListAWSServiceAccessForOrganization"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowKMSOperations",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "arn:aws:kms:*:*:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/SystemsManagerManaged": "true"
        },
```

```
        "ArnLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::do-not-delete-ssm-
diagnosis-*"
        },
        "StringLike": {
          "kms:ViaService": "s3.*.amazonaws.com"
        },
        "Bool": {
          "aws:ViaAWSService": "true"
        }
      }
    },
    {
      "Sid": "AllowReadS3BucketFromOrganization",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::do-not-delete-ssm-diagnosis*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceOrgId": "${aws:PrincipalOrgId}"
        }
      }
    },
    {
      "Sid": "AllowReadS3BucketFromSingleAccount",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::do-not-delete-ssm-diagnosis*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": [
```

```
        "arn:aws:iam::*:role/AWS-SSM-DiagnosisAdminRole*",
        "arn:aws:iam::*:role/AWS-SSM-DiagnosisExecutionRole*",
        "arn:aws:iam::*:role/AWS-SSM-RemediationAdminRole*",
        "arn:aws:iam::*:role/AWS-SSM-RemediationExecutionRole*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "ssm.amazonaws.com"
        }
      }
    },
    {
      "Sid": "IamReadOnly",
      "Effect": "Allow",
      "Action": "iam:GetRole",
      "Resource": [
        "arn:aws:iam::*:role/AWS-QuickSetup-*",
        "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
      ]
    }
  ]
}
```

AWS Systems Manager console operator read-only policy

You can create an IAM policy like the one shown in the following example, and attach the policy to your IAM identities. This policy grants read-only access to use Systems Manager.

- `ssm` – Allows principals access to Systems Manager read-only APIs.
- `ssm-quicksetup` – Allows principals to read their Quick Setup configurations.
- `cloudformation` – Allows principals to read their Quick Setup stacks.
- `iam:GetRole` – Allows principals to get specific role information for Quick Setup roles when they are using Systems Manager.
- `ec2:DescribeRegions` – Allows Systems Manager to determine your enabled AWS Regions.
- `organizations` – Allows principals to read an organization's structure in AWS Organizations when they are onboarding to Systems Manager as an organization.
- `s3` – Allows principals to list and get objects in an Amazon S3 bucket that is created during the Systems Manager onboarding process.

Permissions details

This policy includes the following permissions.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:Describe*",
        "ssm:Get*",
        "ssm:List*",
        "ssm-quicksetup:List*",
        "ssm-quicksetup:Get*",
        "cloudformation:Describe*",
        "cloudformation:Get*",
        "cloudformation:List*",
        "iam:GetRole",
        "ec2:DescribeRegions",
        "organizations:Describe*",
        "organizations:List*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowKMSOperations",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:*:*:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/SystemsManagerManaged": "true"
        },
        "ArnLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::do-not-delete-ssm-
diagnosis-*"
        },
        "StringLike": {
```

```
          "kms:ViaService": "s3.*.amazonaws.com"
        },
        "Bool": {
          "aws:ViaAWSService": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::do-not-delete-ssm-diagnosis*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceOrgId": "${aws:PrincipalOrgId}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::do-not-delete-ssm-diagnosis*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    }
  ]
}
```

Setting up Systems Manager unified console for an organization

The setup process for the Systems Manager unified console experience is completed from the AWS Management Console with just a few clicks. To set up Systems Manager for an AWS Organizations

organization, you must have access to the management account for your organization and another account in your organization to use as a delegated administrator. Access to the management account is only required to enable or disable Systems Manager. To manage your nodes, you'll use the delegated administrator account.

Prerequisites

When managing nodes across an organization, Systems Manager uses various dependent services to set up and enhance the functionality of the unified console. As a result, Systems Manager must enable trusted access and register a delegated administrator account for the following services:

- AWS CloudFormation - Deploys resources required for Systems Manager to your accounts.
- AWS Resource Explorer - Searching and filtering EC2 instances in your accounts.
- AWS Systems Manager Explorer - Monitoring and troubleshooting the health of resources deployed for Systems Manager in your accounts.
- AWS Systems Manager Quick Setup - Deploys Quick Setup configurations required for Systems Manager to your accounts.

Before you begin, make sure you're not already over the quota for delegated administrators for any of these dependent services. Otherwise, you won't be able to register the delegated administrator accounts necessary to enable Systems Manager. When you enable Systems Manager for an organization, every account in your organization is included. At this time, there is no provision for excluding accounts from the setting up process. When you enable Systems Manager, you can choose the AWS Regions you want to include. Only Regions that currently support the Systems Manager unified console can be selected. To learn more about the Regions where the console experience is available, see Supported AWS Regions.

Unified console resources

The setup process for the Systems Manager unified console completes many prerequisite tasks for you. Depending on the features you choose to configure, this includes enabling Default Host Management Configuration to provide the required IAM permissions to your nodes and more. The following is a detailed list of the resources created by Systems Manager for the unified console. Depending on the features you choose to configure, some resources might not be created.

AWS Resource Explorer managed views

- `AWSManagedViewForSSM` – Allows Systems Manager to access resource information indexed by Resource Explorer for your organization. These managed views can only be updated or deleted by Systems Manager. This means that if you want to delete the managed views, or turn off Resource Explorer, you must disable the unified console. For more information about disabling the unified console, see Disabling the Systems Manager unified console. For more information about managed views, see AWS Managed Views in the *Resource Explorer User Guide*.

> **Note**
>
> If you've created an aggregator index for Resource Explorer in a Region different than your home Region, Systems Manager demotes the current index. Then, Systems Manager promotes the local index in your home Region as the new aggregator index. During this time, only nodes for your home Region are displayed. This process can take up to 24 hours to complete.

IAM roles

- `RoleForOnboardingAutomation` – Allows Systems Manager to manage resources during the setting up process. For more information about the policy, see AWSQuickSetupSSMManageResourcesExecutionPolicy.
- `RoleForLifecycleManagement` – Allows Lambda to manage the lifecycle of resources created by the setting up process. For more information about the policy, see AWSQuickSetupSSMLifecycleManagementExecutionPolicy.
- `RoleForAutomation` – A service role for Systems Manager Automation to assume to execute runbooks. For more information, see Create the service roles for Automation using the console.
- `AWSSSMDiagnosisAdminRole` – An adminsitrative role used to start automations that use diagnosis runbooks. For more information about the policies, see AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy, AWS-SSM-Automation-DiagnosisBucketPolicy, and AWS-SSM-DiagnosisAutomation-OperationalAccountAdministrationRolePolicy.
- `AWSSSMDiagnosisExecutionRole` – An automation execution role for the diagnosis runbook. For more information about the policies, see AWS-SSM-DiagnosisAutomation-ExecutionRolePolicy and AWS-SSM-Automation-DiagnosisBucketPolicy.
- `AWSSSMRemediationAdminRole` – An adminsitrative role used to start automations that use remediation runbooks. For more information about the policies, see AWS-SSM-RemediationAutomation-AdministrationRolePolicy, AWS-SSM-

Automation-DiagnosisBucketPolicy, and AWS-SSM-RemediationAutomation-OperationalAccountAdministrationRolePolicy.

- `AWSSSMRemediationExecutionRole` – An automation execution role for the remediation runbook. For more information about the policies, see AWS-SSM-RemediationAutomation-ExecutionRolePolicy and AWS-SSM-Automation-DiagnosisBucketPolicy.
- `ManagedInstanceCrossAccountManagementRole` – Allows Systems Manager to gather managed node information across accounts.

State Manager associations

- `EnableDHMCAssociation` – Runs daily and ensures Default Host Management Configuration is enabled.
- `SystemAssociationForEnablingExplorer` – Runs daily and ensures Explorer is enabled. Explorer is used to sync data from your managed nodes.
- `EnableAREXAssociation` – Runs daily and ensures AWS Resource Explorer is enabled. Resource Explorer is used to determine which Amazon EC2 instances in your organization aren't managed by Systems Manager.
- `SSMAgentUpdateAssociation` – Runs every 14 days and ensures the latest available version of SSM Agent is installed on your managed nodes.
- `SystemAssociationForInventoryCollection` – Runs every 12 hours and collects inventory data from your managed nodes.

S3 buckets

- `DiagnosisBucket` – Stores data collected from the diagnosis runbook execution.

Lambda functions

- `SSMLifecycleOperatorLambda` – Allows principals to access all AWS Systems Manager Quick Setup actions.
- `SSMLifecycleResource` – Custom resource to help manage the lifecycle of resources created by the setting up process.

Additionally, after the setup process completes you can select the **Diagnose and remediate** node task to automatically apply fixes to nodes that aren't reporting as managed by Systems Manager.

This can include identifying issues such as network connectivity issues to the Systems Manager endpoints, and more. For more information, see Diagnosing and remediating.

Set up the unified console

To set up Systems Manager for an organization

1. Log in to the management account for your organization.
2. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
3. Enter the ID of the account you want to register as a delegated administrator.
4. After the delegated administrator account is successfully registered, log in to the delegated administrator account you just registered and return to the Systems Manager console to finish setting up Systems Manager.
5. Select **Enable Systems Manager**.
6. In the **Home Region** section, you determine a Region where you want Systems Manager to aggregate your node data. By default, Systems Manager selects the Region you're currently using. To choose a different home Region, change the console to the Region you want to use before you set up Systems Manager. Node data is replicated across accounts and Regions for your organization and stored in the home Region. The Region you choose can't be changed after Systems Manager is set up. To use a different Region as the home Region for your organization, you must disable the unified console and complete the setup process again. If your organization uses IAM Identity Center, you must select the same Region where you set up IAM Identity Center as your home Region.
7. In the **Regions** section, select the Regions where you want to enable Systems Manager.
8. In the **Feature configurations** section, choose the options that you want to enable for your configuration:

   Enable Default Host Management Configuration (DHMC)

   Allows Systems Manager to configure DHMC. This feature allows Systems Manager to use an IAM role to ensure that all Amazon EC2 instances in the account and Region have the permissions necessary to be managed by Systems Manager. You can also specify the frequency of drift remediation. Configuration drift occurs whenever a user makes any change to a service or feature that conflicts with the selections made through your configuration. Systems Manager checks for configuration drift and attempts to remediate it based on the frequency you specify. You must specify a value between 1 and 31 days. If

you've already configured DHMC in a Region, Systems Manager doesn't change the IAM role you previously selected. For more information about DHMC, see Managing EC2 instances automatically with Default Host Management Configuration.

DHMC makes it possible to manage Amazon EC2 instances without your having to manually create an AWS Identity and Access Management (IAM) instance profile. We encourage you to choose this option to ensure that your EC2 instances have the permissions necessary to be managed by Systems Manager.

Enable inventory metadata collection

Enables Systems Manager to configure collection of the following types of metadata from your nodes:

- **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).

Specify the frequency at which inventory is collected. You must specify a value between 1 and 744 hours. For more information about Inventory, a tool in AWS Systems Manager, see AWS Systems Manager Inventory.

Enable automatic Systems Manager (SSM) Agent updates

Enables Systems Manager to check for a new version of the agent at the frequency you specify. The value for the frequency must be between 1 and 31 days. If there is a new version, then Systems Manager automatically updates the agent on your managed node to the latest released version. Systems Manager doesn't install the agent on instances where it's not already present. For information about which AMIs have SSM Agent preinstalled, see Find AMIs with the SSM Agent preinstalled.

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.

9. Choose **Submit**.

Depending on the size of your organization, it can take an extended amount of time to set up the Systems Manager unified console experience.

Setting up Systems Manager unified console for a single account and Region

To set up the Systems Manager unified console experience for a single AWS account and AWS Region you don't need to use Organizations or register a delegated administrator account. The setup process for the Systems Manager console experience completes many prerequisite tasks for you. Depending on the features you choose to configure, this includes enabling Default Host Management Configuration to provide the required IAM permissions to your nodes and more. The following is a detailed list of the resources created by Systems Manager for the unified console.

Unified console resources

Depending on the features you choose to configure, some resources might not be created.

IAM roles

- `RoleForOnboardingAutomation` – Allows Systems Manager to manage resources during the setting up process. For more information about the policy, see AWSQuickSetupSSMManageResourcesExecutionPolicy.
- `RoleForLifecycleManagement` – Allows Lambda to manage the lifecycle of resources created by the setting up process. For more information about the policy, see AWSQuickSetupSSMLifecycleManagementExecutionPolicy.
- `RoleForAutomation` – A service role for Systems Manager Automation to assume to execute runbooks. For more information, see Create the service roles for Automation using the console.
- `AWSSSMDiagnosisAdminRole` – An automation execution role for the diagnosis runbook. For more information about the policies, see AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy, AWS-SSM-Automation-DiagnosisBucketPolicy, and AWS-SSM-DiagnosisAutomation-OperationalAccountAdministrationRolePolicy.

- `AWSSSMRemediationAdminRole` – An automation execution role for the remediation runbook. For more information about the policies, see AWS-SSM-RemediationAutomation-AdministrationRolePolicy, AWS-SSM-Automation-DiagnosisBucketPolicy, and AWS-SSM-RemediationAutomation-OperationalAccountAdministrationRolePolicy.
- `ManagedInstanceCrossAccountManagementRole` – Allows Systems Manager to gather managed node information across accounts.

State Manager associations

- `EnableDHMCAssociation` – Runs daily and ensures Default Host Management Configuration is enabled.
- `SystemAssociationForEnablingExplorer` – Runs daily and ensures Explorer is enabled. Explorer is used to sync data from your managed nodes.
- `EnableAREXAssociation` – Runs daily and ensures AWS Resource Explorer is enabled. Resource Explorer is used to determine which Amazon EC2 instances in your organization aren't managed by Systems Manager.
- `SSMAgentUpdateAssociation` – Runs every 14 days and ensures the latest available version of SSM Agent is installed on your managed nodes.
- `SystemAssociationForInventoryCollection` – Runs every 12 hours and collects inventory data from your managed nodes.

S3 buckets

- `DiagnosisBucket` – Stores data collected from the diagnosis runbook execution.

Lambda functions

- `SSMLifecycleOperatorLambda` – Allows principals to access all AWS Systems Manager Quick Setup actions.
- `SSMLifecycleResource` – Custom resource to help manage the lifecycle of resources created by the setting up process.

Additionally, after the setup process completes you can select the **Diagnose and remediate** node task to automatically apply fixes to nodes that aren't reporting as managed by Systems Manager.

This can include identifying issues such as network connectivity issues to the Systems Manager endpoints, and more.

Set up the unified console

To set up Systems Manager for a single account and Region

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Enable Systems Manager**.
3. In the **Feature configurations** section, choose the options that you want to enable for your configuration:

Enable Default Host Management Configuration (DHMC)

Allows Systems Manager to configure DHMC. This feature allows Systems Manager to use an IAM role to ensure that all Amazon EC2 instances in the account and Region have the permissions necessary to be managed by Systems Manager. You can also specify the frequency of drift remediation. Configuration drift occurs whenever a user makes any change to a service or feature that conflicts with the selections made through your configuration. Systems Manager checks for configuration drift and attempts to remediate it based on the frequency you specify. You must specify a value between 1 and 31 days. If you've already configured DHMC in a Region, Systems Manager doesn't change the IAM role you previously selected. For more information about DHMC, see Managing EC2 instances automatically with Default Host Management Configuration.

DHMC makes it possible to manage Amazon EC2 instances without your having to manually create an AWS Identity and Access Management (IAM) instance profile. We encourage you to choose this option to ensure that your EC2 instances have the permissions necessary to be managed by Systems Manager.

Enable inventory metadata collection

Enables Systems Manager to configure collection of the following types of metadata from your nodes:

- **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).

Specify the frequency at which inventory is collected. You must specify a value between 1 and 744 hours. For more information about Inventory, a tool in AWS Systems Manager, see AWS Systems Manager Inventory.

Enable automatic Systems Manager (SSM) Agent updates

Enables Systems Manager to check for a new version of the agent at the frequency you specify. The value for the frequency must be between 1 and 31 days. If there is a new version, then Systems Manager automatically updates the agent on your managed node to the latest released version. Systems Manager doesn't install the agent on instances where it's not already present. For information about which AMIs have SSM Agent preinstalled, see Find AMIs with the SSM Agent preinstalled.

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.

4. Choose **Submit**.

Disabling the Systems Manager unified console

To disable the Systems Manager unified console for an organization, you'll need access to the account you registered as a delegated administrator for Systems Manager. After logging in to the delegated administrator account for your organization, you can disable the setup for your organization in the **Settings** section of the unified console. When you disable the unified console setup for your organization, Systems Manager deletes the resources created, including Resource Explorer managed views, during the setting up process. Disabling the setup for your organization

doesn't revoke trusted access or deregister the delegated administrator accounts for dependent services. The following procedure describes how to disable the setup for the unified console.

To disable the setup for the Systems Manager unified console

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Settings** in the navigation pane.
3. Select **Disable**. You must confirm that you want to disable the setup for your organization. This action deletes the resources created for the unified console and can't be undone.

If you're unable to access the delegated administrator account for Systems Manager and want to disable the setup for the unified console, you can also do so from the management account for your organization. Using the AWS CLI or SDK, call the `DeleteConfigurationManager` API operation and pass the `ManagerArn` value for the organization setup in your account. The format for the manager ARN used to set up the unified console is as follows:

`arn:aws:ssm-quicksetup:`*`account-id`*`:configuration-manager/`*`configuration-manager-id`*.

> **Note**
>
> If you don't know the value for *`configuration-manager-id`*, call the `ListConfigurationManagers` API action and filter the results using the `AWSQuickSetupType-SSM` type.

Performing node management tasks with AWS Systems Manager

The following topics describe how to complete common node tasks using the unified AWS Systems Manager console for an AWS Organizations organization and single AWS accounts.

Topics

- Reviewing node insights
- Exploring nodes
- Just-in-time node access using Systems Manager
- Diagnosing and remediating
- Adjusting Systems Manager settings

Reviewing node insights

You can gain insights into the overall status of managed nodes and unmanaged EC2 instances in your organization or account by using the unified Systems Manager console.

Systems Manager provides a visual overview into your managed nodes and EC2 instances that are not yet managed by Systems Manager. (A managed node is any machine configured for use with Systems Manager in hybrid and multicloud environments. For information about supported machine types, see Supported machine types in hybrid and multicloud environments.)

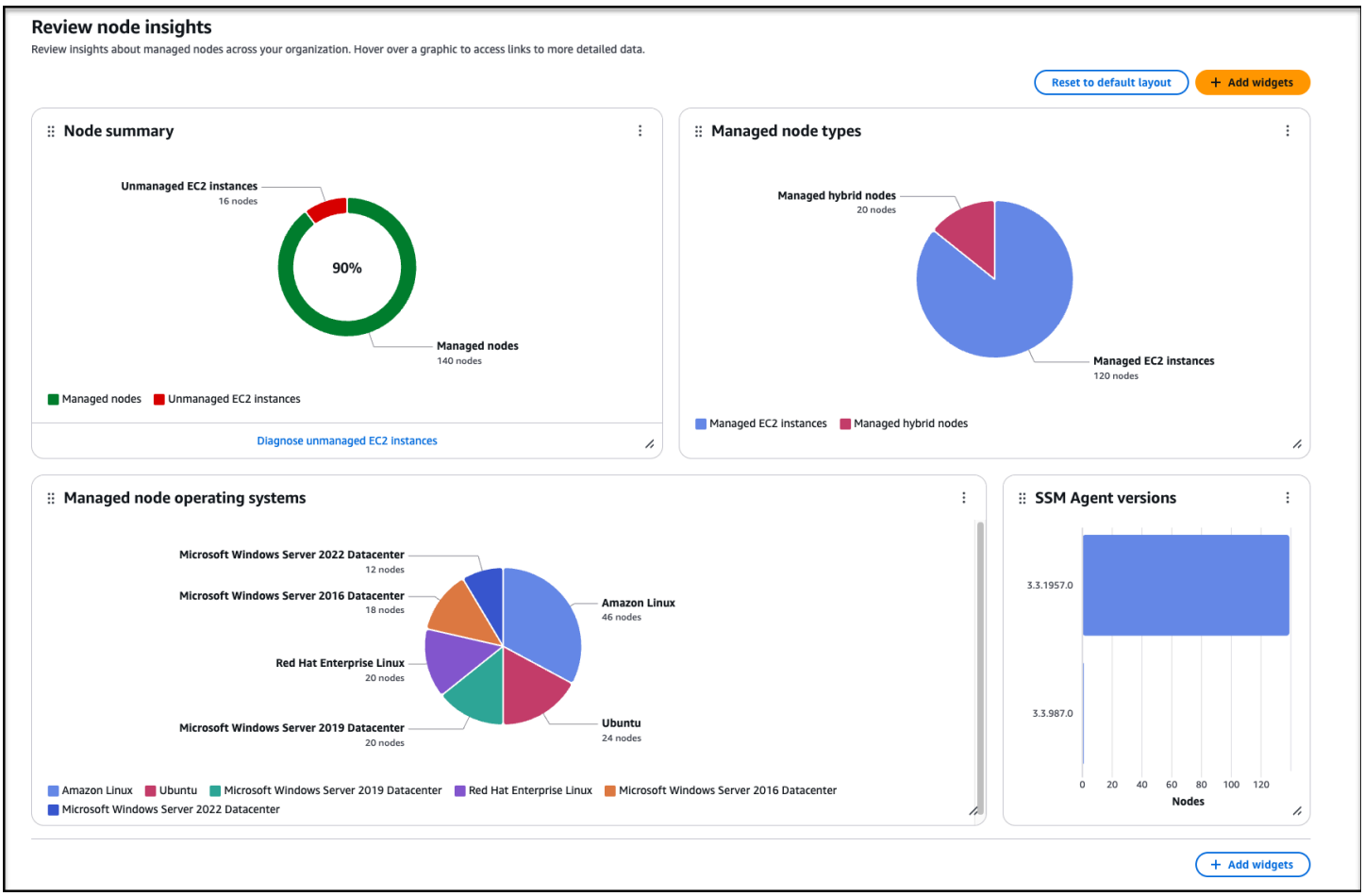
This overview is provided through individual report boxes, called *widgets*, which feature interactive pie charts and other graphics.

Before you begin

In order to review node insights, you must first onboard your organization or account to the unified Systems Manager console. For more information, see Setting up AWS Systems Manager.

After onboarding, open the Systems Manager console and choose **Review node insights**.

The following image shows the individual report boxes, called *widgets*, which are available on the **Review node insights** page.



The display supports widgets that provide you with the following information.

Node summary

Indicates how many EC2 instances in your organization or account aren't currently managed nodes, and how many managed nodes are in your organization's or account's fleet.

What is an unmanaged instance?

When you stop a managed EC2 instance, it is reported as "Unmanaged" in the Systems Manager console. This is expected behavior because SSM Agent doesn't have an active connection to the service.

> **Note**
>
> This is different from how AWS Config defines an instance as unmanaged. If an instance is currently stopped, AWS Config reports what the status of the instance was the last

time a "heartbeat" connection was made between SSM Agent on the instance and the Systems Manager service.

When the instance restarts, it automatically reconnects to the Systems Manager service, and its status in the unified console is restored to "Managed" within five minutes. No manual intervention is required, and all Systems Manager configurations for the instance are preserved during the Stop/Start cycle.

However, if the instance is still not reported as "Managed" several minutes after starting, the instance is likely not properly configured for Systems Manager management. In this case, we recommend running a diagnosis to identify why the instance remains in an unmanaged state. For more information, see Diagnosing and remediating unmanaged Amazon EC2 instances in Systems Manager.

If the diagnostic scan is not able to determine the issue, refer to the following topics to verify that the requirements for SSM Agent, AWS Identity and Access Management (IAM) roles, and Systems Manager prerequisites have all been met:

- Troubleshooting SSM Agent
- Configure instance permissions required for Systems Manager
- Troubleshooting managed node availability

Managed node types

Indicates how many managed nodes in your fleet are EC2 instances and how many are other server types, including servers on your own premises (on-premises servers), AWS IoT Greengrass core devices, AWS IoT and non-AWS edge devices, and virtual machines (VMs), including VMs in other cloud environments. You can hover over the **Node types** graphic to access links to more details in the **Explore nodes** page.

For more information about AWS support for hybrid and multicloud environments, see AWS Solutions for Hybrid and Multicloud.

SSM Agent versions

Provides information about installations of AWS Systems Manager Agent (SSM Agent) in your fleet. SSM Agent is Amazon software that runs on your managed nodes. 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.

For managed nodes in your fleet, this widget reports on the SSM Agent versions in your fleet, from newest to oldest. You can hover over the **SSM Agent versions** graphic to access links to more details in the **Explore nodes** page.

For more information about SSM Agent, see Working with SSM Agent.

Managed node operating systems

Provides a breakdown of the percentage of each operating system on managed nodes in your fleet. You can hover over the **Managed nodes by operating systems** graphic to access links to more details in the **Explore nodes** page.

You can customize widget layout on the **Review node insights** page by using a drag-and-drop capability, and by removing and adding widgets to the display.

Use information in the following topics to help you work with the Systems Manager node insights widgets.

Topics

- Adding or removing widgets from the Review node insights page
- Rearranging widgets in the Review node insights page

Adding or removing widgets from the Review node insights page

You can customize the layout in the Systems Manager **Review node insights** page by adding and removing widgets.

> **Note**
>
> By default, the page displays all available widgets.

To add or remove widgets from the Review node insights page

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the left navigation, choose **Review node insights**.
3. To remove a widget from the display, do the following:

a. Choose the More options menu
()
for the widget.

b. Choose **Remove widget**.

4. To add a widget to the display, do the following:

a. Choose **Add widgets**.

b. In the **Add widgets** pane, click and hold the drag handle
()
of the widget to add to the display.

c. Drag the widget and drop it into the main pane.

Rearranging widgets in the Review node insights page

You can customize the layout in the **Review node insights** page by rearranging widgets.

To rearrange widgets in the Review node insights page

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Review node insights**.
3. To customize the widget layout, choose a widget that you want to move. Click and hold the drag handle
()
of the widget and then drag it to its new location.
4. Repeat this process for each widget that you want to reposition.

If you decide that you don't like the new layout, choose **Reset to default layout** to move all widgets back to their original location.

Exploring nodes

You can use the **Explore nodes** page in Systems Manager to review details of managed nodes in your organization or account according to the criteria you specify in filters. You can also use

Systems Manager integration with Amazon Q Developer (Amazon Q), an AWS generative AI solution, to search using text prompts.

Before you begin

In order to use the **Explore nodes** feature, you must first onboard your organization or account to the unified Systems Manager console. For more information, see Setting up Systems Manager unified console for an organization.

After onboarding, open the Systems Manager console and choose **Explore nodes**.

> **Note**
>
> If you've created an aggregator index for Resource Explorer in a Region different than your home Region, Systems Manager demotes the current index. Then, Systems Manager promotes the local index in your home Region as the new aggregator index. During this time, only nodes for your home Region are displayed. This process can take up to 24 hours to complete.

Topics

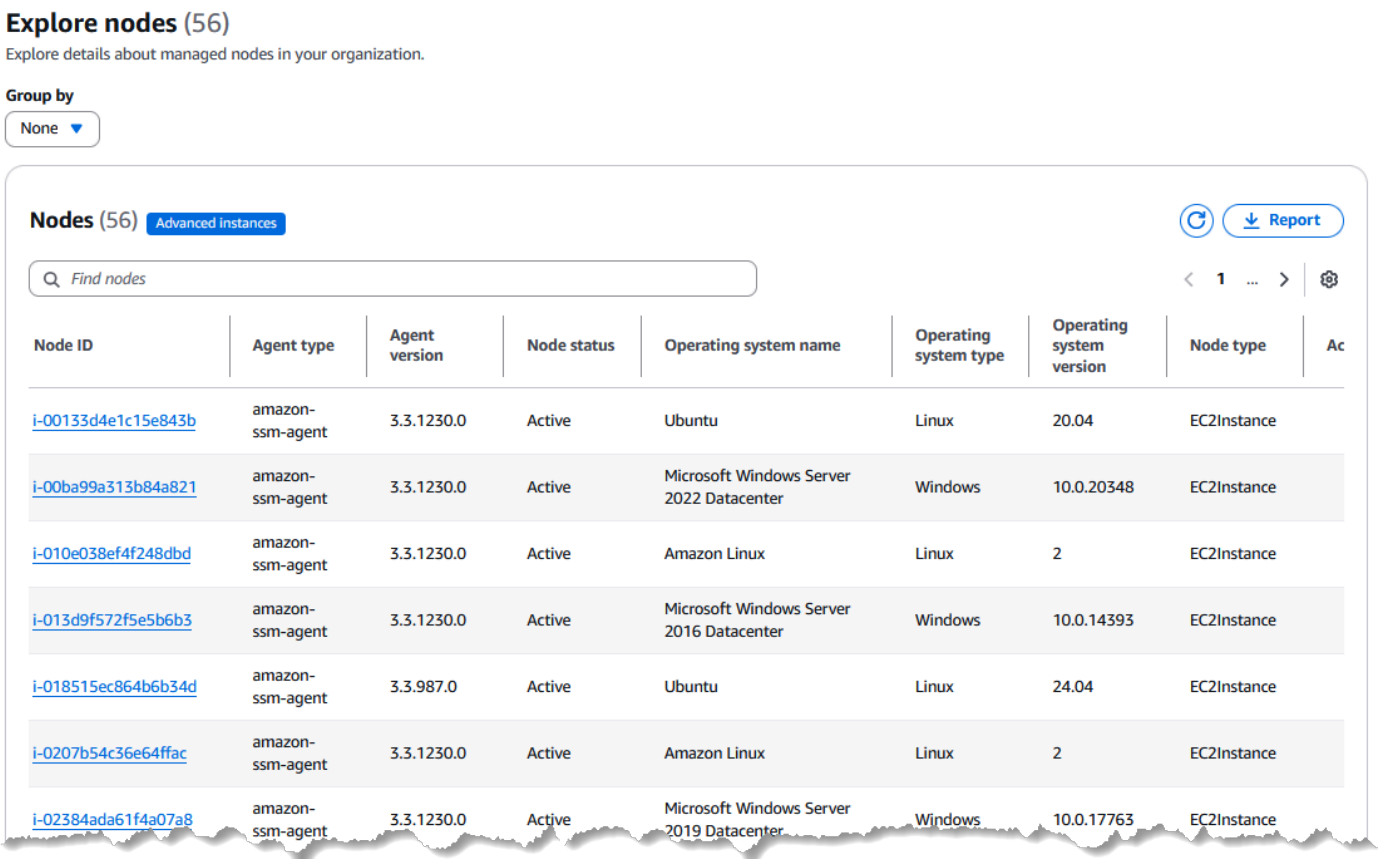
- Exploring nodes using console filters
- Exploring nodes using text prompts in Amazon Q
- Viewing individual node details and taking action on a node
- Downloading or exporting a managed node report
- Managing node report content and appearance

Exploring nodes using console filters

In the Systems Manager console, you can then group your managed nodes according to the following views:

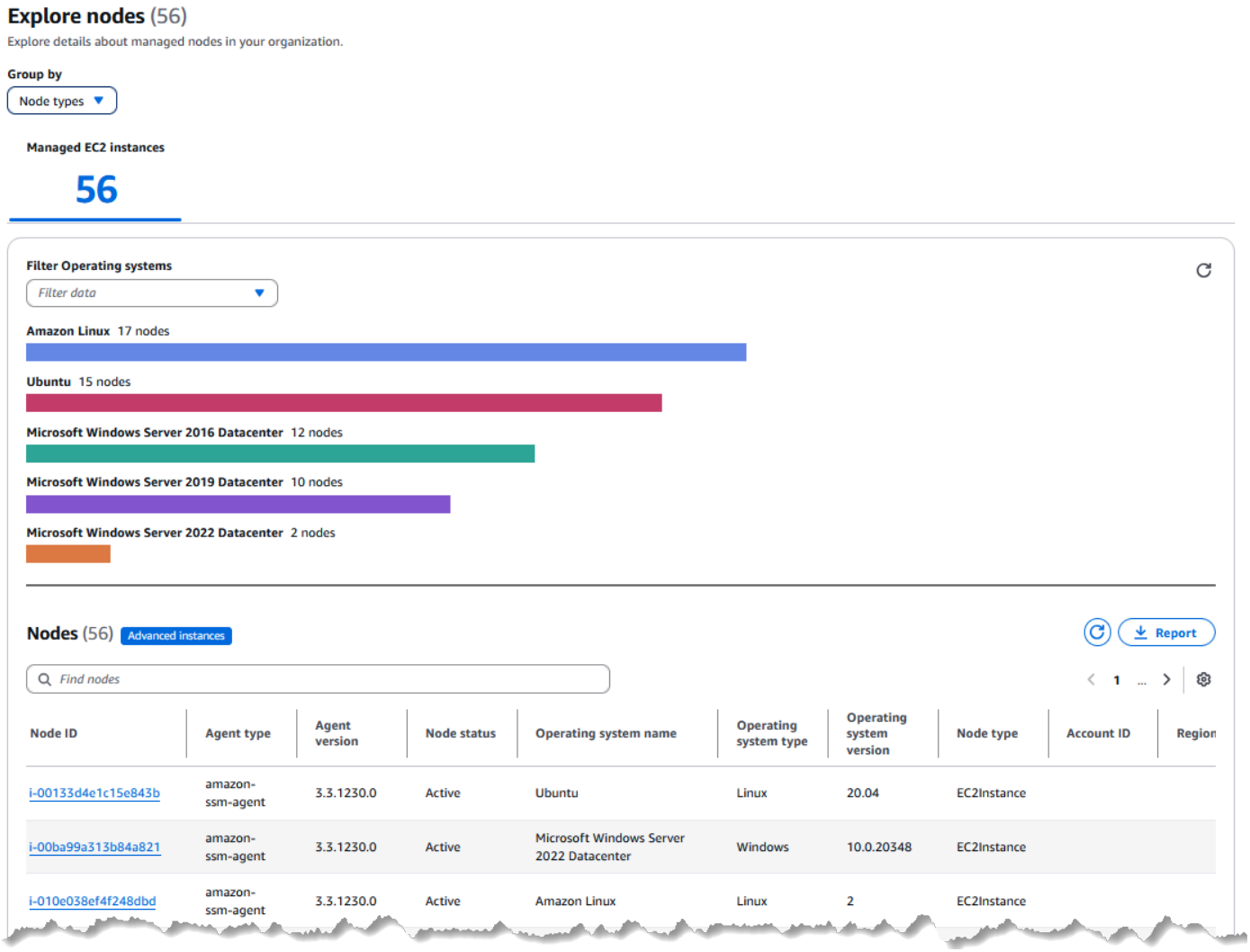
All nodes (No filter)

Lists all managed nodes in your organization or account.



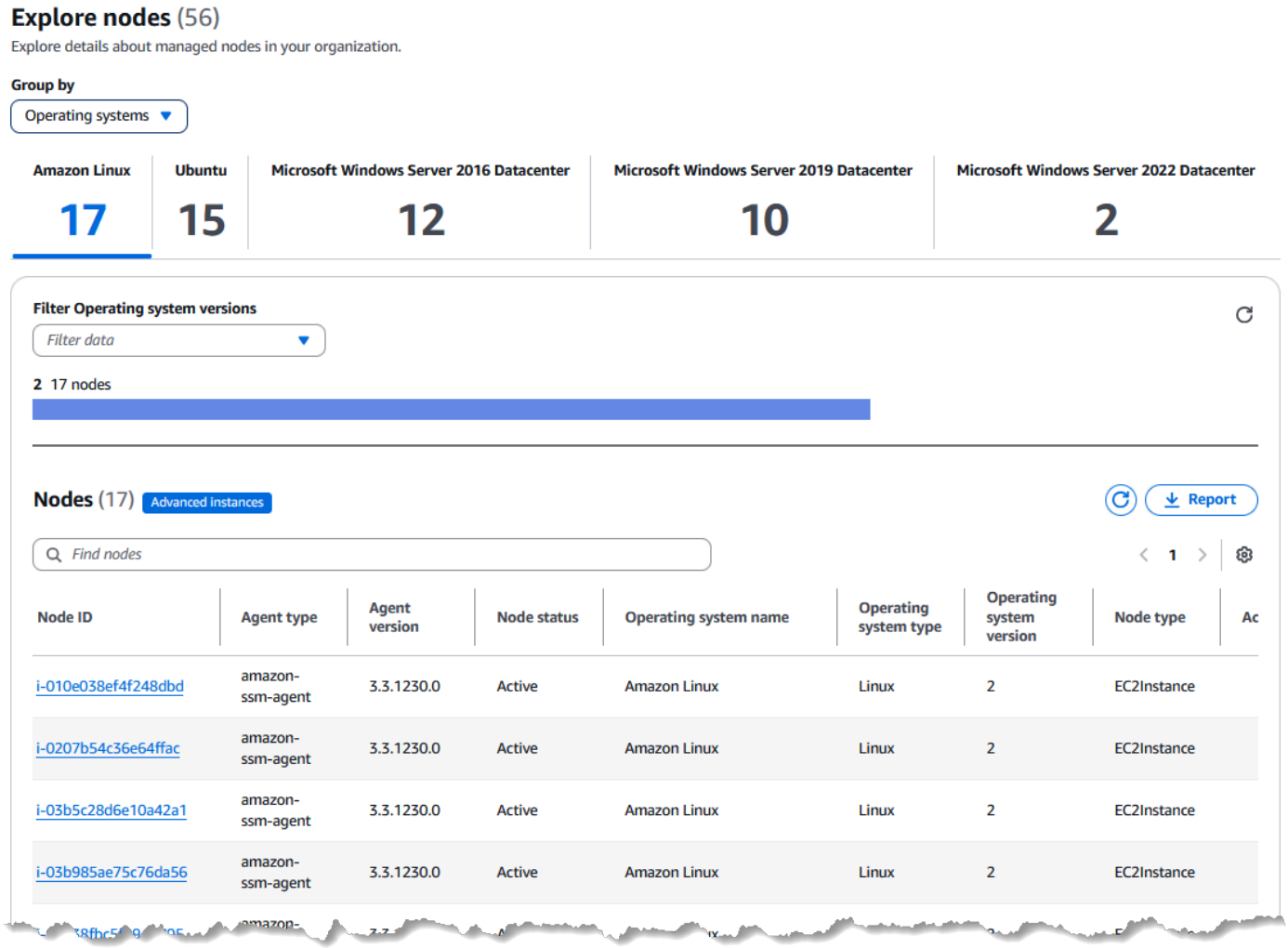
Node types

Provides tabs for viewing data separately for Amazon Elastic Compute Cloud (Amazon EC2) instances and other machine types, including servers on your own premises (on-premises servers), AWS IoT Greengrass core devices, AWS IoT and non-AWS edge devices, and virtual machines (VMs), including VMs in other cloud environments.



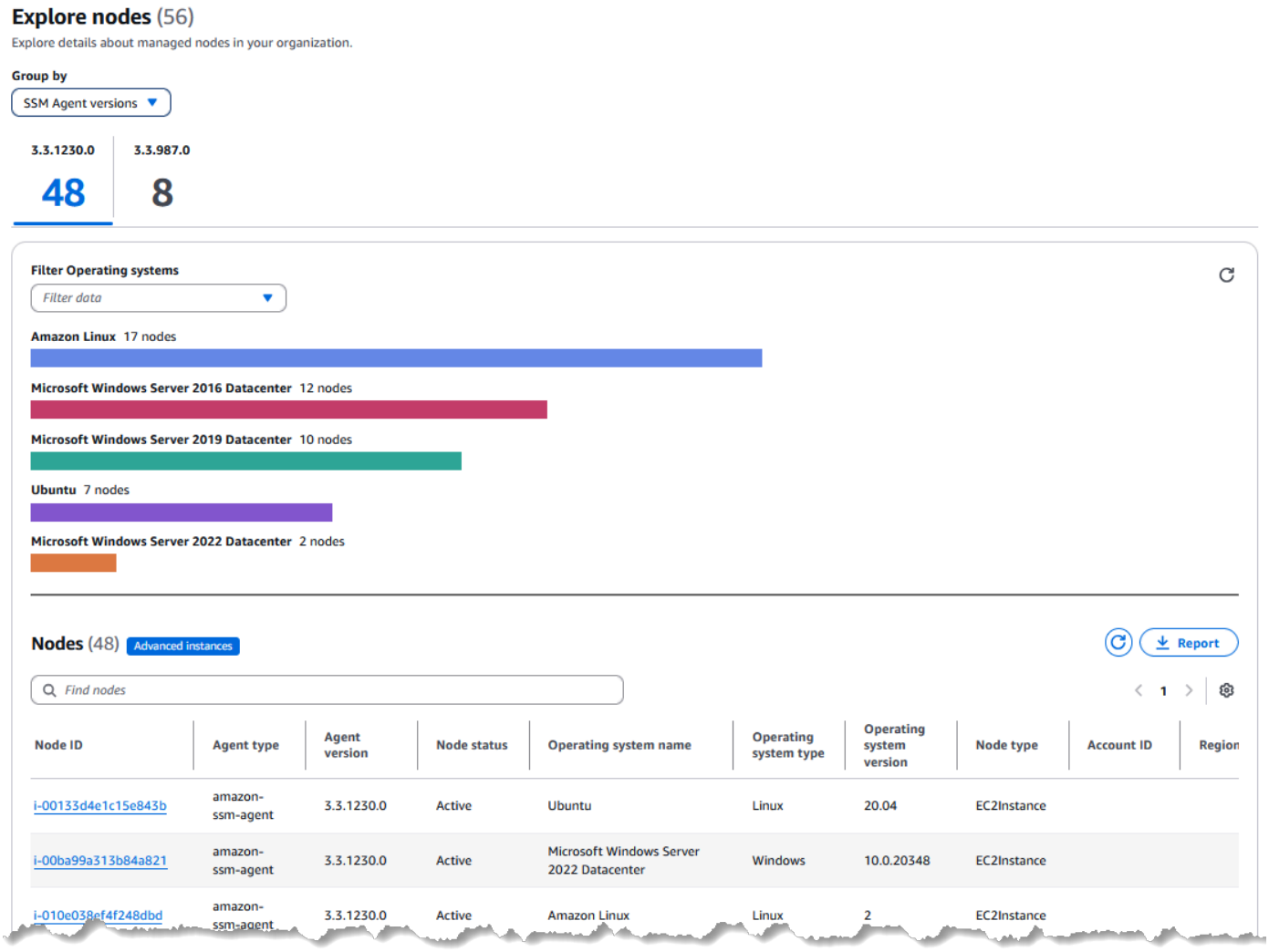
Operating systems

Provides a tab for each operating system type in your organization or account, such as **Amazon Linux** and **Microsoft Windows Server 2022 Datacenter**. On each tab, you can further filter the list by selecting only specific versions of the operating systems, such as *Amazon Linux 2* and *Amazon Linux 2023*.



SSM Agent versions

Provides a tab for each version of SSM Agent installed on managed nodes in your fleet. On each tab, you can further filter the list by selecting only specific operating systems, such as **Amazon Linux** and **Microsoft Windows Server 2022 Datacenter**.



In addition, for each of these views, you can further refine the list of nodes reported by choosing to view only nodes for a certain property, such as node status, AWS account ID, organization unit ID, and more.

You can customize the report display by choosing which of the available data columns are displayed in the **Explore nodes** page. You can also download reports in CSV or JSON formats, or export reports to Amazon S3 in CSV format.

Topics

- Choosing a filter view for managed node summaries

Choosing a filter view for managed node summaries

The **Explore nodes** page in Systems Manager lets you view aggregated data about your fleet according to a number of available filter views.

To choose a filter view for managed node summaries

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Explore nodes**.
3. For **Filter view**, select one of the filter options and optionally further refine the report:

   - **Managed nodes** – In the search () box, you can select a property and delimiter, such as `Node type = Managed EC2 instances`.
   - **Operating systems** – In the **Filter Operating system versions** list, you can select an OS version number. In the search () box, you can select a property and delimiter, such as `Node type = Managed EC2 instances`.
   - **SSM Agent versions** – In the **Filter Operating systems** list, you can select an OS name. In the search () box, you can select a property and delimiter, such as `Node type = Managed EC2 instances`.
   - **Node types** – In the **Filter Operating systems** list, you can select an OS name. In the search () box, you can select a property and delimiter, such as `Node type = Managed EC2 instances`.

After optionally filtering the list, you can view details about a specific managed node by choosing its ID in the **Node ID** column. From that detailed view, you can perform a number of actions on the node.

Exploring nodes using text prompts in Amazon Q

Using Systems Manager integration with Amazon Q Developer, you can use text prompts to view information created by generative AI about your managed nodes.

Amazon Q Developer is a generative AI-powered conversational assistant that can help you to understand, build, extend, and operate AWS applications. To accelerate your building on AWS, the model that powers Amazon Q is augmented with high-quality AWS content to produce more complete, actionable, and referenced answers. For more information, see What is Amazon Q Developer? in the *Amazon Q Developer User Guide*.

The integration between Systems Manager and Amazon Q lets you quickly get visibility and control over large, distributed environments across multiple AWS accounts and Regions. You can use natural language querying to quickly search node data, and then identify issues and take action faster.

When you ask a natural language question about managed nodes or managed instances, Amazon Q uses the Systems Manager `ListNodes` action and creates filters based on your textual input to retrieve results.

For example, say that you give Amazon Q the following prompt:

`List my managed nodes running Red Hat Enterprise Linux 9.2`

Amazon Q determines what filters to include in a request, and then runs a query similar to the following:

```
aws ssm list-nodes \
    --filters Key=PlatformName,Values='Red Hat Enterprise Linux',Type=Equal
 Key=PlatformVersion,Values=9.2,Type=Equal
```

Amazon Q then generates a report about Red Hat Enterprise Linux instances in your account, listing information such as the number of instances, their IDs, and their Regions.

You can also view a JSON summary of each instance's details, as well open a link to view the entire lists of EC2 instances or managed nodes in the Systems Manager **Explore nodes** page. The **Explore nodes** displays results that match the filter criteria you included in your prompt. From there, you can change or refine the filters for your request, as described in Exploring nodes.

Topics

- Learning to craft effective prompts to ask Amazon Q about your fleet
- Exploring managed nodes using Amazon Q

Learning to craft effective prompts to ask Amazon Q about your fleet

The better quality of the question, or the prompt, that you give to Amazon Q, the better the result it provides you with.

Tips for query prompts

Keep in mind the following tips when querying Amazon Q about your fleet:

1. To help improve the accuracy of your results, use the terms "managed nodes" and "managed instances" in your prompts instead of just "nodes" and "instances".
2. To query for results across multiple accounts that are part of an *organization*, as configured in AWS Organizations, you must be logged into the delegated administrator account in the designated home Region.
3. In the delegated administrator account, use terms to help Amazon Q understand that you are asking about nodes and instances across the organization by specifically using terms such as "in my organization" or "in my account 123456789012".

Topics

- Sample questions for Amazon Q
- Supported operating system names and versions for prompts

Sample questions for Amazon Q

In the following table, we provide sample questions demonstrate some ways you can create queries of Amazon Q that lead to better results.

We also provide examples of the filters Amazon Q will apply when running the ListNodes command, which are generated from the content of your prompt.

Sample natural language question	Amazon Q applied filters
`Show me my Windows managed nodes.`	`PlatformType = Windows`

Sample natural language question	Amazon Q applied filters
List my managed instances in account 123456789012.	`AccountId = 123456789012`
Show me all managed nodes running Amazon Linux 2 across my organization.	`PlatformName = Amazon Linux` `PlatformVersion = 2`
Show me all managed instances running Microsoft Windows Server 2019 Datacenter in my organization.	`PlatformName = Microsoft Windows Server 2019 Datacenter`
Can you show me all managed nodes with SSM Agent version 3.3.1142.0?	`AgentType = amazon-ssm-agent` `AgentVersion = 3.3.1142.0`

List all Amazon Linux 2 managed instances in account 123456789012 that have SSM Agent version 3.3.1230.0.	`PlatformName = Amazon Linux`
	`PlatformVersion = 2`
	`AccountId = 123456789012`
	`AgentType = amazon-ssm-agent`
	`AgentVersion = 3.3.1230.0`

What Microsoft Windows Server 2012 R2 Enterprise managed nodes are running in the eu-central-1 region across my entire organization?	`PlatformName = Microsoft Windows Server 2012 R2 Enterprise` `Region = eu-central-1`

Show me all managed instances running Red Hat Linux 7 in ou-d6ty-gxdma6vm.	`PlatformName = RHEL Linux`
	`PlatformVersion = 7`
	`OrganizationalUnitId = ou-d6ty-gxdma6vm`

What Ubuntu managed instances are in account 123456789012?	`PlatformName = Ubuntu` `AccountId = 123456789012`

Sample natural language question	Amazon Q applied filters
List my Linux managed instances.	`PlatformType = Linux`
Find my macOS managed nodes.	`PlatformType = macOS`
Show me all versions of Amazon Linux managed nodes in my org.	`PlatformName = Amazon Linux`
List managed nodes running Amazon Linux 2.	`PlatformName = Amazon Linux` `PlatformVersion = 2`

List the managed nodes with Ubuntu 16.04 in account 123456789012.	`PlatformName = Ubuntu`
	`PlatformVersion = 16.04`
	`AccountId = 123456789012`

Find all managed nodes that have an SSM Agent version that is not 3.3.987.0.	`AgentType = amazon-ssm-agent` `AgentVersion != 3.3.987.0`
List all managed instances that are not running a Linux operating system.	`PlatformType != Linux`

Supported operating system names and versions for prompts

When you are asking Amazon Q about the managed nodes in your account, it's helpful to provide the name of an operating system as its labeled in Systems Manager. You can also provide version numbers to further narrow down your results. For example, as represented in the following tables, you could ask for results specifically about **`macOS 14.5`**, **`Microsoft Windows Server 2019 Datacenter`**, and **`AlmaLinux 9.2 through 9.4`**, to name just a few examples.

These lists might not be exhaustive and are offered as examples only.

macOS

Platform name	Version numbers
macOS	13.2, 13.4, 13.7, 14.1, 14.5, 14.6.1, 15.0

Windows

Releases	Version numbers
Microsoft Windows Server 2012 R2 Datacenter	6.3.9600
Microsoft Windows Server 2012 R2 Standard	6.3.9600
Microsoft Windows Server 2012 Standard	6.2.9200
Microsoft Windows Server 2016 Datacenter	*N/A*
Microsoft Windows Server 2016 Standard	10.0.14393
Microsoft Windows Server 2019 Datacenter	*N/A*
Microsoft Windows Server 2019 Standard	*N/A*
Microsoft Windows Server 2022 Datacenter	*N/A*
Microsoft Windows Server 2022 Standard	10.0.20348

Linux

Platform names	Version numbers
AlmaLinux	8.10, 9.2, 9.3, 9.4
Amazon Linux 2	2.0 and greater
Amazon Linux 2023	2023.0.20230315.0 and greater

Platform names	Version numbers
BottleRocket	1.14.3, 1.16.1, 1.18.0, 1.19.1, 1.19.2, 1.19.5, 1.20.0, 1.20.1, 1.20.2, 1.20.3, 1.20.5, 1.21.1, 1.23.0, 1.24.0, 1.24.1, 1.25.0, 1.26.1,
Debian GNU/Linux	11-12
Oracle Linux Server	7.8, 8.2, 8.3, 8.8, 8.9, 8.10, 9.4
Red Hat Enterprise Linux	8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 9.2, 9.3, 9.4
Red Hat Enterprise Linux Server	17.3, 7.6, 7.7, 7.8,7.9
Rocky Linux	8.6, 8.7, 8.8, 8.9, 8.10, 9.1, 9.2, 9.3, 9.4
Ubuntu Server	16.04, 18.04, 20.04, 22.04, 24.04

Exploring managed nodes using Amazon Q

Systems Manager integration with Amazon Q Developer lets you ask questions about managed nodes in your fleet from anywhere in the AWS Management Console where the Amazon Q interface is available.

For more information about interacting with Amazon Q, see Chatting with Amazon Q Developer about AWS in the *Amazon Q Developer User Guide*.

To explore managed nodes using Amazon Q

1. From anywhere in the AWS Management Console, choose the Amazon Q icon ().
2. In the prompt field at the bottom of the Amazon Q pane, ask a question about managed nodes in your account or organization.

> **Tip**
>
> For tips on creating effective prompts, review the information in Learning to craft effective prompts to ask Amazon Q about your fleet .

3. Examine information about specific nodes, or choose **Open AWS Systems Manager console** to continue exploring.

Viewing individual node details and taking action on a node

From a list in the **Explore nodes** page in Systems Manager, you can select an individual node in order to view comprehensive details about the machine or perform a variety of actions on the node. The **General** page in the detail page presents comprehensive information about the node.

To view individual node details and take action on a node

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Explore nodes**.
3. (Optional) Follow the steps in Choosing a filter view for managed node summaries to refine the list of managed nodes displayed for your organization or account.
4. In the **Node ID** column, choose the linked ID of a node.
5. To view more details about the node, in the left navigation, in the **Properties** list, choose a property to view more information about:

   - **Tags** – View a list of tags applied to the node. You can also add or remove tags.
   - **Inventory** – Choose an inventory type, such as **AWS:Application** or **AWS:Network**, to view inventory details for the node.
   - **Associations** – View details about all State Manager associations applied to the node, including details such as status and associated SSM document name.
   - **Patches** – View summary information about patches and patch status for the node.
   - **Configuration compliance** – View compliance details for the node, such as compliance status and compliance issue severity.

For more information about details on the tabs, see the section called "What is the unified console?".

6. To take actions on the node, use the following options in the **Node actions** menu:

> **Note**
>
> These actions are available only for managed nodes in the AWS account and Region you are currently working in. For managed nodes you might have access to in other accounts or Regions, you can instead access a **Properties** list.

- **Connect, Start terminal session** – Connect to the node using AWS Systems Manager Session Manager.
- **Tools**
  - **View file system** – Browse the contents of the node's directory structure. Add, rename, and remove directories. Cut or copy and paste files.
  - **View performance counters** – View performance information about the node such as CPU utilization, network traffic, and other utilization types.
  - **Managed processes** – View information about resource usage on the node. Start or stop processes on the node.
  - **Manage users and groups** – View, add, or delete user accounts and user groups on the node.
  - **Execute run command** – Use AWS Systems Manager Run Command to manage the configuration of the node. Run Command uses Systems Manager documents to perform on-demand changes such as updating applications or running Linux shell scripts and Windows PowerShell commands.
  - **Patch nodes** – Use the **Patch now** feature in AWS Systems Manager Patch Manager to run an on-demand patching operation on the node from the console.

  > **Note**
  >
  > The preceding tasks can also be initiated from the **Tools** menu in the left navigation.

- **Node settings**

- **Add tags** – Apply additional tag key-value pairs to the node.
- **Reset node user password** – Set a new password for a specified user on the node.
- **Modify IAM role** – Change the IAM role that's associated with the node. Create a new IAM role to attach to the node.

Downloading or exporting a managed node report

You can use the Systems Manager **Explore nodes** feature to view filtered or unfiltered lists of managed nodes for your AWS organization or account in the Systems Manager console. For cases where you want to view the data offline or process it in another application, you can save the report as a `CSV` or `JSON` file.

Depending on the size of the report, you're prompted to download the report to your local machine or export it to an Amazon S3 bucket. Reports are saved to S3 buckets in `CSV` format only.

To download or export a managed node report

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Explore nodes**.
3. (Optional) Follow the steps in Choosing a filter view for managed node summaries to refine the list of managed nodes displayed for your organization or account.
4. Choose **Report** ().
5. If the **Download report** dialog box is displayed, do the following:

   a. For **File name**, enter a name for the file. We recommend specifying a name that represents the scope of the report, such as `all-organization-nodes` or `ec2-instances-out-of-date-agent`.

   b. For **Included columns**, specify whether to include columns for all available node details, or only those you've selected for your current display.

   > **Tip**
   >
   > For information about managing the columns in your report display, see Managing node report content and appearance.

c. For **File format**, select **CSV** or **JSON**, depending on how you will use the file.

d. For **Spreadsheet heading**, to include a column headings row in a CSV file, select **Include row of column names**.

e. Choose **Download**.

The report is saved to the default download location according to your browser's settings.

6. If the **Export to Amazon S3** dialog box is displayed, do the following:

a. For **S3 URI**, enter the URI for the bucket to export the report to.

> **Tip**
>
> To view a list of your buckets in the Amazon S3 console, choose **View**. To select from a list of buckets in your account, choose **Browse S3**.

b. For **Authorization method**, specify the service role to use to provide permissions for exporting the report to the bucket.

If you choose to let Systems Manager create the role for you, it provides all needed permissions and trust statements for the operation.

If you want to use or create your own role, the role must include the required permissions and trust statements. For information about creating this role, see Creating a custom service role to export diagnosis reports to S3.

c. Choose **Submit**.

Creating a custom service role to export diagnosis reports to S3

When you are viewing filtered or unfiltered lists of managed nodes for your AWS organization or account in the Systems Manager **Explore nodes** page, you can export the list as a report to an Amazon S3 bucket as a CSV file.

To do so, you must specify a service role with the necessary permissions and trust policy for the operation. You can choose for Systems Manager to create the role for you during the process of downloading the report. Optionally, you can create the role and its required policy yourself.

To create a custom service role to export diagnosis reports to S3

1. Follow the steps in Creating policies using the JSON editor in the *IAM User Guide*.

   - Use the following for the policy content, making sure to replace the *placeholder values* with your own information.

     JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:ListBucket",
        "s3:PutLifecycleConfiguration",
        "s3:GetLifecycleConfiguration"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:ListNodes"
```

```
      ],
      "Resource": "*"
    }
  ]
}
```

- Give the policy a name to help you recognize it easily in the next step.

2. Follow the steps in Creating an IAM role using a custom trust policy (console) in the *IAM User Guide*.

   - For step 4, enter the following trust policy, making sure to replace the *placeholder values* with your own information.

     JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SSMAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

3. For step 10, choose **Step 2: Add permissions** and select the name of the policy you created in the previous step.

After you create the role, you can select it when following the steps in Downloading or exporting a managed node report.

Managing node report content and appearance

You can use the Systems Manager **Explore nodes** feature to view filtered or unfiltered lists of managed nodes for your AWS organization or account in the Systems Manager console. You can choose from over a dozen fields to include in your lists of nodes, such as **Node ID**, **Operating system name**, **Region**, and more. You can also reorder the columns for your lists and reports and change how the list is displayed in the console.

To manage node report content and appearance

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Explore nodes**.
3. In the **Nodes** area, choose the preferences gear icon ().
4. In the **Preferences** dialog box, do the following:
   a. For **Page size**, choose how many rows are included in each console view display, **10**, **25**, or **50**.
   b. For **Wrap lines**, select the box to display all content of a cell in the available column width.
   c. For **Striped rows**, select the box to display alternating rows of clear and shaded backgrounds.
   d. For **Select visible content**, do the following:
      - Turn on or off individual columns for your list display and reports.
      - To change the order of columns, click and hold the drag handle () of a column name and drag it up or down in the list.
5. Choose **Confirm**.

Just-in-time node access using Systems Manager

Systems Manager helps you to improve the security of your nodes by supporting *just-in-time* access. Just-in-time node access allows users to request temporary, time-bound access to nodes that you can approve only when access is truly needed. This removes the need to provide long

standing access to nodes managed by IAM policies. Additionally, Systems Manager provides session recording for RDP sessions to Windows Server nodes to help you meet compliance requirements, perform root cause analysis, and more. To use just-in-time node access, you must set up the unified Systems Manager console.

With just-in-time node access, you create granular IAM policies to ensure only the users you permit can submit access requests to your nodes. Then you create *approval policies* which define the approvals required to connect to your nodes. For just-in-time node access, there are *auto-approval* policies and *manual approval* policies. An auto-approval policy defines which nodes users can connect to automatically. Manual approval policies define the number and levels of manual approvals that must be provided to access the nodes you specify. Also, you can create a *deny-access* policy. A deny-access policy explicitly prevents the auto-approval of access requests to the nodes you specify. A deny-access policy applies to all accounts in an AWS Organizations organization. Auto-approval and manual approval policies apply only to the AWS accounts and AWS Regions where they're created.

When a user attempts to connect to a node, they're prompted to enter a reason for accessing the node. Then your approval policies are evaluated. Depending on your policies, users either connect automatically to the target node or Systems Manager automatically creates a manual approval request on the requester's behalf. The approvers specified in the manual approval policy that applies to the node are then notified of the access request, and can approve or deny the request. Approvers and requesters can be notified by email, or through Amazon Q Developer in chat applications integration with Slack or Microsoft Teams. Systems Manager only grants access to requested nodes when the specified approvers provide all required approvals. Once all of the required approvals are received, the user can start as many sessions to the node as needed for the duration of the access window specified in the approval policy. Systems Manager doesn't automatically terminate just-in-time node access sessions. As a best practice, specify values for the *maximum session duration* and *idle session timeout* session preferences. These preferences prevent users from staying connected to nodes beyond their approved access window.

We recommend using a combination of approval policies to help you secure nodes with more critical data while allowing users to connect to less critical nodes without intervention. For example, you can require manual approvals for access requests to database nodes, and auto-approve sessions to non-persistent presentation tier nodes.

Systems Manager supports just-in-time node access for users federated with IAM Identity Center or IAM. When a federated user submits an access request, they specify the target node, and the reason for needing to connect to the node. Systems Manager compares the user's identity to

the parameters defined in your organization's approval policies. When the auto-approval policy conditions are met, or approvers manually provide approvals, the requester is able to connect to the target node. When a user attempts to connect to an approved node, Systems Manager creates and uses a temporary token to establish the session.

Since the Systems Manager service handles the authentication for access requests and establishing sessions, you don't have to use IAM policies to manage access to your nodes. By using just-in-time node access, Systems Manager helps your organization move closer to zero standing privileges since you only need to allow users to create access requests instead of allowing them to start sessions with persistent permissions to your nodes. To help you meet compliance requirements, Systems Manager retains all access requests for 1 year. Systems Manager also emits EventBridge events for just-in-time node access for failed access requests and status updates to access requests for manual approvals. For more information see, Monitoring Systems Manager events with Amazon EventBridge.

Topics

- Setting up just-in-time access with Systems Manager
- Start a just-in-time node access session
- Managing just-in-time access requests
- Moving to just-in-time node access from Session Manager
- Disabling just-in-time access with Systems Manager
- Just-in-time node access frequently asked questions

Setting up just-in-time access with Systems Manager

Setting up just-in-time node access with Systems Manager involved multiple steps. First, you choose the *targets* where you want to set up just-in-time node access. Targets consist of AWS Organizations organizational units (OUs) and AWS Regions. By default, the same targets you chose when setting up the unified Systems Manager console are selected for just-in-time node access. You can choose to set up just-in-time node access for all of the same targets, or a subset of the targets you specified when setting up the unified Systems Manager console. Adding new targets that weren't selected when you set up the unified Systems Manager console isn't supported.

Next you'll create *approval policies* to determine when node connections require manual approval and are automatically approved. Approval policies are managed by each account in your

organization. You can also share a policy from the delegated administrator account to explicitly deny the automatic approval of connections to specific nodes.

> **Note**
>
> Setting up just-in-time node access doesn't affect existing IAM policies or preferences you've configured for Session Manager. You must remove permission to the `StartSession` API action from your IAM policies to ensure that only just-in-time node access is used when users attempt to connect to your nodes. After you set up just-in-time node access, we recommend testing your approval policies with a subset of users and nodes to verify your policies are working as desired before removing permissions to Session Manager.

Authentication support

Note the following details about authentication support used for just-in-time node access:

- Just-in-time node access doesn't support the Single Sign-On authentication type when connecting to Windows Server instances with Remote Desktop.
- Only AWS Security Token Service (AWS STS) `AssumeRole` temporary security credentials are supported. For more information, see the following topics in the *IAM User Guide*:
- • Temporary security credentials in IAM
  - Comparing AWS STS credentials
  - Requesting temporary security credentials

The following IAM policies outline the permissions needed to administer and allow users to create just-in-time node access requests to nodes with Systems Manager. After verifying you have the required permissions to use just-in-time node access with Systems Manager, you can continue the setting up process. Replace each *`example resource placeholder`* with your own information.

IAM policy for enabling just-in-time node access

JSON

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

```
            "Sid": "QuickSetupConfigurationManagers",
            "Effect": "Allow",
            "Action": [
                "ssm-quicksetup:CreateConfigurationManager",
                "ssm-quicksetup:DeleteConfigurationManager",
                "ssm-quicksetup:GetConfiguration",
                "ssm-quicksetup:GetConfigurationManager",
                "ssm-quicksetup:GetServiceSettings",
                "ssm-quicksetup:ListConfigurationManagers",
                "ssm-quicksetup:ListConfigurations",
                "ssm-quicksetup:ListQuickSetupTypes",
                "ssm-quicksetup:ListTagsForResource",
                "ssm-quicksetup:TagResource",
                "ssm-quicksetup:UntagResource",
                "ssm-quicksetup:UpdateConfigurationDefinition",
                "ssm-quicksetup:UpdateConfigurationManager",
                "ssm-quicksetup:UpdateServiceSettings"
            ],
            "Resource": "*"
        },
        {
            "Sid": "QuickSetupDeployments",
            "Effect": "Allow",
            "Action": [
                "cloudformation:DescribeStackSetOperation",
                "cloudformation:ListStacks",
                "cloudformation:DescribeStacks",
                "cloudformation:DescribeStackResources",
                "cloudformation:ListStackSetOperations",
                "cloudformation:ListStackInstances",
                "cloudformation:DescribeStackSet",
                "cloudformation:ListStackSets",
                "cloudformation:DescribeStackInstance",
                "cloudformation:DescribeOrganizationsAccess",
                "cloudformation:ActivateOrganizationsAccess",
                "cloudformation:GetTemplate",
                "cloudformation:ListStackSetOperationResults",
                "cloudformation:DescribeStackEvents",
                "cloudformation:UntagResource",
                "ssm:DescribeAutomationExecutions",
                "ssm:GetAutomationExecution",
                "ssm:ListAssociations",
                "ssm:DescribeAssociation",
                "ssm:GetDocument",
```

```
                "ssm:ListDocuments",
                "ssm:DescribeDocument",
                "ssm:GetOpsSummary",
                "organizations:DeregisterDelegatedAdministrator",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization",
                "organizations:ListDelegatedAdministrators",
                "organizations:ListRoots",
                "organizations:ListParents",
                "organizations:ListOrganizationalUnitsForParent",
                "organizations:DescribeOrganizationalUnit",
                "organizations:ListAWSServiceAccessForOrganization",
                "iam:ListRoles",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:CreatePolicy",
                "cloudformation:TagResource"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:RollbackStack",
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack",
                "cloudformation:DeleteStack"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-
JITNA*",
                "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
                "arn:aws:cloudformation:*:*:type/resource/*",
                "arn:aws:cloudformation:*:*:stack/StackSet-SSMQuickSetup"
            ]
        },
        {
            "Sid": "StackSetOperations",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStackSet",
                "cloudformation:UpdateStackSet",
                "cloudformation:DeleteStackSet",
                "cloudformation:DeleteStackInstances",
```

```
                "cloudformation:CreateStackInstances",
                "cloudformation:StopStackSetOperation"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-JITNA*",
                "arn:aws:cloudformation:*:*:type/resource/*",
                "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-
JITNA*:*"
            ]
        },
        {
            "Sid": "IamRolesMgmt",
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "iam:DetachRolePolicy",
                "iam:GetRolePolicy",
                "iam:ListRolePolicies"
            ],
            "Resource": [
                "arn:aws:iam::*:role/AWS-QuickSetup-JITNA*",
                "arn:aws:iam::*:role/service-role/AWS-QuickSetup-JITNA*"
            ]
        },
        {
            "Sid": "IamPassRole",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/AWS-QuickSetup-JITNA*",
                "arn:aws:iam::*:role/service-role/AWS-QuickSetup-JITNA*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ssm.amazonaws.com",
                        "ssm-quicksetup.amazonaws.com",
                        "cloudformation.amazonaws.com"
```

```
                    ]
                }
            }
        },
        {
            "Sid": "SSMAutomationExecution",
            "Effect": "Allow",
            "Action": "ssm:StartAutomationExecution",
            "Resource": "arn:aws:ssm:us-east-1:111122223333:automation-
definition/AWS-EnableExplorer:*"
        },
        {
            "Sid": "SSMAssociationPermissions",
            "Effect": "Allow",
            "Action": [
                "ssm:DeleteAssociation",
                "ssm:CreateAssociation",
                "ssm:StartAssociationsOnce"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:association/*"
        },
        {
            "Sid": "SSMResourceDataSync",
            "Effect": "Allow",
            "Action": [
                "ssm:CreateResourceDataSync",
                "ssm:UpdateResourceDataSync"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:resource-data-sync/
AWS-QuickSetup-*"
        },
        {
            "Sid": "ListResourceDataSync",
            "Effect": "Allow",
            "Action": [
                "ssm:ListResourceDataSync"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateServiceLinkedRoles",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
```

```
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "accountdiscovery.ssm.amazonaws.com",
                        "ssm.amazonaws.com",
                        "ssm-quicksetup.amazonaws.com",
                        "stacksets.cloudformation.amazonaws.com"
                    ]
                }
            },
            "Resource": "*"
        },
        {
            "Sid": "CreateStackSetsServiceLinkedRole",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-
service-role/stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin"
        },
        {
            "Sid": "AllowSsmJitnaPoliciesCrudOperations",
            "Effect": "Allow",
            "Action": [
                "ssm:CreateDocument",
                "ssm:UpdateDocument",
                "ssm:UpdateDocumentDefaultVersion",
                "ssm:GetDocument",
                "ssm:DescribeDocument",
                "ssm:DeleteDocument"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:document/SSM-
JustInTimeAccessDenyAccessOrgPolicy"
            ],
            "Condition": {
                "StringEquals": {
                    "ssm:DocumentType": [
                        "AutoApprovalPolicy"
                    ]
                }
```

```
            }
        },
        {
            "Sid": "AllowAccessRequestOpsItemOperations",
            "Effect": "Allow",
            "Action": [
                "ssm:GetOpsItem",
                "ssm:DescribeOpsItems",
                "ssm:GetOpsSummary",
                "ssm:DeleteOpsItem",
                "ssm:ListOpsItemEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IdentityCenterPermissions",
            "Effect": "Allow",
            "Action": [
                "sso:DescribeRegisteredRegions",
                "sso:ListDirectoryAssociations",
                "identitystore:GetUserId",
                "identitystore:DescribeUser",
                "identitystore:DescribeGroup",
                "identitystore:ListGroupMembershipsForMember"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM policy for configuring just-in-time node access

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSsmJitnaPoliciesCrudOperations",
            "Effect": "Allow",
            "Action": [
                "ssm:CreateDocument",
```

```
                "ssm:UpdateDocument",
                "ssm:UpdateDocumentDefaultVersion",
                "ssm:GetDocument",
                "ssm:DescribeDocument",
                "ssm:DeleteDocument"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:document/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ssm:DocumentType": [
                        "ManualApprovalPolicy",
                        "AutoApprovalPolicy"
                    ]
                }
            }
        },
        {
            "Sid": "AllowSsmJitnaPoliciesListOperations",
            "Effect": "Allow",
            "Action": [
                "ssm:ListDocuments",
                "ssm:ListDocumentVersions"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/SSM-
JustInTimeAccessTokenRole",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "justintimeaccess.ssm.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AllowAccessRequestOpsItemOperations",
            "Effect": "Allow",
            "Action": [
```

```
                "ssm:GetOpsItem",
                "ssm:DescribeOpsItems",
                "ssm:GetOpsSummary",
                "ssm:DeleteOpsItem",
                "ssm:ListOpsItemEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowSessionManagerPreferencesOperation",
            "Effect": "Allow",
            "Action": [
                "ssm:CreateDocument",
                "ssm:GetDocument",
                "ssm:DescribeDocument",
                "ssm:UpdateDocument",
                "ssm:DeleteDocument"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell",
            "Condition": {
                "StringEquals": {
                    "ssm:DocumentType": "Session"
                }
            }
        },
        {
            "Sid": "AllowSessionManagerOperations",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeSessions",
                "ssm:GetConnectionStatus",
                "ssm:TerminateSession"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowRDPConnectionRecordingOperations",
            "Effect": "Allow",
            "Action": [
                "ssm-guiconnect:UpdateConnectionRecordingPreferences",
                "ssm-guiconnect:GetConnectionRecordingPreferences",
                "ssm-guiconnect:DeleteConnectionRecordingPreferences"
            ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "AllowRDPConnectionRecordingKmsOperation",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/SystemsManagerJustInTimeNodeAccessManaged":
"true"
                },
                "StringLike": {
                    "kms:ViaService": "ssm-guiconnect.*.amazonaws.com"
                },
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        },
        {
            "Sid": "AllowFleetManagerOperations",
            "Effect": "Allow",
            "Action": [
                "ssm-guiconnect:GetConnection",
                "ssm-guiconnect:ListConnections"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SNSTopicManagement",
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:SetTopicAttributes"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1:111122223333:SSM-JITNA*"
            ]
        },
        {
            "Sid": "SNSListTopics",
```

```
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EventBridgeRuleManagement",
            "Effect": "Allow",
            "Action": [
                "events:PutRule",
                "events:PutTargets"
            ],
            "Resource": [
                "arn:aws:events:us-east-1::rule/SSM-JITNA*"
            ]
        },
        {
            "Sid": "ChatbotSlackManagement",
            "Effect": "Allow",
            "Action": [
                "chatbot:CreateSlackChannelConfiguration",
                "chatbot:UpdateSlackChannelConfiguration",
                "chatbot:DescribeSlackChannelConfigurations",
                "chatbot:DescribeSlackWorkspaces",
                "chatbot:DeleteSlackChannelConfiguration",
                "chatbot:RedeemSlackOauthCode",
                "chatbot:DeleteSlackWorkspaceAuthorization",
                "chatbot:GetSlackOauthParameters"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ChatbotTeamsManagement",
            "Effect": "Allow",
            "Action": [
                "chatbot:ListMicrosoftTeamsChannelConfigurations",
                "chatbot:CreateMicrosoftTeamsChannelConfiguration",
                "chatbot:UpdateMicrosoftTeamsChannelConfiguration",
                "chatbot:ListMicrosoftTeamsConfiguredTeams",
                "chatbot:DeleteMicrosoftTeamsChannelConfiguration",
                "chatbot:RedeemMicrosoftTeamsOauthCode",
                "chatbot:DeleteMicrosoftTeamsConfiguredTeam",
                "chatbot:GetMicrosoftTeamsOauthParameters",
```

```
                "chatbot:TagResource"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SSMEmailSettings",
            "Effect": "Allow",
            "Action": [
                "ssm:UpdateServiceSetting",
                "ssm:GetServiceSetting"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:servicesetting/ssm/access-
request/email-role-mapping",
                "arn:aws:ssm:us-east-1:111122223333:servicesetting/ssm/access-
request/enabled-email-notifications"
            ]
        },
        {
            "Sid": "AllowViewingJitnaCloudWatchMetrics",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "AWS/SSM/JustInTimeAccess"
                }
            }
        },
        {
            "Sid": "QuickSetupConfigurationManagers",
            "Effect": "Allow",
            "Action": [
                "ssm-quicksetup:ListConfigurationManagers",
                "ssm-quicksetup:ListConfigurations",
                "ssm-quicksetup:ListQuickSetupTypes",
                "ssm-quicksetup:GetConfiguration",
                "ssm-quicksetup:GetConfigurationManager"
            ],
            "Resource": "*"
```

```
        },
        {
            "Sid": "QuickSetupDeployments",
            "Effect": "Allow",
            "Action": [
                "cloudformation:ListStacks",
                "cloudformation:DescribeStacks",
                "organizations:DescribeOrganization",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ManualPolicy",
            "Effect": "Allow",
            "Action": [
                "sso:DescribeRegisteredRegions",
                "ssm:GetServiceSetting",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SessionPreference",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowIamListForKMS",
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers"
            ],
            "Resource": "arn:aws:iam::111122223333:user/*"
        },
        {
            "Sid": "KMSPermission",
            "Effect": "Allow",
            "Action": [
                "kms:TagResource",
                "kms:ListAliases",
```

```
                "kms:CreateAlias"
            ],
            "Resource": "*"
        },
        {
            "Sid": "KMSCreateKey",
            "Effect": "Allow",
            "Action": [
                "kms:CreateKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/SystemsManagerJustInTimeNodeAccessManaged":
"true"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "SystemsManagerJustInTimeNodeAccessManaged"
                    ]
                }
            }
        },
        {
            "Sid": "AllowIamRoleForChatbotAction",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/role name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "chatbot.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AllowIamServiceRoleForChat",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
```

```
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
management.chatbot.amazonaws.com/AWSServiceRoleForAWSChatbot"
        },
        {
            "Sid": "CloudWatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": "arn:aws:logs:*:111122223333:log-group::log-stream:"
        },
        {
            "Sid": "IdentityStorePermissions",
            "Effect": "Allow",
            "Action": [
                "sso:ListDirectoryAssociations",
                "identitystore:GetUserId",
                "sso-directory:SearchUsers",
                "sso-directory:SearchGroups",
                "identitystore:DescribeGroup",
                "identitystore:DescribeUser"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM policy for access request approvers

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccessRequestDescriptions",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeOpsItems",
                "ssm:GetOpsSummary",
                "ssm:ListOpsItemEvents"
            ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "AllowGetSpecificAccessRequest",
            "Effect": "Allow",
            "Action": [
                "ssm:GetOpsItem"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:opsitem/*"
        },
        {
            "Sid": "AllowApprovalRejectionSignal",
            "Effect": "Allow",
            "Action": [
                "ssm:SendAutomationSignal"
            ],
            "Resource": "arn:aws:ssm:*:*:automation-execution/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/SystemsManagerJustInTimeNodeAccessManaged":
"true"
                }
            }
        },
        {
            "Sid": "QuickSetupConfigurationManagers",
            "Effect": "Allow",
            "Action": [
                "ssm-quicksetup:ListConfigurationManagers",
                "ssm-quicksetup:ListConfigurations",
                "ssm-quicksetup:GetConfigurationManager",
                "ssm-quicksetup:ListQuickSetupTypes",
                "ssm-quicksetup:GetConfiguration"
            ],
            "Resource": "*"
        },
        {
            "Sid": "QuickSetupDeployments",
            "Effect": "Allow",
            "Action": [
                "cloudformation:ListStacks",
                "cloudformation:DescribeStacks",
                "organizations:DescribeOrganization",
                "organizations:ListDelegatedAdministrators"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowSsmJitnaPoliciesCrudOperations",
            "Effect": "Allow",
            "Action": [
                "ssm:GetDocument",
                "ssm:DescribeDocument"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:document/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ssm:DocumentType": [
                        "ManualApprovalPolicy",
                        "AutoApprovalPolicy"
                    ]
                }
            }
        },
        {
            "Sid": "AllowSsmJitnaPoliciesListOperations",
            "Effect": "Allow",
            "Action": [
                "ssm:ListDocuments",
                "ssm:ListDocumentVersions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IDCPermissions",
            "Effect": "Allow",
            "Action": [
                "sso:DescribeRegisteredRegions",
                "sso:ListDirectoryAssociations",
                "identitystore:GetUserId",
                "identitystore:DescribeUser",
                "identitystore:DescribeGroup",
                "identitystore:ListGroupMembershipsForMember"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

IAM policy for just-in-time node access users

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowJITNAOperations",
            "Effect": "Allow",
            "Action": [
                "ssm:StartAccessRequest",
                "ssm:GetAccessToken"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowOpsItemCreationAndRetrieval",
            "Effect": "Allow",
            "Action": [
                "ssm:CreateOpsItem",
                "ssm:GetOpsItem"
            ],
            "Resource": "arn:aws:ssm:*:*:opsitem/*"
        },
        {
            "Sid": "AllowListAccessRequests",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeOpsItems",
                "ssm:GetOpsSummary",
                "ssm:ListOpsItemEvents",
                "ssm:DescribeSessions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RequestManualApprovals",
            "Action": "ssm:StartAutomationExecution",
```

```
            "Effect": "Allow",
            "Resource": "arn:aws:ssm:*:*:document/*",
            "Condition": {
                "StringEquals": {
                    "ssm:DocumentType": "ManualApprovalPolicy"
                }
            }
        },
        {
            "Sid": "StartManualApprovalsAutomationExecution",
            "Effect": "Allow",
            "Action": "ssm:StartAutomationExecution",
            "Resource": "arn:aws:ssm:*:*:automation-execution/*"
        },
        {
            "Sid": "AllowManualApprovalAutomationExecutionTagging",
            "Effect": "Allow",
            "Action": [
                "ssm:AddTagsToResource"
            ],
            "Resource": [
                "arn:aws:ssm:*:*:automation-execution/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/SystemsManagerJustInTimeNodeAccessManaged":
"true"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "SystemsManagerJustInTimeNodeAccessManaged"
                    ]
                }
            }
        },
        {
            "Sid": "CancelAccessRequestManualApproval",
            "Effect": "Allow",
            "Action": "ssm:StopAutomationExecution",
            "Resource": "arn:aws:ssm:*:*:automation-execution/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/SystemsManagerJustInTimeNodeAccessManaged":
"true"
```

```
                }
            }
        },
        {
            "Sid": "DescribeEC2Instances",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:GetPasswordData"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowListSSMManagedNodesAndTags",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeInstanceInformation",
                "ssm:ListTagsForResource"
            ],
            "Resource": "*"
        },
        {
            "Sid": "QuickSetupConfigurationManagers",
            "Effect": "Allow",
            "Action": [
                "ssm-quicksetup:ListConfigurationManagers",
                "ssm-quicksetup:GetConfigurationManager",
                "ssm-quicksetup:ListConfigurations",
                "ssm-quicksetup:ListQuickSetupTypes",
                "ssm-quicksetup:GetConfiguration"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowSessionManagerOperations",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeSessions",
                "ssm:GetConnectionStatus"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "AllowRDPOperations",
            "Effect": "Allow",
            "Action": [
                "ssm-guiconnect:ListConnections",
                "ssm:GetConnectionStatus"
            ],
            "Resource": "*"
        },
        {
            "Sid": "QuickSetupDeployments",
            "Effect": "Allow",
            "Action": [
                "cloudformation:ListStacks",
                "cloudformation:DescribeStacks",
                "organizations:DescribeOrganization",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowSsmJitnaPoliciesReadOnly",
            "Effect": "Allow",
            "Action": [
                "ssm:GetDocument",
                "ssm:DescribeDocument"
            ],
            "Resource": [
                "arn:aws:ssm:*:111122223333:document/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ssm:DocumentType": [
                        "ManualApprovalPolicy",
                        "AutoApprovalPolicy"
                    ]
                }
            }
        },
        {
            "Sid": "AllowSsmJitnaPoliciesListOperations",
            "Effect": "Allow",
            "Action": [
                "ssm:ListDocuments",
                "ssm:ListDocumentVersions"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "ExploreNodes",
            "Effect": "Allow",
            "Action": [
                "ssm:ListNodesSummary",
                "ssm:ListNodes",
                "ssm:DescribeInstanceProperties"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IdentityStorePermissions",
            "Effect": "Allow",
            "Action": [
                "sso:DescribeRegisteredRegions",
                "sso:ListDirectoryAssociations",
                "identitystore:GetUserId",
                "identitystore:DescribeUser",
                "identitystore:DescribeGroup"
            ],
            "Resource": "*"
        }
    ]
}
```

Note

To restrict access to API operations that create, update, or delete approval policies, use the `ssm:DocumentType` condition key for the `AutoApprovalPolicy` and `ManualApprovalPolicy` document types. The `StartAccessRequest` and `GetAccessToken` API operations don't support the following global context keys:

- aws:ViaAwsService
- aws:MultiFactorAuthPresent
- aws:SourceVpce
- aws:UserAgent

For more information about condition context keys for Systems Manager, see Condition keys for AWS Systems Manager in the *Service Authorization Reference*.

The following procedure describes how to complete the first set up step for just-in-time node access.

To set up just-in-time node access

1. Log in to the Systems Manager delegated administrator account for your organization.
2. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
3. Select **Just-in-time node access** in the navigation pane.
4. Select **Enable the new experience**.
5. Choose the Regions where you want to enable just-in-time node access. By default, the same Regions you chose when setting up the unified Systems Manager console are selected for just-in-time node access. Choosing new Regions that weren't selected when you set up the unified Systems Manager console isn't supported.
6. Select **Enable just-in-time node access**.

There is no charge to use just-in-time node access for 30 days after enabling the feature. After the 30 day trial period, there is a charge to use just-in-time node access. For more information, see AWS Systems Manager Pricing.

Create approval policies for your nodes

Approval policies define what approvals users need to access a node. Since just-in-time node access removes the need for long standing permissions to nodes through IAM policies, you must create approval policies to allow access to your nodes. If there are no approval policies that apply to a node, users are unable to request access to the node.

In just-in-time node access, there are three types of policies. The policy types are *auto-approval*, *deny-access*, and *manual approval*.

Just-in-time node access policy types

- An auto-approval policy defines which nodes users can connect to automatically.
- Manual approval policies define the number and levels of manual approvals that must be provided to access the nodes you specify.

- A deny-access policy explicitly prevents the auto-approval of access requests to the nodes you specify.

A deny-access policy applies to all accounts in an AWS Organizations organization. For example, you could explicitly deny auto-approvals for the `Intern` group to nodes tagged with the `Production` key. Auto-approval and manual approval policies apply only to the AWS accounts and AWS Regions where they're created. Each member account in your organization manages their own approval policies. Approval policies are evaluated in the following order:

1. Deny-access
2. Auto-approval
3. Manual

While you can only have one deny-access policy per organization, and one auto-approval policy per account and Region, you'll likely have several manual approval policies in an account. When evaluating manual approval policies, just-in-time node access always favors the more specific policy for a node. Manual approval policies are evaluated in the following order:

1. Tag specific target
2. All nodes target

For example, you have a node tagged with the `Demo` key. In the same account, you have a manual approval policy that targets all nodes and requires one approval from one level. You also have a manual approval policy that requires two approvals from two levels for nodes tagged with the `Demo` key. Systems Manager applies the policy that targets the `Demo` tag to the node since it's more specific than the policy that targets all nodes. This allows you to create a general policy for all nodes in your account, ensuring users can submit access requests while enabling you to create more granular policies as needed.

Depending on your organization, there might be multiple tags applied to your nodes. In this scenario, if multiple manual approval policies apply to a node, access requests fail. For example, a node is tagged with the `Production` and `Database` keys. In the same account, you have a manual approval policy that applies to nodes tagged with the `Production` key and another manual approval policy that applies to nodes tagged with the `Database` key. This results in a conflict for the node tagged with both keys and access requests fail. Systems Manager redirects the user to the failed request. There, they can view details about the conflicting policies and tags so they can make

the necessary adjustments if they have the required permissions. Otherwise, they can notify a colleague in their organization with the required permissions to modify the policies. Policy conflicts resulting in failed access requests emit EventBridge events allowing you flexibility in building your own response workflows. Additionally, Systems Manager sends email notifications for policy conflicts resulting in failed access requests to the recipients you specify. For more information about configuring email notifications for policy conflicts, see Configure notifications for just-in-time access requests.

In a *deny-access* policy, you use the Cedar policy language to define which nodes users explicitly can't automatically connect to in your organization. This policy is created and shared from the delegated administrator account for your organization. The deny-access policy supercedes all auto-approval policies. You can only have one deny-access policy per organization.

In an *auto-approval* policy, you use the Cedar policy language to define which users can automatically connect to the specified nodes without manual approval. The access duration for an access request that is automatically approved is 1 hour. This value can't be changed. You can only have one auto-approval policy per account and Region.

In a *manual* approval policy, you specify the access duration, how many levels of approvals are required, the number of approvers required per level, and the nodes they can approve just-in-time access requests to. The access duration for a manual approval policy must be between 1 and 336 hours. If you specify multiple levels of approvals, the approvals for the access request process one level at a time. This means all approvals you require for one level must be provided before the approval process moves to subsequent levels. If you specify multiple tags in a manual approval policy, they are evaluated as `or` statements not `and` statements. For example, if you create a manual approval policy that includes the tags `Application`, `Web`, and `Test`, the policy applies to any node that is tagged with one of those keys. The policy doesn't apply only to nodes that are tagged with all three keys.

We recommend using a combination of manual policies with your auto-approval policy to help you secure nodes with more critical data while allowing users to connect to less critical nodes without intervention. For example, you can require manual approvals for access requests to database nodes, and auto-approve sessions to non-persistent presentation tier nodes.

The following procedures describe how to create approval policies for just-in-time node access.

Topics

- Create manual approval policies for just-in-time node access
- Statement structure and built-in operators for auto-approval and deny-access policies

- Create an auto-approval policy for just-in-time node access
- Create a deny-access policy for just-in-time node access
- Create approval policies for just-in-time node access with Amazon Q

Create manual approval policies for just-in-time node access

The following procedure describes how to create manual approval policies. Systems Manager allows you to create up to 50 manual approval policies per AWS account and AWS Region.

To create a manual approval policy

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Manage node access** in the navigation pane.
3. In the **Policy details** section of the **Create manual approval policy** step, enter a name and description for the approval policy.
4. Enter a value for the **Access duration**. This is the maximum amount of time a user can start sessions to a node after an access request is approved. The value must be between 1 and 336 hours.
5. In the **Node targets** section, enter tag key-value pairs associated with the nodes you want the policy to apply to. If none of the tags specified in the policy are associated with a node, the policy isn't applied to the node.
6. In the **Access request approvers** section, enter the users or groups you want to be able to approve access requests to the node targets in the policy. Access request approvers can be IAM Identity Center users and groups or IAM roles. You can specify up to 5 approvers per level, and up to 5 levels of approvers.
7. Select **Create manual approval policy**.

Statement structure and built-in operators for auto-approval and deny-access policies

The following table shows the structure of auto-approval and deny-access policies.

Component	Syntax
effect	`permit | forbid`

Component	Syntax
scope	`(principal, action, resource)`
condition clause	`when { principal or resource has attribute name };`

Policy components

An auto-approval or deny-access policy contains the following components:

- **Effect** – Either `permit` (allow) or `forbid` (deny) access.
- **Scope** – The principals, actions, and resources to which the effect applies. You can leave the scope in Cedar undefined by not identifying specific principals, actions, or resources. In this case, the policy applies to all possible principals, actions, and resources. For just-in-time node access, the `action` is always `AWS::SSM::Action::"getTokenForInstanceAccess"`.
- **Condition clause** – The context in which the effect applies.

Comments

You can include comments in your policies. Comments are defined as a line starting with `//` and ending with a newline character.

The following example shows comments in a policy.

```
// Allows users in the Engineering group from the Platform org to automatically connect
 to nodes tagged with Engineering and Production keys.
permit (
    principal in AWS::IdentityStore::Group::"d4q81745-r081-7079-d789-14da1EXAMPLE",
    action == AWS::SSM::Action::"getTokenForInstanceAccess",
    resource
)
when {
    principal has organization && resource.hasTag("Engineering") &&
 resource.hasTag("Production") && principal.organization == "Platform"
};
```

Multiple clauses

You can use more than one condition clause in a policy statement using the && operator.

```
// Allow access if node has tag where the tag key is Environment
// & tag value is Development

permit(principal, action == AWS::SSM::getTokenForInstanceAccess, resource)
when {
    resource.hasTag("Environment") &&
    resource.getTag("Environment") == "Development"
};
```

Reserved characters

The following example shows how to write a policy if a context property uses a : (semicolon), which is a reserved character in the policy language.

```
permit (
    principal,
    action == AWS::SSM::Action::"getTokenForInstanceAccess",
    resource
)
when {
    principal has employeeNumber && principal.employeeNumber like "E-1*" &&
 resource.hasTag("Purpose") && resource.getTag("Purpose") == "Testing"
}
```

For additional examples, see Example policy statements.

Just-in-time node access schema

The following is the Cedar schema for just-in-time node access.

```
namespace AWS::EC2 {
    entity Instance tags String;
}


namespace AWS::IdentityStore {
    entity Group;
```

```
    entity User in [Group] {
    employeeNumber?: String,
    costCenter?: String,
    organization?: String,
    division?: String,
    };

}


namespace AWS::IAM {

    entity Role;

    type AuthorizationContext = {
        principalTags: PrincipalTags,
    };

    entity PrincipalTags tags String;
}

namespace AWS::SSM {

    entity ManagedInstance tags String;

    action "getTokenForInstanceAccess" appliesTo {
    principal: [AWS::IdentityStore::User],
    resource: [AWS::EC2::Instance, AWS::SSM::ManagedInstance],
    context: {
        "iam": AWS::IAM::AuthorizationContext
        }
    };
}
```

Built-in operators

When creating the context of an auto-approval or deny-access policy using various conditions, you can use the && operator to add additional conditions. There are also many other built-in operators that you can use to add additional expressive power to your policy conditions. The following table contains all the built-in operators for reference.

Operator	Types and overloads	Description
!	Boolean → Boolean	Logical not.
==	any → any	Equality. Works on arguments of any type, even if the types don't match. Values of different types are never equal to each other.
!=	any → any	Inequality; the exact inverse of equality (see above).
<	(long, long) → Boolean	Long integer less-than.
<=	(long, long) → Boolean	Long integer less-than-or-equal-to.
>	(long, long) → Boolean	Long integer greater-than.
>=	(long, long) → Boolean	Long integer greater-than-or-equal-to.
in	(entity, entity) → Boolean	Hierarchy membership (reflexive: A in A is always true).
	(entity, set(entity)) → Boolean	Hierarchy membership: A in [B, C, ...] is true if (A and B) || (A in C) || … error if the set contains a non-entity.
&&	(Boolean, Boolean) → Boolean	Logical and (short-circuiting).
||	(Boolean, Boolean) → Boolean	Logical or (short-circuiting).
.exists()	entity → Boolean	Entity existence.
has	(entity, attribute) → Boolean	Infix operator. `e has f` tests if the record or entity e has

Operator	Types and overloads	Description
		a binding for the attribute `f`. Returns `false` if `e` does not exist or if `e` does exist but doesn't have the attribute `f`. Attributes can be expressed as identifiers or string literals.
like	(string, string) → Boolean	Infix operator. `t like p` checks if the text `t` matches the pattern `p`, which may include wildcard characters * that match 0 or more of any character. In order to match a literal star character in `t`, you can use the special escaped character sequence `\*` in `p`.
.hasTag()	(entity, string) → Boolean	Checks if entity has the specified tag applied.
.getTag()	(entity, string) → Boolean	Returns the value of the specified tag key.
.contains()	(set, any) → Boolean	Set membership (is B an element of A).
.containsAll()	(set, set) → Boolean	Tests if set A contains all of the elements in set B.
.containsAny()	(set, set) → Boolean	Tests if set A contains any of the elements in set B.

Example policy statements

The following are policy statement examples.

```
// Users assuming IAM roles with a principal tag of "Elevated" can automatically access
 nodes tagged with the "Environment" key when the value equals "prod"
permit(principal, action == AWS::SSM::getTokenForInstanceAccess, resource)
when {
    // Verify IAM role principal tag
    context.iam.principalTags.getTag("AccessLevel") == "Elevated" &&

    // Verify the node has a tag with "Environment" tag key and a tag value of "prod"
    resource.hasTag("Environment") &&
    resource.getTag("Environment") == "prod"
};
```

```
// Identity Center users in the "Contractor" division can automatically access nodes
 tagged with the "Environment" key when the value equals "dev"
permit(principal, action == AWS::SSM::getTokenForInstanceAccess, resource)
when {
    // Verify that the user is part of the "Contractor" division
    principal.division == "Contractor" &&

    // Verify the node has a tag with "Environment" tag key and a tag value of "dev"
    resource.hasTag("Environment") &&
    resource.getTag("Environment") == "dev"
};
```

```
// Identity Center users in a specified group can automatically access nodes tagged
 with the "Environment" key when the value equals "Production"
permit(principal in AWS::IdentityStore::Group::"d4q81745-r081-7079-d789-14da1EXAMPLE",
    action == AWS::SSM::getTokenForInstanceAccess,
    resource)
when {
    resource.hasTag("Environment") &&
    resource.getTag("Environment") == "Production"
};
```

Create an auto-approval policy for just-in-time node access

Auto-approval policies use the Cedar policy language to define which users can automatically connect to the specified nodes without manual approval. An auto-approval policy contains multiple `permit` statements specifying the `principal` and `resource`. Each statement includes a `when` clause defining the conditions for automatic approval.

The following is an example auto-approval policy.

```
permit (
    principal in AWS::IdentityStore::Group::"e8c17310-e011-7089-d989-10da1EXAMPLE",
    action == AWS::SSM::Action::"getTokenForInstanceAccess",
    resource
)
when {
    principal has costCenter && resource.hasTag("CostCenter") && principal.costCenter
 == resource.getTag("CostCenter")
};

permit (
    principal in AWS::IdentityStore::Group::"d4q81745-r081-7079-d789-14da1EXAMPLE",
    action == AWS::SSM::Action::"getTokenForInstanceAccess",
    resource
)
when {
    principal has organization && resource.hasTag("Engineering") &&
 resource.hasTag("Production") && principal.organization == "Platform"
};

permit (
    principal,
    action == AWS::SSM::Action::"getTokenForInstanceAccess",
    resource
)
when {
    principal has employeeNumber && principal.employeeNumber like "E-1*" &&
 resource.hasTag("Purpose") && resource.getTag("Purpose") == "Testing"
};
```

The following procedure describes how to create an auto-approval policy for just-in-time node acces. The access duration for an access request that is automatically approved is 1 hour. This value can't be changed. You can only have one auto-approval policy per AWS account and AWS Region. For more information about how to construct policy statements, see Statement structure and built-in operators for auto-approval and deny-access policies.

To create an auto-approval policy

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

2. Select **Manage node access** in the navigation pane.
3. In the **Approval policies** tab, select **Create an auto-approval policy**.
4. Enter your policy statement for the auto-approval policy in the **Policy statement** section. You can use the **Sample statements** provided to help you create your policy.
5. Select **Create auto-approval policy**.

Create a deny-access policy for just-in-time node access

Deny-access policies use the Cedar policy language to define which nodes users can't automatically connect to without manual approval. A deny-access policy contains multiple `forbid` statements specifying the `principal` and `resource`. Each statement includes a `when` clause defining the conditions for explicitly denying automatic approval.

The following is an example deny-access policy.

```
forbid (
    principal in AWS::IdentityStore::Group::"e8c17310-e011-7089-d989-10da1EXAMPLE",
    action == AWS::SSM::Action::"getTokenForInstanceAccess",
    resource
)
when {
    resource.hasTag("Environment") && resource.getTag("Environment") == "Production"
};

forbid (
    principal,
    action == AWS::SSM::Action::"getTokenForInstanceAccess",
    resource
)
when {
    principal has division && principal.division != "Finance" &&
 resource.hasTag("DataClassification") && resource.getTag("DataClassification") ==
 "Financial"
};


forbid (
    principal,
    action == AWS::SSM::Action::"getTokenForInstanceAccess",
    resource
)
```

```
when {

    principal has employeeNumber && principal.employeeNumber like "TEMP-*" &&
 resource.hasTag("Criticality") && resource.getTag("Criticality") == "High"
};
```

The following procedure describes how to create a deny-access policy for just-in-time node access. For more information about how to construct policy statements, see Statement structure and built-in operators for auto-approval and deny-access policies.

Note

Note the following information.

- You can create deny-access policies while logged into the AWS Management account or the delegated administrator account. Your AWS Organizations organization can have only one deny-access policy.
- Just-in-time node access uses AWS Resource Access Manager (AWS RAM) to share your deny-access policy with member accounts in your organization. If you would like to share your deny-access policy with the member accounts in your organization, resource sharing must be enabled from the management account of your organization. For more information, see Enable resource sharing within AWS Organizations in the *AWS RAM User Guide*.

To create a deny-access policy

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Manage node access** in the navigation pane.
3. In the **Approval policies** tab, select **Create a deny-access policy**.
4. Enter your policy statement for the deny-access policy in the **Policy statement** section. You can use the **Sample statements** provided to help you create your policy.
5. Select **Create deny-access policy**.

Create approval policies for just-in-time node access with Amazon Q

Using Amazon Q Developer for command line provides guidance and support across various aspects of software development. For just-in-time node access, Amazon Q helps you create approval policies by generating and updating the code for the policies, analyzing policy statements, and more. The following information describes how to create approval policies using Amazon Q for command line.

Identify your use case

The first step in creating approval policies is clearly defining your use case. For example, in your organization you might want to automatically approve access requests to nodes with an `Environment:Testing` tag. You might also want to explicitly deny auto-approvals to nodes with an `Environment:Production` tag if an employee ID begins with `TEMP`. For nodes with a `Tier:Database` tag, you might want to require two levels of manual approvals.

In any given scenario, you might prefer one policy or condition, over another. Therefore, we recommend that you clearly define the policy behaviors you want to determine which statements best fit your use case and preferences.

Set up your development environment

Install Amazon Q for command line where you want to develop your approval policies. For information about installing Amazon Q for the command line, see Installing Amazon Q for command line in the *Amazon Q Developer User Guide*.

We also recommend installing the MCP server for AWS documentation. This MCP server connects Amazon Q for command line to the most current documentation resources. For information about using MCP with Amazon Q for the command line, see Using MCP with Amazon Q Developer in the *Amazon Q Developer User Guide*.

For more information about the AWS Documentation MCP server see AWS Documentation MCP Server.

Install and configure the AWS CLI, if you have not already. For information, see Installing or updating the latest version of the AWS CLI.

Develop approval policy content

With your use case identified and environment set up, you're ready to develop the content for your policies. Your use case and preferences will largely dictate the types of approval policies and statements you use.

If you're unsure how to use a particular policy or need more information about the schema for a policy, see Create approval policies for your nodes and the subsequent topics. These topics explain in detail how policies are evaluated and provide working example statements.

The following procedure describes how to create approval policies with Amazon Q for command line.

> **Note**
>
> Amazon Q Developer uses generative AI. You may need to verify responses. See the AWS Responsible AI Policy.

To create a approval policies using Amazon Q for command line

1. Open your terminal and start a session with Amazon Q.

```
$ q chat
```

2. Enter a prompt and describe what kind of policy statement you want to create. For example, using the earlier scenario for an approval policy, you could enter the following prompt:

```
Create a Cedar policy statement that allows a principal to use the
 “getTokenForInstanceAccess" action for a resource
that has a tag key “Environment” and tag value of “Testing”. Use the following
 example statement for reference to verify the syntax.

permit (
    principal in AWS::IdentityStore::Group::"e8c17310-e011-7089-d989-10da1EXAMPLE",
    action == AWS::SSM::Action::"getTokenForInstanceAccess",
    resource
)
when {
    principal has organization && resource.hasTag("Engineering") &&
 resource.hasTag("Production") && principal.organization == "Platform"
};

The Cedar policy statement you create must be valid. Validate the Cedar statement
 using the AWS documentation provided for just-in-time node access policy statement
 structure and make sure you abide by the schema.
```

3. Entering this example prompt resulted in the following output:

```
Based on the AWS documentation for Cedar policy statements and the examples
 provided, I'll create a valid Cedar policy
statement that allows a principal to use the "getTokenForInstanceAccess" action for
 resources with the tag key "Environment" and tag value "Testing".

Here's the Cedar policy statement:

permit (
    principal,
    action == AWS::SSM::Action::"getTokenForInstanceAccess",
    resource
)
when {
    resource.hasTag("Environment") &&
    resource.getTag("Environment") == "Testing"
};


This policy statement:
1. Uses the permit effect to allow access
2. Applies to any principal (since no specific principal is defined)
3. Specifically targets the "getTokenForInstanceAccess" action
4. Applies to resources that have a tag with key "Environment" and value "Testing"
5. Uses the proper syntax for the condition clause with the hasTag() and getTag()
 operators

The statement follows the Cedar policy language structure as documented in the AWS
 Systems Manager documentation for just-in-time node access policies.
```

4. Paste the policy statement into a new Cedar file to validate the syntax. You can also validate the statement using the Cedar playground.
5. After validating the statement, save the Cedar file.
6. From your terminal, run the following AWS CLI command to create an SSM document using your policy content. Before using approval policies in a production environment, test your approval policies with a subset of identities and nodes in an AWS account and AWS Region.

> **Note**
>
> For an auto-approval policy, the document name must be `SSM-JustInTimeAccessAutoApprovalPolicy`. There can only be one auto-approval

policy per AWS account and AWS Region. For a deny-access policy, the document name must be `SSM-JustInTimeAccessDenyAccessOrgPolicy`. There can only be one deny-access policy per AWS Organizations organization, and the policy must be created in the delegated administrator account for Systems Manager. The naming constraints for manual approval policies are the same as other SSM documents. For more information, see CreateDocument.

Linux & macOS

```
aws ssm create-document \
    --content file://path/to/file/policyContent.cedar \
    --name "SSM-JustInTimeAccessAutoApprovalPolicy" \
    --document-type "AutoApproval"
```

Windows

```
aws ssm create-document ^
    --content file://C:\path\to\file\policyContent.cedar ^
    --name "SSM-JustInTimeAccessAutoApprovalPolicy" ^
    --document-type "AutoApproval"
```

PowerShell

```
$cedar = Get-Content -Path "C:\path\to\file\policyContent.cedar" | Out-String
New-SSMDocument `
    -Content $cedar `
    -Name "SSM-JustInTimeAccessAutoApprovalPolicy" `
    -DocumentType "AutoApproval"
```

Update just-in-time node access session preferences

With just-in-time node access, you can specify general session and logging preferences in each AWS account and AWS Region in your organization. Alternatively, you can use AWS CloudFormation StackSets to create a session preferences document in multiple accounts and Regions to help you have consistent session preferences. For information about the schema for session preferences documents, see Session document schema.

For logging purposes, we recommend using the streaming option with Amazon CloudWatch Logs. This feature allows you to send a continual stream of session data logs to 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.

Systems Manager doesn't automatically terminate just-in-time node access sessions. As a best practice, specify values for the *maximum session duration* and *idle session timeout* settings. Using these settings helps you to prevent a user from remaining connected to a node longer than the window of time approved in an access request. The following procedure describes how to update session preferences for just-in-time node access.

> **⚠ Important**
>
> You must tag the AWS KMS keys used for Session Manager encryption and RDP recording in just-in-time node access with the tag key `SystemsManagerJustInTimeNodeAccessManaged` and tag value `true`. For information about tagging KMS keys, see Tags in AWS KMS in the *AWS Key Management Service Developer Guide*.

To update session preferences

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Settings** in the navigation pane.
3. Select the **Just-in-time node access** tab.
4. In the **Session preferences** section, select **Edit**.
5. Update your general and logging preferences as needed and select **Save**.

Configure notifications for just-in-time access requests

You can configure Systems Manager to send notifications when a user creates a just-in-time node access request to the email addresses, or chat client, for approvers and the requester. The notification contains the reason for the access request provided by the requester, the AWS

account, AWS Region, status of the request, and ID of the target node. Currently, Systems Manager supports Slack and Microsoft Teams clients through integration with Amazon Q Developer in chat applications. When using notifications through chat clients, access request approvers can interact directly with access requests. This eliminates the need to log in to the console to take action on access requests.

Before you begin

Before you configure a chat client for just-in-time node access notifications, note the following requirement:

- If you're using IAM roles to manage user identities in your account, you must manually associate the email addresses of the approvers or requesters you want to send notifications to with the associated role. Otherwise the intended recipients can't be notified by email.

The following procedures describe how to configure notifications for just-in-time node access requests.

To configure a chat client for just-in-time node access notifications

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Settings** in the navigation pane.
3. Select the **Just-in-time node access** tab.
4. In the **Chat** section, select **Configure new client**.
5. In the **Select client type** dropdown, choose the type of chat client you want to configure and select **Next**.
6. You're prompted to allow Amazon Q Developer in chat applications to access your chat client. Select **Allow**.
7. In the **Configure channel** section, enter the information for your chat client channel and select the types of notifications you want to receive.
8. If you're configuring Slack notifications, invite "@Amazon Q" to every Slack channel that notifications are being configured in.
9. Select **Configure channel**.

Note

To allow approving/rejecting access requests directly from a Slack channel, make sure the IAM role that is configured with the Slack channel has `ssm:SendAutomationSignal` permissions and has a trust policy that includes chatbot:

```
{
            "Effect": "Allow",
            "Principal": {
                "Service": "chatbot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
}
```

To configure email notifications for just-in-time node access notifications

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Settings** in the navigation pane.
3. Select the **Just-in-time node access** tab.
4. In the **Email** section, select **Edit**.
5. Select **Add emails**, choose the **IAM role** you want to manually associate email addresses with.
6. Enter an email address in the **Email address** field. Whenever an access request is created that requires approval from the IAM role you specified, the email addresses you associate with the role are notified.
7. Select **Add email address**.

Recording RDP connections

Just-in-time node access includes the ability to record RDP connections made to your Windows Server nodes. Recording RDP connections require an S3 bucket and an AWS Key Management Service (AWS KMS) customer managed key. The AWS KMS key is used to temporarily encrypt the recording data while it's generated and stored on Systems Manager resources. The customer managed key must be a symmetric key with a key usage of encrypt and decrypt. You can either

use a multi-Region key for your organization, or you must create a customer managed key in each Region where you've enabled just-in-time node access.

If you have enabled KMS encryption on the S3 bucket where you store recordings, you must provide access to the customer managed key used for bucket encryption to the `ssm-guiconnect` service principal. This customer managed key can be a different one than you specify in the recording settings, which must include for which the `kms:CreateGrant` permission is required for establishing connections.

Configuring S3 bucket encryption for RDP recordings

Your connection recordings are stored in the S3 bucket that you specify when you enable RDP recording.

If you use a KMS key as the default encryption mechanism for the S3 bucket (SSE-KMS), you must allow the `ssm-guiconnect` service principal access to `kms:GenerateDataKey` action on the key. We recommend using a customer managed key when using SSE-KMS encryption with S3 bucket. This is because you can update the associated key policy for a customer managed key. You can't update the key policies for AWS managed keys.

> **Important**
>
> You must tag the AWS KMS keys used for Session Manager encryption and RDP recording in just-in-time node access with the tag key `SystemsManagerJustInTimeNodeAccessManaged` and tag value `true`. For information about tagging KMS keys, see Tags in AWS KMS in the *AWS Key Management Service Developer Guide*.

Use the following customer managed key policy to allow the `ssm-guiconnect` service access to the KMS key for S3 storage. For information about updating a customer managed key, see Change a key policy in the *AWS Key Management Service Developer Guide*.

Replace each *`example resource placeholder`* with your own information:

- *`account-id`* represents the ID of the AWS account that initiates the connection.
- *`region`* represents the AWS Region where the S3 bucket is located . (You can use `*` if the bucket will receive recordings from multiple Regions. Example: `s3.*.amazonaws.com`.)

> **Note**
>
> You can use `aws:SourceOrgID` in the policy instead of `aws:SourceAccount` if the account belongs to an organization in AWS Organizations.

```
{
    "Sid": "Allow the GUI Connect service principal to access S3",
    "Effect": "Allow",
    "Principal": {
        "Service": "ssm-guiconnect.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey*"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "account-id"
        },
        "StringLike": {
            "kms:ViaService": "s3.region.amazonaws.com"
        }
    }
}
```

Configuring IAM permissions for recording RDP connections

In addition to the required IAM permissions for just-in-time node access, the user or role you use must be allowed the following permissions based on the task you need to perform.

Permissions for configuring connection recording

To configure RDP connection recording, the following permissions are required:

- `ssm-guiconnect:UpdateConnectionRecordingPreferences`
- `ssm-guiconnect:GetConnectionRecordingPreferences`
- `ssm-guiconnect:DeleteConnectionRecordingPreferences`
- `kms:CreateGrant`

Permissions for initiating connections

To make RDP connections with just-in-time node access, the following permissions are required:

- `ssm-guiconnect:CancelConnection`
- `ssm-guiconnect:GetConnection`
- `ssm-guiconnect:StartConnection`
- `kms:CreateGrant`

Before you begin

To store your connection recordings, you must first create an S3 bucket and add the following bucket policy. Replace each *`example resource placeholder`* with your own information.

(For information about adding a bucket policy, see Adding a bucket policy by using the Amazon S3 console in the *Amazon Simple Storage Service User Guide*.)

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ConnectionRecording",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm-guiconnect.amazonaws.com"
                ]
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ],
            "Condition":{
            "StringEquals":{
                "aws:SourceAccount":"111122223333"
                }
            }
```

```
        }
    ]
}
```

Enabling and configuring RDP connection recording

The following procedure describes how to enable and configure RDP connection recording.

To enable and configure RDP connection recording

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Settings** in the navigation pane.
3. Select the **Just-in-time node access** tab.
4. In the **RDP recording** section, select **Enable RDP recording**.
5. Choose the S3 bucket you want to upload session recordings to.
6. Choose the customer managed key you want to use to temporarily encrypt the recording data while it's generated and stored on Systems Manager resources. (This can be a different customer managed key than you use to encrypt the bucket.)
7. Select **Save**.

RDP connection recording status values

Valid status values for RPD connection recordings include the following:

- `Recording` - The connection is in the process of being recorded
- `Processing` - The video is being processed after the connection is terminated.
- `Finished` - Successful terminal state:The connection recording video processed successfully and uploaded to the specified bucket.
- `Failed` - Failed terminal state. The connection wasn't recorded successfully.
- `ProcessingError` - One or more intermediate failures/errors occurred during video processing. Potential causes include service dependency failures or missing permissions due to a misconfiguration on the S3 bucket specified for storing recordings. The service continues to attempt processing when the recording is in this state.

> **Note**
>
> `ProcessingError` can be the result of the `ssm-guiconnect` service principal not having permission to upload objects to the S3 bucket after the connection has been established. Another potential cause is missing KMS permissions on the KMS key used for S3 bucket encryption.

Modifying targets

When you set up just-in-time node access, you choose the *targets* where you want to set up just-in-time node access. Targets consist of AWS Organizations organizational units (OUs) and AWS Regions. By default, the same targets you chose when setting up the unified Systems Manager console are selected for just-in-time node access. You can choose to set up just-in-time node access for all of the same targets, or a subset of the targets you specified when setting up the unified Systems Manager console. Adding new targets that weren't selected when you set up the unified Systems Manager console isn't supported. You can change the targets you selected after setting up just-in-time node access.

The following procedure describes how to modify the targets for just-in-time node access.

To modify targets

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Settings** in the navigation pane.
3. Select the **Just-in-time node access** tab.
4. In the **Targets** section, select **Edit**.
5. Select the **Organizational units** and **Regions** where you want to use just-in-time node access.
6. Select **Save**.

Changing identity providers

By default, just-in-time node access uses IAM for an identity provider. After enabling just-in-time node access, customers using the unified console with an organization can modify this setting to use IAM Identity Center. Just-in-time node access doesn't support IAM Identity Center as an identity provider when set up for a single account and Region.

The following procedure describes how to modify the identity provider for just-in-time node access.

To modify identity providers

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Settings** in the navigation pane.
3. Select the **Just-in-time node access** tab.
4. In the **User identity** section, select **Edit**.
5. Choose **AWS IAM Identity Center**.
6. Select **Save**.

Start a just-in-time node access session

After enabling and setting up just-in-time node access, and configuring session and notification preferences, users are ready to start just-in-time node access sessions. You can start sessions using just-in-time node access from the Systems Manager console or from the AWS Command Line Interface using the Session Manager plugin. Just-in-time node access sessions can be started on nodes in the same account and Region. The following procedures describe how to start sessions with just-in-time node access.

> **Note**
>
> If your users previously used Session Manager to connect to nodes, you must remove the Session Manager permissions, for example `ssm:StartSession`, from their IAM policies to start sessions using just-in-time node access. Otherwise, when connecting to nodes they'll continue to use Session Manager.

To start a session with just-in-time node access using the console

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Explore nodes** in the navigation pane.
3. Select the node you want to connect to.

4. In the **Actions** dropdown, select **Connect**.

If your organization's approval policies don't allow you to automatically connect to the node, you're prompted to submit an access request. After you fill out the information requested and submit the acccess request, you'll be able to start sessions to the node once all of the required approvals are received.

To start a session with just-in-time node access using the AWS CLI

1. Run the following command to start the access request workflow, making sure to replace the *placeholder values* with your own information.

   ```
   aws ssm start-access-request \
       --targets  Key=InstanceIds,Values=i-02573cafcfEXAMPLE
       --reason "Troubleshooting networking performance issue"
   ```

   Depending on your organization's approval policies, you'll either be automatically connected to the node or the manual approval process is started. For requests that require manual approvals, note the ID of the access request that is returned in the response.
2. Wait for all of the required approvals to be provided.
3. After all required approvals have been provided, run the following command to get an access token containing temporary credentials. Replace the *placeholder values* with your own information.

   ```
   aws ssm get-access-token \
       --access-request-id oi-12345abcdef
   ```

   Note the access token returned in the response.
4. Run the following command to use the temporary credential in the AWS CLI, making sure to replace the *placeholder values* with your own information.

   ```
   export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>
   ```
5. Run the following command to start a session to the node, making sure to replace the *placeholder values* with your own information.

   ```
   aws ssm start-session \
   ```

```
    --target i-02573cafcfEXAMPLE
```

Managing just-in-time access requests

For increased visibility across your organization, Systems Manager replicates access requests to the delegated administrator account for your organization. To help you meet compliance requirements, Systems Manager retains all access requests for 1 year. The following topics describe how to manage just-in-time node access requests. This information is intended for access request approvers. Before you begin, we commend reviewing your IAM policies and making sure you have the required permissions for administering just-in-time node access. For more information see, Setting up just-in-time access with Systems Manager.

Topics

- Approving and denying just-in-time node access requests

Approving and denying just-in-time node access requests

Access request approvers can approve or deny just-in-time node access requests from the unified Systems Manager console or using your preferred command line tool. This information is intended for access request approvers. If you don't have the permissions required to approve or reject access requests, contact your administrator. The following procedures describe how to approve or deny just-in-time node access requests.

To approve or deny just-in-time node access requests using the console

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Select **Manage node access** in the navigation pane.
3. Select the **Access requests** tab.
4. Select the **Requests for me** toggle.
5. Select the checkbox next to the access request you want to approve or deny.
6. Select **Approve** or **Reject**.

After approving an access request you can revoke your approval at any time by selecting **Revoke**.

To approve or deny just-in-time node access requests using the command line

1. Note the access request ID from the notification. For example, *oi-12345abcdef*.
2. Run the following command to return details about the access request approval workflow, making sure to replace the *placeholder values* with your own information.

   ```
   aws ssm get-ops-item \
       --ops-item-id oi-12345abcdef
   ```

   Note the `automationExecutionId` value in the `/aws/accessrequest` field for the `OperationalData`. For example, *9231944f-61c6-40be-8bce-8ee2bEXAMPLE*.
3. Run the following command to approve or deny the access request. Use the `Approve` signal type to approve the request, and `Deny` to deny the request. Make sure to replace the *placeholder values* with your own information.

   ```
   aws ssm send-automation-signal \
       --automation-execution-id 9231944f-61c6-40be-8bce-8ee2bEXAMPLE \
       --signal-type "Approve"
   ```

Moving to just-in-time node access from Session Manager

When you enable just-in-time node access, Systems Manager doesn't make any changes to your existing resources for Session Manager. This ensures there's no disruption to your existing environment and users can continue to start sessions while you create and validate approval policies. Once you're ready to test your approval policies, you must modify your existing IAM policies to complete the transition to just-in-time node access. This includes adding the required permissions for just-in-time node access to identities, and removing permission for the `StartSession` API operation for Session Manager. We recommend testing approval policies with a subset of identities and nodes in an AWS account and AWS Region.

For more information about the permissions required for just-in-time node access, see Setting up just-in-time access with Systems Manager.

For more information about modifying and identity's IAM permissions, see Adding and removing IAM identity permissions in the *IAM User Guide*.

The following describes a detailed method of how you can move to just-in-time node access from Session Manager.

Moving from Session Manager to just-in-time node access requires careful planning and testing to ensure a smooth transition without disrupting your operations. The following sections describe how you can complete this process.

Prerequisites

Before you begin, ensure that you have completed the following tasks:

- Set up the Systems Manager unified console.
- Verified you have permissions to modify IAM policies in your account.
- Identified all IAM policies and roles that currently grant Session Manager permissions.
- Documented your current Session Manager configuration, including session preferences and logging settings.

Assessment

Assess your current environment and outline desired approval behaviors by completing the following tasks:

1. **Inventory your nodes** - Identify all nodes that users currently access through Session Manager.
2. **Identify user access patterns** - Document which users or roles need access to which nodes, and under what circumstances.
3. **Map approval workflows** - Determine who should approve access requests for different types of nodes.
4. **Review tagging strategy** - Ensure your nodes are properly tagged to support your planned approval policies.
5. **Audit existing IAM policies** - Identify all policies that include Session Manager permissions.

Planning

Phased strategy

When moving from Session Manager to just-in-time node access, we recommend using a phased approach like the following:

1. **Phase 1: Setup and configuration** - Enable just-in-time node access without modifying existing Session Manager permissions.

2. **Phase 2: Policy development** - Create and test approval policies for your nodes.
3. **Phase 3: Pilot migration** - Modify a small group of non-critical nodes and users or roles from Session Manager to just-in-time node access.
4. **Phase 4: Full migration** - Gradually migrate all remaining nodes and users or roles.

Timeline considerations

Consider the following factors when creating your timeline to move from Session Manager to just-in-time node access:

- Allow time for user training and adjustment to the new approval workflow.
- Schedule migrations during periods of lower operational activity.
- Include buffer time for troubleshooting and adjustments.
- Plan for a period of parallel operation where both systems are available.

Implementation steps

Phase 1: Setup and configuration

1. Enable just-in-time node access in the Systems Manager console. For detailed steps, see Setting up just-in-time access with Systems Manager.
2. Configure session preferences for just-in-time node access to match your current Session Manager settings. For more information, see Update just-in-time node access session preferences.
3. Set up notification preferences for access requests. For more information, see Configure notifications for just-in-time access requests.
4. If you use RDP connections to Windows Server nodes, configure RDP recording. For more information, see Recording RDP connections.

Phase 2: Policy development

1. Create IAM policies for just-in-time node access administrators and users.
2. Develop approval policies based on your security requirements and use case.
3. Test your policies in a non-production environment to ensure they work as expected.

Phase 3: Pilot migration

1. Select a small group of users and non-critical nodes for the pilot.
2. Create new IAM policies for pilot users that include just-in-time node access permissions.
3. Remove Session Manager permissions (`ssm:StartSession`) from the pilot users' IAM policies.
4. Train pilot users on the new access request workflow.
5. Monitor the pilot for issues and gather feedback.
6. Adjust policies and procedures based on pilot results.

Example IAM policy modification for pilot users

Original policy with Session Manager permissions:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:StartSession",
        "ssm:ResumeSession",
        "ssm:TerminateSession"
      ],
      "Resource": "*"
    }
  ]
}
```

Modified policy for just-in-time node access:

JSON

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

```
      "Effect": "Allow",
      "Action": [
        "ssm:StartAccessRequest",
        "ssm:GetAccessToken",
        "ssm:ResumeSession",
        "ssm:TerminateSession"
      ],
      "Resource": "*"
    }
  ]
}
```

Phase 4: Full migration

Develop a schedule for migrating remaining users and nodes in batches.

Testing methodology

Throughout the migration process, conduct the following tests:

- **Policy validation** - Verify that approval policies correctly apply to the intended nodes and users.
- **Access request workflow** - Test the complete workflow from access request to session establishment for both auto-approval and manual approval scenarios.
- **Notifications** - Verify that approvers receive notifications through configured channels (email, Slack, Microsoft Teams).
- **Logging and monitoring** - Verify that session logs and access requests are properly captured and stored.

Best practices for a successful migration

- **Communicate early and often** - Inform users about the migration timeline and benefits of just-in-time node access.
- **Start with non-critical systems** - Begin migration with development or test environments before moving to production.
- **Document everything** - Maintain detailed records of your approval policies, IAM policy changes, and configuration settings.
- **Monitor and adjust** - Continuously monitor access requests and approval workflows, adjusting policies as needed.

- **Establish governance** - Create a process for regularly reviewing and updating approval policies as your environment changes.

Disabling just-in-time access with Systems Manager

The following procedure describes how to disable just-in-time node access. After disabling just-in-time node access, users in your organization might be unable to connect to your nodes unless you've already implemented other connection methods.

To disable just-in-time node access

1. Log in to the Systems Manager delegated administrator account for your organization.
2. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
3. Select **Settings** in the navigation pane.
4. In the **Just-in-time node access** tab, select **Disable**.

Just-in-time node access frequently asked questions

How do I move from Session Manager to just-in-time node access?

After setting up the unified console and enabling just-in-time node access, you must modify your existing IAM policies to complete the move to just-in-time node access. This includes adding the required permissions for just-in-time node access and removing permission for the `StartSession` API operation for Session Manager. For more information about IAM policies for just-in-time node access see Setting up just-in-time access with Systems Manager.

Do I have to set up the unified console to use just-in-time node access?

Yes, setting up the unified console is a prerequisite for just-in-time node access. However, after you set up the unified console and enable just-in-time node access, there are several methods for connecting to your nodes. For example, you can start just-in-time node access sessions from the Amazon EC2 console and the AWS CLI. For more information about setting up the unified console, see Setting up Systems Manager unified console for an organization.

Is there cost associated with just-in-time node access?

Systems Manager provides a 30 day free trial for just-in-time node access. After the trial, just-in-time node access incurs costs. For more information, see AWS Systems Manager Pricing.

What is the precedence for just-in-time node access approval policies?

Approval policies are evaluated in the following order:

1. Deny-access
2. Auto-approval
3. Manual

How are manual approval policies evaluated?

Just-in-time node access always favors the more specific policy for a node. Manual approval policies are evaluated in the following order:

1. Tag specific target
2. All nodes target

What happens if there isn't an approval policy that applies to a node?

To connect to a node using just-in-time node access, an approval policy must apply to the node. If there are no approval policies that apply to a node, users are unable to request access to the node.

Can multiple approval policies target a tag?

A tag can only be targeted once in your approval policies.

What happens if multiple manual approval policies apply to a node as a result of overlapping tags?

When multiple manual approval policies apply to a node, this results in a conflict and users are unable to request access to the node. Keep this in mind when creating your manual approval policies since some instances might have multiple tags depending on your case.

Can I use just-in-time node access to request access and start sessions on nodes across accounts and Regions?

Just-in-time node access supports requesting access to and starting sessions on nodes in the same account and Region as the requester.

Can I use just-in-time node access to request access and start sessions on nodes registered with a hybrid activation?

Yes, just-in-time node access supports requesting access to and starting sessions on nodes registered with a hybrid activation. The node must be registered in the same account and Region as the requester.

Diagnosing and remediating

Using the unified Systems Manager console, you can identify problems across your fleet in a single diagnosis operation. For organizations, you can then attempt remediation on all or only select targets using a single Automation operation. For an organization, as a delegated account administrator, you can select targets across all accounts and Regions. If you are working in a single account, you can select targets in a single Region at a time.

Systems Manager can diagnose and help you remediate several types of deployment failures, as well as drifted configurations. Systems Manager can also identify Amazon Elastic Compute Cloud (Amazon EC2) instances in your account or organization that Systems Manager isn't able to treat as a *managed node*. The EC2 instance diagnosis process can identify issues related to misconfigurations for a virtual private cloud (VPC), in a Domain Name Service (DNS) setting, or in an Amazon Elastic Compute Cloud (Amazon EC2) security group.

> **Note**
>
> Systems Manager supports both EC2 instances and other machine types in a hybrid and multicloud environment as *managed nodes*. To be a managed node, AWS Systems Manager Agent (SSM Agent) must be installed on the machine, and Systems Manager must have permission to perform actions on the machine.
> For EC2 instances, this permission can be provided at the account level using an AWS Identity and Access Management (IAM) role, or at the instance level using an instance profile. For more information, see Configure instance permissions required for Systems Manager.

For non-EC2 machines, this permission is provided using an IAM service role. For more information, see Create the IAM service role required for Systems Manager in hybrid and multicloud environments.

Before you begin

In order to use the **Diagnose and remediate** feature to detect unmanaged EC2 instances, you must first onboard your organization or account to the unified Systems Manager console. During this process, you must choose the option to create IAM roles and managed policies required for these operations. For more information, see Setting up Systems Manager unified console for an organization.

Use the following topics to help you identify and fix certain common types of failed deployments, drifted configurations, and unmanaged EC2 instances.

Topics

- Diagnosing and remediating failed deployments
- Diagnosing and remediating drifted configurations
- Diagnosing and remediating unmanaged Amazon EC2 instances in Systems Manager
- Remediation impact types of runbook actions
- Viewing execution progress and history for remediations in Systems Manager

Diagnosing and remediating failed deployments

Systems Manager can diagnose and then help you remediate the following types of failed deployments:

- Core setup for organization member accounts
- Core setup for delegated administrator account
- Core setup for your account

Use the following procedure to attempt to remediate these types of issues.

To diagnose and remediate failed deployments

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Diagnose and remediate**.
3. Choose the **Deployment issues** tab.
4. In the **Failed deployments** section, review the list of findings for failed deployments.
5. In the **Setup step** column, choose the name of a finding to review additional details about the issue. For example: **Core setup for organization member accounts**.
6. In the detail page for that failed deployment, you can view a list of accounts and how many Regions in each have experienced deployment failures.
7. Select an account ID to view information about the reason for failures in that account.
8. In the **Failed Regions** area, examine the information provided for **Status reason**. This information can indicate a reason for the failed deployment, which might provide insight into configuration changes that need to be made.
9. If you want to retry the deployment without making configuration changes, choose **Redeploy**.

Diagnosing and remediating drifted configurations

Systems Manager can diagnose and then help you remediate the following types of drifted configurations:

- Core setup for organization member accounts
- Core setup for delegated administrator account
- Core setup for your account

Use the following procedure to attempt to remediate these types of drifted configurations.

To diagnose and remediate drifted configurations

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Diagnose and remediate**.
3. Choose the **Deployment issues** tab.
4. In the **Drifted deployments** section, review the list of finding for failed deployments.

-or-

To run a new diagnosis, choose **Detect drift**.

5. In the **Setup step** column, choose the name of a finding to review additional details about the issue. For example: **Core setup for organization member accounts**.
6. In the detail page for that failed deployment, you can view a list of accounts and how many Regions in each have experienced configuration drifts.
7. Select an account ID to view information about the reason for configuration drifts in that account.
8. In the **Drifted resources** area, the **Resource** column reports names of resources that have experienced drift. The **Drift type** column reports whether the resource was modified or deleted..
9. To redeploy the intended configuration, choose **Redeploy**.

Diagnosing and remediating unmanaged Amazon EC2 instances in Systems Manager

To help you manage your Amazon Elastic Compute Cloud (Amazon EC2) instances with Systems Manager, you can use the unified Systems Manager console to do the following:

1. Run a manual or scheduled diagnosis process to identify which EC2 instances in your account or organization aren't currently managed by Systems Manager.
2. Identify network or other issues that are preventing Systems Manager from taking over management of the instances.
3. Run an Automation execution to automatically remediate the problem, or access information to help you manually address the issue.

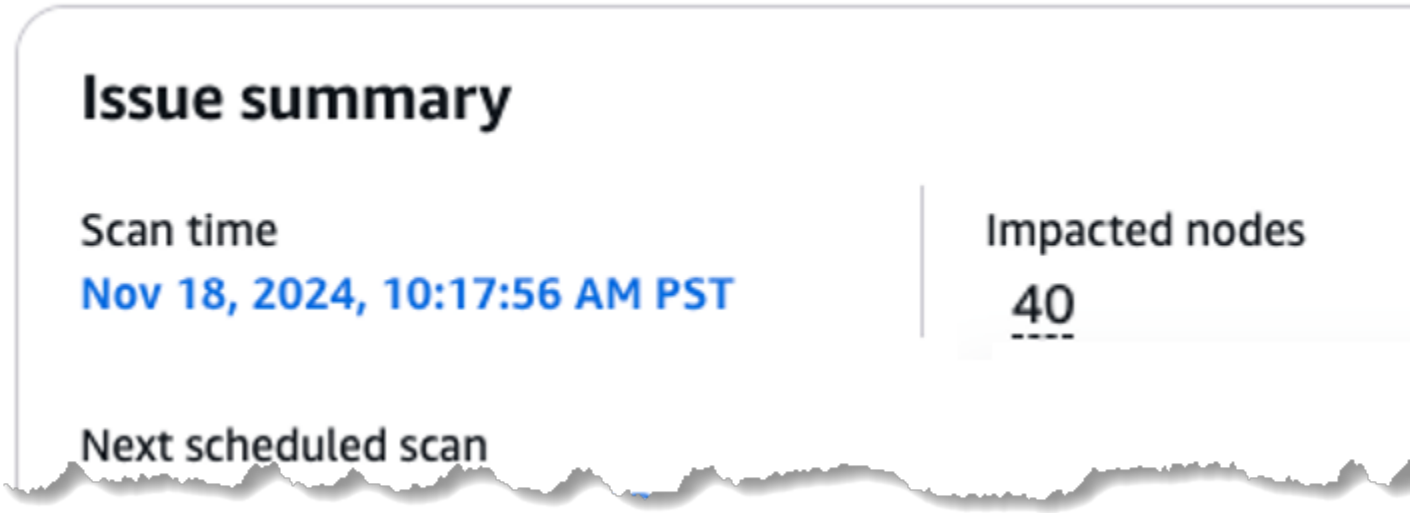
Use the information in the following topics to help you diagnose and remediate issues that are preventing Systems Manager from managing your EC2 instances.

How Systems Manager counts impacted nodes for the 'Unmanaged EC2 instance issues' list

The number of nodes reported as unmanaged on the **Unmanaged EC2 instances issues** tab represents to the total number of instances with any of the follow status values at the diagnosis scan time:

- `Running`
- `Stopped`
- `Stopping`

This number is reported as **Impacted nodes** in the **Issue summary** area. In the following image, this number of impacted nodes not currently managed by Systems Manager is `40`.



Unlike the report of unmanaged EC2 instances on the **Review node insights** page, this count of EC2 instances is not dynamic. It represents findings made during the last reported diagnostic scan, shown as the **Scan time** value. We therefore recommend running a diagnostic scan for unmanaged EC2 instances on a regular schedule to keep this reported number of impacted nodes up to date.

For information about unmanaged instance counts on the **Review node insights** page, see What is an unmanaged instance? in the topic Reviewing node insights.

Topics

- Categories of diagnosable unmanaged EC2 instance issues
- Running a diagnosis and optional remediation for unmanaged EC2 instances

- Scheduling a recurring scan for unmanaged EC2 instances

Categories of diagnosable unmanaged EC2 instance issues

This topic lists the major categories of EC2 management issues, and the specific issues in each category, that Systems Manager can help you diagnose and remediate. Note that for some of the issues, Systems Manager can identify the issue, but not provide automatic remediation. In those cases, the Systems Manager console directs you to information to help you manually resolve an issue.

The diagnosis process examines each group of EC2 instances at once according to the virtual private cloud (VPC) they belong to.

Issue types

- Problem category: Security group configuration and HTTPS communications
- Problem category: DNS or DNS host name configuration
- Problem category: VPC endpoint configuration

Problem category: Security group configuration and HTTPS communications

A diagnosis operation might find that SSM Agent isn't able to communicate with the Systems Manager service over HTTPS. In those cases, you can choose to execute an Automation runbook that attempts to update security groups that are attached to the instances.

> **Note**
>
> Occasionally, Systems Manager might not be able to automatically remediate these issues, but you can manually edit the affected security groups.

Supported issue types

- **Instance security group**: Outbound traffic is not allowed on port 443
- **`ssm` VPC endpoint's security group**: Inbound traffic is not allowed on port 443
- **`ssmmessages` VPC endpoint's security group**: Inbound traffic not allowed on port 443
- **`ec2messages` VPC endpoint's security group**: Inbound traffic not allowed on port 443

For more information, see Verify ingress rules on endpoint security groups in the topic Troubleshooting SSM Agent.

Problem category: DNS or DNS host name configuration

A diagnosis operation might find that Doman Name System (DNS) or DNS host names aren't properly configured for the VPC. In those cases, you can choose to execute an Automation runbook that attempts to enable the `enableDnsSupport` and `enableDnsHostnames` attributes of the affected VPC.

Supported issue types

- DNS support is disabled in a VPC.
- A DNS hostname is disabled in a VPC.

For more information, see Verify your VPC DNS-related attributes in the topic Troubleshooting SSM Agent.

Problem category: VPC endpoint configuration

A diagnosis operation might find that VPC endpoints aren't properly configured for the VPC.

If VPC endpoints required by SSM Agent don't exist, Systems Manager attempts to execute an Automation runbook to create the VPC endpoints and associates them with one subnet in each relevant regional availability zone (AZ). If VPC the required endpoints exist but aren't associated with a subnet in which the issue is found, the runbook associates the VPC endpoints to the affected subnet.

> **Note**
>
> Systems Manager doesn't support remediating all misconfigured VPC endpoint issues. In those cases, Systems Manager directs you to manual remedy instructions instead of running an Automation runbook.

Supported issue types

- No `ssm.region.amazonaws.com` endpoint for PrivateLink was found.
- No `ssmmessages.region.amazonaws.com` endpoint for PrivateLink was found.

- No `ec2messages.region.amazonaws.com` endpoint for PrivateLink was found.

Diagnosable issue types

Systems Manager can diagnose the following issue types, but currently no runbook is available for remediating their issues. You can edit your configuration manually for these issues.

- An instance's subnet is not attached to an `ssm.region.amazonaws.com` endpoint.
- An instance's subnet is not attached to an `ssmmessages.region.amazonaws.com` endpoint.
- An instance's subnet not attached to an `ec2messages.region.amazonaws.com` endpoint.

For more information, see Verify your VPC configuration in the topic Troubleshooting SSM Agent.

Running a diagnosis and optional remediation for unmanaged EC2 instances

Use the following procedure to diagnose the network-related and VPC-related issues that might be preventing Systems Manager from managing your EC2 instances.

The diagnosis operation can detect and group together issues of the following types:

- **Network configurations issues** – Types of networking issues that might be preventing EC2 instances from communicating with the Systems Manager service in the cloud. Remediation operations might be available for these issues. For more information about the network configuration issues, see Categories of diagnosable unmanaged EC2 instance issues.
- **Unidentified issues** – A list of findings for cases where the diagnostic operation was unable to determine why EC2 instances are not able to communicate with the Systems Manager service in the cloud.

To run a diagnosis and remediation for unmanaged EC2 instances

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Diagnose and remediate**.
3. Choose the **Unmanaged EC2 instances issue** tab.
4. In the **Issue summary** section, choose **Run new diagnosis**.

   -or-

If this is your first time to diagnose unmanaged EC2 issues, in the **Diagnose unmanaged EC2 instances** section, choose **Execute**.

> **Tip**
>
> While the diagnosis is running, choose **View progress** or **View executions** to monitor the current state of the execution. For more information, see Viewing execution progress and history for remediations in Systems Manager.

5. After the diagnosis completes, do the following:

   - For any issues reported in the **Unidentified issues** section, choose the **Learn more** link for information about resolving the problem.
   - For issues reported in the **Network configurations issues** section, continue with the next step.

6. In the list of finding types, in the **Recommendations** column, for a particular issue, choose the link, such as **2 recommendations**.
7. In the **Recommendations** pane that opens, choose from the available mitigations:

   - **Learn more** – Open a topic with information about how to resolve an issue manually.
   - **View runbook** – Open a pane with information about the Automation runbook you can execute to resolve the issue with your EC2 instances, as well as options for generating a *preview* of the actions that runbook would take. Continue with the next step.

8. In the runbook pane, do the following:

   a. For **Document description**, review the content, which provides an overview of the actions the runbook can take to remediate your unmanaged EC2 instance issues. Choose **View steps** to preview the individual actions the runbook would take.

   b. For **Targets**, do the following:

      - If you are managing remediations for an organization, for **Accounts**, specify whether this runbook would target all accounts, or only a subset of accounts you choose.
      - For **Regions**, specify whether this runbook would target all AWS Regions in your account or organization, or only a subset of Regions you choose.

   c. For **Runbook preview**, carefully review the information. This information explains what the scope and impact would be if you choose to execute the runbook.

> **Note**
>
> Choosing to execute the runbook would incur charges. Review the preview information carefully before deciding whether to proceed.

The **Runbook preview** content provides the following information:

- How many Regions the runbook operation would occur in.
- (Organizations only) How many organizational units (OUs) the operation would run in.
- The types of actions that would be taken, and how many of each.

  Action types include the following:

  - **Mutating**: The runbook step would make changes to the targets through actions that create, modify, or delete resources.
  - **Non-mutating**: The runbook step would retrieve data about resources but not make changes to them. This category generally includes `Describe*`, `List*`, `Get*`, and similar read-only API actions.
  - **Undetermined**: An undetermined step invokes executions performed by another orchestration service like AWS Lambda, AWS Step Functions, or AWS Systems Manager Run Command. An undetermined step might also call a third-party API. Systems Manager Automation doesn't know the outcome of the orchestration processes or third-party API executions, so the results of the steps are undetermined.

d. At this point, you can choose one of the following actions:

- Stop and do not execute the runbook.
- Choose **Execute** to run the runbook with the options you have already selected.

If you choose to run the operation, choose **View progress** or **View executions** to monitor the current state of the execution. For more information, see Viewing execution progress and history for remediations in Systems Manager.

Scheduling a recurring scan for unmanaged EC2 instances

You can run an on-demand scan for Amazon EC2 instances in your account or organization that Systems Manager isn't able to manage due to various configuration issues. You can also schedule this scan to occur automatically on a regular schedule.

To schedule a recurring scan for unmanaged EC2 instances

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Diagnose and remediate**.
3. Choose the **Unmanaged EC2 instances issue** tab.
4. In the **Diagnose unmanaged EC2 instances** section, turn on **Schedule recurring diagnosis**.
5. For **Diagnostic frequency**, select whether to run the diagnosis once a day or once a week.
6. (Optional) For **Start time**, enter a time, in 24-hour format, for the diagnosis to begin. For example, for 8:15 PM, enter **`20:15`**.

   The time you enter is for your current local time zone.

   If you don't specify a time, the diagnostic scan runs immediately. Systems Manager also schedules the scan to run in the future at the current time. If you specify a time, Systems Manager waits to run the diagnostic scan at the specified time.
7. Choose **Execute**. The diagnosis runs immediately, but will also run on the schedule you have specified.

Remediation impact types of runbook actions

Systems Manager can run diagnosis operations that discover certain types of failed deployments and drifted configurations, as well as certain types of configuration issues that are preventing Systems Manager from managing EC2 instances. The results of the diagnosis might include recommendations for Automation runbooks that you can execute to attempt to remedy a problem. For more information about these diagnosis operations, see the following topics:

- Diagnosing and remediating failed deployments
- Diagnosing and remediating drifted configurations
- Diagnosing and remediating unmanaged Amazon EC2 instances in Systems Manager

When Systems Manager identifies an issue that might be fixed by running an Automation runbook on the affected resources, it provides you with an *execution preview*. The execution preview provides information about the *types* of changes the runbook execution would make to your targets. This information includes how many of each of three types of changes the diagnosis identified.

These change types are as follows:

- `Mutating`: A runbook step would make changes to the targets through actions that create, modify, or delete resources.
- `Non-Mutating`: A runbook step would retrieve data about resources but not make changes to them. This category generally includes `Describe*`, `List*`, `Get*`, and similar read-only API actions.
- `Undetermined`: An undetermined step invokes executions performed by another orchestration service like AWS Lambda, AWS Step Functions, or Run Command, a tool in AWS Systems Manager. An undetermined step might also call a third-party API or run a Python or PowerShell script. Systems Manager Automation can't detect what the outcome would be of the orchestration processes or third-party API executions, and therefore can't evaluate them. The results of those steps would have to be manually reviewed to determine their impact.

  See the following table for information about the impact type of supported Automation actions.

Impact types of supported remediation actions

The table presents the impact type—Mutating, Non-mutating, and Undetermined—of various actions that can be included in a remediation runbook.

Action[1]	Impact type
aws:approve	Non-mutating
aws:assertAwsResourceProperty	Non-mutating
aws:branch	Non-mutating
aws:changeInstanceState	Mutating
aws:copyImage	Mutating

Action[1]	Impact type
aws:createImage	Mutating
aws:createStack	Mutating
aws:createTags	Mutating
aws:deleteImage	Mutating
aws:deleteStack	Mutating
aws:executeAutomation	Undetermined
aws:executeAwsApi	Undetermined
aws:executeScript	Undetermined
aws:executeStateMachine	Undetermined
aws:invokeLambdaFunction	Undetermined
aws:invokeWebhook	Undetermined
aws:loop	Varies. Depends on the actions in the loop.
aws:pause	Non-mutating
aws:runCommand	Undetermined
aws:runInstances	Mutating
aws:sleep	Non-mutating
aws:updateVariable	Mutating
aws:waitForAwsResourceProperty	Non-mutating

[1] For more information about Automation actions, see Systems Manager Automation actions reference.

Viewing execution progress and history for remediations in Systems Manager

You can view a list of all in-progress and completed remediation operations made using the **Diagnose and remediate** feature in Systems Manager.

Data in the execution history list reports the following types of information:

- The type of execution, `Diagnosis` or `Remediation`.
- The execution status, such as `Success` or `Failed`.
- The times that the execution started and ended.

To view execution progress and history for remediations

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Diagnose and remediate**.
3. Choose **View executions**.

   > **Tip**
   >
   > When an execution is running, you can also choose **View progress** to open the **Execution history** page.

4. (Optional) In the search
   ()
   box, enter a phrase to help narrow down the execution list, such as **EC2** or **VPC**.
5. (Optional) To view additional details about an execution, in the **Execution name** column, choose an operation name, such as **AWS-DiagnoseUnmanagedEC2NetworkIssues**.

   In the details pane, you can review information about all the steps attempted during the operation, and about all the inputs and outputs for the execution.

Adjusting Systems Manager settings

The options on the **Settings** pages enable and configure features in the Systems Manager unified console. The options displayed depend on the account you are logged into and whether or not you have already set up Systems Manager.

> **Note**
>
> The options on the **Settings** page don't affect Systems Manager tools (formerly called capabilities).

Account setup settings

If Systems Manager is enabled, and if you are logged into an account that is not a member of Organizations or if the delegated administrator has not added your Organizations account to Systems Manager, the **Account setup** page shows the option to **Disable Systems Manager**. Disabling Systems Manager means Systems Manager doesn't display the unified console. All Systems Manager tools still function.

Organizational setup settings

On the **Organizational setup** tab, the **Home Region** section displays the AWS Region chosen as the home Region during setup. In multi-account and multi-Region environments that use AWS Organizations, Systems Manager automatically aggregates node data from all accounts and Regions to the home Region. Aggregating data in this way enables you to view node data across accounts and Regions in a single location.

> **Note**
>
> If you want to change the home Region, you must disable Systems Manager and enable it again. To disable Systems Manager, choose **Disable**.

The **Organizational setup** section displays the AWS organizational units and AWS Regions chosen during setup. To change which organizational units and Regions display node data in Systems Manager, choose **Edit**. For more information about setting up Systems Manager for Organizations, see Setting up AWS Systems Manager.

Feature configurations

The **Feature configurations** section allows you to enable and configure key Systems Manager capabilities that enhance node management across your organization. These features work together to provide automated management, compliance monitoring, and maintenance of your managed nodes.

You can configure these features during initial Systems Manager setup or modify them later through the Settings page. Each feature can be enabled or disabled independently based on your organization's requirements.

Default Host Management Configuration

Default Host Management Configuration (DHMC) automatically configures Amazon Elastic Compute Cloud (Amazon EC2) instances in your organization to be managed by Systems Manager. When enabled, DHMC ensures that new and existing EC2 instances have the necessary AWS Identity and Access Management (IAM) permissions and configurations to communicate with Systems Manager services.

DHMC provides the following benefits:

- **Automatic IAM role assignment** - Ensures EC2 instances have the required IAM roles and policies to function as managed nodes
- **Drift remediation** - Automatically corrects configuration drift when instances lose their managed node status
- **Simplified onboarding** - Reduces manual configuration steps for new instances
- **Consistent configuration** - Maintains uniform settings across your EC2 fleet

Configuring drift remediation frequency

Drift remediation automatically detects and corrects when EC2 instances lose their managed node configuration. You can configure how frequently Systems Manager checks for and remediates configuration drift.

To configure Default Host Management Configuration

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

3. In the **Feature configurations** section, locate **Default Host Management Configuration**.
4. To enable DHMC, turn on the toggle switch.
5. For **Drift remediation frequency**, choose how often you want Systems Manager to check for and remediate configuration drift:
   - **Daily** - Checks and remediates drift once per day
   - **Weekly** - Checks and remediates drift once per week
   - **Monthly** - Checks and remediates drift once per month
6. Choose **Save**.

> **Note**
>
> When you enable DHMC, Systems Manager creates the necessary IAM roles and policies in your account. These roles allow EC2 instances to communicate with Systems Manager services. For more information about the IAM roles created by DHMC, see Managing EC2 instances with Systems Manager.

Inventory metadata collection

Inventory metadata collection automatically gathers detailed information about your managed nodes, including installed applications, network configurations, system updates, and other system metadata. This information helps you maintain compliance, perform security analysis, and understand your infrastructure composition.

Inventory collection provides the following benefits:

- **Compliance monitoring** - Track installed software and configurations for compliance reporting
- **Security analysis** - Identify outdated software and potential security vulnerabilities
- **Asset management** - Maintain an up-to-date inventory of your infrastructure
- **Query capabilities** - Use collected data with Amazon Q Developer for natural language queries

Types of inventory data collected

When inventory metadata collection is enabled, Systems Manager collects the following types of information from your managed nodes:

- **Applications** - Installed software packages and applications
- **Network configurations** - Network interfaces, IP addresses, and network settings
- **System updates** - Installed patches and available updates
- **System properties** - Hardware specifications, operating system details, and system configurations
- **Services** - Running services and their configurations

Configuring inventory collection frequency

You can configure how frequently Systems Manager collects inventory metadata from your managed nodes. More frequent collection provides more up-to-date information but may increase AWS service usage.

To configure inventory metadata collection

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Settings**.
3. In the **Feature configurations** section, locate **Inventory metadata collection**.
4. To enable inventory collection, turn on the toggle switch.
5. For **Collection frequency**, choose how often you want Systems Manager to collect inventory data:
   - **Daily** - Collects inventory data once per day
   - **Weekly** - Collects inventory data once per week
   - **Monthly** - Collects inventory data once per month
6. Choose **Save**.

> **Important**
>
> Inventory collection requires managed nodes to have the necessary permissions to gather system information. Ensure your managed nodes have the appropriate IAM roles and policies. For more information about required permissions, see AWS Systems Manager Inventory.

SSM Agent updates

Automatic SSM Agent updates ensure that your managed nodes are running the latest version of the SSM Agent. Keeping the agent up-to-date provides access to the latest features, security improvements, and bug fixes.

SSM Agent automatic updates provide the following benefits:

- **Latest features** - Access to new Systems Manager capabilities and improvements
- **Security updates** - Automatic installation of security patches and fixes
- **Improved reliability** - Bug fixes and stability improvements
- **Reduced maintenance** - Eliminates the need for manual agent updates

Configuring automatic agent updates

You can configure how frequently Systems Manager checks for and installs SSM Agent updates on your managed nodes. Regular updates help ensure optimal performance and security.

To configure SSM Agent updates

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Settings**.
3. In the **Feature configurations** section, locate **SSM Agent updates**.
4. To enable automatic updates, turn on the toggle switch.
5. For **Update frequency**, choose how often you want Systems Manager to check for and install agent updates:
   - **Daily** - Checks for updates once per day
   - **Weekly** - Checks for updates once per week
   - **Monthly** - Checks for updates once per month
6. Choose **Save**.

Diagnose and remediate settings

The **Diagnose and remediate** settings determine whether or not Systems Manager automatically scans your nodes to ensure they can communicate with Systems Manager. If enabled, the feature

runs automatically according to a schedule you define. The feature identifies which nodes can't connect to Systems Manager and why. This feature also provides recommended runbooks for remediating networking issues and other problems preventing nodes from being configured as managed nodes.

Scheduling a recurring diagnostic scan

Systems Manager can diagnose and help you remediate several types of deployment failures, as well as drifted configurations. Systems Manager can also identify Amazon Elastic Compute Cloud (Amazon EC2) instances in your account or organization that Systems Manager isn't able to treat as a *managed node*. The EC2 instance diagnosis process can identify issues related to misconfigurations for a virtual private cloud (VPC), in a Domain Name Service (DNS) setting, or in an Amazon Elastic Compute Cloud (Amazon EC2) security group.

To simply the task of identifying nodes that can't connect to Systems Manager, the **Schedule recurring diagnosis** feature enables you to automate a recurring diagnostic scan. The scans help identify which nodes can't connect to Systems Manager and why. Use the following procedure to enable and configure a recurring diagnostic scan of your nodes.

To schedule a recurring diagnostic scan

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Settings**, and then choose the **Diagnose and remediate** tab.
3. Turn on the **Schedule recurring diagnosis** option.
4. For **Scanning period**, choose how often you want the scan to run.
5. (Optional) For **Start time**, enter a time, in 24-hour format, for the diagnosis to begin. For example, for 8:15 PM, enter **`20:15`**.

   The time you enter is for your current local time zone.

   If you don't specify a time, the diagnostic scan runs immediately. Systems Manager also schedules the scan to run in the future at the current time. If you specify a time, Systems Manager waits to run the diagnostic scan at the specified time.
6. Choose **Save**.
7. After the scan completes, view the details by choosing **Diagnose and remediate** in the left navigation.

For more information about the **Diagnose and remediate** feature, see Diagnosing and remediating.

Updating S3 bucket encryption

When you onboard Systems Manager, Quick Setup creates an Amazon Simple Storage Service (Amazon S3) bucket in the delegated administrator account for AWS Organizations setups. For single-account setups, the bucket is stored in the account being set up. This bucket is used to store the metadata generated during diagnostic scans.

For more information about setting up the unified Systems Manager console, see Setting up AWS Systems Manager.

By default, your data in the bucket is encrypted using a AWS Key Management Service (AWS KMS) key that AWS owns and manages for you.

You can choose to use a different AWS KMS key for your bucket encryption. As another alternative, you can use server-side encryption with AWS KMS keys (SSE-KMS) using a customer managed key (CMK). For information, see Working with Amazon S3 buckets and bucket policies for Systems Manager.

To use a different AWS KMS key for S3 bucket encryption

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Settings**, and then choose the **Diagnose and remediate** tab.
3. In the **Update S3 bucket encryption** area, choose **Edit**.
4. Select the **Customize encryption settings (advanced)** check box.
5. For **Choose an AWS KMS key**, choose or enter the Amazon Resource Name (ARN) of the key.

   > **Tip**
   >
   > To create a new key, choose **Create an AWS KMS key**.

6. Choose **Save**.

Working with Amazon S3 buckets and bucket policies for Systems Manager

During the onboarding process for AWS Systems Manager, Quick Setup creates an Amazon Simple Storage Service (Amazon S3) bucket in the delegated administrator account for organization setups. For single-account setups, the bucket is stored in the account being set up.

You can use Systems Manager to run diagnostic operations on your fleet to identify cases of failed deployments and drifted configurations. Systems Manager can also detect cases where configuration issues are preventing Systems Manager from managing EC2 instances in your account or organization. The results of these diagnostic operations are stored in this Amazon S3 bucket, which is protected by both an encryption method and an S3 bucket policy. For information about the diagnostic operations that output data to this bucket, see Diagnosing and remediating.

Changing the bucket encryption method

By default, the S3 bucket uses server-side encryption with Amazon S3 managed keys (SSE-S3).

You can instead use server-side encryption with AWS KMS keys (SSE-KMS) using a customer managed key (CMK) as an alternative to Amazon S3 managed keys, as explained in Changing to an AWS KMS customer managed key to encrypt S3 resources.

Contents of the bucket policy

The bucket policy prevents member accounts in an organization from discovering one another. Read and write permissions to the bucket are allowed only for the diagnosis and remediation roles created for Systems Manager. The contents of these system-generated policies are presented in S3 bucket policies for the unified Systems Manager console.

> **Warning**
>
> Modifying the default bucket policy might allow member accounts in an organization to discover one another, or read diagnosis outputs for instances in another account. We recommend using extreme caution if you choose to modify this policy.

Topics

- Changing to an AWS KMS customer managed key to encrypt S3 resources

- S3 bucket policies for the unified Systems Manager console

Changing to an AWS KMS customer managed key to encrypt S3 resources

During the onboarding process for the unified Systems Manager console, Quick Setup creates an Amazon Simple Storage Service (Amazon S3) bucket in the delegated administrator account. This bucket is used to store the diagnosis output data generated during remediation runbook executions. By default, the bucket uses server-side encryption with Amazon S3 managed keys (SSE-S3).

You can review the content of these policies in S3 bucket policies for the unified Systems Manager console.

However, you can instead use server-side encryption with AWS KMS keys (SSE-KMS) using a customer managed key (CMK) as an alternative to an AWS KMS key.

Complete the following tasks in order to configure Systems Manager to use your CMK.

Task 1: Add a tag to an existing CMK

AWS Systems Manager uses your CMK only if it is tagged with the following key-value pair:

- Key: `SystemsManagerManaged`
- Value: `true`

Use the following procedure to provide access for encrypting the S3 bucket with your CMK.

To add a tag to your existing CMK

1. Open the AWS KMS console at https://console.aws.amazon.com/kms.
2. In the left navigation, choose **Customer managed keys**.
3. Select the AWS KMS key to use with AWS Systems Manager.
4. Choose the **Tags** tab, and then choose **Edit**.
5. Choose **Add tag**.
6. Do the following:

    a. For **Tag key**, enter **`SystemsManagerManaged`**.

b. For **Tag value**, enter **`true`**.

7. Choose **Save**.

Task 2: Modify an existing CMK key policy

Use the following procedure to update the KMS key policy of your CMK to allow AWS Systems Manager roles to encrypt the S3 bucket on your behalf.

To modify an existing CMK key policy

1. Open the AWS KMS console at https://console.aws.amazon.com/kms.
2. In the left navigation, choose **Customer managed keys**.
3. Select the AWS KMS key to use with AWS Systems Manager.
4. On the **Key policy** tab, choose **Edit**.
5. Add the following JSON statement to the `Statement` field, and replace the *`placeholder values`* with your own information.

   Ensure that you add all AWS account IDs that are onboarded in your organization to AWS Systems Manager in the `Principal` field.

   To locate the correct bucket name in the Amazon S3 console, in the delegated administrator account, locate the bucket in the format `do-not-delete-ssm-`*`operational-account-id`*`-`*`home-region`*`-`*`disambiguator`*.

```
{
     "Sid": "EncryptionForSystemsManagerS3Bucket",
     "Effect": "Allow",
     "Principal": {
         "AWS": [
             "account-id-1",
             "account-id-2",
             ...
         ]
     },
     "Action": ["kms:Decrypt", "kms:GenerateDataKey"],
     "Resource": "*",
     "Condition": {
         "StringEquals": {
             "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket"
```

```
            },
            "StringLike": {
                "kms:ViaService": "s3.*.amazonaws.com"
            },
            "ArnLike": {
                "aws:PrincipalArn": "arn:aws:iam::*:role/AWS-SSM-*"
            }
        }
    }
```

Tip

Alternatively, you can update the CMK key policy using the aws:PrincipalOrgID condition key to grant AWS Systems Manager access to your CMK.

Task 3: Specify the CMK in Systems Manager settings

After completing the previous two tasks, use the following procedure to change the S3 bucket encryption. This change ensures that the associated Quick Setup configuration process can add permissions for Systems Manager to accept your CMK.

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Settings**.
3. On the **Diagnose and remediate** tab, in the **Update S3 bucket encryption** section, choose **Edit**.
4. Select the **Customize encryption settings (advanced)** check box.
5. In the search (🔍) box, choose the ID of an existing key, or paste the ARN of an existing key.
6. Choose **Save**.

S3 bucket policies for the unified Systems Manager console

This topic includes the Amazon S3 bucket policies created by Systems Manager when you onboard an organization or single account to the unified Systems Manager console.

Warning

Modifying the default bucket policy might allow member accounts in an organization to discover one another, or read diagnosis outputs for instances in another account. We recommend using extreme caution if you choose to modify this policy.

Amazon S3 bucket policy for an organization

The diagnosis bucket is created with the following default bucket policy when onboarding an organization to Systems Manager.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyHTTPRequests",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ],
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        },
        {
            "Sid": "DenyNonSigV4Requests",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ],
```

```
            "Condition": {
                "StringNotEquals": {
                    "s3:SignatureVersion": "AWS4-HMAC-SHA256"
                }
            }
        },
        {
            "Sid": "AllowAccessLog",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/access-logs/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "000000000000"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-bucket"
                }
            }
        },
        {
            "Sid": "AllowCrossAccountRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/actions/*/
${aws:PrincipalAccount}/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "organization-id"
                }
            }
        },
        {
            "Sid": "AllowCrossAccountWrite",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:PutObject",
                "s3:DeleteObject"
            ],
```

```
            "Resource": "arn:aws:s3:::bucket-name/actions/*/
${aws:PrincipalAccount}/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "organization-id"
                },
                "ArnLike": {
                    "aws:PrincipalArn": [
                        "arn:aws:iam::*:role/AWS-SSM-
DiagnosisExecutionRole-operational-account-id-home-region",
                        "arn:aws:iam::*:role/AWS-SSM-
DiagnosisAdminRole-operational-account-id-home-region",
                        "arn:aws:iam::*:role/AWS-SSM-
RemediationExecutionRole-operational-account-id-home-region",
                        "arn:aws:iam::*:role/AWS-SSM-
RemediationAdminRole-operational-account-id-home-region"
                    ]
                }
            }
        },
        {
            "Sid": "AllowCrossAccountListUnderAccountOwnPrefix",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "organization-id"
                },
                "StringLike": {
                    "s3:prefix": "*/${aws:PrincipalAccount}/*"
                }
            }
        },
        {
            "Sid": "AllowCrossAccountGetConfigWithinOrganization",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetEncryptionConfiguration",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "organization-id"
```

```
                }
            }
        }
    ]
}
```

Amazon S3 bucket policy for a single account

The diagnosis bucket is created with the following default bucket policy when onboarding a single account to Systems Manager.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyHTTPRequests",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      }
    },
    {
      "Sid": "DenyNonSigV4Requests",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {
        "StringNotEquals": {
```

```
          "s3:SignatureVersion": "AWS4-HMAC-SHA256"
        }
      }
    }
  ]
}
```

Using AWS Systems Manager tools

Systems Manager groups tools into four categories. The following documentation describes the various tools of AWS Systems Manager and how to set up and use these tools. Choose the tabs under each category to learn more about each tool.

Node tools

A *managed node* is any machine configured for use with Systems Manager in hybrid and multicloud environments.

Compliance

Use Compliance 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.

Distributor

Use Distributor 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.

Fleet Manager

Fleet Manager 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.

Hybrid Activations

To set up non-EC2 machines in your hybrid and multicloud environment as managed nodes, create a 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 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.

Inventory

Inventory 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.

Patch Manager

Use Patch Manager 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 tool 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.

Run Command

Use Run Command 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.

Session Manager

Use Session Manager 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.

State Manager

Use State Manager 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.

Change Management tools

Automation

Use Automation 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 helps you set up date and time ranges when actions you specify (for example, in Systems Manager Automation 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 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.

Change Manager

Change 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 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.

Documents

A Systems Manager document (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.

Maintenance Windows

Use Maintenance Windows to set up recurring schedules for managed instances to run administrative tasks such as installing patches and updates without interrupting business-critical operations.

Quick Setup

Use Quick Setup 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.

Application tools

AppConfig

AppConfig 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.

Application Manager

Application Manager 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 tools to a single AWS Management Console.

Parameter Store

Parameter Store 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.

Operations tools

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.

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 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 tool in Systems Manager, to run Automation runbooks and resolve those issues.

Incident Manager

Incident Manager 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.

OpsCenter

OpsCenter provides a central location where operations engineers and IT professionals can view, investigate, and resolve operational work items (OpsItems) related to AWS resources. OpsCenter is designed to reduce mean time to resolution for issues impacting AWS resources. This Systems Manager tool 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.

Perform a node task with Systems Manager

Use this tutorial to get started with AWS Systems Manager. You'll learn how to launch an Amazon Elastic Compute Cloud (Amazon EC2) instance that is managed by Systems Manager, and how to connect to the managed instance.

Because Systems Manager is a collection of multiple tools, no single walkthrough or tutorial can introduce the entire service. This tutorial provides an introduction to some of the tools.

Prerequisites

Before you begin, be sure that you've completed the steps in Managing EC2 instances with Systems Manager.

Launch an instance using an AMI with SSM Agent preinstalled

You can launch an Amazon EC2 instance using the AWS Management Console as described in the following procedure. This tutorial is intended to help you launch your first managed instance quickly, so it doesn't cover all possible options.

To launch an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the EC2 console dashboard, in the **Launch instance** box, choose **Launch instance**, and then choose **Launch instance** from the options that appear.
3. For **Name and tags**, for **Name**, enter a descriptive name for your instance.
4. For **Application and OS Images (Amazon Machine Image)**, do the following:
   a. Choose the **Quick Start** tab, and then choose Amazon Linux. This is the operating system (OS) for your instance.
   b. For **Amazon Machine Image (AMI)**, choose an HVM version of Amazon Linux 2.
5. For **Instance type**, from the **Instance type** list, choose the hardware configuration for your instance. Choose the `t2.micro` instance type, which is selected by default. The `t2.micro` instance type is eligible for the AWS Free Tier. In AWS Regions where `t2.micro` is unavailable, you can use a `t3.micro` instance under the Free Tier. For more information, see AWS Free Tier.
6. For **Key pair (login)**, for **Key pair name**, choose a key pair.

7. For **Network settings**, choose **Edit**. For **Security group name**, notice that the wizard created and selected a security group for you. You can use this security group, or alternatively you can select a security group that you created previously using the following steps:

   a. Choose **Select existing security group**.

   b. From **Common security groups**, choose your security group from the list of existing security groups.

8. If you aren't using Default Host Management Configuration, expand the **Advanced details** section, and for **IAM instance profile**, choose the instance profile that you created when getting set up in Configure instance permissions required for Systems Manager.

9. Keep the default selections for the other configuration settings for your instance.

10. Review a summary of your instance configuration in the **Summary** pane. When you're ready, choose **Launch instance**.

11. A confirmation page informs you that your instance is launching. Choose **View all instances** to close the confirmation page and return to the console.

12. On the **Instances** screen, you can view the status of the launch. It takes a short time for an instance to launch.

13. It can take a few minutes for the instance to show as managed and be ready for you to connect to it. To check that your instance passed its status checks, view this information in the **Status check** column.

Connect to your managed instance using Systems Manager

To connect to your managed instance

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

2. In the navigation pane, choose **Fleet Manager**.

3. Choose the button next to the instance that you want to connect to.

4. In the **Node actions** menu, choose **Start terminal session**.

5. Select **Connect**.

Clean up your instance

If you're done working with the managed instance that you created for this tutorial, terminate it. Terminating an instance effectively deletes it.

To terminate your instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Instances**. In the list of instances, select the instance.
3. Choose **Instance state**, **Terminate instance**.
4. Choose **Terminate** when prompted for confirmation.

   Amazon EC2 shuts down and terminates your instance. After your instance is terminated, it remains visible on the console briefly, and then the entry is deleted automatically. You can't remove the terminated instance from the console display yourself.

AWS Systems Manager Node tools

AWS Systems Manager provides the following tools for accessing, managing, and configuring your *managed nodes*. A managed node is any machine configured for use with Systems Manager in a hybrid and multicloud environment.

Topics

- AWS Systems Manager Compliance
- AWS Systems Manager Distributor
- AWS Systems Manager Fleet Manager
- AWS Systems Manager Hybrid Activations
- AWS Systems Manager Inventory
- AWS Systems Manager Patch Manager
- AWS Systems Manager Run Command
- AWS Systems Manager Session Manager
- AWS Systems Manager State Manager

AWS Systems Manager Compliance

You can use Compliance, a tool in 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 tools in 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.

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 tool in 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 tool is supported as an *event* type in Amazon EventBridge rules. For information, see Monitoring Systems Manager events with Amazon EventBridge and Reference: Amazon EventBridge event patterns and types for Systems Manager.

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.

Pricing

Compliance is offered at no additional charge. You only pay for the AWS resources that you use.

Contents

- Getting started with Compliance
- Configuring permissions for Compliance
- Creating a resource data sync for Compliance
- Learn details about Compliance
- Deleting a resource data sync for Compliance
- Remediating compliance issues using EventBridge
- Assign custom compliance metadata using the AWS CLI

Getting started with Compliance

To get started with Compliance, a tool in 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 tools in AWS Systems Manager.) Compliance also works with custom compliance types on managed nodes that are managed using Systems Manager. Verify that you have completed the setup requirements for your Amazon Elastic Compute Cloud (Amazon EC2) instances and non-EC2 machines in a hybrid and multicloud environment.	Setting up Systems Manager unified console for an organization
Update the AWS Identity and Access Management (IAM) role used by your managed nodes to restrict Compliance permissions.	Configuring permissions for Compliance
If you plan to monitor patch compliance, verify that you've configured Patch Manager.	AWS Systems Manager Patch Manager

Task	For more information
You must perform patching operations by using Patch Manager before Compliance can display patch compliance data.	
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.	AWS Systems Manager State Manager
(Optional) Configure the system to view compliance history and change tracking.	Viewing compliance configuration history and change tracking
(Optional) Create custom compliance types.	Assign custom compliance metadata using the AWS CLI
(Optional) Create a resource data sync to aggregate all compliance data in a target Amazon Simple Storage Service (Amazon S3) bucket.	Creating a resource data sync for Compliance

Configuring permissions for Compliance

As a security best practice, we recommend that you update the AWS Identity and Access Management (IAM) role used by your managed nodes with the following permissions to restrict the node's ability to use the PutComplianceItems API action. This API action registers a compliance type and other compliance details on a designated resource, such as an Amazon EC2 instance or a managed node.

If your node is an Amazon EC2 instance, you must update the IAM instance profile used by the instance with the following permissions. For more information about instance profiles for EC2 instance managed by Systems Manager, see Configure instance permissions required for Systems Manager. For other types of managed nodes, update the IAM role used by the node with the following permissions. For more information, see Update permissions for a role in the *IAM User Guide*.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:PutComplianceItems"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:SourceInstanceARN": "${ec2:SourceInstanceARN}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:PutComplianceItems"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ssm:SourceInstanceARN": "${ssm:SourceInstanceARN}"
                }
            }
        }
    ]
}
```

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 hybrid and multicloud 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 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 amzn-s3-demo-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.

   JSON

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

```
            "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/Bucket-Prefix/*/
accountid=111122223333/*"],
            "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**.
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 S3 bucket prefix (subdirectory).
7. In the **Bucket region** field, choose **This region** if the S3 bucket you created is located in the current AWS Region. If the bucket is located in a different AWS Region, choose **Another region**, and enter the name of the Region.

   > **Note**
   >
   > If the sync and the target 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**.

Learn details about Compliance

Compliance, a tool in 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 tools in 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.

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 Patch compliance state values.

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`.

When State Manager executes an association on a managed node, it triggers a compliance aggregation process that updates compliance status for all associations on that node. The `ExecutionTime` value in compliance reports represents when the compliance status was captured by Systems Manager, not when the association was executed on the managed node. This means multiple associations might display identical `ExecutionTime` values even if they were executed at different times. To determine actual association execution times, refer to the association execution

history using the AWS CLI command describe-association-execution-targets or by viewing the execution details in the console.

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:`. Replace each *`example resource placeholder`* with your own information.

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 Assign custom compliance metadata using the AWS CLI.

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.

Topics

- Viewing current compliance data (console)
- Viewing current compliance data (AWS CLI)

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**.
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.

Viewing current compliance data (AWS CLI)

You can view summaries of compliance data for patching, associations, and custom compliance types in the in the AWS CLI by using the following AWS CLI commands.

list-compliance-summaries

Returns a summary count of compliant and non-compliant association statuses according to the filter you specify. (API: ListComplianceSummaries)

list-resource-compliance-summaries

Returns a resource-level summary count. The summary includes information about compliant and non-compliant statuses and detailed compliance-item severity counts, according to the filter criteria you specify. (API: ListResourceComplianceSummaries)

You can view additional compliance data for patching by using the following AWS CLI commands.

describe-patch-group-state

Returns high-level aggregated patch compliance state for a patch group. (API: DescribePatchGroupState)

describe-instance-patch-states-for-patch-group

Returns the high-level patch state for the instances in the specified patch group. (API: DescribeInstancePatchStatesForPatchGroup)

> **Note**
>
> For an illustration of how to configure patching and view patch compliance details by using the AWS CLI, see Tutorial: Patch a server environment using the AWS CLI.

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**.
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 tools. 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 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 tool in 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 and SSM Command document for patching: `AWS-RunPatchBaseline`.

For more information about associations, see Working with associations in Systems Manager.

For more information about running a command, see AWS Systems Manager Run Command.

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-RefreshAssocation` 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**.
3. Choose **Create rule**.
4. 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.
5. For **Event bus**, 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 **default**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
6. For **Rule type**, choose **Rule with an event pattern**.
7. Choose **Next**.
8. For **Event source**, choose **AWS events or EventBridge partner events**.
9. In the **Event pattern** section, choose **Event pattern form**.
10. For **Event source**, choose **AWS services**.
11. For **AWS service**, choose **Systems Manager**.
12. For **Event type**, choose **Configuration Compliance**.
13. For **Specific detail type(s)**, choose **Configuration Compliance State Change**.
14. Choose **Next**.
15. For **Target types**, choose **AWS service**.
16. For **Select a target**, choose **Systems Manager Run Command**.
17. 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.
18. 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.

19. Choose **Next**.
20. (Optional) Enter one or more tags for the rule. For more information, see Tagging Your Amazon EventBridge Resources in the *Amazon EventBridge User Guide*.
21. Choose **Next**.
22. Review the details of the rule and choose **Create rule**.

Assign custom compliance metadata using the 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.

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 Installing or updating the latest version of the AWS CLI.

2. Run the following command to assign custom compliance metadata to a managed node. Replace each *example resource placeholder* with your own information. 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}"
```

4. 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-02573cafcfEXAMPLE \
    --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-02573cafcfEXAMPLE ^
    --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-02573cafcfEXAMPLE
```

Windows

```
aws ssm list-resource-compliance-summaries ^
    --filters Key=OverallSeverity,Values=UNSPECIFIED
 Key=ComplianceType,Values=Association Key=InstanceId,Values=i-02573cafcfEXAMPLE
```

5. Run the following command to view a summary of compliance statuses. Use filters to drill down into specific compliance data.

```
aws ssm list-resource-compliance-summaries --filters 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
```

6. Run the following command to view a summary count of compliant and non-compliant resources for a compliance type. Use filters to drill down into specific compliance data.

```
aws ssm list-compliance-summaries --filters 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 Distributor

Distributor, a tool in 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
- On a schedule by using AWS Systems Manager State Manager

> **Important**
>
> Packages distributed by third-party sellers are not managed by AWS and are published by the vendor of the package. We encourage you to conduct 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.

- **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 tools in 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”.

  > **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.

What is a package in Distributor?

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.

When you choose **Simple** package creation in the Distributor console, Distributor generates the installation and uninstallation scripts, file hashes, and the JSON package manifest for you, based on the software executable file name and target platforms and architectures.

Supported package platforms and architectures

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.

> **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	Supported architectures
AlmaLinux	`almalinux`	x86_64 ARM64
Amazon Linux 2 and Amazon Linux 2023	`amazon`	x86_64 or x86 ARM64 (Amazon Linux 2 and AL2023, A1 instance types)

Platform	Code value in manifest file	Supported architectures
CentOS	`centos`	x86_64 or x86
Debian Server	`debian`	x86_64 or x86
openSUSE	`opensuse`	x86_64
openSUSE Leap	`opensuseleap`	x86_64
Oracle Linux	`oracle`	x86_64
Red Hat Enterprise Linux (RHEL)	`redhat`	x86_64 ARM64 (RHEL 7.6 and later, A1 instance types)
Rocky Linux	`rocky`	x86_64 ARM64
SUSE Linux Enterprise Server (SLES)	`suse`	x86_64
Ubuntu Server	`ubuntu`	x86_64 or x86 ARM64 (Ubuntu Server 16 and later, A1 instance types)
Windows Server	`windows`	x86_64

Topics

- Setting up Distributor
- Working with Distributor packages
- Auditing and logging Distributor activity
- Troubleshooting AWS Systems Manager Distributor

Setting up Distributor

Before you use Distributor, a tool in AWS Systems Manager, to create, manage, and deploy software packages, follow these steps.

Complete Distributor prerequisites

Before you use Distributor, a tool in AWS Systems Manager, be sure your environment meets the following requirements.

Distributor prerequisites

Requirement	Description
SSM Agent	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. To install or update SSM Agent, see Working with SSM Agent.
AWS CLI	(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. For more information about how to install or upgrade the CLI, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.
AWS Tools for PowerShell	(Optional) To use the Tools for PowerShel l instead of the Systems Manager console to create and manage packages, install the newest release of Tools for PowerShell on your local computer.

Requirement	Description
	For more information about how to install or upgrade the Tools for PowerShell, see Setting up the AWS Tools for Windows PowerShell or AWS Tools for PowerShell Core in the *AWS Tools for PowerShell User Guide*.

> **Note**
>
> Systems Manager doesn't support distributing packages to Oracle Linux managed nodes by using Distributor.

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 tools, not just Distributor.

> **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 peform actions on it. You don't need to configure managed edge devices with an instance profile.

If you already use other Systems Manager tools, 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 Configure instance permissions required for Systems Manager.

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 tools in 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: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 users or groups, see Creating IAM Policies and Adding and Removing IAM Policies in the *IAM User Guide*.

Create or choose an Amazon S3 bucket to store Distributor packages

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 tool in 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 packages

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 tool in 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 Distributor packages.

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 in Distributor
- Create a package in Distributor

- Edit Distributor package permissions in the console
- Edit Distributor package tags in the console
- Add a version to a Distributor package
- Install or update Distributor packages
- Uninstall a Distributor package
- Delete a Distributor package

View packages in Distributor

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 tool in 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 using the 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 in Distributor

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.

When you create a package, the system creates a new SSM document. 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 using the Simple workflow
- Create a package using the Advanced workflow

Create a package using the Simple workflow

This section describes how to create a package in Distributor by choosing the **Simple** package creation workflow in the Distributor console. Distributor is a tool in 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. For more information about supported platforms, see Supported package platforms and architectures.

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 using the Simple workflow

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 on supported versions of both Amazon Linux and Ubuntu Server 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 using the Advanced workflow.
   - 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.
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.
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.

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.
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 Manager Distributor.

Create a package using the Advanced workflow

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
- Step 2: Create the JSON package manifest
- Step 3: Upload the package and manifest to an Amazon S3 bucket
- Step 4: Add a package to Distributor

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, supported versions of 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 Distributor packages.

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.

   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`.

     ```
     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 2023.8 and _any versions are entries under amazon. There is one platform (Amazon Linux), but different supported versions, architectures, and associated .zip files.

```
{
    "amazon": {
        "2023.8": {
            "_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 8.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": {
        "2023.8.20250715 ": {
            "x86_64": {
                "file": "test.zip"
            }
        }
    },
    "redhat": {
        "8.*": {
            "x86_64": {
                "file": "test.zip"
            }
        }
    }
},
```

5. Add the list of .zip files that are part of this package from step 4. Each file entry requires the file name and `sha256` hash value checksum. Checksum values in the manifest must match the `sha256` hash value in the zipped assets to prevent the package installation from failing.

   To get the exact checksum from your installables, you can run the following commands. On Linux, run `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-rhel8-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-rhel8-x86.rpm.zip": {
            "checksums": {
                "sha256":
 "EXAMPLE13df60aa3219bf117638167e5bae0a55467e947a363fff0a51EXAMPLE"
            }
        }
    }
```

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 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. The document allows you to deploy the package to managed nodes.

Topics

- Add a package using the console
- Add a package using the AWS CLI

Add a package using the 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.

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".
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.
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**.

Add a package using the 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.

To add a package to Amazon S3 using the 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. amzn-s3-demo-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="amzn-s3-demo-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/amzn-s3-demo-
   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 Distributor package permissions in the console

After you add a package to Distributor, a tool in 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 in the 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 Distributor package tags in the console

After you have added a package to Distributor, a tool in 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 in the 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 version to a Distributor package

To add a package version, create a package, 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 tool in 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 using the console
- Adding a package version using the AWS CLI

Adding a package version using the console

Before you perform these steps, follow the instructions in Create a package in Distributor 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 using the Simple workflow

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 using the Simple workflow

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 Server 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 using the Advanced workflow.
   - 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.
10. For **Platform version**, verify that the operating system platform version shown is either **_any**, a major release version followed by a wildcard (8.*), 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.
11. For **Architecture**, choose the correct processor architecture for each installable file from the drop-down list. For more information about supported architectures, see Supported package platforms and architectures.
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.

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 Manager Distributor.
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 using the Advanced workflow

To add a package version, create a package, 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 using the Advanced workflow

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.
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 using the 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 using the AWS CLI

1. Run the following command to edit the AWS Systems Manager document with an entry for a new package version. Replace *`document-name`* with the name of your document. Replace

amzn-s3-demo-bucket with the URL of the JSON manifest that you copied in Step 3: Upload the package and manifest to an Amazon S3 bucket. *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="amzn-s3-demo-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/amzn-s3-demo-bucket/ExamplePackage/
manifest.json" \
    --attachments Key="SourceUrl",Values="https://s3.amazonaws.com/amzn-s3-demo-
bucket/ExamplePackage" \
    --version-name 1.1.1 \
    --document-version $LATEST
```

2. Run the following command to verify that your package was updated and show the package manifest. Replace *package-name* with the name of your package, and optionally, *document-version* with the version number of the document (not the same as the package version) that you updated. If this package version is associated with the latest version of the document, you can specify `$LATEST` for the value of the optional `--document-version` parameter.

```
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 Distributor packages

You can deploy packages to your AWS Systems Manager managed nodes by using Distributor, a tool in 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 tool in 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.

> **Important**
>
> Packages that you install using Distributor should be uninstalled only by using Distributor. Otherwise, Systems Manager can still register the application as `INSTALLED` and lead to other unintended results.

Preference	AWS Systems Manager action	More info
Install or update a package immediately.	Run Command	• Installing or updating a package one time using the console • Installing a package one time using the AWS CLI • Updating a package one time using the AWS CLI
Install or update a package on a schedule, so that the installation always includes the default version.	State Manager	• Scheduling a package installation or update using the console • Scheduling a package installation using the AWS CLI

Preference	AWS Systems Manager action	More info
		• Scheduling a package update using the AWS CLI
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	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 Understanding targets and rate controls in State Manager associations.

Topics

- Installing or updating a package one time using the console
- Scheduling a package installation or update using the console
- Installing a package one time using the AWS CLI
- Updating a package one time using the AWS CLI
- Scheduling a package installation using the AWS CLI
- Scheduling a package update using the AWS CLI

Installing or updating a package one time using the 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, a tool in AWS Systems Manager, to perform the installation.

To install or update a package one time using the 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.

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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

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 using the 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 to install or update.

To schedule a package installation using the 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 to 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 Working with associations in Systems Manager. 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 Working with associations in Systems Manager in this guide.

For more information about working with the options in **Advanced options**, **Rate control**, and **Output options**, see Working with associations in Systems Manager.

Installing a package one time using the 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 using the 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-00000000000000" \
    --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 using the 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 using the 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-02573cafcfEXAMPLE" \
    --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 using the 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-02573cafcfEXAMPLE\",
\"i-0471e04240EXAMPLE\"]}]
```

For information about other options you can use with the **create-association** command, see **create-association** in the AWS Systems Manager section of the *AWS CLI Command Reference*.

Scheduling a package update using the 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-02573cafcfEXAMPLE\",
\"i-0471e04240EXAMPLE\"]}]
```

For information about other options you can use with the **create-association** command, see **create-association** in the AWS Systems Manager section of the *AWS CLI Command Reference*.

Uninstall a Distributor 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 tools in 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.

> **Important**
>
> Packages that you install using Distributor should be uninstalled only by using Distributor. Otherwise, Systems Manager can still register the application as `INSTALLED` and lead to other unintended results.

Topics

- Uninstalling a package using the console
- Uninstalling a package using the AWS CLI

Uninstalling a package using the console

You can use Run Command in the Systems Manager console to uninstall a package one time. Distributor uses AWS Systems Manager Run Command to uninstall packages.

To uninstall a package 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**.

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 using the AWS CLI

You can use the AWS CLI to uninstall a Distributor package from managed nodes by using Run Command.

To uninstall a package using the 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-02573cafcfEXAMPLE" \
    --parameters '{"action":["Uninstall"],"name":["Test-ConfigureAWSPackage"]}'
```

For information about other options you can use with the **send-command** command, see **send-command** in the AWS Systems Manager section of the *AWS CLI Command Reference*.

Delete a Distributor package

This section describes how to a delete a package. You can't delete a version of a package, only the entire package.

Deleting a package using the console

You can use the AWS Systems Manager console to delete a package or a package version from Distributor, a tool in AWS Systems Manager. Deleting a package deletes all versions of a package from Distributor.

To delete a package using the 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 delete.
4. On the package's details page, choose **Delete package**.
5. When you're prompted to confirm the deletion, choose **Delete package**.

Deleting a package version using the console

You can use the Systems Manager console to delete a package version from Distributor.

To delete a package version using the 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 delete a version of.
4. On the versions page for the package, choose the version to delete and choose **Delete version**.

5. When you're prompted to confirm the deletion, choose **Delete package version**.

Deleting a package using the command line

You can use your preferred command line tool to delete a package from Distributor.

Linux & macOS

To delete a package using the 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 using the 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 using 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 using the command line

You can use your preferred command line tool to delete a package version from Distributor.

Linux & macOS

To delete a package version using the 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 using the 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 using 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 tool in AWS Systems Manager. For more information about auditing and logging options for Systems Manager, see Logging and monitoring in AWS Systems Manager.

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 tools in 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.

Troubleshooting AWS Systems Manager Distributor

The following information can help you troubleshoot problems that might occur when you use Distributor, a tool in AWS Systems Manager.

Topics

- Wrong package with the same name is installed
- Error: Failed to retrieve manifest: Could not find latest version of package
- Error: Failed to retrieve manifest: Validation exception
- Package isn't supported (package is missing install action)
- Error: Failed to download manifest : Document with name does not exist
- Upload failed.
- Error: Failed to find platform: no manifest found for platform: oracle, version 8.9, architecture x86_64

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 Updating the SSM Agent using Run Command or Walkthrough: Automatically update SSM Agent with the AWS CLI.

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 Updating the SSM Agent using Run Command or Walkthrough: Automatically update SSM Agent with the AWS CLI.

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 1: Distributor can't find the package by the package name when sharing a Distributor package from another account.

Solution 1: When sharing a package from another account, use the full Amazon Resource Name (ARN) for the package and not just its name.

Cause 2: When using a VPC, you haven't provided your IAM instance profile with access to the AWS managed S3 bucket that contains the document `AWS-ConfigureAWSPackage` for the AWS Region you are targeting.

Solution 2: Ensure that your IAM instance profile provides SSM Agent with access to the AWS managed S3 bucket that contains the document `AWS-ConfigureAWSPackage` for the AWS Region you are targeting, as explained in SSM Agent communications with AWS managed S3 buckets.

Upload failed.

Problem: You received an error like the following.

```
Upload failed. At least one of your files was not successfully uploaded to your S3
 bucket.
```

Cause: The name of your software package includes a space. For example, `Hello World.msi` would fail to upload.

Error: Failed to find platform: no manifest found for platform: oracle, version 8.9, architecture x86_64

Problem: You received an error like the following.

```
Failed to find platform: no manifest found for platform: oracle, version 8.9,
 architecture x86_64
```

Cause: The error means that the JSON package manifest does not have any definition for that platform, in this case, Oracle Linux..

Solution: Download the package you want to distribute from the Trend Micro Deep Security Software site. Create an `.rpm` software package using the Distributor Simple workflow process. Set the following values for the package and complete the upload software package using Distributor:

```
Platform version: _any
Target platform: oracle
Architecture: x86_64
```

AWS Systems Manager Fleet Manager

Fleet Manager, a tool in 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 who wants a centralized way to manage their node fleet should use Fleet Manager.

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.
- 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 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, 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

- Setting up Fleet Manager
- Working with managed nodes
- Managing EC2 instances automatically with Default Host Management Configuration
- Connecting to a Windows Server managed instance using Remote Desktop
- Managing Amazon EBS volumes on managed instances
- Accessing the Red Hat Knowledge base portal
- Troubleshooting managed node availability

Setting up Fleet Manager

Before users in your AWS account can use Fleet Manager, a tool in AWS Systems Manager, to monitor and manage your managed nodes, they must be granted the necessary permissions. In addition, any 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 must be Systems Manager *managed nodes*. A managed node is any machine configured for use with Systems Manager in hybrid and multicloud environments.

This means your nodes must meet certain prerequisites and be configured with the AWS Systems Manager Agent (SSM Agent).

Depending on the machine type, refer to one of the following topics to ensure your machines meet the requirements for managed nodes.

- Amazon EC2 instances: Managing EC2 instances with Systems Manager

> **Tip**
>
> You can also use Quick Setup, a tool in 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 Set up Amazon EC2 host management using Quick Setup.

- On-premises and other server types in the cloud: Managing nodes in hybrid and multicloud environments with Systems Manager
- AWS IoT Greengrass (edge) devices: Managing edge devices with Systems Manager

Sample policies

- Controlling access to Fleet Manager

Controlling access to Fleet Manager

To use Fleet Manager, a tool in 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. You then grant these permissions to users, or identities, in your account.

Task 1: Create IAM policies to define access permissions

Follow one of the methods provided in the followig topic in the *IAM User Guide* to create an IAM to provide identities (users, roles, or user groupss) with access to Fleet Manager:

- Define custom IAM permissions with customer managed policies

You can use one of the sample policies we provide below, or modify them according to the permissions you want to grant. We provide sample policies for full Fleet Manager access and read-only access.

Task 2: Attach the IAM policies to users to grant permissions

After you have created the IAM policy or policies that define access permissions to Fleet Manager, use one of the following procedures in the *IAM User Guide* to grant these permissions to identities in your account:

- Adding IAM identity permissions (console)
- Adding IAM identity permissions (AWS CLI)
- Adding IAM identity permissions (AWS API)

Topics

- Sample policy for Fleet Manager administrator access
- Sample policy for Fleet Manager read-only access

Sample policy for Fleet Manager administrator access

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 Server registry keys or values. Replace each *example resource placeholder* with your own information.

JSON

```
{
    "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": "DefaultHostManagement",
            "Effect": "Allow",
            "Action": [
                "ssm:ResetServiceSetting",
                "ssm:UpdateServiceSetting"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:servicesetting/ssm/
managed-instance/default-ec2-instance-management-role"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/
AWSSystemsManagerDefaultEC2InstanceManagementRole",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ssm.amazonaws.com"
                    ]
                }
            }
        },
```

```
        {
            "Sid": "SendCommand",
            "Effect": "Allow",
            "Action": [
                "ssm:GetDocument",
                "ssm:SendCommand",
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ec2:*:111122223333:instance/*",
                "arn:aws:ssm:*:111122223333:managed-instance/*",
                "arn:aws:ssm:*:111122223333:document/SSM-SessionManagerRunShell",
                "arn:aws:ssm:*:*:document/AWS-PasswordReset",
                "arn:aws:ssm:*:*:document/AWSFleetManager-AddUsersToGroups",
                "arn:aws:ssm:*:*:document/AWSFleetManager-CopyFileSystemItem",
                "arn:aws:ssm:*:*:document/AWSFleetManager-CreateDirectory",
                "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-DeleteFileSystemItem",
                "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-GetDiskInformation",
                "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-GetProcessDetails",
                "arn:aws:ssm:*:*:document/AWSFleetManager-GetUsers",
                "arn:aws:ssm:*:*:document/AWSFleetManager-GetWindowsEvents",
                "arn:aws:ssm:*:*:document/AWSFleetManager-
GetWindowsRegistryContent",
                "arn:aws:ssm:*:*:document/AWSFleetManager-MountVolume",
                "arn:aws:ssm:*:*:document/AWSFleetManager-MoveFileSystemItem",
                "arn:aws:ssm:*:*:document/AWSFleetManager-RemoveUsersFromGroups",
                "arn:aws:ssm:*:*:document/AWSFleetManager-RenameFileSystemItem",
```

```
                "arn:aws:ssm:*:*:document/AWSFleetManager-
SetWindowsRegistryValue",
                "arn:aws:ssm:*:*:document/AWSFleetManager-StartProcess",
                "arn:aws:ssm:*:*:document/AWSFleetManager-TerminateProcess"
            ]
        },
        {
            "Sid": "TerminateSession",
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/aws:ssmmessages:session-id": [
                        "${aws:userid}"
                    ]
                }
            }
        }
    ]
}
```

Sample policy for Fleet Manager read-only access

The following policy provides permissions to read-only Fleet Manager features. Replace each *example resource placeholder* with your own information.

JSON

```
{
    "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:*:111122223333:instance/*",
                "arn:aws:ssm:*:111122223333:managed-instance/*",
                "arn:aws:ssm:*:111122223333:document/SSM-SessionManagerRunShell",
                "arn:aws:ssm:*:*:document/AWSFleetManager-GetDiskInformation",
                "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-GetProcessDetails",
                "arn:aws:ssm:*:*:document/AWSFleetManager-GetUsers",
                "arn:aws:ssm:*:*:document/AWSFleetManager-GetWindowsEvents",
                "arn:aws:ssm:*:*:document/AWSFleetManager-
GetWindowsRegistryContent"
            ]
```

```
        },
        {
            "Sid": "TerminateSession",
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/aws:ssmmessages:session-id": [
                        "${aws:userid}"
                    ]
                }
            }
        }
    ]
}
```

Working with managed nodes

A *managed node* is any machine configured for AWS Systems Manager. You can configure the following machine types as managed nodes:

- Amazon Elastic Compute Cloud (Amazon EC2) instances
- Servers on your own premises (on-premises servers)
- AWS IoT Greengrass core devices
- AWS IoT and non-AWS edge devices
- Virtual machines (VMs), including VMs in other cloud environments

In the Systems Manager console, any machine prefixed with "mi-" has been configured as a managed node using a *hybrid activation*. Edge devices display their AWS IoT Thing name.

> **Note**
>
> The only supported feature for macOS instances is viewing the file system.

About Systems Manager instances tiers

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier. Both support managed nodes in your hybrid and multicloud 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. For more information about pricing, see AWS Systems Manager Pricing.

Note the following additional information about the standard-instances tier and advanced-instances tier:

- Advanced instances also allow you to connect to your non-EC2 nodes in a hybrid and multicloud environment 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.
- 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 Patching applications released by Microsoft on Windows Server.

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 the setup steps for your managed nodes meet Systems Manager requirements. For information, see Setting up managed nodes for AWS Systems Manager.
3. For non-EC2 machines, verify that you completed the hybrid activation process. For more information, see Managing nodes in hybrid and multicloud environments with Systems Manager.

Note the following additional information:

- The Fleet Manager console does not display Amazon EC2 nodes that have been terminated.
- 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.
- When you create or edit tags, the system can take up to one hour to display changes in the table filter.
- After the status of a managed node has been `Connection Lost` for at least 30 days, the node might no longer be listed in the Fleet Manager console. To restore it to the list, the issue that caused the lost connection must be resolved. For troubleshooting tips, see Troubleshooting managed node availability.

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.

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. (In order to re-register the same machine, you must use a different hybrid Activation Code and Activation ID than previously used to register it.) For information about deregistering managed nodes, see Deregistering managed nodes in a hybrid and multicloud environment.

Topics

- Configuring instance tiers
- Resetting passwords on managed nodes
- Deregistering managed nodes in a hybrid and multicloud environment

- Working with OS file systems using Fleet Manager
- Monitoring managed node performance
- Working with processes
- Viewing logs on managed nodes
- Managing OS user accounts and groups on managed nodes using Fleet Manager
- Managing the Windows registry on managed nodes

Configuring instance tiers

This topic describes the scenarios when you must activate the advanced-instanced tier.

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for non-EC2 machines in a hybrid and multicloud environment.

You can register up to 1,000 standard hybrid-activated nodes per account per AWS Region at no additional cost. However, registering more than 1,000 hybrid nodes requires that you activate the advanced-instances tier. There is a charge to use the advanced-instances tier. For more information, see AWS Systems Manager Pricing.

Even with fewer than 1,000 registered hybrid-activated nodes, two other scenarios require the advanced-instances tier:

- You want to use Session Manager to connect to non-EC2 nodes.
- You want to patch applications (not operating systems) released by Microsoft on non-EC2 nodes.

> **Note**
>
> There is no charge to patch applications released by Microsoft on Amazon EC2 instances.

Advanced-instances tier detailed scenarios

The following information provides details on the three scenarios for which you must activate the advanced-instances tier.

Scenario 1: You want to register more than 1,000 hybrid-activated nodes

Using the standard-instances tier, you can register a maximum of 1,000 non-EC2 nodes in a hybrid and multicloud environment per AWS Region in a specific account without additional

charge. If you need to register more than 1,000 non-EC2 nodes in a Region, you must use the advanced-instances tier. You can then activate as many machines for your hybrid and multicloud environment as you want. Charges for the advanced-instances tier are based on the number of advanced nodes activated as Systems Manager managed nodes and the hours those nodes are running.

All Systems Manager managed nodes that use the activation process described in Create a hybrid activation to register nodes with Systems Manager are then subject to charge if you exceed 1,000 on-premises nodes in a Region in a specific account .

> **Note**
>
> You can also activate existing Amazon Elastic Compute Cloud (Amazon EC2) instances using Systems Manager hybrid activations and work with them as non-EC2 instances, such as for testing. These also qualify as hybrid nodes. This isn't a common scenario.

Scenario 2: Patching Microsoft-released applications on hybrid-activated nodes

The advanced-instances tier is also required if you want to patch Microsoft-released applications on non-EC2 nodes in a hybrid and multicloud environment. If you activate the advanced-instances tier to patch Microsoft applications on non-EC2 nodes, charges are then incurred for all on-premises nodes, even if you have fewer than 1,000.

There is no additional charge to patch applications released by Microsoft on Amazon Elastic Compute Cloud (Amazon EC2) instances. For more information, see Patching applications released by Microsoft on Windows Server.

Scenario 3: Connecting to hybrid-activated nodes using Session Manager

Session Manager provides interactive shell access to your instances. To connect to hybrid-activated managed nodes using Session Manager, you must activate the advanced-instances tier. Charges are then incurred for all hybrid-activated nodes, even if you have fewer than 1,000.

Summary: When do I need the advanced-instances tier?

Use the following table to review when you must use the advanced-instances tier, and for which scenarios additional charges apply.

Scenario	Advanced-instances tier required?	Additional charges apply?
The number of hybrid-activated nodes in my Region in a specific account is more than 1,000.	Yes	Yes
I want to use Patch Manager to patch Microsoft-released applications on any number of hybrid-activated nodes, even less than 1,000.	Yes	Yes
I want to use Session Manager to connect to any number of hybrid-activated nodes, even less than 1,000.	Yes	Yes
1. The number of hybrid-activated nodes in a Region in a specific account is 1,000 or less; and 2. I am not patching Microsoft applications on any hybrid-activated nodes; and 3. I am not connecting to any hybrid-activated nodes using Session Manager.	No	No

Topics

- Turning on the advanced-instances tier
- Reverting from the advanced-instances tier to the standard-instances tier

Turning on the advanced-instances tier

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for non-EC2 machines in a hybrid and multicloud environment. The standard-instances tier lets you register a maximum of 1,000 hybrid-activated machines per AWS account per AWS Region. The advanced-instances tier is also required to use Patch Manager to patch Microsoft-released applications on non-EC2 nodes, and to connect to non-EC2 nodes using Session Manager. For more information, see Turning on the advanced-instances tier.

This section describes how to configure your hybrid and multicloud environment to use the advanced-instances tier.

Before you begin

Review pricing details for advanced instances. 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` IAM policy attached to your 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:

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetServiceSetting"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:ResetServiceSetting",
                "ssm:UpdateServiceSetting"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333: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.

JSON

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetServiceSetting"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "ssm:ResetServiceSetting",
                "ssm:UpdateServiceSetting"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about creating and editing IAM policies, see Creating IAM Policies in the *IAM User Guide*.

Turning on the advanced-instances tier (console)

The following procedure shows you how to use the Systems Manager console to change *all* non-EC2 nodes that were added using managed-instance activation, in the specified AWS account and AWS Region, to use the advanced-instances tier.

Before you begin

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.

Verify that you have completed the setup requirements for your Amazon Elastic Compute Cloud (Amazon EC2) instances and non-EC2 machines in a hybrid and multicloud environment. For information, see Setting up managed nodes for AWS Systems Manager.

> **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**.
3. Choose **Settings**, and then choose **Change instance tier settings**.
4. Review the information in the dialog box about changing account settings.
5. If you approve, choose the option to accept, and then choose **Change setting**.

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.

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. Replace each *example resource placeholder* with your own information.

   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.

2. 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

1. Open AWS Tools for Windows PowerShell and run the following command. Replace each *example resource placeholder* with your own information.

   ```
   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.

2. 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.

Reverting from the advanced-instances tier to the standard-instances tier

This section describes how to change hybrid-activated nodes running in the advanced-instances tier back to the standard-instances tier. This configuration applies to all hybrid-activated nodes 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-activated nodes in the account and Region. You must first deregister nodes until you have 1,000 or fewer. This also applies to Amazon Elastic Compute Cloud (Amazon EC2) instances that use a Systems Manager hybrid activation (which isn't a common

scenario). For more information, see Deregistering managed nodes in a hybrid and multicloud environment.

- After you revert, you won't be able to use Session Manager, a tool in AWS Systems Manager, to interactively access your hybrid-activated nodes.
- After you revert, you won't be able to use Patch Manager, a tool in AWS Systems Manager, to patch applications released by Microsoft on hybrid-activated nodes.
- The process of reverting all hybrid-activated nodes back to the standard-instance tier can take 30 minutes or more to complete.

This section describes how to revert all hybrid-activated nodes 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 hybrid-activated nodes in your hybrid and multicloud 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**.
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 hybrid-activated nodes in your hybrid and multicloud 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. Replace each *example resource placeholder* with your own information.

   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 hybrid-activated nodes in your hybrid and multicloud 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 tool in 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.
- 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.

  > **Note**
  >
  > Session Manager support for on-premises nodes is provided for the advanced-instances tier only. For more information, see Turning on the advanced-instances tier.

- 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.

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.

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).

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**.
3. Choose the button next to the node that needs a new password.
4. Choose **Instance actions, 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 tool in 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. Replace each *example resource placeholder* with your own information.

   > **Note**
   >
   > To use the AWS CLI to reset a password, the Session Manager plugin must be installed on your local machine. For information, see Install the Session Manager plugin for the AWS CLI.

   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"
   ```

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. For other problems, use the following information to help you troubleshoot password reset issues.

Topics

- Managed node not available
- SSM Agent not up-to-date (console)
- Password reset options aren't provided (AWS CLI)

- No authorization to run ssm:SendCommand
- Session Manager error message

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 Configure instance permissions required for Systems Manager.

  To use a non-EC2 machine with Systems Manager, create an IAM service role that gives Systems Manager permission to perform actions on your managed nodes. For more information, see Create the IAM service role required for Systems Manager in hybrid and multicloud environments. (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.)

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 tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. 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 tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. 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 policy must include permission to run the `ssm:SendCommand` command. For information, see Restricting Run Command access based on tags.

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 and Troubleshooting Session Manager.

Deregistering managed nodes in a hybrid and multicloud 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-activated node removes it from the list of managed nodes in Systems Manager. AWS Systems Manager Agent (SSM Agent) running on the hybrid-activated node 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. Systems Manager stores the command history for a deregistered managed node for 30 days.

> **Note**
>
> You can reregister an on-premises server, edge device, or VM using the same activation code and ID as long as you haven't reached the instance limit for the designated activation code and ID. You can verify the instance limit in the console by choosing **Node tools**, and then choose **Hybrid activations**. If the value of **Registered instances** is less than **Registration limit**, you can reregister a machine using the same activation code and ID. If it's greater, you must use a different activation code and ID.

The following procedure describes how to deregister a hybrid-activated node by using the Systems Manager console. For information about how to do this by using the AWS Command Line Interface, see deregister-managed-instance.

For related information, see the following topics:

- Deregister and reregister a managed node (Linux) (Linux)
- Deregister and reregister a managed node (Windows Server) (Windows Server)

To deregister a hybrid-activated 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**.
3. Select the checkbox next to the managed node that you want to deregister.
4. Choose **Node actions, Tools, Deregister this managed node**.
5. Review the information in the **Deregister this managed node** dialog box. If you approve, choose **Deregister**.

Working with OS file systems using Fleet Manager

You can use Fleet Manager, a tool in 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 tool in 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 Add Session Manager permissions to an existing IAM role.

Topics

- Viewing the OS file system using Fleet Manager
- Previewing OS files using Fleet Manager
- Tailing OS files using Fleet Manager
- Copying, cutting, and pasting OS files or directories using Fleet Manager
- Renaming OS files and directories using Fleet Manager
- Deleting OS files and directories using Fleet Manager
- Creating OS directories using Fleet Manager
- Cutting, copying, and pasting OS directories using Fleet Manager

Viewing the OS file system using Fleet Manager

You can use Fleet Manager to view the OS file system on a Systems Manager managed node.

To view the file OS system using 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**.
3. Select the link of the managed node with the file system you want to view.
4. Choose **Tools, File system**.

Previewing OS files using Fleet Manager

You can use Fleet Manager to preview text files on an OS.

To view text previews of files using 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**.
3. Select the link of the managed node with the files you want to preview.
4. Choose **Tools, 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. Choose **Actions, Preview as text**.

Tailing OS files using Fleet Manager

You can use Fleet Manager to tail a file on a managed node.

To tail OS 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**.

3. Select the link of the managed node with the files you want to tail.
4. Choose **Tools, 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. Choose **Actions, Tail file**.

Copying, cutting, and pasting OS files or directories using Fleet Manager

You can use Fleet Manager to copy, cut, and paste OS files on a managed node.

To copy or cut and paste files or directories using 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**.
3. Select the link of the managed node with the files you want to copy, or cut and paste.
4. Choose **Tools, 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**.

Renaming OS files and directories using Fleet Manager

You can use Fleet Manager to rename files and directories on a managed node in your account.

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**.

3. Select the link of the managed node with the files or directories you want to rename.
4. Choose **Tools, 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. Choose **Actions, Rename**.
8. For **File name**, enter the new name for the file and select **Rename**.

Deleting OS files and directories using Fleet Manager

You can use Fleet Manager to delete files and directories on a managed node in your account.

To delete files or directories using 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**.
3. Select the link of the managed node with the files or directories you want to delete.
4. Choose **Tools, 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. Choose **Actions, Delete**.

Creating OS directories using Fleet Manager

You can use Fleet Manager to create directories on a managed node in your account.

To create a directory using 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**.

3. Select the link of the managed node you want to create a directory in.
4. Choose **Tools, File system**.
5. Select the **File name** of the directory where you want to create a new directory.
6. Select **Create directory**.
7. For **Directory name**, enter the name for the new directory, and then select **Create directory**.

Cutting, copying, and pasting OS directories using Fleet Manager

You can use Fleet Manager to cut, copy, and paste directories on a managed node in your account.

To copy or cut and paste 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**.
3. Select the link of the managed node with the files you want to copy, or cut and paste.
4. Choose **Tools, File system**.
5. Choose the button next to the directory that you want to copy or cut and then proceed to step 8.
6. In the **Actions** menu, choose **Copy** or **Cut**.
7. In the **File system** view, choose the button next to the directory you want to paste the file in.
8. In the **Actions** menu, choose **Paste**.

Monitoring managed node performance

You can use Fleet Manager, a tool in 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 tool in 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 Add Session Manager permissions to an existing IAM role.

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**.
3. Choose the button next to the managed node whose performance you want to monitor.
4. Choose **View details**.
5. Choose **Tools, Performance counters**.

Working with processes

You can use Fleet Manager, a tool in AWS Systems Manager, to work with processes on your managed nodes. 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.

> **Note**
>
> Fleet Manager uses Session Manager, a tool in AWS Systems Manager, to retrieve process 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 Add Session Manager permissions to an existing IAM role.

Topics

- Viewing details about OS processes using Fleet Manager
- Starting an OS process on a managed node using Fleet Manager
- Terminating an OS process using Fleet Manager

Viewing details about OS processes using Fleet Manager

You can use Fleet Manager view details about processes on your managed nodes.

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**.
3. Select the link of the node whose processes you want to view.
4. Choose **Tools, Processes**.

Starting an OS process on a managed node using Fleet Manager

You can use Fleet Manager to start a process on a managed node.

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**.
3. Select the link of the managed node you want to start a process on.
4. Choose **Tools, Processes**.
5. Select **Start new process**.
6. For **Process name or full path**, enter the name of the process or the full path to the executable.
7. (Optional) For **Working directory**, enter the directory path where you want the process to run.

Terminating an OS process using Fleet Manager

To terminate an OS process using 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**.
3. Select the link of the managed node you want to start a process on.
4. Choose **Tools, Processes**.
5. Choose the button next to the process you want to terminate.
6. Choose **Actions, Terminate process** or **Actions, Terminate process tree**.

> **Note**
>
> Terminating a process tree also terminates all processes and applications using that process.

Viewing logs on managed nodes

You can use Fleet Manager, a tool in 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 OS file systems using Fleet Manager.

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**.
3. Choose the button next to the managed node whose event logs you want to view.
4. Choose **View details**.
5. Choose **Tools, 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.

Managing OS user accounts and groups on managed nodes using Fleet Manager

You can use Fleet Manager, a tool in AWS Systems Manager, to manage operating system (OS) user accounts and groups 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, tools in 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 SSM Agent, see Restricting access to root-level commands through SSM Agent. 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 Controlling access to Fleet Manager.

Topics

- Creating an OS user or group using Fleet Manager
- Updating user or group membership using Fleet Manager
- Deleting an OS user or group using Fleet Manager

Creating an OS user or group using Fleet Manager

> **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 Add Session Manager permissions to an existing IAM role.

Instead of logging on directly to a server to create a user account or group, you can use the Fleet Manager console to perform the same tasks.

To create an OS user account using 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**.
3. Choose the button next to the managed node you want to create a new user on.
4. Choose **View details**.
5. Choose **Tools, 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 using 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**.
3. Choose the button next to the managed node you want to create a group in.
4. Choose **View details**.
5. Choose **Tools, 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**.

Updating user or group membership using Fleet Manager

Instead of logging on directly to a server to update a user account or group, you can use the Fleet Manager console to perform the same tasks.

To add an OS user account to a new group using 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**.
3. Choose the button next to the managed node where the user account exists that you want to update.
4. Choose **View details**.
5. Choose **Tools, Users and groups**.
6. Choose the **Users** tab.
7. Choose the button next to the user you want to update.
8. Choose **Actions, 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 using 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**.
3. Choose the button next to the managed node where the group exists that you want to update.
4. Choose **View details**.
5. Choose **Tools, Users and groups**.
6. Choose the **Groups** tab.

7. Choose the button next to the group you want to update.
8. Choose **Actions, Modify group**.
9. Choose the users you want to add or remove under **Group members**.
10. Select **Modify group**.

Deleting an OS user or group using Fleet Manager

Instead of logging on directly to a server to delete a user account or group, you can use the Fleet Manager console to perform the same tasks.

To delete an OS user account using 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**.
3. Choose the button next to the managed node where the user account exists that you want to delete.
4. Choose **View details**.
5. Choose **Users and groups**.
6. Choose the **Users** tab.
7. Choose the button next to the user you want to delete.
8. Choose **Actions, Delete local user**.

To delete an OS group using 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**.
3. Choose the button next to the managed node where the group exists that you want to delete.
4. Choose **View details**.
5. Choose **Tools, Users and groups**.
6. Choose the **Group** tab.
7. Choose the button next to the group you want to update.
8. Choose **Actions, Delete local group**.

Managing the Windows registry on managed nodes

You can use Fleet Manager, a tool in AWS Systems Manager, to manage the registry on your Windows Server 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**.
3. Choose the button next to the managed node you want to create a registry key on.
4. Choose **View details**.
5. Choose **Tools, Windows registry**.
6. Choose the hive you want to create a new registry key in by selecting the **Registry name**.
7. Choose **Create, 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**.
3. Choose the button next to the instance you want to create a registry entry on.
4. Choose **View details**.
5. Choose **Tools, 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. Choose **Create, 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**.
3. Choose the button next to the managed node you want to update a registry entry on.
4. Choose **View details**.
5. Choose **Tools, 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. Choose **Actions, 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**.
3. Choose the button next to the managed node you want to delete a registry key on.
4. Choose **Tools, 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. Choose **Actions, 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**.
3. Choose the button next to the managed node you want to delete a registry entry on.
4. Choose **View details**.
5. Choose **Tools, 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. Choose **Actions, Delete registry entry**.

Managing EC2 instances automatically with Default Host Management Configuration

The Default Host Management Configuration setting allows AWS Systems Manager to manage your Amazon EC2 instances automatically as *managed instances*. A managed instance is an EC2 instance that is configured for use with Systems Manager.

The benefits of managing your instances with Systems Manager include the following:

- Connect to your EC2 instances securely using Session Manager.
- Perform automated patch scans using Patch Manager.
- View detailed information about your instances using Systems Manager Inventory.
- Track and manage instances using Fleet Manager.
- Keep SSM Agent up to date automatically.

Fleet Manager, Inventory, Patch Manager, and Session Manager are tools in Systems Manager.

Using Default Host Management Configuration, you can manage EC2 instances without having to manually create an AWS Identity and Access Management (IAM) instance profile. Instead, Default Host Management Configuration creates and applies a default IAM role to ensure that Systems Manager has permissions to manage all instances in the AWS account and AWS Region where it's activated.

If the permissions provided aren't sufficient for your use case, you can also add policies to the default IAM role created by the Default Host Management Configuration. Alternatively, if you don't need permissions for all of the capabilities provided by the default IAM role, you can create your own custom role and policies. Any changes made to the IAM role you choose for Default Host Management Configuration applies to all managed Amazon EC2 instances in the Region and account.

For more information about the policy used by Default Host Management Configuration, see AWS managed policy: AmazonSSMManagedEC2InstanceDefaultPolicy.

Implement least privilege access

The procedures in this topic are intended to be performed only by administrators. Therefore, we recommend implementing *least privilege access* in order to prevent non-administrative users from configuring or modifying the Default Host Management Configuration. To view example policies that restrict access to the Default Host Management Configuration, see Least privilege policy examples for Default Host Management Configuration later in this topic.

Important

Registration information for instances registered using Default Host Management Configuration is stored locally in the `var/lib/amazon/ssm` or `C:\ProgramData\Amazon` directories. Removing these directories or their files will prevent the instance from acquiring the necessary credentials to connect to Systems Manager using Default

Host Management Configuration. In these cases, you must use an IAM instance profile to provide the required permissions to your instance, or recreate the instance.

Topics

- Prerequisites
- Activating the Default Host Management Configuration setting
- Deactivating the Default Host Management Configuration setting
- Least privilege policy examples for Default Host Management Configuration

Prerequisites

In order to use Default Host Management Configuration in the AWS Region and AWS account where you activate the setting, the following requirements must be met.

- An instance to be managed must use Instance Metadata Service Version 2 (IMDSv2).

  Default Host Management Configuration doesn't support Instance Metadata Service Version 1. For information about transitioning to IMDSv2, see Transition to using Instance Metadata Service Version 2 in the *Amazon EC2 User Guide*
- SSM Agent version 3.2.582.0 or later must be installed on the instance to be managed.

  For information about checking the version of SSM Agent installed on your instance, see Checking the SSM Agent version number.

  For information about updating SSM Agent, see Automatically updating SSM Agent.
- You, as the administrator performing the tasks in this topic, must have permissions for the GetServiceSetting, ResetServiceSetting, and UpdateServiceSetting API operations. Additionally, you must have permissions for the `iam:PassRole` permission for the `AWSSystemsManagerDefaultEC2InstanceManagementRole` IAM role. The following is an example policy providing these permissions. Replace each *`example resource placeholder`* with your own information.

  JSON

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

```
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetServiceSetting",
                "ssm:ResetServiceSetting",
                "ssm:UpdateServiceSetting"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:servicesetting/ssm/
managed-instance/default-ec2-instance-management-role"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/
AWSSystemsManagerDefaultEC2InstanceManagementRole",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ssm.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

- If an IAM instance profile is already attached to an EC2 instance that is to be managed using Systems Manager, you must remove any permissions from it that allow the `ssm:UpdateInstanceInformation` operation. SSM Agent attempts to use instance profile permissions before using the Default Host Management Configuration permissions. If you allow the `ssm:UpdateInstanceInformation` operation in your own IAM instance profile, the instance will not use the Default Host Management Configuration permissions.

Activating the Default Host Management Configuration setting

You can activate Default Host Management Configuration from the Fleet Manager console, or by using the AWS Command Line Interface or AWS Tools for Windows PowerShell.

You must turn on the Default Host Management Configuration one by one in each Region you where you want your Amazon EC2 instances managed by this setting.

After turning on Default Host Management Configuration, it might take up to 30 minutes for your instances to use the credentials of the role you choose in step 5 in the following procedure.

To activate Default Host Management Configuration (console)

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Fleet Manager**.
3. Choose **Account management, Configure Default Host Management Configuration**.
4. Turn on **Enable Default Host Management Configuration**.
5. Choose the AWS Identity and Access Management (IAM) role used to enable Systems Manager tools for your instances. We recommend using the default role provided by Default Host Management Configuration. It contains the minimum set of permissions necessary to manage your Amazon EC2 instances using Systems Manager. If you prefer to use a custom role, the role's trust policy must allow Systems Manager as a trusted entity.
6. Choose **Configure** to complete setup.

To activate Default Host Management Configuration (command line)

1. Create a JSON file on your local machine containing the following trust relationship policy.

   JSON

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

2. Open the AWS CLI or Tools for Windows PowerShell and run one of the following commands, depending on the operating system type of your local machine, to create a service role in your account. Replace each *example resource placeholder* with your own information.

   Linux & macOS

   ```
   aws iam create-role \
   --role-name AWSSystemsManagerDefaultEC2InstanceManagementRole \
   --path /service-role/ \
   --assume-role-policy-document file://trust-policy.json
   ```

   Windows

   ```
   aws iam create-role ^
   --role-name AWSSystemsManagerDefaultEC2InstanceManagementRole ^
   --path /service-role/ ^
   --assume-role-policy-document file://trust-policy.json
   ```

   PowerShell

   ```
   New-IAMRole `
   -RoleName "AWSSystemsManagerDefaultEC2InstanceManagementRole" `
   -Path "/service-role/" `
   -AssumeRolePolicyDocument "file://trust-policy.json"
   ```

3. Run the following command to attach the `AmazonSSMManagedEC2InstanceDefaultPolicy` managed policy to your newly created role. Replace each *example resource placeholder* with your own information.

   Linux & macOS

   ```
   aws iam attach-role-policy \
   --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedEC2InstanceDefaultPolicy \
   --role-name AWSSystemsManagerDefaultEC2InstanceManagementRole
   ```

   Windows

   ```
   aws iam attach-role-policy ^
   --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedEC2InstanceDefaultPolicy ^
   --role-name AWSSystemsManagerDefaultEC2InstanceManagementRole
   ```

PowerShell

```
Register-IAMRolePolicy `
-PolicyArn "arn:aws:iam::aws:policy/AmazonSSMManagedEC2InstanceDefaultPolicy" `
-RoleName "AWSSystemsManagerDefaultEC2InstanceManagementRole"
```

4. Open the AWS CLI or Tools for Windows PowerShell and run the following command. Replace each *example resource placeholder* with your own information.

   Linux & macOS

   ```
   aws ssm update-service-setting \
   --setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/managed-instance/
   default-ec2-instance-management-role \
   --setting-value service-role/AWSSystemsManagerDefaultEC2InstanceManagementRole
   ```

   Windows

   ```
   aws ssm update-service-setting ^
   --setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/managed-instance/
   default-ec2-instance-management-role ^
   --setting-value service-role/AWSSystemsManagerDefaultEC2InstanceManagementRole
   ```

   PowerShell

   ```
   Update-SSMServiceSetting `
   -SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/managed-instance/
   default-ec2-instance-management-role" `
   -SettingValue "service-role/AWSSystemsManagerDefaultEC2InstanceManagementRole"
   ```

   There is no output if the command succeeds.

5. Run the following command to view the current service settings for Default Host Management Configuration 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/managed-instance/
default-ec2-instance-management-role
```

Windows

```
aws ssm get-service-setting ^
--setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/managed-instance/
default-ec2-instance-management-role
```

PowerShell

```
Get-SSMServiceSetting `
-SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/managed-instance/
default-ec2-instance-management-role"
```

The command returns information like the following.

```
{
    "ServiceSetting": {
        "SettingId": "/ssm/managed-instance/default-ec2-instance-management-role",
        "SettingValue": "service-role/
AWSSystemsManagerDefaultEC2InstanceManagementRole",
        "LastModifiedDate": "2022-11-28T08:21:03.576000-08:00",
        "LastModifiedUser": "System",
        "ARN": "arn:aws:ssm:us-east-2:-123456789012:servicesetting/ssm/managed-
instance/default-ec2-instance-management-role",
        "Status": "Custom"
    }
}
```

Deactivating the Default Host Management Configuration setting

You can deactivate Default Host Management Configuration from the Fleet Manager console, or by using the AWS Command Line Interface or AWS Tools for Windows PowerShell.

You must turn off the Default Host Management Configuration setting one by one in each Region where you no longer want your your Amazon EC2 instances managed by this configuration. Deactivating it in one Region doesn't deactivate it in all Regions.

If you deactivate Default Host Management Configuration, and you have not attached an instance profile to your Amazon EC2 instances that allows access to Systems Manager, they will no longer be managed by Systems Manager.

To deactivate Default Host Management Configuration (console)

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Fleet Manager**.
3. Choose **Account management, Default Host Management Configuration**.
4. Turn off **Enable Default Host Management Configuration**.
5. Choose **Configure** to disable Default Host Management Configuration.

To deactivate Default Host Management Configuration (command line)

- Open the AWS CLI or Tools for Windows PowerShell and run the following command. Replace each *example resource placeholder* with your own information.

  Linux & macOS

  ```
  aws ssm reset-service-setting \
  --setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/managed-instance/
  default-ec2-instance-management-role
  ```

  Windows

  ```
  aws ssm reset-service-setting ^
  --setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/managed-instance/
  default-ec2-instance-management-role
  ```

  PowerShell

  ```
  Reset-SSMServiceSetting `
  -SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/managed-instance/
  default-ec2-instance-management-role"
  ```

Least privilege policy examples for Default Host Management Configuration

The following sample policies demonstrate how to prevent members of your organization from making changes to the Default Host Management Configuration setting in your account.

Service control policy for AWS Organizations

The following policy demonstrates how to prevent non-administrative members in your AWS Organizations from updating your Default Host Management Configuration setting. Replace each *example resource placeholder* with your own information.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Deny",
            "Action": [
                "ssm:UpdateServiceSetting",
                "ssm:ResetServiceSetting"
            ],
            "Resource": "arn:aws:ssm:*:*:servicesetting/ssm/managed-instance/
default-ec2-instance-management-role",
            "Condition": {
                "StringNotEqualsIgnoreCase": {
                    "aws:PrincipalTag/job-function": [
                        "administrator"
                    ]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/service-role/
AWSSystemsManagerDefaultEC2InstanceManagementRole",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ssm.amazonaws.com"
                },
                "StringNotEqualsIgnoreCase": {
```

```
                    "aws:PrincipalTag/job-function": [
                        "administrator"
                    ]
                }
            }
        },
        {
            "Effect": "Deny",
            "Resource": "arn:aws:iam::*:role/service-role/
AWSSystemsManagerDefaultEC2InstanceManagementRole",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:DeleteRole"
            ],
            "Condition": {
                "StringNotEqualsIgnoreCase": {
                    "aws:PrincipalTag/job-function": [
                        "administrator"
                    ]
                }
            }
        }
    ]
}
```

Policy for IAM principals

The following policy demonstrates how to prevent IAM groups, roles, or users in your AWS Organizations from updating your Default Host Management Configuration setting. Replace each *example resource placeholder* with your own information.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ssm:UpdateServiceSetting",
                "ssm:ResetServiceSetting"
```

```
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:servicesetting/ssm/
managed-instance/default-ec2-instance-management-role"
        },
        {
            "Effect": "Deny",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:DeleteRole",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/
AWSSystemsManagerDefaultEC2InstanceManagementRole"
        }
    ]
}
```

Connecting to a Windows Server managed instance using Remote Desktop

You can use Fleet Manager, a tool in AWS Systems Manager, to connect to your Windows Server Amazon Elastic Compute Cloud (Amazon EC2) instances using the Remote Desktop Protocol (RDP). Fleet Manager Remote Desktop, which is powered by Amazon DCV, provides you with secure connectivity to your Windows Server instances directly from the Systems Manager console. You can have up to four simultaneous connections in a single browser window.

The Fleet Manager Remote Desktop API is named AWS Systems Manager GUI Connect. For information about using the Systems Manager GUI Connect API, see the *AWS Systems Manager GUI Connect API Reference*.

Currently, you can only use Remote Desktop with instances that are running Windows Server 2012 RTM or higher. Remote Desktop supports only English language inputs.

Fleet Manager Remote Desktop is a console-only service and doesn't support command-line connections to your managed instances. To connect to a Windows Server managed instance through a shell, you can use Session Manager, another tool in AWS Systems Manager. For more information, see AWS Systems Manager Session Manager.

> **Note**
>
> The duration of an RDP connection is not determined by the duration of your AWS Identity and Access Management (IAM) credentials. Instead, the connection persists until the

maximum connection duration or idle time limit is met, whichever comes first. For more information, see Remote connection duration and concurrency.

For information about configuring AWS Identity and Access Management (IAM) permissions to allow your instances to interact with Systems Manager, see Configure instance permissions for Systems Manager.

Topics

- Setting up your environment
- Configuring IAM permissions for Remote Desktop
- Authenticating Remote Desktop connections
- Remote connection duration and concurrency
- Systems Manager GUI Connect handling of AWS IAM Identity Center attributes
- Connect to a managed node using Remote Desktop
- Viewing information about current and completed connections

Setting up your environment

Before using Remote Desktop, verify that your environment meets the following requirements:

- **Managed node configuration**

  Make sure that your Amazon EC2 instances are configured as managed nodes in Systems Manager.

- **SSM Agent minimum version**

  Verify that nodes are running SSM Agent version 3.0.222.0 or higher. For information about how to check which agent version is running on a node, see Checking the SSM Agent version number. For information about installing or updating SSM Agent, see Working with SSM Agent.

- **RDP port configuration**

  To accept remote connections, the Remote Desktop Services service on your Windows Server nodes must use default RDP port 3389. This is the default configuration on Amazon Machine Images (AMIs) provided by AWS. You are not explicitly required to open any inbound ports to use Remote Desktop.

- **PSReadLine module version for keyboard functionality**

  To ensure that your keyboard functions properly in PowerShell, verify that nodes running Windows Server 2022 have PSReadLine module version 2.2.2 or higher installed. If they are running an older version, you can install the required version using the following commands.

  ```
  Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force
  ```

  After the NuGet package provider is installed, run the following command.

  ```
  Install-Module `
   -Name PSReadLine `
   -Repository PSGallery `
   -MinimumVersion 2.2.2 -Force
  ```

- **Session Manager configuration**

  Before you can use Remote Desktop, you must complete the prerequisites for Session Manager setup. When you connect to an instance using Remote Desktop, any session preferences defined for your AWS account and AWS Region are applied. For more information, see Setting up Session Manager.

  > **Note**
  >
  > If you log Session Manager activity using Amazon Simple Storage Service (Amazon S3), then your Remote Desktop connections will generate the following error in `bucket_name/Port/stderr`. This error is expected behavior and can be safely ignored.
  >
  > ```
  > Setting up data channel with id SESSION_ID failed: failed to create websocket
  >  for datachannel with error: CreateDataChannel failed with no output or
  >  error: createDataChannel request failed: unexpected response from the service
  >  <BadRequest>
  > <ClientErrorMessage>Session is already terminated</ClientErrorMessage>
  > </BadRequest>
  > ```

Configuring IAM permissions for Remote Desktop

In addition to the required IAM permissions for Systems Manager and Session Manager, the user or role you use must be allowed permissions for initiating connections.

Permissions for initiating connections

To make RDP connections to EC2 instances in the console, the following permissions are required:

- `ssm-guiconnect:CancelConnection`
- `ssm-guiconnect:GetConnection`
- `ssm-guiconnect:StartConnection`

Permissions for listing connections

In order to view lists of connections in the console, the following permission is required:

`ssm-guiconnect:ListConnections`

The following are example IAM policies that you can attach to a user or role to allow different types of interaction with Remote Desktop. Replace each *`example resource placeholder`* with your own information.

Standard policy for connecting to EC2 instances

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:GetPasswordData"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SSM",
            "Effect": "Allow",
```

```
            "Action": [
                "ssm:DescribeInstanceProperties",
                "ssm:GetCommandInvocation",
                "ssm:GetInventorySchema"
            ],
            "Resource": "*"
        },
        {
            "Sid": "TerminateSession",
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/aws:ssmmessages:session-id": [
                        "${aws:userid}"
                    ]
                }
            }
        },
        {
            "Sid": "SSMStartSession",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ec2:*:111122223333:instance/*",
                "arn:aws:ssm:*:111122223333:managed-instance/*",
                "arn:aws:ssm:*::document/AWS-StartPortForwardingSession"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "ssm-guiconnect.amazonaws.com"
                }
            }
        },
        {
            "Sid": "GuiConnect",
            "Effect": "Allow",
            "Action": [
                "ssm-guiconnect:CancelConnection",
```

```
                "ssm-guiconnect:GetConnection",
                "ssm-guiconnect:StartConnection",
                "ssm-guiconnect:ListConnections"
            ],
            "Resource": "*"
        }
    ]
}
```

Policy for connecting to EC2 instances with specific tags

> **Note**
>
> In the following IAM policy, the `SSMStartSession` section requires an Amazon Resource Name (ARN) for the `ssm:StartSession` action. As shown, the ARN you specify does *not* require an AWS account ID. If you specify an account ID, Fleet Manager returns an `AccessDeniedException`.
> The `AccessTaggedInstances` section, which is located lower in the example policy, also requires ARNs for `ssm:StartSession`. For those ARNs, you do specify AWS account IDs.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:GetPasswordData"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SSM",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeInstanceProperties",
```

```
                "ssm:GetCommandInvocation",
                "ssm:GetInventorySchema"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SSMStartSession",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ssm:*::document/AWS-StartPortForwardingSession"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "ssm-guiconnect.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AccessTaggedInstances",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ec2:*:111122223333:instance/*",
                "arn:aws:ssm:*:111122223333:managed-instance/*"
            ],
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/tag key": [
                        "tag value"
                    ]
                }
            }
        },
        {
            "Sid": "GuiConnect",
            "Effect": "Allow",
            "Action": [
                "ssm-guiconnect:CancelConnection",
                "ssm-guiconnect:GetConnection",
```

```
                "ssm-guiconnect:StartConnection",
                "ssm-guiconnect:ListConnections"
            ],
            "Resource": "*"
        }
    ]
}
```

Policy for AWS IAM Identity Center users to connect to EC2 instances

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SSO",
            "Effect": "Allow",
            "Action": [
                "sso:ListDirectoryAssociations*",
                "identitystore:DescribeUser"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EC2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:GetPasswordData"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SSM",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeInstanceInformation",
                "ssm:DescribeInstanceProperties",
                "ssm:GetCommandInvocation",
                "ssm:GetInventorySchema"
            ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "TerminateSession",
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/aws:ssmmessages:session-id": [
                        "${aws:userName}"
                    ]
                }
            }
        },
        {
            "Sid": "SSMStartSession",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ssm:*:*:managed-instance/*",
                "arn:aws:ssm:*:*:document/AWS-StartPortForwardingSession"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "ssm-guiconnect.amazonaws.com"
                }
            }
        },
        {
            "Sid": "SSMSendCommand",
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ssm:*:*:managed-instance/*",
                "arn:aws:ssm:*:*:document/AWSSSO-CreateSSOUser"
```

```
            ]
        },
        {
            "Sid": "GuiConnect",
            "Effect": "Allow",
            "Action": [
                "ssm-guiconnect:CancelConnection",
                "ssm-guiconnect:GetConnection",
                "ssm-guiconnect:StartConnection",
                "ssm-guiconnect:ListConnections"
            ],
            "Resource": "*"
        }
    ]
}
```

Authenticating Remote Desktop connections

When establishing a remote connection, you can authenticate using Windows credentials or the Amazon EC2 key pair (`.pem` file) that is associated with the instance. For information about using key pairs, see Amazon EC2 key pairs and Windows instances in the *Amazon EC2 User Guide*.

Alternatively, if you're authenticated to the AWS Management Console using AWS IAM Identity Center, you can connect to your instances without providing additional credentials. For an example of a policy to allow remote connection authentication using IAM Identity Center, see Configuring IAM permissions for Remote Desktop.

Before you begin

Note the following conditions for using IAM Identity Center authentication before you begin connecting using Remote Desktop.

- Remote Desktop supports IAM Identity Center authentication for nodes in the same AWS Region where you enabled IAM Identity Center.
- Remote Desktop supports IAM Identity Center user names of up to 16 characters.
- Remote Desktop supports IAM Identity Center user names consisting of alphanumeric characters and the following special characters: `. - _`

> **Important**
>
> Connections won't succeed for IAM Identity Center user names that contain the following characters: + = ,
> IAM Identity Center supports these characters in user names, but Fleet Manager RDP connections do not.
> In addition, if an IAM Identity Center user name contains one or more @ symbols, Fleet Manager disregards the first @ symbol and all characters that follow it, whether or not the @ introduces the domain portion of an email address. For instance, for the IAM Identity Center user name `diego_ramirez@example.com`, the `@example.com` portion is ignored and the user name for Fleet Manager becomes `diego_ramirez`. For `diego_r@mirez@example.com`, Fleet Manager disregards `@mirez@example.com`, and the username for Fleet Manager becomes `diego_r`.

- When a connection is authenticated using IAM Identity Center, Remote Desktop creates a local Windows user in the instance's Local Administrators group. This user persists after the remote connection has ended.
- Remote Desktop does not allow IAM Identity Center authentication for nodes that are Microsoft Active Directory domain controllers.
- Although Remote Desktop allows you to use IAM Identity Center authentication for nodes *joined* to an Active Directory domain, we do not recommend doing so. This authentication method grants administrative permissions to users which might override more restrictive permissions granted by the domain.

Supported Regions for IAM Identity Center authentication

Remote Desktop connections using IAM Identity Center authentication are supported in the following AWS Regions:

- US East (Ohio) (us-east-2)
- US East (N. Virginia) (us-east-1)
- US West (N. California) (us-west-1)
- US West (Oregon) (us-west-2)
- Africa (Cape Town) (af-south-1)
- Asia Pacific (Hong Kong) (ap-east-1)

- Asia Pacific (Mumbai) (ap-south-1)
- Asia Pacific (Tokyo) (ap-northeast-1)
- Asia Pacific (Seoul) (ap-northeast-2)
- Asia Pacific (Osaka) (ap-northeast-3)
- Asia Pacific (Singapore) (ap-southeast-1)
- Asia Pacific (Sydney) (ap-southeast-2)
- Asia Pacific (Jakarta) (ap-southeast-3)
- Canada (Central) (ca-central-1)
- Europe (Frankfurt) (eu-central-1)
- Europe (Stockholm) (eu-north-1)
- Europe (Ireland) (eu-west-1)
- Europe (London) (eu-west-2)
- Europe (Paris) (eu-west-3)
- Israel (Tel Aviv) (il-central-1)
- South America (São Paulo) (sa-east-1)
- Europe (Milan) (eu-south-1)
- Middle East (Bahrain) (me-south-1)
- AWS GovCloud (US-East) (us-gov-east-1)
- AWS GovCloud (US-West) (us-gov-west-1)

Remote connection duration and concurrency

The following conditions apply to active Remote Desktop connections:

- **Connection duration**

  By default, a Remote Desktop connection is disconnected after 60 minutes. To prevent a connection from being disconnected, you can choose **Renew session** before being disconnected to reset the duration timer.
- **Connection timeout**

  A Remote Desktop connection disconnects after it has been idle for more than 10 minutes.
- **Connection persistence**

After you connect to a Windows Server using Remote Desktop, the connection persists until the maximum connection duration (60 minutes) or idle timeout limit (10 minutes) is met. Connection duration is not determined by the duration of your AWS Identity and Access Management (IAM) credentials. The connection persists after IAM credentials expire if the connection duration limits are not met. When using Remote Desktop, you should terminate your connection after your IAM credentials expire by leaving the browser page.

- **Concurrent connections**

  By default, you can have a maximum of 5 active Remote Desktop connections at one time for the same AWS account and AWS Region. To request a service quota increase of up to 50 concurrent connections, see Requesting a quota increase in the *Service Quotas User Guide*.

> **Note**
>
> The standard license for Windows Server allows for two concurrent RDP connections. To support more connections, you must purchase additional Client Access Licenses (CALs) from Microsoft or Microsoft Remote Desktop Services licenses from AWS. For more information on supplemental licensing, see the following topics:
>
> - Client Access Licenses and Management Licenses on the Microsoft website
> - Use License Manager user-based subscriptions for supported software products in the *License Manager User Guide*

Systems Manager GUI Connect handling of AWS IAM Identity Center attributes

Systems Manager GUI Connect is the API that supports Fleet Manager connections to EC2 instances using RDP. The following IAM Identity Center user data is retained after a connection is closed:

- `username`

Systems Manager GUI Connect encrypts this identity attribute at rest using an AWS managed key by default. Customer managed keys are not supported for encrypting this attribute in Systems Manager GUI Connect. If you delete a user in your IAM Identity Center instance, Systems Manager GUI Connect continues to retain the `username` attribute associated with that user for 7 years, after which it is deleted. This data is retained to support auditing events, such as listing Systems Manager GUI Connect connection history. The data can't be deleted manually.

Connect to a managed node using Remote Desktop

Browser copy/paste support for text

Using the Google Chrome and Microsoft Edge browsers, you can copy and paste text from a managed node to your local machine, and from your local machine to a managed node that you are connected to.

Using the Mozilla Firefox browser, you can copy and paste text from a managed node to your local machine only. Copying from your local machine to the managed node is not supported.

To connect to a managed node using Fleet Manager Remote Desktop

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Fleet Manager**.
3. Choose the node that you want to connect to. You can select either the check box or the node name.
4. On 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 a Windows user account on the node that you're connecting to. If you choose **Key pair**, you can provide authentication using one of the following methods:

   a. Choose **Browse local machine** if you want to select the PEM key associated with your instance from your local file system.

   - or -

   b. Choose **Paste key pair content** if you want to copy the contents of the PEM file and paste them in to the provided field.
6. Select **Connect**.
7. To choose your preferred display resolution, in the **Actions** menu, choose **Resolutions**, and then select from the following:

   - **Adapt Automatically**
   - **1920 x 1080**
   - **1400 x 900**
   - **1366 x 768**

- **800 x 600**

The **Adapt Automatically** option sets the resolution based on your detected screen size.

Viewing information about current and completed connections

You can use the Fleet Manager section of the Systems Manager console to view information about RDP connections that have been made in your account. Using a set of filters, you can limit the list of connections displayed to a time range, a specific instance, the user who made the connections, and connections of a specific status. The console also provides tabs that show information about all currently active connections, and all past connections.

To view information about current and completed connections

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Fleet Manager**.
3. Choose **Account management, Connect with Remote Desktop**.
4. Choose one of the following tabs:
   - **Active connections**
   - **Connection history**
5. To further narrow the list of connection results displayed, specify one or more filters in the search
   ()
   box. You can also enter a free-text search term.

Managing Amazon EBS volumes on managed instances

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with Amazon Elastic Compute Cloud (EC2) instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances.

You can use Fleet Manager, a tool in AWS Systems Manager, to manage Amazon EBS volumes on your managed instances. For example, you can initialize an EBS volume, format a partition, and mount the volume to make it available for use.

> **Note**
>
> Fleet Manager currently supports Amazon EBS volume management for Windows Server instances only.

View EBS volume details

To view details for an EBS volume 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**.
3. Choose the button next to the managed instance for which you want to view EBS volume details.
4. Choose **View details**.
5. Choose **Tools, EBS volumes**.
6. To view details for an EBS volume, choose its ID in the **Volume ID** column.

Initialize and format an EBS volume

To initialize and format an EBS volume 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**.
3. Choose the button next to the managed instance for which you want to initialize, format, and mount an EBS volume. You can only initialize an EBS volume if its disk is empty.
4. Choose **View details**.
5. In the **Tools** menu, choose **EBS volumes**.
6. Choose the button next to the EBS volume you want to initialize and format.
7. Choose **Initialize and format**.
8. In **Partition style**, choose the partition style you want to use for the EBS volume.
9. (Optional) Choose a **Drive letter** for the partition.

10. (Optional) Enter a **Partition name** to identify the partition.
11. Choose the **File system** to use to organize files and data stored in the partition.
12. Choose **Confirm** to make the EBS volume available for use. You can't change the partition configuration from the AWS Management Console after confirming, however, you can use SSH or RDP to log into the instance to change the partition configuration.

Accessing the Red Hat Knowledge base portal

You can use Fleet Manager, a tool in AWS Systems Manager, to access the Knowledge base 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 Knowledge base 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 Knowledge base 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**.
3. Choose the RHEL instance you want to use to connect to the Red Hat Knowledgebase Portal.
4. Choose **Account management**, **Access Red Hat Knowledgebase** to open the Red Hat Knowledge base page.

If you use RHEL on AWS to run fully supported RHEL workloads, you can also access the Red Hat Knowledge base through Red Hat's website by using your AWS credentials.

Troubleshooting managed node availability

For several AWS Systems Manager tools 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 preinstalled. (You can also configure a custom AMI to preinstall SSM Agent.) For more information, see Find AMIs with the SSM Agent preinstalled.

- 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 the status of a managed node has been `Connection Lost` for at least 30 days, the node might no longer be listed in the Fleet Manager console. To restore it to the list, the issue that caused the lost connection must be resolved.

After you verify that a managed node is running, you can use the following command to check whether SSM Agent 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-02573cafcfEXAMPLE",
            "Status": "Pending",
            "DetailedStatus": "Associated"
        },
        {
            "AssociationId": "f9ec7a0f-6104-4273-8975-82e34EXAMPLE",
            "Name": "AWS-RunPatchBaseline",
            "DocumentVersion": "1",
            "AssociationVersion": "1",
            "InstanceId": "i-02573cafcfEXAMPLE",
            "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
- Solution 2: Verify that an IAM instance profile has been specified for the instance (EC2 instances only)
- Solution 3: Verify service endpoint connectivity
- Solution 4: Verify target operating system support
- Solution 5: Verify you're working in the same AWS Region as the Amazon EC2 instance
- Solution 6: Verify the proxy configuration you applied to SSM Agent on your managed node
- Solution 7: Install a TLS certificate on managed instances
- Troubleshooting managed node availability using ssm-cli

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.

To install or reinstall SSM Agent on a managed node, see the following topics:

- Manually installing and uninstalling SSM Agent on EC2 instances for Linux
- How to install the SSM Agent on hybrid Linux nodes
- Manually installing and uninstalling SSM Agent on EC2 instances for Windows Server
- How to install the SSM Agent on hybrid Windows nodes

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 has an IAM trust policy that allows your user 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 the IAM service role required for Systems Manager in hybrid and multicloud environments.

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 Configure instance permissions required for Systems Manager.

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 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 Improve the security of EC2 instances by using VPC endpoints for Systems Manager.

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 SSM Agent on your managed node

Verify that the proxy configuration you applied to 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 on Linux nodes and Configure SSM Agent to use a proxy for Windows Server instances.

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 managed node availability using `ssm-cli`

The `ssm-cli` is a standalone command line tool included in the SSM Agent installation. When you install SSM Agent 3.1.501.0 or later on a machine, you can run `ssm-cli` commands on that machine. The output of those commands helps you determine whether the machine meets the minimum requirements for an Amazon EC2 instance or non-EC2 machine to be managed by AWS

Systems Manager, and therefore added to lists of managed nodes in Systems Manager. (SSM Agent version 3.1.501.0 was released in November, 2021.)

Minimum requirements

For an Amazon EC2 instance or non-EC2 machine to be managed by AWS Systems Manager, and available in lists of managed nodes, it must meet three primary requirements:

- SSM Agent must be installed and running on a machine with a supported operating system.

  Some AWS managed Amazon Machine Images (AMIs) for EC2 are configured to launch instances with SSM Agent preinstalled. (You can also configure a custom AMI to preinstall SSM Agent.) For more information, see Find AMIs with the SSM Agent preinstalled.
- An AWS Identity and Access Management (IAM) instance profile (for EC2 instances) or IAM service role (for non-EC2 machines) that supplies the required permissions to communicate with the Systems Manager service must be attached to the machine.
- SSM Agent must be able to connect to a Systems Manager endpoint 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 managed node's health.

Preconfigured commands in `ssm-cli`

Preconfigured commands are included that gather the required information to help you diagnose why a machine that you have confirmed is running isn't included in your lists of managed nodes in Systems Manager. These commands are run when you specify the `get-diagnostics` option.

On the machine, run the following command to use `ssm-cli` to help you troubleshoot managed node availability.

Linux & macOS

```
ssm-cli get-diagnostics --output table
```

Windows

On Windows Server machines, you must navigate to the `C:\Program Files\Amazon\SSM` directory before running the command.

```
ssm-cli.exe get-diagnostics --output table
```

PowerShell

On Windows Server machines, you must navigate to the `C:\Program Files\Amazon\SSM` directory before running the command.

```
.\ssm-cli.exe get-diagnostics --output table
```

The command returns output as a table similar to the following.

> **Note**
>
> 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 Server and PowerShell

```
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 3.2.815.0,
 latest agent version in us-east-2   #
#                                       #         # is 3.2.985.0
                                     #
```

```
################################################################################################
```

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	Indicates whether the managed node is able to reach the metadata service. A failed test indicates a connectivity issue to `http://169.254.169.254` which can be caused by local route, proxy, or operating system (OS) firewall and proxy configurations.
Hybrid instance registration	Indicates whether SSM Agent is registered using a hybrid activation.
Connectivity to `ssm` endpoint	Indicates whether the node is able to reach the service endpoints for Systems Manager on TCP port 443. A failed test indicates connectivity issues to `https://ssm.`*`region`*`.amazonaws.com` depending on the AWS Region where the node 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	Indicates whether the node is able to reach the service endpoints for Systems Manager on TCP port 443. A failed test indicates connectivity issues to `https://ec2messages.`*`region`*`.amazonaws.com` depending on the AWS Region where the node is located. Connectivity issues can be caused by the VPC configuration including security groups,

Check	Details
	network access control lists, route tables, or OS firewalls and proxies.
Connectivity to `ssmmessages` endpoint	Indicates whether the node is able to reach the service endpoints for Systems Manager on TCP port 443. A failed test indicates connectivity issues to `https://ssmmessages.` *region* `.amazonaws.com` depending on the AWS Region where the node 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	Indicates whether the node is able to reach the service endpoint for Amazon Simple Storage Service on TCP port 443. A failed test indicates connectivity issues to `https://s3.` *region* `.amazonaws.com` depending on the AWS Region where the node is located. Connectivity to this endpoint is not required for a node to appear in your managed nodes list.
Connectivity to `kms` endpoint	Indicates whether the node is able to reach the service endpoint for AWS Key Management Service on TCP port 443. A failed test indicates connectivity issues to `https://kms.` *region* `.amazonaws.com` depending on the AWS Region where the node is located. Connectivity to this endpoint is not required for a node to appear in your managed nodes list.

Check	Details
Connectivity to `logs` endpoint	Indicates whether the node is able to reach the service endpoint for Amazon CloudWatch Logs on TCP port 443. A failed test indicates connectivity issues to `https://logs.` *`region`*`.amazonaws.com` depending on the AWS Region where the node is located. Connectivity to this endpoint is not required for a node to appear in your managed nodes list.
Connectivity to `monitoring` endpoint	Indicates whether the node is able to reach the service endpoint for Amazon CloudWatch on TCP port 443. A failed test indicates connectivity issues to `https://monitoring.` *`region`*`.amazonaws.com` depending on the AWS Region where the node is located. Connectivity to this endpoint is not required for a node to appear in your managed nodes list.
AWS Credentials	Indicates whether SSM Agent has the required credentials based on the IAM instance profile (for EC2 instances) or IAM service role (for non-EC2 machines) attached to the machine. A failed test indicates that no IAM instance profile or IAM service role is attached to the machine, or it does not contain the required permissions for Systems Manager.
Agent service	Indicates whether SSM Agent service is running, and whether the service is running as root for Linux or macOS, or SYSTEM for Windows Server. A failed test indicates SSM Agent service is not running or is not running as root or SYSTEM.

Check	Details
Proxy configuration	Indicates whether SSM Agent is configured to use a proxy.
Sysprep image state (Windows only)	Indicates the state of `Sysprep` on the node. SSM Agent will not start on the node if the `Sysprep` state is a value other than `IMAGE_STATE_COMPLETE` .
SSM Agent version	Indicates whether the latest available version of SSM Agent is installed.

AWS Systems Manager Hybrid Activations

To configure non-EC2 machines for use with AWS Systems Manager in a hybrid and multicloud environment, you create a *hybrid activation*. Non-EC2 machine types supported as managed nodes include the following:

- Servers on your own premises (on-premises servers)
- AWS IoT Greengrass core devices
- AWS IoT and non-AWS edge devices
- Virtual machines (VMs), including VMs in other cloud environments

When you run the create-activation command to start a hybrid activation process, you receive an activation code and ID in the command response. You then include the activation code and ID with the command to install SSM Agent on the machine, as described in step 3 of Install SSM Agent on hybrid Linux nodes and step 4 of Install SSM Agent on hybrid Windows Server nodes.

This activation process applies to all non-EC2 machine types *except* AWS IoT Greengrass core devices. For information about configuring AWS IoT Greengrass core devices for Systems Manager, see Managing edge devices with Systems Manager.

> **Note**
>
> Support isn't currently provided for non-EC2 macOS machines.

About Systems Manager instances tiers

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier. Both support managed nodes in your hybrid and multicloud 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. For more information about pricing, see AWS Systems Manager Pricing.

Note the following additional information about the standard-instances tier and advanced-instances tier:

- Advanced instances also allow you to connect to your non-EC2 nodes in a hybrid and multicloud environment 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.
- 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 Patching applications released by Microsoft on Windows Server.

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 by using Amazon Athena. 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.

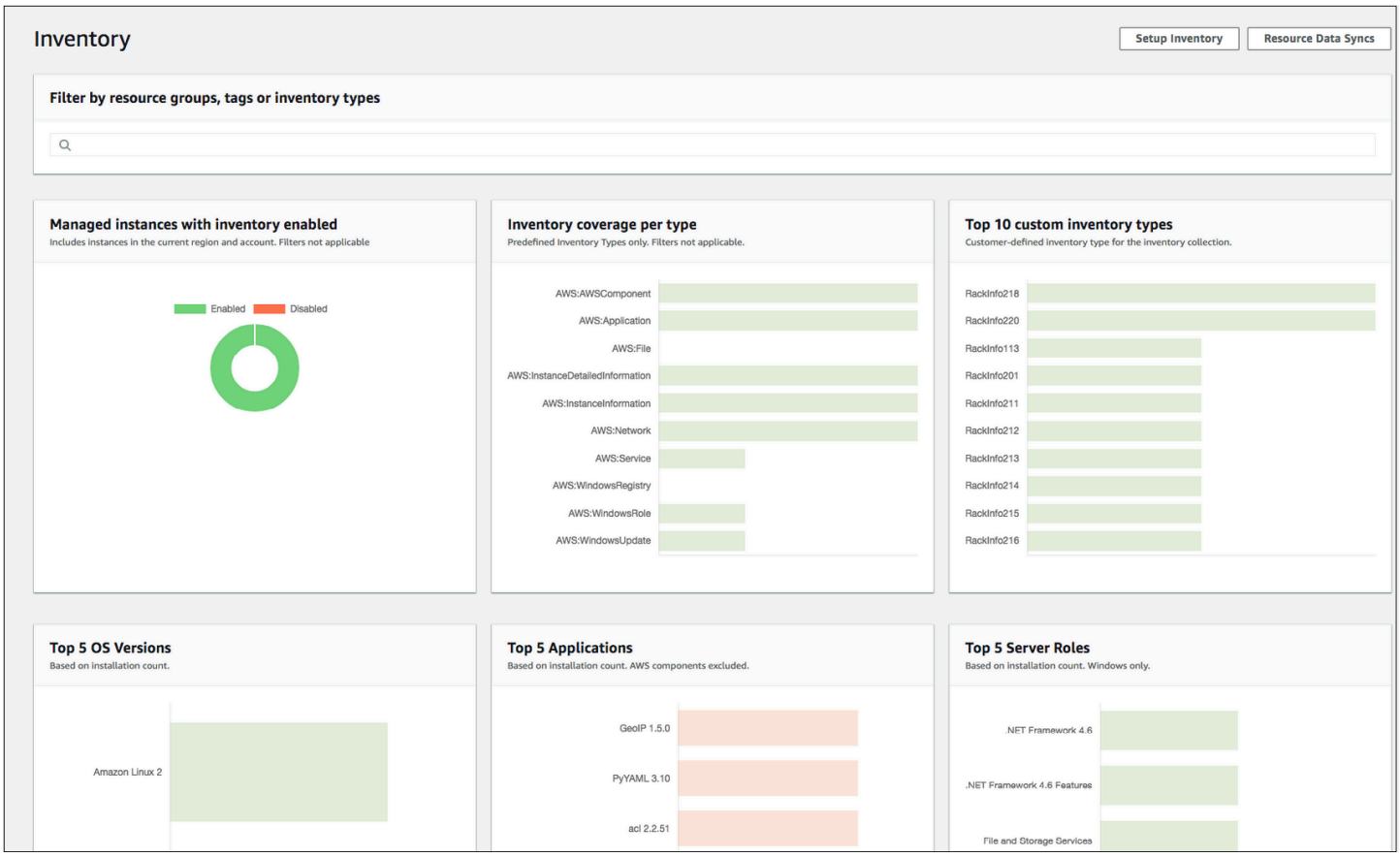
Configuration	Details
Metadata types	You can configure Inventory to collect the following types of data: • **Applications**: Application names, publishers, versions, etc. • **AWS components**: EC2 driver, agents, versions, etc. • **Files**: Name, size, version, installed date, modification and last accessed times, etc. • **Network configuration**: IP address, MAC address, DNS, gateway, subnet mask, etc. • **Windows updates**: Hotfix ID, installed by, installed date, etc. • **Instance details**: CPUModel, CPUCores, CPUs, CPUSpeedMHz, CPUSockets, CPUHyperThreadEnabled, OSService Packetc.

Configuration	Details
	• **Services**: Name, display name, status, dependent services, service type, start type, etc. • **Tags**: Tags assigned to your nodes. • **Windows Registry**: Registry key path, value name, value type, and value. • **Windows roles**: Name, display name, path, feature type, installed state, etc. • **Custom inventory**: Metadata that was assigned to a managed node as described in Working with custom inventory. **Note** To view a list of all metadata collected by Inventory, see Metadata collected by Inventory.
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 collectin g inventory data from all of your managed nodes, see Inventory all managed nodes in your AWS account.
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.

EventBridge support

This Systems Manager tool is supported as an *event* type in Amazon EventBridge rules. For information, see Monitoring Systems Manager events with Amazon EventBridge and Reference: Amazon EventBridge event patterns and types for Systems Manager.

Contents

- Learn more about Systems Manager Inventory
- Setting up Systems Manager Inventory
- Configuring inventory collection
- Querying inventory data from multiple Regions and accounts
- Querying an inventory collection by using filters
- Aggregating inventory data
- Working with custom inventory
- Viewing inventory history and change tracking
- Stopping data collection and deleting inventory data
- Assigning custom inventory metadata to a managed node
- Using the AWS CLI to configure inventory data collection
- Walkthrough: Using resource data sync to aggregate inventory data
- Troubleshooting problems with Systems Manager Inventory

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
- Working with file and Windows registry inventory

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: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",                "dataType": "STRING",
"isContext": "true"},
      { "name": "ExecutionTime",                "dataType": "STRING",
"isContext": "true"},
      { "name": "Id",                           "dataType": "STRING"},
      { "name": "Title",                        "dataType": "STRING"},
      { "name": "Status",                       "dataType": "STRING"},
      { "name": "Severity",                     "dataType": "STRING"},
      { "name": "DocumentName",                 "dataType": "STRING"},
      { "name": "DocumentVersion",              "dataType": "STRING"},
      { "name": "Classification",               "dataType": "STRING"},
      { "name": "PatchBaselineId",              "dataType": "STRING"},
      { "name": "PatchSeverity",                "dataType": "STRING"},
      { "name": "PatchState",                   "dataType": "STRING"},
      { "name": "PatchGroup",                   "dataType": "STRING"},
      { "name": "InstalledTime",                "dataType": "STRING"},
      { "name": "InstallOverrideList",          "dataType": "STRING",
"isOptional": "true"},
      { "name": "DetailedText",                 "dataType": "STRING",
"isOptional": "true"},
      { "name": "DetailedLink",                 "dataType": "STRING",
"isOptional": "true"},
```

```
      { "name": "CVEIds",                        "dataType": "STRING",
 "isOptional": "true"}
    ]
  },
  {
    "typeName": "AWS:ComplianceSummary",
    "version":"1.0",
    "attributes":[
      { "name": "ComplianceType",                "dataType": "STRING"},
      { "name": "PatchGroup",                    "dataType": "STRING"},
      { "name": "PatchBaselineId",               "dataType": "STRING"},
      { "name": "Status",                        "dataType": "STRING"},
      { "name": "OverallSeverity",               "dataType": "STRING"},
      { "name": "ExecutionId",                   "dataType": "STRING"},
      { "name": "ExecutionType",                 "dataType": "STRING"},
      { "name": "ExecutionTime",                 "dataType": "STRING"},
      { "name": "CompliantCriticalCount",        "dataType": "NUMBER"},
      { "name": "CompliantHighCount",            "dataType": "NUMBER"},
      { "name": "CompliantMediumCount",          "dataType": "NUMBER"},
      { "name": "CompliantLowCount",             "dataType": "NUMBER"},
      { "name": "CompliantInformationalCount",   "dataType": "NUMBER"},
      { "name": "CompliantUnspecifiedCount",     "dataType": "NUMBER"},
      { "name": "NonCompliantCriticalCount",     "dataType": "NUMBER"},
      { "name": "NonCompliantHighCount",         "dataType": "NUMBER"},
      { "name": "NonCompliantMediumCount",       "dataType": "NUMBER"},
      { "name": "NonCompliantLowCount",          "dataType": "NUMBER"},
      { "name": "NonCompliantInformationalCount", "dataType": "NUMBER"},
      { "name": "NonCompliantUnspecifiedCount",  "dataType": "NUMBER"}
    ]
  },
  {
    "typeName": "AWS:InstanceDetailedInformation",
    "version":"1.0",
    "attributes":[
      { "name": "CPUModel",                      "dataType": "STRING"},
      { "name": "CPUCores",                      "dataType": "NUMBER"},
      { "name": "CPUs",                          "dataType": "NUMBER"},
      { "name": "CPUSpeedMHz",                   "dataType": "NUMBER"},
      { "name": "CPUSockets",                    "dataType": "NUMBER"},
      { "name": "CPUHyperThreadEnabled",         "dataType": "STRING"},
      { "name": "OSServicePack",                 "dataType": "STRING"}
    ]
   },
   {
```

```
     "typeName": "AWS:Service",
     "version":"1.0",
     "attributes":[
       { "name": "Name",                          "dataType": "STRING"},
       { "name": "DisplayName",                   "dataType": "STRING"},
       { "name": "ServiceType",                   "dataType": "STRING"},
       { "name": "Status",                        "dataType": "STRING"},
       { "name": "DependentServices",             "dataType": "STRING"},
       { "name": "ServicesDependedOn",            "dataType": "STRING"},
       { "name": "StartType",                     "dataType": "STRING"}
     ]
    },
    {
      "typeName": "AWS:WindowsRegistry",
      "version":"1.0",
      "attributes":[
        { "name": "KeyPath",                          "dataType": "STRING"},
        { "name": "ValueName",                        "dataType": "STRING"},
        { "name": "ValueType",                        "dataType": "STRING"},
        { "name": "Value",                            "dataType": "STRING"}
      ]
    },
    {
      "typeName": "AWS:WindowsRole",
      "version":"1.0",
      "attributes":[
        { "name": "Name",                           "dataType": "STRING"},
        { "name": "DisplayName",                    "dataType": "STRING"},
        { "name": "Path",                           "dataType": "STRING"},
        { "name": "FeatureType",                    "dataType": "STRING"},
        { "name": "DependsOn",                      "dataType": "STRING"},
        { "name": "Description",                    "dataType": "STRING"},
        { "name": "Installed",                      "dataType": "STRING"},
        { "name": "InstalledState",                 "dataType": "STRING"},
        { "name": "SubFeatures",                    "dataType": "STRING"},
        { "name": "ServerComponentDescriptor",      "dataType": "STRING"},
        { "name": "Parent",                         "dataType": "STRING"}
      ]
    },
    {
      "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.
- Some metadata is not available from Linux instances. Specifically, for "typeName": "AWS:Network", the following metadata types are not yet supported for Linux instances. They ARE supported for Windows.
  - { "name": "SubnetMask", "dataType": "STRING"},
  - { "name": "DHCPServer", "dataType": "STRING"},
  - { "name": "DNSServer", "dataType": "STRING"},
  - { "name": "Gateway", "dataType": "STRING"},

Working with file and Windows registry inventory

AWS Systems Manager Inventory allows you to search and inventory files on Windows Server, 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. 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"]}
```

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

- Creating a resource data sync for Inventory
- Using EventBridge to monitor Inventory events

Creating a 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.

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

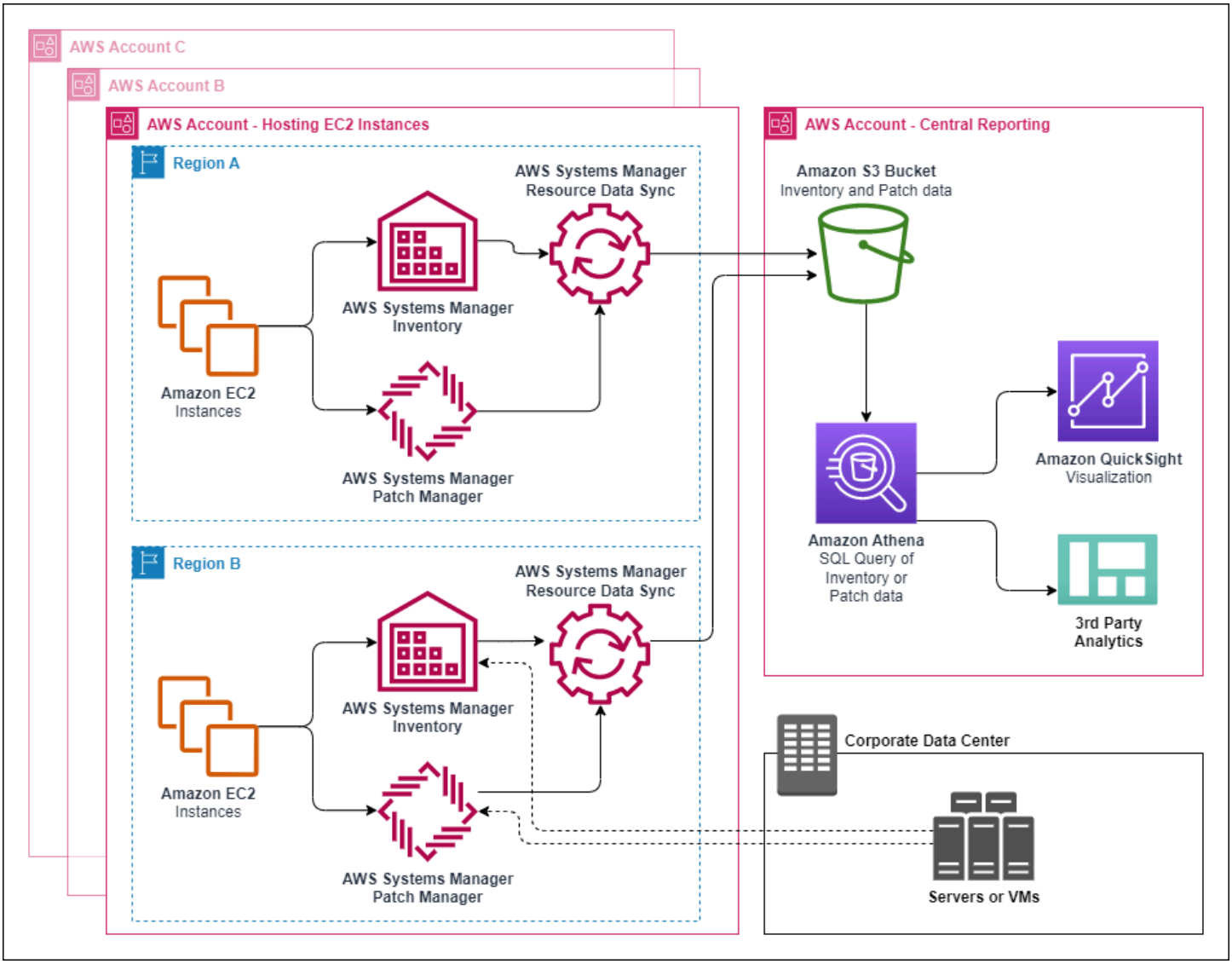
sync then automatically updates the centralized data when new inventory data is collected. With all inventory data stored in a target Amazon S3 bucket, you can use services like Amazon Athena and Amazon QuickSight to query and analyze the aggregated data.

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 an on-premises 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 Amazon EC2 and other machine types in a hybrid and multicloud 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: Using resource data sync to aggregate inventory data.

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 *amzn-s3-demo-bucket* with the name of the S3 bucket you created. Replace *account_ID_number* with a valid AWS account ID number.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SSMBucketPermissionsCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "ssm.amazonaws.com"
            },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
            "Sid": " SSMBucketDelivery",
            "Effect": "Allow",
            "Principal": {
                "Service": "ssm.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*/accountid=111122223333/
*",
                "arn:aws:s3:::amzn-s3-demo-bucket/*/accountid=444455556666/
*",
                "arn:aws:s3:::amzn-s3-demo-bucket/*/accountid=123456789012/
*",
                "arn:aws:s3:::amzn-s3-demo-bucket/*/accountid=777788889999/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:ssm:*:111122223333:resource-
data-sync/*"
                }
            }
        }
    ]
```

```
}
```

5. Save your changes.

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 Using the AWS CLI to configure inventory data collection.

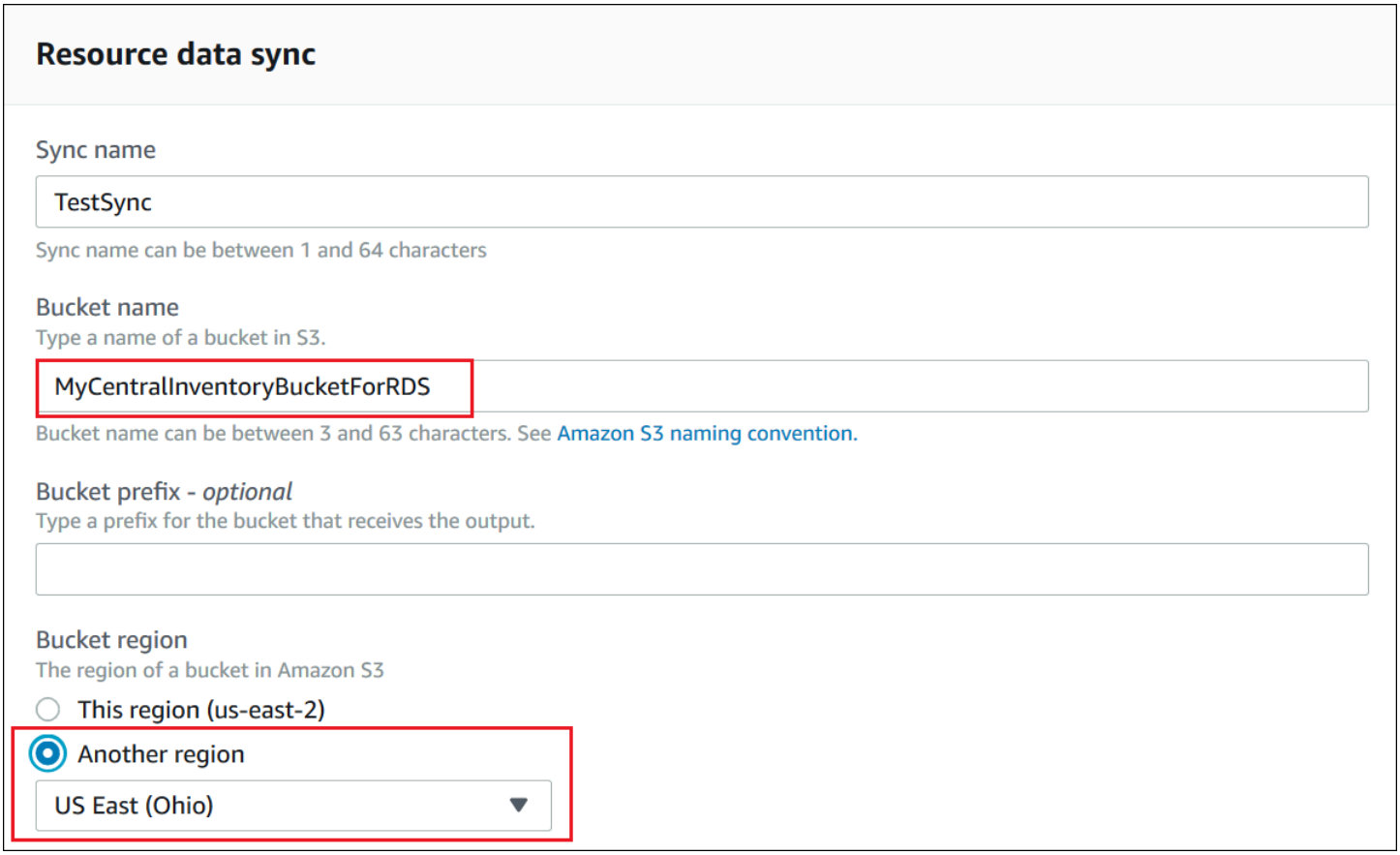
To create a resource data sync

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Fleet Manager**.
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 (subdirectory).
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.



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 *amzn-s3-demo-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.

   JSON

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

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

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 Installing or updating the latest version of the AWS CLI.

2. Run the following command to verify that you don't have any other *AWS Organizations-based* resource data syncs. You can have multiple standard syncs, including multiple standard syncs and an Organizations-based sync. But, you can only have one Organizations-based resource data sync.

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

   If the command returns other Organizations-based resource data sync, you must delete them 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 amzn-s3-demo-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=amzn-s3-demo-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.

Managing resource data syncs

Each AWS account can have 5 resource data syncs per AWS Region. You can use the AWS Systems Manager Fleet Manager console to manage your resource data syncs.

To view resource data syncs

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Fleet Manager**.
3. In the **Account management** dropdown, choose **Resource data syncs**.
4. Select a resource data sync from the table, and then choose **View details** to view information about your resource data sync.

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 **Fleet Manager**.
3. In the **Account management** dropdown, choose **Resource data syncs**.
4. Select a resource data sync from the table, and then choose **Delete**.

Using EventBridge to monitor 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 verison 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.

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 Using the AWS CLI to configure inventory data collection.

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 with the AWS CLI.
- Verify that you have completed the setup requirements for your Amazon Elastic Compute Cloud (Amazon EC2) instances and non-EC2 machines in a hybrid and multicloud environment. For information, see Setting up managed nodes for AWS Systems Manager.
- For Microsoft Windows Server 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 Walkthrough: Using resource data sync to aggregate inventory data.
- (Optional) Create a JSON file to collect custom inventory. For more information, see Working with custom inventory.

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.

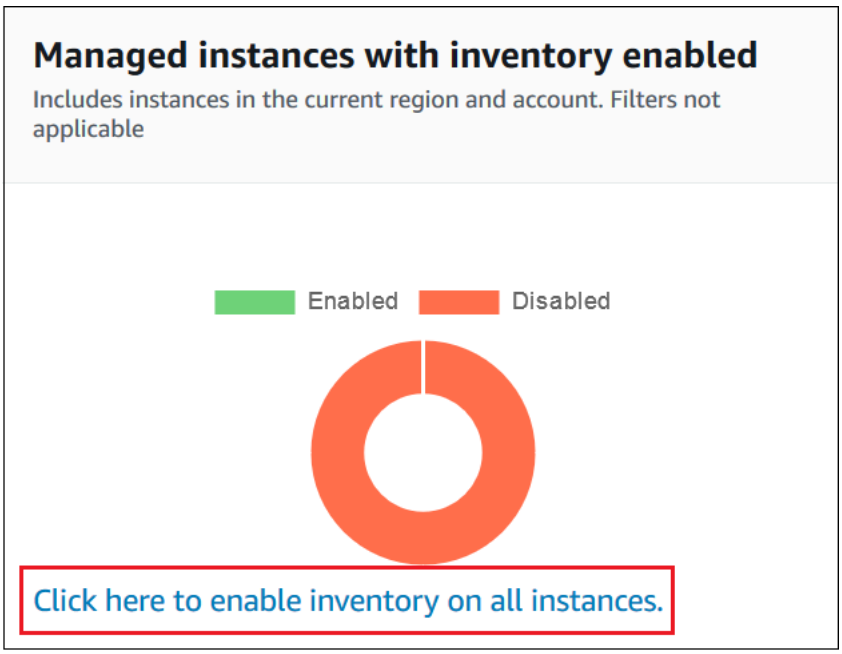
> **Note**
>
> - If a managed node is configured for the global inventory association, and you assign a specific association to that node, then Systems Manager Inventory deprioritizes the global association and applies the specific association.
> - Global inventory associations are available in SSM Agent version 2.0.790.0 or later. For information about how to update SSM Agent on your nodes, see Updating the SSM Agent using Run Command.

Configuring inventory collection with one click (console)

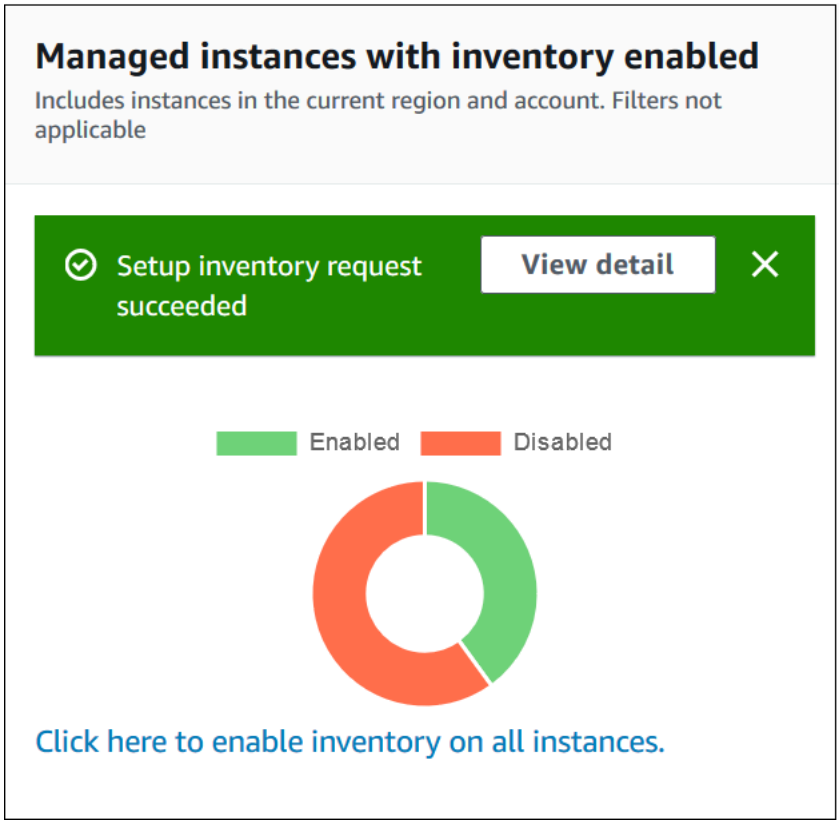
Use the following procedure to configure Systems Manager Inventory for all managed nodes in your AWS account and in a single AWS Region.

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**.
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.

> **Note**
>
> Before completing this procedure, check to see if a global inventory association already exists. If a global inventory association already exists, anytime you launch a new instance, the association will be applied to it, and the new instance will be inventoried.

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**.
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.
   - **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 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. For more information about collecting File and Windows Registry inventory, see Working with file and Windows registry inventory.
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.

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 Walkthrough: Using resource data sync to aggregate inventory data.

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 IAM entity with permission to view the data.

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 entity and the `Amazon-GlueServiceRoleForSSM` service role for AWS KMS encryption.

Topics

- Configuring your IAM entity to access the Detailed View page
- (Optional) Configure permissions for viewing AWS KMS encrypted data

Configuring your IAM entity to access the Detailed View page

The following describes the minimum permissions required to view inventory data on the **Detailed View** page.

The **`AWSQuicksightAthenaAccess`** managed policy

The following `PassRole` and additional required permissions block

JSON

```
{
    "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": [
                "arn:aws:iam::111122223333:role/SSMInventoryGlueRole",
                "arn:aws:iam::111122223333:role/SSMInventoryServiceRole"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "glue.amazonaws.com"
                }
            }
        },
        {
            "Sid": "iamRoleCreation",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:CreateRole",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::111122223333:role/*"
        },
        {
            "Sid": "iamPolicyCreation",
            "Effect": "Allow",
            "Action": "iam:CreatePolicy",
            "Resource": "arn:aws:iam::111122223333:policy/*"
        }
    ]
}
```

(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"
    ]
}
```

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.

- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.

- IAM users:

- Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
- (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

(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 entity and the **Amazon-GlueServiceRoleForSSM** role with `kms:Decrypt` permissions for the AWS KMS key.

Before you begin

To provide the `kms:Decrypt` permissions for the AWS KMS key, add the following policy block to your IAM entity:

```
{
    "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.

   JSON

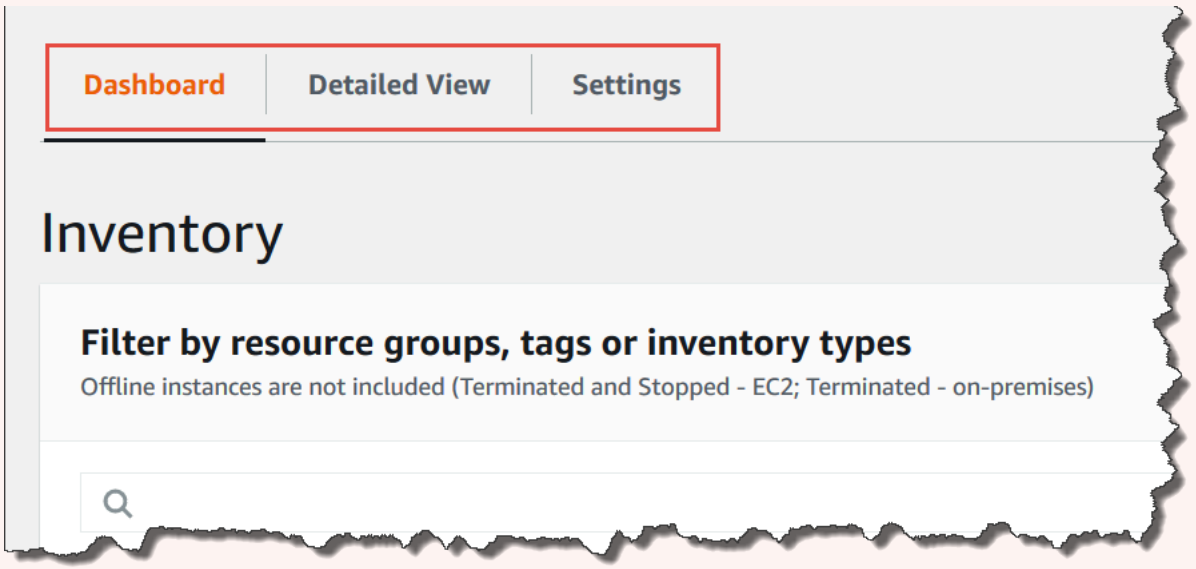
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:111122223333:key/key_ARN"
            ]
        }
    ]
}
```

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

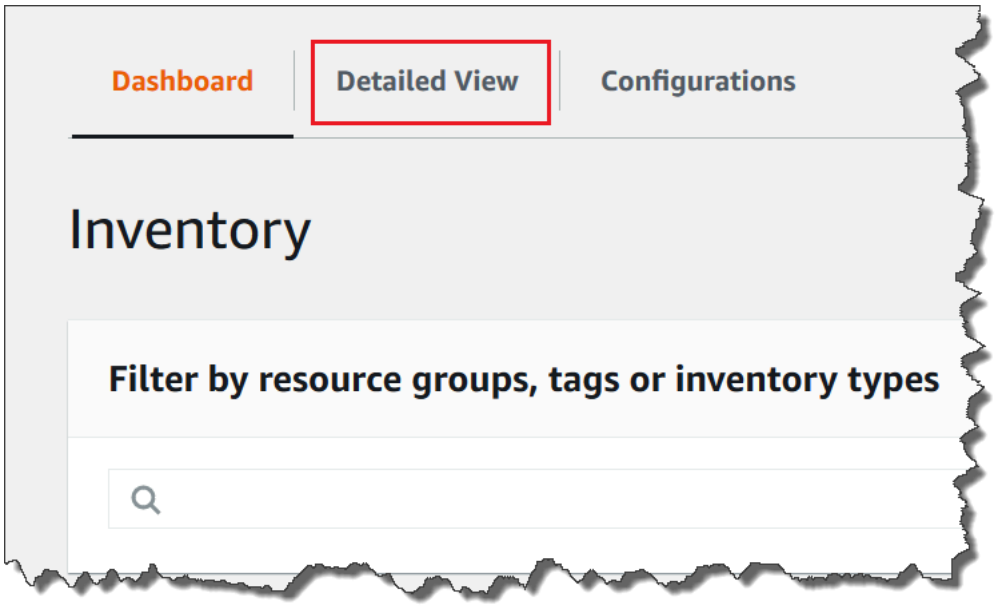
Use the following procedure to view inventory data from multiple AWS Regions and AWS accounts on the Systems Manager Inventory **Detailed View** page.

> **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.

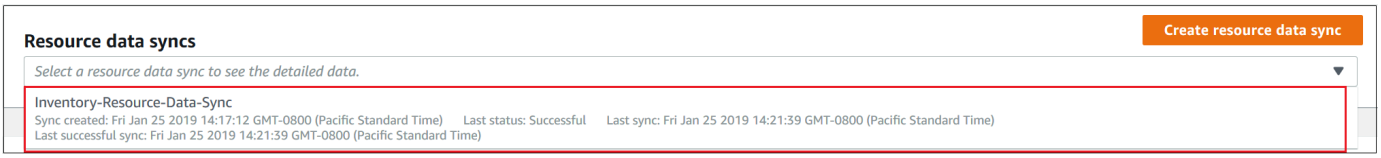


To view inventory data from multiple Regions and accounts in the AWS Systems Manager console

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Inventory**.
3. Choose the **Detailed View** tab.



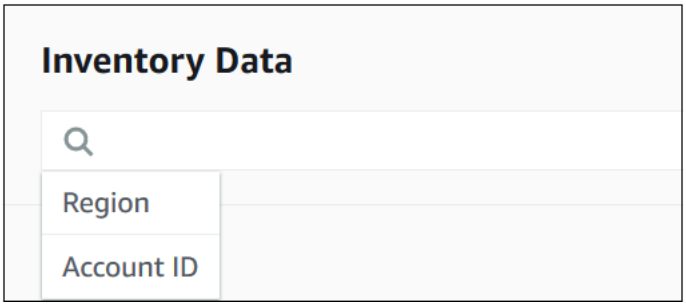
4. Choose the resource data sync for which you want to query data.



5. In the **Inventory Type** list, choose the type of inventory data that you want to query, and then press **Enter**.



6. To filter the data, choose the Filter bar, and then choose a filter option.



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-s3-bucket-name-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**.
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 Server, 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 systems 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 Inventory all managed nodes in your AWS account.

Topics

- Aggregating inventory data to see counts of nodes that collect specific types of data
- Aggregating inventory data with groups to see which nodes are and aren't configured to collect an inventory type

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": [
                {
                    "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": [
                {
                    "DataType": "STRING",
                    "Name": "Name"
                },
                {
                    "DataType": "STRING",
                    "Name": "DisplayName"
                },
                {
                    "DataType": "STRING",
                    "Name": "FeatureType"
                },
                {
                    "DataType": "STRING",
                    "Name": "Installed"
                }
            ]
        }
    ]
}
```

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": "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 Assigning custom inventory metadata to a managed node. 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 examples 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"
      }
 }
```

The value of `TypeName` is limited to 100 characters. Also, the `TypeName` value must begin with the capitalized word `Custom`. For example, `Custom:PuppetModuleInfo`. Therefore, the following examples would result in an exception: `CUSTOM:PuppetModuleInfo`, `custom:PuppetModuleInfo`.

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/*`node-id`*/inventory/custom

Operating system	Path
macOS	`/opt/aws/ssm/data/node-id/inventory/custom`
Windows Server	%SystemDrive%\ProgramData\Amazon\SSM\InstanceData*node-id*\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 Installing or updating the latest version of the AWS CLI.

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.

3. 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.

4. 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
```

5. 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.

1. 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-0ee2d86a2cfc371f6"
   ],
   "detail":{
      "action-status":"failed",
      "action":"put",
      "resource-type":"managed-instance",
      "resource-id":"i-0ee2d86a2cfc371f6",
      "action-reason":"The inventory item with type name Custom:MyInfo was sent with a
 disabled schema version 1.0. You must send a version greater than 1.0",
      "type-name":"Custom:MyInfo"
   }
}
```

Use the following procedure to create 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**.

3. Choose **Create rule**.
4. 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.
5. For **Event bus**, 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 **default**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
6. For **Rule type**, choose **Rule with an event pattern**.
7. Choose **Next**.
8. For **Event source**, choose **AWS events or EventBridge partner events**.
9. In the **Event pattern** section, choose **Event pattern form**.
10. For **Event source**, choose **AWS services**.
11. For **AWS service**, choose **Systems Manager**.
12. For **Event type**, choose **Inventory**.
13. For **Specific detail type(s)**, choose **Inventory Resource State Change**.
14. Choose **Next**.
15. For **Target types**, choose **AWS service**.
16. For **Select a target**, choose **SNS topic**, and then for **Topic**, choose your topic.
17. In the **Additional settings** section, for **Configure target input**, verify that **Matched event** is selected.
18. Choose **Next**.
19. (Optional) Enter one or more tags for the rule. For more information, see Tagging Your Amazon EventBridge Resources in the *Amazon EventBridge User Guide*.
20. Choose **Next**.
21. Review the details of the rule and choose **Create rule**.

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 Learn details about Compliance.

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 Installing or updating the latest version of the AWS CLI.

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
- Deleting an Inventory resource data sync

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**.
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**.

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 tools. 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.

Assigning 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.

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 Installing or updating the latest version of the AWS CLI.
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=[{Rac
 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": [
                {
                    "DataType": "STRING",
                    "Name": "RackLocation"
                }
            ]
        }
    ]
```

```
}
```

Using the AWS CLI to configure inventory data collection

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 Installing or updating the latest version of the AWS CLI.
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,instanceDetailedInfo
```

Windows

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

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 Installing or updating the latest version of the AWS CLI.
2. Run the following command to create a State Manager association that runs Systems Manager Inventory on the node. Replace each *example resource placeholder* with your own information. 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\": \"amzn-s3-demo-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\": \"amzn-s3-demo-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-1a2b3c-1a2b3c4d5e6f7g",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1480544990.06,
        "Date": 1480544990.06,
        "Targets": [
            {
                "Values": [
                   "i-02573cafcfEXAMPLE"
                ],
                "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\": \"amzn-s3-demo-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\": \"amzn-s3-demo-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.

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
```

3. Run the following command to view the association status.

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

The system responds with information like the following.

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

```
            "DocumentVersion": "1"
        }
]
}
```

Walkthrough: Using 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 Walkthrough: Using resource data sync to aggregate inventory data. 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.

> **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 or Using the AWS CLI to configure inventory data collection.
- (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 `Amazon-GlueServiceRoleForSSM` 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](#).

- (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.

  JSON

```
{
    "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-1:123456789012:key/KMS_key_id",
            "Condition": {
                "StringLike": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "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 amzn-s3-demo-bucket and *account-id* with the name of the Amazon S3 bucket you created and a valid AWS account ID. When adding multiple accounts, add an additional condition string and ARN for each account. Remove the additional placeholders from the example when adding one account. Optionally, replace *bucket-prefix* with the name of an Amazon S3 prefix (subdirectory). If you didn't created a prefix, remove *bucket-prefix/* from the ARN in the policy.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": " SSMBucketDelivery",
            "Effect": "Allow",
            "Principal": {
                "Service": "ssm.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/bucket-prefix/*/
accountid=111122223333/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": [
                        "111122223333",
                        "444455556666",
                        "123456789012",
                        "777788889999"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
```

```
                        "arn:aws:ssm:*:111122223333:resource-data-sync/*",
                        "arn:aws:ssm:*:444455556666:resource-data-sync/*",
                        "arn:aws:ssm:*:123456789012:resource-data-sync/*",
                        "arn:aws:ssm:*:777788889999: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 Installing or updating the latest version of the AWS CLI.

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. Replace each *example resource placeholder* with your own information.

   Linux & macOS

```
aws s3 cp ./A_file_in_the_bucket s3://amzn-s3-demo-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://amzn-s3-demo-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=amzn-s3-demo-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=amzn-s3-demo-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=amzn-s3-demo-
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=amzn-s3-demo-
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=amzn-s3-demo-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=amzn-s3-demo-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 amzn-s3-demo-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://amzn-s3-demo-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 amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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.

Topics

- Multiple apply all associations with document 'AWS-GatherSoftwareInventory' are not supported
- Inventory execution status never exits pending

- The AWS-ListWindowsInventory document fails to run
- Console doesn't display Inventory Dashboard | Detailed View | Settings tabs
- UnsupportedAgent
- Skipped
- Failed
- Inventory compliance failed for an Amazon EC2 instance
- S3 bucket object contains old data

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 Alternative configuration for EC2 instance permissions.

At a minimum, the instance profile must have the following IAM permissions.

JSON

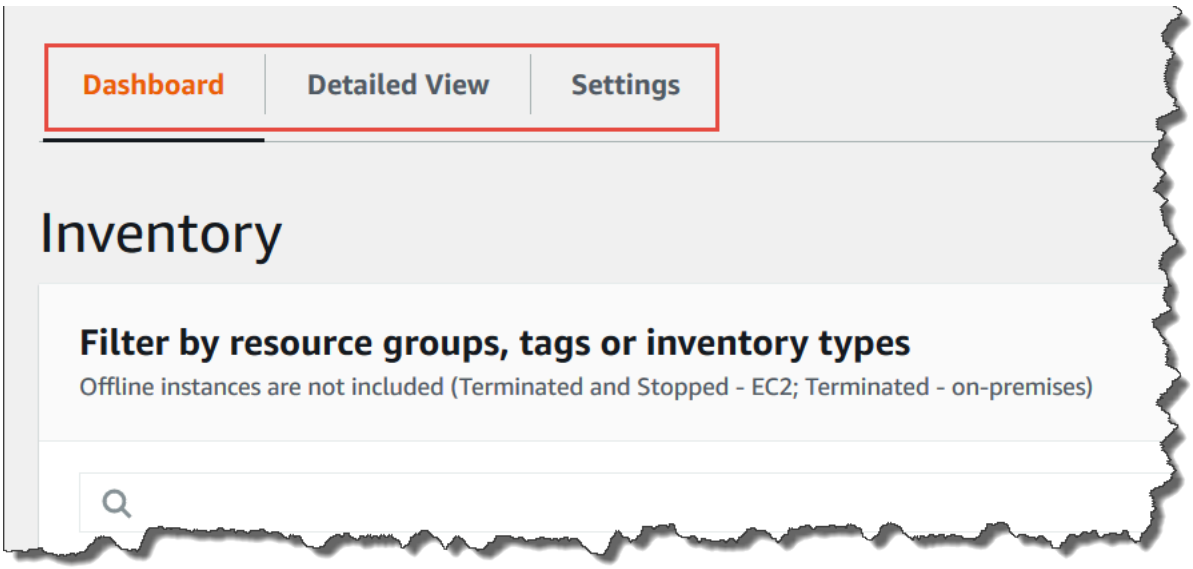
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeAssociation",
                "ssm:ListAssociations",
                "ssm:ListInstanceAssociations",
                "ssm:PutInventory",
                "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.

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 Updating the SSM Agent using Run Command.

Skipped

If the status of the inventory association for a node shows **Skipped**, this means that a higher-priority inventory association is already running on that node. Systems Manager follows a specific priority order when multiple inventory associations could apply to the same managed node.

Inventory association priority order

Systems Manager applies inventory associations in the following priority order:

1. **Quick Setup inventory associations** - Associations created using Quick Setup and the unified console. These associations have names that start with `AWS-QuickSetup-SSM-CollectInventory-` and target all managed nodes.
2. **Explicit inventory associations** - Associations that target specific managed nodes using:
   - Instance IDs

- Tag key-value pairs
- AWS resource groups

3. **Global inventory associations** - Associations that target all managed nodes (using `--targets "Key=InstanceIds,Values=*"`) but were **not** created through Quick Setup.

Common scenarios

Scenario 1: Quick Setup association overrides explicit association

- You have a Quick Setup inventory association targeting all instances
- You create a manual association targeting specific managed nodes by tag
- Result: The manual association shows `Skipped` with detailed status `OverriddenByExplicitInventoryAssociation`
- The Quick Setup association continues to collect inventory from all instances

Scenario 2: Explicit association overrides global association

- You have a global inventory association targeting all instances (not created by Quick Setup)
- You create an association targeting specific instances
- Result: The global association shows `Skipped` for the specifically targeted instances
- The explicit association runs on the targeted instances

Resolution steps

If you want to use your own inventory association instead of Quick Setup:

1. **Identify Quick Setup associations**: In the Systems Manager console, go to State Manager and look for associations with names starting with `AWS-QuickSetup-SSM-CollectInventory-`.
2. **Remove Quick Setup configuration**:
   - Go to Quick Setup in the Systems Manager console.
   - Find your inventory collection configuration.
   - Delete the Quick Setup configuration (this removes the associated inventory association).

> **Note**
>
> You don't need to manually delete the association created by Quick Setup.

3. **Verify your association runs**: After removing the Quick Setup configuration, your explicit inventory association should start running successfully.

If you want to modify existing behavior:

- To view all existing inventory associations, choose **State Manager** in the Systems Manager console and locate associations that use the `AWS-GatherSoftwareInventory` SSM document.
- Remember that each managed node can only have one active inventory association at a time.

> **Important**
>
> - Inventory data is still collected from skipped nodes when their assigned (higher-priority) inventory association runs.
> - Quick Setup inventory associations take precedence over all other types, even those with explicit targeting.
> - The detailed status message `OverriddenByExplicitInventoryAssociation` appears when any association is overridden by a higher-priority one, regardless of the association type.

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
```

Inventory compliance failed for an Amazon EC2 instance

Inventory compliance for an Amazon Elastic Compute Cloud (Amazon EC2) instance can fail if you assign multiple inventory associations to the instance.

To resolve this issue, delete one or more inventory associations assigned to the instance. For more information, see Deleting an association.

> **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.

S3 bucket object contains old data

Data inside the Amazon S3 bucket object is updated when the inventory association is successful and new data is discovered. The Amazon S3 bucket object is updated for each node when the association runs and fails, but the data inside the object is not updated in this case. Data inside the Amazon S3 bucket object will update only when the association runs successfully. When the inventory association fails, you will see old data in the Amazon S3 bucket object.

AWS Systems Manager Patch Manager

Patch Manager, a tool in AWS Systems Manager, automates the process of patching managed nodes with both security-related updates and other types of updates.

> **Note**
>
> Systems Manager provides support for *patch policies* in Quick Setup, a tool in AWS Systems Manager. Using patch policies is the recommended method for configuring your patching operations. Using a single patch policy configuration, you can define patching for all accounts in all Regions in your organization; for only the accounts and Regions you choose; or for a single account-Region pair. For more information, see Patch policy configurations in Quick Setup.

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, on-premises servers, and virtual machines (VMs) by operating system type. This includes supported versions of several operating systems, as listed in Patch Manager prerequisites. 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**.

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 Red Hat Enterprise Linux (RHEL) 7.0 to RHEL 8.0.

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).

How can Patch Manager benefit my organization?

Patch Manager automates the process of patching managed nodes with both security-related updates and other types of updates. It provides several key benefits:

- **Centralized patching control** –Using patch policies, you can set up recurring patching operations for all accounts in all Regions in your organization, specific accounts and Regions, or a single account-Region pair.

- **Flexible patching operations** – You can choose to scan instances to see only a report of missing patches, or scan and automatically install all missing patches.
- **Comprehensive compliance reporting** – After scanning operations, you can view detailed information about which managed nodes are out of patch compliance and which patches are missing.
- **Cross-platform support** – Patch Manager supports multiple operating systems including various Linux distributions, macOS, and Windows Server.
- **Custom patch baselines** – You can define what constitutes patch compliance for your organization through custom patch baselines that specify which patches are approved for installation.
- **Integration with other AWS services** – Patch Manager integrates with AWS Organizations, AWS Security Hub, AWS CloudTrail, and AWS Config for enhanced management and security.
- **Deterministic upgrades** – Support for deterministic upgrades through versioned repositories for operating systems like Amazon Linux 2023.

Who should use Patch Manager?

Patch Manager is designed for the following:

- IT administrators who need to maintain patch compliance across their fleet of managed nodes
- Operations managers who need visibility into patch compliance status across their infrastructure
- Cloud architects who want to implement automated patching solutions at scale
- DevOps engineers who need to integrate patching into their operational workflows
- Organizations with multi-account/multi-Region deployments who need centralized patch management
- Anyone responsible for maintaining the security posture and operational health of AWS managed nodes, edge devices, on-premises servers, and virtual machines

What are the main features of Patch Manager?

Patch Manager offers several key features:

- **Patch policies** – Configure patching operations across multiple AWS accounts and Regions using a single policy through integration with AWS Organizations.

- **Custom patch baselines** – Define rules for auto-approving patches within days of their release, along with approved and rejected patch lists.
- **Multiple patching methods** – Choose from patch policies, maintenance windows, or on-demand "Patch now" operations to meet your specific needs.
- **Compliance reporting** – Generate detailed reports on patch compliance status that can be sent to an Amazon S3 bucket in CSV format.
- **Cross-platform support** – Patch both operating systems and applications across Windows Server, various Linux distributions, and macOS.
- **Scheduling flexibility** – Set different schedules for scanning and installing patches using custom CRON or Rate expressions.
- **Lifecycle hooks** – Run custom scripts before and after patching operations using Systems Manager documents.
- **Security focus** – By default, Patch Manager focuses on security-related updates rather than installing all available patches.
- **Rate control** – Configure concurrency and error thresholds for patching operations to minimize operational impact.

What is compliance in Patch Manager?

The benchmark for what constitutes *patch compliance* for the managed nodes in your Systems Manager fleets is not defined by AWS, by operating system (OS) vendors, or by third parties such as security consulting firms.

Instead, you define what patch compliance means for managed nodes in your organization or account in a *patch baseline*. A patch baseline is a configuration that specifies rules for which patches must be installed on a managed node. A managed node is patch compliant when it is up to date with all the patches that meet the approval criteria that you specify in the patch baseline.

Note that being *compliant* with a patch baseline doesn't mean that a managed node is necessarily *secure*. Compliant means that the patches defined by the patch baseline that are both *available* and *approved* have been installed on the node. The overall security of a managed node is determined by many factors outside the scope of Patch Manager. For more information, see *Security*.

Each patch baseline is a configuration for a specific supported operating system (OS) type, such as Red Hat Enterprise Linux (RHEL), macOS, or Windows Server. A patch baseline can define patching

rules for all supported versions of an OS or be limited to only those you specify, such as RHEL 7.8. and RHEL 9.3.

In a patch baseline, you could specify that all patches of certain classifications and severity levels are approved for installation. For example, you might include all patches classified as `Security` but exclude other classifications, such as `Bugfix` or `Enhancement`. And you could include all patches with a severity of `Critical` and exclude others, such as `Important` and `Moderate`.

You can also define patches explicitly in a patch baseline by adding their IDs to lists of specific patches to approve or reject, such as KB2736693 for Windows Server or `dbus.x86_64:1:1.12.28-1.amzn2023.0.1` for Amazon Linux 2023 (AL2023). You can optionally specify a certain number of days to wait for patching after a patch becomes available. For Linux and macOS, you have the option of specifying an external list of patches for compliance (an Install Override list) instead of those defined by the patch baseline rules.

When a patching operation runs, Patch Manager compares the patches currently applied to a managed node to those that should be applied according to the rules set up in the patch baseline or an Install Override list. You can choose for Patch Manager to show you only a report of missing patches (a `Scan` operation), or you can choose for Patch Manager to automatically install all patches it find are missing from a managed node (a `Scan and install` operation).

Patch Manager provides predefined patch baselines that you can use for your patching operations; however, these predefined configurations are provided as examples and not as recommended best practices. We recommend that you create custom patch baselines of your own to exercise greater control over what constitutes patch compliance for your fleet.

For more information about patch baselines, see the following topics:

- Predefined and custom patch baselines
- Package name formats for approved and rejected patch lists
- Viewing AWS predefined patch baselines
- Working with custom patch baselines
- Working with patch compliance reports

Primary components

Before you start working with the Patch Manager tool, you should familiarize yourself with some major components and features of the tool's patching operations.

Patch baselines

Patch Manager uses *patch baselines*, which include rules for auto-approving patches within days of their release, in addition to optional lists of approved and rejected patches. When a patching operation runs, Patch Manager compares the patches currently applied to a managed node to those that should be applied according to the rules set up in the patch baseline. You can choose for Patch Manager to show you only a report of missing patches (a `Scan` operation), or you can choose for Patch Manager to automatically install all patches it find are missing from a managed node (a `Scan and install` operation).

Patching operation methods

Patch Manager currently offers four methods for running `Scan` and `Scan and install` operations:

- **(Recommended) A patch policy configured in Quick Setup** – Based on integration with AWS Organizations, a single patch policy can define patching schedules and patch baselines for an entire organization, including multiple AWS accounts and all AWS Regions those accounts operate in. A patch policy can also target only some organizational units (OUs) in an organization. You can use a single patch policy to scan and install on different schedules. For more information, see Configure patching for instances in an organization using a Quick Setup patch policy and Patch policy configurations in Quick Setup.
- **A Host Management option configured in Quick Setup** – Host Management configurations are also supported by integration with AWS Organizations, making it possible to run a patching operation for up to an entire Organization. However, this option is limited to scanning for missing patches using the current default patch baseline and providing results in compliance reports. This operation method can't install patches. For more information, see Set up Amazon EC2 host management using Quick Setup.
- **A maintenance window to run a patch `Scan` or `Install` task** – A maintenance window, which you set up in the Systems Manager tool called Maintenance Windows, can be configured to run different types of tasks on a schedule you define. A Run Command-type task can be used to run `Scan` or `Scan and install` tasks a set of managed nodes that you choose. Each maintenance window task can target managed nodes in only a single AWS account-AWS Region pair. For more information, see Tutorial: Create a maintenance window for patching using the console.
- **An on-demand Patch now operation in Patch Manager** – The **Patch now** option lets you bypass schedule setups when you need to patch managed nodes as quickly as possible. Using **Patch now**, you specify whether to run `Scan` or `Scan and install` operation and which managed

nodes to run the operation on. You can also choose to running Systems Manager documents (SSM documents) as lifecycle hooks during the patching operation. Each **Patch now** operation can target managed nodes in only a single AWS account-AWS Region pair. For more information, see Patching managed nodes on demand.

Compliance reporting

After a `Scan` operation, you can use the Systems Manager console to view information about which of your managed nodes are out of patch compliance, and which patches are missing from each of those nodes. You can also generate patch compliance reports in .csv format 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. After a report is generated, you can use a tool like Amazon QuickSight to import and analyze the data. For more information, see Working with patch compliance reports.

> **Note**
>
> A compliance item generated through the use of a patch policy has an execution type of `PatchPolicy`. A compliance item not generated in a patch policy operation has an execution type of `Command`.

Integrations

Patch Manager integrates with the following other AWS services:

- **AWS Identity and Access Management (IAM)** – Use IAM to control which users, groups, and roles have access to Patch Manager operations. For more information, see How AWS Systems Manager works with IAM and Configure instance permissions required for Systems Manager.
- **AWS CloudTrail** – Use CloudTrail to record an auditable history of patching operation events initiated by users, roles, or groups. For more information, see Logging AWS Systems Manager API calls with AWS CloudTrail.
- **AWS Security Hub** – 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.

- **AWS Config** – Set up recording in AWS Config to view Amazon EC2 instance management data in the Patch Manager Dashboard. For more information, see Viewing patch Dashboard summaries.

Topics

- Patch policy configurations in Quick Setup
- Patch Manager prerequisites
- How Patch Manager operations work
- SSM Command documents for patching managed nodes
- Patch baselines
- Using Kernel Live Patching on Amazon Linux 2 managed nodes
- Working with Patch Manager resources and compliance using the console
- Working with Patch Manager resources using the AWS CLI
- AWS Systems Manager Patch Manager tutorials
- Troubleshooting Patch Manager

Patch policy configurations in Quick Setup

AWS recommends the use of *patch policies* to configure patching for your organization and AWS accounts. Patch policies were introduced in Patch Manager in December, 2022.

A patch policy is a configuration you set up using Quick Setup, a tool in AWS Systems Manager. Patch policies provide more extensive and more centralized control over your patching operations than is available with previous methods of configuring patching. Patch policies can be used with all operating systems supported by Patch Manager, including supported versions of Linux, macOS, and Windows Server. For instructions for creating a patch policy, see Configure patching for instances in an organization using a Quick Setup patch policy.

Major features of patch policies

Instead of using other methods of patching your nodes, use a patch policy to take advantage of these major features:

- **Single setup** – Setting up patching operations using a maintenance window or State Manager association can require multiple tasks in different parts of the Systems Manager console. Using a patch policy, all your patching operations can be set up in a single wizard.

- **Multi-account/Multi-Region support** – Using a maintenance window, a State Manager association, or the **Patch now** feature in Patch Manager, you're limited to targeting managed nodes in a single AWS account-AWS Region pair. If you use multiple accounts and multiple Regions, your setup and maintenance tasks can require a great deal of time, because you must perform setup tasks in each account-Region pair. However, if you use AWS Organizations, you can set up one patch policy that applies to all your managed nodes in all AWS Regions in all your AWS accounts. Or, if you choose, a patch policy can apply to only some organizational units (OUs) in the accounts and Regions you choose. A patch policy can also apply to a single local account, if you choose.
- **Installation support at the organizational level** – The existing Host Management configuration option in Quick Setup provides support for a daily scan of your managed nodes for patch compliance. However, this scan is done at a predetermined time and results in patch compliance information only. No patch installations are performed. Using a patch policy, you can specify different schedules for scanning and installing. You can also choose the frequency and time of these operations by using custom CRON or Rate expressions. For example, you could scan for missing patches every day to provide you with regularly updated compliance information. But, your installation schedule could be just once a week to avoid unwanted downtime.
- **Simplified patch baseline selection** – Patch policies still incorporate patch baselines, and there are no changes to the way patch baselines are configured. However, when you create or update a patch policy, you can select the AWS managed or custom baseline you want to use for each operating system (OS) type in a single list. It's not necessary to specify the default baseline for each OS type in separate tasks.

> **Note**
>
> When patching operations based on a patch policy run, they use the `AWS-RunPatchBaseline` SSM document. For more information, see SSM Command document for patching: `AWS-RunPatchBaseline`.

Related information

Centrally deploy patching operations across your AWS Organization using Systems Manager Quick Setup (AWS Cloud Operations and Migrations Blog)

Other differences with patch policies

Here are some other differences to note when using patch policies instead of previous methods of configuring patching:

- **No patch groups required** – In previous patching operations, you could tag multiple nodes to belong to a patch group, and then specify the patch baseline to use for that patch group. If no patch group was defined, Patch Manager patched instances with the current default patch baseline for the OS type. Using patch policies, it's no longer necessary to set up and maintain patch groups.

  > **Note**
  >
  > Patch group functionality is not supported in the console for account-Region pairs that did not already use patch groups before patch policy support was released on December 22, 2022. Patch group functionality is still available in account-Region pairs that began using patch groups before this date.

- **'Configure patching' page removed** – Before the release of patch policies, you could specify defaults for which nodes to patch, a patching schedule, and a patching operation on a **Configure patching** page. This page has been removed from Patch Manager. These options are now specified in patch policies.
- **No 'Patch now' support** – The ability to patch nodes on demand is still limited to a single AWS account-AWS Region pair at a time. For information, see Patching managed nodes on demand.
- **Patch policies and compliance information** – When your managed nodes are scanned for compliance according to a patching policy configuration, compliance data is made available to you. You can view and work with the data in the same way as with other methods of compliance scanning. Although you can set up a patch policy for an entire organization or multiple organizational units, compliance information is reported individually for each AWS account-AWS Region pair. For more information, see Working with patch compliance reports.
- **Association compliance status and patch policies** – The patching status for a managed node that's under a Quick Setup patch policy matches the status of the State Manager association execution for that node. If the association execution status is `Compliant`, the patching status for the managed node is also marked `Compliant`. If the association execution status is `Non-Compliant`, the patching status for the managed node is also marked `Non-Compliant`.

AWS Regions supported for patch policies

Patch policy configurations in Quick Setup are currently supported in the following Regions:

- US East (Ohio) (us-east-2)
- US East (N. Virginia) (us-east-1)
- US West (N. California) (us-west-1)
- US West (Oregon) (us-west-2)
- Asia Pacific (Mumbai) (ap-south-1)
- Asia Pacific (Seoul) (ap-northeast-2)
- Asia Pacific (Singapore) (ap-southeast-1)
- Asia Pacific (Sydney) (ap-southeast-2)
- Asia Pacific (Tokyo) (ap-northeast-1)
- Canada (Central) (ca-central-1)
- Europe (Frankfurt) (eu-central-1)
- Europe (Ireland) (eu-west-1)
- Europe (London) (eu-west-2)
- Europe (Paris) (eu-west-3)
- Europe (Stockholm) (eu-north-1)
- South America (São Paulo) (sa-east-1)

Patch Manager prerequisites

Make sure that you have met the required prerequisites before using Patch Manager, a tool in AWS Systems Manager.

Topics

- SSM Agent version
- Python version
- Connectivity to the patch source
- S3 endpoint access
- Permissions to install patches locally
- Supported operating systems for Patch 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 tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. Subscribe to the SSM Agent Release Notes page on GitHub to get notifications about SSM Agent updates.

Python version

For macOS and most Linux operating systems (OSs), Patch Manager currently supports Python versions 2.6 - 3.10. The AlmaLinux, Debian Server, and Ubuntu Server OSs require a supported version of Python 3 (3.0 - 3.10).

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.

Windows Server: Ensure connectivity to Windows Update Catalog or Windows Server Update Services (WSUS)

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.

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 and Improve the security of EC2 instances by using VPC endpoints for Systems Manager.

Permissions to install patches locally

On Windows Server and Linux operating systems, Patch Manager assumes the Administrator and root user accounts, respectively, to install patches.

On macOS, however, for Brew and Brew Cask, Homebrew doesn't support its commands running under the root user account. 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. Therefore, in order to install patches, the owner of the `homebrew` directory also needs recursive owner permissions for the `/usr/local` directory.

Tip

The following command provides this permission for the specified user:

```
sudo chown -R $USER:admin /usr/local
```

Supported operating systems for Patch Manager

The Patch Manager tool might not support all the same operating systems versions that are supported by other Systems Manager tools. (For the full list of Systems Manager-supported operating systems, see Supported operating systems for Systems Manager.) 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.

Note

Patch Manager relies on the patch repositories that are configured on a managed node, such as Windows Update Catalog and Windows Server Update Services for Windows, to retrieve available patches to install. Therefore, for end of life (EOL) operating system versions, if no new updates are available, Patch Manager might not be able to report on the

new updates. This can be because no new updates are released by the Linux distribution maintainer, Microsoft, or Apple, or because the managed node does not have the proper license to access the new updates.
Patch Manager reports compliance status against the available patches on the managed node. Therefore, if an instance is running an EOL operating system, and no updates are available, Patch Manager might report the node as Compliant, depending on the patch baselines configured for the patching operation.

Operating system	Details
Linux	• AlmaLinux 8.*x*, 9.*x* • Amazon Linux 2 version 2.0 and all later versions • Amazon Linux 2023 • Debian Server 11.*x*, and 12.*x* • Oracle Linux 7.5–8.*x*, 9.*x* • Red Hat Enterprise Linux (RHEL) 7.x–8.*x*, 9.*x*, 10.*x* • Rocky Linux 8.*x*, 9.*x* • Ubuntu Server 16.04 LTS, 18.04 LTS, 20.04 LTS, 22.04 LTS, and 25.04
macOS	*macOS is supported for Amazon EC2 instances only.* 13.0–13.7 (Ventura) 14.*x* (Sonoma) 15.*x* (Sequoia) **macOS OS updates** Patch Manager doesn't support operating system (OS) updates or upgrades for macOS, such as from 13.1 to 13.2. To

Operating system	Details
	perform OS version updates on macOS, we recommend using Apple's built-in OS upgrade mechanisms. For more information, see Device Management on the Apple Developer Documentation website. **Homebrew support** Patch Manager requires Homebrew, the open-source software package management system, to be installed at either of the following default install locations: • `/usr/local/bin` • `/opt/homebrew/bin` Patching operations using Patch Manager fail to function correctly when Homebrew is not installed. **Region support** macOS is not supported in all AWS Regions. For more information about Amazon EC2 support for macOS, see Amazon EC2 Mac instances in the *Amazon EC2 User Guide*. **macOS edge devices** SSM Agent for AWS IoT Greengrass core devices is not supported on macOS. You can't use Patch Manager to patch macOS edge devices.

Operating system	Details
Windows	Windows Server 2012 through Windows Server 2025, including R2 versions. **Note** SSM Agent for AWS IoT Greengras s core devices is not supported on Windows 10. You can't use Patch Manager to patch Windows 10 edge devices. **Windows Server 2012 and 2012 R2 support** Windows Server 2012 and 2012 R2 reached end of support on October 10, 2023. To use Patch Manager with these versions, we recommend also using Extended Security Updates (ESUs) from Microsoft. For more information, see Windows Server 2012 and 2012 R2 reaching end of support on the Microsoft website.

How Patch Manager operations work

This section provides technical details that explain how Patch Manager, a tool in 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.

Note

The information in the following topics applies no matter which method or type of configuration you are using for your patching operations:

- A patch policy configured in Quick Setup
- A Host Management option configured in Quick Setup
- A maintenance window to run a patch `Scan` or `Install` task
- An on-demand **Patch now** operation

Topics

- How package release dates and update dates are calculated
- How security patches are selected
- How to specify an alternative patch source repository (Linux)
- How patches are installed
- How patch baseline rules work on Linux-based systems
- Patching operation differences between Linux and Windows Server

How package release dates and update dates are calculated

Important

The information on this page applies to the Amazon Linux 2 and Amazon Linux 2023 operating systems (OSs) for Amazon Elastic Compute Cloud (Amazon EC2) instances. The packages for these OS types are created and maintained by Amazon Web Services. How the manufacturers of other operating systems manage their packages and repositories affect how their release dates and update dates are calculated. For OSs besides Amazon Linux 2 and Amazon Linux 2023, such as Red Hat Enterprise Linux, refer to the manufacturer's documentation for information about how their packages are updated and maintained.

In the settings for custom patch baselines you create, for most OS types, you can specify that patches are auto-approved for installation after a certain number of days. AWS provides several predefined patch baselines that include auto-approval dates of 7 days.

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, you create a rule using the `CriticalUpdates` classification and configure it for 7 days auto-approval delay. As a result, a new critical patch with a release date or last update date of July 7 is automatically approved on July 14.

To avoid unexpected results with auto-approval delays on Amazon Linux 2 and Amazon Linux 2023, it's important to understand how their release dates and update dates are calculated.

In most cases, the auto-approval wait time before patches are installed is calculated from an `Updated Date` value in `updateinfo.xml`, not a `Release Date` value. The following are important details about these date calculations:

- The `Release Date` is the date a *notice* is released. It does not mean the package is necessarily available in the associated repositories yet.
- The `Update Date` is the last date the notice was updated. An update to a notice can represent something as small as a text or description update. It does not mean the package was released from that date or is necessarily available in the associated repositories yet.

  This means that a package could have an `Update Date` value of July 7 but not be available for installation until (for example) July 13. Suppose for this case that a patch baseline that specifies a 7-day auto-approval delay runs in an `Install` operation on July 14. Because the `Update Date` value is 7 days prior to the run date, the patches and updates in the package are installed on July 14. The installation happens even though only 1 day has passed since the package became available for actual installation.
- A package containing operating system or application patches can be updated more than once after initial release.
- A package can be released into the AWS managed repositories but then rolled back if issues are later discovered with it.

In some patching operations, these factors might not be important. For example, if a patch baseline is configured to install a patch with severity values of `Low` and `Medium`, and a classification of `Recommended`, any auto-approval delay might have little impact on your operations.

However, in cases where the timing of critical or high-severity patches is more important, you might want to exercise more control over when patches are installed. The recommended method

for doing this is to use alternative patch source repositories instead of the default repositories for patching operations on a 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 more information, see How to specify an alternative patch source repository (Linux).

How security patches are selected

The primary focus of Patch Manager, a tool in 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.

By default, Patch Manager doesn't replace a package that has been marked as obsolete in a package repository with any differently named replacement packages unless this replacement is required by the installation of another package update. Instead, for commands that update a package, Patch Manager only reports and installs missing updates for the package that is installed but obsolete. This is because replacing an obsolete package typically requires uninstalling an existing package and installing its replacement. Replacing the obsolete package could introduce breaking changes or additional functionality that you didn't intend.

This behavior is consistent with YUM and DNF's `update-minimal` command, which focuses on security updates rather than feature upgrades. For more information, see the section called “How patches are installed”.

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).

The remainder of this section explains how Patch Manager selects security patches for the different supported operating systems.

Amazon Linux 2 and Amazon Linux 2023

Preconfigured repositories are handled differently on Amazon Linux 2 than on Amazon Linux 2023.

On 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:

On Amazon Linux 2

- **Repo ID**: `amzn2-core/2/`*`architecture`*

  Repo name: `Amazon Linux 2 core repository`
- **Repo ID**: `amzn2extra-docker/2/`*`architecture`*

  Repo name: `Amazon Extras repo for docker`

> **Note**
>
> *`architecture`* can be x86_64 or (for Graviton processors) aarch64.

When you create an Amazon Linux 2023 (AL2023) instance, it contains the updates that were available in the version of AL2023 and the specific AMI you selected. Your AL2023 instance doesn't automatically receive additional critical and important security updates at launch time. Instead, with the *deterministic upgrades through versioned repositories* feature supported for AL2023, which is turned on by default, you can apply updates based on a schedule that meets your specific needs. For more information, see Deterministic upgrades through versioned repositories in the *Amazon Linux 2023 User Guide*.

On AL2023, the preconfigured repository is the following:

- **Repo ID**: `amazonlinux`

  Repo name: Amazon Linux 2023 repository

On Amazon Linux 2023 (preview release), the preconfigured repositories are tied to *locked versions* of package updates. When new Amazon Machine Images (AMIs) for AL2023 are released, they are locked to a specific version. For patch updates, Patch Manager retrieves the latest locked version of the patch update repository and then updates packages on the managed node based on the content of that locked version.

Package managers

Amazon Linux 2 managed nodes use Yum as the package manager. Amazon Linux 2023 use DNF as the package manager.

Both package managers use the concept of an *update notice* as a file named `updateinfo.xml`. An update notice is simply a collection of packages that fix specific problems. All packages that are in an update notice are considered Security by Patch Manager. Individual packages 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 **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.
> For more information about the **Include non-security updates** option, see How patches are installed and How patch baseline rules work on Linux-based systems.

Debian Server

On Debian Server, the Systems Manager patch baseline service uses preconfigured repositories (repos) on the instance. These preconfigured repos are used to pull an updated list of available 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 each version of Debian Server, Patch Manager only identifies upgrades that are part of the associated repo for that version, as follows:

- Debian Server 11: `debian-security bullseye`
- Debian Server 12: `debian-security bookworm`

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**: `ol7_UEKR5/x86_64`

  Repo name: `Latest Unbreakable Enterprise Kernel Release 5 for Oracle Linux 7Server (x86_64)`
- **Repo ID**: `ol7_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)`

Oracle Linux 9:

- **Repo ID**: `ol9_baseos_latest`

  Repo name: `Oracle Linux 9 BaseOS Latest (x86_64)`
- **Repo ID**: `ol9_appstream`

  Repo name: `Oracle Linux 9 Application Stream Packages(x86_64)`

- **Repo ID**: `ol9_UEKR7`

  Repo name: `Oracle Linux UEK Release 7 (x86_64)`

> **Note**
>
> All updates are downloaded from the remote repos configured on the managed node. Therefore, the node must have outbound access to the internet in order 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 **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.

AlmaLinux, RHEL, and Rocky Linux

On AlmaLinux, Red Hat Enterprise Linux, 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 have outbound access to the internet in order to connect to the repos so the patching can be performed.

> **Note**
>
> If you select the **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. AlmaLinux, Red Hat Enterprise Linux 8, 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.

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)`

AlmaLinux, 8 RHEL 8, and Rocky Linux 8

- **Repo ID**: `rhel-8-appstream-rhui-rpms`

  Repo name: `Red Hat Enterprise Linux 8 for x86_64 - AppStream from RHUI (RPMs)`
- **Repo ID**: `rhel-8-baseos-rhui-rpms`

  Repo name: `Red Hat Enterprise Linux 8 for x86_64 - BaseOS from RHUI (RPMs)`
- **Repo ID**: `rhui-client-config-server-8`

  Repo name: `Red Hat Update Infrastructure 3 Client Configuration Server 8`

AlmaLinux 9, RHEL 9, and Rocky Linux 9

- **Repo ID**: `rhel-9-appstream-rhui-rpms`

  Repo name: `Red Hat Enterprise Linux 9 for x86_64 - AppStream from RHUI (RPMs)`
- **Repo ID**: `rhel-9-baseos-rhui-rpms`

  Repo name: `Red Hat Enterprise Linux 9 for x86_64 - BaseOS from RHUI (RPMs)`
- **Repo ID**: `rhui-client-config-server-9`

  Repo name: `Red Hat Enterprise Linux 9 Client Configuration`

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 16.04 LTS: `xenial-security`
- Ubuntu Server 18.04 LTS: `bionic-security`
- Ubuntu Server 20.04 LTS: `focal-security`
-
- Ubuntu Server 22.04 LTS (`jammy-security`)
-
-
- Ubuntu Server 24.04 LTS (`noble-security`)
-
- Ubuntu Server 25.04 (`plucky-security`)

Windows Server

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).

> **Note**
>
> Patch Manager only makes available patches for Windows Server operating system versions that are supported for Patch Manager. For example, Patch Manager can't be used to patch Windows RT.

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.

Similarly, Patch Manager can install only patches that are available on the managed node during the time of the patching operation. By default, Windows Server 2019 and Windows Server 2022 remove updates that are replaced by later updates. As a result, if you use the `ApproveUntilDate` parameter in a Windows Server patch baseline, but the date selected in

the `ApproveUntilDate` parameter is *before* the date of the latest patch, then the following scenario occurs:

- The superseded patch is removed from the node and therefore can't be installed using Patch Manager.
- The latest, replacement patch is present on the node but not yet approved for installation per the `ApproveUntilDate` parameter's specified date.

This means that the managed node is compliant in terms of Systems Manager operations, even though a critical patch from the previous month might not be installed. This same scenario can occur when using the `ApproveAfterDays` parameter. Because of the Microsoft superseded patch behavior, it is possible to set a number (generally greater than 30 days) so that patches for Windows Server are never installed if the latest available patch from Microsoft is released before the number of days in `ApproveAfterDays` has elapsed. Note that this system behavior doesn't apply if you have modified your Windows Group Policy Object (GPO) settings to make the superseded patch available on your managed nodes.

> **Note**
>
> In some cases, Microsoft releases patches for applications that don't specify 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 tool in 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.

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 25.04 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 later in this topic.

For information about default and custom patch baselines, see Predefined and custom patch baselines.

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 for Linux.

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.

Topics

- Important considerations for alternative repositories
- Sample uses for alternative patch source repositories

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 2, or Red Hat Enterprise Linux, 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 **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 supported Amazon Linux versions

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 tool in 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 2 the `yum` package manager is used.

Patch Manager doesn't install a new package that replaces an obsolete package that's currently installed. (Exceptions: The new package is a dependency of another package updating being installed, or the new package has the same name as the obsolete package.) Instead, Patch Manager reports on and installs available updates to installed packages. This approach helps prevent unexpected changes to your system functionality that might occur when one package replaces another.

If you need to uninstall a package that has been made obsolete and install its replacement, you might need to use a custom script or use package manager commands outside of Patch Manager's standard operations.

The remainder of this section explains how Patch Manager installs patches on an operating system.

Amazon Linux 2 and Amazon Linux 2023

On Amazon Linux 2 and Amazon Linux 2023 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 (Amazon Linux 2) or the DNF update API (Amazon Linux 2023) is applied to approved patches as follows:
   - For predefined default patch baselines provided by AWS, only patches specified in `updateinfo.xml` are applied (security updates only). This is because the **Include**

nonsecurity updates check box is not selected. The predefined baselines are equivalent to a custom baseline with the following:

- The **Include nonsecurity updates** check box is not selected
- A SEVERITY list of `[Critical, Important]`
- A CLASSIFICATION list of `[Security, Bugfix]`

For Amazon Linux 2, the equivalent yum command for this workflow is:

```
sudo yum update-minimal --sec-severity=Critical,Important --bugfix -y
```

For Amazon Linux 2023, the equivalent dnf command for this workflow is:

```
sudo dnf upgrade-minimal --sec-severity=Critical --sec-severity=Important --
bugfix -y
```

If the **Include nonsecurity updates** check box is selected, patches in `updateinfo.xml` and those not in `updateinfo.xml` are all applied (security and nonsecurity updates).

For Amazon Linux 2, if a baseline with **Include nonsecurity updates** is selected, has a SEVERITY list of `[Critical, Important]` and a CLASSIFICATION list of `[Security, Bugfix]`, the equivalent yum command is:

```
sudo yum update --security --sec-severity=Critical,Important --bugfix -y
```

For Amazon Linux 2023, the equivalent dnf command is:

```
sudo dnf upgrade --security --sec-severity=Critical --sec-severity=Important --
bugfix -y
```

> **Note**
>
> New packages that replace now-obsolete packages with different names are installed if you run these `yum` or `dnf` commands outside of Patch Manager. However, they are *not* installed by the equivalent Patch Manager operations.

Additional patching details for Amazon Linux 2023

Support for severity level 'None'

Amazon Linux 2023 also supports the patch severity level `None`, which is recognized by the DNF package manager.

Support for severity level 'Medium'

For Amazon Linux 2023, a patch severity level of `Medium` is equivalent to a severity of `Moderate` that might be defined in some external repositories. If you include `Medium` severity patches in the patch baseline, `Moderate` severity patches from external patches are also installed on the instances.

When you query for compliance data using the API action DescribeInstancePatches, filtering for the severity level `Medium` reports patches with severity levels of both `Medium` and `Moderate`.

Transitive dependency handling for Amazon Linux 2023

For Amazon Linux 2023, Patch Manager might install different versions of transitive dependencies than the equivalent `dnf` commands install. Transitive dependencies are packages that are automatically installed to satisfy the requirements of other packages (dependencies of dependencies).

For example, `dnf upgrade-minimal --security` installs the *minimal* versions of transitive dependencies needed to resolve known security issues, while Patch Manager installs the latest available versions of the same transitive dependencies.

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`.)

Debian Server

On Debian Server 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.)
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, 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.

> **Note**
>
> Patch Manager does not support using the APT `Pin-Priority` option to assign priorities to packages. Patch Manager aggregates available updates from all enabled repositories and selects the most recent update that matches the baseline for each installed package.

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`.)

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 **Include non-security updates** check box is *not* selected, only security updates are applied.
   - For custom patch baselines where the **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`.)

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 **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 **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 and 9 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 **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
  ```

  Note

  For Oracle Linux, Patch Manager might install different versions of transitive dependencies than the equivalent `dnf` commands install. Transitive dependencies are packages that are automatically installed to satisfy the requirements of other packages (dependencies of dependencies).
  For example, `dnf upgrade-minimal --security` installs the *minimal* versions of transitive dependencies needed to resolve known security issues, while Patch Manager installs the *latest available versions* of the same transitive dependencies.

- For custom patch baselines where the **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
  ```

Note

New packages that replace now-obsolete packages with different names are installed if you run these `yum` or `dnf` commands outside of Patch Manager. However, they are *not* installed by the equivalent Patch Manager operations.

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`.)

AlmaLinux, RHEL, and Rocky Linux

On AlmaLinux, Red Hat Enterprise Linux, 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 AlmaLinux 8 and 9, RHEL 8, 9, and 10, and Rocky Linux 8 and 9) is applied to approved patches according to the following rules:

Scenario 1: Non-security updates excluded

- **Applies to**: Predefined default patch baselines provided by AWS and custom patch baselines.
- **Include non-security updates** check box: *Not* selected.
- **Patches applied**: Patches specified in `updateinfo.xml` (security updates only) are applied *only* if they both match the patch baseline configuration and are found in the configured repos.

  In some cases, a patch specified in `updateinfo.xml` might no longer be available in a configured repo. Configured repos usually have only the latest version of a patch, which is a cumulative roll-up of all prior updates, but the latest version might not match the patch baseline rules and is omitted from the patching operation.
- **Commands**: For RHEL 7, the equivalent yum command for this workflow is:

  ```
  sudo yum update-minimal --sec-severity=Critical,Important --bugfix -y
  ```

  For AlmaLinux, RHEL 8, 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
  ```

  Note

  For AlmaLinux, RHEL, and Rocky LinuxRocky Linux, Patch Manager might install different versions of transitive dependencies than the equivalent `dnf` commands install. Transitive dependencies are packages that are automatically installed to satisfy the requirements of other packages (dependencies of dependencies). For example, `dnf upgrade-minimal --security` installs the *minimal* versions of transitive dependencies needed to resolve known security issues, while Patch Manager installs the *latest available versions* of the same transitive dependencies.

Scenario 2: Non-security updates included

- **Apples to**: Custom patch baselines.
- **Include non-security updates** check box: Selected.
- **Patches applied**: Patches in `updateinfo.xml` *and* those not in `updateinfo.xml` are applied (security and nonsecurity updates).
- **Commands**: For RHEL 7, the equivalent yum command for this workflow is:

```
sudo yum update --security --bugfix -y
```

For AlmaLinux 8 and 9, RHEL 8 and 9, and Rocky Linux 8 and 9, the equivalent dnf command for this workflow is:

```
sudo dnf update --security --bugfix -y
```

> **Note**
>
> New packages that replace now-obsolete packages with different names are installed if you run these `yum` or `dnf` commands outside of Patch Manager. However, they are *not* installed by the equivalent Patch Manager operations.

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`.)

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 each version of Ubuntu Server, patch candidate versions are limited to patches that are part of the associated repo for that version, as follows:
>
> - Ubuntu Server 16.04 LTS: `xenial-security`
> - Ubuntu Server 18.04 LTS: `bionic-security`
> - Ubuntu Server 20.04 LTS): `focal-security`
> - Ubuntu Server 22.04 LTS: `jammy-security`
> - Ubuntu Server 24.04 LTS (`noble-security`)

- Ubuntu Server 25.04 (`plucky-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.

> **Note**
>
> Patch Manager does not support using the APT `Pin-Priority` option to assign priorities to packages. Patch Manager aggregates available updates from all enabled repositories and selects the most recent update that matches the baseline for each installed package.

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`.)

Windows Server

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. Windows allows only the latest available version of a KB to be installed. Patch Manager installs the latest version of a KB when it, or any previous version of the KB, matches the applied patch baseline. 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`.)
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 KBs, 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 KBs from Microsoft's Windows Update site because Patch Manager uses the Windows Update API to drive the download and installation of KBs. 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 KB repository and configure your managed nodes to target that WSUS server using Group Policies.
>
> Patch Manager might reference KB IDs when creating custom patch baselines for Windows Server, such as when an **Approved patches** list or **Rejected patches** list is included the the baseline configuration. Only updates that are assigned a KB ID in Microsoft Windows Update or a WSUS server are installed by Patch Manager. Updates that lack a KB ID are not included in patching operations.
>
> For information about creating custom patch baselines, see the following topics:
>
> - Creating a custom patch baseline for Windows Server
> - Create a patch baseline (CLI)
> - Package name formats for Windows operating systems

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 tool in 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 2 and Amazon Linux 2023
- How patch baseline rules work on Debian Server
- How patch baseline rules work on macOS
- How patch baseline rules work on Oracle Linux
- How patch baseline rules work on AlmaLinux, RHEL, and Rocky Linux
- How patch baseline rules work on Ubuntu Server

How patch baseline rules work on Amazon Linux 2 and Amazon Linux 2023

> **Note**
>
> Amazon Linux 2023 (AL2023) uses versioned repositories that can be locked to a specific version through one or more system settings. For all patching operations on AL2023 EC2 instances, Patch Manager uses the latest repository versions, independent of the system configuration. For more information, see Deterministic upgrades through versioned repositories in the *Amazon Linux 2023 User Guide*.

On Amazon Linux 2 and Amazon Linux 2023, the patch selection process is as follows:

1. On the managed node, the YUM library (Amazon Linux 2) or the DNF library (Amazon Linux 2023) accesses the `updateinfo.xml` file for each configured repo.

   If no `updateinfo.xml` file is found, whether patches are installed depend on settings for **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 PatchFilter data type. Denotes the type of package included in the update notice. You can view the list of supported values by using the AWS CLI command **describe-patch-properties** or the API 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 packages included in the update notice. Usually only applicable for *Security* update notices. You can view the list of supported values by using the AWS CLI command **describe-patch-properties** or the API 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.

Attribute	Description
update_id	Denotes the advisory ID, such as *ALAS-2017-867*. The advisory ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.
references	Contains additional information about the update notice, such as a CVE ID (format: *CVE-2017-1234567*). The CVE ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.
updated	Corresponds to ApproveAfterDays in the patch baseline. Denotes the released date (updated date) of the packages included in the update notice. A comparison between the current timestamp and the value of this attribute plus the `ApproveAfterDays` is used to determine if the patch is approved for deployment.

For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.

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 **Include non-security updates** check box is *not* selected	For each update notice in `updateinfo.xml` , the patch baseline is used as a filter, allowing only the qualified packages to be included in the update. If multiple packages are applicable after applying the patch baseline definition, the latest version is used.

Security option	Patch selection

	For Amazon Linux 2, the equivalent yum command for this workflow is:
	`sudo yum update-minimal --sec-severity=Critical,Important --bugfix -y`
	For Amazon Linux 2023, the equivalent dnf command for this workflow is:
	`sudo dnf upgrade-minimal --sec-severity=Critical --sec-severity=Important --bugfix -y`

Security option	Patch selection

Custom patch baselines where the **Include non-security updates** check box *is* selected with a SEVERITY list of `[Critical, Important]` and a CLASSIFICATION list of `[Security, Bugfix]`	In addition to applying the security updates that were selected from `updateinfo.xml` , Patch Manager applies nonsecurity updates that otherwise meet the patch filtering rules.
	
	For Amazon Linux 2, the equivalent yum command for this workflow is:
	
	`sudo yum update --security --sec-severity=Critical,Important --bugfix -y`
	
	For Amazon Linux 2023, the equivalent dnf command for this workflow is:
	
	`sudo dnf upgrade --security --sec-severity=Critical --sec-severity=Important --bugfix -y`
	
	Note
	New packages that replace now-obsolete packages with different names are installed if you run these `yum` or `dnf` commands outside of Patch Manager. However, they are *not* installed by the equivalent Patch Manager operations.

For information about patch compliance status values, see Patch compliance state values.

How patch baseline rules work on Debian Server

On Debian Server , the patch baseline service offers filtering on the *Priority* and *Section* fields. These fields are typically present for all Debian Server packages. To determine whether a patch is selected by the patch baseline, Patch Manager does the following:

1. On Debian 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 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 11: `debian-security bullseye`
   - Debian Server 12: `debian-security bookworm`

   If nonsecurity updates are included, patches from other repositories are considered as well.

   For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.

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 Patch compliance state values.

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.
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 **Include non-security updates** check box is *not* 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 **Include non-security updates** check box *is* selected	In addition to applying the security updates that were identified by using `InstallHistory.plist`, Patch Manager applies

Security option	Patch selection
	nonsecurity updates that otherwise meet the patch filtering rules.

For information about patch compliance status values, see Patch compliance state values.

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 **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 PatchFilter data type. Denotes the type of package included in the update notice. You can view the list of supported values by using the AWS CLI command **describe-patch-properties** or the API operation **DescribePatchProperties**. You can also view the list in the **Approval rules** area of the **Create patch baseline** page or **Edit**

Attribute	Description
	patch baseline page in the Systems Manager console.
severity	Corresponds to the value of the Severity key attribute in the patch baseline's PatchFilter data type. Denotes the severity of the packages included in the update notice. Usually only applicable for *Security* update notices. You can view the list of supported values by using the AWS CLI command **describe-patch-properties** or the API 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.
update_id	Denotes the advisory ID, such as *CVE-2019-17055*. The advisory ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.
references	Contains additional information about the update notice, such as a CVE ID (format: *CVE-2019-17055*) or a Bugzilla ID (format: *1463241*). The CVE ID and Bugzilla ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.

Attribute	Description
updated	Corresponds to ApproveAfterDays in the patch baseline. Denotes the released date (updated date) of the packages included in the update notice. A comparison between the current timestamp and the value of this attribute plus the `ApproveAfterDays` is used to determine if the patch is approved for deployment.

For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.

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 **Include non-security updates** check box is *not* selected	For each update notice in `updateinfo.xml` , the patch baseline is used as a filter, allowing only the qualified packages to be included in the update. If multiple packages are applicable after applying the patch baseline definition, the latest version is used. For version 7 managed nodes, the equivalent yum command for this workflow is: `sudo yum update-minimal --security --sev erity=Important,Moderate --bugfix -y` For version 8 and 9 managed nodes, the equivalent dnf command for this workflow is:

Security option	Patch selection
	`sudo dnf upgrade-minimal --security --sec-severity=Moderate --sec-severity=Important`

Custom patch baselines where the **Include non-security updates** check box *is* selected with a SEVERITY list of `[Critical, Important]` and a CLASSIFICATION list of `[Security, Bugfix]`	In addition to applying the security updates that were selected from `updateinfo.xml`, Patch Manager applies nonsecurity updates that otherwise meet the patch filtering rules.
	
	For version 7 managed nodes, the equivalent yum command for this workflow is:
	
	`sudo yum update --security --sec-severity=Critical,Important --bugfix -y`
	
	For version 8 and 9 managed nodes, the equivalent dnf command for this workflow is:
	
	`sudo dnf upgrade --security --sec-severity=Critical, --sec-severity=Important --bugfix y`
	
	Note
	New packages that replace now-obsolete packages with different names are installed if you run these `yum` or `dnf` commands outside of Patch Manager. However, they are *not* installed by the equivalent Patch Manager operations.

For information about patch compliance status values, see Patch compliance state values.

How patch baseline rules work on AlmaLinux, RHEL, and Rocky Linux

On AlmaLinux, Red Hat Enterprise Linux (RHEL), and Rocky Linux, the patch selection process is as follows:

1. On the managed node, the YUM library (RHEL 7) or the DNF library (AlmaLinux 8 and 9, RHEL 8, 9, and 10, and Rocky Linux 8 and 9) 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	Corresponds to the value of the Classification key attribute in the patch baseline's PatchFilter data type. Denotes the type of package included in the update notice. You can view the list of supported values by using the AWS CLI command **describe-patch-properties** or the API 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 packages included in the update notice.

Attribute	Description
	Usually only applicable for *Security* update notices. You can view the list of supported values by using the AWS CLI command **describe-patch-properties** or the API 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.
update_id	Denotes the advisory ID, such as *RHSA-2017:0864*. The advisory ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.
references	Contains additional information about the update notice, such as a CVE ID (format: *CVE-2017-1000371*) or a Bugzilla ID (format: *1463241*). The CVE ID and Bugzilla ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.
updated	Corresponds to ApproveAfterDays in the patch baseline. Denotes the released date (updated date) of the packages included in the update notice. A comparison between the current timestamp and the value of this attribute plus the `ApproveAfterDays` is used to determine if the patch is approved for deployment.

For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.

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 **Include non-security updates** check box is *not* selected in any rule	For each update notice in `updateinfo.xml` , the patch baseline is used as a filter, allowing only the qualified packages to be included in the update. If multiple packages are applicable after applying the patch baseline definition, the latest version is used.
	
	For RHEL 7, the equivalent yum command for this workflow is:
	
	`sudo yum update-minimal --sec-severity=Critical,Important --bugfix -y`
	
	For AlmaLinux 8 and 9, RHEL 8, 9, and 10, and Rocky Linux 8 and 9, the equivalent dnf command for this workflow is:
	
	`sudo dnf upgrade-minimal --sec-severity=Critical --sec-severity=Important --bugfix -y`

Custom patch baselines where the **Include non-security updates** check box *is* selected with a SEVERITY list of `[Critical, Important]` and a CLASSIFICATION list of `[Security, Bugfix]`	In addition to applying the security updates that were selected from `updateinfo.xml` , Patch Manager applies nonsecurity updates that otherwise meet the patch filtering rules. For RHEL 7, the equivalent yum command for this workflow is:

Security option	Patch selection

	`sudo yum update --security --sec-severity=Critical,Important --bugfix -y`
	
	For AlmaLinux 8 and 9, RHEL 8, 9, and 10, and Rocky Linux 8 and 9, the equivalent dnf command for this workflow is:
	
	`sudo dnf upgrade --sec-severity=Critical --sec-severity=Important --bugfix -y`
	
	Note
	New packages that replace now-obsolete packages with different names are installed if you run these `yum` or `dnf` commands outside of Patch Manager. However, they are *not* installed by the equivalent Patch Manager operations.

For information about patch compliance status values, see Patch compliance state values.

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 16.04 LTS: `xenial-security`
- Ubuntu Server 18.04 LTS: `bionic-security`
- Ubuntu Server 20.04 LTS: `focal-security`
- Ubuntu Server 22.04 LTS (`jammy-security`)
- Ubuntu Server 24.04 LTS (`noble-security`)
- Ubuntu Server 25.04 (`plucky-security`)

If nonsecurity updates are included, patches from other repositories are considered as well.

For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.

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 Patch compliance state values.

Patching operation differences between Linux and Windows Server

This topic describes important differences between Linux and Windows Server patching in Patch Manager, a tool in 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 tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. Subscribe to the SSM Agent Release Notes page on GitHub to get notifications about SSM Agent updates.

Difference 1: Patch evaluation

Patch Manager uses different processes on Windows managed nodes and Linux managed nodes in order to evaluate which patches should be present.

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

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 SSM Command documents for patching managed nodes and SSM Command document for patching: `AWS-RunPatchBaseline`.

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).

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.

Difference 5: Rejected patch list options in custom patch baselines

When you create a custom patch baseline, you can specify one or more patches for a **Rejected patches** list. For Linux managed nodes, you can also choose to allow them to be installed if they're dependencies for another patch allowed by the baseline.

Windows Server, however, doesn't support the concept of patch dependencies. You can add a patch to the **Rejected patches** list in a custom baseline for Windows Server, but the result depends on (1) whether or not the rejected patch is already installed on a managed node, and (2) which option you choose for **Rejected patches action**.

Refer to the following table for details about rejected patch options on Windows Server.

Installation status	Option: "Allow as dependency"	Option: "Block"
Patch is already installed	Reported status: `INSTALLED_OTHER`	Reported status: `INSTALLED_REJECTED`
Patch is not already installed	Patch skipped	Patch skipped

Each patch for Windows Server that Microsoft releases typically contains all the information needed for the installation to succeed. Occasionally, however, a prerequisite package might be required, which you must install manually. Patch Manager doesn't report information about these prerequisites. For related information, see Windows Update issues troubleshooting on the Microsoft website.

SSM Command 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.

Recommended SSM documents for patching

We recommend using the following five SSM documents in your patching operations.

- `AWS-ConfigureWindowsUpdate`
- `AWS-InstallWindowsUpdates`
- `AWS-RunPatchBaseline`
- `AWS-RunPatchBaselineAssociation`
- `AWS-RunPatchBaselineWithHooks`

Legacy SSM documents for patching

The following four legacy SSM documents remain available in some AWS Regions but are no longer updated or supported. These documents aren't guaranteed to work in IPv6 environments and support only IPv4. They can't be guaranteed to work in all scenarios and might lose support in the future. We recommend that you don't use these documents for patching operations.

- `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
- Legacy SSM documents for patching managed nodes
- SSM Command document for patching: AWS-RunPatchBaseline
- SSM Command document for patching: AWS-RunPatchBaselineAssociation
- SSM Command document for patching: AWS-RunPatchBaselineWithHooks
- Sample scenario for using the InstallOverrideList parameter in AWS-RunPatchBaseline or AWS-RunPatchBaselineAssociation
- Using the BaselineOverride parameter

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
- AWS-InstallWindowsUpdates
- AWS-RunPatchBaseline
- AWS-RunPatchBaselineAssociation
- AWS-RunPatchBaselineWithHooks

`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 tool in AWS Systems Manager, to ensure Windows Update maintains its configuration. You can also run it manually using Run Command, a tool in 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.

`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). For more information about the Systems Manager Compliance tools, see AWS Systems Manager Compliance.

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 SSM Command document for patching: AWS-RunPatchBaseline.

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 State Manager associations created using Quick Setup, a tool in AWS Systems Manager. Specifically, when you use 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 or AWS-RunPatchBaselineWithHooks instead of AWS-RunPatchBaselineAssociation.

  For more information, see the following topics:

  - AWS Systems Manager Quick Setup
  - SSM Command document for patching: AWS-RunPatchBaselineAssociation
- 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). For more information about the Systems Manager Compliance tools, see AWS Systems Manager Compliance.

Replaces legacy documents:

- **None**

For more information about the `AWS-RunPatchBaselineAssociation` SSM document, see SSM Command document for patching: `AWS-RunPatchBaselineAssociation`.

`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. Not supported on macOS.

`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 SSM Command document for patching: `AWS-RunPatchBaselineWithHooks`.

Legacy SSM documents for patching managed nodes

The following four SSM documents are still available in some AWS Regions. However, they are no longer updated and might be no longer supported in the future, so we don't recommend their use. Instead, use the documents described in SSM documents recommended for patching managed nodes.

Legacy SSM Documents

- AWS-ApplyPatchBaseline

- AWS-FindWindowsUpdates
- AWS-InstallMissingWindowsUpdates
- AWS-InstallSpecificWindowsUpdates

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`*

SSM Command document for patching: `AWS-RunPatchBaseline`

AWS Systems Manager supports `AWS-RunPatchBaseline`, a Systems Manager document (SSM document) for Patch Manager, a tool in 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 Patch groups.

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 Server

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 2, Oracle Linux, and RHEL 7 managed nodes use YUM. For YUM operations, Patch Manager requires `Python 2.6` or a later supported version (2.6 - 3.10). Amazon Linux 2023 uses DNF. For DNF operations, Patch Manager requires a supported version of `Python 2` or `Python 3` (2.6 - 3.10).
- RHEL 8 managed nodes use DNF. For DNF operations, Patch Manager requires a supported version of `Python 2` or `Python 3` (2.6 - 3.10). (Neither version is installed by default on RHEL 8. You must install one or the other manually.)
- Debian Server, and Ubuntu Server instances use APT. For APT operations, Patch Manager requires a supported version of `Python 3` (3.0 - 3.10).

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 3 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`.

For information about viewing patch compliance data, see About patch compliance.

`AWS-RunPatchBaseline` parameters

`AWS-RunPatchBaseline` supports six 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
- Parameter name: AssociationId
- Parameter name: Snapshot ID
- Parameter name: InstallOverrideList
- Parameter name: RebootOption
- Parameter name: BaselineOverride
- Parameter name: StepTimeoutSeconds

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`.)

> **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: `AssociationId`

Usage: Optional.

`AssociationId` is the ID of an existing association in State Manager, a tool in AWS Systems Manager. It's used by Patch Manager to add compliance data to a specified association. This association is related to a patching operation that's set up in a patch policy in Quick Setup.

> **Note**
>
> With the `AWS-RunPatchBaseline`, if an `AssociationId` value is provided along with a patch policy baseline override, patching is done as a `PatchPolicy` operation and

the `ExecutionType` value reported in `AWS:ComplianceItem` is also `PatchPolicy`. If no `AssociationId` value is provided, patching is done as a `Command` operation and the `ExecutionType` value report in on the `AWS:ComplianceItem` submitted is also `Command`.

If you don't already have an association you want to use, you can create one by running [create-association] the command.

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.	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. If you do specify a value in this scenario, note that

Mode	Best practice	Details
		the snapshot of the patch baseline might not remain in place for more than 3 days. After that, a new snapshot will be generated even if you specify the same ID after the snapshot expires.

Mode	Best practice	Details
Running AWS-RunPatchBaseline outside of a maintenance window	Generate and specify a custom GUID value for the Snapshot ID.[1]	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. For instance, say that you're running the AWS-RunPatchBaseline document directly through Run Command, a tool in 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

Mode	Best practice	Details
		nodes, ensuring that they end up in a consistent state.

[1] You can use any tool capable of generating a GUID to generate a value for the Snapshot ID parameter. For example, in PowerShell, you can use the `New-Guid` cmdlet to generate a GUID in the format of `12345699-9405-4f69-bc5e-9315aEXAMPLE` .

Parameter name: `InstallOverrideList`

Usage: Optional.

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.

> **Important**
>
> The `InstallOverrideList` file name can't contain the following characters: backtick (`), single quote ('), double quote ("), and dollar sign ($).

The patching operation behavior when using the `InstallOverrideList` parameter differs between Linux & macOS managed nodes and Windows Server managed nodes. On Linux & macOS, Patch Manager attempts to apply patches included in the `InstallOverrideList` patch list that are present in any repository enabled on the node, whether or not the patches match the patch baseline rules. On Windows Server nodes, however, patches in the `InstallOverrideList` patch list are applied *only* if they also match the patch baseline rules.

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.

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.aws-api-domain/amzn-s3-demo-bucket/my-windows-override-list.yaml
```

- **Amazon S3 path-style URL**:

```
s3://amzn-s3-demo-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:

In this case, apt version 1.2.27 will be blocked from installation and reported as “Failed-NonCompliant.”

Windows Server

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 2**

```
patches:
    -
```

```
        id: 'kernel.x86_64'
    -
        id: 'bind*.x86_64'
        title: '39.11.4-26.P2.amzn2.5.2'

        id: 'glibc*'
    -
        id: 'dhclient*'
        title: '*4.2.5-58.amzn2'
    -
        id: 'dhcp*'
        title: '*4.2.5-77.amzn2'
```

- **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'
```

- **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`.

Important

We don't recommend using Patch Manager for patching cluster instances in Amazon EMR (previously called Amazon Elastic MapReduce). In particular, don't select the `RebootIfNeeded` option for the `RebootOption` parameter. (This option is available in the SSM Command documents for patching `AWS-RunPatchBaseline`, `AWS-RunPatchBaselineAssociation`, and `AWS-RunPatchBaselineWithHooks`.) The underlying commands for patching using Patch Manager use `yum` and `dnf` commands. Therefore, the operations result in incompatibilities because of how packages are installed. For information about the preferred methods for updating software on Amazon EMR clusters, see Using the default AMI for Amazon EMR in the *Amazon EMR Management Guide*.

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, or that a patch was installed outside of Patch Manager since the last time the managed node was rebooted.

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.

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`.

> **Important**
>
> The `NoReboot` option only prevents operating system-level restarts. Service-level restarts can still occur as part of the patching process. For example, when Docker is updated, dependent services such as Amazon Elastic Container Service might automatically restart even with `NoReboot` enabled. If you have critical services that must not be disrupted, consider additional measures such as temporarily removing instances from service or scheduling patching during maintenance windows.

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

> **Important**
>
> The `BaselineOverride` file name can't contain the following characters: backtick (`), single quote ('), double quote ("), and dollar sign ($).

For more information about how to use the `BaselineOverride` parameter, see Using the BaselineOverride parameter.

Parameter name: `StepTimeoutSeconds`

Usage: Optional.

The time in seconds—between 1 and 36000 seconds (10 hours)—for a command to be completed before it is considered to have failed.

SSM Command document for patching: `AWS-RunPatchBaselineAssociation`

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.)

This document supports Amazon Elastic Compute Cloud (Amazon EC2) instances for Linux, macOS, and Windows Server. It does not support non-EC2 nodes in a hybrid and multicloud environment. 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 State Manager associations created using Quick Setup, a tool in AWS Systems Manager. Specifically, when you use 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` or `AWS-RunPatchBaselineWithHooks` 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 tool in 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.

  > **Note**
  >
  > 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 for Amazon Elastic Compute Cloud (Amazon EC2) instances. For more information, see Parameter name: `BaselineTags`.

  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`

> **Important**
>
> Tag keys and values can't contain the following characters: backtick (`), single quote ('), double quote ("), and dollar sign ($).

- 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-02573cafcfEXAMPLE" \
    --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-02573cafcfEXAMPLE",
            "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 nodes, the appropriate package manager for each node type is used to install packages:

- Amazon Linux 2, Oracle Linux, and RHEL instances use YUM. For YUM operations, Patch Manager requires `Python 2.6` or a later supported version (2.6 - 3.10). Amazon Linux 2023 uses DNF. For DNF operations, Patch Manager requires a supported version of `Python 2` or `Python 3` (2.6 - 3.10)
- Debian Server and Ubuntu Server instances use APT. For APT operations, Patch Manager requires a supported version of `Python 3` (3.0 - 3.10).

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`.

For information about viewing patch compliance data, see About patch compliance.

`AWS-RunPatchBaselineAssociation` parameters

`AWS-RunPatchBaselineAssociation` supports five parameters. The `Operation` and `AssociationId` parameters are required. The `InstallOverrideList`, `RebootOption`, and `BaselineTags` parameters are optional.

Parameters

- Parameter name: Operation
- Parameter name: BaselineTags
- Parameter name: AssociationId
- Parameter name: InstallOverrideList
- Parameter name: RebootOption

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`.)

> **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`*

> **Important**
>
> Tag keys and values can't contain the following characters: backtick (`), single quote ('), double quote ("), and dollar sign ($).

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 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 tool in AWS Systems Manager. It's used by Patch Manager to add compliance data to a specified association. This association is related to a patch `Scan` operation enabled in a Host Management configuration created in Quick Setup. 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 "MyPatchHostConfigAssociation" \
    --targets
 "Key=instanceids,Values=[i-02573cafcfEXAMPLE,i-07782c72faEXAMPLE,i-07782c72faEXAMPLE]"
 \
    --parameters "Operation=Scan" \
    --schedule-expression "cron(0 */30 * * * ? *)" \
    --sync-compliance "MANUAL" \
    --region us-east-2
```

Windows Server

```
aws ssm create-association ^
    --name "AWS-RunPatchBaselineAssociation" ^
    --association-name "MyPatchHostConfigAssociation" ^
    --targets
 "Key=instanceids,Values=[i-02573cafcfEXAMPLE,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.

> **Important**
>
> The `InstallOverrideList` file name can't contain the following characters: backtick (`), single quote ('), double quote ("), and dollar sign ($).

The patching operation behavior when using the `InstallOverrideList` parameter differs between Linux & macOS managed nodes and Windows Server managed nodes. On Linux & macOS, Patch Manager attempts to apply patches included in the `InstallOverrideList` patch list that are present in any repository enabled on the node, whether or not the patches match the patch baseline rules. On Windows Server nodes, however, patches in the `InstallOverrideList` patch list are applied *only* if they also match the patch baseline rules.

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/amzn-s3-demo-bucket/my-windows-override-list.yaml
```

- **Amazon S3 path-style URL example**:

```
s3://amzn-s3-demo-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/Red Hat Enterprise Linux (RHEL):

```
{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 2**

```
patches:
    -
        id: 'kernel.x86_64'
    -
```

```
        id: 'bind*.x86_64'
        title: '39.11.4-26.P2.amzn2.5.2'

        id: 'glibc*'
    -
        id: 'dhclient*'
        title: '*4.2.5-58.amzn2'
    -
        id: 'dhcp*'
        title: '*4.2.5-77.amzn2'
```

- **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'
```

- **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`.

Important

The `NoReboot` option only prevents operating system-level restarts. Service-level restarts can still occur as part of the patching process. For example, when Docker is updated, dependent services such as Amazon Elastic Container Service might automatically restart even with `NoReboot` enabled. If you have critical services that must not be disrupted, consider additional measures such as temporarily removing instances from service or scheduling patching during maintenance windows.

Important

We don't recommend using Patch Manager for patching cluster instances in Amazon EMR (previously called Amazon Elastic MapReduce). In particular, don't select the `RebootIfNeeded` option for the `RebootOption` parameter. (This option is available in the SSM Command documents for patching `AWS-RunPatchBaseline`, `AWS-RunPatchBaselineAssociation`, and `AWS-RunPatchBaselineWithHooks`.)
The underlying commands for patching using Patch Manager use `yum` and `dnf` commands. Therefore, the operations result in incompatibilities because of how packages are installed. For information about the preferred methods for updating software on Amazon EMR clusters, see Using the default AMI for Amazon EMR in the *Amazon EMR Management Guide*.

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, or that a patch was installed outside of Patch Manager since the last time the managed node was rebooted.

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`

SSM Command document for patching: AWS-RunPatchBaselineWithHooks

AWS Systems Manager supports AWS-RunPatchBaselineWithHooks, a Systems Manager document (SSM document) for Patch Manager, a tool in 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 Patch groups.

You can use the document AWS-RunPatchBaselineWithHooks 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 and Windows Server managed nodes. The document will perform the appropriate actions for each platform.

> **Note**
>
> AWS-RunPatchBaselineWithHooks isn't supported on macOS.

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 2, Oracle Linux, and RHEL 7 managed nodes use YUM. For YUM operations, Patch Manager requires `Python 2.6` or a later supported version (2.6 - 3.10). Amazon Linux 2023 uses DNF. For DNF operations, Patch Manager requires a supported version of `Python 2` or `Python 3` (2.6 - 3.10).
- RHEL 8 managed nodes use DNF. For DNF operations, Patch Manager requires a supported version of `Python 2` or `Python 3` (2.6 - 3.10). (Neither version is installed by default on RHEL 8. You must install one or the other manually.)
- Debian Server and Ubuntu Server instances use APT. For APT operations, Patch Manager requires a supported version of `Python 3` (3.0 - 3.10).

Windows Server

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.

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.

> **Important**
>
> While the NoReboot option prevents operating system restarts, it does not prevent service-level restarts that might occur when certain packages are updated. For example, updating packages like Docker may trigger automatic restarts of dependent services (such as container orchestration services) even when NoReboot is specified.

For information about viewing patch compliance data, see About patch compliance.

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. This means that the operation continues to Step 7 and the managed node is rebooted in either of the following cases:
      i. 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.)
      ii. 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, or that a patch was installed outside of Patch Manager since the last time the managed node was rebooted.

      If no patches meeting these criteria 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.
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
- Parameter name: Snapshot ID
- Parameter name: RebootOption
- Parameter name: PreInstallHookDocName
- Parameter name: PostInstallHookDocName
- Parameter name: OnExitHookDocName

Parameter name: `Operation`

Usage: Required.

Options: `Scan` | `Install`.

Scan

> When you choose the `Scan` option, the systems 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`.)

> **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-RunPatchBaselineWithHooks` 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-RunPatchBaselineWithHooks`, you shouldn't provide your own generated Snapshot

Mode	Best practice	Details
		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-RunPatchBaselineWithHooks` 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 3 days. After that, a new snapshot will be generated even if you specify the same ID after the snapshot expires.

Mode	Best practice	Details
Running `AWS-RunPatchBaselineWithHooks` outside of a maintenance window	Generate and specify a custom GUID value for the Snapshot ID.[1]	When you aren't using a maintenance window to run `AWS-RunPatchBaselineWithHooks`, we recommend that you generate and specify a unique Snapshot ID for each patch baseline, particularly if you're running the `AWS-RunPatchBaselineWithHooks` 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 nodes. For instance, say that you're running the `AWS-RunPatchBaselineWithHooks` document directly through Run Command, a tool inAWS 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,

Mode	Best practice	Details
		ensuring that they end up in a consistent state.

[1] You can use any tool capable of generating a GUID to generate a value for the Snapshot ID parameter. For example, in PowerShell, you can use the `New-Guid` cmdlet to generate a GUID in the format of `12345699-9405-4f69-bc5e-9315aEXAMPLE`.

Parameter name: `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`.

> **Important**
>
> We don't recommend using Patch Manager for patching cluster instances in Amazon EMR (previously called Amazon Elastic MapReduce). In particular, don't select the `RebootIfNeeded` option for the `RebootOption` parameter. (This option is available in the SSM Command documents for patching `AWS-RunPatchBaseline`, `AWS-RunPatchBaselineAssociation`, and `AWS-RunPatchBaselineWithHooks`.)
> The underlying commands for patching using Patch Manager use `yum` and `dnf` commands. Therefore, the operations result in incompatibilities because of how packages are installed. For information about the preferred methods for updating software on Amazon EMR clusters, see Using the default AMI for Amazon EMR in the *Amazon EMR Management Guide*.

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, or that a patch was installed outside of Patch Manager since the last time the managed node was rebooted.

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 `arn:aws: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 Command document plugin reference). 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 document content.

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 `arn:aws: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 Command document plugin reference). 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 document content.

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 `arn:aws: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 Command document plugin reference). 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 document content.

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 tool in 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.

For more information and sample lists, see Parameter name: `InstallOverrideList`.

Using the BaselineOverride parameter

You can define patching preferences at runtime using the baseline override feature in Patch Manager, a tool in 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.

> **Important**
>
> The `BaselineOverride` file name can't contain the following characters: backtick (`), single quote ('), double quote ("), and dollar sign ($).

Except when a patching operation uses a patch policy, 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 tool in 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`.

When a patch operation uses a patch policy, however, the system passes the override parameter from the associated S3 bucket, and the compliance value is updated for the managed node. For more information about patch policy behaviors, see Patch policy configurations in Quick Setup.

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": "SUSE",
        "RejectedPatches": [
            "python*"
        ],
        "RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
        "Sources": []
    }
]
```

Patch baselines

The topics in this section provide information about how patch baselines work in Patch Manager, a tool in AWS Systems Manager, when you run a `Scan` or `Install` operation on your managed nodes.

Topics

- Predefined and custom patch baselines
- Package name formats for approved and rejected patch lists

- Patch groups
- Patching applications released by Microsoft on Windows Server

Predefined and custom patch baselines

Patch Manager, a tool in 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 Custom baselines and Working with custom patch baselines.

> **Note**
>
> The information in this topic applies no matter which method or type of configuration you are using for your patching operations:
>
> - A patch policy configured in Quick Setup
> - A Host Management option configured in Quick Setup
> - A maintenance window to run a patch `Scan` or `Install` task
> - An on-demand **Patch now** operation

Topics

- Predefined baselines
- Custom baselines

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.

Name	Supported operating system	Details
AWS-AlmaLinuxDefaultPatchBaseline	AlmaLinux	Approves all operating system patches that are classified as "Security" and that have a severity level of "Critical" or "Important". Also approves all patches that are classified as "Bugfix". Patches are auto-approved 7 days after they are released or updated.[1]
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". Also approves all patches with a classification of "Bugfix". Patches are auto-approved 7 days after release.[1]
AWS-AmazonLinux2023DefaultPatchBaseline	Amazon Linux 2023	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-DebianDefaultPatchBaseline	Debian Server	Immediately approves all operating system security-related patches that have a priority of "Required",

Name	Supported operating system	Details
		"Important", "Standard," "Optional," or "Extra." There is no wait before approval because reliable release dates aren't available in the repositories.
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". Also approves all patches that are classified as "Bugfix" 7 days after release. Patches are auto-approved 7 days after they are released or updated.[1]
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". Also approves all patches that are classified as "Bugfix". Patches are auto-approved 7 days after they are released or updated.[1]

Name	Supported operating system	Details
AWS-RockyLinuxDefaultPatchBaseline	Rocky Linux	Approves all operating system patches that are classified as "Security" and that have a severity level of "Critical" or "Important". Also approves all patches that are classified as "Bugfix". Patches are auto-approved 7 days after they are released or updated.[1]
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 repositories.
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 7 days after they are released or updated.[2]

Name	Supported operating system	Details
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 7 days after they are released or updated.[2]
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 7 days after they are released or updated.[2]

[1] For Amazon Linux 2, the 7-day wait before patches are auto-approved is calculated from an `Updated Date` value in `updateinfo.xml`, not a `Release Date` value. Various factors can affect the `Updated Date` value. Other operating systems handle release and update dates differently. For information to help you avoid unexpected results with auto-approval delays, see How package release dates and update dates are calculated.

[2] For Windows Server, default baselines include a 7-day auto-approval delay. To install a patch within 7 days after release, you must create a custom baseline.

Custom baselines

Use the following information to help you create custom patch baselines to meet your patching goals.

Topics

- Using auto-approvals in custom baselines
- Additional information for creating patch baselines

Using auto-approvals in custom baselines

If you create your own patch baseline, you can choose which patches to auto-approve by using the following categories.

- **Operating system**: Supported versions of Windows Server, Amazon Linux, Ubuntu Server, and so on.
- **Product name** (for operating systems): For example, RHEL 7.5, Amazon Linux 2023 2023.8.20250808, 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 or last updated, before the patch is automatically approved for patching. For example, if you create a rule using the `CriticalUpdates` classification and configure it for 7 days auto-approval delay, then a new critical patch released on July 7 is automatically approved on July 14.

If a Linux repository does not provide release date information for packages, Patch Manager uses the build time of the package as the date for auto-approval date specifications for Amazon Linux 2, and Red Hat Enterprise Linux (RHEL). If the build time of the package can't be determined, Patch Manager uses a default date of January 1st, 1970. This results in Patch Manager bypassing any auto-approval date specifications in patch baselines that are configured to approve patches for any date after January 1st, 1970.

When you specify an auto-approval cutoff date, Patch Manager automatically applies all patches released or last updated on or before that date. For example, if you specify July 7, 2023 as the cutoff date, no patches released or last updated on or after July 8, 2023 are installed automatically.

When you create a custom patch baseline, you can specify a compliance severity level for patches approved by that patch baseline, such as `Critical` or `High`. If the patch state of any approved patch is reported as `Missing`, then the patch baseline's overall reported compliance severity is the severity level you specified.

Additional information for creating patch baselines

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.

- By default, Windows Server 2019 and Windows Server 2022 remove updates that are replaced by later updates. As a result, if you use the `ApproveUntilDate` parameter in a Windows Server patch baseline, but the date selected in the `ApproveUntilDate` parameter is before the date of the latest patch, then the new patch isn't installed when the patching operation runs. For more information about Windows Server patching rules, see the Windows Server tab in How security patches are selected.

  This means that the managed node is compliant in terms of Systems Manager operations, even though a critical patch from the previous month might not be installed. This same scenario can occur when using the `ApproveAfterDays` parameter. Because of the Microsoft superseded patch behavior, it is possible to set a number (generally greater than 30 days) so that patches for Windows Server are never installed if the latest available patch from Microsoft is released before the number of days in `ApproveAfterDays` has elapsed.
- For Windows Server only, an available security update patch that is not approved by the patch baseline can have a compliance value or `Compliant` or `Non-Compliant`, as defined in a custom patch baseline.

  When you create or update a patch baseline, you choose the status you want to assign to security patches that are available but not approved because they don't meet the installation criteria specified in the patch baseline. For example, security patches that you might want installed can be skipped if you have specified a long period to wait after a patch is released before installation. If an update to the patch is released during your specified waiting period, the waiting period for installing the patch starts over. If the waiting period is too long, multiple versions of the patch could be released but never installed.

  Using the console to create or update a patch baseline, you specify this option in the **Available security updates compliance status** field. Using the AWS CLI to run the create-patch-baseline or update-patch-baseline command, you specify this option in the `available-security-updates-compliance-status` parameter.
- 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 Package name formats for approved and rejected patch lists.
- If you are using a patch policy configuration in Quick Setup, updates you make to custom patch baselines are synchronized with Quick Setup once an hour.

  If a custom patch baseline that was referenced in a patch policy is deleted, a banner displays on the Quick Setup **Configuration details** page for your patch policy. The banner informs you that the patch policy references a patch baseline that no longer exists, and that subsequent patching operations will fail. In this case, return to the Quick Setup **Configurations** page, select the Patch Manager configuration , and choose **Actions**, **Edit configuration**. The deleted patch baseline name is highlighted, and you must select a new patch baseline for the affected operating system.

For information about creating a patch baseline, see Working with custom patch baselines and Tutorial: Patch a server environment using the AWS CLI.

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 2, Amazon Linux 2023, Oracle Linux, and Red Hat Enterprise Linux (RHEL)
- Debian Server and Ubuntu Server

Amazon Linux 2, Amazon Linux 2023, Oracle Linux, and Red Hat Enterprise Linux (RHEL)

Package manager: YUM, except for Amazon Linux 2023, and RHEL 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`
- Package names that are constructed using one or more of the available components for package naming. To illustrate, for the package named `dbus.x86_64:1:1.12.28-1.amzn2023.0.1`, the components are as follows:
  - `name`: `dbus`
  - `architecture`: `x86_64`
  - `epoch`: `1`
  - `version`: `1.12.28`
  - `release`: `1.amzn2023.0.1`

  Package names with the following constructions are supported:
  - `name`
  - `name.arch`
  - `name-version`
  - `name-version-release`
  - `name-version-release.arch`
  - `version`
  - `version-release`
  - `epoch:version-release`
  - `name-epoch:version-release`
  - `name-epoch:version-release.arch`
  - `epoch:name-version-release.arch`
  - `name.arch:epoch:version-release`

  Some examples:
  - `dbus.x86_64`
  - `dbus-1.12.28`
  - `dbus-1.12.28-1.amzn2023.0.1`
  - `dbus-1:1.12.28-1.amzn2023.0.1.x86_64`

- We also support package name components with a single wild card in the above formats, such as the following:
  - `dbus*`
  - `dbus-1.12.2*`
  - `dbus-*:1.12.28-1.amzn2023.0.1.x86_64`

Rejected patches: For rejected patches, you can specify any of the following:

- Package names that are constructed using one or more of the available components for package naming. To illustrate, for the package named `dbus.x86_64:1:1.12.28-1.amzn2023.0.1`, the components are as follows:
  - `name`: `dbus`
  - `architecture`; `x86_64`
  - `epoch`: `1`
  - `version`: `1.12.28`
  - `release`: `1.amzn2023.0.1`

  Package names with the following constructions are supported:
  - `name`
  - `name.arch`
  - `name-version`
  - `name-version-release`
  - `name-version-release.arch`
  - `version`
  - `version-release`
  - `epoch:version-release`
  - `name-epoch:version-release`
  - `name-epoch:version-release.arch`
  - `epoch:name-version-release.arch`
  - `name.arch:epoch:version-release`

  Some examples:
  - `dbus.x86_64`

- `dbus-1.12.28`
- `dbus-1.12.28-1.amzn2023.0.1`
- `dbus-1:1.12.28-1.amzn2023.0.1.x86_64`

- We also support package name components with a single wild card in the above formats, such as the following:
  - `dbus*`
  - `dbus-1.12.2*`
  - `dbus-*:1.12.28-1.amzn2023.0.1.x86_64`

Debian Server 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, 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`

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
```

Patch groups

> **Note**
>
> Patch groups are not used in patching operations that are based on *patch policies*. For information about working with patch policies, see Patch policy configurations in Quick Setup.
>
> Patch group functionality is not supported in the console for account-Region pairs that did not already use patch groups before patch policy support was released on December 22, 2022. Patch group functionality is still available in account-Region pairs that began using patch groups before this date.

You can use a *patch group* to associate managed nodes with a specific patch baseline in Patch Manager, a tool in 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` or other SSM Command documents for patching, 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.

Using tags to define patch groups

You create a patch group by using tags applied to your Amazon Elastic Compute Cloud (Amazon EC2) instances and non-EC2 nodes in a hybrid and multicloud environment. Note the following details about using tags for patch groups:

- A patch group must be defined using either the tag key `Patch Group` or `PatchGroup` applied to your managed nodes. When registering a patch group for a patch baseline, any identical *values* specified for these two keys are interpreted to be part of the same group. For instance, say that you have tagged five nodes with the first of the following key-value pairs, and five with the second:
  - `key=PatchGroup,value=DEV`
  - `key=Patch Group,value=DEV`

  The Patch Manager command to create a baseline combines these 10 managed nodes into a single group based on the value `DEV`. The AWS CLI equivalent for the command to create a patch baseline for patch groups is as follows:

  ```
  aws ssm register-patch-baseline-for-patch-group \
      --baseline-id pb-0c10e65780EXAMPLE \
      --patch-group DEV
  ```

  Combining values from different keys into a single target is unique to this Patch Manager command for creating a new patch group and not supported by other API actions. For example, if you run send-command actions using `PatchGroup` and `Patch Group` keys with the same values, you are targeting two completely different sets of nodes:

  ```
  aws ssm send-command \
      --document-name AWS-RunPatchBaseline \
      --targets "Key=tag:PatchGroup,Values=DEV"
  ```

  ```
  aws ssm send-command \
      --document-name AWS-RunPatchBaseline \
      --targets "Key=tag:Patch Group,Values=DEV"
  ```

- There are limits on tag-based targeting. Each array of targets for `SendCommand` can contain a maximum of five key-value pairs.

- We recommend that you choose only one of these tag key conventions, either `PatchGroup` (without a space) or `Patch Group` (with a space). However, if you have allowed tags in EC2 instance metadata on an instance, you must use `PatchGroup`.
- The key is case-sensitive. You can specify any *value* to help you identify and target the resources in that group, for example "web servers" or "US-EAST-PROD", but the key must be `Patch Group` or `PatchGroup`.

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 Creating and managing patch groups and Add a patch group to a patch baseline.

To view an example of creating a patch baseline and patch groups by using the AWS Command Line Interface (AWS CLI), see Tutorial: Patch a server environment using the AWS CLI. For more information about Amazon EC2 tags, see Tag your Amazon EC2 resources in the *Amazon EC2 User Guide*.

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.

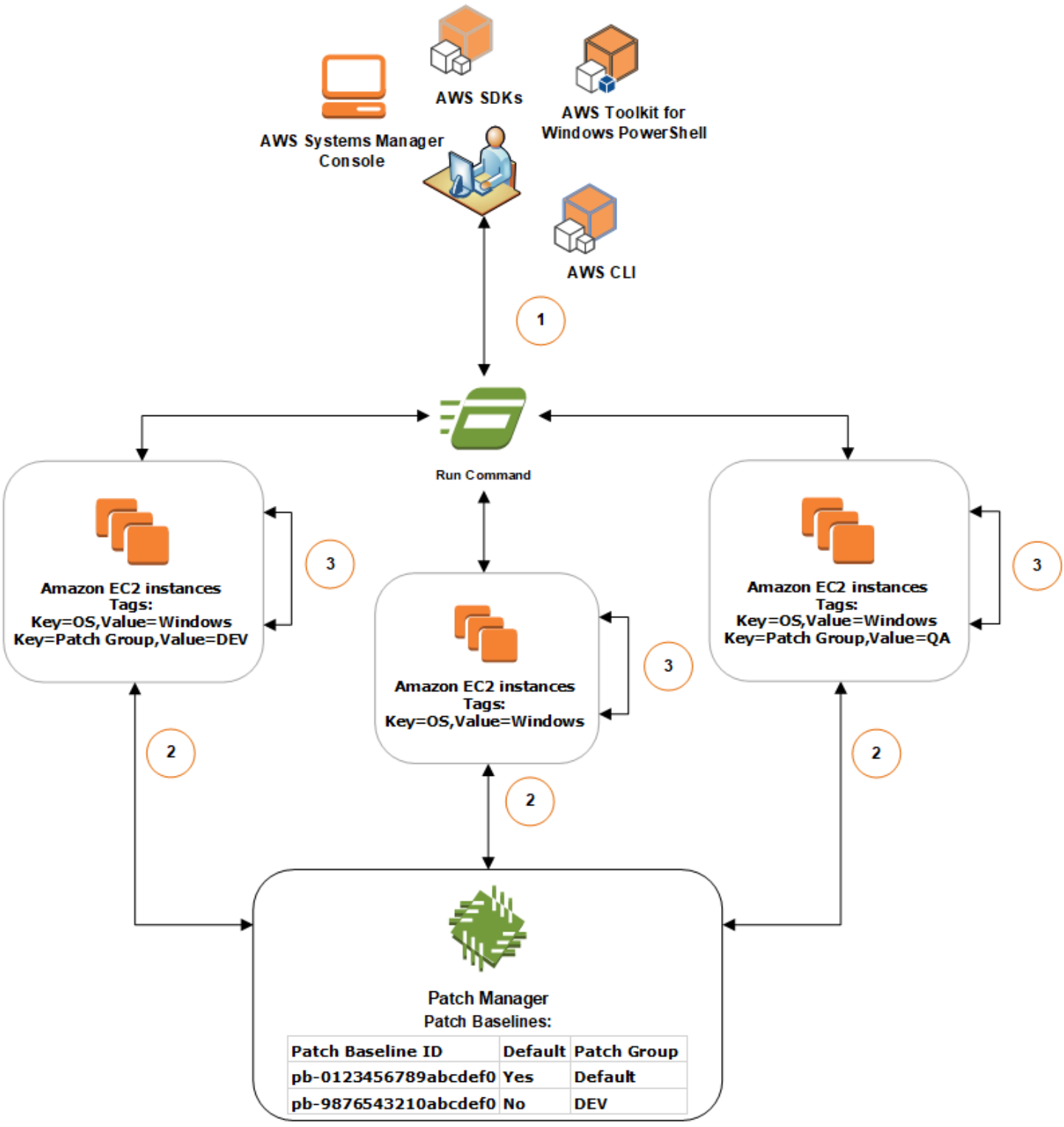
A patch group can be registered with only one patch baseline for each operating system type.

You can't apply the `Patch Group` tag (with a space) to an Amazon EC2 instance if the **Allow tags in instance metadata** option is enabled on the instance. Allowing tags in instance metadata prevents tag key names from containing spaces. If you have allowed tags in EC2 instance metadata, you must use the tag key `PatchGroup` (without a space).

Diagram 1: General example of patching operations process flow

The following illustration 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. These processes determine which patch baselines to use in patching operations. (A similar process is used when a maintenance window is configured to send a command to patch using Patch Manager.)

The full process is explained below the illustration.



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` `key=PatchGroup,value=DEV`
Group 2	`key=OS,value=Windows`
Group 3	`key=OS,value=Windows` `key=PatchGroup,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`

The general process to scan or install patches using Run Command, a tool in AWS Systems Manager, and Patch Manager is as follows:

1. **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`.
2. **Patch baseline determination**: SSM Agent verifies the patch group tags applied to the EC2 instance and queries Patch Manager for the corresponding patch baseline.
   - **Matching patch group value associated with patch baseline:**
     1. SSM Agent, which is installed on EC2 instances in group one, receives the command issued in Step 1 to begin a patching operation. SSM Agent validates that the EC2 instances have the patch group tag-value `DEV` applied and queries Patch Manager for an associated patch baseline.

2. Patch Manager verifies that patch baseline `pb-9876543210abcdef0` has the patch group `DEV` associated and notifies SSM Agent.
3. SSM Agent retrieves a patch baseline snapshot from Patch Manager based on the approval rules and exceptions configured in `pb-9876543210abcdef0` and proceeds to the next step.

- **No patch group tag added to instance:**

1. SSM Agent, which is installed on EC2 instances in group two, receives the command issued in Step 1 to begin a patching operation. SSM Agent validates that the EC2 instances don't have a `Patch Group` or `PatchGroup` 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.

More info

- Patch compliance state values

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 tool in 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 don't specify 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 on-premises servers, edge devices, VMs, and other non-EC2 nodes

To patch applications released by Microsoft on virtual machines (VMs) and other non-EC2 managed nodes, you must turn on the advanced-instances tier. There is a charge to use the advanced-

instances tier. **However, there is no additional charge to patch applications released by Microsoft on Amazon Elastic Compute Cloud (Amazon EC2) instances.** For more information, see Configuring instance tiers.

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**. We also recommend configuring the GPO with additional parameters that prevent unplanned automatic updates and reboots outside of Patch Manager. For more information, see Configuring Automatic Updates in a Non-Active Directory Environment on the Microsoft technical documentation website.

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 tool in 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.aws-api-domain/amzn-s3-demo-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" \
    --approval-rules
 "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=PRODUCT,Values='Office 2016'},
{Key=PATCH_SET,Values='APPLICATION'}]},ApproveAfterDays=5}]"
```

Windows Server

```
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 Server

```
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 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 AL2 in the *Amazon Linux 2 User Guide*.

After you turn on Kernel Live Patching on an Amazon Linux 2 managed node, you can use Patch Manager, a tool in 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*.

Topics

- Kernel Live Patching using Patch Manager
- How Kernel Live Patching using Patch Manager works
- Turning on Kernel Live Patching using Run Command
- Applying kernel live patches using Run Command

- Turning off Kernel Live Patching using Run Command

Kernel Live Patching using 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.

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 Kernel Live Patching using Patch Manager 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 7-day waiting period after a patch is released or last updated before it's installed automatically. If you don't want to wait 7 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.

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 tool in 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 tool in 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*.

> **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**.
3. Choose **Run command**.
4. In the **Command document** list, choose the custom SSM document `AWS-ConfigureKernelLivePatching`.

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.
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-ConfigureKernelLivePatching" \
      --parameters "EnableOrDisable=Enable" \
      --targets "Key=instanceids,Values=instance-id"
  ```

  Windows Server

  ```
  aws ssm send-command ^
      --document-name "AWS-ConfigureKernelLivePatching" ^
      --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-02573cafcfEXAMPLE. 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*.

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**.
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.
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 Server

```
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 Server

```
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-02573cafcfEXAMPLE. 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*.

> **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**.
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.
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 Server

  ```
  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-02573cafcfEXAMPLE. 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 resources and compliance using the console

To use Patch Manager, a tool in 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 Elastic Compute Cloud (Amazon EC2) tags (optional, but recommended).
3. Do one of the following:
   - (Recommended) Configure a patch policy in Quick Setup, a tool in Systems Manager, that lets you install missing patches on a schedule for an entire organization, a subset of organizational units, or a single AWS account. For more information, see Configure patching for instances in an organization using a Quick Setup patch policy.
   - Create a maintenance window that uses the Systems Manager document (SSM document) `AWS-RunPatchBaseline` in a Run Command task type. For more information, see Tutorial: Create a maintenance window for patching using the console.
   - Manually run `AWS-RunPatchBaseline` in a Run Command operation. For more information, see Running commands from the console.
   - Manually patch nodes on demand using the **Patch now** feature. For more information, see Patching managed nodes on demand.
4. Monitor patching to verify compliance and investigate failures.

Topics

- Creating a patch policy
- Viewing patch Dashboard summaries
- Working with patch compliance reports
- Patching managed nodes on demand
- Working with patch baselines
- Viewing available patches
- Creating and managing patch groups
- Integrating Patch Manager with AWS Security Hub

Creating a patch policy

A patch policy is a configuration you set up using Quick Setup, a tool in AWS Systems Manager. Patch policies provide more extensive and more centralized control over your patching operations than is available with other methods of configuring patching. A patch policy defines the schedule and baseline to use when automatically patching your nodes and applications.

For more information, see the following topics:

- Patch policy configurations in Quick Setup
- Configure patching for instances in an organization using a Quick Setup patch policy

Viewing patch Dashboard summaries

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 tool in 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**.
3. Choose the **Dashboard** tab.
4. Scroll to the section containing summary data that you want to view:

    - **Amazon EC2 instance management**
    - **Compliance summary**
    - **Noncompliance counts**
    - **Compliance reports**
    - **Non-patch policy-based operations**
    - **Non-patch policy-based recurring 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 tool in AWS Systems Manager.

The information in the following topics apply no matter which method or type of configuration you're using for your patching operations:

- A patch policy configured in Quick Setup
- A Host Management option configured in Quick Setup
- A maintenance window to run a patch `Scan` or `Install` task
- An on-demand **Patch now** operation

> **Important**
>
> If you have multiple types of operations in place to scan your instances for patch compliance, note that each scan overwrites the patch compliance data of previous scans. As a result, you might end up with unexpected results in your patch compliance data. For more information, see Avoiding unintentional patch compliance data overwrites.
> To verify which patch baseline was used to generate the latest compliance information, navigate to the **Compliance reporting** tab in Patch Manager, locate the row for the managed node you want information about, and then choose the baseline ID in the **Baseline ID used** column.

Topics

- Viewing patch compliance results
- Generating .csv patch compliance reports
- Remediating noncompliant managed nodes with Patch Manager
- Avoiding unintentional patch compliance data overwrites

Viewing patch compliance results

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 noncompliant managed nodes.

> **Note**
>
> The patch scanning operations for Quick Setup and Explorer use the `AWS-RunPatchBaselineAssociation` document. Quick Setup and Explorer are both tools in AWS Systems Manager.

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:

- AlmaLinux
- Amazon Linux 2
- Amazon Linux 2023
- Oracle Linux
- Red Hat Enterprise Linux (RHEL)
- Rocky Linux

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
- Package name formats for approved and rejected patch lists
- How patch baseline rules work on Linux-based systems
- How patches are installed

> **Note**
>
> In some cases, Microsoft releases patches for applications that don't specify 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

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.

To view patch compliance results

1. Do one of the following.

   Option 1 (recommended) – Navigate from Patch Manager, a tool in AWS Systems Manager:

   - In the navigation pane, choose **Patch Manager**.
   - Choose the **Compliance reporting** tab.
   - In the **Node patching details** area, choose the node ID of the managed node for which you want to review patch compliance results.
   - In the **Details** area, in the **Properties** list, choose **Patches**.

   Option 2 – Navigate from Compliance, a tool in AWS Systems Manager:

   - In the navigation pane, choose **Compliance**.
   - 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**.
   - Below, in the **Resource** list, choose the ID of the managed node for which you want to review patch compliance results.

- In the **Details** area, in the **Properties** list, choose **Patches**.

Option 3 – Navigate from Fleet Manager, a tool in AWS Systems Manager.

- In the navigation pane, choose **Fleet Manager**.
- In the **Managed instances** area, choose the ID of the managed node for which you want to review patch compliance results.
- In the **Details** area, in the **Properties** list, choose **Patches**.

2. (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

3. 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**.

   > **Note**
   >
   > Currently, CVE ID values are reported only for patches with a status of `Missing` or `Failed`.

4. 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.

For information about patch compliance states, see Patch compliance state values.

Generating .csv patch compliance reports

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.

After a report is generated, you can use a tool like Amazon QuickSight to import and analyze the data. 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*.

> **Note**
>
> When you create a custom patch baseline, you can specify a compliance severity level for patches approved by that patch baseline, such as `Critical` or `High`. If the patch state of any approved patch is reported as `Missing`, then the patch baseline's overall reported compliance severity is the severity level you specified.

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 an Automation assume role named `AWS-SystemsManager-PatchSummaryExportRole` to use for the export process to S3.

Note

If you are exporting compliance data to an encrypted S3 bucket, you must update its associated AWS KMS key policy to provide the necessary permissions for `AWS-SystemsManager-PatchSummaryExportRole`. For instance, add a permission similar to this to your S3 bucket's AWS KMS policy:

```
{
    "Effect": "Allow",
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "role-arn"
}
```

Replace *`role-arn`* with the Amazon Resource Name (ARN) of the created in your account, in the format `arn:aws:iam::111222333444:role/service-role/AWS-SystemsManager-PatchSummaryExportRole`.
For more information, see Key policies in AWS KMS in the *AWS Key Management Service Developer Guide*.

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.

Topics

- What's in a generated patch compliance report?
- Generating patch compliance reports for a single managed node
- Generating patch compliance reports for all managed nodes
- Viewing patch compliance reporting history

- Viewing patch compliance reporting schedules
- Troubleshooting patch compliance report generation

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

> **Note**
>
> When you create a custom patch baseline, you can specify a compliance severity level for patches approved by that patch baseline, such as `Critical` or `High`. If the patch state of any approved patch is reported as `Missing`, then the patch baseline's overall reported compliance severity is the severity level you specified.

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**.
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.
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, expend 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.

For information about viewing details of reporting schedules you have created, see Viewing patch compliance reporting schedules.

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 noncompliant 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 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**.
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.
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, expend 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.

For information about viewing details of reporting schedules you have created, see Viewing patch compliance reporting schedules.

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**.
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**.
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 tool in AWS Systems Manager.

Topics

- A message reports that the AWS-SystemsManager-PatchManagerExportRolePolicy policy is corrupted
- After deleting patch compliance policies or roles, scheduled reports aren't generated successfully

A message reports that the `AWS-SystemsManager-PatchManagerExportRolePolicy` policy is corrupted

Problem: You receive an error message similar to the following, indicating the `AWS-SystemsManager-PatchManagerExportRolePolicy` is corrupted:

```
An error occurred while updating the AWS-SystemsManager-PatchManagerExportRolePolicy
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**: Use the Patch Manager console or AWS CLI to delete the affected roles and policies before generating a new patch compliance report.

  To delete the corrupt policy using the console

  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-PatchManagerExportRolePolicy`, then delete the policy. Next, choose **Roles**, search for `AWS-SystemsManager-PatchSummaryExportRole`, then delete the role.

     Scheduled reports – If the problem 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-PatchManagerExportRolePolicy`, 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.

  To delete the corrupt policy using the AWS CLI

  Replace the *`placeholder values`* with your account ID.

- If the problem occurred while generating a one-time on-demand report, run the following commands:

```
aws iam delete-policy --policy-arn arn:aws:iam::account-id:policy/AWS-
SystemsManager-PatchManagerExportRolePolicy
```

```
aws iam delete-role --role-name AWS-SystemsManager-PatchSummaryExportRole
```

If the problem occurred while generating a report on a schedule, run the following commands:

```
aws iam delete-policy --policy-arn arn:aws:iam::account-id:policy/AWS-
EventBridge-Start-SSMAutomationRolePolicy
```

```
aws iam delete-policy --policy-arn arn:aws:iam::account-id:policy/AWS-
SystemsManager-PatchManagerExportRolePolicy
```

```
aws iam delete-role --role-name AWS-EventBridge-Start-SSMAutomationRole
```

```
aws iam delete-role --role-name AWS-SystemsManager-PatchSummaryExportRole
```

After completing either procedure, 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-PatchManagerExportRolePolicy`). 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 noncompliant 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 noncompliant managed nodes
- Patch compliance state values
- Patching noncompliant managed nodes

Identifying noncompliant 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 tool in 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 noncompliant 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.

	AWS-RunPatchBaseline	AWS-RunPatchBaselineAssociation
Processes that use the document	**Patch on demand** - You can scan or patch managed nodes on demand using the **Patch now** option. For information, see Patching managed nodes on demand. **Systems Manager Quick Setup patch policies** – You can create a patching configuration in Quick Setup, a tool in AWS Systems Manager, that can scan for or install missing patches on separate schedules for an entire organization, a subset of organizational units, or a single AWS account. For information, see Configure patching for instances in an organization using a Quick Setup patch policy. **Run a command** – You can manually run `AWS-RunPatchBaseline` in an operation in Run Command, a tool in AWS Systems Manager. For information, see Running commands from the console. **Maintenance window** – You can create a maintenan	**Systems Manager Quick Setup Host Management** – You can enable a Host Management configuration option in Quick Setup to scan your managed instances for patch compliance each day. For information, see Set up Amazon EC2 host management using Quick Setup. **Systems Manager Explorer** – When you allow Explorer, a tool in AWS Systems Manager, it regularly scans your managed instances for patch compliance and reports results in the Explorer dashboard.

	AWS-RunPatchBaseline	AWS-RunPatchBaselineAssociation
	ce window that uses the SSM document AWS-RunPatchBaseline in a Run Command task type. For information, see Tutorial: Create a maintenance window for patching using the console.	
Format of the patch scan result data	After AWS-RunPatchBaseline runs, Patch Manager sends an AWS:PatchSummary object to Inventory, a tool in 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 Systems Manager Configuration Compliance and Working with managed nodes. For more information, see Viewing patch compliance results.	If you use Quick Setup to scan your managed instances for patch compliance, you can see the compliance report in Systems Manager Fleet Manager. In the Fleet Manager console, choose the node ID of your managed node. In the **General** menu, choose **Configuration compliance**. If you use Explorer to scan your managed instances for patch compliance, you can see the compliance report in both Explorer and Systems Manager OpsCenter.

	AWS-RunPatchBaseline	AWS-RunPatchBaselineAssociation
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. • describe-instance-patch-states • describe-instance-patch-states-for-patch-group • describe-patch-group-state	For processes that use AWS-RunPatchBaselineAssociation, you can use the following AWS CLI command to view summary information about patches on an instance. • list-compliance-items
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 noncompliant 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 info	SSM Command document for patching: AWS-RunPatchBaseline	SSM Command document for patching: AWS-RunPatchBaselineAssociation

For information about the various patch compliance states you might see reported, see Patch compliance state values

For information about remediating managed nodes that are out of patch compliance, see Patching noncompliant managed nodes.

Patch compliance state values

The information about patches for a managed node include a report of the state, or status, of each individual patch.

> **Tip**
>
> 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 and Ubuntu Server

For Debian Server 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 in the following repositories: .
>
> - Ubuntu Server 16.04 LTS: `xenial-security`
> - Ubuntu Server 18.04 LTS: `bionic-security`
> - Ubuntu Server 20.04 LTS: `focal-security`
> - Ubuntu Server 22.04 LTS (`jammy-security`)
> - Ubuntu Server 24.04 LTS (`noble-security`)
> - Ubuntu Server 25.04 (`plucky-security`)
> - `debian-security` (Debian Server

If you do select the **Include nonsecurity updates** check box, patches from other repositories are considered as well.

Patch state	Description	Compliance status
INSTALLED	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 `AWS-RunPatchBaseline` document was run on the managed node.	Compliant
INSTALLED_OTHER	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 InstallOverrideList operation. If you don't specify `Block` as the **Rejected patches** action, `INSTALLED_OTHER` patches also includes installed but rejected patches.	Compliant

Patch state	Description	Compliance status
INSTALLED_PENDING_REBOOT	INSTALLED_PENDING_REBOOT can mean either of two things: • The Patch Manager Install operation applied the patch to the managed node, but the node has not been rebooted since the patch was applied. This typically means the NoReboot option was selected for the RebootOption parameter when the AWS-RunPatchBaseline document was last run on the managed node. For more information, see Parameter name: RebootOption . • A patch was installed outside of Patch Manager since the last time the managed node was rebooted. In neither case does it mean that a patch with this status *requires* a reboot, only that the node hasn't been rebooted since the patch was installed.	Non-Compliant

Patch state	Description	Compliance status
INSTALLED_REJECTED	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
MISSING	Packages that are filtered through the baseline and not already installed.	Non-Compliant
FAILED	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 and Ubuntu Server, the rules for package classification into the different compliance states are described in the following table.

Patch state	Description	Compliance value
INSTALLED	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 `AWS-RunPatchBaseline` document was run on the node.	Compliant
INSTALLED_OTHER [1]	The patch isn't in the baseline, but it is installed on	Compliant

Patch state	Description	Compliance value
	the managed node. There are two possible reasons for this: 1. The patch might have been installed manually. 2. Linux only: The package might have been installed as a required *dependency* of a different, approved patch. If you specify `Allow as dependency` as the **Rejected patches** action, patches that are installed as dependencies are given the reporting status `INSTALLED_OTHER` . Note that Windows Server doesn't support the concept of patch dependencies. For information about how Patch Manager handles patches in the **Rejected patches** list on Windows Server, see Rejected patch list options in custom patch baselines.	

Patch state	Description	Compliance value
INSTALLED_REJECTED	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

Patch state	Description	Compliance value
INSTALLED_PENDING_REBOOT	INSTALLED_PENDING_REBOOT can mean either of two things: • The Patch Manager Install operation applied the patch to the managed node, but the node has not been rebooted since the patch was applied. This typically means the NoReboot option was selected for the RebootOption parameter when the AWS-RunPatchBaseline document was last run on the managed node. For more information, see Parameter name: RebootOption . • A patch was installed outside of Patch Manager since the last time the managed node was rebooted. In neither case does it mean that a patch with this status *requires* a reboot, only that the node hasn't been rebooted since the patch was installed.	Non-Compliant

Patch state	Description	Compliance value
MISSING	The patch is approved in the baseline, but it isn't installed on the managed node. If you configure the `AWS-RunPatchBaseline` document task to scan (instead of install), the system reports this status for patches that were located during the scan but haven't been installed.	Non-Compliant
FAILED	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.	Non-Compliant

Patch state	Description	Compliance value
NOT_APPLICABLE [1]	*This compliance state is reported only for Windows Server operating systems.* 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 Informati on Services (IIS) would show `NOT_APPLICABLE` if it was approved in the baseline, but the web service isn't installed on the managed node. A patch can also be marked `NOT_APPLICABLE` if it has been superseded by a subsequent update. This means that the later update is installed and the `NOT_APPLI CABLE` update is no longer required.	Not applicable

Patch state	Description	Compliance value
AVAILABLE_SECURITY_UPDATES	*This compliance state is reported only for Windows Server operating systems.* An available security update patch that is not approved by the patch baseline can have a compliance value or `Compliant` or `Non-Compliant`, as defined in a custom patch baseline. When you create or update a patch baseline, you choose the status you want to assign to security patches that are available but not approved because they don't meet the installation criteria specified in the patch baseline. For example, security patches that you might want installed can be skipped if you have specified a long period to wait after a patch is released before installation. If an update to the patch is released during your specified waiting period, the waiting period for installing the patch starts over. If the waiting period is too long, multiple versions of the patch could be released but never installed.	Compliant or Non-Compliant, depending on the option selected for available security updates. **Note** Using the console to create or update a patch baseline, you specify this option in the **Available security updates compliance status** field. Using the AWS CLI to run the create-patch-baseline or update-patch-baseline command, you specify this option in the `available-security-updates-compliance-status` parameter.

Patch state	Description	Compliance value
	For patch summary counts, when a patch is reported as `AvailableSecurityUpdate`, it will always be included in `AvailableSecurityUpdateCount`. If the baseline is configured to report these patches as `NonCompliant`, it is also included in `SecurityNonCompliantCount`. If the baseline is configured to report these patches as `Compliant`, they are not included in `SecurityNonCompliantCount`. These patches are always reported with an unspecified severity and are never included in the `CriticalNonCompliantCount`.	

[1] For patches with the state `INSTALLED_OTHER` and `NOT_APPLICABLE`, Patch Manager omits some data from query results based on the describe-instance-patches command, such as the values for `Classification` and `Severity`. This is done to help prevent exceeding the data limit for individual nodes in Inventory, a tool in AWS Systems Manager. To view all patch details, you can use the describe-available-patches command.

Patching noncompliant 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 tool in AWS Systems Manager, must run a `Scan and install` operation. (If your goal is only to identify

noncompliant managed nodes and not remediate them, run a `Scan` operation instead. For more information, see Identifying noncompliant managed nodes.)

Install patches using Systems Manager

You can choose from several tools to run a `Scan and install` operation:

- (Recommended) Configure a patch policy in Quick Setup, a tool in Systems Manager, that lets you install missing patches on a schedule for an entire organization, a subset of organizational units, or a single AWS account. For more information, see Configure patching for instances in an organization using a Quick Setup patch policy.
- Create a maintenance window that uses the Systems Manager document (SSM document) `AWS-RunPatchBaseline` in a Run Command task type. For information, see Tutorial: Create a maintenance window for patching using the console.
- Manually run `AWS-RunPatchBaseline` in a Run Command operation. For information, see Running commands from the console.

- Install patches on demand using the **Patch now** option. For information, see Patching managed nodes on demand.

Avoiding unintentional patch compliance data overwrites

If you have multiple types of operations in place to scan your instances for patch compliance, each scan overwrites the patch compliance data of previous scans. As a result, you might end up with unexpected results in your patch compliance data.

For example, suppose you create a patch policy that scans for patch compliance each day at 2 AM local time. That patch policy uses a patch baseline that targets patches with severity marked as `Critical`, `Important`, and `Moderate`. This patch baseline also specifies a few specifically rejected patches.

Also suppose that you already had a maintenance window set up to scan the same set of managed nodes each day at 4 AM local time, which you don't delete or deactivate. That maintenance window's task uses a different patch baseline, one that targets only patches with a `Critical` severity and doesn't exclude any specific patches.

When this second scan is performed by the maintenance window, the patch compliance data from the first scan is deleted and replaced with patch compliance from the second scan.

Therefore, we strongly recommend using only one automated method for scanning and installing in your patching operations. If you're setting up patch policies, you should delete or deactivate other methods of scanning for patch compliance. For more information, see the following topics:

- To remove a patching operation task from a maintenance window – Updating or deregistering maintenance window tasks using the console
- To delete a State Manager association – Deleting associations.

To deactivate daily patch compliance scans in a Host Management configuration, do the following in Quick Setup:

1. In the navigation pane, choose **Quick Setup**.
2. Select the Host Management configuration to update.
3. Choose **Actions, Edit configuration**.
4. Clear the **Scan instances for missing patches daily** check box.
5. Choose **Update**.

> **Note**
>
> Using the **Patch now** option to scan a managed node for compliance also results in an overwrite of patch compliance data.

Patching managed nodes on demand

Using the **Patch now** option in Patch Manager, a tool in 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 tool in 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.

> **Note**
>
> Patching on demand is supported for a single AWS account-AWS Region pair at a time. It can't be used with patching operations that are based on *patch policies*. We recommend using patch policies for keeping all your managed nodes in compliance. For more information about working with patch policies, see Patch policy configurations in Quick Setup.

Topics

- How 'Patch now' works
- Running 'Patch now'

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 Patch groups.

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%
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 SSM Command document for patching: `AWS-RunPatchBaselineWithHooks`.

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
Reboot if needed	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 after the patching operation is complete and instances are rebooted. Example use: Ensure that applications are running as expected after patching.	*Not available*
Do not reboot my instances	Same as above.	Run an SSM document at the end of the patching operation.	*Not available*	*Not available*

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.		
Schedule a reboot time	Same as above.	Same as for **Do not reboot my instances**.	*Not available*	Run an SSM document immediately after a scheduled reboot is complete. 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. 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 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.

   > **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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create the IAM service role required for Systems Manager in hybrid and multicloud environments. 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. (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 tool in 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 tool in 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
- Working with custom patch baselines
- Setting an existing patch baseline as the default

More info

- Patch baselines

Viewing AWS predefined patch baselines

Patch Manager, a tool in 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 Predefined and custom patch baselines.

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**.
3. In the patch baselines list, choose the baseline ID of one of the predefined patch baselines.

   -or-

If you are accessing Patch Manager for the first time in the current AWS Region, choose **Start with an overview**, choose the **Patch baselines** tab, and then 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.

4. In the **Approval rules** section, review the patch baseline configuration.
5. If the configuration is acceptable for your managed nodes, you can skip ahead to the procedure Creating and managing patch groups.

   -or-

   To create your own default patch baseline, continue to the topic Working with custom patch baselines.

Working with custom patch baselines

Patch Manager, a tool in 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 Predefined and custom patch baselines.

Topics

- Creating a custom patch baseline for Linux
- Creating a custom patch baseline for macOS
- Creating a custom patch baseline for Windows Server
- Updating or deleting a custom patch baseline

Creating a custom patch baseline for Linux

Use the following procedure to create a custom patch baseline for Linux managed nodes in Patch Manager, a tool in AWS Systems Manager.

For information about creating a patch baseline for macOS managed nodes, see Creating a custom patch baseline for macOS. For information about creating a patch baseline for Windows managed nodes, see Creating a custom patch baseline for Windows Server.

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**.
3. Choose the **Patch baselines** tab, and then choose **Create patch baseline**.

   -or-

   If you are accessing Patch Manager for the first time in the current AWS Region, choose **Start with an overview**, choose the **Patch baselines** tab, and then choose **Create patch baseline**.
4. For **Name**, enter a name for your new patch baseline, for example, `MyRHELPatchBaseline`.
5. (Optional) For **Description**, enter a description for this patch baseline.
6. For **Operating system**, choose an operating system, for example, `Red Hat Enterprise Linux`.
7. If you want to begin using this patch baseline as the default for the selected operating system as soon as you create it, check the box next to **Set this patch baseline as the default patch baseline for *`operating system name`* instances**.

   > **Note**
   >
   > This option is available only if you first accessed Patch Manager before the patch policies release on December 22, 2022.

For information about setting an existing patch baseline as the default, see Setting an existing patch baseline as the default.

8. In the **Approval rules for operating-systems** section, use the fields to create one or more auto-approval rules.

   - **Products**: 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

parameter in the `AWS-RunPatchBaseline` document. For more information, see SSM Command document for patching: `AWS-RunPatchBaseline`.

- **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 or last updated 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 or updated on or before that date. For example, if you specify July 7, 2023, no patches released or last updated on or after July 8, 2023, are installed automatically.
- (Optional) **Compliance reporting**: The severity level you want to assign to patches approved by the baseline, such as `Critical` or `High`.

> **Note**
>
> If you specify a compliance reporting level and the patch state of any approved patch is reported as `Missing`, then the patch baseline's overall reported compliance severity is the severity level you specified.

- **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.

For more information about working with approval rules in a custom patch baseline, see Custom baselines.

9. If you want to explicitly approve any patches in addition to those meeting your approval rules, do the following in the **Patch exceptions** section:

   - For **Approved patches**, enter a comma-separated list of the patches you want to approve.

     For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.
   - (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 **Include non-security updates** check box for these patches to be installed on your Linux operating system as well.

10. If you want to explicitly reject any patches that otherwise meet your approval rules, do the following in the **Patch exceptions** section:

    - For **Rejected patches**, enter a comma-separated list of the patches you want to reject.

      For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.
    - 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 by Patch Manager under any circumstances. If a package was installed before it was added to the **Rejected patches** list, or is installed outside of Patch Manager afterward, it's considered noncompliant with the patch baseline and its status is reported as *InstalledRejected*.

> **Note**
>
> Patch Manager searches for patch dependencies recursively.

11. (Optional) If you want to specify alternative patch repositories for different versions of an operating system, such as *AmazonLinux2016.03* and *AmazonLinux2017.09*, do the following for each product in the **Patch sources** section:

    - In **Name**, enter a name to help you identify the source configuration.
    - In **Product**, select the version of the operating systems the patch source repository is for, such as `RedhatEnterpriseLinux7.4`.
    - In **Configuration**, enter the value of the repository configuration to use in the appropriate format:

      Example for yum repositories

      ```
      [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).

      Examples for Ubuntu Server and Debian Server

      ```
      deb http://security.ubuntu.com/ubuntu jammy main
      ```

      ```
      deb https://site.example.com/debian distribution component1
      component2 component3
      ```

      Repo information for Ubuntu Server repositories must be specifed in a single line. For more examples and information, see jammy (5) sources.list.5.gz on the *Ubuntu Server Manuals* website and sources.list format on the *Debian Wiki*.

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).

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 operating system family it applies to, and the environment type. In this case, you could specify tags similar to the following key name/value pairs:

    - `Key=PatchSeverity,Value=Critical`
    - `Key=OS,Value=RHEL`
    - `Key=Environment,Value=Production`

13. Choose **Create patch baseline**.

Creating a custom patch baseline for macOS

Use the following procedure to create a custom patch baseline for macOS managed nodes in Patch Manager, a tool in AWS Systems Manager.

For information about creating a patch baseline for Windows Server managed nodes, see Creating a custom patch baseline for Windows Server. For information about creating a patch baseline for Linux managed nodes, see Creating a custom patch baseline for Linux.

> **Note**
>
> macOS is not supported in all AWS Regions. For more information about Amazon EC2 support for macOS, see Amazon EC2 Mac instances in the *Amazon EC2 User Guide*.

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**.
3. Choose the **Patch baselines** tab, and then choose **Create patch baseline**.

   -or-

   If you are accessing Patch Manager for the first time in the current AWS Region, choose **Start with an overview**, choose the **Patch baselines** tab, and then choose **Create patch baseline**.
4. For **Name**, enter a name for your new patch baseline, for example, `MymacOSPatchBaseline`.
5. (Optional) For **Description**, enter a description for this patch baseline.
6. For **Operating system**, choose macOS.
7. 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**.

   > **Note**
   >
   > This option is available only if you first accessed Patch Manager before the patch policies release on December 22, 2022.
   > For information about setting an existing patch baseline as the default, see Setting an existing patch baseline as the default.
8. In the **Approval rules for operating-systems** section, use the fields to create one or more auto-approval rules.

   - **Products**: The version of the operating systems the approval rule applies to, such as `BigSur11.3.1` or `Ventura13.7`. 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 `Critical` or `High`.

> **Note**
>
> If you specify a compliance reporting level and the patch state of any approved patch is reported as `Missing`, then the patch baseline's overall reported compliance severity is the severity level you specified.

For more information about working with approval rules in a custom patch baseline, see Custom baselines.

9. If you want to explicitly approve any patches in addition to those meeting your approval rules, do the following in the **Patch exceptions** section:

   - For **Approved patches**, enter a comma-separated list of the patches you want to approve.

     For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.

   - (Optional) For **Approved patches compliance level**, assign a compliance level to the patches in the list.

10. If you want to explicitly reject any patches that otherwise meet your approval rules, do the following in the **Patch exceptions** section:

    - For **Rejected patches**, enter a comma-separated list of the patches you want to reject.

      For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.

    - 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 by Patch Manager under any circumstances. If a package

was installed before it was added to the **Rejected patches** list, or is installed outside of Patch Manager afterward, it's considered noncompliant with the patch baseline and its status is reported as *InstalledRejected*.

11. (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`

12. Choose **Create patch baseline**.

Creating a custom patch baseline for Windows Server

Use the following procedure to create a custom patch baseline for Windows managed nodes in Patch Manager, a tool in AWS Systems Manager.

For information about creating a patch baseline for Linux managed nodes, see Creating a custom patch baseline for Linux. Fo information about creating a patch baseline for macOS managed nodes, see Creating a custom patch baseline for macOS.

For an example of creating a patch baseline that is limited to installing Windows Service Packs only, see Tutorial: Create a patch baseline for installing Windows Service Packs using the console.

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**.
3. Choose the **Patch baselines** tab, and then choose **Create patch baseline**.

   -or-

If you are accessing Patch Manager for the first time in the current AWS Region, choose **Start with an overview**, choose the **Patch baselines** tab, and then choose **Create patch baseline**.

4. For **Name**, enter a name for your new patch baseline, for example, `MyWindowsPatchBaseline`.
5. (Optional) For **Description**, enter a description for this patch baseline.
6. For **Operating system**, choose `Windows`.
7. For **Available security updates compliance status**, choose the status you want to assign to security patches that are available but not approved because they don't meet the installation criteria specified in the patch baseline, **Non-Compliant** or **Compliant**.

   Example scenario: Security patches that you might want installed can be skipped if you have specified a long period to wait after a patch is released before installation. If an update to the patch is released during your specified waiting period, the waiting period for installing the patch starts over. If the waiting period is too long, multiple versions of the patch could be released but never installed.
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**.

   > **Note**
   >
   > This option is available only if you first accessed Patch Manager before the patch policies release on December 22, 2022.
   > For information about setting an existing patch baseline as the default, see Setting an existing patch baseline as the default.
9. In the **Approval rules for operating systems** section, use the fields to create one or more auto-approval rules.

   - **Products**: 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 Tutorial: Create a patch baseline for installing Windows Service Packs using the console.

- **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 or updated 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 or updated on or before that date. For example, if you specify July 7, 2023, no patches released or last updated on or after July 8, 2023, are installed automatically.
- (Optional) **Compliance reporting**: The severity level you want to assign to patches approved by the baseline, such as `High`.

> **Note**
>
> If you specify a compliance reporting level and the patch state of any approved patch is reported as `Missing`, then the patch baseline's overall reported compliance severity is the severity level you specified.

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`.
- **Products**: 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 or updated 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 or updated on or before that date. For example, if you specify July 7, 2023, no patches released or last updated on or after July 8, 2023, are installed automatically.
- (Optional) **Compliance reporting**: The severity level you want to assign to patches approved by the baseline, such as `Critical` or `High`.

> **Note**
>
> If you specify a compliance reporting level and the patch state of any approved patch is reported as `Missing`, then the patch baseline's overall reported compliance severity is the severity level you specified.

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.

      For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.

- (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.

      For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.

    - For **Rejected patches action**, select the action for Patch Manager to take on patches included in the **Rejected patches** list.
      - **Allow as dependency**: Windows Server doesn't support the concept of package dependencies. If a package in the **Rejected patches** list and already installed on the node, its status is reported as `INSTALLED_OTHER`. Any package not already installed on the node is skipped.
      - **Block**: Packages in the **Rejected patches** list aren't installed by Patch Manager under any circumstances. If a package was installed before it was added to the **Rejected patches** list, or is installed outside of Patch Manager afterward, it's considered noncompliant with the patch baseline and its status is reported as `INSTALLED_REJECTED`.

      For more information about rejected package actions, see Rejected patch list options in custom patch baselines.

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

You can update or delete a custom patch baseline that you have created in Patch Manager, a tool in 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

Follow these steps to update or delete a patch baseline.

> **Important**
>
> Use caution when deleting a custom patch baseline that might be used by a patch policy configuration in Quick Setup.
> If you are using a patch policy configuration in Quick Setup, updates you make to custom patch baselines are synchronized with Quick Setup once an hour.
> If a custom patch baseline that was referenced in a patch policy is deleted, a banner displays on the Quick Setup **Configuration details** page for your patch policy. The banner informs you that the patch policy references a patch baseline that no longer exists, and that subsequent patching operations will fail. In this case, return to the Quick Setup **Configurations** page, select the Patch Manager configuration , and choose **Actions**, **Edit configuration**. The deleted patch baseline name is highlighted, and you must select a new patch baseline for the affected operating system.

To update or delete a patch baseline

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, 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.

Setting an existing patch baseline as the default

> **Important**
>
> Any default patch baseline selections you make here do not apply to patching operations that are based on a patch policy. Patch policies use their own patch baseline specifications. For more information about patch policies, see Patch policy configurations in Quick Setup.

When you create a custom patch baseline in Patch Manager, a tool in 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.

You can also set an existing patch baseline as the default for an operating system type.

> **Note**
>
> The steps you follow depend on whether you first accessed Patch Manager before or after the patch policies release on December 22, 2022. If you used Patch Manager before that date, you can use the console procedure. Otherwise, use the AWS CLI procedure. The **Actions** menu referenced in the console procedure is not displayed in Regions where Patch Manager wasn't used before the patch policies release.

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**.
3. Choose the **Patch baselines** tab.
4. 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.

   The **Default baseline** column indicates which baselines are currently set as the defaults.
5. In the **Actions** menu, choose **Set default patch baseline**.

   > **Important**
   >
   > The **Actions** menu is not available if you didn't work with Patch Manager in the current AWS account and Region before December 22, 2022. See the **Note** earlier in this topic for more information.
6. In the confirmation dialog box, choose **Set default**.

To set a patch baseline as the default (AWS CLI)

1. Run the describe-patch-baselines command to view a list of available patch baselines and their IDs and Amazon Resource Names (ARNs).

   ```
   aws ssm describe-patch-baselines
   ```
2. Run the register-default-patch-baseline command to set a baseline as the default for the operating system it's associated with. Replace *baseline-id-or-ARN* with the ID of the custom patch baseline or predefined baseline to use.

   Linux & macOS

   ```
   aws ssm register-default-patch-baseline \
       --baseline-id baseline-id-or-ARN
   ```

   The following is an example of a setting a custom baseline as the default.

```
aws ssm register-default-patch-baseline \
    --baseline-id pb-abc123cf9bEXAMPLE
```

The following is an example of a setting a predefined baseline managed by AWS as the default.

```
aws ssm register-default-patch-baseline \
    --baseline-id arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-0574b43a65ea646e
```

Windows Server

```
aws ssm register-default-patch-baseline ^
    --baseline-id baseline-id-or-ARN
```

The following is an example of a setting a custom baseline as the default.

```
aws ssm register-default-patch-baseline ^
    --baseline-id pb-abc123cf9bEXAMPLE
```

The following is an example of a setting a predefined baseline managed by AWS as the default.

```
aws ssm register-default-patch-baseline ^
    --baseline-id arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-071da192df1226b63
```

Viewing available patches

With Patch Manager, a tool in 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**.
3. Choose the **Patches** tab.

   -or-

   If you are accessing Patch Manager for the first time in the current AWS Region, choose **Start with an overview**, and then choose the **Patches** tab.

   > **Note**
   >
   > For Windows Server, the **Patches** tab displays updates that are available from Windows Server Update Service (WSUS).

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 and managing patch groups

If you are *not* using patch policies in your operations, you can organize your patching efforts by adding managed nodes to patch groups by using tags.

> **Note**
>
> Patch groups are not used in patching operations that are based on *patch policies*. For information about working with patch policies, see Patch policy configurations in Quick Setup.
>
> Patch group functionality is not supported in the console for account-Region pairs that did not already use patch groups before patch policy support was released on December 22, 2022. Patch group functionality is still available in account-Region pairs that began using patch groups before this date.

To use tags in patching operations, you must apply the tag key `Patch Group` or `PatchGroup` to your managed nodes. You must also specify the name that you want to give the patch group as the value of the tag. You can specify any tag value, but the tag key must be `Patch Group` or `PatchGroup`.

`PatchGroup` (without a space) is required if you have allowed tags in EC2 instance metadata.

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. For more information about patch groups, see Patch groups.

Complete the tasks in this topic to prepare your managed nodes for patching using tags with your nodes and patch baseline. Task 1 is required only if you are patching Amazon EC2 instances. Task 2 is required only if you are patching non-EC2 instances in a hybrid and multicloud environment. Task 3 is required for all managed nodes.

> **Tip**
>
> You can also add tags to managed nodes using the AWS CLI command `add-tags-to-resource` or the Systems Manager API operation ssm-agent-minimum-s3-permissions-required`AddTagsToResource`.

Tasks

- Task 1: Add EC2 instances to a patch group using tags
- Task 2: Add managed nodes to a patch group using tags

- Task 3: Add a patch group to a patch baseline

Task 1: Add EC2 instances to a patch group using tags

You can add tags to EC2 instances using the Systems Manager console or the Amazon EC2 console. This task is required only if you are patching Amazon EC2 instances.

> **Important**
>
> You can't apply the `Patch Group` tag (with a space) to an Amazon EC2 instance if the **Allow tags in instance metadata** option is enabled on the instance. Allowing tags in instance metadata prevents tag key names from containing spaces. If you have allowed tags in EC2 instance metadata, you must use the tag key `PatchGroup` (without a space).

Option 1: To add EC2 instances 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**.
3. In the **Managed nodes** list, choose the ID of a managed EC2 instance that you want to configure for patching. Node IDs for EC2 instances begin with `i-`.

   > **Note**
   >
   > When using the Amazon EC2 console and AWS CLI, it's possible to apply `Key = Patch Group` or `Key = PatchGroup` 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 for troubleshooting tips.

4. Choose the **Tags** tab, then choose **Edit**.
5. In the left column, enter **`Patch Group`** or **`PatchGroup`**. If you have allowed tags in EC2 instance metadata, you must use `PatchGroup` (without a space).
6. In the right column, enter a tag value to serve as the name for the patch group.
7. Choose **Save**.
8. Repeat this procedure to add other EC2 instances to the same patch group.

Option 2: 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**, **Manage tags**.
4. Choose **Add new tag**.
5. For **Key**, enter **`Patch Group`** or **`PatchGroup`**. If you have allowed tags in EC2 instance metadata, you must use `PatchGroup` (without a space).
6. For **Value**, enter a value to serve as the name for the patch group.
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

Follow the steps in this topic to add tags to AWS IoT Greengrass core devices and non-EC2 hybrid-activated managed nodes (mi-*). This task is required only if you are patching non-EC2 instances in a hybrid and multicloud environment.

> **Note**
>
> You can't add tags for non-EC2 managed nodes using the Amazon EC2 console.

To add non-EC2 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**.
3. In the **Managed nodes** list, choose the name of the 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 for troubleshooting tips.

4. Choose the **Tags** tab, then choose **Edit**.
5. In the left column, enter **`Patch Group`** or **`PatchGroup`**. If you have allowed tags in EC2 instance metadata, you must use `PatchGroup` (without a space).
6. In the right column, enter a tag value to serve as the name for the patch group.
7. Choose **Save**.
8. 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. This task is required whether you are patching EC2 instances, non-EC2 managed nodes, or both.

For more information about patch groups, see Patch groups.

> **Note**
>
> The steps you follow depend on whether you first accessed Patch Manager before or after the patch policies release on December 22, 2022.

To add a patch group to a patch baseline (Systems Manager 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. If you're accessing Patch Manager for the first time in the current AWS Region and the Patch Manager start page opens, choose **Start with an overview**.
4. Choose the **Patch baselines** tab, and then in the **Patch baselines** list, choose the name of the patch baseline that you want to configure for your patch group.

   If you didn't first access Patch Manager until after the patch policies release, you must choose a custom baseline that you have created.
5. If the **Baseline ID** details page includes an **Actions** menu, do the following:

   - Choose **Actions**, then **Modify patch groups**.

- Enter the tag *value* you added to your managed nodes in Task 2: Add managed nodes to a patch group using tags, then choose **Add**.

If the **Baseline ID** details page does *not* include an **Actions** menu, patch groups can't be configured in the console. Instead, you can do either of the following:

- (Recommended) Set up a patch policy in Quick Setup, a tool in AWS Systems Manager, to map a patch baseline to one or more EC2 instances.

  For more information, see Using Quick Setup patch policies and Automate organization-wide patching using a Quick Setup patch policy.
- Use the register-patch-baseline-for-patch-group command in the AWS Command Line Interface (AWS CLI) to configure a patch group.

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 tool in AWS Systems Manager, and Security Hub, you can send findings about noncompliant nodes 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 patch-related findings in its analysis of your security posture.

The information in the following topics applies no matter which method or type of configuration you are using for your patching operations:

- A patch policy configured in Quick Setup
- A Host Management option configured in Quick Setup
- A maintenance window to run a patch `Scan` or `Install` task
- An on-demand **Patch now** operation

Contents

- How Patch Manager sends findings to Security Hub

- Types of findings that Patch Manager sends
- Latency for sending findings
- Retrying when Security Hub isn't available
- Viewing findings in Security Hub
- Typical finding from Patch Manager
- Turning on and configuring the integration
- How to stop sending findings

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 tools 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, tools in 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.

> **Note**
>
> After reporting a noncompliant node to Security Hub. Patch Manager doesn't send an update to Security Hub after the node is made compliant. You can manually resolve findings in Security Hub after the required patches have been applied to the managed node.

For more information about compliance definitions, see Patch compliance state values. 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.

Viewing findings in Security Hub

This procedure describes how to view findings in Security Hub about managed nodes in your fleet that are out of patch compliance.

To review Security Hub findings for patch compliance

1. Sign in to the AWS Management Console and open the AWS Security Hub console at https://console.aws.amazon.com/securityhub/.

2. In the navigation pane, choose **Findings**.
3. Choose the **Add filters** () box.
4. In the menu, under **Filters**, choose **Product name**.
5. In the dialog box that opens, choose **is** in the first field and then enter **`Systems Manager Patch Manager`** in the second field.
6. Choose **Apply**.
7. Add any additional filters you want to help narrow down your results.
8. In the list of results, choose the title of a finding you want more information about.

   A pane opens on the right side of the screen with more details about the resource, the issue discovered, and a recommended remediation.

   > **Important**
   >
   > At this time, Security Hub reports the resource type of all managed nodes as `EC2 Instance`. This includes on-premises servers and virtual machines (VMs) that you have registered for use with Systems Manager.

Severity classifications

The list of findings for **`Systems Manager Patch Manager`** includes a report of the severity of the finding. **Severity** levels include the following, from lowest to highest:

- **INFORMATIONAL** – No issue was found.
- **LOW** – The issue does not require remediation.
- **MEDIUM** – The issue must be addressed but is not urgent.
- **HIGH** – The issue must be addressed as a priority.
- **CRITICAL** – The issue must be remediated immediately to avoid escalation.

Severity is determined by the most severe noncompliant package on an instance. Because you can have multiple patch baselines with multiple severity levels, the highest severity is reported out of all the noncompliant packages. For example, suppose you have two noncompliant packages

where the severity of package A is "Critical" and the severity of package B is "Low". "Critical" will be reported as the severity.

Note that the severity field correlates directly with the Patch Manager `Compliance` field. This is a field that you set assign to individual patches that match the rule. Because this `Compliance` field is assigned to individual patches, it is not reflected at the Patch Summary level.

Related content

- Findings in the *AWS Security Hub User Guide*
- Multi-Account patch compliance with Patch Manager and Security Hub in the *AWS Management & Governance Blog*

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-02573cafcfEXAMPLE/
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/fleet-manager/
i-02573cafcfEXAMPLE/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-02573cafcfEXAMPLE/
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-02573cafcfEXAMPLE",
      "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**.
2. Choose the **Settings** tab.

   -or-

   If you are accessing Patch Manager for the first time in the current AWS Region, choose **Start with an overview**, and then 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 resources using the 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 tool in AWS Systems Manager.

For an illustration of using the AWS CLI to patch a server environment by using a custom patch baseline, see Tutorial: Patch a server environment using the AWS CLI.

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
- AWS CLI commands for patch groups
- AWS CLI commands for viewing patch summaries and details
- AWS CLI commands for scanning and patching managed nodes

AWS CLI commands for patch baselines

Sample commands for patch baselines

- Create a patch baseline
- Create a patch baseline with custom repositories for different OS versions
- Update a patch baseline
- Rename a patch baseline
- Delete a patch baseline
- List all patch baselines
- List all AWS-provided patch baselines
- List my patch baselines
- Display a patch baseline
- Get the default patch baseline
- Set a custom patch baseline as the default
- Reset an AWS patch baseline as the default
- Tag a patch baseline
- List the tags for a patch baseline
- Remove a tag from a patch baseline

Create a patch baseline

The following command creates a patch baseline that approves all critical and important security updates for Windows Server 2012 R2 5 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 Server

```
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 2",
    "Name": "Amazon-Linux-2",
    "OperatingSystem": "AMAZON_LINUX_2",
    "ApprovalRules": {
        "PatchRules": [
            {
                "ApproveAfterDays": 7,
                "EnableNonSecurity": true,
                "PatchFilterGroup": {
                    "PatchFilters": [
                        {
                            "Key": "SEVERITY",
                            "Values": [
                                "Important",
                                "Critical"
                            ]
                        },
                        {
                            "Key": "CLASSIFICATION",
                            "Values": [
                                "Security",
                                "Bugfix"
                            ]
                        },
                        {
```

```
                            "Key": "PRODUCT",
                            "Values": [
                                "AmazonLinux2"
                            ]
                        }
                    ]
                }
            }
        ]
    },
    "Sources": [
        {
            "Name": "My-AL2",
            "Products": [
                "AmazonLinux2"
            ],
            "Configuration": "[amzn-main] \nname=amzn-main-Base
\nmirrorlist=http://repo./$awsregion./$awsdomain//$releasever/main/
mirror.list //nmirrorlist_expire=300//nmetadata_expire=300 \npriority=10
 \nfailovermethod=priority \nfastestmirror_enabled=0 \ngpgcheck=1
 \ngpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-ga \nenabled=1 \nretries=3
 \ntimeout=5\nreport_instanceid=yes"
        }
    ]
}
```

2. In the directory where you saved the file, run the following command.

```
aws ssm create-patch-baseline --cli-input-json file://my-patch-repository.json
```

The system returns information like the following.

```
{
    "BaselineId": "pb-0c10e65780EXAMPLE"
}
```

Update a patch baseline

The following command adds two patches as rejected and one patch as approved to an existing patch baseline.

For information about accepted formats for lists of approved patches and rejected patches, see Package name formats for approved and rejected patch lists.

Linux & macOS

```
aws ssm update-patch-baseline \
    --baseline-id pb-0c10e65780EXAMPLE \
    --rejected-patches "KB2032276" "MS10-048" \
    --approved-patches "KB2124261"
```

Windows Server

```
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 Server

```
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 Server

```
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 Server

```
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 Server

```
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 Server

```
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 Server

```
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 Server

```
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 Server

```
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 Server

```
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
- Register a patch group "web servers" with a patch baseline
- Register a patch group "Backend" with the AWS-provided patch baseline
- Display patch group registrations
- Deregister a patch group from a patch baseline

Create a patch group

> **Note**
>
> Patch groups are not used in patching operations that are based on *patch policies*. For information about working with patch policies, see Patch policy configurations in Quick Setup.

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` or `PatchGroup`. If you have allowed tags in EC2 instance metadata, you must use `PatchGroup` (without a space). You can specify any tag value, but the tag key must be `Patch Group` or `PatchGroup`. For more information about patch groups, see Patch groups.

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` or `Key = PatchGroup` 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` or `Key = PatchGroup` tag, see Troubleshooting managed node availability for troubleshooting tips.

Run the following command to add the `PatchGroup` tag to an EC2 instance.

```
aws ec2 create-tags --resources "i-1234567890abcdef0" --tags
 "Key=PatchGroup,Value=GroupValue"
```

Task 2: Add managed nodes to a patch group using tags

Run the following command to add the `PatchGroup` tag to a managed node.

Linux & macOS

```
aws ssm add-tags-to-resource \
    --resource-type "ManagedInstance" \
    --resource-id "mi-0123456789abcdefg" \
    --tags "Key=PatchGroup,Value=GroupValue"
```

Windows Server

```
aws ssm add-tags-to-resource ^
    --resource-type "ManagedInstance" ^
    --resource-id "mi-0123456789abcdefg" ^
    --tags "Key=PatchGroup,Value=GroupValue"
```

Task 3: Add a patch group to a patch baseline

Run the following command to associate a `PatchGroup` 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 Server

```
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 Server

```
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 Server

```
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 Server

```
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
- Get all patches for AmazonLinux2018.03 that have a Classification SECURITY and Severity of Critical
- Get all patches for Windows Server 2012 that have a MSRC severity of Critical
- Get all available patches
- Get patch summary states per-managed node
- Get patch compliance details for a managed node
- View patching compliance results (AWS CLI)

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 Server

```
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,
           "MsrcClassification":"Critical",
           "Language":"All",
           "KbNumber":"KB2876331",
           "MsrcNumber":"MS13-089",
           "Id":"e74ccc76-85f0-4881-a738-59e9fc9a336d"
        }
     },
     {
        "PatchStatus":{
           "ApprovalDate":1428858000.0,
           "DeploymentStatus":"APPROVED"
        },
        "Patch":{
           "ContentUrl":"https://support.microsoft.com/en-us/kb/2919355",
           "ProductFamily":"Windows",
           "Product":"WindowsServer2012R2",
           "Vendor":"Microsoft",
           "Description":"Windows Server 2012 R2 Update is a cumulative
              set of security updates, critical updates and updates. You
              must install Windows Server 2012 R2 Update to ensure that
              your computer can continue to receive future Windows Updates,
              including security updates. For a complete listing of the
              issues that are included in this update, see the associated
```

```
               Microsoft Knowledge Base article for more information. After
               you install this item, you may have to restart your computer.",
            "Classification":"SecurityUpdates",
            "Title":"Windows Server 2012 R2 Update (KB2919355)",
            "ReleaseDate":1428426000.0,
            "MsrcClassification":"Critical",
            "Language":"All",
            "KbNumber":"KB2919355",
            "MsrcNumber":"MS14-018",
            "Id":"8452bac0-bf53-4fbd-915d-499de08c338b"
         }
      }
     ---output truncated---
```

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 Server

```
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 Server

```
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,
         "MsrcClassification":"Critical",
```

```
         "Language":"All",
         "KbNumber":"KB2727528",
         "MsrcNumber":"MS12-072",
         "Id":"1eb507be-2040-4eeb-803d-abc55700b715"
      },
      {
         "ContentUrl":"https://support.microsoft.com/en-us/kb/2729462",
         "ProductFamily":"Windows",
         "Product":"WindowsServer2012",
         "Vendor":"Microsoft",
         "Description":"A security issue has been identified that could
           allow an unauthenticated remote attacker to compromise your
           system and gain control over it. You can help protect your
           system by installing this update from Microsoft. After you
           install this update, you may have to restart your system.",
         "Classification":"SecurityUpdates",
         "Title":"Security Update for Microsoft .NET Framework 3.5 on
           Windows 8 and Windows Server 2012 for x64-based Systems (KB2729462)",
         "ReleaseDate":1352829600.0,
         "MsrcClassification":"Critical",
         "Language":"All",
         "KbNumber":"KB2729462",
         "MsrcNumber":"MS12-074",
         "Id":"af873760-c97c-4088-ab7e-5219e120eab4"
      }

---output truncated---
```

Get all available patches

```
aws ssm describe-available-patches --region us-east-2
```

The system returns information like the following.

```
{
   "NextToken":"--token string truncated--",
   "Patches":[
      {
            "Classification": "SecurityUpdates",
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4074588",
            "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.",
            "Id": "11adea10-0701-430e-954f-9471595ae246",
            "KbNumber": "KB4074588",
            "Language": "All",
            "MsrcNumber": "",
            "MsrcSeverity": "Critical",
            "Product": "WindowsServer2016",
            "ProductFamily": "Windows",
            "ReleaseDate": 1518548400,
            "Title": "2018-02 Cumulative Update for Windows Server 2016 (1709) for x64-
based
            Systems (KB4074588)",
            "Vendor": "Microsoft"
        },
        {
            "Classification": "SecurityUpdates",
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4074590",
            "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.",
            "Id": "f5f58231-ac5d-4640-ab1b-9dc8d857c265",
            "KbNumber": "KB4074590",
            "Language": "All",
            "MsrcNumber": "",
            "MsrcSeverity": "Critical",
            "Product": "WindowsServer2016",
            "ProductFamily": "Windows",
            "ReleaseDate": 1518544805,
            "Title": "2018-02 Cumulative Update for Windows Server 2016 for x64-based
            Systems (KB4074590)",
            "Vendor": "Microsoft"
        }
      ---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 Server

```
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-1eae-411b-8aa7-973e6EXAMPLE",
```

```
            "InstalledCount": 36,
            "InstalledOtherCount": 396,
            "InstalledPendingRebootCount": 0,
            "InstalledRejectedCount": 0,
            "MissingCount": 3,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": -1,
            "NotApplicableCount": 420,
            "OperationStartTime": "2020-01-24T12:37:34-08:00",
            "OperationEndTime": "2020-01-24T12:37:37-08:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot"
        }
    ---output truncated---
```

Get patch compliance details for 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-02573cafcfEXAMPLE` or `mi-0282f7c436EXAMPLE`.

The systems returns information like the following.

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-02573cafcfEXAMPLE",
            "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 you 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[*].{Instance: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[*].{Instance:InstanceId,
 InstalledPendingRebootCount:InstalledPendingRebootCount}'
```

The system returns information like the following.

```
1       i-02573cafcfEXAMPLE
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 section to view information about patch status and compliance.

Sample commands

- Scan managed nodes for patch compliance (AWS CLI)
- Install patches on managed nodes (AWS CLI)

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-02573cafcfEXAMPLE,i-0471e04240EXAMPLE' \
    --parameters 'Operation=Scan' \
    --timeout-seconds 600
```

Windows Server

```
aws ssm send-command ^
    --document-name "AWS-RunPatchBaseline" ^
    --targets Key=InstanceIds,Values="i-02573cafcfEXAMPLE,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-02573cafcfEXAMPLE,
                     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:PatchGroup',Values='Web servers' \
    --parameters 'Operation=Scan' \
    --timeout-seconds 600
```

Windows Server

```
aws ssm send-command ^
    --document-name "AWS-RunPatchBaseline" ^
    --targets Key="tag:PatchGroup",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:PatchGroup",
                "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 SSM Command document for patching: `AWS-RunPatchBaseline`.

Linux & macOS

```
aws ssm send-command \
    --document-name 'AWS-RunPatchBaseline' \
    --targets Key=InstanceIds,Values='i-02573cafcfEXAMPLE,i-0471e04240EXAMPLE' \
    --parameters 'Operation=Install' \
    --timeout-seconds 600
```

Windows Server

```
aws ssm send-command ^
```

```
    --document-name "AWS-RunPatchBaseline" ^
    --targets Key=InstanceIds,Values="i-02573cafcfEXAMPLE,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-02573cafcfEXAMPLE,
                     i-0471e04240EXAMPLE"
                ]
            }
        ],
        "RequestedDateTime": 1621953101.791,
        "Status": "Pending",
        "StatusDetails": "Pending",
        "TimeoutSeconds": 600,

    ---output truncated---

    }
}
```

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:PatchGroup',Values='Web servers' \
    -parameters 'Operation=Install' \
    --timeout-seconds 600
```

Windows Server

```
aws ssm send-command ^
    --document-name "AWS-RunPatchBaseline" ^
    --targets Key="tag:PatchGroup",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:PatchGroup",
                "Values": [
                    "Web servers"
                ]
            }
        ],
        "RequestedDateTime": 1621953207.865,
        "Status": "Pending",
        "StatusDetails": "Pending",
```

```
        "TimeoutSeconds": 600,

    ---output truncated---

    }
}
```

AWS Systems Manager Patch Manager tutorials

The tutorials in this section demonstrate how to use Patch Manager, a tool in AWS Systems Manager, for several patching scenarios.

Topics

- Tutorial: Create a patch baseline for installing Windows Service Packs using the console
- Tutorial: Update application dependencies, patch a managed node, and perform an application-specific health check using the console
- Tutorial: Patch a server environment using the AWS CLI

Tutorial: Create a patch baseline for installing Windows Service Packs using the 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 tool in 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 create a patch baseline for installing Windows Service Packs (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. Choose the **Patch baselines** tab, and then 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**.

> **Note**
>
> This option is available only if you first accessed Patch Manager before the patch policies release on December 22, 2022.
> For information about setting an existing patch baseline as the default, see Setting an existing patch baseline as the default.

8. In the **Approval rules for operating systems** section, use the fields to create one or more auto-approval rules.

- **Products**: 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 or updated 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 or updated on or before that date. For example, if you specify July 7, 2023, no patches released or last updated on or after July 8, 2023, 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 you specify a compliance reporting level and the patch state of any approved Service Pack is reported as `Missing`, then the patch baseline's overall reported compliance severity is the severity level you specified.

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**.

Tutorial: Update application dependencies, patch a managed node, and perform an application-specific health check using the console

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 tutorial 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 might 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 tutorial 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 tool in 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 document content.

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 `NodeJSApp0nExitPatch`. 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 tool in 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 Understanding targets and rate controls in State Manager associations.
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 Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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**.

Tutorial: Patch a server environment using the 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 Updating the SSM Agent using Run Command.
- Configure roles and permissions for Maintenance Windows, a tool in AWS Systems Manager. For more information, see Setting up Maintenance Windows.
- Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

  For information, see Installing or updating the latest version of the AWS CLI.

To configure Patch Manager and patch managed nodes (command line)

1. Run the following command to create a patch baseline for Windows named `Production-Baseline`. This patch baseline approves patches for a production environment 7 days after they're released or last updated. 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,Impor
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,ServicePacks,UpdateRollups,CriticalU
 \
    --description "Baseline containing all updates approved for production
 systems"
```

Windows Server

```
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,Impor
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,ServicePacks,UpdateRollups,CriticalU
 ^
    --description "Baseline containing all updates approved for production
 systems"
```

The system returns information like the following.

```
{
   "BaselineId":"pb-0c10e65780EXAMPLE"
}
```

2. Run the following commands to register the "Production-Baseline" patch baseline for two patch groups. The groups are named "Database Servers" and "Front-End Servers".

   Linux & macOS

   ```
   aws ssm register-patch-baseline-for-patch-group \
       --baseline-id pb-0c10e65780EXAMPLE \
       --patch-group "Database Servers"
   ```

   Windows Server

   ```
   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 Server

```
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 Server

```
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 Server

```
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:PatchGroup,Values=Database Servers" \
       --owner-information "Database Servers" \
       --resource-type "INSTANCE"
   ```

   Windows Server

   ```
   aws ssm register-target-with-maintenance-window ^
       --window-id mw-0c50858d01EXAMPLE ^
       --targets "Key=tag:PatchGroup,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:PatchGroup,Values=Front-End Servers" \
   --owner-information "Front-End Servers" \
   --resource-type "INSTANCE"
   ```

Windows Server

```
aws ssm register-target-with-maintenance-window ^
    --window-id mw-9a8b7c6d5eEXAMPLE ^
    --targets "Key=tag:PatchGroup,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 Server

```
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 Server

```
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 Server

```
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
}
```

7. 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 Server

   ```
   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 resources using the AWS CLI.

Troubleshooting Patch Manager

Use the following information to help you troubleshoot problems with Patch Manager, a tool in AWS Systems Manager.

Topics

- Issue: "Invoke-PatchBaselineOperation : Access Denied" error or "Unable to download file from S3" error for baseline_overrides.json
- Issue: Patching fails without an apparent cause or error message
- Issue: Unexpected patch compliance results
- Errors when running AWS-RunPatchBaseline on Linux
- Errors when running AWS-RunPatchBaseline on Windows Server
- Using AWS Support Automation runbooks
- Contacting AWS Support

Issue: "Invoke-PatchBaselineOperation : Access Denied" error or "Unable to download file from S3" error for `baseline_overrides.json`

Problem: When the patching operations specified by your patch policy run, you receive an error similar to the following example.

Example error on Windows Server

```
----------ERROR-------
Invoke-PatchBaselineOperation : Access Denied
At C:\ProgramData\Amazon\SSM\InstanceData\i-02573cafcfEXAMPLE\document\orchestr
ation\792dd5bd-2ad3-4f1e-931d-abEXAMPLE\PatchWindows\_script.ps1:219 char:13
+ $response = Invoke-PatchBaselineOperation -Operation Install -Snapsho ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+ CategoryInfo : OperationStopped: (Amazon.Patch.Ba...UpdateOpera
tion:InstallWindowsUpdateOperation) [Invoke-PatchBaselineOperation], Amazo
nS3Exception
```

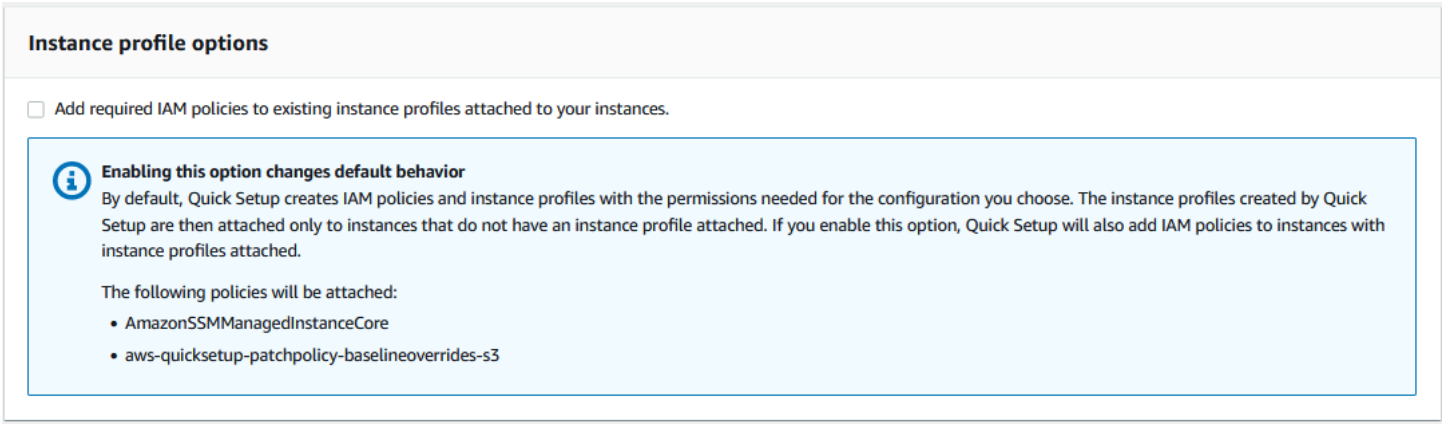
```
+ FullyQualifiedErrorId : PatchBaselineOperations,Amazon.Patch.Baseline.Op
erations.PowerShellCmdlets.InvokePatchBaselineOperation
failed to run commands: exit status 0xffffffff
```

Example error on Linux

```
[INFO]: Downloading Baseline Override from s3://aws-quicksetup-
patchpolicy-123456789012-abcde/baseline_overrides.json
[ERROR]: Unable to download file from S3: s3://aws-quicksetup-
patchpolicy-123456789012-abcde/baseline_overrides.json.
[ERROR]: Error loading entrance module.
```

Cause: You created a patch policy in Quick Setup, and some of your managed nodes already had an instance profile attached (for EC2 instances) or a service role attached (for non-EC2 machines).

However, as shown in the following image, you didn't select the **Add required IAM policies to existing instance profiles attached to your instances** check box.



When you create a patch policy, an Amazon S3 bucket is also created to store the policy's configuration `baseline_overrides.json` file. If you don't select the **Add required IAM policies to existing instance profiles attached to your instances** check box when creating the policy, the IAM policies and resource tags that are needed to access `baseline_overrides.json` in the S3 bucket are not automatically added to your existing IAM instance profiles and service roles.

Solution 1: Delete the existing patch policy configuration, then create a replacement, making sure to select the **Add required IAM policies to existing instance profiles attached to your instances** check box. This selection applies the IAM policies created by this Quick Setup configuration to nodes that already have an instance profile or service role attached. (By default, Quick Setup adds the required policies to instances and nodes that do *not* already have instance profiles or service

roles.) For more information, see Automate organization-wide patching using a Quick Setup patch policy.

Solution 2: Manually add the required permissions and tags to each IAM instance profile and IAM service role that you use with Quick Setup. For instructions, see Permissions for the patch policy S3 bucket.

Issue: Patching fails without an apparent cause or error message

Problem: A patching operation fails without returning an error message.

Possible cause: If more than one invocation of `AWS-RunPatchBaseline` occurs at a time, they can conflict with one another, causing patching tasks to fail. This might not be indicated in patching logs.

To check whether concurrent patching operations might have interrupted each other, review the command history in Run Command, a tool in AWS Systems Manager. For a managed node with a patching failure, check to see if multiple operations attempted to patch the machine within 2 minutes of one another. This scenario can sometimes cause a failure.

You can also use the AWS Command Line Interface (AWS CLI) to check for concurrent patching attempts by using the following command. Replace the value for *`node-id`* with the ID for your managed node.

```
aws ssm list-commands \
    --filter "key=DocumentName,value=AWS-RunPatchBaseline" \
    --query 'Commands[*].
{CommandId:CommandId,RequestedDateTime:RequestedDateTime,Status:Status}' \
    --instance-id node-id \
    --output table
```

Solution: If you determine that patching failed because of competing patching operations on the same managed node, adjust your patching configurations to avoid this occurring again. For example, if two maintenance windows specify overlapping patching times, remove or revise one of them. If a maintenance windows specifies one patching operation, but a patch policy specifies a different one for the same time, consider removing the task from the maintenance window.

If you determine that conflicting patching operations weren't the cause of the failure in this scenario, we recommend contacting AWS Support.

Issue: Unexpected patch compliance results

Problem: When reviewing the patching compliance details generated after a `Scan` operation, the results include information that don't reflect the rules set up in your patch baseline. For example, an exception you added to the **Rejected patches** list in a patch baseline is listed as `Missing`. Or patches classified as `Important` are listed as missing even though your patch baseline specifies `Critical` patches only.

Cause: Patch Manager currently supports multiple methods of running `Scan` operations:

- A patch policy configured in Quick Setup
- A Host Management option configured in Quick Setup
- A maintenance window to run a patch `Scan` or `Install` task
- An on-demand **Patch now** operation

When a `Scan` operation runs, it overwrites the compliance details from the most recent scan. If you have more than one method set up to run a `Scan` operation, and they use different patch baselines with different rules, they will result in differing patch compliance results.

Solution: To avoid unexpected patch compliance results, we recommend using only one method at a time for running the Patch Manager `Scan` operation. For more information, see Avoiding unintentional patch compliance data overwrites.

Errors when running `AWS-RunPatchBaseline` on Linux

Topics

- Issue: 'No such file or directory' error
- Issue: 'another process has acquired yum lock' error
- Issue: 'Permission denied / failed to run commands' error
- Issue: 'Unable to download payload' error
- Issue: 'unsupported package manager and python version combination' error
- Issue: Patch Manager isn't applying rules specified to exclude certain packages
- Issue: Patching fails and Patch Manager reports that the Server Name Indication extension to TLS is not available
- Issue: Patch Manager reports 'No more mirrors to try'

- Issue: Patching fails with 'Error code returned from curl is 23'
- Issue: Patching fails with ‘Error unpacking rpm package…’ message
- Issue: Patching fails with 'Encounter service side error when uploading the inventory'
- Issue: Patching fails with ‘Errors were encountered while downloading packages’ message
- Issue: Patching fails with a message that 'The following signatures couldn't be verified because the public key is not available'
- Issue: Patching fails with a 'NoMoreMirrorsRepoError' message
- Issue: Patching fails with an 'Unable to download payload' message
- Issue: Patching fails with a message 'install errors: dpkg: error: dpkg frontend is locked by another process'
- Issue: Patching on Ubuntu Server fails with a 'dpkg was interrupted' error
- Issue: The package manager utility can't resolve a package dependency

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 tool in 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 tool in 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.amzn-s3-demo-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.

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: A supported version of python3 isn't installed on the Debian Server or Ubuntu Server instance.

Solution: Install a supported version of python3 (3.0 - 3.10) on the server, which is required for Debian Server 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 might 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 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.

Issue: Patching fails with 'Error code returned from curl is 23'

Problem: A patching operating that uses `AWS-RunPatchBaseline` fails with an error similar to the following:

```
05/01/2025 17:04:30 root [ERROR]: Error code returned from curl is 23
```

Cause: The curl tool in use on your systems lacks the permissions needed to write to the filesystem. This can occur when if the package manager's default curl tool was replaced by a different version, such as one installed with snap.

Solution: If the curl version provided by the package manager was uninstalled when a different version was installed, reinstall it.

If you need to keep multiple curl versions installed, ensure that the version associated with the package manager is in the first directory listed in the `PATH` variable. You can check this by running the command `echo $PATH` to see the current order of directories that are checked for executable files on your system.

Issue: Patching fails with 'Error unpacking rpm package...' message

Problem: A patching operation fails with an error similar to the following:

```
Error : Error unpacking rpm package python-urllib3-1.25.9-1.amzn2.0.2.noarch
python-urllib3-1.25.9-1.amzn2.0.1.noarch was supposed to be removed but is not!
failed to run commands: exit status 1
```

Cause 1: When a particular package is present in multiple package installers, such as both `pip` and `yum` or `dnf`, conflicts can occur when using the default package manager.

A common example occurs with the `urllib3` package, which is found in `pip`, `yum`, and `dnf`.

Cause 2: The `python-urllib3` package is corrupted. This can happen if the package files were installed or updated by `pip` after the `rpm` was package was previously installed by `yum` or `dnf`.

Solution: Remove the `python-urllib3` package from pip by running the command `sudo pip uninstall urllib3`, keeping the package only in the default package manager (`yum` or `dnf`).

Issue: Patching fails with 'Encounter service side error when uploading the inventory'

Problem: When running the `AWS-RunPatchBaseline` document, you receive the following error message:

```
Encounter service side error when uploading the inventory
```

Cause: Two commands to run `AWS-RunPatchBaseline` were running at the same time on the same managed node. This creates a race condition when initializing boto3 client during patching operations.

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: Patching fails with ‘Errors were encountered while downloading packages’ message

Problem: During patching, you receive an error similar to the following:

```
YumDownloadError: [u'Errors were encountered while downloading
packages.', u'libxml2-2.9.1-6.el7_9.6.x86_64: [Errno 5] [Errno 12]
Cannot allocate memory', u'libxslt-1.1.28-6.el7.x86_64: [Errno 5]
[Errno 12] Cannot allocate memory', u'libcroco-0.6.12-6.el7_9.x86_64:
[Errno 5] [Errno 12] Cannot allocate memory', u'openldap-2.4.44-25.el7_9.x86_64:
[Errno 5] [Errno 12] Cannot allocate memory',
```

Cause: This error can occur when insufficient memory is available on a managed node.

Solution: Configure the swap memory, or upgrade the instance to a different type to increase the memory support. Then start a new patching operation.

Issue: Patching fails with a message that 'The following signatures couldn't be verified because the public key is not available'

Problem: Patching fails on Ubuntu Server with an error similar to the following:

```
02/17/2022 21:08:43 root [ERROR]: W:GPG error:
http://repo.mysql.com/apt/ubuntu  bionic InRelease: The following
signatures couldn't be verified because the public key is not available:
NO_PUBKEY 467B942D3A79BD29, E:The repository ' http://repo.mysql.com/apt/ubuntu bionic
```

Cause: The GNU Privacy Guard (GPG) key has expired or is missing.

Solution: Refresh the GPG key, or add the key again.

For example, using the error shown previously, we see that the 467B942D3A79BD29 key is missing and must be added. To do so, run either of the following commands:

```
sudo apt-key adv --keyserver hkps://keyserver.ubuntu.com --recv-keys 467B942D3A79BD29
```

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 467B942D3A79BD29
```

Or, to refresh all keys, run the following command:

```
sudo apt-key adv --keyserver hkps://keyserver.ubuntu.com --refresh-keys
```

If the error recurs after this, we recommend reporting the issue to the organization that maintains the repository. Until a fix is available, you can edit the `/etc/apt/sources.list` file to omit the repository during the patching process.

To do so, open the `sources.list` file for editing, locate the line for the repository, and insert a # character at the beginning of the line to comment it out. Then save and close the file.

Issue: Patching fails with a 'NoMoreMirrorsRepoError' message

Problem: You receive an error similar to the following:

```
NoMoreMirrorsRepoError: failure: repodata/repomd.xml from pgdg94: [Errno 256] No more
 mirrors to try.
```

Cause: There is an error in the source repository.

Solution: We recommend reporting the issue to the organization that maintains the repository. Until the error is fixed, you can disable the repository at the operating system level. To do so, run the following command, replacing the value for *`repo-name`* with your repository name:

```
yum-config-manager --disable repo-name
```

Following is an example.

```
yum-config-manager --disable pgdg94
```

After you run this command, run another patching operation.

Issue: Patching fails with an 'Unable to download payload' message

Problem: You receive an error similar to the following:

```
Unable to download payload:
https://s3.dualstack.eu-west-1.amazonaws.com/aws-ssm-eu-west-1/patchbaselineoperations/
linux/payloads/patch-baseline-operations-1.83.tar.gz.
failed  to run commands: exit status 156
```

Cause: The managed node configuration contains errors or is incomplete.

Solution: Make sure that the managed node is configured with the following:

- Outbound TCP 443 rule in security group.
- Egress TCP 443 rule in NACL.
- Ingress TCP 1024-65535 rule in NACL.
- NAT/IGW in route table to provide connectivity to an S3 endpoint. If the instance doesn't have internet access, provide it connectivity with the S3 endpoint. To do that, add an S3 gateway endpoint in the VPC and integrate it with the route table of the managed node.

Issue: Patching fails with a message 'install errors: dpkg: error: dpkg frontend is locked by another process'

Problem: Patching fails with an error similar to the following:

```
install errors: dpkg: error: dpkg frontend is locked by another process
failed to run commands: exit status 2
Failed to install package; install status Failed
```

Cause: The package manager is already running another process on a managed node at the operating system level. If that other process takes a long time to complete, the Patch Manager patching operation can time out and fail.

Solution: After the other process that’s using the package manager completes, run a new patching operation.

Issue: Patching on Ubuntu Server fails with a 'dpkg was interrupted' error

Problem: On Ubuntu Server, patching fails with an error similar to the following:

```
E: dpkg was interrupted, you must manually run
'dpkg --configure -a' to correct the problem.
```

Cause: One or more packages is misconfigured.

Solution: Perform the following steps:

1. Check to see which packages are affected, and what the issues are with each package by running the following commands, one at a time:

   ```
   sudo apt-get check
   ```

   ```
   sudo dpkg -C
   ```

   ```
   dpkg-query -W -f='${db:Status-Abbrev} ${binary:Package}\n' | grep -E ^.[^nci]
   ```

2. Correct the packages with issues by running the following command:

   ```
   sudo dpkg --configure -a
   ```

3. If the previous command didn't fully resolve the issue, run the following command:

   ```
   sudo apt --fix-broken install
   ```

Issue: The package manager utility can't resolve a package dependency

Problem: The native package manager on the managed node is unable to resolve a package dependency and patching fails. The following error message example indicates this type of failure on an operating system that uses `yum` as the package manager.

```
09/22/2020 08:56:09 root [ERROR]: yum update failed with result code: 1,
message: [u'rpm-python-4.11.3-25.amzn2.0.3.x86_64 requires rpm = 4.11.3-25.amzn2.0.3',
u'awscli-1.18.107-1.amzn2.0.1.noarch requires python2-botocore = 1.17.31']
```

Cause: On Linux operating systems, Patch Manager uses the native package manager on the machine to run patching operations. such as `yum`, `dnf`, `apt`, and `zypper`. The applications automatically detect, install, update, or remove dependent packages as required. However, some

conditions can result in the package manager being unable to complete a dependency operation, such as:

- Multiple conflicting repositories are configured on the operating system.
- A remote repository URL is inaccessible due to network-related issues.
- A package for the wrong architecture is found in the repository.

Solution: Patching might fail because of a dependency issue for a wide variety of reasons. Therefore, we recommend that you contact AWS Support to assist with troubleshooting.

Errors when running `AWS-RunPatchBaseline` on Windows Server

Topics

- Issue: mismatched product family/product pairs
- Issue: AWS-RunPatchBaseline output returns an HRESULT (Windows Server)
- Issue: managed node doesn't have access to Windows Update Catalog or WSUS
- Issue: PatchBaselineOperations PowerShell module is not downloadable
- Issue: missing patches

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 following microsoft.com topics to identify troubleshooting steps for resolving the error:

- Windows Update error codes by component
- Windows Update common errors and mitigation

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.aws-api-
domain/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(

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 -Snapsho ...

+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

+ 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(
 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.aws-api-
domain/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. 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 Improve the security of EC2 instances by using VPC endpoints for Systems Manager.

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**.
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, 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.

Using AWS Support Automation runbooks

AWS Support provides two Automation runbooks you can use to troubleshoot certain issues related to patching.

- `AWSSupport-TroubleshootWindowsUpdate` – The `AWSSupport-TroubleshootWindowsUpdate` runbook is used to identify issues that could fail the Windows Server updates for Amazon Elastic Compute Cloud (Amazon EC2) Windows Server instances.
- `AWSSupport-TroubleshootPatchManagerLinux` – The `AWSSupport-TroubleshootPatchManagerLinux` runbook troubleshoots common issues that can cause a patch failure on Linux-based managed nodes using Patch Manager. The main goal of this runbook is to identify the patch command failure root cause and suggest a remediation plan.

> **Note**
>
> There is a charge to run Automation runbooks. For information, see AWS Systems Manager Pricing for Automation.

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 Support plan, you can create a technical support case at AWS Support.

Before you contact Support, collect the following items:

- SSM agent logs
- 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
  - 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 directory `/var/lib/amazon/ssm/`*`instance-id`*`/document/orchestration/`*`Run-Command-execution-id`*`/awsrunShellScript/PatchLinux`

AWS Systems Manager Run Command

Using Run Command, a tool in 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 or non-EC2 machine in your hybrid and multicloud 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.

The Run Command API follows an eventual consistency model, due to the distributed nature of the system supporting the API. This means that the result of an API command you run that affects your resources might not be immediately visible to all subsequent commands you run. You should keep this in mind when you carry out an API command that immediately follows a previous API command.

Getting Started

The following table includes information to help you get started with Run Command.

Topic	Details
Setting up managed nodes for AWS Systems Manager	Verify that you have completed the setup requirements for your Amazon Elastic Compute Cloud (Amazon EC2) instances and non-EC2 machines in a hybrid and multicloud environment.
Managing nodes in hybrid and multicloud environments with Systems Manager	(Optional) Register on-premises servers and VMs with AWS so you can manage them using Run Command.
the section called “Managing edge devices with Systems Manager”	(Optional) Configure edge devices so you can manage them using Run Command.
Running commands on managed nodes	Learn how to run a command that targets one or more managed nodes by using the AWS Management Console.
Run Command walkthroughs	Learn how to run commands using either Tools for Windows PowerShell or the AWS CLI.

EventBridge support

This Systems Manager tool 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 and Reference: Amazon EventBridge event patterns and types for Systems Manager.

More info

- 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

Topics

- Setting up Run Command
- Running commands on managed nodes
- Using exit codes in commands
- Understanding command statuses
- Run Command walkthroughs
- Troubleshooting Systems Manager Run Command

Setting up Run Command

Before you can manage nodes by using Run Command, a tool in AWS Systems Manager, configure an AWS Identity and Access Management (IAM) policy for any user who will run commands. If you use any global condition keys for the `SendCommand` action in your IAM policies, you must include the `aws:ViaAWSService` condition key and set the boolean value to `true`. The following is an example.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:document/YourDocument"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "aws:SourceVpce": [
                        "vpce-1234567890abcdef0"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:document/YourDocument"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        }
    ]
}
```

You must also configure your nodes for Systems Manager. For more information, see Setting up managed nodes for AWS Systems Manager.

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.

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.

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 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.

Restricting Run Command access based on tags

This section describes how to restrict a user's ability to run commands on managed nodes by specifying a tag condition in an IAM policy. Managed nodes include Amazon EC2 instances and non-EC2 nodes in a hybrid and multicloud 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 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`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ssm:SendCommand"
         ],
         "Resource":[
            "arn:aws:ssm:*:*:document/*"
         ]
```

```
      },
      {
         "Effect":"Allow",
         "Action":[
            "ssm:SendCommand"
         ],
         "Resource":[
            "arn:aws:ec2:*:*:instance/*"
         ],
         "Condition":{
            "StringLike":{
               "ssm:resourceTag/Finance":[
                  "WebServers"
               ]
            }
         }
      }
   ]
}
```

You can create IAM policies that 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`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ssm:SendCommand"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "ssm:resourceTag/tag_key1":[
                  "tag_value1"
               ],
```

```
                "ssm:resourceTag/tag_key2":[
                   "tag_value2"
                ]
             }
          }
       },
       {
          "Effect":"Allow",
          "Action":[
             "ssm:SendCommand"
          ],
          "Resource":[
             "arn:aws:ssm:us-west-1::document/AWS-*",
             "arn:aws:ssm:us-east-2::document/AWS-*"
          ]
       },
       {
          "Effect":"Allow",
          "Action":[
             "ssm:UpdateInstanceInformation",
             "ssm:ListCommands",
             "ssm:ListCommandInvocations",
             "ssm:GetDocument"
          ],
          "Resource":"*"
       }
    ]
}
```

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.

JSON

```
{
   "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 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*.

Running commands on managed nodes

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.

Note that if your node is configured with the `noexec` mount option for the var directory, Run Command is unable to successfuly run commands.

> **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 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 Parameter Store parameters using IAM policies.

Execution history retention

The history of each command is available for up to 30 days. In addition, you can store a copy of all log files in Amazon Simple Storage Service or have an audit trail of all API calls in AWS CloudTrail.

Related information

For information about sending commands using other tools, see the following topics:

- Walkthrough: Use the AWS Tools for Windows PowerShell with Run Command or the examples in the AWS Systems Manager section of the AWS Tools for PowerShell Cmdlet Reference.
- Walkthrough: Use the AWS CLI with Run Command or the examples in the SSM CLI Reference

Contents

- Running commands from the console
- Running commands using a specific document version
- Run commands at scale
- Canceling a command

Running commands from the console

You can use Run Command, a tool in 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 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 all Systems Manager setup requirements.

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**.
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.

   > **Tip**
   >
   > If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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) Choose a CloudWatch alarm to apply to your command for monitoring. To attach a CloudWatch alarm to your command, the IAM principal that runs the command must have permission for the `iam:createServiceLinkedRole` action. For more information about CloudWatch alarms, see Using Amazon CloudWatch alarms. Note that if your alarm activates, any pending command invocations do not run.

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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.
12. Choose **Run**.

For information about canceling a command, see the section called "Canceling a command".

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**.
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.

Running commands using a specific document version

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 Installing or updating the latest version of the AWS CLI.

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. Replace *document name* with your own information.

   ```
   aws ssm list-document-versions \
       --name "document name"
   ```

4. Run the following command to run a command that uses an SSM document version. Replace each *example resource placeholder* with your own information.

   ```
   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 Installing or updating the latest version of the AWS CLI.

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. Replace *document name* with your own information.

   ```
   aws ssm list-document-versions ^
       --name "document name"
   ```

4. Run the following command to run a command that uses an SSM document version. Replace each *example resource placeholder* with your own information.

   ```
   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 (Tools for Windows PowerShell), if you haven't already.

   For information, see Installing the AWS Tools for PowerShell.

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. Replace *document name* with your own information.

```
Get-SSMDocumentVersionList `
    -Name "document name"
```

4. Run the following command to run a command that uses an SSM document version. Replace each *example resource placeholder* with your own information.

```
Send-SSMCommand `
    -DocumentName "AWS-RunShellScript" `
    -Parameter @{commands = "echo helloWorld"} `
    -InstanceIds "instance-ID" `
    -DocumentVersion $LATEST
```

Run commands at scale

You can use Run Command, a tool in AWS Systems Manager, to run commands on a fleet of managed nodes by using the `targets`. 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 Tagging your AWS resources in the *Tagging AWS Resources 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
- Using rate controls

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). Replace each *example resource placeholder* with your own information. 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 tool in 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" \
    [...]
```

Windows

```
aws ssm send-command ^
    --document-name document-name ^
    --targets Key=tag:OS,Values="Windows Server 2016" ^
    [...]
```

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" \
    [...]
```

Windows

```
aws ssm send-command ^
    --document-name document-name ^
    --targets Key="tag:Operating System",Values="Windows Server 2016" ^
    [...]
```

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
- Using error controls

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 ^
    [...]
```

Canceling 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**.
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. Replace each *example resource placeholder* with your own information.

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.

Using exit codes in commands

In some cases, you might need to manage how your commands are handled by using exit codes.

Specify exit codes in commands

Using Run Command, a tool in AWS Systems Manager, you can specify exit codes to determine how commands are handled. 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
```

Handling reboots when running commands

If you use Run Command, a tool in 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 Server example installs the Telnet-Client on the node, and then restarts it.

Amazon Linux 2

```
#!/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
    }
```

Understanding command statuses

Run Command, a tool in 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 **Commands** tab in the Run Command console interface.
- 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 Command document plugin reference.

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 Run commands at scale.

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 **Timeout (seconds)** parameter and the **Execution timeout** parameter, the status changes to `Delivery Timed Out`.
InProgress	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 `Success`, `Failed`, `Delivery Timed Out`, or `Execution Timed Out`. *Exception*: If the agent isn't running or available on the node, the command status remains at `In Progress` until the agent is

Status	Details
	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.

Status	Details
Success	This status is returned under a variety of conditions. This status *doesn't* mean the command was processed on the node. For example, the command can be received by SSM Agent on the managed node and return an exit code of zero as a result of your PowerShell `ExecutionPolicy` preventing the command from running. This is a terminal state. Conditions that result in a command returning a `Success` status are: • When targeting a single instance, the command was received by SSM Agent on the managed node and returned an exit code of zero. • When targeting multiple instances, the number of failed invocations has not crossed the error threshold specified in the command. • When targeting multiple instances, at least 1 invocation has succeeded while others have timed out. The specified error threshold still applies. • When targeting a tag, no instances are found associated with the tag. • When targeting a tag, the number of failed invocations has not crossed the error threshold specified in the command. • When targeting a tag, at least 1 invocation has succeeded while others have timed out. The specified error threshold still applies. • You have applications or policies enforced at the OS level that prevent or override the

Status	Details
	execution of a command resulting in an exit code of zero being returned. **Note** The same conditions apply when targeting resource groups. 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).
DeliveryTimedOut	The command wasn't delivered to the managed node before the total timeout expired. Total timeouts don't count against the parent command's `max-errors` limit, but they do contribute to whether the parent command status is `Success`, `Incomplete` , or `Delivery Timed Out`. This is a terminal state.
ExecutionTimedOut	Command automation started on the managed node, but the command wasn't completed before the execution timeout expired. Execution timeouts 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.

Status	Details
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 `max-errors` limit of the parent command. This is a terminal state.
Cancelled	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 `max-errors` 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 `max-errors` limit and subsequent command invocations were canceled by the system. This is a terminal state.

Status	Details
InvalidPlatform	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 max-errors 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.
AccessDenied	The AWS Identity and Access Managemen t (IAM) user or role initiating the command doesn't have access to the targeted managed node. `Access Denied` doesn't count against the parent command's `max-errors` 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 `Access Denied`, then the command status returned is `Failed`. However, if a command has five invocations, four of which return the status `Access Denied` and one of which returns the status `Success`, then the parent command's status is `Success`. 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.
InProgress	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 `max-errors` wasn't reached. This status *doesn't* mean the command was successfully processed on all specified or targeted managed nodes. This is a terminal state. **Note** To troubleshoot errors or get more information about the command execution, send a command that handles errors or exceptions by returning appropriate exit codes (non-zero exit codes for command failure).
DeliveryTimedOut	The command wasn't delivered to the managed node before the total timeout expired. The value of `max-errors` or more command invocations shows a status of

Status	Details
	`Delivery Timed Out`. This is a terminal state.
Failed	The command wasn't successful on the managed node. The value of `max-errors` or more 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.
Cancelled	The command was canceled before it was completed. This is a terminal state.
RateExceeded	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.

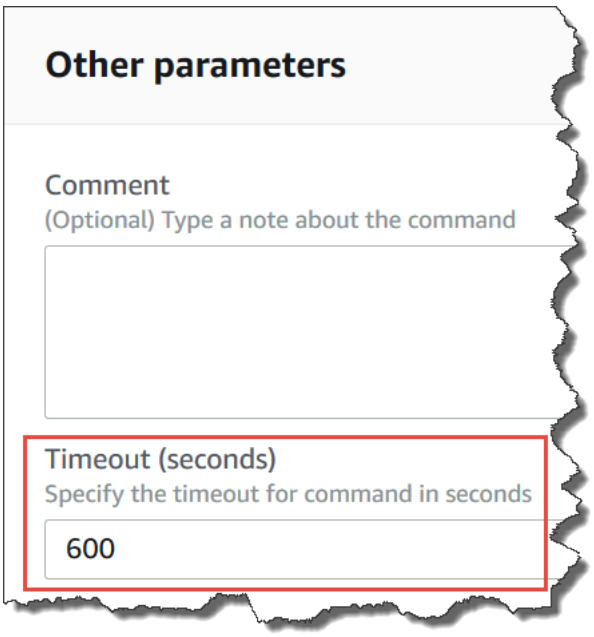
Status	Details
AccessDenied	The user or role initiating the command doesn't have access to the targeted resource group. `AccessDenied` doesn't count against the parent command's `max-errors` limit, but does contribute to whether the parent command status is `Success` or `Failed`. (For example, if all 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:

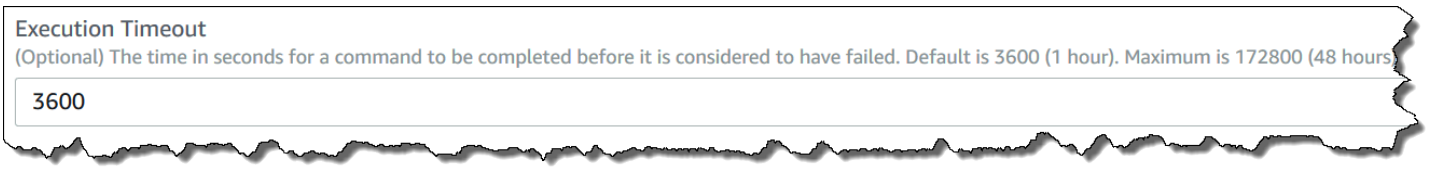
```
"executionTimeout": {
    "type": "String",
    "default": "3600",

"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. The **Execution Timeout** field is only displayed when a corresponding input parameter has been defined in the SSM document. 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 tool in AWS Systems Manager, using either the AWS Command Line Interface (AWS CLI) or AWS Tools for Windows PowerShell.

Contents

- Updating software using Run Command
- Walkthrough: Use the AWS CLI with Run Command
- Walkthrough: Use the AWS Tools for Windows PowerShell with Run Command

You can also view sample commands in the following references.

- Systems Manager AWS CLI Reference
- AWS Tools for Windows PowerShell - AWS Systems Manager

Updating software using Run Command

The following procedures describe how to update software on your managed nodes.

Updating the SSM Agent 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.

> **Note**
>
> If an instance is running macOS version 13.0 (Ventura) or later, the instance must have the SSM Agent version 3.1.941.0 or higher to run the AWS-UpdateSSMAgent document. If the instance is running a version of SSM Agent released before 3.1.941.0, you can update your SSM Agent to run the AWS-UpdateSSMAgent document by running `brew update` and `brew upgrade amazon-ssm-agent` commands.

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**.
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.

   > **Tip**
   >
   > If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

11. Choose **Run**.

Updating 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**.
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.

   > **Tip**
   >
   > If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.
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.

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.

Topics

- Step 1: Getting started
- Step 2: Run shell scripts to view resource details
- Step 3: Send simple commands using the AWS-RunShellScript document
- Step 4: Run a simple Python script using Run Command
- Step 5: Run a Bash script using Run Command

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 Setting up managed nodes for AWS Systems Manager.

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 Installing or updating the latest version of the AWS CLI.

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 an 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 tool in 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

The following command displays the default user 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
\""]}' \
    --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 and Running scripts from GitHub.

```
#!/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 S3 bucket in thee 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.

Topics

- Configure AWS Tools for Windows PowerShell session settings
- List all available documents
- Run PowerShell commands or scripts
- Install an application using the AWS-InstallApplication document
- Install a PowerShell module using the AWS-InstallPowerShellModule JSON document
- Join a managed node to a Domain using the AWS-JoinDirectoryServiceDomain JSON document
- Send Windows metrics to Amazon CloudWatch Logs using the AWS-ConfigureCloudWatch document
- Update EC2Config using the AWS-UpdateEC2Config document
- Turn on or turn off Windows automatic update using the AWS-ConfigureWindowsUpdate document
- Manage Windows updates using Run Command

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 Setting up managed nodes for AWS Systems Manager.

```
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 Handling reboots when running commands.

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.

```
$runPSCommand = 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 $runPSCommand.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 $runPSCommand.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 $runPSCommand.CommandId `
    -Details $true `
```

```
    -InstanceId instance-ID | Select -ExpandProperty CommandPlugins
```

Cancel a command

The following command cancels the `Send-SSMCommand` for the `AWS-RunPowerShellScript` document.

```
$cancelCommand = Send-SSMCommand `
    -InstanceIds @("instance-ID-1","instance-ID-2") `
    -DocumentName "AWS-RunPowerShellScript" `
    -Comment "Demo AWS-RunPowerShellScript with two instances" `
    -Parameter @{'commands'='Start-Sleep -Seconds 120; dir C:\'}

Stop-SSMCommand -CommandId $cancelCommand.CommandId
```

Check the command status

The following command checks the status of the `Cancel` command.

```
Get-SSMCommand `
    -CommandId $cancelCommand.CommandId
```

Install an application using the `AWS-InstallApplication` document

Using Run Command and the `AWS-InstallApplication` document, you can install, repair, or uninstall applications on 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 Handling reboots when running commands.

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 "amzn-s3-demo-bucket".

```
$installPSCommand = Send-SSMCommand `
    -InstanceId instance-ID `
    -DocumentName "AWS-InstallPowerShellModule" `
    -Parameter @{'source'='https://gallery.technet.microsoft.com/EZOut-33ae0fb7/
file/110351/1/EZOut.zip';'commands'=@('Add-WindowsFeature -name XPS-Viewer -restart')}
 `
    -OutputS3BucketName amzn-s3-demo-bucket
```

Get command information per managed node

The following command uses the `CommandId` to get the status of the command execution.

```
Get-SSMCommandInvocation `
    -CommandId $installPSCommand.CommandId `
    -Details $true
```

Get command information with response data for the managed node

The following command returns the output of the original `Send-SSMCommand` for the specific `CommandId`.

```
Get-SSMCommandInvocation `
    -CommandId $installPSCommand.CommandId `
    -Details $true | Select -ExpandProperty CommandPlugins
```

Join 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 Handling reboots when running commands.

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 amzn-s3-demo-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.CloudWa
 "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.PerformanceCounterInputComp
 "Parameters":{"CategoryName":"Memory", "CounterName":"Available
 MBytes", "InstanceName":"", "MetricName":"AvailableMemory",
 "Unit":"Megabytes","DimensionName":"", "DimensionValue":""}},{"Id":"CloudWatch",
```

```
 "FullName":"AWS.EC2.Windows.CloudWatch.CloudWatch.CloudWatchOutputComponent,AWS.EC2.Windows.Cl
 "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 officially supported on Windows Server 2012 and later versions.

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 Handling reboots when running commands.

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 tool in AWS Systems Manager, provides status details with each command execution. For more information about the details of command statuses, see Understanding command statuses. You can also use the information in this topic to help troubleshoot problems with Run Command.

Topics

- Some of my managed nodes are missing
- A step in my script failed, but the overall status is 'succeeded'

- SSM Agent isn't running properly

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 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 Specify exit codes in commands.

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.

AWS Systems Manager Session Manager

Session Manager is a fully managed AWS Systems Manager tool. With Session Manager, you can manage your Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, 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 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 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.

- **Connect to both Amazon EC2 instances and non-EC2 managed nodes in hybrid and multicloud environments**

  You can connect to both Amazon Elastic Compute Cloud (Amazon EC2) instances and non-EC2 nodes in your hybrid and multicloud environment.

  To connect to non-EC2 nodes using Session Manager, you must first activate the advanced-instances tier. **There is a charge to use the advanced-instances tier.** However, there is no additional charge to connect to EC2 instances using Session Manager. For information, see Configuring instance tiers.

- **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 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 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.
  - **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).
  - **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).
  - **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).

> **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 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, on-premises servers, and virtual machines (VMs). For a list of supported operating system types, see Setting up Session Manager.

  > **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.

- **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 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 Install the Session Manager plugin for the AWS CLI. To view the plugin on GitHub, see session-manager-plugin.

- **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 support**

  Session Manager provide you with options for logging session histories in your AWS account through integration with a number of other AWS services. For more information, see Logging session activity and Enabling and disabling session logging.

- **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 Enabling and disabling session logging and Configure session preferences.

- **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 Improve the security of EC2 instances by using VPC endpoints for Systems Manager.

- **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
- Working with Session Manager
- Logging session activity
- Enabling and disabling session logging
- Session document schema
- Troubleshooting Session Manager

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
- Step 2: Verify or add instance permissions for Session Manager
- Step 3: Control session access to managed nodes
- Step 4: Configure session preferences
- Step 5: (Optional) Restrict access to commands in a session
- Step 6: (Optional) Use AWS PrivateLink to set up a VPC endpoint for Session Manager
- Step 7: (Optional) Turn on or turn off ssm-user account administrative permissions
- Step 8: (Optional) Allow and control permissions for SSH connections through Session Manager

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 non-EC2 machines in your hybrid and multicloud environment that use the *advanced-instances* tier. Session Manager supports the following operating system versions: **Note** Session Manager supports EC2 instances, edge devices, and on-premises servers and virtual machines (VMs) in your hybrid and multicloud environment that use the *advanced-instances* tier. For more informati on about advanced instances, see Configuring instance tiers. **Linux and macOS** Session Manager supports all the versions of Linux and macOS that are supported by AWS Systems Manager. For information, see Supported operating systems and machine types. **Windows** Session Manager supports Windows Server 2012 and later versions.

Requirement	Description
	Note Microsoft Windows Server 2016 Nano isn't supported.

Requirement	Description
SSM Agent	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. To use the option to encrypt session data using a key created in AWS Key Managemen t Service (AWS KMS), version 2.3.539.0 or later of SSM Agent must be installed on the managed node. To use shell profiles in a session, SSM Agent version 3.0.161.0 or later must be installed on the managed node. 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. To stream session data using Amazon CloudWatch Logs, SSM Agent version 3.0.284.0 or later must be installed on the managed node. For information about how to determine the version number running on an instance, see Checking the SSM Agent version number. For information about manually installing or automatically updating SSM Agent, see Working with SSM Agent. **About the ssm-user account** 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 `ssm-user`. (On versions

Requirement	Description
	before 2.3.612.0, the account is created when SSM Agent starts or restarts. On version 2.3.612.0 and later, `ssm-user` is created the first time a session starts on the managed node.) Sessions are launched using the administrative credentials of this user account. For information about restricting administr ative control for this account, see Turn off or turn on ssm-user account administrative permissions. **ssm-user on Windows Server domain controllers** Beginning with SSM Agent version 2.3.612.0, the `ssm-user` account isn't created automatic ally 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 controlle r, you must create the `ssm-user` 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 for the `ssm-user` account each time a session starts, so you don't need to specify a password when you create the account.

Requirement	Description
Connectivity to endpoints	The managed nodes you connect to must also allow HTTPS (port 443) outbound traffic to the following endpoints: • ec2messages.*region*.amazonaws.com • ssm.*region*.amazonaws.com • ssmmessages.*region*.amazonaws.com For more information, see the following topics: • Reference: ec2messages, ssmmessages, and other API operations • How do I create VPC endpoints so that I can use Systems Manager to manage private EC2 instances without internet access? in the AWS re:Post Knowledge Center. Alternatively, you can connect to the required endpoints by using interface endpoints. For more information, see Step 6: (Optional) Use AWS PrivateLink to set up a VPC endpoint for Session Manager.

Requirement	Description
AWS CLI	(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. You can call `aws --version` to check the version. If you need to install or upgrade the CLI, see Installing the AWS Command Line Interface in the AWS Command Line Interface User Guide. **Important** An updated version of SSM Agent is released whenever new tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools and features. For that reason, we recommend that you automate the process of keeping SSM Agent up to date on your machines. For informati on, see Automating updates to SSM Agent. Subscribe to the SSM Agent Release Notes page on GitHub to get notifications about SSM Agent updates.

Requirement	Description
	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 Install the Session Manager plugin for the AWS CLI.
Turn on advanced-instances tier (hybrid and multicloud environments)	To connect to non-EC2 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 non-EC2 machines as managed nodes. There is a charge to use the advanced-instances tier. For more information about the advanced-instance tier, see Configuring instance tiers.

Requirement	Description
Verify IAM service role permissions (hybrid and multicloud environments)	Hybrid-activated nodes 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 hybrid and multicloud machines using Session Manager. If your service role contains the AWS managed policy `AmazonSSMManagedInstanceCore`, the required permissions for Session Manager are already provided. 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 Deregistering managed nodes in a hybrid and multicloud environment. For more information about creating IAM policies with Session Manager permissions, see Step 2: Verify or add instance permissions for Session Manager.

Step 2: Verify or add instance permissions for Session Manager

By default, AWS Systems Manager doesn't have permission to perform actions on your instances. You can provide instance permissions at the account level using an AWS Identity and Access Management (IAM) role, or at the instance level using an instance profile. If your use case allows, we recommend granting access at the account level using the Default Host Management Configuration. If you've already set up the Default Host Management Configuration for your account using the `AmazonSSMManagedEC2InstanceDefaultPolicy` policy, you can proceed to the next step. For more information about the Default Host Management Configuration, see Managing EC2 instances automatically with Default Host Management Configuration.

Alternatively, you can use instance profiles to provide the required permissions to your instances. 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 tools, 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.

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 Add Session Manager permissions to an existing IAM role.

- **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.

- **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 Configure instance permissions required for Systems Manager.

Topics

- Add Session Manager permissions to an existing IAM role
- Create a custom IAM role for Session Manager

Add Session Manager permissions to an existing IAM role

Use the following procedure to add Session Manager permissions to an existing AWS Identity and Access Management (IAM) role. By adding permissions to an existing role, you can enhance the security of your computing environment without having to use the AWS `AmazonSSMManagedInstanceCore` policy for instance permissions.

> **Note**
>
> Note the following information:
>
> - 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.
> - The following policy example includes an `s3:GetEncryptionConfiguration` action. This action is required if you chose the **Enforce S3 log encryption** option in Session Manager logging preferences.

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. Select the name of the role that you are adding the permissions to.
4. Choose the **Permissions** tab.
5. Choose **Add permissions**, and then select **Create inline policy**.
6. Choose the **JSON** tab.
7. Replace the default policy content with the following content. Replace *`key-name`* with the Amazon Resource Name (ARN) of the AWS Key Management Service key (AWS KMS key) that you want to use.

JSON

```
{
    "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": "arn:aws:kms:us-east-1:111122223333:key/key-name"
        }
    ]
}
```

For information about using a KMS key to encrypt session data, see Turn on KMS key encryption of session data (console).

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.

Create a custom IAM role for Session Manager

You can create an AWS Identity and Access Management (IAM) role that grants Session Manager the permission to perform actions on your Amazon EC2 managed instances. You can also include a policy to grant 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, for information about how to attach the role to an instance, see Attach or Replace an Instance Profile at the AWS re:Post website. For more information about IAM instance profiles and roles, see Using instance profiles in the *IAM User Guide* and IAM roles for Amazon EC2 in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*. For more information about creating an IAM service role for on-premises machines, see Create the IAM service role required for Systems Manager in hybrid and multicloud environments.

Topics

- Creating an IAM role with minimal Session Manager permissions (console)
- Creating an IAM role with permissions for Session Manager and Amazon S3 and CloudWatch Logs (console)

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. To encrypt session data using AWS Key Management Service (AWS KMS), replace *key-name* with the Amazon Resource Name (ARN) of the AWS KMS key that you want to use.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:UpdateInstanceInformation",
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-name"
        }
    ]
```

```
}
```

For information about using a KMS key to encrypt session data, see Turn on KMS key encryption of session data (console).

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.

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 `s3:PutObjectAcl` permission to the IAM role policy. Additionally, you must ensure that the bucket policy grants cross-account access to the IAM role used by the owning account to grant Systems Manager permissions for managed instances. If the bucket uses Key Management Service (KMS) encryption, then the bucket's KMS policy must also grant this cross-account access. For more information about configuring cross-account bucket permissions in Amazon S3, see Granting cross-account bucket permissions in the *Amazon Simple Storage Service User Guide*. If the cross-account permissions aren'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 Enabling and disabling session logging.

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.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel",
                "ssm:UpdateInstanceInformation"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/s3-prefix/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetEncryptionConfiguration"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/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 session access to managed nodes

You grant or revoke Session Manager access to managed nodes by using AWS Identity and Access Management (IAM) policies. You can create a policy and attach it to an IAM user or group that specifies which managed nodes the user or group can connect to. You can also specify the Session Manager API operations the user or groups can perform on those managed nodes.

To help you get started with IAM permission policies for Session Manager, we've created sample policies for an end user and an administrator user. You can use these policies with only minor changes. Or, use them as a guide to create custom IAM policies. For more information, see Sample IAM policies for Session Manager. For information about how to create IAM policies and attach them to users or groups, see Creating IAM Policies and Adding and Removing IAM Policies in the *IAM User Guide*.

About session ID ARN formats

When you create an IAM policy for Session Manager access, you specify a session ID as part of the Amazon Resource Name (ARN). The session ID includes the user name as a variable. To help illustrate this, here's the format of a Session Manager ARN and an example:

```
arn:aws:ssm:region-id:account-id:session/session-id
```

For example:

```
arn:aws:ssm:us-east-2:123456789012:session/JohnDoe-1a2b3c4d5eEXAMPLE
```

For more information about using variables in IAM policies, see IAM Policy Elements: Variables.

Topics

- Start a default shell session by specifying the default session document in IAM policies
- Start a session with a document by specifying the session documents in IAM policies
- Sample IAM policies for Session Manager
- Additional sample IAM policies for Session Manager

Start a default shell session by specifying the default session document in IAM policies

When you configure Session Manager for your AWS account or when you change session preferences in the Systems Manager console, the system creates an SSM session document called

`SSM-SessionManagerRunShell`. This is the default session document. Session Manager uses this document to store your session preferences, which include information like the following:

- A location where you want to save session data, such an Amazon Simple Storage Service (Amazon S3) bucket or a Amazon CloudWatch Logs log group.
- An AWS Key Management Service (AWS KMS) key ID for encrypting session data.
- Whether Run As support is allowed for your sessions.

Here is an example of the information contained in the `SSM-SessionManagerRunShell` session preferences document.

```
{
  "schemaVersion": "1.0",
  "description": "Document to hold regional settings for Session Manager",
  "sessionType": "Standard_Stream",
  "inputs": {
    "s3BucketName": "amzn-s3-demo-bucket",
    "s3KeyPrefix": "MyS3Prefix",
    "s3EncryptionEnabled": true,
    "cloudWatchLogGroupName": "MyCWLogGroup",
    "cloudWatchEncryptionEnabled": false,
    "kmsKeyId": "1a2b3c4d",
    "runAsEnabled": true,
    "runAsDefaultUser": "RunAsUser"
  }
}
```

By default, Session Manager uses the default session document when a user starts a session from the AWS Management Console. This applies to either Fleet Manager or Session Manager in the Systems Manager console, or EC2 Connect in the Amazon EC2 console. Session Manager also uses the default session document when a user starts a session by using an AWS CLI command like the following example:

```
aws ssm start-session \
    --target i-02573cafcfEXAMPLE
```

To start a default shell session, you must specify the default session document in the IAM policy, as shown in the following example.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnableSSMSession",
      "Effect": "Allow",
      "Action": [
        "ssm:StartSession"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:111122223333:instance/instance-id",
        "arn:aws:ssm:us-east-1:111122223333:document/SSM-SessionManagerRunShell"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssmmessages:OpenDataChannel"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Start a session with a document by specifying the session documents in IAM policies

If you use the start-session AWS CLI command using the default session document, you can omit the document name. The system automatically calls the `SSM-SessionManagerRunShell` session document.

In all other cases, you must specify a value for the `document-name` parameter. When a user specifies the name of a session document in a command, the systems checks their IAM policy to verify they have permission to access the document. If they don't have permission, the connection request fails. The following examples includes the `document-name` parameter with the `AWS-StartPortForwardingSession` session document.

```
aws ssm start-session \
    --target i-02573cafcfEXAMPLE \
    --document-name AWS-StartPortForwardingSession \
    --parameters '{"portNumber":["80"], "localPortNumber":["56789"]}'
```

For an example of how to specify a Session Manager session document in an IAM policy, see Quickstart end user policies for Session Manager.

> **Note**
>
> To start a session using SSH, you must complete configuration steps on the target managed node *and* the user's local machine. For information, see (Optional) Allow and control permissions for SSH connections through Session Manager.

Sample 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 entities (users or 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
- Quickstart administrator policy for Session Manager

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 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 Fleet Manager

Use this sample policy to give users the ability to start and resume sessions from only the Session Manager and Fleet Manager consoles.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ec2:us-
east-1:111122223333:instance/i-02573cafcfEXAMPLE",
                "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell"
            ]
        },
        {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
         "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
       },
        {
            "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:userid}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession",
                "ssm:SendCommand"
```

```
            ],
            "Resource": [
                "arn:aws:ec2:us-
east-1:111122223333:instance/i-02573cafcfEXAMPLE",
                "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell"
            ]
        },
        {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
         "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
       },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetConnectionStatus",
                "ssm:DescribeInstanceInformation"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession",
                "ssm:ResumeSession"
            ],
            "Resource": [
                "arn:aws:ssm:*:*:session/${aws:username}-*"
            ]
        }
    ]
}
```

AWS CLI

Use this sample policy to give users the ability to start and resume sessions from the AWS CLI.

JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession",
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ec2:us-
east-1:111122223333:instance/i-02573cafcfEXAMPLE",
                "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell"
            ]
        },
        {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
         "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
       },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession",
                "ssm:ResumeSession"
            ],
            "Resource": [
                "arn:aws:ssm:*:*:session/${aws:userid}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-name"
        }
    ]
}
```

Note

`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) Allow and control permissions for SSH connections through Session Manager.
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).
The permission for SendCommand is needed for cases where a user attempts to start a session from the Amazon EC2 console, but the SSM Agent must be updated to the minimum required version for Session Manager first. Run Command is used to send a command to the instance to update the agent.

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 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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ec2:*:111122223333:instance/*"
            ],
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/Finance": [
                        "WebServers"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:OpenDataChannel"
```

```
            ],
            "Resource": [
                "arn:aws:ssm:*:*:session/${aws:userid}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeSessions",
                "ssm:GetConnectionStatus",
                "ssm:DescribeInstanceProperties",
                "ec2:DescribeInstances"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:CreateDocument",
                "ssm:UpdateDocument",
                "ssm:GetDocument",
                "ssm:StartSession"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": [
                "arn:aws:ssm:*:*:session/${aws:userid}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession",
                "ssm:ResumeSession"
            ],
            "Resource": [
                "arn:aws:ssm:*:*:session/${aws:userid}-*"
            ]
```

```
        }
    ]
}
```

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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession",
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ec2:us-east-1:111122223333:instance/*"
            ],
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/tag-key": [
                        "tag-value"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell"
            ]
```

```
        },
        {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
         "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
       },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetConnectionStatus",
                "ssm:DescribeInstanceInformation"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession",
                "ssm:ResumeSession"
            ],
            "Resource": [
                "arn:aws:ssm:*:*:session/${aws:userid}-*"
            ]
        }
    ]
}
```

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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession",
                "ssm:SendCommand"
```

```
            ],
            "Resource": [
                "arn:aws:ec2:us-east-1:111122223333:instance/*"
            ],
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/tag-key": [
                        "tag-value"
                    ]
                }
            }
        },
        {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
         "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
       },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeSessions",
                "ssm:GetConnectionStatus",
                "ssm:DescribeInstanceInformation",
                "ssm:DescribeInstanceProperties",
                "ec2:DescribeInstances"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:CreateDocument",
                "ssm:UpdateDocument",
                "ssm:GetDocument",
                "ssm:StartSession"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession",
                "ssm:ResumeSession"
```

```
            ],
            "Resource": [
                "arn:aws:ssm:*:*:session/${aws:userid}-*"
            ]
        }
    ]
}
```

> **Note**
>
> 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: Grant access to documents in the console
- Example 2: Restrict access to specific managed nodes
- Example 3: Restrict access based on tags
- Example 4: Allow a user to end only sessions they started
- Example 5: Allow full (administrative) access to all sessions

Example 1: Grant access to documents in the console

You can allow users to specify a custom document when they launch a session using the Session Manager console. The following example IAM policy grants permission to access documents with names that begin with **`SessionDocument-`** in the specified AWS Region and AWS account.

To use this policy, replace each *`example resource placeholder`* with your own information.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetDocument",
                "ssm:ListDocuments"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:document/SessionDocument-*"
            ]
        }
    ]
}
```

Note

The Session Manager console only supports Session documents that have a `sessionType` of `Standard_Stream` which are used to define session preferences. For more information, see Session document schema.

Example 2: Restrict access to specific managed nodes

You can create an IAM policy that defines which managed nodes that a user is allowed to connect to using Session Manager. For example, the following policy grants a user the permission to start, end, and resume their sessions on three specific nodes. The policy restricts the user from connecting to nodes other than those specified.

Note

For federated users, see Example 4: Allow a user to end only sessions they started.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ec2:us-east-1:111122223333:instance/
i-1234567890EXAMPLE",
                "arn:aws:ec2:us-east-1:111122223333:instance/i-
abcdefghijEXAMPLE",
                "arn:aws:ec2:us-east-1:111122223333:instance/
i-0e9d8c7b6aEXAMPLE",
                "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell"
            ]
        },
        {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
         "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
       },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession",
                "ssm:ResumeSession"
            ],
            "Resource": [
                "arn:aws:ssm:*:*:session/${aws:userid}-*"
            ]
        },
        {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
         "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
       }
    ]
```

```
}
```

Example 3: 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`.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ec2:us-east-1:111122223333:instance/*"
            ],
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/Finance": [
                        "WebServers"
                    ]
                }
            }
        },
        {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
         "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
       },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ssm:TerminateSession",
                "ssm:ResumeSession"
            ],
            "Resource": [
                "arn:aws:ssm:*:*:session/${aws:userid}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell"
            ]
        }
    ]
}
```

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`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ssm:StartSession"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "ssm:resourceTag/tag-key1":[
                  "tag-value1"
               ],
```

```
                "ssm:resourceTag/tag-key2":[
                   "tag-value2"
                ]
             }
          }
       },
       {
          "Effect": "Allow",
          "Action": ["ssmmessages:OpenDataChannel"],
          "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
        },
       {
             "Effect": "Allow",
             "Action": [
                 "ssm:StartSession"
             ],
             "Resource": [
                 "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell"
             ]
       }
    ]
}
```

For more information about creating IAM policies, see Managed Policies and Inline Policies in the *IAM User Guide*. For more information about tagging managed nodes, see Tagging your Amazon EC2 resources in the *Amazon EC2 User Guide* (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

Example 4: Allow a user to end only sessions they started

Session Manager provides two methods to control which sessions a federated user in your AWS account is allowed to end.

- Use the variable `{aws:userid}` in an AWS Identity and Access Management (IAM) permissions policy. Federated users can end only sessions they started. For unfederated users, use Method 1. For federated users, use Method 2.
- 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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ssm:DescribeSessions"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": [
                "ssm:TerminateSession"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:ssm:*:*:session/${aws:username}-*"
            ]
        }
    ]
}
```

Method 2: Grant TerminateSession privileges using tags supplied by AWS

You can control which sessions that a user can end by including conditional tag key variables in an IAM 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-02573cafcfEXAMPLE.

JSON

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Action": [
                "ssm:TerminateSession"
             ],
             "Resource": "*",
             "Condition": {
                 "StringLike": {
                     "ssm:resourceTag/aws:ssmmessages:target-id": [
                        "i-02573cafcfEXAMPLE"
                     ]
                 }
             }
         }
```

```
    ]
}
```

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*.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "ssm:TerminateSession"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/aws:ssmmessages:session-id": [
                       "AIDIODR4TAW7CSEXAMPLE"
                    ]
                }
            }
        }
    ]
}
```

The following example demonstrates a policy for cases where the caller type is `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, `AIDIODR4TAW7CSEXAMPLE:MyRole`.

> **Important**
>
> In order for system tags to be applied, the role ID you supply can contain the following characters only: Unicode letters, 0-9, space, _, ., :, /, =, +, -, @, and \.

To retrieve the role ID for a role in your AWS account, use the `get-caller-identity` command. For information, see get-caller-identity in the AWS CLI Command Reference.

JSON

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Action": [
                "ssm:TerminateSession"
             ],
             "Resource": "*",
             "Condition": {
                 "StringLike": {
                     "ssm:resourceTag/aws:ssmmessages:session-id": [
                        "${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.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ssm:TerminateSession"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "ssm:resourceTag/aws:ssmmessages:target-id":[
                  "i-02573cafcfEXAMPLE"
               ],
               "ssm:resourceTag/aws:ssmmessages:session-id":[
                  "${aws:userid}*"
               ]
            }
         }
      }
   ]
}
```

Example 5: 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.

JSON

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

```
    "Statement": [
        {
            "Action": [
                "ssm:StartSession",
                "ssm:TerminateSession",
                "ssm:ResumeSession",
                "ssm:DescribeSessions",
                "ssm:GetConnectionStatus"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
         "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
       }
    ]
}
```

Step 4: Configure session preferences

Users that have been granted administrative permissions in their AWS Identity and Access Management (IAM) policy can configure session preferences, including 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 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.

For more information about the permissions needed to configue Session Manager preferences, see the section called “Grant or deny a user permissions to update Session Manager preferences”.

Topics

- Grant or deny a user permissions to update Session Manager preferences
- Specify an idle session timeout value
- Specify maximum session duration
- Allow configurable shell profiles
- Turn on Run As support for Linux and macOS managed nodes
- Turn on KMS key encryption of session data (console)
- Create a Session Manager preferences document (command line)
- Update Session Manager preferences (command line)

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)
- Streaming session data using Amazon CloudWatch Logs (console)
- Logging session data using Amazon CloudWatch Logs (console)

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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ssm:CreateDocument",
                "ssm:GetDocument",
                "ssm:UpdateDocument",
                "ssm:DeleteDocument"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333: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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ssm:CreateDocument",
                "ssm:GetDocument",
                "ssm:UpdateDocument",
                "ssm:DeleteDocument"
            ],
            "Effect": "Deny",
            "Resource": [
```

```
                "arn:aws:ssm:us-east-1:111122223333:document/SSM-
SessionManagerRunShell"
            ]
        }
    ]
}
```

Specify an idle session timeout value

Session Manager, a tool in 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 tool in AWS Systems Manager, allows you to specify the maximum duration of a session before it ends. By default, sessions do not have a maximum duration. 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 a Session Manager preferences document (command line) and Quickstart end user policies for Session Manager.
> - 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.

```
exec /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, Session Manager authenticates connections using the credentials of the system-generated `ssm-user` account that is created on a managed node. (On Linux and macOS machines, this account is added to `/etc/sudoers/`.) If you choose, you can instead authenticate sessions using the credentials of an operating system (OS) user account, or a domain user for instances joined to an Active Directory. In this case, Session Manager verifies that the OS account that you specified exists on the node, or in the domain, before starting the session. If you attempt to start a session using an OS account that doesn't exist on the node, or in the domain, the connection fails.

> **Note**
>
> Session Manager does not support using an operating system's `root` user account to authenticate connections. For sessions that are authenticated using an OS user account, the node's OS-level and directory policies, like login restrictions or system resource usage restrictions, might not apply.

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 entity (user or role) been tagged with `SSMSessionRunAs = os user account name`?

   If Yes, does the OS 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 entity has *not* been tagged with `SSMSessionRunAs = os user account name`, continue to step 2.
2. If the IAM entity 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 OS user name exist on the managed node? If it does, start the session. If it doesn't, don't allow a session to start.

> **Note**
>
> When you activate Run As support, it prevents Session Manager from starting sessions using the `ssm-user` account on a managed node. This means that if Session Manager fails to connect using the specified OS user account, it doesn't fall back to connecting using the default method.
> If you activate Run As without specifying an OS account or tagging an IAM entity, and you have not specified an OS account in Session Manager preferences, session connection attempts will fail.

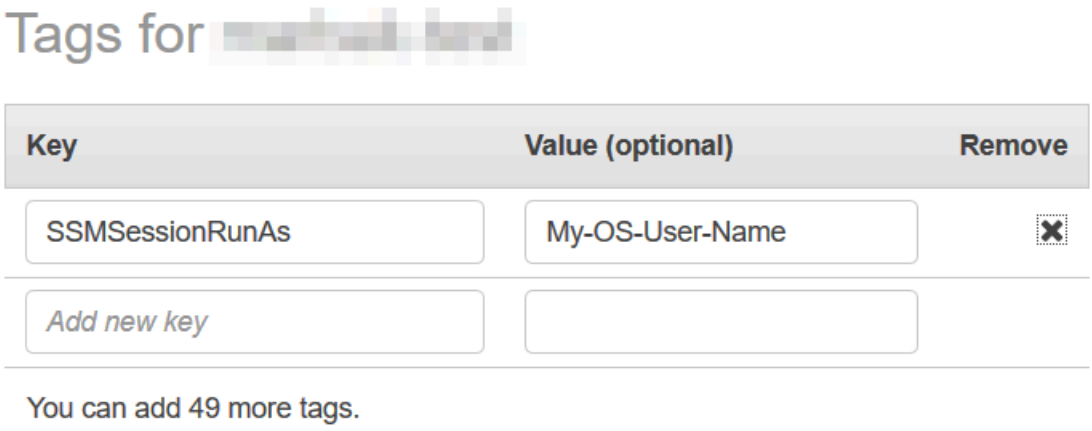
To turn on Run As support for Linux and macOS managed nodes

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 that you want to use to start sessions. Using this option, all sessions are run by the same OS user for all users in your AWS account who connect using Session Manager.
   - **Option 2** (Recommended): Choose the **Open 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 an OS user account for the key value. Choose **Save changes**.

     Using this option, you can specify unique OS users for different IAM entities if you choose. For more information about tagging IAM entities (users or roles), see Tagging IAM resources in the *IAM User Guide*.

     The following is an example.



6. Choose **Save**.

Turn on KMS key encryption of session data (console)

Use AWS Key Management Service (AWS KMS) to create and manage encryption 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 1.2/1.3 encryption that AWS already provides by default.) To encrypt Session Manager session data, create a *symmetric* KMS key using 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.

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: Sample IAM policies for Session Manager.
- Add AWS KMS permissions for managed nodes in your account: Step 2: Verify or add instance permissions for Session Manager.

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 a Session Manager preferences document (command line) or Update Session Manager preferences (command line).

> **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 a Session Manager preferences document (command line)

Use the following procedure to create SSM documents that define your preferences for AWS Systems Manager Session Manager sessions. You can use the document to configure session options including data encryption, session duration, and logging. For example, you can specify whether to store session log data in an Amazon Simple Storage Service (Amazon S3) bucket or Amazon CloudWatch Logs log group. You can create documents that define general preferences for all sessions for an AWS account and AWS Region, or that define preferences for individual sessions.

> **Note**
>
> You can also configure general session preferences by using the Session Manager console.

Documents used to set Session Manager preferences must have a `sessionType` of `Standard_Stream`. For more information about Session documents, see the section called "Session document schema".

For information about using the command line to update existing Session Manager preferences, see Update Session Manager preferences (command line).

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**
>
> This procedure describes how to create documents for setting Session Manager preferences at the AWS account level. To create documents that will be used for setting session-level preferences, specify a value other than `SSM-SessionManagerRunShell` for the file name related command inputs .
>
> To use your document to set preferences for sessions started from the AWS Command Line Interface (AWS CLI), provide the document name as the `--document-name` parameter value. To set preferences for sessions started from the Session Manager console, you can type or select the name of your document from a list.

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":""
   }
}
```

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 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": "amzn-s3-demo-bucket",
    "s3KeyPrefix": "MyS3Prefix",
    "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: Sample IAM policies for Session Manager
- Add AWS KMS permissions for managed nodes in your account: Step 2: Verify or add instance permissions for Session Manager

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": "amzn-s3-demo-bucket",
    "s3KeyPrefix": "MyS3Prefix",
    "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: Sample IAM policies for Session Manager.
- Add AWS KMS permissions for managed nodes in your account: Step 2: Verify or add instance permissions for Session Manager.

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",
        "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 that a user can run in an AWS Systems Manager Session Manager session by using a custom `Session` type AWS Systems Manager (SSM) document. In the document, you define the command that is run when the user starts a session and the parameters that the user can provide to the command. 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 that you define. For more information about using IAM policies to restrict access to commands in a session, see IAM policy examples for interactive commands.

Documents with the `sessionType` of `InteractiveCommands` are only supported for sessions started from the AWS Command Line Interface (AWS CLI). The user provides the custom document name as the `--document-name` parameter value and provides any command parameter values using the `--parameters` option. For more information about running interactive commands, see Starting a session (interactive and noninteractive commands).

Use following procedure 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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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
        }
    }
}
```

2. Run the following commands to create an SSM document using your content that defines the command a user can run in a Session Manager session.

   Linux & 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

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"ssm:StartSession",
         "Resource":[
            "arn:aws:ec2:us-east-1:444455556666:instance/i-02573cafcfEXAMPLE",
            "arn:aws:ssm:us-east-1:444455556666:document/allowed-session-
document"
         ]
      },
      {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
         "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
      }
   ]
}
```

Allow a user to run an interactive command on all managed nodes

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"ssm:StartSession",
         "Resource":[
            "arn:aws:ec2:us-east-1:444455556666:instance/*",
            "arn:aws:ssm:us-east-1:444455556666:document/allowed-session-
document"
         ]
      },
      {
         "Effect": "Allow",
         "Action": ["ssmmessages:OpenDataChannel"],
```

```
            "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
        }
    ]
}
```

Allow a user to run multiple interactive commands on all managed nodes

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":"ssm:StartSession",
            "Resource":[
                "arn:aws:ec2:us-east-1:444455556666:instance/*",
                "arn:aws:ssm:us-east-1:444455556666:document/allowed-session-
document",
                "arn:aws:ssm:us-east-1:444455556666:document/allowed-session-
document-2"
            ]
        },
        {
            "Effect": "Allow",
            "Action": ["ssmmessages:OpenDataChannel"],
            "Resource": ["arn:aws:ssm:*:*:session/${aws:userid}-*"]
        }
    ]
}
```

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 Improve the security of EC2 instances by using VPC endpoints for Systems Manager.

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.

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
- Managing ssm-user Administrator account permissions on Windows Server

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 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 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 control 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
- Controlling user permissions for SSH connections through Session Manager

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:

     - Manually installing and uninstalling SSM Agent on EC2 instances for Windows Server.
     - Manually installing and uninstalling SSM Agent on EC2 instances for Linux
     - Manually installing and uninstalling SSM Agent on EC2 instances for macOS
     - How to install the SSM Agent on hybrid Windows nodes
     - How to install the SSM Agent on hybrid Linux nodes

     > **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 Configuring instance tiers.

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 Install the Session Manager plugin for the AWS CLI.

   - 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'"
    User ec2-user
```

     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).)

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. Replace each *example resource placeholder* with your own information.

    JSON

    ```
    {
    ```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:StartSession",
            "Resource": [
                "arn:aws:ec2:us-east-1:111122223333:instance/instance-id",
                "arn:aws:ssm:*:*:document/AWS-StartSSHSession"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ssmmessages:OpenDataChannel",
            "Resource": "arn:aws:ssm:*:*:session/${aws:userid}-*"
        }
    ]
}
```

- **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.

    JSON

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

```
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": "arn:aws:ssm:*:*:document/AWS-StartSSHSession"
        },
        {
            "Effect": "Allow",
            "Action": "ssmmessages:OpenDataChannel",
            "Resource": "arn:aws:ssm:*:*:session/${aws:userid}-*"
        }
    ]
}
```

- **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 and Specify maximum session duration.

For more information about sessions, see What is a session?

Topics

- Install the Session Manager plugin for the AWS CLI
- Start a session
- End a session
- View session history

Install the Session Manager plugin for the AWS CLI

To initiate Session Manager sessions with your managed nodes by using the AWS Command Line Interface (AWS CLI), you must install the *Session Manager plugin* on your local machine. You can install the plugin on supported versions of Microsoft Windows Server, macOS, Linux, and Ubuntu Server.

> **Note**
>
> To use the Session Manager plugin, you must have AWS CLI version 1.16.12 or later installed on your local machine. For more information, see Installing or updating the latest version of the AWS Command Line Interface.

Topics

- Session Manager plugin latest version and release history
- Install the Session Manager plugin on Windows
- Install the Session Manager plugin on macOS
- Install the Session Manager plugin on Linux
- Verify the Session Manager plugin installation
- Session Manager plugin on GitHub
- (Optional) Turn on Session Manager plugin logging

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.

> **Important**
>
> We recommend you always run the latest version. The latest version includes enhancements that improve the experience of using the plugin.

Version	Release date	Details
1.2.707.0	February 6, 2025	**Enhancement**: Upgraded the Go version to 1.23 in the Dockerfile. Updated the version configuration step in the README.
1.2.694.0	November 20, 2024	**Bug fix**: Rolled back change that added credentials to OpenDataChannel requests.
1.2.688.0	November 6, 2024	**This version was deprecated on 11/20/2024.** **Enhancements**: • Added credentials to OpenDataChannel requests. • Upgraded the `testify` and `objx` dependent packages.
1.2.677.0	October 10, 2024	**Enhancement**: Added support for passing the plugin version with OpenDataChannel requests.
1.2.650.0	July 02, 2024	**Enhancement**: Upgraded aws-sdk-go to 1.54.10. **Bug fix**: Reformated comments for gofmt check.
1.2.633.0	May 30, 2024	**Enhancement**: Updated the Dockerfile to use an Amazon Elastic Container Registry (Amazon ECR) image.
1.2.553.0	January 10, 2024	**Enhancement**: Upgraded aws-sdk-go and dependent Golang packages.

Version	Release date	Details
1.2.536.0	December 4, 2023	**Enhancement**: Added support for passing a StartSession API response as an environment variable to session-manager-plugin.
1.2.497.0	August 1, 2023	**Enhancement**: Upgraded Go SDK to v1.44.302.
1.2.463.0	March 15, 2023	**Enhancement**: Added Mac with Apple silicon support for Apple Mac (M1) in macOS bundle installer and signed installer.
1.2.398.0	October 14, 2022	**Enhancement**: Support golang version 1.17. Update default session-manager-plugin runner for macOS to use python3. Update import path from SSMCLI to session-manager-plugin.
1.2.339.0	June 16, 2022	**Bug fix**: Fix idle session timeout for port sessions.
1.2.331.0	May 27, 2022	**Bug fix**: Fix port sessions closing prematurely when the local server doesn't connect before timeout.
1.2.323.0	May 19, 2022	**Bug fix**: Disable smux keep alive to use idle session timeout feature.
1.2.312.0	March 31, 2022	**Enhancement**: Supports more output message payload types.
1.2.295.0	January 12, 2022	**Bug fix**: Hung sessions caused by client resending stream data when agent becomes inactive, and incorrect logs for `start_publication` and `pause_publication` messages.
1.2.279.0	October 27, 2021	**Enhancement**: Zip packaging for Windows platform.
1.2.245.0	August 19, 2021	**Enhancement**: Upgrade `aws-sdk-go` to latest version (v1.40.17) to support AWS IAM Identity Center.

Version	Release date	Details
1.2.234.0	July 26, 2021	**Bug fix**: Handle session abruptly terminated scenario in interactive session type.
1.2.205.0	June 10, 2021	**Enhancement**: Added support for signed macOS installer.
1.2.54.0	January 29, 2021	**Enhancement**: Added support for running sessions in NonInteractiveCommands execution mode.
1.2.30.0	November 24, 2020	**Enhancement**: (Port forwarding sessions only) Improved overall performance.
1.2.7.0	October 15, 2020	**Enhancement**: (Port forwarding sessions only) Reduced latency and improved overall performance.
1.1.61.0	April 17, 2020	**Enhancement**: Added ARM support for Linux and Ubuntu Server.
1.1.54.0	January 6, 2020	**Bug fix**: Handle race condition scenario of packets being dropped when the Session Manager plugin isn't ready.
1.1.50.0	November 19, 2019	**Enhancement**: Added support for forwarding a port to a local unix socket.
1.1.35.0	November 7, 2019	**Enhancement**: (Port forwarding sessions only) Send a TerminateSession command to SSM Agent when the local user presses `Ctrl+C`.
1.1.33.0	September 26, 2019	**Enhancement**: (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	**Enhancement**: Update to keep port forwarding session open until remote server closes the connection.
1.1.26.0	July 30, 2019	**Enhancement**: Update to limit the rate of data transfer during a session.

Version	Release date	Details
1.1.23.0	July 9, 2019	**Enhancement**: Added support for running SSH sessions using Session Manager.
1.1.17.0	April 4, 2019	**Enhancement**: Added support for further encryption of session data using AWS Key Management Service (AWS KMS).
1.0.37.0	September 20, 2018	**Enhancement**: Bug fix for Windows version.
1.0.0.0	September 11, 2018	Initial release of the Session Manager plugin.

Install the Session Manager plugin on Windows

You can install the Session Manager plugin on 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**
>
> Note the following information.
>
> - The Session Manager plugin installer needs Administrator rights to install the plugin.
> - 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 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.

   > **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).

Install the Session Manager plugin on macOS

Choose one of the following topics to install the Session Manager plugin on macOS.

> **Note**
>
> The signed installer is a signed `.pkg` file. The bundled installer uses a `.zip` file. After the file is unzipped, you can install the plugin using the binary.

Install the Session Manager plugin on macOS with the signed installer

This section describes how to 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.

   x86_64

   ```
   curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/mac/
   session-manager-plugin.pkg" -o "session-manager-plugin.pkg"
   ```

   Mac with Apple silicon

   ```
   curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/
   mac_arm64/session-manager-plugin.pkg" -o "session-manager-plugin.pkg"
   ```

2. Run the install commands. If the command fails, verify that the `/usr/local/bin` folder exists. If it doesn't, create it and run the command again.

   ```
   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.

Install the Session Manager plugin on macOS

This section describes how to install the Session Manager plugin on macOS using the bundled installer.

> **Important**
>
> Note the following important information.
>
> - By default, the installer requires sudo access to run, because the script installs the plugin to the `/usr/local/sessionmanagerplugin` system directory. If you don't want to install the plugin using sudo, manually update the installer script to install the plugin to a directory that doesn't require sudo access.
> - 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.

   x86_64

   ```
   curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/mac/
   sessionmanager-bundle.zip" -o "sessionmanager-bundle.zip"
   ```

   Mac with Apple silicon

   ```
   curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/
   mac_arm64/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/python3.8 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.

> **Note**
>
> 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 Linux

This section includes information about verifying the signature of the Session Manager plugin installer package and installing the plugin on the following Linux distributions:

- Amazon Linux 2
- AL2023
- RHEL
- Debian Server
- Ubuntu Server

Topics

- Verify the signature of the Session Manager plugin
- Install the Session Manager plugin on Amazon Linux 2, Amazon Linux 2023, and Red Hat Enterprise Linux distributions
- Install the Session Manager plugin on Debian Server and Ubuntu Server

Verify the signature of the Session Manager plugin

The Session Manager plugin RPM and Debian installer packages for Linux instances are cryptographically signed. You can use a public key to verify that the plugin binary and package is original and unmodified. If the file is altered or damaged, the verification fails. You can verify the signature of the installer package using the GNU Privacy Guard (GPG) tool. The following information is for Session Manager plugin versions 1.2.707.0 or later.

Complete the following steps to verify the signature of the Session Manager plugin installer package.

Topics

- Step 1: Download the Session Manager plugin installer package
- Step 2: Download the associated signature file
- Step 3: Install the GPG tool
- Step 4: Verify the Session Manager plugin installer package on a Linux server

Step 1: Download the Session Manager plugin installer package

Download the Session Manager plugin installer package you want to verify.

Amazon Linux 2, AL2023, and RHEL RPM packages

x86_64

```
curl -o "session-manager-plugin.rpm" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/linux_64bit/session-manager-plugin.rpm"
```

ARM64

```
curl -o "session-manager-plugin.rpm" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/linux_arm64/session-manager-plugin.rpm"
```

Debian Server and Ubuntu Server Deb packages

x86_64

```
curl -o "session-manager-plugin.deb" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_64bit/session-manager-plugin.deb"
```

ARM64

```
curl -o "session-manager-plugin.deb" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_arm64/session-manager-plugin.deb"
```

Step 2: Download the associated signature file

After you download the installer package, download the associated signature file for package verification. To provide an extra layer of protection against unauthorized copying or use of the session-manager-plugin binary file inside the package, we also offer binary signatures, which you can use to validate individual binary files. You can choose to use these binary signatures based on your security needs.

Amazon Linux 2, AL2023, and RHEL signature packages

x86_64

Package:

```
curl -o "session-manager-plugin.rpm.sig" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/linux_64bit/session-manager-plugin.rpm.sig"
```

Binary:

```
curl -o "session-manager-plugin.sig" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/linux_64bit/session-manager-plugin.sig"
```

ARM64

Package:

```
curl -o "session-manager-plugin.rpm.sig" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/linux_arm64/session-manager-plugin.rpm.sig"
```

Binary:

```
curl -o "session-manager-plugin.sig" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/linux_arm64/session-manager-plugin.sig"
```

Debian Server and Ubuntu Server Deb signature packages

x86_64

Package:

```
curl -o "session-manager-plugin.deb.sig" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_64bit/session-manager-plugin.deb.sig"
```

Binary:

```
curl -o "session-manager-plugin.sig" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_64bit/session-manager-plugin.sig"
```

ARM64

Package:

```
curl -o "session-manager-plugin.deb.sig" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_arm64/session-manager-plugin.deb.sig"
```

Binary:

```
curl -o "session-manager-plugin.sig" "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_arm64/session-manager-plugin.sig"
```

Step 3: Install the GPG tool

To verify the signature of the Session Manager plugin, you must have the GNU Privacy Guard (GPG) tool installed on your system. The verification process requires GPG version 2.1 or later. You can check your GPG version by running the following command:

```
gpg --version
```

If your GPG version is older than 2.1, update it before proceeding with the verification process. For most systems, you can update the GPG tool using your package manager. For example, on supported Amazon Linux and RHEL versions, you can use the following commands:

```
sudo yum update
sudo yum install gnupg2
```

On supported Ubuntu Server and Debian Server systems, you can use the following commands:

```
sudo apt-get update
sudo apt-get install gnupg2
```

Ensure you have the required GPG version before continuing with the verification process.

Step 4: Verify the Session Manager plugin installer package on a Linux server

Use the following procedure to verify the Session Manager plugin installer package on a Linux server.

> **Note**
>
> Amazon Linux 2 doesn't support the gpg tool version 2.1 or higher. If the following procedure doesn't work on your Amazon Linux 2 instances, verify the signature on a different platform before installing it on your Amazon Linux 2 instances.

1. Copy the following public key, and save it to a file named session-manager-plugin.gpg.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mFIEZ5ERQxMIKoZIzj0DAQcCAwQjuZy+IjFoYg57sLTGhF3aZLBaGpzB+gY6j7Ix
P7NqbpXyjVj8a+dy79gSd64OEaMxUb7vw/jug+CfRXwVGRMNtIBBV1MgU1NNIFNl
c3Npb24gTWFuYWdlciA8c2Vzc2lvbi1tYW5hZ2VyLXBsdWdpbi1zaWduZXJAYW1h
em9uLmNvbT4gKEFXUyBTeXN0ZW1zIE1hbmFnZXIgU2Vzc2lvbiBNYW5hZ2VyIFBs
dWdpbiBMaW51eCBTaWduZXIgS2V5KYkBAAQQEwgAqAUCZ5ERQ4EcQVdTIFNTTSBT
ZXNzaW9uIE1hbmFnZXIgPHNlc3Npb24tbWFuYWdlci1wbHVnaW4tc2lnbmVyQGFt
YXpvbi5jb20+IChBV1MgU3lzdGVtcyBNYW5hZ2VyIFNlc3Npb24gTWFuYWdlciBQ
bHVnaW4gTGludXggU2lnbmVyIEtleSkWIQR5WWNxJM4JOtUB1HosTUr/b2dX7gIe
AwIbAwIVCAAKCRAsTUr/b2dX7rO1AQCa1kig3lQ78W/QHGU76uHx3XAyv0tfpE9U
oQBCIwFLSgEA3PDHt3lZ+s6m9JLGJsy+Cp5ZFzpiF6RgluR/2gA861M=
```

```
=2DQm
-----END PGP PUBLIC KEY BLOCK-----
```

2. Import the public key into your keyring. The returned key value should be 2C4D4AFF6F6757EE.

```
$ gpg --import session-manager-plugin.gpg
gpg: key 2C4D4AFF6F6757EE: public key "AWS SSM Session Manager <session-manager-
plugin-signer@amazon.com> (AWS Systems Manager Session Manager Plugin Linux Signer
 Key)" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

3. Run the following command to verify the fingerprint.

```
gpg --fingerprint 2C4D4AFF6F6757EE
```

The fingerprint for the command output should match the following.

```
7959 6371 24CE 093A D501 D47A 2C4D 4AFF 6F67 57EE
```

```
pub   nistp256 2025-01-22 [SC]
      7959 6371 24CE 093A D501  D47A 2C4D 4AFF 6F67 57EE
uid           [ unknown] AWS SSM Session Manager <session-manager-plugin-
signer@amazon.com> (AWS Systems Manager Session Manager Plugin Linux Signer Key)
```

If the fingerprint doesn't match, don't install the plugin. Contact AWS Support.

4. Verify the installer package signature. Replace the *signature-filename* and *downloaded-plugin-filename* with the values you specified when downloading the signature file and session-manager-plugin, as listed in the table earlier in this topic.

```
gpg --verify signature-filename downloaded-plugin-filename
```

For example, for the x86_64 architecture on Amazon Linux 2, the command is as follows:

```
gpg --verify session-manager-plugin.rpm.sig session-manager-plugin.rpm
```

This command returns output similar to the following.

```
gpg: Signature made Mon Feb 3 20:08:32 2025 UTC gpg: using ECDSA key
 2C4D4AFF6F6757EE
gpg: Good signature from "AWS Systems Manager Session Manager <session-manager-
plugin-signer@amazon.com> (AWS Systems Manager Session Manager Plugin Linux Signer
 Key)" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 7959 6371 24CE 093A D501 D47A 2C4D 4AFF 6F67 57EE
```

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 package. 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 Session Manager plugin with version 1.2.707.0 or later.

Install the Session Manager plugin on Amazon Linux 2, Amazon Linux 2023, and Red Hat Enterprise Linux distributions

Use the following procedure to install the Session Manager plugin on Amazon Linux 2, Amazon Linux 2023 (AL2023), and RHEL distributions.

1. Download and install the Session Manager plugin RPM package.

   x86_64

   On Amazon Linux 2 and RHEL 7, run the following command:

   ```
   sudo yum install -y https://s3.amazonaws.com/session-manager-downloads/plugin/
   latest/linux_64bit/session-manager-plugin.rpm
   ```

   On AL2023 and RHEL 8 and 9, run the following command:

   ```
   sudo dnf install -y https://s3.amazonaws.com/session-manager-downloads/plugin/
   latest/linux_64bit/session-manager-plugin.rpm
   ```

   ARM64

   On Amazon Linux 2 and RHEL 7, run the following command:

```
sudo yum install -y https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_arm64/session-manager-plugin.rpm
```

On AL2023 and RHEL 8 and 9, run the following command:

```
sudo dnf install -y https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_arm64/session-manager-plugin.rpm
```

2. Verify that the installation was successful. For information, see Verify the Session Manager plugin installation.

> **Note**
>
> If you want to uninstall the plugin, run `sudo yum erase session-manager-plugin -y`

Install the Session Manager plugin on Debian Server and 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"
   ```

   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.

> **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 start-session command in the the AWS Command Line Interface (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 you have installed and configured the AWS CLI, and 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)
- Enable logging for the Session Manager plugin (Linux and macOS)

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 `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.

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)
- Starting a session (Amazon EC2 console)
- Starting a session (AWS CLI)
- Starting a session (SSH)
- Starting a session (port forwarding)
- Starting a session (port forwarding to remote host)
- Starting a session (interactive and noninteractive commands)

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.

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**.
3. Choose **Start session**.

4. (Optional) Enter a session description in the **Reason for session** field.
5. For **Target instances**, choose the option button to the left of the managed node that you want to connect to.

   If the node that you want isn't in the list, or if you select a node and receive a configuration error, see Managed node not available or not configured for Session Manager for troubleshooting steps.
6. Choose **Start session** to launch the session immediately.

   -or-

   Choose **Next** for session options.
7. (Optional) For **Session document**, select the document that you want to run when the session starts. If your document supports runtime parameters, you can enter one or more comma-separated values in each parameter field.
8. Choose **Next**.
9. 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.

> **Important**
>
> If you want to allow users to specify a document when starting sessions in the Session Manager console, note the following:
>
> - You must grant users the `ssm:GetDocument` and `ssm:ListDocuments` permissions in their IAM policy. For more information, see Grant access to custom Session documents in the console.
> - The console only supports Session documents that have the `sessionType` defined as `Standard_Stream`. For more information, see Session document schema.

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 sign-in credentials. Ask that person to update your policies to allow you to start sessions from the Amazon EC2 console. If you're an administrator, see Sample IAM policies for Session Manager 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 Installing or updating the latest version of the AWS CLI.

Before you start a session, make sure that you have completed the setup steps for Session Manager. For information, see Setting up Session Manager.

To use the AWS CLI to run session commands, the Session Manager plugin must also be installed on your local machine. For information, see Install the Session Manager plugin for the AWS CLI.

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) Allow and control permissions for SSH connections through Session Manager.
- 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*.
- 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.

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.
To use the AWS CLI to run session commands, you must install the Session Manager plugin on your local machine. For information, see Install the Session Manager plugin for the AWS CLI.
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"
```

`portNumber` is the remote port on the managed node where you want the session traffic to be redirected. For example, you might specify port 3389 for connecting to a Windows node using the Remote Desktop Protocol (RDP). If you don't specify the `portNumber` parameter, Session Manager uses `80` as the default value.

`localPortNumber` is the port on your local computer where traffic starts, 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 Manager Session 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 3.1.1374.0 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.
> To use the AWS CLI to run session commands, you must install the Session Manager plugin on your local machine. For information, see Install the Session Manager plugin for the AWS CLI.
> 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 AWS 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 `host` value represents the hostname or IP address of the remote host that you want to connect to. General connectivity and name resolution requirements between the managed node and the remote host still apply.

`portNumber` is the remote port on the managed node where you want the session traffic to be redirected. For example, you might specify port 3389 for connecting to a Windows node using the Remote Desktop Protocol (RDP). If you don't specify the `portNumber` parameter, Session Manager uses `80` as the default value.

`localPortNumber` is the port on your local computer where traffic starts, 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 with an Amazon ECS task

Session Manager supports starting a port forwarding session with a task inside an Amazon Elastic Container Service (Amazon ECS) cluster. To do so, you must update the task role in IAM to include the following permissions:

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
       "Effect": "Allow",
       "Action": [
            "ssmmessages:CreateControlChannel",
            "ssmmessages:CreateDataChannel",
            "ssmmessages:OpenControlChannel",
            "ssmmessages:OpenDataChannel"
       ],
      "Resource": "*"
      }
   ]
}
```

To start a port forwarding session with an Amazon ECS task, run the following command from the AWS CLI. Replace each *example resource placeholder* with your own information.

> **Note**
>
> Remove the < and > symbols from the `target` parameter. These symbols are provided for reader clarification only.

Linux & macOS

```
aws ssm start-session \
    --target ecs:<ECS_cluster_name>_<ECS_container_ID>_<container_runtime_ID> \
    --document-name AWS-StartPortForwardingSessionToRemoteHost \
    --parameters '{"host":["URL"],"portNumber":["port_number"], "localPortNumber":
["port_number"]}'
```

Windows

```
aws ssm start-session ^
    --target ecs:<ECS_cluster_name>_<ECS_container_ID>_<container_runtime_ID> ^
```

```
    --document-name AWS-StartPortForwardingSessionToRemoteHost ^
    --parameters host="URL",portNumber="port_number",localPortNumber="port_number"
```

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.

To use the AWS CLI to run session commands, the Session Manager plugin must also be installed on your local machine. For information, see Install the Session Manager plugin for the AWS CLI.

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.

More info

- Use port forwarding in AWS Systems Manager Session Manager to connect to remote hosts
- Amazon EC2 instance port forwarding with AWS Systems Manager
- Manage AWS Managed Microsoft AD resources with Session Manager port forwarding
- Port Forwarding Using AWS Systems Manager Session Manager on the *AWS News Blog*.

End a session

You can end a session that you started in your account using the AWS Systems Manager console or the AWS Command Line Interface (AWS CLI). When you choose the **Terminate** button for a session in the console or call the TerminateSession API action by using the AWS CLI, Session Manager permanently ends the session and closes the data connection between the Session Manager client and SSM Agent on the managed node. You can't resume a terminated session.

If there is no user activity in an open session for 20 minutes, the idle state triggers a timeout. Session Manager doesn't call `TerminateSession`, but it does close the underlying channel. You can't resume a session closed because of idle timeout.

We recommend always explicitly terminating a session by using the `terminate-session` command, when using the AWS CLI, or the **Terminate** button when using the console. (**Terminate** buttons are located on both the session window and main Session Manager console page.) If you only close a browser or command window, the session remains listed as **Active** in the console for 30 days. When you don't explicitly terminate a session, or when a session times out, any processes that were running on the managed node at the time will continue to run.

Topics

- Ending a session (console)
- Ending a session (AWS CLI)

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 Enabling and disabling session logging.

Topics

- Viewing session history (console)
- Viewing session history (AWS CLI)

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 the **Session history** tab.

   -or-

   If the Session Manager home page opens first, choose **Configure Preferences** and then 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.

Logging 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 log 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.

> **Note**
>
> For recurring, historical, analytical analysis of your log files, consider querying CloudTrail logs using CloudTrail Lake or a table you maintain. For more information, see Querying AWS CloudTrail logs in the *AWS CloudTrail User Guide*.

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 Amazon EventBridge rules that react to events in the *Amazon EventBridge User Guide*.

   As you follow the steps to create the rule, make the following selections:

   - For **AWS service**, 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 **Select a target**, 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*.

Enabling and disabling session logging

Session logging records information about current and completed sessions in the Systems Manager console. You can also log details about commands run during sessions in your AWS account. Session logging enables 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 for session logging 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 actions performed during a session, such as running an AWS Lambda function, starting an AWS CodePipeline pipeline, or running an AWS Systems Manager Run Command document.

Important

Note 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 2, AL2023 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).

Refer to the following topics for more information about logging options for Session Manager.

Topics

- Streaming session data using Amazon CloudWatch Logs (console)
- Logging session data using Amazon S3 (console)
- Logging session data using Amazon CloudWatch Logs (console)
- Configuring session logging to disk
- Adjusting how long the Session Manager temporary log file is stored on disk
- Disabling Session Manager logging in CloudWatch Logs and Amazon S3

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 about updating PowerShell using Run Command, see Updating PowerShell using Run Command.

> **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).

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).

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*.

Configuring session logging to disk

After you enable Session Manager logging to CloudWatch or Amazon S3, all commands executed during a session (and the resulting output from those commands) are logged to a temporary file on the disk of the target instance. The temporary file is named `ipcTempFile.log`.

The `ipcTempFile.log` is controlled by the `SessionLogsDestination` parameter in the SSM Agent configuration file. This parameter accepts the following values:

- **disk**: If you specify this parameter and session logging to CloudWatch or Amazon S3 are *enabled*, SSM Agent creates the `ipcTempFile.log` temporary log file and logs session commands and output to disk. Session Manager uploads this log to either CloudWatch or S3 during or after the session, depending on the logging configuration. The log is then deleted according to the duration specified for the SSM Agent `SessionLogsRetentionDurationHours` configuration parameter.

  If you specify this parameter and session logging to CloudWatch and Amazon S3 are *disabled*, SSM Agent still logs command history and output in the `ipcTempFile.log` file. The file will be deleted according to the duration specified for the SSM Agent `SessionLogsRetentionDurationHours` configuration parameter.
- **none**: If you specify this parameter and session logging to CloudWatch or Amazon S3 are *enabled*, logging to disk works exactly as it does as if you'd specified the `disk` parameter. SSM Agent requires the temporary file when session logging to CloudWatch or Amazon S3 are enabled.

If you specify this parameter and session logging to CloudWatch or Amazon S3 are *disabled*, SSM Agent doesn't create the `ipcTempFile.log` file.

Use the following procedure to enable or disable creating the `ipcTempFile.log` temporary log file to disk when a session is stared.

To enable or disable creating the Session Manager temporary log file to disk

1. Either install SSM Agent on your instance or upgrade to version 3.2.2086 or higher. For information about how to check the agent version number, see Checking the SSM Agent version number. For information about how to manually install the agent, locate the procedure for your operating system in the following sections:
   - Manually installing and uninstalling SSM Agent on EC2 instances for Linux
   - Manually installing and uninstalling SSM Agent on EC2 instances for macOS
   - Manually installing and uninstalling SSM Agent on EC2 instances for Windows Server
2. Connect to your instance and locate the `amazon-ssm-agent.json` file in the following location.
   - **Linux**: /etc/amazon/ssm/
   - **macOS**: /opt/aws/ssm/
   - **Windows Server**: C:\Program Files\Amazon\SSM

   If the file `amazon-ssm-agent.json` doesn't exist, copy the contents of the `amazon-ssm-agent.json.template` to a new file in the same directory. Name the new file `amazon-ssm-agent.json`.
3. Specify either none or `disk` for the `SessionLogsDestination` parameter. Save your changes.
4. Restart SSM Agent.

If you specified `disk` for the `SessionLogsDestination` parameter, you can verify that SSM Agent creates the temporary log file by starting a new session and then locating the `ipcTempFile.log` in the following location:

- **Linux**: /var/lib/amazon/ssm/*target ID*/session/orchestration/*session ID*/Standard_Stream/ipcTempFile.log
- **macOS**: /opt/aws/ssm/data/*target ID*/session/orchestration/*session ID*/Standard_Stream/ipcTempFile.log
- **Windows Server**: C:\ProgramData\Amazon\SSM\InstanceData*target ID*\session\orchestration*session ID*\Standard_Stream\ipcTempFile.log

> **Note**
>
> By default, the temporary log file is saved on the instance for 14 days.

If you want to update the `SessionLogsDestination` parameter across multiple instances, we recommend you create an SSM Document that specifies the new configuration. You can then use Systems Manager Run Command to implement the change on your instances. For more information, see Writing your own AWS Systems Manager documents (blog) and Running commands on managed nodes.

Adjusting how long the Session Manager temporary log file is stored on disk

After you enable Session Manager logging to CloudWatch or Amazon S3, all commands executed during a session (and the resulting output from those commands) are logged to a temporary file on the disk of the target instance. The temporary file is named `ipcTempFile.log`. During a session, or after it is completed, Session Manager uploads this temporary log to either CloudWatch or S3. The temporary log is then deleted according to the duration specified for the SSM Agent `SessionLogsRetentionDurationHours` configuration parameter. By default, the temporary log file is saved on the instance for 14 days in the following location:

- **Linux**: /var/lib/amazon/ssm/*target ID*/session/orchestration/*session ID*/Standard_Stream/ipcTempFile.log
- **macOS**: /opt/aws/ssm/data/*target ID*/session/orchestration/*session ID*/Standard_Stream/ipcTempFile.log
- **Windows Server**: C:\ProgramData\Amazon\SSM\InstanceData*target ID*\session\orchestration*session ID*\Standard_Stream\ipcTempFile.log

Use the following procedure to adjust how long the Session Manager temporary log file is stored on disk.

To adjust how long the `ipcTempFile.log` file is stored on disk

1. Connect to your instance and locate the `amazon-ssm-agent.json` file in the following location.

   - **Linux**: /etc/amazon/ssm/
   - **macOS**: /opt/aws/ssm/
   - **Windows Server**: C:\Program Files\Amazon\SSM

   If the file `amazon-ssm-agent.json` doesn't exist, copy the contents of the `amazon-ssm-agent.json.template` to a new file in the same directory. Name the new file `amazon-ssm-agent.json`.
2. Change the value of `SessionLogsRetentionDurationHours` to the desired number of hours. If `SessionLogsRetentionDurationHours` is set to 0, the temporary log file is created during the session and deleted when the session is completed. This setting should ensure the log file doesn't persist after the session ends.
3. Save your changes.
4. Restart SSM Agent.

Disabling Session Manager logging in CloudWatch Logs and Amazon S3

You can use the Systems Manager console or AWS CLI to disable session logging in your account.

To disable session logging (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. To disable CloudWatch logging, in the **CloudWatch logging** section, clear the **Enable** checkbox.
5. To disable S3 logging, in the **S3 logging** section, clear the **Enable** checkbox.
6. Choose **Save**.

To disable session logging (AWS CLI)

To disable session logging using the AWS CLI, follow the instructions in Update Session Manager preferences (command line).

In your JSON file, ensure that the `s3BucketName` and `cloudWatchLogGroupName` inputs contain no values. For example:

```
"inputs": {
        "s3BucketName": "",
        ...
        "cloudWatchLogGroupName": "",
        ...
    }
```

Alternatively, to disable logging, you can remove all `S3*` and `cloudWatch*` inputs from your JSON file.

> **Note**
>
> Depending on your configuration, after you disable CloudWatch or S3, a temporary log file might still be generated to disk by SSM Agent. For information about how to disable logging to disk, see Configuring session logging to disk.

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

The schema version of the Session document. Session documents only support version 1.0.

Type: String

Required: Yes

description

A description you specify for the Session document. For example, "Document to start port forwarding session with Session Manager".

Type: String

Required: No

sessionType

The type of session the Session document is used to establish.

Type: String

Required: Yes

Valid values: `InteractiveCommands` | `NonInteractiveCommands` | `Port` | `Standard_Stream`

inputs

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

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

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

If set to `true`, the Amazon S3 bucket you specified in the `s3BucketName` input must be encrypted.

Type: Boolean

Required: Yes

cloudWatchLogGroupName

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

If set to `true`, the log group you specified in the `cloudWatchLogGroupName` input must be encrypted.

Type: Boolean

Required: Yes

cloudWatchStreamingEnabled

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

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

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 and macOS managed nodes.

Type: Boolean

Required: Yes

runAsDefaultUser

The name of the user account to start sessions with on Linux and macOS 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

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

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

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

The shell preferences, environment variables, working directories, and commands you specify for sessions on Windows Server managed nodes.

Type: String

Required: No

linux

The shell preferences, environment variables, working directories, and commands you specify for sessions on Linux and macOS managed nodes.

Type: String

Required: No

parameters

An object that defines the parameters the document accepts. For more information about defining document parameters, see **parameters** in the Top-level data elements. 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.

Type: StringMap

Required: No

properties

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. You can also determine whether commands are run as `root` using the `runAsElevated` boolean property. For more information, see Restrict access to commands in a session.

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

- AccessDeniedException when calling the TerminateSession operation
- Document process failed unexpectedly: document worker timed out
- Session Manager can't connect from the Amazon EC2 console
- No permission to start a session
- SSM Agent not online
- No permission to change session preferences
- Managed node not available or not configured for Session Manager

- Session Manager plugin not found
- Session Manager plugin not automatically added to command line path (Windows)
- Session Manager plugin becomes unresponsive
- TargetNotConnected
- Blank screen displays after starting a session
- Managed node becomes unresponsive during long running sessions
- An error occurred (InvalidDocument) when calling the StartSession operation

AccessDeniedException when calling the TerminateSession operation

Problem: When attempting to terminate a session, Systems Manager returns the following error:

```
An error occurred (AccessDeniedException) when calling the TerminateSession operation:
User: <user_arn> is not authorized to perform: ssm:TerminateSession on resource:
<ssm_session_arn> because no identity-based policy allows the ssm:TerminateSession
 action.
```

Solution A: Confirm that the latest version of the Session Manager plugin is installed on the node

Enter the following command in the terminal and press Enter.

```
session-manager-plugin --version
```

Solution B: Install or reinstall the latest version of the plugin

For more information, see Install the Session Manager plugin for the AWS CLI.

Solution C: Attempt to reestablish a connection to the node

Verify that the node is responding to requests. Try reestablishing the session. Or, if necessary, open the Amazon EC2 console and verify the status of the instance is running.

Document process failed unexpectedly: document worker timed out

Problem: When starting a session to a Linux host, Systems Manager returns the following error:

```
document process failed unexpectedly: document worker timed out,
```

```
check [ssm-document-worker]/[ssm-session-worker] log for crash reason
```

If you configured SSM Agent logging, as described in Viewing SSM Agent logs, you can view more details in the debugging log. For this issue, Session Manager shows the following log entry:

```
failed to create channel: too many open files
```

This error typically indicates that there are too many Session Manager worker processes running and the underlying operating system reached a limit. You have two options for resolving this issue.

Solution A: Increase the operating system file notification limit

You can increase the limit by running the following command from a separate Linux host. This command uses Systems Manager Run Command. The specified value increases `max_user_instances` to 8192. This value is considerably higher than the default value of 128, but it won't strain host resources:

```
aws ssm send-command --document-name AWS-RunShellScript \
--instance-id i-02573cafcfEXAMPLE  --parameters \
"commands=sudo sysctl fs.inotify.max_user_instances=8192"
```

Solution B: Decrease the file notifications used by Session Manager in the target host

Run the following command from a separate Linux host to list sessions running on the target host:

```
aws ssm describe-sessions --state Active --filters key=Target,value=i-02573cafcfEXAMPLE
```

Review the command output to identify sessions that are no longer needed. You can terminate those session by running the following command from a separate Linux host:

```
aws ssm terminate-session –session-id session ID
```

Optionally, once there are no more sessions running on the remote server, you can free additional resources by running the following command from a separate Linux host. This command terminates all Session Manager processes running on the remote host, and consequently all sessions to the remote host. Before you run this command, verify there are no ongoing sessions you would like to keep:

```
aws ssm send-command --document-name AWS-RunShellScript \
            --instance-id i-02573cafcfEXAMPLE --parameters \
'{"commands":["sudo kill $(ps aux | grep ssm-session-worker | grep -v grep | awk
 '"'"'{print $2}'"'"')"]}'
```

Session Manager can't connect from the Amazon EC2 console

Problem: After creating a new instance, the **Connect** button > **Session Manager** tab in the Amazon Elastic Compute Cloud (Amazon EC2) console doesn't give you the option to connect.

Solution A: Create an instance profile: If you haven't already done so (as instructed by the information on the **Session Manager** tab in the EC2 console), create an AWS Identity and Access Management (IAM) instance profile by using Quick Setup. Quick Setup is a tool in AWS Systems Manager.

Session Manager requires an IAM instance profile to connect to your instance. You can create an instance profile and assign it to your instance by creating a host management configuration with Quick Setup. A *host management configuration* creates an instance profile with the required permissions and assigns it to your instance. A host management configuration also enables other Systems Manager tools and creates IAM roles for running those tools. There is no charge to use Quick Setup or the tools enabled by the host management configuration. Open Quick Setup and create a host management configuration.

> **Important**
>
> After you create the host management configuration, Amazon EC2 can take several minutes to register the change and refresh the **Session Manager** tab. If the tab doesn't show you a **Connect** button after two minutes, reboot your instance. After it reboots, if you still don't see the option to connect, open Quick Setup and verify you have only one host management configuration. If there are two, delete the older configuration and wait a few minutes.

If you still can't connect after creating a host management configuration, or if you receive an error, including an error about SSM Agent, see one of the following solutions:

- Solution B: No error, but still can't connect
- Solution C: Error about missing SSM Agent

Solution B: No error, but still can't connect

If you created the host management configuration, waited several minutes before trying to connect, and still can't connect, then you might need to manually apply the host management configuration to your instance. Use the following procedure to update a Quick Setup host management configuration and apply changes to an instance.

To update a host management configuration using 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**.
3. In the **Configurations** list, choose the **Host Management** configuration you created.
4. Choose **Actions**, and then choose **Edit configuration**.
5. Near the bottom of the **Targets** section, under **Choose how you want to target instances**, choose **Manual**.
6. In the **Instances** section, choose the instance you created.
7. Choose **Update**.

Wait a few minutes for EC2 to refresh the **Session Manager** tab. If you still can't connect or if you receive an error, review the remaining solutions for this issue.

Solution C: Error about missing SSM Agent

If you weren't able to create a host management configuration by using Quick Setup, or if you received an error about SSM Agent not being installed, you may need to manually install SSM Agent on your instance. SSM Agent is Amazon software that enables Systems Manager to connect to your instance by using Session Manager. SSM Agent is installed by default on most Amazon Machine Images (AMIs). If your instance was created from a non-standard AMI or an older AMI, you might have to manually install the agent. For the procedure to install SSM Agent, see the following topic that corresponds to your instance operating system.

- Windows Server
- macOS
- AlmaLinux
- Amazon Linux 2 and AL2023
- Debian Server

- Oracle Linux
- Red Hat Enterprise Linux
- Rocky Linux
- Ubuntu Server

For issues with SSM Agent, see Troubleshooting SSM Agent.

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.

SSM Agent not online

Problem: You see a message on the Amazon EC2 instance **Session Manager** tab that states: "SSM Agent is not online. The SSM Agent was unable to connect to a Systems Manager endpoint to register itself with the service."

Solution: SSM Agent is Amazon software that runs on Amazon EC2 instances so that Session Manager can connect to them. If you see this error, SSM Agent is unable to establish a connection with the Systems Manager endpoint. Possible sources of the problem could be firewall restrictions, routing problems, or lack of internet connectivity. To resolve this issue, investigate network connectivity problems. For more information, see Troubleshooting SSM Agent and Troubleshooting managed node availability. For information about Systems Manager endpoints, see AWS Systems Manager endpoints and quotas in the AWS General Reference.

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.

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 Systems Manager unified console for an organization.

  > **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 on Linux nodes and Configure SSM Agent to use a proxy for Windows Server instances.

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 tool. For information, see Verify or Create an IAM Instance Profile with Session Manager Permissions.
- **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 and uninstalling SSM Agent on EC2 instances for Windows Server, Manually installing and uninstalling SSM Agent on EC2 instances for Linux, or Manually installing and uninstalling SSM Agent on EC2 instances for macOS, 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 Updating the SSM Agent using Run Command.

> **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 with the AWS CLI.
> - Run `AWS-UpdateSSMAgent` as part of a maintenance window. For information about working with maintenance windows, see Create and manage maintenance windows using the console and Tutorial: Create and configure a maintenance window using the AWS CLI.

- **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 Install the Session Manager plugin for the AWS CLI.

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 the following example.

   - **64-bit machines**: `C:\Program Files\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 Install the Session Manager plugin for the AWS CLI.

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.
- **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 Collect metrics, logs, and traces with the CloudWatch agent in the *Amazon CloudWatch User Guide*.
- **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.
- **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.

An error occurred (InvalidDocument) when calling the StartSession operation

Problem: You receive the following error when starting a session by using the AWS CLI.

```
An error occurred (InvalidDocument) when calling the StartSession operation: Document
 type: 'Command' is not supported. Only type: 'Session' is supported for Session
 Manager.
```

Solution: The SSM document you specified for the `--document-name` parameter isn't a *Session* document. Use the following procedure to view a list of Session documents in the AWS Management Console.

To view a list of Session documents

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 **Categories** list, choose **Session documents**.

AWS Systems Manager State Manager

State Manager, a tool in 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.

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.
- Run scripts on Linux, macOS, and Windows Server managed nodes throughout their lifecycle.

To manage configuration drift across other AWS resources, you can use Automation, a tool in 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 Window AMIs.

For information about using State Manager with Automation runbooks, see Scheduling automations with State Manager associations.

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 most 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.

> **Note**
>
> State Manager doesn't currently support specifying months in cron expressions for associations.

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 information about building cron and rate expressions, see Reference: Cron and rate expressions for Systems Manager.

- **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.

- **Amazon S3 support**

  Store the command output from association runs in an Amazon S3 bucket of your choice. For more information, see Working with associations in Systems Manager.

- **EventBridge support**

  This Systems Manager tool 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 and Reference: Amazon EventBridge event patterns and types for Systems Manager.

Is there a charge to use State Manager?

State Manager is available at no additional charge.

Topics

- Understanding how State Manager works
- Working with associations in Systems Manager

- Creating associations that run MOF files
- Creating associations that run Ansible playbooks
- Creating associations that run Chef recipes
- Walkthrough: Automatically update SSM Agent with the AWS CLI
- Walkthrough: Automatically update PV drivers on EC2 instances for Windows Server

More info

- Combating Configuration Drift Using Amazon EC2 Systems Manager and Windows PowerShell DSC
- Configure Amazon EC2 Instances in an Auto Scaling Group Using State Manager

Understanding how State Manager works

State Manager, a tool in AWS Systems Manager, is a secure and scalable service that automates the process of keeping managed nodes in a hybrid and multicloud 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. If your targets include more than 1,000 nodes, the system uses an hourly throttling mechanism. This means you might see inaccuracies in your status aggregation count since the aggregation process runs hourly and only when the execution status for a node changes.
- 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.

> **Note**
>
> State Manager doesn't currently support specifying months in cron expressions for associations.

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 Working with associations in Systems Manager.

> **Note**
>
> API operations that are initiated by the SSM document during an association run are not logged in AWS CloudTrail.

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. 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.

Understanding when associations are applied to resources

When you create an association, you specify an SSM document that defines the configuration, a list of target resources, and a schedule for applying the configuration. By default, State Manager

runs the association when you create it and then according to your schedule. State Manager also attempts to run the association in the 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 SSM document that defines the association's configuration state. 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 the document that was specified when the association was created.
- **Manual start** – State Manager runs the association when initiated by the user from either the Systems Manager console or programmatically.
- **Target changes** – State Manager runs the association after any of the following activity occurs on a target node:
  - A managed node comes online for the first time.
  - A managed node comes online after missing a scheduled association run.
  - A managed node comes online after being stopped for more than 30 days.

> **Note**
>
> State Manager doesn't monitor documents or packages used in associations across AWS accounts. If you update a document or package in one account, the update won't cause the association to run in the second account. You must manually run the association in the second account.

Preventing associations from running when a target changes

In some cases, you might not want an association to run when a target that consists of managed nodes changes, but only according to its specified schedule.

> **Note**
>
> Running an Automation runbook incurs a cost. If an association with an Automation runbook targets all instances in your account and you regularly launch a large number of instances, the runbook is run on each of the instances when it launches. This can lead to elevated automation charges.

To prevent an association from running when the targets for that association change, select the **Apply association only at the next specified cron interval check box**. This check box is located in the **Specify schedule** area of the **Create association** and **Edit association** pages.

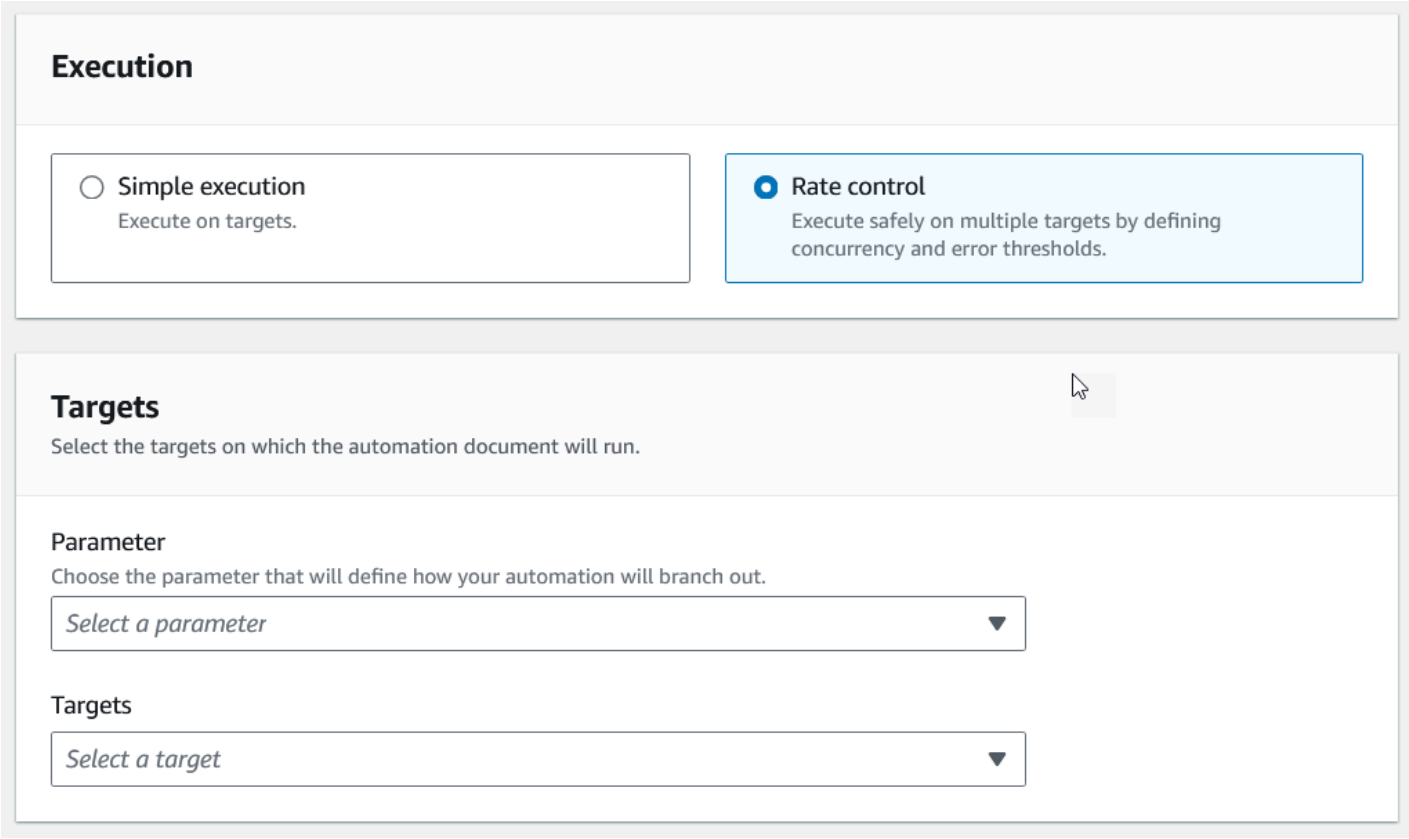
This option applies to associations that incorporate either an Automation runbook or an SSM document.

About target updates with Automation runbooks

In order for associations that are created with Automation runbooks to be applied when new target nodes are detected, the following conditions must be true:

- The association must have been created by a Quick Setup configuration. Quick Setup is a tool in AWS Systems Manager. Associations created by other processes are not currently supported.
- The Automation runbook must explicitly target the resource type `AWS::EC2::Instance` or `AWS::SSM::ManagedInstance`.
- The association must specify both parameters and targets.

  In the console, the **Parameter** and **Targets** fields are displayed when you choose a rate control execution.



When you use the CreateAssociation, CreateAssociationBatch, or UpdateAssociation API actions, you can specify these values using the `AutomationTargetParameterName` and `Targets` inputs. In each of these API actions, you can also prevent the association from running each time a target changes by setting the `ApplyOnlyAtCronInterval` parameter to `true`.

For information about using the console to control when associations run, including details for avoiding unexpectedly high costs for Automation executions, see Understanding when associations are applied to resources.

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

- Understanding targets and rate controls in State Manager associations
- Creating associations

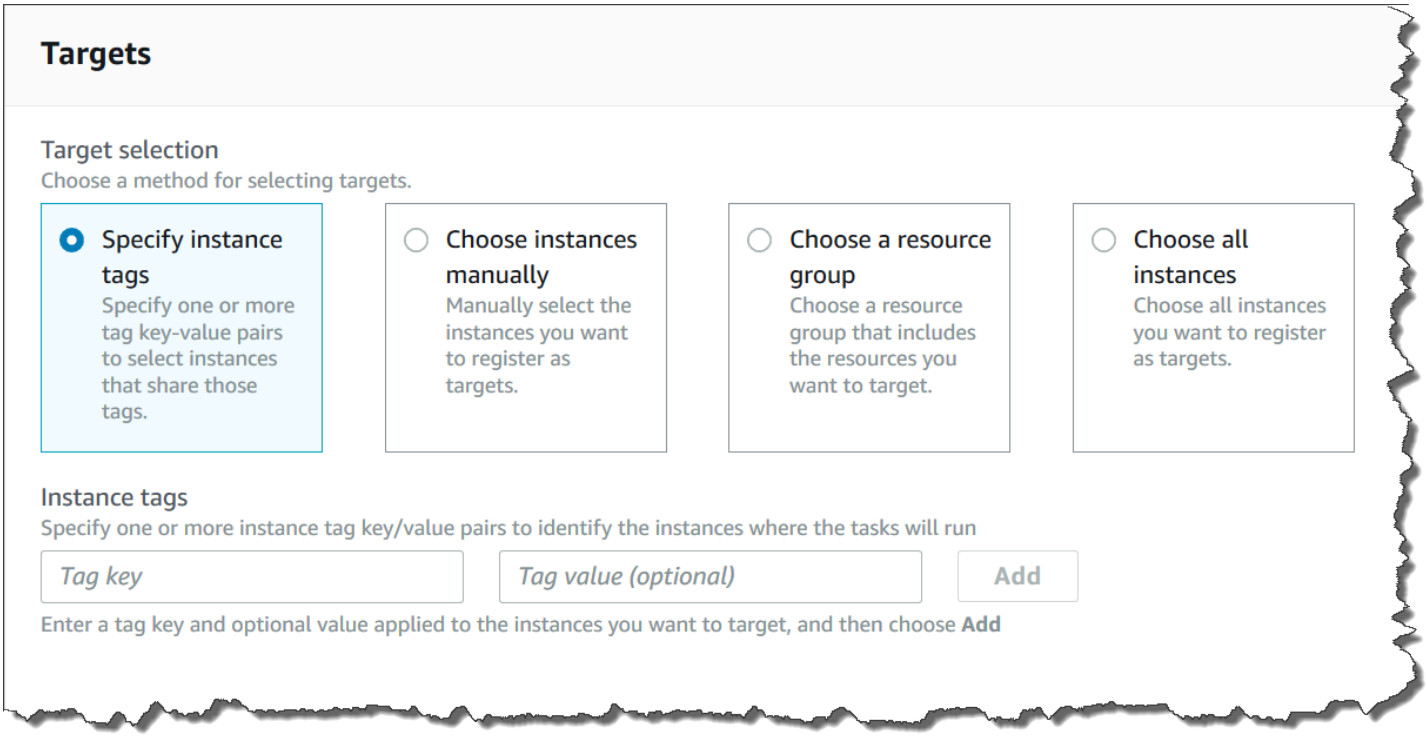
- Editing and creating a new version of an association
- Deleting associations
- Running Auto Scaling groups with associations
- Viewing association histories
- Working with associations using IAM

Understanding targets and rate controls in State Manager associations

This topic describes State Manager features that help you deploy an association to dozens or hundreds of nodes while controlling the number of nodes that run the association at the scheduled time. State Manager is a tool in AWS Systems Manager.

Using 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.

Each managed node can be targeted by a maximum of 20 associations.

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.

> **Note**
>
> If you use Automation runbooks with State Manager and the tagging limitation prevents you from achieving a specific goal, consider using Automation runbooks with Amazon EventBridge. For more information, see Run automations based on EventBridge events. For more information about using runbooks with State Manager, see Scheduling automations with State Manager associations.

As a best practice, we recommend using tags when creating associations that use a Command or Policy document. We also recommend using tags when creating associations to run Auto Scaling groups. For more information, see Running Auto Scaling groups with associations.

> **Note**
>
> Note the following information.
>
> - When creating an association in the AWS Management Console that targets nodes by using tags, you can specify only one tag key for an automation association and five tag keys for a command association. *All* tag keys specified in the association must be

currently assigned to the node. If they aren't, State Manager fails to target the node for an association.

- If you want to use the console *and* you want to target your nodes by using more than one tag key for an automation association and five tag keys for a command association, assign the tag keys to an AWS Resource Groups group and add the nodes to it. You can then choose the **Resource Group** option in the **Targets** list when you create the State Manager association.
- You can specify a maximum of five tag keys by using the AWS CLI. If you use the AWS CLI, *all* tag keys specified in the `create-association` command must be currently assigned to the node. If they aren't, State Manager fails to target the node for an association.

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 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 by design, the association runs on both Amazon Elastic Compute Cloud (Amazon EC2) instances and non-EC2 nodes in a hybrid and multicloud environment.

  The converse is also true. If you create an association that targets a resource group and the `AWS::EC2::Instance` resource type was specified for that group, then by design, the association runs on both non-EC2 nodes in a hybrid and multicloud environment and (Amazon EC2) instances.
- 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 create an association that targets a resource group using tags, you can't choose the **(empty value)** option for the tag value.
- 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.

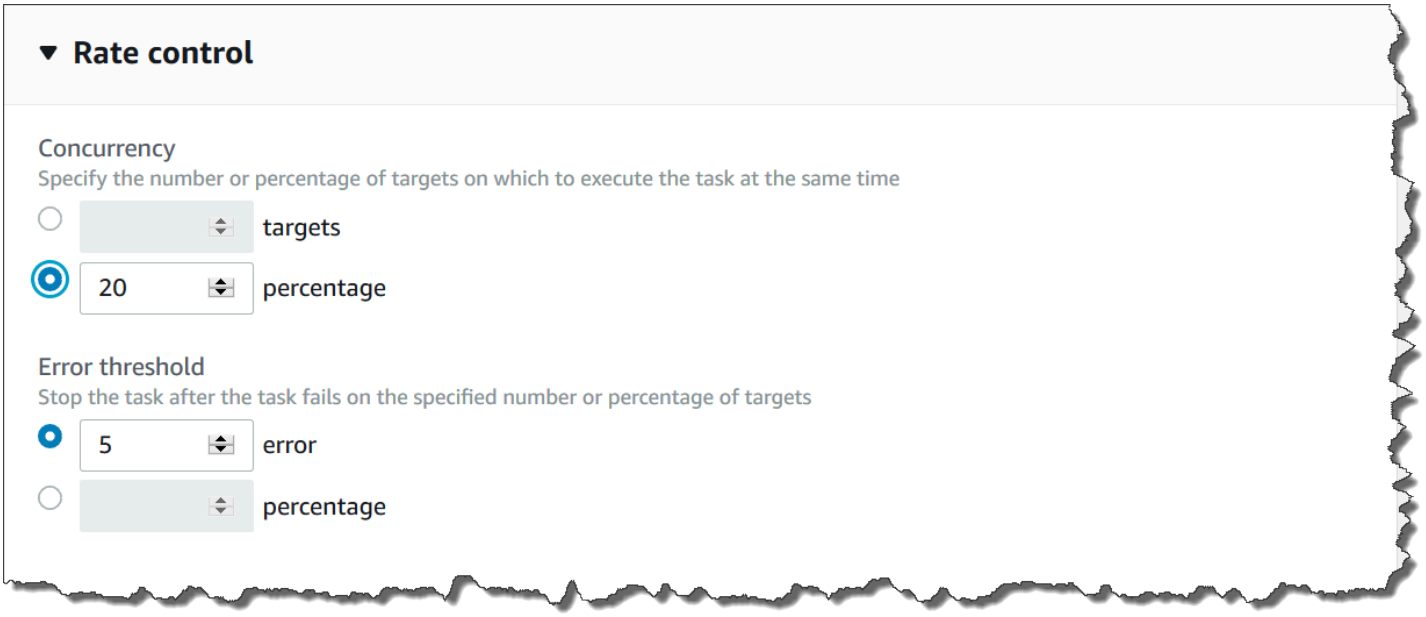
For more information about Resource Groups, see What Is AWS Resource Groups? in the *AWS Resource Groups User Guide*.

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.

Using 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 tool in 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.

> **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.

Association targets and rate controls

An association 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 Understanding targets and rate controls in State Manager associations.

Tagging associations

You can assign tags to an association when you create it by using a command line tool such as the AWS CLI or AWS Tools for PowerShell. Adding tags to an association by using the Systems Manager console isn't supported.

Running associations

By default, State Manager runs an association immediately after you create it, and then according to the schedule that you've defined.

The system also runs associations 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 runs the association after changes to the association's configuration, target nodes, documents, or parameters. For more information, see Understanding when associations are applied to resources
- State Manager records history for all skipped intervals. You can view the history on the **Execution History** tab.

Scheduling associations

You can schedule associations to run at basic intervals such as *every 10 hours*, or you can create more advanced schedules using custom cron and rate expressions. You can also prevent associations from running when you first create them.

Using cron and rate expressions to schedule association runs

In addition to standard cron and rate expressions, 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 select **Apply association only at the next specified Cron interval** in the console or specify the `ApplyOnlyAtCronInterval` parameter from the command line. When either of these options are activated, State Manager doesn't run the association immediately after you create it.

For more information about cron and rate expressions, see Reference: Cron and rate expressions for Systems Manager.

Create an association (console)

The following procedure describes how to use the Systems Manager console to create a State Manager association.

> **Note**
>
> Note the following information.
>
> - This procedure describes 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 Scheduling automations with State Manager associations.
> - When creating an association, you can specify a maximum of five tag keys by using the AWS Management Console. *All* tag keys specified for the association must be currently assigned to the node. If they aren't, State Manager fails to target the node for the association.

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**.
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 Scheduling automations with State Manager associations.

   > **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. (Optional) Choose a CloudWatch alarm to apply to your association for monitoring.

   > **Note**
   >
   > Note the following information about this step.
   >
   > - The alarms list displays a maximum of 100 alarms. If you don't see your alarm in the list, use the AWS Command Line Interface to create the association. For more information, see Create an association (command line).
   > - To attach a CloudWatch alarm to your command, the IAM principal that creates the association must have permission for the `iam:createServiceLinkedRole` action. For more information about CloudWatch alarms, see Using Amazon CloudWatch alarms.

- If your alarm activates, any pending command invocations or automations do not run.

8. For **Targets**, choose an option. For information about using targets, see Understanding targets and rate controls in State Manager associations.

> **Note**
>
> In order for associations that are created with Automation runbooks to be applied when new target nodes are detected, certain conditions must be met. For information, see About target updates with Automation runbooks.

9. 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**.

10. (Optional) In the **Schedule offset** field, specify a number between 1 and 6.
11. 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.

    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.

12. In the **Rate control** section, choose options to control how the association runs on multiple nodes. For more information about using rate controls, see Understanding targets and rate controls in State Manager associations.

    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.

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 Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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 can further restrict access by attaching IAM policies to 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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:PutObjectAcl"
            ],
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-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*.

> **Note**
>
> API operations that are initiated by the SSM document during an association run are not logged in AWS CloudTrail.

14. 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 Server) 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 Understanding targets and rate controls in State Manager associations.

> **Important**
>
> This procedure describes 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 Scheduling automations with State Manager associations.

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-02573cafcfEXAMPLE,i-0471e04240EXAMPLE,i-07782c72faEXAMPLE
```

Target instances by using 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
```

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 form another account, you must set the document version to `default`.

- State Manager doesn't support the `IncludeChildOrganizationUnits` parameter for TargetLocation.
- You can specify a maximum of five tag keys by using the AWS CLI. If you use the AWS CLI, *all* tag keys specified in the `create-association` command must be currently assigned to the node. If they aren't, State Manager fails to target the node for an association.
- 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.
- In order for associations that are created with Automation runbooks to be applied when new target nodes are detected, certain conditions must be met. For information, see About target updates with Automation runbooks.

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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

2. Use the following format to create a command that creates a State Manager association. Replace each *`example resource placeholder`* with your own information.

   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-expression "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 \
    --tags "Key=tag_key,Value=tag_value"
```

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-expression "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 ^
    --tags "Key=tag_key,Value=tag_value"
```

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 `
    -Tags "Key=tag_key,Value=tag_value"
```

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-expression "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-expression "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 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-expression "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-expression "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-expression "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-expression "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-expression "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-expression "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-expression "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-expression "cron(0 2 ? * SUN *)"
```

PowerShell

```
New-SSMAssociation `
```

```
  -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, targets, or other values. For associations based on SSM Command-type documents, 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.

Note

In order for associations that are created with Automation runbooks to be applied when new target nodes are detected, certain conditions must be met. For information, see About target updates with Automation runbooks.

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 form 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

For associations that use SSM Command documents, not Automation runbooks, 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**.
3. Choose an existing association, and then choose **Edit**.
4. Reconfigure the association to meet your current requirements.

   For information about association options with `Command` and `Policy` documents, see Creating associations. For information about association options with Automation runbooks, see Scheduling automations with State Manager associations.
5. Choose **Save Changes**.

6. (Optional) To view association information, 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.
7. (Optional) To view output for associations based on SSM Command documents, do the following:

   a. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   b. 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.)

   c. Drill down several levels, through the `awsrunPowerShell` folder, to the `stdout` file.

   d. 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 Server) 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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

2. Use the following format to create a command to edit and create a new version of an existing State Manager association. Replace each *`example resource placeholder`* with your own information.

   > **Important**
   >
   > When you call `update-association`, 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 action, we recommend that you call the `describe-association` operation and make a note of all optional parameters required for your `update-association` 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-expression "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-expression "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=amzn-s3-demo-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=amzn-s3-demo-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 amzn-s3-demo-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=amzn-s3-demo-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=amzn-s3-demo-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 amzn-s3-demo-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=amzn-s3-demo-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=amzn-s3-demo-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-SSMAssociation `
  -AssociationId b85ccafe-9f02-4812-9b81-01234EXAMPLE `
  -AssociationName MultiCalendarAssociation `
  -Parameter @{"commands"="echo Association"} `
  -S3Location_OutputS3BucketName amzn-s3-demo-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-SSMAssociation `
  -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": "amzn-s3-demo-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": "amzn-s3-demo-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"
            }
        ]
    }
}
```

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 associations

Use the following procedure to delete an association by using the AWS Systems Manager 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**.
3. Select an association and then choose **Delete**.

You can delete multiple associations in a single operation by running an automation from the AWS Systems Manager console. When you select multiple associations for deletion, State Manager launches the automation runbook start page with the association IDs entered as input parameter values.

To delete multiple associations in a single operation

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **State Manager**.
3. Select each association that you want to delete and then choose **Delete**.
4. (Optional) In the **Additional input parameters** area, select the Amazon Resource Name (ARN) for the *assume role* that you want the automation to use while running. To create a new assume role, choose **Create**.
5. Choose **Submit**.

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 Working with associations in Systems Manager.

   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 tool in 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.

> **Note**
>
> API operations that are initiated by the SSM document during an association run are not logged in AWS CloudTrail.

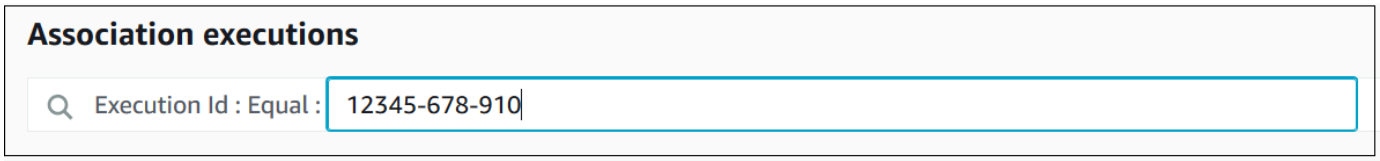
Viewing association histories (console)

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

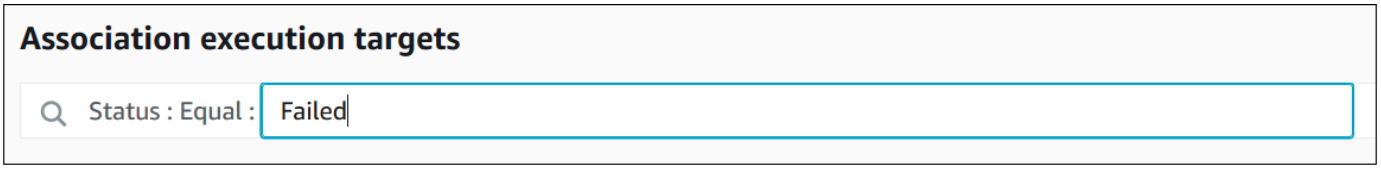
1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. Choose **State Manager**.
3. In the **Association id** field, choose an association for which you want to view the history.
4. Choose the **View details** button.
5. Choose the **Execution history** tab.
6. 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 Server) 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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.
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-SSMAssociationExecution `
  -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"="GREAT` value.

The system returns information like the following.

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-SSMAssociationExecution `
  -AssociationId 14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE `
  -Filter
 @{"Key"="CreatedTime";"Value"="2019-06-01T19:15:38.372Z";"Type"="LESS_THAN"}
```

The following returns all associations that were *successfully* run after a specific date and time.

Linux & macOS

```
aws ssm describe-association-executions \
  --association-id ID \
  --filters Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=GREATER_THAN
 Key=Status,Value=Success,Type=EQUAL
```

Windows

```
aws ssm describe-association-executions ^
  --association-id ID ^
  --filters Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=GREATER_THAN
 Key=Status,Value=Success,Type=EQUAL
```

PowerShell

```
Get-SSMAssociationExecution `
  -AssociationId 14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE `
  -Filter @{
      "Key"="CreatedTime";
      "Value"="2019-06-01T19:15:38.372Z";
      "Type"="GREATER_THAN"
    },
    @{
      "Key"="Status";
      "Value"="Success";
      "Type"="EQUAL"
    }
```

3. Run the following command to view all targets where the specific execution ran.

Linux & 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-02573cafcfEXAMPLE"
 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-02573cafcfEXAMPLE"
 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-02573cafcfEXAMPLE"
    },
    @{
      "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 tool in AWS Systems Manager, uses targets 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 node. 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 or Actions, resources, and condition keys for AWS Systems Manager in the *Service Authorization Reference*.

Creating associations that run MOF files

You can run Managed Object Format (MOF) files to enforce a target state on Windows Server managed nodes with State Manager, a tool in 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
- Resolving credentials in MOF files
- Using tokens in MOF files
- Prerequisites for creating associations that run MOF files
- Creating an association that runs MOF files
- Troubleshooting issues when creating associations that run MOF files
- Viewing DSC resource compliance details

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. 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 for creating associations that run MOF files

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**.
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. Systems Manager iterates through the list of MOF files and runs them in the order specified by the comma separated list.

    - An Amazon S3 bucket name. Bucket names must use lowercase letters. Specify this information by using the following format.

```
s3:amzn-s3-demo-bucket:MOF_file_name.mof
```

If you want to specify an AWS Region, then use the following format.

```
s3:bucket_Region:amzn-s3-demo-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 systems 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:amzn-s3-demo-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:amzn-s3-demo-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 Understanding targets and rate controls in State Manager associations.

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.

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 Understanding targets and rate controls in State Manager associations.

    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 Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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 tool in AWS Systems Manager. For more information, see AWS Systems Manager Run Command.

Troubleshooting issues when creating associations that run MOF files

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 when creating associations that run MOF files

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*:*amzn-s3-demo-bucket*:*mof-file-name*.mof.

  Here is an example: `s3:us-west-2:amzn-s3-demo-bucket:my-mof.mof`

  The Report, Status, and Module Source bucket names should be formatted as follows.

  bucket-region:*amzn-s3-demo-bucket*. Here is an example: `us-west-1:amzn-s3-demo-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
     ```

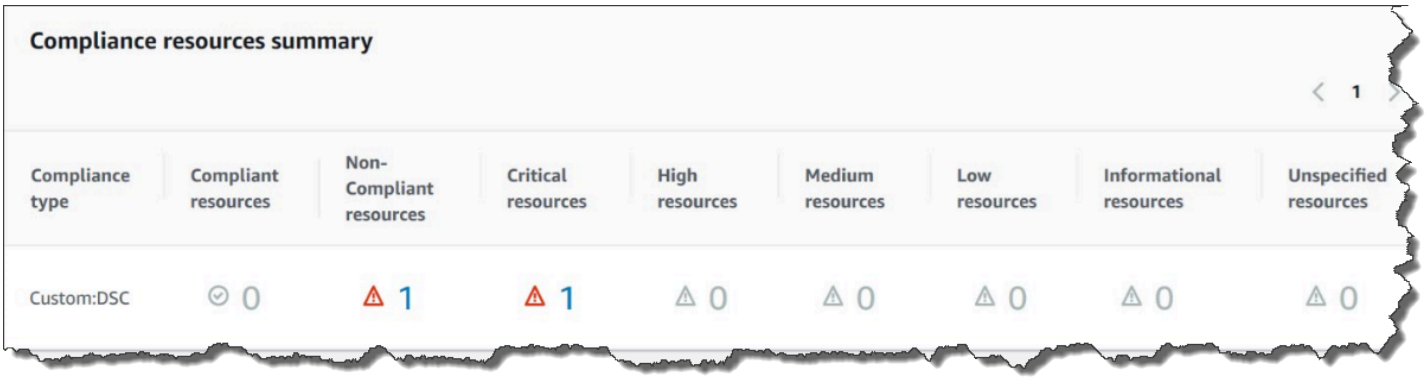
If the ping failed, it means that either Amazon S3 is down, or a firewall/transparent proxy is blocking access to the Amazon S3 Region, or the node can't access the internet.

Viewing DSC resource compliance details

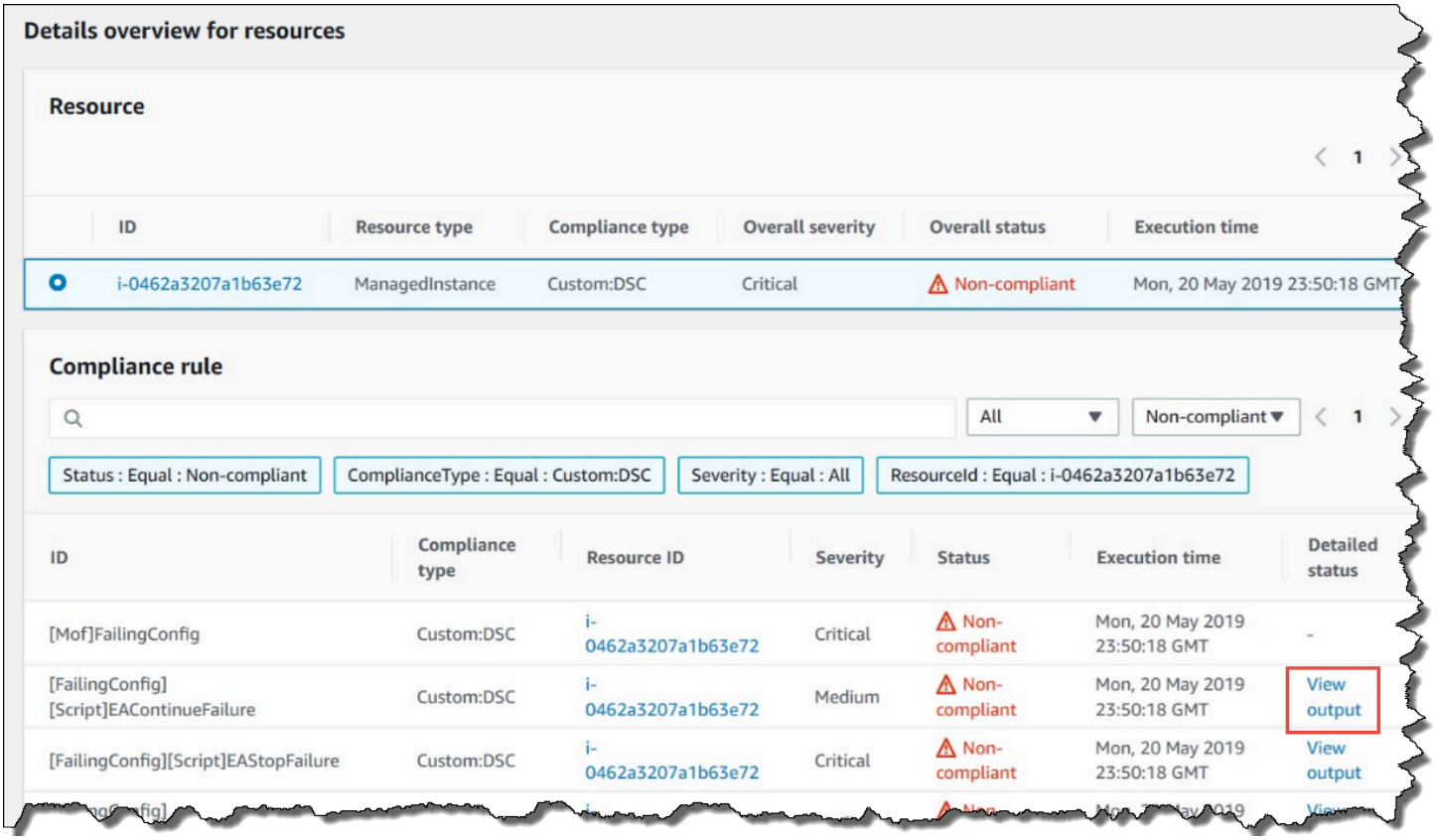
Systems Manager captures compliance information about DSC resource failures in the Amazon S3 **Status Bucket** you specified when you ran the `AWS-ApplyDSCMofs` document. Searching for information about DSC resource failures in an Amazon S3 bucket can be time consuming. Instead, you can view this information 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.

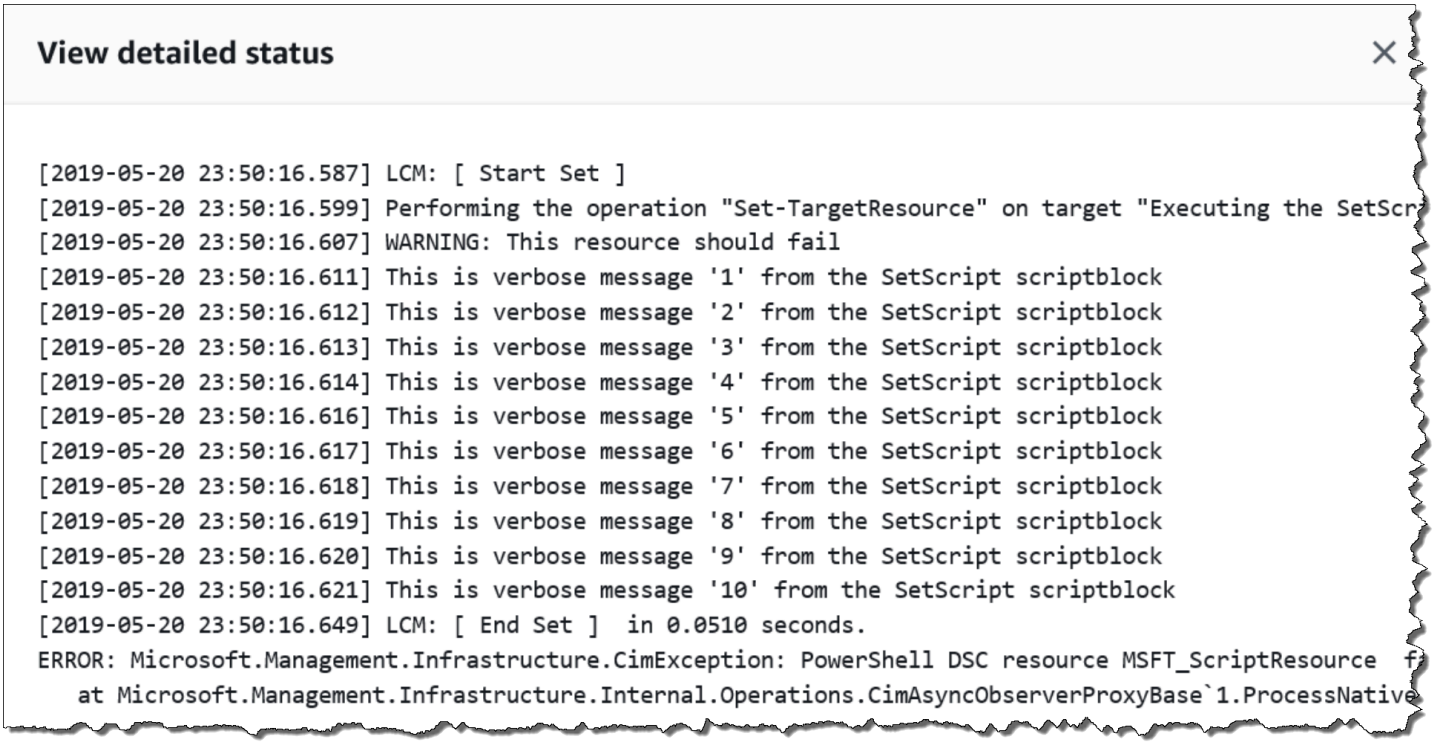
> **Note**
>
> Custom:DSC is the default **ComplianceType** value in the `AWS-ApplyDSCMofs` document. This value is customizable.



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 **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.



For information about how to view compliance information, see AWS Systems Manager Compliance.

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.

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 tool in 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/amzn-s3-demo-bucket/playbook.yml"
}
```

Example 2: Download the contents of a directory

```
{
   "path":"https://s3.amazonaws.com/amzn-s3-demo-bucket/ansible/webservers/"
}
```

> **Important**
>
> If you specify Amazon S3, then the AWS Identity and Access Management (IAM) instance profile on your managed nodes must include permissions for the S3 bucket. For more information, see Configure instance permissions required for Systems Manager.

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.

Understanding 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** supported versions: 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**.
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 Understanding targets and rate controls in State Manager associations.
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.
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 Understanding targets and rate controls in State Manager associations.

    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 Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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 Installing or updating the latest version of the AWS CLI.

2. Run one of the following commands to create an association that runs Ansible playbooks by targeting nodes using tags. Replace each *example resource placeholder* with your own information. 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"],"TimeoutSeconds":["3600"]}' \
    --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"], "TimeoutSeconds":["3600"]}' ^
    --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/amzn-s3-demo-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.

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"
```

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 tool in 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 18).
- Automatically installs the Chef client software on target nodes.
- Optionally runs Systems Manager compliance checks 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.

- Optionally applies custom JSON attributes from a source file that is stored at a location that you specify.

You can use Git, GitHub, HTTP, or Amazon S3 buckets as download 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.

Getting started

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 18) 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.

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 Working with associations in Systems Manager.

Applying JSON attributes to targets when running a recipe

You can specify JSON attributes for your Chef client to apply to target nodes during an association run. When setting up the association, you can provide raw JSON or provide the path to a JSON file stored in Amazon S3.

Use JSON attributes when you want to customize how the recipe is run without having to modify the recipe itself, for example:

- **Overriding a small number of attributes**

  Use custom JSON to avoid having to maintain multiple versions of a recipe to accommodate minor differences.

- **Providing variable values**

  Use custom JSON to specify values that may change from run-to-run. For example, if your Chef cookbooks configure a third-party application that accepts payments, you might use custom JSON to specify the payment endpoint URL.

Specifying attributes in raw JSON

The following is an example of the format you can use to specify custom JSON attributes for your Chef recipe.

```
{"filepath":"/tmp/example.txt", "content":"Hello, World!"}
```

Specifying a path to a JSON file

The following is an example of the format you can use to specify the path to custom JSON attributes for your Chef recipe.

```
{"sourceType":"s3", "sourceInfo":"someS3URL1"}, {"sourceType":"s3",
 "sourceInfo":"someS3URL2"}
```

Use Git as a cookbook source

The `AWS-ApplyChefRecipes` document uses the aws:downloadContent plugin to download Chef cookbooks. To download content from Git, specify information about your Git repository in JSON format as in the following example. Replace each *`example-resource-placeholder`* with your own information.

```
{
   "repository":"GitCookbookRepository",
   "privateSSHKey":"{{ssm-secure:ssh-key-secure-string-parameter}}",
   "skipHostKeyChecking":"false",
```

```
    "getOptions":"branch:refs/head/main",
    "username":"{{ssm-secure:username-secure-string-parameter}}",
    "password":"{{ssm-secure:password-secure-string-parameter}}"
}
```

Use GitHub as a cookbook source

The `AWS-ApplyChefRecipes` document uses the aws:downloadContent plugin to download cookbooks. To download content from GitHub, specify information about your GitHub repository in JSON format as in the following example. Replace each *example-resource-placeholder* with your own information.

```
{
    "owner":"TestUser",
    "repository":"GitHubCookbookRepository",
    "path":"cookbooks/HelloWorld",
    "getOptions":"branch:refs/head/main",
    "tokenInfo":"{{ssm-secure:token-secure-string-parameter}}"
}
```

Use HTTP as a cookbook source

You can store Chef cookbooks at a custom HTTP location as either a single `.zip` or `tar.gz` file, or a directory structure. To download content from HTTP, specify the path to the file or directory in JSON format as in the following example. Replace each *example-resource-placeholder* with your own information.

```
{
    "url":"https://my.website.com/chef-cookbooks/HelloWorld.zip",
    "allowInsecureDownload":"false",
    "authMethod":"Basic",
    "username":"{{ssm-secure:username-secure-string-parameter}}",
    "password":"{{ssm-secure:password-secure-string-parameter}}"
}
```

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 in JSON format as in the following examples. Replace each *example-resource-placeholder* with your own information.

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 Configure instance permissions required for Systems Manager.

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**.
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**, select either **Git**, **GitHub**, **HTTP**, or **S3**.
7. For **Source info**, enter cookbook source information using the appropriate format for the **Source Type** that you selected in step 6. For more information, see the following topics:
   - the section called “Use Git as a cookbook source”
   - the section called “Use GitHub as a cookbook source”

- the section called “Use HTTP as a cookbook source”
- the section called “Use Amazon S3 as a cookbook source”

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. Replace each *example-resource-placeholder* with your own information.

```
recipe[cookbook-name1::recipe-name],recipe[cookbook-name2::recipe-name]
```

9. (Optional) Specify custom JSON attributes that you want the Chef client to pass to your target nodes.

   a. In **JSON attributes content**, add any attributes that you want the Chef client to pass to your target nodes.

   b. In **JSON attributes sources**, add the paths to any attributes that you want the Chef client to pass to your target nodes.

   For more information, see the section called “Applying JSON attributes to targets when running a recipe”.

10. For **Chef client version**, specify a Chef version. Valid values are 11 through 18, or None. If you specify a number between 11 18 (inclusive), 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.

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 Learn details about Compliance 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 Understanding targets and rate controls in State Manager associations.

    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 Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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 Installing or updating the latest version of the AWS CLI.

2. Run one of the following commands to create an association that runs Chef cookbooks on target nodes that have the specified tags. Use the command that is appropriate for your cookbook source type and operating system. Replace each *example-resource-placeholder* with your own information.

   a. **Git source**

   Linux & macOS

```
aws ssm create-association --name "AWS-ApplyChefRecipes" \
    --targets Key=tag:TagKey,Values=TagValue \
    --parameters '{"SourceType":["Git"],"SourceInfo":["{\"repository\":
\"repository-name\", \"getOptions\": \"branch:branch-name\", \"username
\": \"{{ ssm-secure:username-secure-string-parameter }}\", \"password\":
 \"{{ ssm-secure:password-secure-string-parameter }}\"}"], "RunList":
["{\"recipe[cookbook-name-1::recipe-name]\", \"recipe[cookbook-
name-2::recipe-name]\"}"], "JsonAttributesContent": ["{custom-json-
content}"], "JsonAttributesSources": "{\"sourceType\":\"s3\", \"sourceInfo
\":\"s3-bucket-endpoint-1\"}, {\"sourceType\":\"s3\", \"sourceInfo\":
\"s3-bucket-endpoint-2\"}", "ChefClientVersion": ["version-number"],
 "ChefClientArguments":["{chef-client-arguments}"], "WhyRun": boolean,
 "ComplianceSeverity": ["severity-value"], "ComplianceType":
 ["Custom:Chef"], "ComplianceReportBucket": ["s3-bucket-name"]}' \
    --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":["Git"],"SourceInfo":["{\"repository\":
\"repository-name\", \"getOptions\": \"branch:branch-name\", \"username
```

```
\": \"{{ ssm-secure:username-secure-string-parameter }}\", \"password\":
 \"{{ ssm-secure:password-secure-string-parameter }}\"}"], "RunList":
["{\"recipe[cookbook-name-1::recipe-name]\", \"recipe[cookbook-
name-2::recipe-name]\"}"], "JsonAttributesContent": ["{custom-json}"],
 "JsonAttributesSources": "{\"sourceType\":\"s3\", \"sourceInfo\":
\"s3-bucket-endpoint-1\"}, {\"sourceType\":\"s3\", \"sourceInfo\":
\"s3-bucket-endpoint-2\"}", "ChefClientVersion": ["version-number"],
 "ChefClientArguments":["{chef-client-arguments}"], "WhyRun": boolean,
 "ComplianceSeverity": ["severity-value"], "ComplianceType":
 ["Custom:Chef"], "ComplianceReportBucket": ["s3-bucket-name"]}' ^
    --association-name "name" ^
    --schedule-expression "cron-or-rate-expression"
```

b. **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-name-1::recipe-name]\", \"recipe[cookbook-
name-2::recipe-name]\"}"], "JsonAttributesContent": ["{custom-json}"],
 "ChefClientVersion": ["version-number"], "ChefClientArguments":["{chef-
client-arguments}"], "WhyRun": boolean, "ComplianceSeverity": ["severity-
value"], "ComplianceType": ["Custom:Chef"], "ComplianceReportBucket": ["s3-
bucket-name"]}' \
    --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-name-1::recipe-name]\", \"recipe[cookbook-
name-2::recipe-name]\"}"], "JsonAttributesContent": ["{custom-json}"],
 "ChefClientVersion": ["version-number"], "ChefClientArguments":["{chef-
client-arguments}"], "WhyRun": boolean, "ComplianceSeverity": ["severity-
```

```
value"], "ComplianceType": ["Custom:Chef"], "ComplianceReportBucket": ["s3-
bucket-name"]}' ^
    --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"],
 "ChefClientArguments":["{--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"],
 "ChefClientArguments":["{--fips}"], "WhyRun": false, "ComplianceSeverity":
 ["Medium"], "ComplianceType": ["Custom:Chef"], "ComplianceReportBucket":
 ["ChefComplianceResultsBucket"]}' ^
    --association-name "MyChefAssociation" ^
    --schedule-expression "cron(0 2 ? * SUN *)"
```

c. **HTTP source**

Linux & macOS

```
aws ssm create-association --name "AWS-ApplyChefRecipes" \
    --targets Key=tag:TagKey,Values=TagValue \
    --parameters '{"SourceType":["HTTP"],"SourceInfo":["{\"url\":\"url-
to-zip-file|directory|cookbook\", \"authMethod\": \"auth-method\",
 \"username\": \"{{ ssm-secure:username-secure-string-parameter }}\",
 \"password\": \"{{ ssm-secure:password-secure-string-parameter }}\"}"],
 "RunList":["{\"recipe[cookbook-name-1::recipe-name]\", \"recipe[cookbook-
name-2::recipe-name]\"}"], "JsonAttributesContent": ["{custom-json-
content}"], "JsonAttributesSources": "{\"sourceType\":\"s3\", \"sourceInfo
\":\"s3-bucket-endpoint-1\"}, {\"sourceType\":\"s3\", \"sourceInfo\":
\"s3-bucket-endpoint-2\"}", "ChefClientVersion": ["version-number"],
 "ChefClientArguments":["{chef-client-arguments}"], "WhyRun": boolean,
 "ComplianceSeverity": ["severity-value"], "ComplianceType":
 ["Custom:Chef"], "ComplianceReportBucket": ["s3-bucket-name"]}' \
    --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":["HTTP"],"SourceInfo":["{\"url\":\"url-
to-zip-file|directory|cookbook\", \"authMethod\": \"auth-method\",
 \"username\": \"{{ ssm-secure:username-secure-string-parameter }}\",
 \"password\": \"{{ ssm-secure:password-secure-string-parameter }}\"}"],
 "RunList":["{\"recipe[cookbook-name-1::recipe-name]\", \"recipe[cookbook-
name-2::recipe-name]\"}"], "JsonAttributesContent": ["{custom-json-
content}"], "JsonAttributesSources": "{\"sourceType\":\"s3\", \"sourceInfo
\":\"s3-bucket-endpoint-1\"}, {\"sourceType\":\"s3\", \"sourceInfo\":
\"s3-bucket-endpoint-2\"}", "ChefClientVersion": ["version-number"],
 "ChefClientArguments":["{chef-client-arguments}"], "WhyRun": boolean,
 "ComplianceSeverity": ["severity-value"], "ComplianceType":
 ["Custom:Chef"], "ComplianceReportBucket": ["s3-bucket-name"]}' \
    --association-name "name" ^
    --schedule-expression "cron-or-rate-expression"
```

d. **Amazon 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": ["amzn-s3-demo-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"], "ComplianceType":
 ["Custom:Chef"], "ComplianceReportBucket": ["amzn-s3-demo-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/amzn-s3-demo-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/amzn-s3-demo-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 *)"
```

The system creates the association, and unless your specified cron or rate expression prevents it, the system runs the association on the target nodes.

> **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.

3. Run the following command to view the status of the association you just created.

```
aws ssm describe-association --association-id "ID"
```

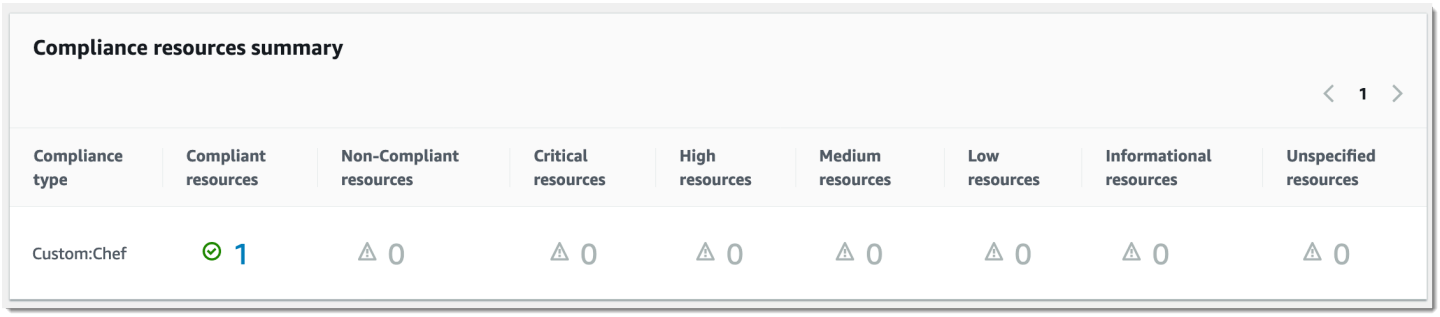
Viewing Chef resource compliance details

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.

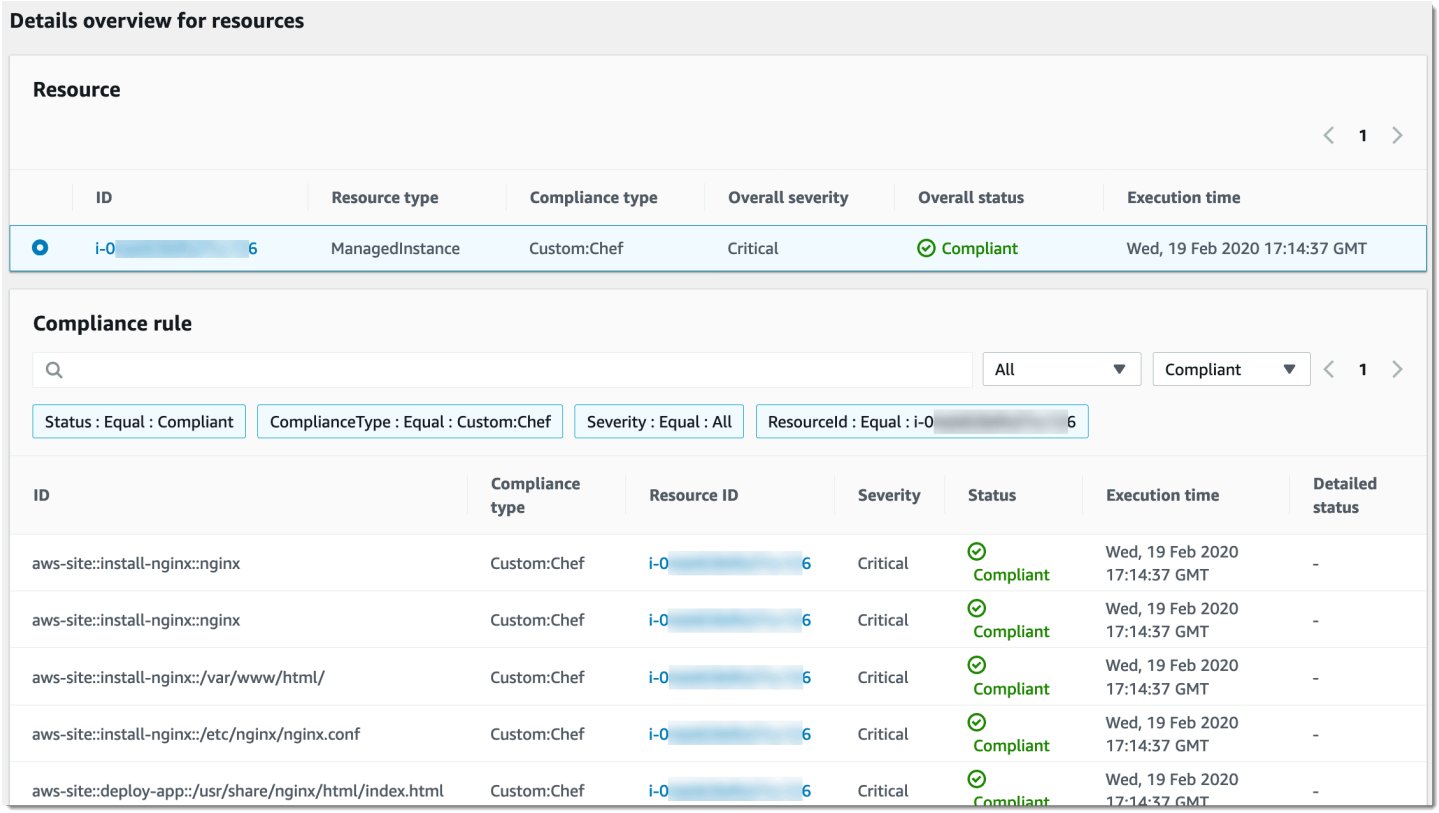
A Systems Manager Compliance scan collects information about resources on your managed nodes that were created or checked in the most recent Chef run. The resources can include files, directories, `systemd` services, `yum` packages, templated files, `gem` packages, and dependent cookbooks, among others.

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.



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.



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.

> **Important**
>
> 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 with the AWS 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. To customize the update schedule for SSM Agent using the console, see Automatically updating SSM Agent.

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 Setting up managed nodes for AWS Systems Manager.

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 Installing or updating the latest version of the AWS CLI.

2. Run the following command to create an association by targeting instances using Amazon Elastic Compute Cloud (Amazon EC2) tags. Replace each *example resource placeholder* with your own information. 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.

   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

Amazon Windows Server 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 tool in 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 Setting up managed nodes for AWS Systems Manager.

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**.
3. Choose **Create association**.
4. In the **Name** field, enter a descriptive name for the association.
5. In the **Document** list, choose `AWS-ConfigureAWSPackage`.
6. In the **Parameters** area, do the following:

   - For **Action**, choose **Install**.
   - For **Installation type**, choose **Uninstall and reinstall**.

     > **Note**
     >
     > In-place upgrades are not supported for this package. It must be uninstalled and reinstalled.

   - For **Name**, enter **`AWSPVDriver`**.

You don't need to enter anything for **Version** and **Additional Arguments**.

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.

> **Tip**
>
> If a managed node you expect to see isn't listed, see Troubleshooting managed node availability for troubleshooting tips.

> **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 Server instance but fails on the Linux instances. The overall status of the association shows **Failed**.

8. In the **Specify schedule** area, choose whether to run the association on a schedule that you configure, or just once. Updated PV drivers are released a several times a year, so you can schedule the association to run once a month, if you want.
9. In the **Advanced options** area, 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.
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 **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 Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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. (Optional) In the **CloudWatch alarm** section, for **Alarm name**, choose a CloudWatch alarm to apply to your association for monitoring.

> **Note**
>
> Note the following information about this step.
>
> - The alarms list displays a maximum of 100 alarms. If you don't see your alarm in the list, use the AWS Command Line Interface to create the association. For more information, see Create an association (command line).
> - To attach a CloudWatch alarm to your command, the IAM principal that creates the association must have permission for the `iam:createServiceLinkedRole` action. For more information about CloudWatch alarms, see Using Amazon CloudWatch alarms.
> - If your alarm activates, any pending command invocations or automations do not run.

13. 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 Setting up Systems Manager unified console for an organization.

AWS Systems Manager change management tools

AWS Systems Manager provides the following tools for making changes to your AWS resources.

Topics

- AWS Systems Manager Automation
- AWS Systems Manager Change Calendar
- AWS Systems Manager Change Manager
- AWS Systems Manager Documents
- AWS Systems Manager Maintenance Windows
- AWS Systems Manager Quick Setup

AWS Systems Manager Automation

Automation, a tool in 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.

> **Note**
>
> For customers who are new to Systems Manager Automation as of August 14, 2025, the Automation free tier is not available. For customers already using Automation, the free tier of service ends on December 31, 2025. For information about current service costs, see AWS Systems Manager pricing.

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.

- **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.

- **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 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 Run automated operations at scale.

- **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 Tutorials.

- **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.

- **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 and Reference: Amazon EventBridge event patterns and types for Systems Manager.

- **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	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

Concept	Details
	steps and parameters that you specify. Steps run in sequential order. For more information, see Creating your own runbooks. 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.
Automation action	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 `mainSteps` section of your runbook. Automation supports 20 distinct action types. For more information, see the Systems Manager Automation actions reference.

Concept	Details
Automation quota	Each AWS account can run 100 automations simultaneously. This includes child automatio ns (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. This quota can be adjusted using adaptive concurrency. For more informati on, see Allowing Automation to adapt to your concurrency needs.For more informati on about running automations, see Run an automated operation powered by Systems Manager Automation.
Automation queue quota	If you attempt to run more automations than the concurrent automation limit, subsequen t automations are added to a queue. Each AWS account can queue 5,000 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 Run automated operations at scale.

Concept	Details
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 automatio n is complete (or reaches a terminal state), the first automation in the queue is started.

Topics

- Setting up Automation
- Run an automated operation powered by Systems Manager Automation
- Rerunning automation executions
- Run an automation that requires approvals
- Run automated operations at scale
- Running automations in multiple AWS Regions and accounts
- Run automations based on EventBridge events
- Run an automation step by step
- Scheduling automations with State Manager associations
- Schedule automations with maintenance windows
- Systems Manager Automation actions reference
- Creating your own runbooks
- Systems Manager Automation Runbook Reference
- Tutorials
- Learn about statuses returned by Systems Manager Automation
- Troubleshooting Systems Manager Automation

Setting up Automation

To set up Automation, a tool in 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.

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 user, 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 user, 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 user, 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.

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

Topics

- Create service roles for Automation by using AWS CloudFormation
- Create the service roles for Automation using the console
- Allowing Automation to adapt to your concurrency needs
- Configuring automatic retry for throttled operations
- Implement change controls for Automation

Create service roles for Automation by using AWS CloudFormation

You can create a service role for Automation, a tool in 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`.

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 or customer managed policy to invoke other AWS services.

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.

Create the service roles for Automation using the console

If you need to create a service role for Automation, a tool in AWS Systems Manager, complete the following tasks. For more information about when a service role is required for Automation, see Setting up Automation.

Tasks

- Task 1: Create a service role for Automation
- Task 2: Attach the iam:PassRole policy to your Automation role

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.

JSON

```
{
  "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 or customer managed 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 or customer managed 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. In the **Add permissions** dropdown, choose **Attach policies** or **Create inline policy**.
6. If you choose **Attach policies**, select the check box next to the policy you want to add and choose **Add permissions**.
7. If you choose **Create inline policy**, choose the **JSON** tab.
8. Enter a JSON Policy document for the AWS services you want to invoke. Here are two example JSON Policy documents.

Amazon S3 PutObject and GetObject Example

JSON

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

Amazon EC2 CreateSnapshot and DescribeSnapShots Example

JSON

```
{
   "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*.

9. When you're finished, choose **Review policy**. The Policy Validator reports any syntax errors.
10. 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.
11. 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 tools 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**.

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 using the AWS Management 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**.

To turn on adaptive concurrency using the command line

- Open the AWS CLI or Tools for Windows PowerShell and run the following command to turn on adaptive concurrency for your account in the requesting Region.

  Linux & macOS

  ```
  aws ssm update-service-setting \
      --setting-id /ssm/automation/enable-adaptive-concurrency \
      --setting-value True
  ```

  Windows

  ```
  aws ssm update-service-setting ^
      --setting-id /ssm/automation/enable-adaptive-concurrency ^
      --setting-value True
  ```

  PowerShell

  ```
  Update-SSMServiceSetting `
      -SettingId "/ssm/automation/enable-adaptive-concurrency" `
      -SettingValue "True"
  ```

Configuring automatic retry for throttled operations

There is a limit on the number of concurrent automation executions that can run in each account. Attempting to run several automations concurrently in an account can lead to throttling issues. You can use the automatic throttling retry capability to configure retry behavior for throttled automation steps.

Automatic throttling retry for automation actions provides a more resilient execution environment for high-scale operations. The throttling retry capability supports all automation actions except for `aws:executeScript`.

The throttling retry setting works in addition to the existing `maxAttempts` step property. When both are configured, the system first attempts throttling retries within the specified time limit, then applies the `maxAttempts` setting if the step continues to fail.

To configure throttling retry using the AWS Management 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. In the **Throttling retry time limit** field, enter a value between 0 and 3600 seconds. This specifies the maximum time that the system retries a step that is throttled.
5. Choose **Save**.

To configure throttling retry using the command line

- Open the AWS CLI or Tools for Windows PowerShell and run the following command to configure throttling retry for your account in the requesting Region.

  Linux & macOS

  ```
  aws ssm update-service-setting \
      --setting-id /ssm/automation/throttling-retry-time-limit \
      --setting-value 3600
  ```

  Windows

  ```
  aws ssm update-service-setting ^
      --setting-id /ssm/automation/throttling-retry-time-limit ^
      --setting-value 3600
  ```

  PowerShell

  ```
  Update-SSMServiceSetting `
      -SettingId "/ssm/automation/throttling-retry-time-limit" `
      -SettingValue "3600"
  ```

Implement change controls for Automation

By default, Automation allows you to use runbooks without date and time constraints. By integrating Automation with Change Calendar, you can implement change controls to all

automations in your AWS account. With this setting, AWS Identity and Access Management (IAM) principals in your account can only run automations during the time periods allowed by your change calendar. To learn more about working with Change Calendar, see Working with Change Calendar.

To turn on change controls (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 **Turn on Change Calendar integration**.
5. In the **Choose a change calendar** dropdown list, choose the change calendar that you want Automation to follow.
6. Choose **Save**.

Run an automated operation powered by Systems Manager Automation

When you run an automation, by default, the automation runs in the context of the user who initiated the automation. This means, for example, if your user 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 that is known in this case as an *assume* role that is configured with the AmazonSSMAutomationRole managed policy. You might need to add additional IAM policies to your assume role to use various runbooks. 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 tool in 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.

Before you begin

Before you complete the following procedures, you must create the IAM service role and configure a trust relationship for Automation, a tool in AWS Systems Manager. For more information, see Task 1: Create a service role for Automation.

The following procedures describe how to use the Systems Manager console or your preferred command line tool to run a simple automation.

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. (Optional) Choose a CloudWatch alarm to apply to your automation for monitoring. To attach a CloudWatch alarm to your automation, the IAM principal that starts the automation must have permission for the `iam:createServiceLinkedRole` action. For more information about CloudWatch alarms, see Using Amazon CloudWatch alarms. Note that if your alarm activates, the automation is stopped. If you use AWS CloudTrail, you will see the API call in your trail.
9. Choose **Execute**.

The console displays the status of the automation. If the automation fails to run, see Troubleshooting Systems Manager Automation.

After an automation execution completes, you can rerun the execution with the same or modified parameters. For more information, see Rerunning automation executions.

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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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-SSMAutomationExecution `
     -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-02573cafcfEXAMPLE"
   ```

   Windows

   ```
   aws ssm start-automation-execution ^
   ```

```
    --document-name "AWS-RestartEC2Instance" ^
    --parameters "InstanceId=i-02573cafcfEXAMPLE"
```

PowerShell

```
Start-SSMAutomationExecution `
  -DocumentName AWS-RestartEC2Instance `
  -Parameter @{"InstanceId"="i-02573cafcfEXAMPLE"}
```

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-SSMAutomationExecutionList | `
  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                     : {}
```

Rerunning automation executions

You can rerun AWS Systems Manager automation executions to repeat tasks with either identical or modified parameters. The rerun capability allows you to efficiently replicate automation executions without manually recreating automation configurations, reducing operational overhead and potential configuration errors.

When you rerun an automation execution, Systems Manager preserves the original runbook parameters, Amazon CloudWatch alarms, and tags from the previous execution. The system creates a new execution with a new execution ID and updated timestamps. You can rerun any type of automation execution, including simple executions, rate control executions, cross-account and cross-region executions, and manual executions.

Rerun an automation execution (console)

The following procedures describe how to use the Systems Manager console to rerun an automation execution.

To rerun an automation execution from the Automation home page

Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

1. In the navigation pane, choose **Automation**.
2. In the executions list, select the execution that you want to rerun.
3. Choose **Rerun execution**.
4. On the **Execute automation document** page, review the pre-populated parameters, execution mode, and target configuration from the original execution.
5. (Optional) Modify any parameters, targets, or other settings as needed for your rerun.
6. Choose **Execute** to start the rerun with a new execution ID.

To rerun an automation execution from the execution details page

Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

1. In the navigation pane, choose **Automation**.
2. Choose the execution ID of the automation that you want to rerun.
3. On the execution details page, choose **Rerun execution**.
4. On the **Execute automation document** page, review the pre-populated parameters, execution mode, and target configuration from the original execution.
5. (Optional) Modify any parameters, targets, or other settings as needed for your rerun.
6. Choose **Execute** to start the rerun with a new execution ID.

To copy an automation execution to a new execution

Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

1. In the navigation pane, choose **Automation**.
2. Choose the execution ID of the automation that you want to copy.
3. On the execution details page, choose **Actions**, and then choose **Copy to new**.
4. On the **Execute automation document** page, review the pre-populated parameters, execution mode, and target configuration from the original execution.
5. (Optional) Modify any parameters, targets, or other settings as needed for your new execution.
6. Choose **Execute** to start the new execution.

Run an automation that requires approvals

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 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.

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 a user name 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 user names 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. If you're targeting more than 100 accounts and Regions, you must specify the `AWS-SystemsManager-AutomationAdministrationRole`.
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.

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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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 user names 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-02573cafcfEXAMPLE,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-02573cafcfEXAMPLE,Approvers=arn:aws:iam::123456789012:role/
Administrator,SNSTopicArn=arn:aws:sns:region:123456789012:AutomationApproval"
```

PowerShell

```
Start-SSMAutomationExecution `
    -DocumentName AWS-StartEC2InstanceWithApproval `
    -Parameters @{
        "InstanceId"="i-02573cafcfEXAMPLE"
        "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.

Run automated operations at scale

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 and error threshold features are collectively called *rate controls*. 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.

For more information about concurrency and error thresholds, see Control automations at scale. For more information about targets, see Mapping targets for an automation.

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. (Optional) Choose a CloudWatch alarm to apply to your automation for monitoring. To attach a CloudWatch alarm to your automation, the IAM principal that starts the automation must

have permission for the `iam:createServiceLinkedRole` action. For more information about CloudWatch alarms, see Using Amazon CloudWatch alarms. Note that if your alarm activates, the automation is stopped. If you use AWS CloudTrail, you will see the API call in your trail.

12. Choose **Execute**.

To view automations started by your rate control automation, in the navigation pane, choose Automation, and then select **Show child automations**.

After an automation execution completes, you can rerun the execution with the same or modified parameters. For more information, see Rerunning automation executions.

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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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-SSMAutomationExecution `
    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-SSMAutomationExecution `
    -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 nname \
    --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-SSMAutomationExecution `
    -DocumentName "runbook name" `
    -Targets $Targets `
    -TargetParameterName "parameter name" `
    -Parameter @{"input parameter name"="input parameter value"} `
    -MaxConcurrency "10" `
    -MaxError "25%"
```

Targeting all Amazon EC2 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-SSMAutomationExecution `
    -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-SSMAutomationExecutionList | `
    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-SSMAutomationExecution `
       -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-02573cafcfEXAMPLE"
            ],
            "Truncated": false
        },
        "ExecutionEndTime": 1564681619.915,
        "Targets": [
            {
                "Values": [
                    "DEV"
                ],
                "Key": "tag:ENV"
            }
        ],
        "DocumentVersion": "1",
        "ExecutionStartTime": 1564681576.09,
        "ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/
Admin",
        "StepExecutions": [
            {
                "Inputs": {
                    "InstanceId": "i-02573cafcfEXAMPLE"
                },
                "Outputs": {},
                "StepName": "i-02573cafcfEXAMPLE",
                "ExecutionEndTime": 1564681619.093,
                "StepExecutionId": "86c7b811-3896-4b78-b897-01234EXAMPLE",
                "ExecutionStartTime": 1564681576.836,
                "Action": "aws:executeAutomation",
                "StepStatus": "Success"
            }
        ],
        "TargetParameterName": "InstanceId",
        "Mode": "Auto"
    }
}
```

Windows

```
{
    "AutomationExecution": {
        "StepExecutionsTruncated": false,
        "AutomationExecutionStatus": "Success",
```

```
        "MaxConcurrency": "1",
        "Parameters": {},
        "MaxErrors": "1",
        "Outputs": {},
        "DocumentName": "AWS-StopEC2Instance",
        "AutomationExecutionId": "a4a3c0e9-7efd-462a-8594-01234EXAMPLE",
        "ResolvedTargets": {
            "ParameterValues": [
                "i-02573cafcfEXAMPLE"
            ],
            "Truncated": false
        },
        "ExecutionEndTime": 1564681619.915,
        "Targets": [
            {
                "Values": [
                    "DEV"
                ],
                "Key": "tag:ENV"
            }
        ],
        "DocumentVersion": "1",
        "ExecutionStartTime": 1564681576.09,
        "ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/
Admin",
        "StepExecutions": [
            {
                "Inputs": {
                    "InstanceId": "i-02573cafcfEXAMPLE"
                },
                "Outputs": {},
                "StepName": "i-02573cafcfEXAMPLE",
                "ExecutionEndTime": 1564681619.093,
                "StepExecutionId": "86c7b811-3896-4b78-b897-01234EXAMPLE",
                "ExecutionStartTime": 1564681576.836,
                "Action": "aws:executeAutomation",
                "StepStatus": "Success"
            }
        ],
        "TargetParameterName": "InstanceId",
        "Mode": "Auto"
    }
}
```

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.

Mapping targets for an automation

Use the `Targets` parameter to quickly define which resources are targeted by 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.

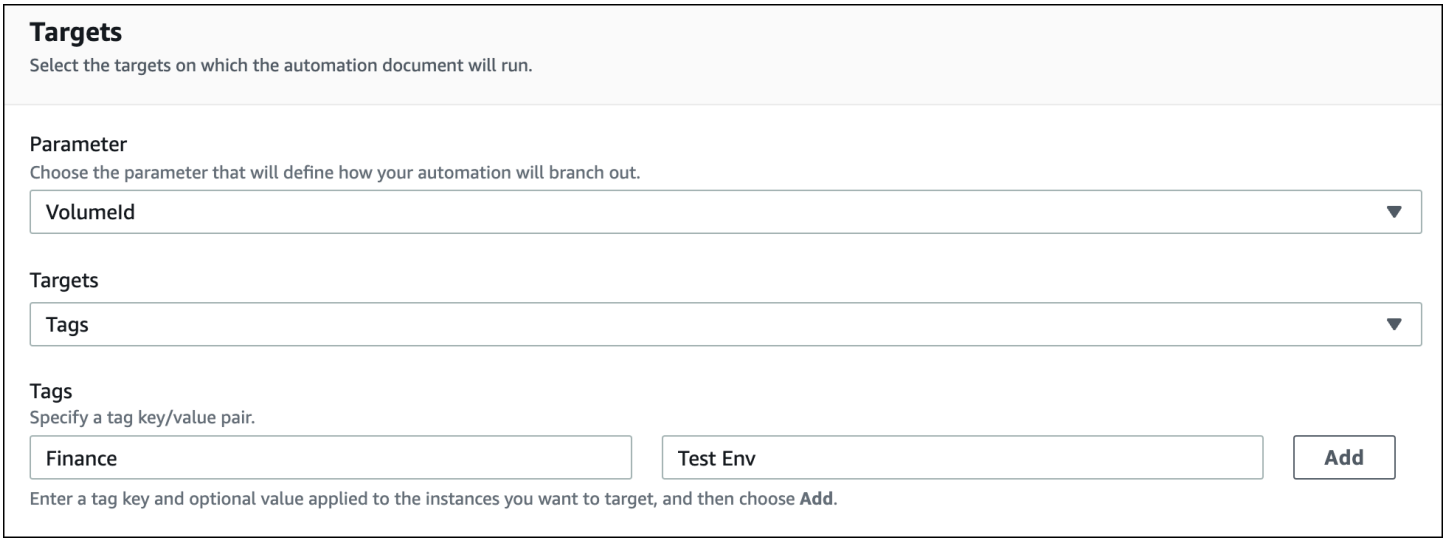
Targeting a tag

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. When creating a custom runbook you must specify the **Target type** as `/AWS::EC2::Instance` to ensure only instances are used. Otherwise, all resources with the same tag will be targeted. 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 `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.



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-02573cafcfEXAMPLE,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://amzn-s3-demo-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"]
  },
```

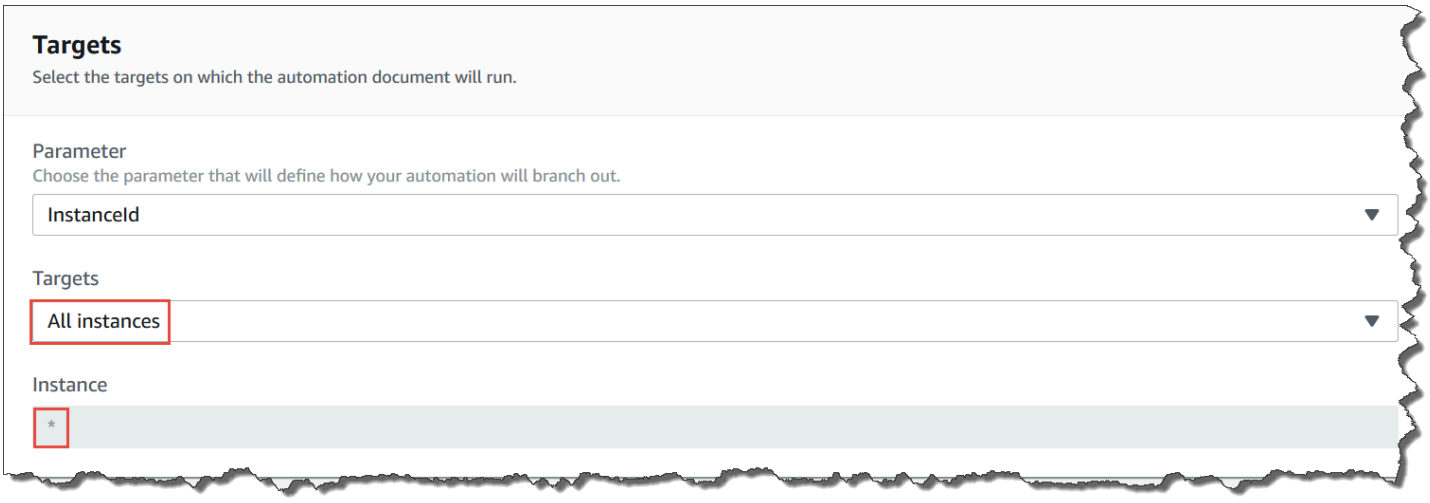
```
  {
    "instanceType" : ["t2.medium"],
    "imageId" : ["ami-cfe4b2b0"]
  },
  {
    "instanceType" : ["t2.medium"],
    "imageId" : ["ami-cfe4b2b0"]
  },
  {
    "instanceType" : ["t2.medium"],
    "imageId" : ["ami-cfe4b2b0"]
  }
]
```

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 Amazon EC2 instances

You can run an automation on all Amazon EC2 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 Amazon EC2 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 **InstanceId** field in the **Input parameters** field unavailable for changes. Making these fields unavailable for changes is expected behavior if you choose to target all instances.

Control automations at scale

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.

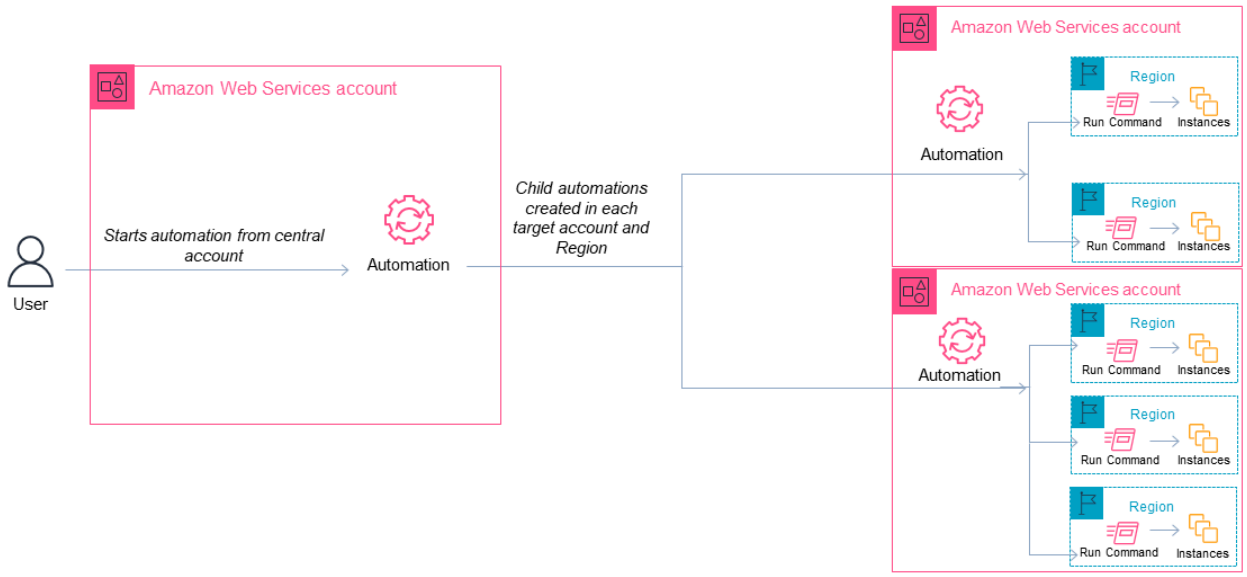
Running automations in multiple AWS Regions and accounts

You can run AWS Systems Manager automations across multiple AWS Regions and AWS accounts or AWS Organizations organizational units (OUs) from a central account. Automation is a tool in 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 do the following by using automation runbooks:

- Implement patching and security updates centrally.
- Remediate compliance drift on VPC configurations or Amazon S3 bucket policies.
- Manage resources, such as Amazon Elastic Compute Cloud (Amazon EC2) EC2 instances, at scale.

The following diagram shows an example of a user who is running the `AWS-RestartEC2Instances` runbook in multiple Regions and accounts from a central account. The automation locates the instances by using the specified tags in the targeted Regions and accounts.



Choose a central account for Automation

If you want to run automations across OUs, the central account must have permissions to list all of the accounts in the OUs. This is only possible from a delegated administrator account, or the management account of the organization. We recommend that you follow AWS Organizations best practices and use a delegated administrator account. For more information about AWS Organizations best practices, see Best practices for the management account in the *AWS Organizations User Guide*. To create a delegated administrator account for Systems Manager, you can use the `register-delegated-administrator` command with the AWS CLI as shown in the following example.

```
aws organizations register-delegated-administrator \
    --account-id delegated admin account ID \
    --service-principal ssm.amazonaws.com
```

If you want to run automations across multiple accounts that are not managed by AWS Organizations, we recommend creating a dedicated account for automation management. Running all cross-account automations from a dedicated account simplifies IAM permissions management, troubleshooting efforts, and creates a layer of separation between operations and administration. This approach is also recommended if you use AWS Organizations, but only want to target individual accounts and not OUs.

How running automations works

Running automations across multiple Regions and accounts or OUs works as follows:

1. Sign in to the account that you want to configure as the Automation central account.
2. Use the Setting up management account permissions for multi-Region and multi-account automation procedure in this topic to create the following IAM roles:
   - **`AWS-SystemsManager-AutomationAdministrationRole`** - This role gives the user permission to run automations in multiple accounts and OUs.
   - **`AWS-SystemsManager-AutomationExecutionRole`** - This role gives the user permission to run automations in the targeted accounts.
3. Choose the runbook, Regions, and accounts or OUs where you want to run the automation.

> **Note**
>
> Be sure that 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. For information about using shared runbooks, see Using shared SSM documents.

4. Run the automation.

> **Note**
>
> 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 contains `aws:executeAutomation` steps for each of the target accounts.

5. 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-AutomationAdministrationRole`** role. You only need to create this role in the Automation central account. This procedure also 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.

To create the required IAM administration role for multi-Region and multi-account automations by using AWS CloudFormation

1. Download and unzip the `AWS-SystemsManager-AutomationAdministrationRole.zip`. Or, if your accounts are managed by AWS Organizations `AWS-SystemsManager-AutomationAdministrationRole (org).zip`. This file contains the `AWS-`

SystemsManager-AutomationAdministrationRole.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**.
5. Choose **Choose file**, and then choose the AWS-SystemsManager-AutomationAdministrationRole.yaml AWS CloudFormation template file.
6. Choose **Next**.
7. On the **Specify stack details** page, in the **Stack name** field, enter a name.
8. Choose **Next**.
9. On the **Configure stack options** page, enter values for any options you want to use. Choose **Next**.
10. On the **Review** page, scroll down and choose the **I acknowledge that AWS CloudFormation might create IAM resources with custom names** option.
11. Choose **Create stack**.

AWS CloudFormation shows the **CREATE_IN_PROGRESS** status for approximately three minutes. The status changes to **CREATE_COMPLETE**.

You must repeat the following procedure in *every* account that you want to target to run multi-Region and multi-account automations.

To create the required IAM automation role for multi-Region and multi-account automations by using AWS CloudFormation

1. Download the AWS-SystemsManager-AutomationExecutionRole.zip. Or, if your accounts are managed by AWS Organizations AWS-SystemsManager-AutomationExecutionRole (org).zip. This file contains the AWS-SystemsManager-AutomationExecutionRole.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**.
5. Choose **Choose file**, and then choose the AWS-SystemsManager-AutomationExecutionRole.yaml 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 **AdminAccountId** field, enter the ID for the Automation central account.
9. If you are setting up this role for an AWS Organizations environment, there is another field in the section called **OrganizationID**. Enter the ID of your AWS organization.
10. Choose **Next**.
11. On the **Configure stack options** page, enter values for any options you want to use. Choose **Next**.
12. On the **Review** page, scroll down and choose the **I acknowledge that AWS CloudFormation might create IAM resources with custom names** option.
13. Choose **Create stack**.

AWS CloudFormation shows the **CREATE_IN_PROGRESS** status for approximately three minutes. The status changes to **CREATE_COMPLETE**.

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:

- The 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.
- 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, organizational units (OUs), and roots** 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.

   a. (Optional) Select the **Include child OUs** checkbox to include all child organizational units within the specified OUs.

   b. (Optional) In the **Exclude accounts and organizational units (OUs)** field, enter a comma-separated list of account IDs and OU IDs that you want to exclude from the expanded entities entered above.

8. Use the **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. Choose the `AWS-SystemsManager-AutomationAdministrationRole` 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. (Optional) Choose a CloudWatch alarm to apply to your automation for monitoring. To attach a CloudWatch alarm to your automation, the IAM principal that starts the automation must have permission for the `iam:createServiceLinkedRole` action. For more information about CloudWatch alarms, see Using Amazon CloudWatch alarms. Note that if your alarm activates, the automation is cancelled and any `0nCancel` steps you have defined run. If you use AWS CloudTrail, you will see the API call in your trail.

13. 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.

14. 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.

15. Choose **Execute**.

After an automation execution completes, you can rerun the execution with the same or modified parameters. For more information, see Rerunning automation executions.

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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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-SSMAutomationExecution `
        -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" }
```

Examples: Running an automation in multiple Regions and accounts

The following are examples demonstrating how to use the AWS CLI and PowerShell to run automations in multiple accounts and Regions with a single command.

Example 1: This example restarts EC2 instances in three Regions across an entire AWS Organizations organization. This is achieved by targeting the root ID of the organization, and including child OUs.

Linux & macOS

```
aws ssm start-automation-execution \
        --document-name "AWS-RestartEC2Instance" \
        --target-parameter-name InstanceId \
        --targets '[{"Key":"AWS::EC2::Instance","Values":["*"]}]' \
        --target-locations '[{
            "Accounts": ["r-example"],
            "IncludeChildOrganizationUnits": true,
            "Regions": ["us-east-1", "us-east-2", "us-west-2"]
        }]'
```

Windows

```
aws ssm start-automation-execution \
        --document-name "AWS-RestartEC2Instance" ^
        --target-parameter-name InstanceId ^
        --targets '[{"Key":"AWS::EC2::Instance","Values":["*"]}]' ^
        --target-locations '[{
            "Accounts": ["r-example"],
            "IncludeChildOrganizationUnits": true,
            "Regions": ["us-east-1", "us-east-2", "us-west-2"]
        }]'
```

PowerShell

```
Start-SSMAutomationExecution `
        -DocumentName "AWS-RestartEC2Instance" `
        -TargetParameterName "InstanceId" `
        -Targets '[{"Key":"AWS::EC2::Instance","Values":["*"]}]'
        -TargetLocation @{
            "Accounts"="r-example";
            "Regions"="us-east-1", "us-east-2", "us-west-2";
            "IncludeChildOrganizationUnits"=true}
```

Example 2: This example restarts specific EC2 instances in different accounts and Regions.

> **Note**
>
> The `TargetLocationMaxConcurrency` option is available using the AWS CLI and AWS SDKs.

Linux & macOS

```
aws ssm start-automation-execution \
        --document-name "AWS-RestartEC2Instance" \
        --target-parameter-name InstanceId \
        --target-locations '[{
            "Accounts": ["123456789012"],
            "Targets": [{
                "Key":"ParameterValues",
                "Values":["i-02573cafcfEXAMPLE", "i-0471e04240EXAMPLE"]
            }],
            "TargetLocationMaxConcurrency": "100%",
            "Regions": ["us-east-1"]
        }, {
            "Accounts": ["987654321098"],
            "Targets": [{
                "Key":"ParameterValues",
                "Values":["i-07782c72faEXAMPLE"]
            }],
            "TargetLocationMaxConcurrency": "100%",
            "Regions": ["us-east-2"]
        }]'
```

Windows

```
aws ssm start-automation-execution ^
        --document-name "AWS-RestartEC2Instance" ^
        --target-parameter-name InstanceId ^
        --target-locations '[{
            "Accounts": ["123456789012"],
            "Targets": [{
                "Key":"ParameterValues",
                "Values":["i-02573cafcfEXAMPLE", "i-0471e04240EXAMPLE"]
            }],
            "TargetLocationMaxConcurrency": "100%",
```

```
            "Regions": ["us-east-1"]
        }, {
            "Accounts": ["987654321098"],
            "Targets": [{
                "Key":"ParameterValues",
                "Values":["i-07782c72faEXAMPLE"]
            }],
            "TargetLocationMaxConcurrency": "100%",
            "Regions": ["us-east-2"]
        }]'
```

PowerShell

```
Start-SSMAutomationExecution `
        -DocumentName "AWS-RestartEC2Instance" `
        -TargetParameterName "InstanceId" `
        -Targets '[{"Key":"AWS::EC2::Instance","Values":["*"]}]'
        -TargetLocation @({
            "Accounts"="123456789012",
            "Targets"= @{
                "Key":"ParameterValues",
                "Values":["i-02573cafcfEXAMPLE", "i-0471e04240EXAMPLE"]
            },
            "TargetLocationMaxConcurrency"="100%",
            "Regions"=["us-east-1"]
        }, {
            "Accounts"="987654321098",
            "Targets": @{
                "Key":"ParameterValues",
                "Values":["i-07782c72faEXAMPLE"]
            },
            "TargetLocationMaxConcurrency": "100%",
            "Regions"=["us-east-2"]
        })
```

Example 3: This example demonstrates specifying multiple AWS accounts and Regions where the automation should run using the `--target-locations-url` option. The value for this option must be a JSON file in a publicly accessible presigned Amazon S3 URL.

> **Note**
>
> `--target-locations-url` is available when using the AWS CLI and AWS SDKs.

Linux & macOS

```
aws ssm start-automation-execution \
    --document-name "MyCustomAutomationRunbook" \
    --target-locations-url "https://amzn-s3-demo-bucket.s3.amazonaws.com/target-
locations.json"
```

Windows

```
aws ssm start-automation-execution ^
    --document-name "MyCustomAutomationRunbook" ^
    --target-locations-url "https://amzn-s3-demo-bucket.s3.amazonaws.com/target-
locations.json"
```

PowerShell

```
Start-SSMAutomationExecution `
    -DocumentName "MyCustomAutomationRunbook" `
    -TargetLocationsUrl "https://amzn-s3-demo-bucket.s3.amazonaws.com/target-
locations.json"
```

Sample content for the JSON file:

```
[
{
        "Accounts": [ "123456789012", "987654321098", "456789123012" ],
        "ExcludeAccounts": [ "111222333444", "999888444666" ],
        "ExecutionRoleName": "MyAutomationExecutionRole",
        "IncludeChildOrganizationUnits": true,
        "Regions": [ "us-east-1", "us-west-2", "ap-south-1", "ap-northeast-1" ],
        "Targets": ["Key": "AWS::EC2::Instance", "Values": ["i-2"]],
        "TargetLocationMaxConcurrency": "50%",
        "TargetLocationMaxErrors": "10",
        "TargetsMaxConcurrency": "20",
```

```
        "TargetsMaxErrors": "12"
 }
]
```

Example 4: 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-SSMAutomationExecution `
        -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 5: 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-SSMAutomationExecution `
        -DocumentName "AWS-RestartEC2Instance" `
```

```
        -Parameter @{
        "AutomationAssumeRole"="arn:aws:iam::123456789012:role/AWS-
SystemsManager-AutomationAdministrationRole" } `
        -TargetParameterName "InstanceId" `
        -Target $Targets `
        -TargetLocation @{
        "Accounts"="123456789012","987654321098";
        "Regions"="eu-central-1";
        "ExecutionRoleName"="AWS-SystemsManager-AutomationExecutionRole" }
```

Example 6: 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-SSMAutomationExecution `
        -DocumentName "AWS-RestartEC2Instance" `
        -Parameter @{
        "AutomationAssumeRole"="arn:aws:iam::123456789012:role/AWS-
SystemsManager-AutomationAdministrationRole" } `
        -TargetParameterName "InstanceId" `
        -Target $Targets `
        -TargetLocation @{
        "Accounts"="ou-1a2b3c-4d5e6c";
        "Regions"="us-west-1";
        "ExecutionRoleName"="AWS-SystemsManager-AutomationExecutionRole" }
```

The system returns information similar to the following.

Linux & 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-SSMAutomationExecutionList | `
        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-SSMAutomationExecution `
        -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.

More info

Centralized multi-account and multi-Region patching with AWS Systems Manager Automation

Run automations based on EventBridge events

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 Creating your own runbooks.

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**.
3. Choose **Create rule**.
4. 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.
5. For **Event bus**, 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 **default**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
6. Choose how the rule is triggered.

To create a rule based on...	Do this...
Event	a. For **Rule type**, choose **Rule with an event pattern**. b. Choose **Next**. c. For **Event source**, choose **AWS events or EventBridge partner events**. d. In the **Event pattern** section, do one of the following: • To use a template to create your event pattern, choose **Event pattern form** and choose **Event source**, **AWS service**, and **Event type**. If you choose **All Events** as the event type, all events emitted by the AWS service will match the rule. To customize the template, choose **Custom pattern (JSON editor)** and make your changes. • To use a custom event pattern, choose **Custom pattern (JSON editor)** and create your event pattern.

To create a rule based on...	Do this...
Schedule	a. For **Rule type**, choose **Schedule**. b. Choose **Next**. c. For **Schedule pattern**, do one of the following: • To use a cron expression to define the schedule, choose **A fine-grained schedule that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month** and enter the cron expression. • To use a rate expression to define the schedule, choose **A schedule that runs at a regular rate, such as every 10 minutes** and enter the rate expression.

7. Choose **Next**.
8. For **Target types**, choose **AWS service**.
9. For **Select a target**, choose **Systems Manager Automation**.
10. For **Document**, choose a runbook to use when your target is invoked.
11. In the **Configure automation parameter(s)** section, either keep the default parameter values (if available) or enter your own values.

> **Note**
>
> To create a target, you must specify a value for each required parameter. If you don't, the system creates the rule, but the rule won't run.

12. For many target types, EventBridge needs permissions to send events to the target. In these cases, EventBridge can create the IAM role needed for your rule to run. Do one of the following:

    - 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** and select the existing role from the dropdown. Note that you might need to update the trust policy for your IAM role to include EventBridge. The following is an example:

    JSON

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

13. Choose **Next**.
14. (Optional) Enter one or more tags for the rule. For more information, see Tagging Your Amazon EventBridge Resources in the *Amazon EventBridge User Guide*.
15. Choose **Next**.

16. Review the details of the rule and choose **Create rule**.

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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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:region:account ID:automation-definition/runbook
 name","Input":"{\"Message\":[\"{\\\"Key\\\":\\\"key name\\\",\\\"Values\\\":[\\
\"value\\\"]}\"]}","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:region:account ID:automation-definition/runbook
 name","Input":"{\"Message\":[\"{\\\"Key\\\":\\\"key name\\\",\\\"Values\\\":[\\
\"value\\\"]}\"]}","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:region:account ID: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:region:*:automation-definition/AWS-
StartEC2Instance","Input":"{\"InstanceId\":[\"i-02573cafcfEXAMPLE\"],
\"AutomationAssumeRole\":[\"arn:aws:iam::123456789012:role/AutomationServiceRole
\"]}","Id": "Target1","RoleArn": "arn:aws:iam::123456789012:role/service-role/
AWS_Events_Invoke_Start_Automation_Execution_1213609520"}'
```

Windows

```
aws events put-targets ^
--rule DailyAutomationRule ^
--targets '{"Arn": "arn:aws:ssm:region:*:automation-definition/AWS-
StartEC2Instance","Input":"{\"InstanceId\":[\"i-02573cafcfEXAMPLE\"],
\"AutomationAssumeRole\":[\"arn:aws:iam::123456789012:role/AutomationServiceRole
\"]}","Id": "Target1","RoleArn": "arn:aws:iam::123456789012:role/service-role/
AWS_Events_Invoke_Start_Automation_Execution_1213609520"}'
```

PowerShell

```
$Target = New-Object Amazon.CloudWatchEvents.Model.Target
$Target.Id = "Target1"
$Target.Arn = "arn:aws:ssm:region:*:automation-definition/AWS-StartEC2Instance"
$Target.RoleArn = "arn:aws:iam::123456789012:role/service-role/
AWS_Events_Invoke_Start_Automation_Execution_1213609520"
$Target.Input = '{"InstanceId":["i-02573cafcfEXAMPLE"],"AutomationAssumeRole":
["arn:aws:iam::123456789012:role/AutomationServiceRole"]}'

Write-CWETarget `
-Rule "DailyAutomationRule" `
-Target $Target
```

The system returns information like the following.

Linux & macOS

```
{
"FailedEntries": [],
"FailedEntryCount": 0
}
```

Windows

```
{
"FailedEntries": [],
"FailedEntryCount": 0
}
```

PowerShell

There is no output if the command succeeds for PowerShell.

Run an automation step by step

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 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.

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.

10. After you complete all steps specified in the runbook, choose **Complete and view results** to finish the automation and view the results.

After an automation execution completes, you can rerun the execution with the same or modified parameters. For more information, see Rerunning automation executions.

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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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-SSMAutomationExecution `
    -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-02573cafcfEXAMPLE"
```

Windows

```
aws ssm start-automation-execution ^
    --document-name "AWS-RestartEC2Instance" ^
    --mode Interactive ^
    --parameters "InstanceId=i-02573cafcfEXAMPLE"
```

PowerShell

```
Start-SSMAutomationExecution `
    -DocumentName AWS-RestartEC2Instance `
    -Mode Interactive
    -Parameter @{"InstanceId"="i-02573cafcfEXAMPLE"}
```

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-SSMAutomationSignal `
       -AutomationExecutionId ba9cd881-1b36-4d31-a698-0123456789ab `
       -SignalType StartStep
       -Payload @{"StepName"="stopInstances"}
   ```

   There is no output if the command succeeds.

4. Run the following command to retrieve the status of each step execution in the automation.

   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-SSMAutomationStepExecution `
       -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-02573cafcfEXAMPLE\"]"
               },
               "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-02573cafcfEXAMPLE\"]"
            },
            "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-02573cafcfEXAMPLE\"]"
            },
            "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-02573cafcfEXAMPLE\"]"
            },
            "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-02573cafcfEXAMPLE"]]}
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}
```

5. 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-SSMAutomationExecution `
       -AutomationExecutionId ba9cd881-1b36-4d31-a698-0123456789ab `
       -Type Complete
   ```

   There is no output if the command succeeds.

Scheduling automations with State Manager associations

You can start an automation by creating a State Manager association with a runbook. State Manager is a tool in 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 Window 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). For general information about associations and information about creating an association that uses an SSM `Command` document or `Policy` document, see Creating associations.

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 tool in AWS Systems Manager. For more information, see Setting up Automation.
- 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*.
- When running an automation, State Manager does not log the API operations initiated by the automation in AWS CloudTrail.
- 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 `
-AWSServiceName ssm.amazonaws.com
```

For more information about service-linked roles, see Using service-linked roles for Systems Manager.

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.
      - **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.
      - **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.

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 you specified.

> **Important**
>
> You must specify a role ARN in the **AutomationAssumeRole** field. State Manager uses the assume role to call AWS services specified in the runbook and run Automation associations on your behalf.

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.

   > **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 Run automated operations at scale.

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 Server) 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 tool in AWS Systems Manager. For more information, see Task 1: Create a service role for Automation.

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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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-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 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-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" `
-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 automations run by State Manager associations

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.

Schedule automations with maintenance windows

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 Creating your own runbooks. Automation is a tool in 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 using the console.
- Assign targets to a maintenance window using the console

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”.
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 **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.

12. In the **IAM service role** area, choose a role to provide permissions for Systems Manager to start the automation.

To create a service role for maintenance window tasks, see Setting up Maintenance Windows.

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 Server) 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 using the AWS CLI.
- Step 2: Register a target node with the maintenance window using the AWS CLI

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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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.

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.

PowerShell

```
Register-SSMTaskWithMaintenanceWindow `
-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.

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-SSMTaskWithMaintenanceWindow `
-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
```

3. 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
```

Systems Manager Automation actions reference

This reference describes the Automation actions that you can specify in an Automation runbook. Automation is a tool in 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 Command document plugin reference.

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.

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 or customer managed policy to invoke other AWS services.

Topics

- Properties shared by all actions
- aws:approve – Pause an automation for manual approval
- aws:assertAwsResourceProperty – Assert an AWS resource state or event state
- aws:branch – Run conditional automation steps
- aws:changeInstanceState – Change or assert instance state
- aws:copyImage – Copy or encrypt an Amazon Machine Image
- aws:createImage – Create an Amazon Machine Image
- aws:createStack – Create an AWS CloudFormation stack
- aws:createTags – Create tags for AWS resources
- aws:deleteImage – Delete an Amazon Machine Image
- aws:deleteStack – Delete an AWS CloudFormation stack
- aws:executeAutomation – Run another automation
- aws:executeAwsApi – Call and run AWS API operations
- aws:executeScript – Run a script
- aws:executeStateMachine – Run an AWS Step Functions state machine
- aws:invokeWebhook – Invoke an Automation webhook integration

- aws:invokeLambdaFunction – Invoke an AWS Lambda function
- aws:loop – Iterate over steps in an automation
- aws:pause – Pause an automation
- aws:runCommand – Run a command on a managed instance
- aws:runInstances – Launch an Amazon EC2 instance
- aws:sleep – Delay an automation
- aws:updateVariable – Updates a value for a runbook variable
- aws:waitForAwsResourceProperty – Wait on an AWS resource property
- Automation system variables

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.

description

Information you provide to describe the purpose of a runbook or a step.

Type: String

Required: No

name

An identifier that must be unique across all step names in the runbook.

Type: String

Allowed pattern: [a-zA-Z0-9_]+$

Required: Yes

action

The name of the action the step is to run. `aws:runCommand – Run a command on a managed instance` is an example of an action you can specify here. This document provides detailed information about all available actions.

Type: String

Required: Yes

maxAttempts

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

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

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

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:loop`
- `aws:pause`
- `aws:runInstances`
- `aws:sleep`

isEnd

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

Specifies which step in an automation to process next after successfully completing a step.

Type: String

Required: No

isCritical

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

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: python3.8
    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

This action doesn't support multi-account and Region automations. The default timeout for this action is 7 days (604800 seconds) and the maximum value is 30 days (2592000 seconds). You can limit or extend the timeout by specifying the `timeoutSeconds` parameter for an `aws:approve` step.

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

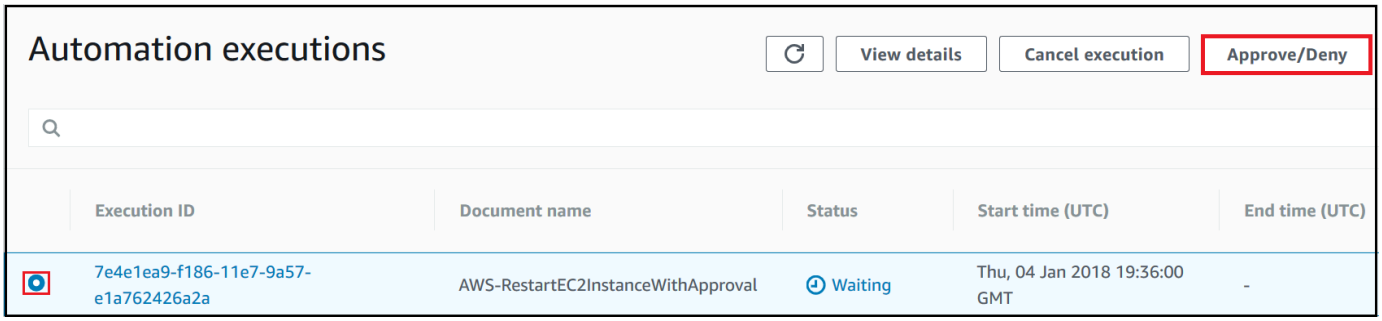
```
{
   "description":"RunInstancesDemo1",
   "schemaVersion":"0.3",
   "assumeRole":"{{ assumeRole }}",
   "parameters":{
      "assumeRole":{
         "type":"String"
      },
      "message":{
         "type":"String"
      }
   },
   "mainSteps":[
      {
         "name":"approve",
         "action":"aws:approve",
         "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**.
3. Choose the option next to an Automation with a status of **Waiting**.



4. Choose **Approve/Deny**.
5. Review the details of the Automation.
6. Choose either **Approve** or **Deny**, type an optional comment, and then choose **Submit**.

Input example

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:

- A user name
- A user ARN
- An IAM role ARN
- An IAM assume role ARN

Type: StringList

Required: Yes

EnhancedApprovals

This input is only used for Change Manager templates. A list of AWS authenticated principals who are able to either approve or reject the action, the type of IAM principal, and the minimum number of approvers. The following is an example:

```
schemaVersion: "0.3"
emergencyChange: false
autoApprovable: false
mainSteps:
    - name: ApproveAction1
    action: aws:approve
    timeoutSeconds: 604800
    inputs:
        Message: Please approve this change request
        MinRequiredApprovals: 3
        EnhancedApprovals:
        Approvers:
            - approver: John Stiles
            type: IamUser
```

```
            minRequiredApprovals: 0
            - approver: Ana Carolina Silva
            type: IamUser
            minRequiredApprovals: 0
            - approver: GroupOfThree
            type: IamGroup
            minRequiredApprovals: 0
            - approver: RoleOfTen
            type: IamRole
            minRequiredApprovals: 0
```

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.

> **Note**
>
> The `aws:assertAwsResourceProperty` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

For more examples of how to use this action, see Additional runbook examples.

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 process 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 Using conditional statements in runbooks.

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.
- **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 Using conditional statements in runbooks.

`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 Server, 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.

> **Note**
>
> The `aws:changeInstanceState` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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.

> **Note**
>
> The `aws:copyImage` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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, and polls for the `ImageState` to be `available`.

> **Note**
>
> The `aws:createImage` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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 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.

> **Note**
>
> The `aws:createStack` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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/amzn-s3-demo-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/amzn-s3-demo-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 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.

JSON

```
{
   "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.

> **Note**
>
> The `aws:createTags` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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.

> **Note**
>
> The aws:deleteImage action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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.

> **Note**
>
> The `aws:deleteStack` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "sqs:*",
            "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 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 Create the service roles for Automation using the console.

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.

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. When specifying a value for this parameter, outputs aren't returned to the parent automation. If needed, you must make subsequent calls to API operations to retrieve the output from child automations.

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. Streaming API operations, such as the GetObject operation, aren't supported. If you're not sure if an API operation you want to use is a streaming operation, review the Boto3 documentation for the service to determine if an API requires streaming inputs or outputs. We regularly update the Boto3 version used by this action. However, following the release of a new Boto3 version it can take up to a few weeks for changes to be reflected in this action. Each `aws:executeAwsApi` action can run up to a maximum duration of 25 seconds. For more examples of how to use this action, see Additional runbook examples.

> **Note**
>
> The `aws:executeAwsApi` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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. 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.

If you want to send output from `aws:executeScript` actions to CloudWatch Logs, or if the scripts you specify for `aws:executeScript` actions call AWS API operations, an AWS Identity and Access Management (IAM) service role (or assume role) is always required to run the runbook.

> **Note**
>
> The `aws:executeScript` action does not support automatic throttling retry. If your script makes AWS API calls that might be throttled, you must implement your own retry logic in your script code.

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 amzn-s3-demo-bucket
  ```

- Amazon EC2 cmdlets.

  ```
  Install-Module AWS.Tools.EC2 -Force
  Get-EC2InstanceStatus -InstanceId instance-id
  ```

- 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-02573cafcfEXAMPLE -Tag $tag
```

For examples of installing and importing `AWS.Tools` modules, and using PowerShell Core cmdlets in runbooks, see Visual design experience for Automation runbooks.

Input

Provide the information required to run your script. Replace each *example resource placeholder* with your own information.

> **Note**
>
> The attachment for a Python script can be a .py file or a .zip file that contains the script. PowerShell scripts must be stored in .zip files.

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": {
            "scriptInput": "{{parameterValue}}"
        },
        "Attachment": "scriptAttachment.zip"
```

```
    }
}
```

Runtime

The runtime language to be used for running the provided script. `aws:executeScript` supports Python 3.7 (`python3.7`), Python 3.8 (`python3.8`), Python 3.9 (`python3.9`)Python 3.10 (`python3.10`), Python 3.11 (`python3.11`) PowerShell Core 6.0 (`dotnetcore2.1`), PowerShell 7.0 (`dotnetcore3.1`) scripts, and PowerShell 7.4 (`dotnet8`) scripts.

Supported values: **`python3.7`** | **`python3.8`** | **`python3.9`** | **`python3.10`** | **`python3.11`** | **`PowerShell Core 6.0`** | **`PowerShell 7.0`** | **`PowerShell 7.4`**

Type: String

Required: Yes

> **Note**
>
> For python runtimes, the environment provides 512MB of memory and 512MB of disk space. For PowerShell runtimes, the environment provides 1024MB of memory and 512MB of disk space.

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

Python

```
description: Tag an instance
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.'
    InstanceId:
        type: String
        description: (Required) The ID of the EC2 instance you want to tag.
mainSteps:
  - name: tagInstance
    action: 'aws:executeScript'
    inputs:
        Runtime: "python3.11"
        Handler: tagInstance
        InputPayload:
            instanceId: '{{InstanceId}}'
        Script: |-
          def tagInstance(events,context):
            import boto3

            #Initialize client
            ec2 = boto3.client('ec2')
            instanceId = events['instanceId']
            tag = {
                "Key": "Env",
                "Value": "ExamplePython"
            }
            print(f"Adding tag {tag} to instance id {instanceId}")
            ec2.create_tags(
                Resources=[instanceId],
                Tags=[tag]
            )
            return tag
    outputs:
      - Type: String
        Name: TagKey
        Selector: $.Payload.Key
```

```
outputs:
  - tagInstance.TagKey
```

PowerShell

```
description: Tag an instance
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.
  InstanceId:
    type: String
    description: (Required) The ID of the EC2 instance you want to tag.
mainSteps:
  - name: tagInstance
    action: aws:executeScript
    isEnd: true
    inputs:
      Runtime: PowerShell 7.0
      InputPayload:
        instanceId: '{{InstanceId}}'
      Script: |-
        Install-Module AWS.Tools.EC2 -Force
        Import-Module AWS.Tools.EC2

        $input = $env:InputPayload | ConvertFrom-Json

        $tag = New-Object Amazon.EC2.Model.Tag
        $tag.Key = "Env"
        $tag.Value = "ExamplePowerShell"

        Write-Information "Adding tag key: $($tag.Key) and value: $($tag.Value)
 to instance id $($input.instanceId)"
        New-EC2Tag -Resource $input.instanceId -Tag $tag

        return $tag
    outputs:
      - Type: String
```

```
        Name: TagKey
        Selector: $.Payload.Key
outputs:
  - tagInstance.TagKey
```

Script

An embedded script that you want to run during the automation.

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 of your runbook. For more information, see Schema version 0.3.

To invoke a file for Python, use the `filename.method_name` format in `Handler`.

> **Note**
>
> The attachment for a Python script can be a .py file or a .zip file that contains the script. PowerShell scripts must be stored in .zip files.

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. If you output a list, a maximum of 100 items is returned.

Using attachments with aws:executeScript

Attachments provide a powerful way to package and reuse complex scripts, multiple modules, and external dependencies with your `aws:executeScript` actions. Use attachments when you need to:

- Package multiple Python modules or PowerShell scripts together.
- Reuse the same script logic across multiple runbooks.
- Include external libraries or dependencies with your scripts.
- Keep your runbook definition clean by separating complex script logic.
- Share script packages across teams or automation workflows.

Attachment structure and packaging

You can attach either single files or zip packages containing multiple files. The structure depends on your use case:

Single file attachments

For simple scripts, you can attach a single `.py` file (Python) or a `.zip` file containing a single PowerShell script.

Multi-module packages

For complex automation that requires multiple modules, create a zip package with the following recommended structure:

```
my-automation-package.zip
### main.py                    # Entry point script
### utils/
#   ### __init__.py           # Required for Python module imports
#   ### helper_functions.py   # Utility functions
```

```
#   ### aws_operations.py     # AWS-specific operations
### config/
#   ### __init__.py
#   ### settings.py           # Configuration settings
### requirements.txt          # Optional: document dependencies
```

> **Important**
>
> For Python packages, you must include an empty `__init__.py` file in each directory that contains Python modules. This allows you to import modules using standard Python import syntax like `from utils import helper_functions`.

PowerShell package structure

PowerShell attachments must be packaged in zip files with the following structure:

```
my-powershell-package.zip
### Main.ps1                  # Entry point script
### Modules/
#   ### HelperFunctions.ps1   # Utility functions
#   ### AWSOperations.ps1     # AWS-specific operations
### Config/
    ### Settings.ps1          # Configuration settings
```

Creating runbooks with attachments

Follow these steps to create runbooks that use attachments:

1. **Upload your attachment to Amazon S3**

   Upload your script file or zip package to an S3 bucket that your automation role can access. Note the S3 URI for use in the next step.

   ```
   aws s3 cp my-automation-package.zip s3://my-automation-bucket/scripts/
   ```

2. **Calculate the attachment checksum**

   Calculate the SHA-256 checksum of your attachment file for security verification:

   ```
   # Linux/macOS
   ```

```
shasum -a 256 my-automation-package.zip

# Windows PowerShell
Get-FileHash -Algorithm SHA256 my-automation-package.zip
```

3. **Define the files section in your runbook**

   Add a `files` section at the top level of your runbook to reference your attachment:

```
files:
  my-automation-package.zip:
    sourceType: "S3"
    sourceInfo:
      path: "s3://my-automation-bucket/scripts/my-automation-package.zip"
      checksums:
        sha256: "your-calculated-checksum-here"
```

4. **Reference the attachment in your executeScript step**

   Use the `Attachment` parameter to reference your uploaded file:

```
- name: runMyScript
  action: aws:executeScript
  inputs:
    Runtime: python3.11
    Handler: main.process_data
    Attachment: my-automation-package.zip
    InputPayload:
      inputData: "{{InputParameter}}"
```

aws:executeScript attachment examples

The following examples demonstrate different ways to use attachments with the `aws:executeScript` action.

Example 1: Single file attachment

This example shows how to use a single Python file as an attachment to process EC2 instance data.

Attachment file: process_instance.py

Create a Python file with the following content:

```
import boto3
import json

def process_instance_data(events, context):
    """Process EC2 instance data and return formatted results."""
    try:
        instance_id = events.get('instanceId')
        if not instance_id:
            raise ValueError("instanceId is required")

        ec2 = boto3.client('ec2')

        # Get instance details
        response = ec2.describe_instances(InstanceIds=[instance_id])
        instance = response['Reservations'][0]['Instances'][0]

        # Format the response
        result = {
            'instanceId': instance_id,
            'instanceType': instance['InstanceType'],
            'state': instance['State']['Name'],
            'availabilityZone': instance['Placement']['AvailabilityZone'],
            'tags': {tag['Key']: tag['Value'] for tag in instance.get('Tags', [])}
        }

        print(f"Successfully processed instance {instance_id}")
        return result

    except Exception as e:
        print(f"Error processing instance: {str(e)}")
        raise
```

Complete runbook

Here's the complete runbook that uses the single file attachment:

```
description: Process EC2 instance data using single file attachment
schemaVersion: '0.3'
assumeRole: '{{AutomationAssumeRole}}'
parameters:
  AutomationAssumeRole:
    type: String
    description: (Required) IAM role for automation execution
```

```
  InstanceId:
    type: String
    description: (Required) EC2 instance ID to process

files:
  process_instance.py:
    sourceType: "S3"
    sourceInfo:
      path: "s3://my-automation-bucket/scripts/process_instance.py"
      checksums:
        sha256: "abc123def456..."

mainSteps:
  - name: processInstance
    action: aws:executeScript
    inputs:
      Runtime: python3.11
      Handler: process_instance.process_instance_data
      Attachment: process_instance.py
      InputPayload:
        instanceId: '{{InstanceId}}'
    outputs:
      - Type: StringMap
        Name: InstanceData
        Selector: $.Payload

outputs:
  - processInstance.InstanceData
```

Example 2: Multi-module package

This example demonstrates using a zip package containing multiple Python modules for complex S3 bucket operations.

Package structure

Create a zip package with the following structure:

```
s3-operations.zip
### main.py
### utils/
#   ### __init__.py
#   ### s3_helper.py
```

```
#   ### validation.py
### config/
    ### __init__.py
    ### settings.py
```

main.py (entry point)

The main script that orchestrates the operations:

```
from utils.s3_helper import S3Operations
from utils.validation import validate_bucket_name
from config.settings import get_default_settings

def cleanup_s3_bucket(events, context):
    """Clean up S3 bucket based on specified criteria."""
    try:
        bucket_name = events.get('bucketName')
        max_age_days = events.get('maxAgeDays', 30)

        # Validate inputs
        if not validate_bucket_name(bucket_name):
            raise ValueError(f"Invalid bucket name: {bucket_name}")

        # Initialize S3 operations
        s3_ops = S3Operations()
        settings = get_default_settings()

        # Perform cleanup
        deleted_objects = s3_ops.delete_old_objects(
            bucket_name,
            max_age_days,
            settings['dry_run']
        )

        result = {
            'bucketName': bucket_name,
            'deletedCount': len(deleted_objects),
            'deletedObjects': deleted_objects[:10],  # Return first 10 for brevity
            'dryRun': settings['dry_run']
        }

        print(f"Cleanup completed for bucket {bucket_name}")
        return result
```

```
    except Exception as e:
        print(f"Error during S3 cleanup: {str(e)}")
        raise
```

Troubleshooting aws:executeScript attachments

Use the following guidance to resolve common issues with `aws:executeScript` attachments:

Module import errors

If you receive import errors when using multi-module packages:

- Ensure you have included an empty `__init__.py` file in each directory containing Python modules.
- Verify that your import statements match the actual file and directory structure in your zip package.
- Use relative imports (e.g., `from .utils import helper`) or absolute imports (e.g., `from utils import helper`) consistently.

Attachment not found errors

If your automation fails to find the attachment:

- Verify that the `Attachment` parameter value exactly matches the key in your `files` section.
- Check that your S3 bucket path and file name are correct in the `files` section.
- Ensure your automation role has `s3:GetObject` permission for the attachment S3 location.
- Verify that the checksum in your runbook matches the actual file checksum.

Handler function errors

If you receive handler-related errors:

- For Python: Use the format `filename.function_name` in the `Handler` parameter (e.g., `main.process_data`).
- Ensure your handler function accepts exactly two parameters: `events` and `context`.
- For PowerShell: Do not specify a `Handler` parameter; the script runs directly.

Script execution failures

If your script fails during execution:

- Check the automation execution history for detailed error messages and stack traces.
- Use `print()` statements (Python) or `Write-Information` (PowerShell) to add debugging output.
- Verify that all required AWS permissions are granted to your automation role.
- Test your script logic locally before packaging it as an attachment.

Exit codes and error handling

To properly handle errors and return exit codes:

- In Python: Use `raise Exception("error message")` to indicate script failure.
- In PowerShell: Use `throw "error message"` or `Write-Error` to indicate failure.
- Return structured data from your functions to provide detailed success/failure information.
- Use try-catch blocks to handle exceptions gracefully and provide meaningful error messages.

`aws:executeStateMachine` – Run an AWS Step Functions state machine

Runs an AWS Step Functions state machine.

> **Note**
>
> The `aws:executeStateMachine` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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

Outputs

The following outputs are predefined for this action.

executionArn

The ARN of the execution.

Type: String

input

The string that contains the JSON input data of the execution. Length constraints apply to the payload size, and are expressed as bytes in UTF-8 encoding..

Type: String

name

The name of the execution.

Type: String

output

The JSON output data of the execution. Length constraints apply to the payload size, and are expressed as bytes in UTF-8 encoding.

Type: String

startDate

The date the execution is started.

Type: String

stateMachineArn

The ARN of the executed stated machine.

Type: String

status

The current status of the execution.

Type: String

stopDate

If the execution has already ended, the date the execution stopped.

Type: String

`aws:invokeWebhook` – Invoke an Automation webhook integration

Invokes the specified Automation webhook integration. For information about creating Automation integrations, see Creating webhook integrations for Automation.

> **Note**
>
> The `aws:invokeWebhook` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

> **Note**
>
> To use the `aws:invokeWebhook` action, your 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.

Note

The `aws:invokeLambdaFunction` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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

InputPayload

A YAML or JSON object that is passed to the first parameter of the handler. You can use this input to pass data to the function. This input provides more flexibility and support than

the legacy `Payload` input. If you define both `InputPayload` and `Payload` for the action, `InputPayload` takes precedence and the `Payload` value is not used.

Type: StringMap

Required: No

Payload

A JSON string that's passed to the first parameter of the handler. This can be used to pass input data to the function. We recommend using `InputPayload` input for added functionality.

Type: String

Required: No

Output

StatusCode

The HTTP status code.

FunctionError

If present, it indicates that an error occurred while executing the function. Error details are included in the response payload.

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`.

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:loop – Iterate over steps in an automation

This action iterates over a subset of steps in an automation runbook. You can choose a do `while` or `for each` style loop. To construct a do `while` loop, use the `LoopCondition` input parameter. To construct a `for each` loop, use the `Iterators` and `IteratorDataType` input parameters. When using an `aws:loop` action, only specify either the `Iterators` or `LoopCondition` input parameter. The maximum number of iterations is 100.

The onCancel property can only be used for steps defined within a loop. The onCancel property isn't supported for the aws:loop action. The onFailure property can be used for an aws:loop action, however it will only be used if an unexpected error occurs causing the step to fail. If you define onFailure properties for the steps within a loop, the aws:loop action inherits those properties and reacts accordingly when a failure occurs.

Examples

The following are examples of how to construct the different types of loop actions.

do while

```
name: RepeatMyLambdaFunctionUntilOutputIsReturned
action: aws:loop
inputs:
    Steps:
    - name: invokeMyLambda
        action: aws:invokeLambdaFunction
        inputs:
        FunctionName: LambdaFunctionName
        outputs:
        - Name: ShouldRetry
            Selector: $.Retry
            Type: Boolean
    LoopCondition:
        Variable: "{{ invokeMyLambda.ShouldRetry }}"
        BooleanEquals: true
    MaxIterations: 3
```

for each

```
name: stopAllInstancesWithWaitTime
action: aws:loop
inputs:
    Iterators: "{{ DescribeInstancesStep.InstanceIds }}"
    IteratorDataType: "String"
    Steps:
    - name: stopOneInstance
        action: aws:changeInstanceState
        inputs:
        InstanceIds:
            - "{{stopAllInstancesWithWaitTime.CurrentIteratorValue}}"
        CheckStateOnly: false
```

```
        DesiredState: stopped
    - name: wait10Seconds
        action: aws:sleep
        inputs:
        Duration: PT10S
```

Input

The input is as follows.

Iterators

The list of items for the steps to iterate over. The maximum number of iterators is 100.

Type: StringList

Required: No

IteratorDataType

An optional parameter to specify the data type of the `Iterators`. A value for this parameter can be provided along with the `Iterators` input parameter. If you don't specify a value for this parameter and `Iterators`, then you must specify a value for the `LoopCondition` parameter.

Type: String

Valid values: Boolean | Integer | String | StringMap

Default: String

Required: No

LoopCondition

Consists of a `Variable` and an operator condition to evaluate. If you don't specify a value for this parameter, then you must specify values for the `Iterators` and `IteratorDataType` parameters. You can use complex operator evaluations by using a combination of `And`, `Not`, and `Or` operators. The condition is evaluated after the steps in the loop complete. If the condition is `true` and the `MaxIterations` value has not been reached, the steps in the loop run again. The operator conditions are as follows:

String operations

- StringEquals

- EqualsIgnoreCase
- StartsWith
- EndsWith
- Contains

Numeric operations

- NumericEquals
- NumericGreater
- NumericLesser
- NumericGreaterOrEquals
- NumericLesser
- NumericLesserOrEquals

Boolean operation

- BooleanEquals

Type: StringMap

Required: No

MaxIterations

The maximum number of times the steps in the loop run. Once the value specified for this input is reached, the loop stops running even if the `LoopCondition` is still `true` or if there are objects remaining in the `Iterators` parameter.

Type: Integer

Valid values: 1 - 100

Required: No

Steps

The list of steps to run in the loop. These function like a nested runbook. Within these steps you can access the current iterator value for a `for each` loop using the syntax `{{loopStepName.CurrentIteratorValue}}`. You can also access an integer value of the current iteration for both loop types using the syntax `{{loopStepName.CurrentIteration}}`.

Type: List of steps

Required: Yes

Output

CurrentIteration

The current loop iteration as an integer. Iteration values start at 1.

Type: Integer

CurrentIteratorValue

The value of the current iterator as a string. This output is only present in `for each` loops.

Type: String

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.

> **Note**
>
> The default timeout for this action is 7 days (604800 seconds) and the maximum value is 30 days (2592000 seconds). You can limit or extend the timeout by specifying the `timeoutSeconds` parameter for an `aws:pause` step.

Input

The input is as follows.

YAML

```
name: pauseThis
action: aws:pause
timeoutSeconds: 1209600
```

```
inputs: {}
```

JSON

```
{
    "name": "pauseThis",
    "action": "aws:pause",
    "timeoutSeconds": "1209600",
    "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.

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 Using pseudo parameters when registering maintenance window tasks.

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 Run commands at scale.

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 Run commands at scale.

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.

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. Your managed node must have permissions for the S3 bucket to successfully log output.

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

Valid values: 30-2592000

Output

CommandId

The ID of the command.

Status

The status of the command.

ResponseCode

The response code of the command. If the document you run has more than 1 step, a value isn't returned for this output.

Output

The output of the command. If you target a tag or multiple instances with your command, no output value is returned. You can use the `GetCommandInvocation` and `ListCommandInvocations` API operations to retrieve output for individual instances.

`aws:runInstances` – Launch an Amazon EC2 instance

Launches a new Amazon Elastic Compute Cloud (Amazon EC2) instance.

> **Note**
>
> The `aws:runInstances` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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"
               },
               {
                  "Key":"Category",
                  "Value":"HighAvailabilityFleetHost"
               }
            ]
         }
      ]
   }
}
```

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

ImageId

The ID of the Amazon Machine Image (AMI).

Type: String

Required: Yes

InstanceInitiatedShutdownBehavior

Indicates whether the instance stops or terminates on system shutdown.

Type: String

Required: No

InstanceType

The instance type.

> **Note**
>
> If an instance type value isn't provided, the m1.small instance type is used.

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 be launched.

Type: String

Required: No

MetadataOptions

The metadata options for the instance. For more information, see InstanceMetadataOptionsRequest.

Type: StringMap

Required: No

MinInstanceCount

The minimum number of instances to be launched.

Type: String

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 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.

InstanceStates

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 604799 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:updateVariable` – Updates a value for a runbook variable

This action updates a value for a runbook variable. The data type of the value must match the data type of the variable you want to update. Data type conversions aren't supported. The on`Cancel` property isn't supported for the `aws:updateVariable` action.

Input

The input is as follows.

YAML

```
name: updateStringList
action: aws:updateVariable
```

```
inputs:
    Name: variable:variable name
    Value:
    - "1"
    - "2"
```

JSON

```
{
    "name": "updateStringList",
    "action": "aws:updateVariable",
    "inputs": {
        "Name": "variable:variable name",
        "Value": ["1","2"]
    }
}
```

Name

The name of the variable whose value you want to update. You must use the format `variable:variable name`

Type: String

Required: Yes

Value

The new value to assign to the variable. The value must match the data type of the variable. Data type conversions aren't supported.

Type: Boolean | Integer | MapList | String | StringList | StringMap

Required: Yes

Constraints:

- MapList can contain a maximum number of 200 items.
- Key lengths can be a minimum length of 1 and a maximum length of 50.
- StringList can be a minimum number of 0 items and a maximum number of 50 items.
- String lengths can be a minimum length of 1 and a maximum length of 512.

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 examples of how to use this action, see Additional runbook examples.

> **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.

> **Note**
>
> The `aws:waitForAwsResourceProperty` action supports automatic throttling retry. For more information, see Configuring automatic retry for throttled operations.

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: String

Required: Yes

DesiredValues

The expected status or state on which to continue the automation.

Type: MapList, StringList

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
`global:ACCOUNT_ID`	The AWS account ID of the user or role in which Automation runs.
`global:DATE`	The date (at run time) in the format yyyy-MM-dd.
`global:DATE_TIME`	The date and time (at run time) in the format yyyy-MM-dd_HH.mm.ss.
`global:AWS_PARTITION`	The partition that the resource is in. For standard AWS Regions, the partition is `aws`. For resources in other partitions, the partition is `aws-`*`partitionname`* . For example, the

Variable	Details
	partition for resources in the AWS GovCloud (US-West) Region is `aws-us-gov` .
`global:REGION`	The Region that the runbook is run in. For example, us-east-2.

Automation variables

Automation runbooks support the following automation variables.

Variable	Details
`automation:EXECUTION_ID`	The unique identifier assigned to the current automation. For example, `1a2b3c-1a2b3c-1a2b3c-1a2b3c1a2b3c1a2b3c` .

Topics

- Terminology
- Supported scenarios
- Unsupported scenarios

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, `instanceId`).	`{ "description": "Create Image Demo",`

Term	Definition	Example
	The parameter is used in a basic string replace. Its value is supplied at Start Execution time.	"version": "0.3", "assumeRole": "*Your_Automation_Assume_Role_ARN* ", "parameters":{ **"instanceId": { "type": "String", "description": "Instance to create image from"** } }

Term	Definition	Example
System variable	A general variable substituted into the runbook when any part of the runbook is evaluated.	`"activities": [{ "id": "copyImage", "activityType": "AWS-CopyImage", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "ImageName": "{{imageName}}", "SourceImageId": "{{sourceImageId}}", "SourceRegion": "{{sourceRegion}}", "Encrypted": true, "ImageDescription": "Test CopyImage Description created on **{{global:DATE}}** " } }]`

Term	Definition	Example
Automation variable	A variable relating to the automation substituted into the runbook when any part of the runbook is evaluated.	`{ "name": "runFixedCmds", "action": "aws:runCommand", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "DocumentName": "AWS-RunPowerShellScript", "InstanceIds": ["{{LaunchInstance.InstanceIds}}"], "Parameters": { "commands": ["dir", "date", "“{{outputFormat}}” -f “left”,”right”,”{{global:DATE}}”,” **{{automation:EXECUTION_ID}}** ”] } } }`

Term	Definition	Example

Systems Manager Parameter	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.	description: Launch new Windows test instance
		schemaVersion: '0.3'
		assumeRole: '{{AutomationAssumeRole}}'
		parameters:
		  AutomationAssumeRole:
		    type: String
		    default: ''
		    description: >-
		      (Required) The ARN of the role that allows Automation to perform the
		      actions on your behalf. If no role is specified, Systems Manager
		      Automation uses your IAM permissions to run this runbook.
		  LatestAmi:
		    type: String
		    default: >-
		      {{ssm:/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-Base}}
		    description: The latest Windows Server 2016 AMI queried from the public parameter.
		mainSteps:
		  - name: launchInstance
		    action: 'aws:runInstances'
		    maxAttempts: 3

Term	Definition	Example
		timeoutSeconds: 1200 onFailure: Abort inputs: ImageId: '{{LatestAmi}}' ...

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` .	{ "description": "Test all Automation resolvable parameters", "schemaVersion": "0.3", "assumeRole": ***"arn:aws:iam::123456789012:role/roleName"*** , "parameters": { ...
Runbook parameter supplied for `AssumeRole` when the automation is started.	Must be defined in the parameter list of the runbook.	{ "description": "Test all Automation resolvable parameters", "schemaVersion": "0.3", "assumeRole": **"{{dynamicARN}}"** , "parameters": { ...

Scenario	Comments	Example

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":
		"ami-12345678 ",
		      "description":
		"list of commands to
		run as part of first
		step"
		    },
		...</pre>
		Inputs to Start Automation Execution include : {"amiId" : ["ami-12345678 "] }

Scenario	Comments	Example

Systems Manager Parameter referenced within runbook content.	The variable exists within the customer's account, or is a publicly accessibly 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.	```
		...
		parameters:
		    LatestAmi:
		    type: String
		    default: >-
		      {{ssm:/aws/
		service/ami-wind
		ows-latest/Windows
		_Server-2016-English-
		Full-Base}}
		    description: The
		 latest Windows Server
		 2016 AMI queried from
		 the public parameter.
		mainSteps:
		  - name: launchIns
		tance
		    action: 'aws:runI
		nstances'
		    maxAttempts: 3
		    timeoutSeconds:
		 1200
		    onFailure: Abort
		    inputs:
		      ImageId: '{{Latest
		Ami}}'
		...
		``

Scenario	Comments	Example
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.	... "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}}}"] } } }, ...

Scenario	Comments	Example
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 **automation:EXECUTION_ID**.	... "mainSteps": [{ "name": "invokeLambdaFunction", "action": "aws:invokeLambdaFunction", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "FunctionName": "Hello-World-LambdaFunction", "Payload" : "{ "executionId" : "{{automation:EXECUTION_ID}}" }" } } ...

Scenario	Comments	Example

Refer to output from previous step within next step definitio n.	This is parameter redirection. The output of a previous step is referenced using the syntax `{{stepName.OutputN ame}}` . 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.	```
		...
		"mainSteps": [
		    {
		      "name": "LaunchIn
		stance",
		      "action":
		 "aws:runInstances",
		      "maxAttempts": 1,
		      "onFailure":
		 "Continue",
		      "inputs": {
		        "ImageId":
		 "{{amiId}}",
		        "MinInsta
		nceCount": 1,
		        "MaxInsta
		nceCount": 2
		      }
		    },
		    {
		      "name":"c
		hangeState",
		      "action":
		 "aws:changeInstanc
		eState",
		      "maxAttempts": 1,
		      "onFailure":
		 "Continue",
		      "inputs": {
		        "InstanceIds":
		 ["{{LaunchInstance
		.InstanceIds}}"],
		        "DesiredState":
		 "terminated"
		      }
		    }
		
		...
		``

Unsupported scenarios

Scenario	Comment	Example

Systems Manager Parameter supplied for `assumeRole` at create	Not supported.	`...`
		
		`{`
		`  "description":`
		` "Test all Automation`
		` resolvable parameter`
		`s",`
		`  "schemaVersion":`
		` "0.3",`
		`  "assumeRole":`
		` "{{ssm:administrato`
		`rRoleARN}}  ",`
		`  "parameters": {`
		
		`...`

Systems Manager Parameter directly referenced in step input.	Returns `InvalidDocumentContent` exception at create time.	`...`
		`mainSteps:`
		`  - name: launchIns`
		`tance`
		`    action: 'aws:runI`
		`nstances'`
		`    maxAttempts: 3`
		`    timeoutSeconds:`
		` 1200`
		`    onFailure: Abort`
		`    inputs:`
		`      ImageId: '{{ssm:/`
		`aws/service/ami-win`
		`dows-latest/Window`
		`s_Server-2016-Engl`
		`ish-Full-Base}}'`
		`...`

Scenario	Comment	Example
Variable step definition	The definition of a step in the runbook is constructed by variables.	... "mainSteps": [{ "name": "LaunchInstance", "action": "aws:runInstances", "**{{attemptModel}}** ": 1, "onFailure": "Continue", "inputs": { "ImageId": "*ami-12345678* ", "MinInstanceCount": 1, "MaxInstanceCount": 2 } ... User supplies input : { "attemptModel" : "**minAttempts** " }

Scenario	Comment	Example
Cross referencing runbook parameters	The user supplies an input parameter at start time, which is a reference to another parameter in the runbook.	... "parameters": { "amiId": { "type": "String", "default": "ami-7f2e6015 ", "description": "list of commands to run as part of first step" }, "alternateAmiId": { "type": "String", "description": "The alternate AMI to try if this first fails". "default" : "{{amiId} }" }, ...

Scenario	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.	... "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}}*** }}"] }

Scenario	Comment	Example
		... Note: The customer intention here would be to run a command of "echo {Hello world}"

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.	... 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

Scenario	Comment	Example
		`outputs a "StringList" type variable.`

Creating your own 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 tool in 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.

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 or customer managed policy to invoke other AWS services.

For information about the actions that you can specify in a runbook, see Systems Manager Automation actions reference.

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 the visual designer to create a custom runbook, see Visual design experience for Automation runbooks.

Contents

- Visual design experience for Automation runbooks
  - Before you begin
  - Overview of the visual design experience interface
    - Actions browser
    - Canvas
    - Form
    - Keyboard shortcuts
  - Using the visual design experience
    - Create a runbook workflow

    - Design a runbook
    - Update your runbook
    - Export your runbook
- Configuring inputs and outputs for your actions
    - Provide input data for an action
    - Define output data for an action
- Error handling with the visual design experience
    - Retry action on error
    - Timeouts
    - Failed actions
    - Canceled actions
    - Critical actions
    - Ending actions
- Tutorial: Create a runbook using the visual design experience
    - Step 1: Navigate to the visual design experience
    - Step 2: Create a workflow
    - Step 3: Review the auto-generated code
    - Step 4: Run your new runbook
    - Step 5: Clean up
- Authoring Automation runbooks
    - Identify your use case
    - Set up your development environment
    - Develop runbook content
    - Example 1: Creating parent-child runbooks
        - Create the child runbook
        - Create the parent runbook
    - Example 2: Scripted runbook
    - Additional runbook examples
        - Deploy VPC architecture and Microsoft Active Directory domain controllers
        - Restore a root volume from the latest snapshot

        - Create an AMI and cross-Region copy
- Creating input parameters that populate AWS resources
- Using Document Builder to create runbooks
  - Create a runbook using Document Builder
  - Create a runbook that runs scripts
- Using scripts in runbooks
  - Permissions for using runbooks
  - Adding scripts to runbooks
  - Script constraints for runbooks
- Using conditional statements in runbooks
  - Working with the aws:branch action
    - Creating an aws:branch step in a runbook
      - About creating the output variable
    - Example aws:branch runbooks
    - Creating complex branching automations with operators
  - Examples of how to use conditional options
- Using action outputs as inputs
  - Using JSONPath in runbooks
- Creating webhook integrations for Automation
  - Creating integrations (console)
  - Creating integrations (command line)
  - Creating webhooks for integrations
- Handling timeouts in runbooks

Visual design experience for Automation runbooks

AWS Systems Manager Automation provides a low-code visual design experience that helps you create automation runbooks. The visual design experience provides a drag-and-drop interface with the option to add your own code so you can create and edit runbooks more easily. With the visual design experience, you can do the following:

- Control conditional statements.

- Control how input and output is filtered or transformed for each action.
- Configure error handling.
- Prototype new runbooks.
- Use your prototype runbooks as the starting point for local development with the AWS Toolkit for Visual Studio Code.

When you create or edit a runbook, you can access the visual design experience from the Automation console. As you create a runbook, the visual design experience validates your work and auto-generates code. You can review the generated code, or export it for local development. When you're finished, you can save your runbook, run it, and examine the results in the Systems Manager Automation console.

Before you begin

To use the visual design experience, you need an AWS account, and credentials that provide the correct permissions for any resources that you want to use.

In the visual design experience, Automation integrates with Amazon CodeGuru Security to help you detect security policy violations and vulnerabilities in your Python scripts. To use this feature for `aws:executeScript` actions, your AWS Identity and Access Management (IAM) policy must include the following permissions:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codeguru-security:CreateUploadUrl",
                "codeguru-security:CreateScan",
                "codeguru-security:GetScan",
                "codeguru-security:GetFindings"
            ],
            "Resource": "*"
        }
    ]
}
```

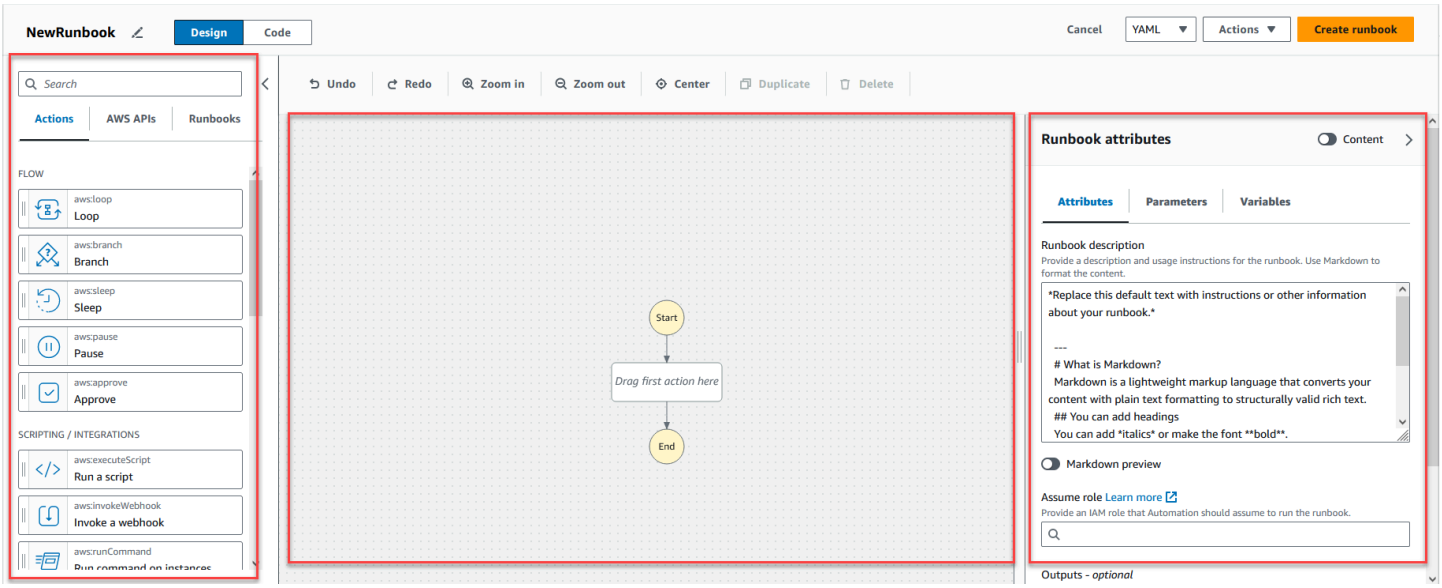
Topics

- Overview of the visual design experience interface
- Using the visual design experience
- Configuring inputs and outputs for your actions
- Error handling with the visual design experience
- Tutorial: Create a runbook using the visual design experience

Overview of the visual design experience interface

The visual design experience for Systems Manager Automation is a low-code visual workflow designer that helps you create automation runbooks.

Get to know the visual design experience with an overview of the interface components:



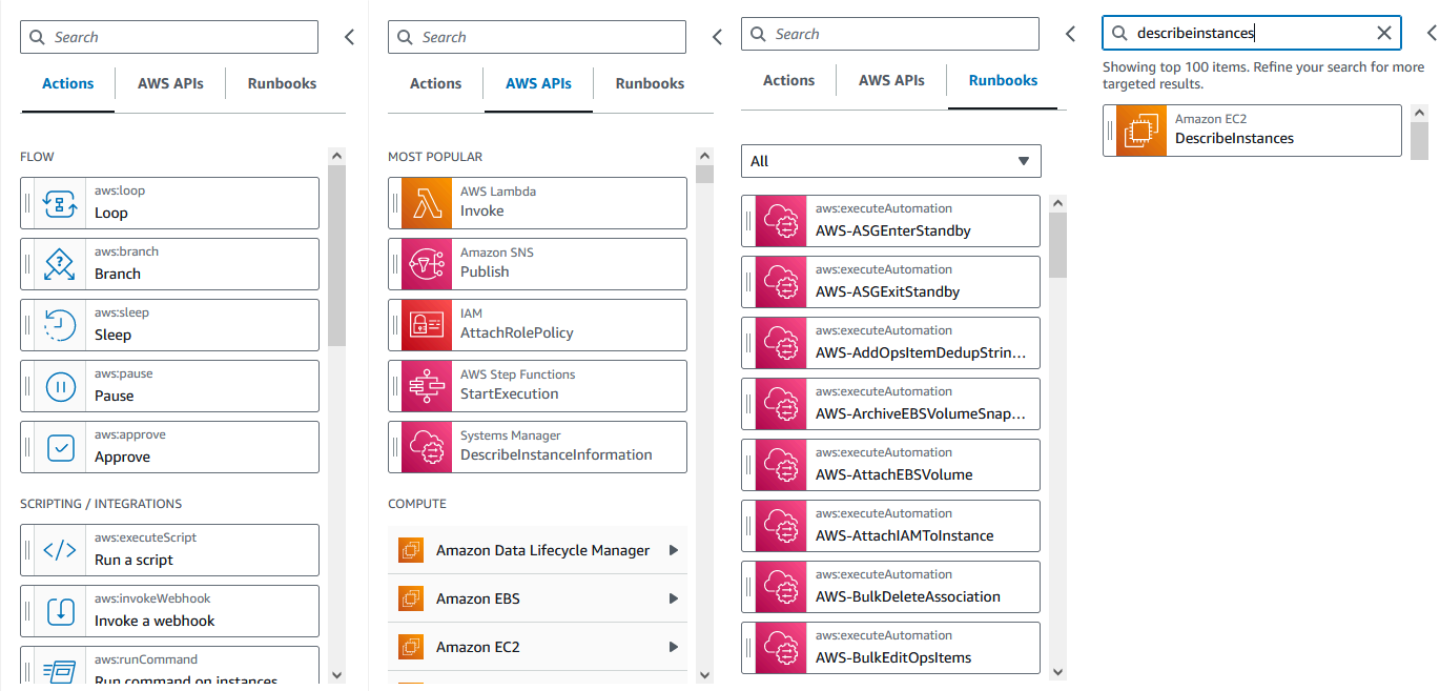
- The **Actions** browser contains the **Actions**, **AWS APIs**, and **Runbooks** tabs.
- The *canvas* is where you drag and drop actions into your workflow graph, change the order of actions, and select actions to configure or view.
- The **Form** panel is where you can view and edit the properties of any action that you selected on the canvas. Select the **Content** toggle to view the YAML or JSON for your runbook, with the currently selected action highlighted.

Info links open a panel with contextual information when you need help. These panels also include links to related topics in the Systems Manager Automation documentation.

Actions browser

From the **Actions** browser, you can select actions to drag and drop into your workflow graph. You can search all actions using the search field at the top of the **Actions** browser. The **Actions** browser contains the following tabs:

- The **Actions** tab provides a list of automation actions that you can drag and drop into your runbook's workflow graph in the canvas.
- The **AWS APIs** tab provides a list of AWS APIs that you can drag and drop into your runbook's workflow graph in the canvas.
- The **Runbooks** tab provides several ready-to-use, reusable runbooks as building blocks that you can use for a variety of use cases. For example, you can use runbooks to perform common remediation tasks on Amazon EC2 instances in your workflow without having to re-create the same actions.

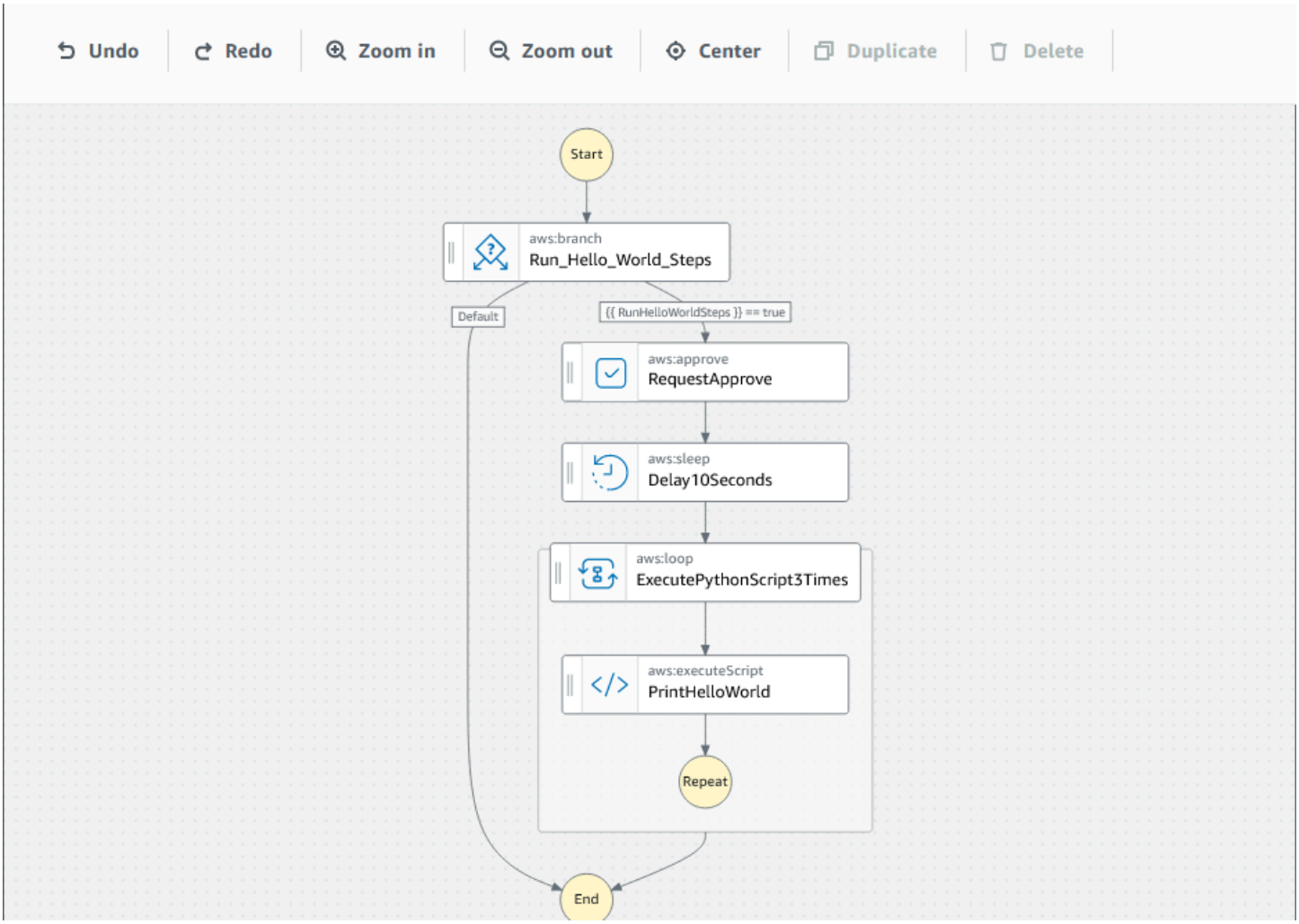


Canvas

After you choose an action to add to your automation, drag it to the canvas and drop it into your workflow graph. You can also drag and drop actions to move them to different places in your runbook's workflow. If your workflow is complex, you might not be able to view all of it in the

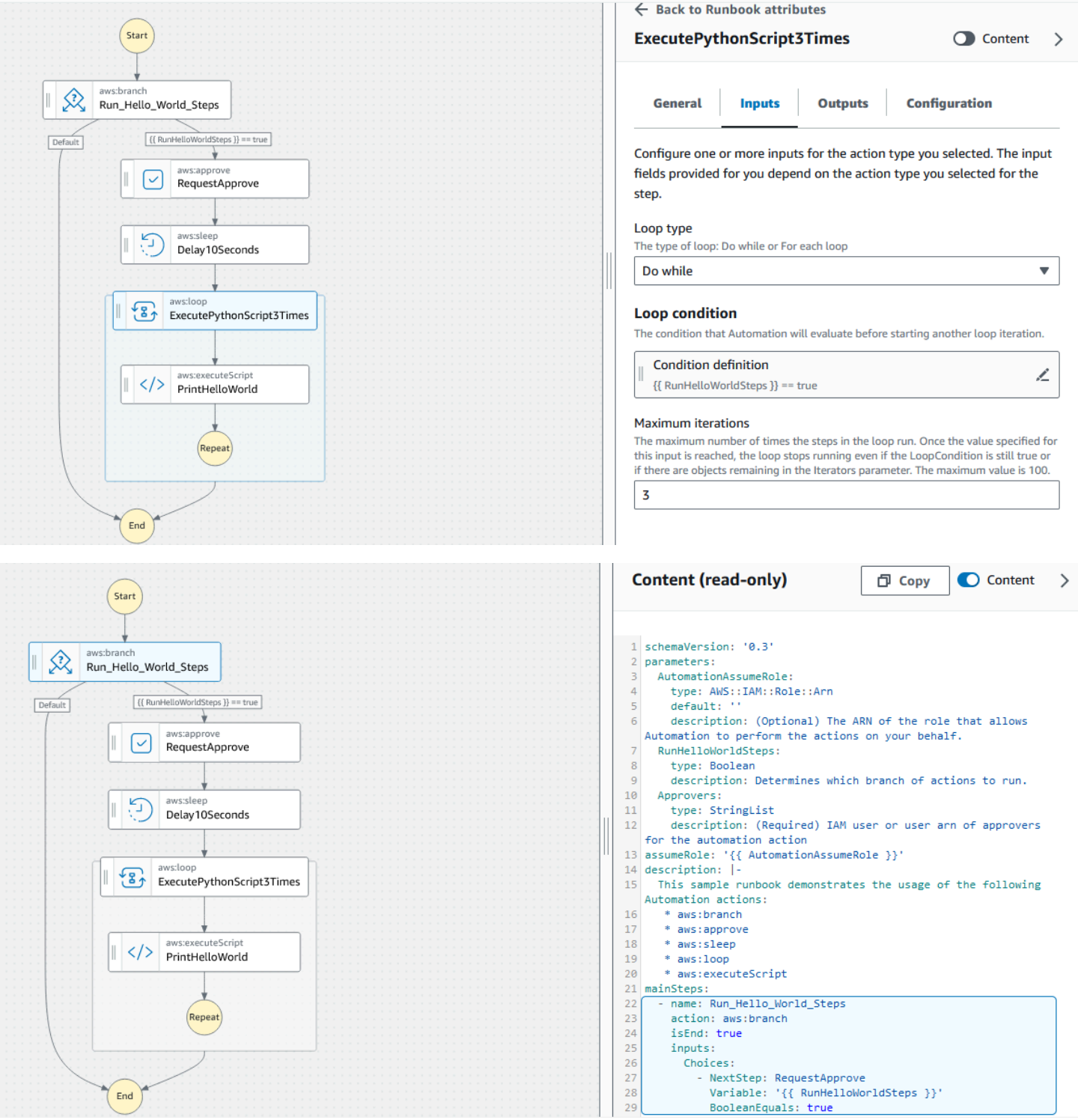
canvas panel. Use the controls at the top of the canvas to zoom in or out. To view different parts of a workflow, you can drag the workflow graph in the canvas.

Drag an action from the **Actions** browser, and drop it into your runbook's workflow graph. A line shows where it will be placed in your workflow. To change the order of an action, you can drag it to a different place in your workflow. The new action has been added to your workflow, and its code is auto-generated.



Form

After you add an action to your runbook workflow, you can configure it to meet your use case. Choose the action that you want to configure, and you will see its parameters and options in the **Form** panel. You can also see the YAML or JSON code by choosing the **Content** toggle. The code associated with the action you have selected is highlighted.



Keyboard shortcuts

The visual design experience supports the keyboard shortcuts shown in the following table.

Keyboard shortcut	
Ctrl+Z	Undo the last operation.
Ctrl+Shift+Z	Redo the last operation.
Ctrl+C	Center the workflow in the canvas.
Backspace	Remove all selected states.
Delete	Remove all selected states.
Ctrl+D	Duplicate the selected state.

Using the visual design experience

Learn to create, edit and run runbook workflows using the visual design experience. After your workflow is ready, you can save it or export it. You can also use the visual design experience for rapid prototyping.

Create a runbook workflow

1. Sign in to the Systems Manager Automation console.
2. Choose **Create runbook**.
3. In the **Name** box, enter a name for your runbook, for example, *MyNewRunbook*.
4. Next to the **Design** and **Code** toggle, select the pencil icon and enter a name for your runbook.

You can now design a workflow for your new runbook.

Design a runbook

To design a runbook workflow using the visual design experience, you drag an automation action from the **Actions** browser into the canvas, placing it where you want it in your runbook's workflow. You can also re-order actions in your workflow by dragging them to a different location. As you drag an action onto the canvas, a line appears wherever you can drop the action in your workflow. After an action is dropped onto the canvas, its code is auto-generated and added inside your runbook's content.

If you know the name of the action you want to add, use the search box at the top of the **Actions** browser to find the action.

After you drop an action onto the canvas, configure it using the **Form** panel on the right. This panel contains the **General**, **Inputs**, **Outputs**, and **Configuration** tabs for each automation action or API action that you place on the canvas. For example, the **General** tab consists of the following sections:

- The **Step name** identifies the step. Specify a unique value for the step name.
- The **Description** helps you describe what the action is doing in your runbook's workflow.

The **Inputs** tab contains fields that vary based on the action. For example, the `aws:executeScript` automation action consists of the following sections:

- The **Runtime** is the language to use for running the provided script.
- The **Handler** is the name of your function. You must ensure that the function defined in the handler has two parameters: `events` and `context`. The PowerShell runtime doesn't support this parameter.
- The **Script** is an embedded script that you want to run during the workflow.
- (Optional) The **Attachment** is for standalone scripts or .zip files that can be invoked by the action. This parameter is required for JSON runbooks.

The **Outputs** tab helps you specify the values that you want to output from an action. You can reference output values in later actions of your workflow, or generate output from actions for logging purposes. Not all actions will have an **Outputs** tab because not all actions support outputs. For example, the `aws:pause` action doesn't support outputs. For actions that do support outputs, the **Outputs** tab consists of the following sections:

- The **Name** is the name to be used for the output value. You can reference outputs in later actions of your workflow.
- The **Selector** is a JSONPath expression string beginning with `"$."` that is used to select one or more components within a JSON element.
- The **Type** is the data type for the output value. For example, a `String` or `Integer` data type.

The **Configuration** tab contains properties and options that all automation actions can use. The action consists of the following sections:

- The **Max attempts** property is the number of times an action retries if it fails.
- The **Timeout seconds** property specifies the timeout value for an action.
- The **Is critical** property determines if the action failure stops the entire automation.
- The **Next step** property determines which action the automation goes to next in the runbook.
- The **On failure** property determines which action the automation goes to next in the runbook if the action fails.
- The **On cancel** property determines which action the automation goes to next in the runbook if the action is canceled by a user.

To delete an action, you can use backspace, the toolbar above the canvas, or right-click and choose **Delete action**.

As your workflow grows, it might not fit in the canvas. To help make the workflow fit in the canvas, try one of the following options:

- Use the controls on the side panels to resize or close the panels.
- Use the toolbar at the top of the canvas to zoom the workflow graph in or out.

Update your runbook

You can update an existing runbook workflow by creating a new version of your runbook. Updates to your runbooks can be made by using the visual design experience, or by editing the code directly. To update an existing runbook, use the following procedure:

1. Sign in to the Systems Manager Automation console.
2. Choose the runbook that you want to update.
3. Choose **Create new version**.
4. The visual design experience has two panes: A code pane and a visual workflow pane. Choose **Design** in the visual workflow pane to edit your workflow with the visual design experience. When you're done, choose **Create new version** to save your changes and exit.
5. (Optional) Use the code pane to edit the runbook content in YAML or JSON.

Export your runbook

To export your runbook's workflow YAML or JSON code, and also a graph of your workflow, use the following procedure:

1. Choose your runbook in the **Documents** console.
2. Choose **Create new version**.
3. In the **Actions** dropdown, choose whether you want to export the graph or runbook, and which format you prefer.

Configuring inputs and outputs for your actions

Each automation action responds based on input that it receives. In most cases, you then pass output to the subsequent actions. In the visual design experience, you can configure an action's input and output data in the **Inputs** and **Outputs** tabs of the **Form** panel.

For detailed information about how to define and use output for automation actions, see Using action outputs as inputs.

Provide input data for an action

Each automation action has one or more inputs that you must provide a value for. The value you provide for an action's input is determined by the data type and format that's accepted by the action. For example, the `aws:sleep` actions requires an ISO 8601 formatted string value for the `Duration` input.

Generally, you use actions in your runbook's workflow that return output that you want to use in subsequent actions. It's important to make sure your input values are correct to avoid errors in your runbook's workflow. Input values are also important because they determine whether the action returns the expected output. For example, when using the `aws:executeAwsApi` action, you want to make sure that you're providing the right value for the API operation.

Define output data for an action

Some automation actions return output after performing their defined operations. Actions that return output either have predefined outputs, or allow you to define the outputs yourself. For example, the `aws:createImage` action has predefined outputs that return an `ImageId` and `ImageState`. Comparatively, with the `aws:executeAwsApi` action, you can define the outputs you that want from the specified API operation. As a result, you can return one or more values from a single API operation to use in subsequent actions.

Defining your own outputs for an automation action requires that you specify a name of the output, the data type, and the output value. To continue using the `aws:executeAwsApi` action as an example, let's say you're calling the `DescribeInstances` API operation from Amazon EC2. In this example, you want to return, or output, the `State` of an Amazon EC2 instance and branch your runbook's workflow based on the output. You choose to name the output **`InstanceState`**, and use the **`String`** data type.

The process to define the actual value of the output differs, depending on the action. For example, if you're using the `aws:executeScript` action, you must use `return` statements in your functions to provide data to your outputs. With other actions like `aws:executeAwsApi`, `aws:waitForAwsResourceProperty`, and `aws:assertAwsResourceProperty`, a `Selector` is required. The `Selector`, or `PropertySelector` as some actions refer to it, is a JSONPath string that is used to process the JSON response from an API operation. It's important to understand how the JSON response object from an API operation is structured so you can select the correct

value for your output. Using the `DescribeInstances` API operation mentioned earlier, see the following example JSON response:


```
{
  "reservationSet": {
    "item": {
      "reservationId": "r-1234567890abcdef0",
      "ownerId": 123456789012,
      "groupSet": "",
      "instancesSet": {
        "item": {
          "instanceId": "i-1234567890abcdef0",
          "imageId": "ami-bff32ccc",
          "instanceState": {
            "code": 16,
            "name": "running"
          },
          "privateDnsName": "ip-192-168-1-88.eu-west-1.compute.internal",
          "dnsName": "ec2-54-194-252-215.eu-west-1.compute.amazonaws.com",
          "reason": "",
          "keyName": "my_keypair",
          "amiLaunchIndex": 0,
          "productCodes": "",
          "instanceType": "t2.micro",
          "launchTime": "2018-05-08T16:46:19.000Z",
          "placement": {
            "availabilityZone": "eu-west-1c",
            "groupName": "",
            "tenancy": "default"
          },
          "monitoring": {
            "state": "disabled"
          },
          "subnetId": "subnet-56f5f000",
          "vpcId": "vpc-11112222",
          "privateIpAddress": "192.168.1.88",
          "ipAddress": "54.194.252.215",
          "sourceDestCheck": true,
          "groupSet": {
            "item": {
              "groupId": "sg-e4076000",
              "groupName": "SecurityGroup1"
            }
```

```
        },
        "architecture": "x86_64",
        "rootDeviceType": "ebs",
        "rootDeviceName": "/dev/xvda",
        "blockDeviceMapping": {
          "item": {
            "deviceName": "/dev/xvda",
            "ebs": {
              "volumeId": "vol-1234567890abcdef0",
              "status": "attached",
              "attachTime": "2015-12-22T10:44:09.000Z",
              "deleteOnTermination": true
            }
          }
        },
        "virtualizationType": "hvm",
        "clientToken": "xMcwG14507example",
        "tagSet": {
          "item": {
            "key": "Name",
            "value": "Server_1"
          }
        },
        "hypervisor": "xen",
        "networkInterfaceSet": {
          "item": {
            "networkInterfaceId": "eni-551ba000",
            "subnetId": "subnet-56f5f000",
            "vpcId": "vpc-11112222",
            "description": "Primary network interface",
            "ownerId": 123456789012,
            "status": "in-use",
            "macAddress": "02:dd:2c:5e:01:69",
            "privateIpAddress": "192.168.1.88",
            "privateDnsName": "ip-192-168-1-88.eu-west-1.compute.internal",
            "sourceDestCheck": true,
            "groupSet": {
              "item": {
                "groupId": "sg-e4076000",
                "groupName": "SecurityGroup1"
              }
            },
            "attachment": {
              "attachmentId": "eni-attach-39697adc",
```

```
                "deviceIndex": 0,
                "status": "attached",
                "attachTime": "2018-05-08T16:46:19.000Z",
                "deleteOnTermination": true
              },
              "association": {
                "publicIp": "54.194.252.215",
                "publicDnsName": "ec2-54-194-252-215.eu-west-1.compute.amazonaws.com",
                "ipOwnerId": "amazon"
              },
              "privateIpAddressesSet": {
                "item": {
                  "privateIpAddress": "192.168.1.88",
                  "privateDnsName": "ip-192-168-1-88.eu-west-1.compute.internal",
                  "primary": true,
                  "association": {
                    "publicIp": "54.194.252.215",
                    "publicDnsName": "ec2-54-194-252-215.eu-
west-1.compute.amazonaws.com",
                    "ipOwnerId": "amazon"
                  }
                }
              },
              "ipv6AddressesSet": {
                "item": {
                  "ipv6Address": "2001:db8:1234:1a2b::123"
                }
              }
            }
          },
          "iamInstanceProfile": {
            "arn": "arn:aws:iam::123456789012:instance-profile/AdminRole",
            "id": "ABCAJEDNCAA64SSD123AB"
          },
          "ebsOptimized": false,
          "cpuOptions": {
            "coreCount": 1,
            "threadsPerCore": 1
          }
        }
      }
    }
  }
```

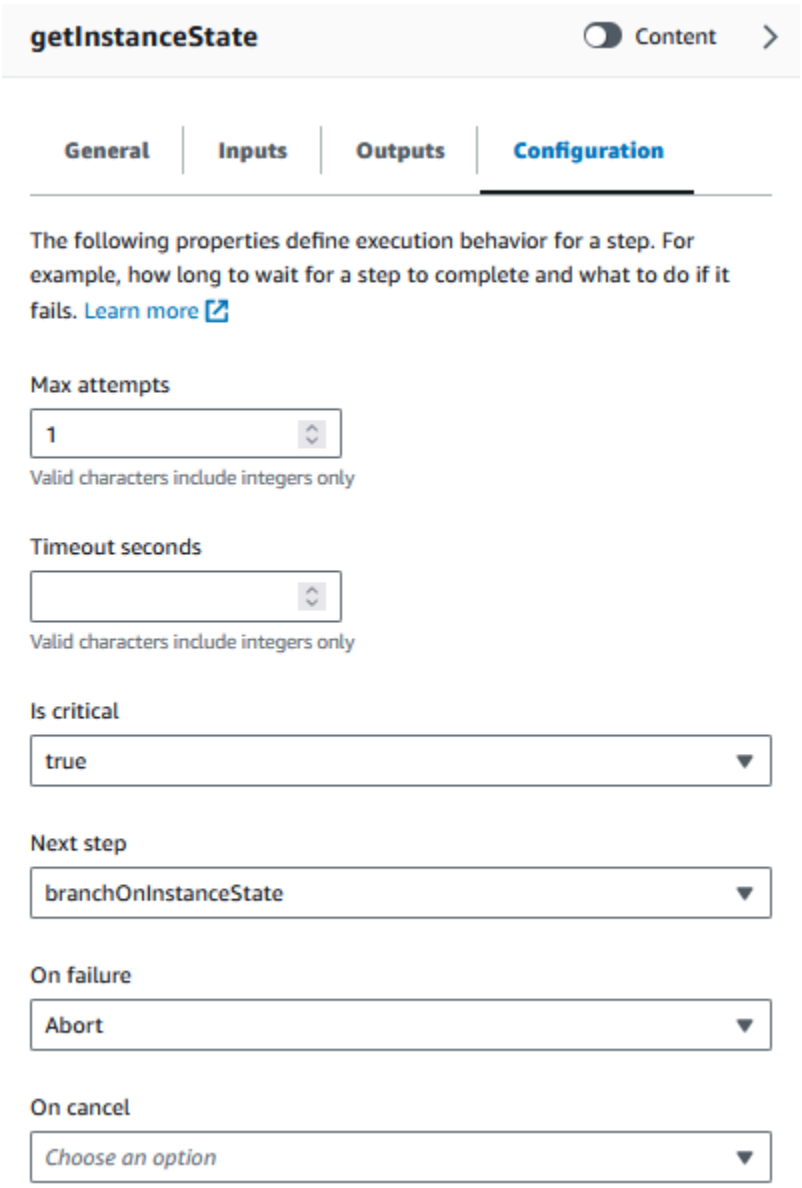
```
}
```

In the JSON response object, the instance `State` is nested in an `Instances` object, which is nested in the `Reservations` object. To return the value of the instance `State`, use the following string for the `Selector` so the value can be used in our output: **`$.Reservations[0].Instances[0].State.Name`**.

To reference an output value in subsequent actions of your runbook's workflow, the following format is used: `{{ StepName.NameOfOutput }}`. For example, **`{{ GetInstanceState.InstanceState }}`**. In the visual design experience, you can choose output values to use in subsequent actions using the dropdown for the input. When using outputs in subsequent actions, the data type of the output must match the data type for the input. In this example, the `InstanceState` output is a `String`. Therefore, to use the value in a subsequent action's input, the input must accept a `String`.

Error handling with the visual design experience

By default, when an action reports an error, Automation stops the runbook's workflow entirely. This is because the default value for the `onFailure` property on all actions is `Abort`. You can configure how Automation handles errors in your runbook's workflow. Even if you have configured error handling, some errors might still cause an automation to fail. For more information, see Troubleshooting Systems Manager Automation. In the visual design experience, you configure error handling in the **Configuration** panel.



Retry action on error

To retry an action in case of an error, specify a value for the **Max attempts** property. The default value is 1. If you specify a value greater than 1, the action isn't considered to have failed until all of the retry attempts have failed.

Timeouts

You can configure a timeout for actions to set the maximum number of seconds your action can run before it fails. To configure a timeout, enter the number of seconds that your action should wait before the action fails in the **Timeout seconds** property. If the timeout is reached and the

action has a value of `Max attempts` that is greater than 1, the step isn't considered to have timed out until the retries complete.

Failed actions

By default, when an action fails, Automation stops the runbook's workflow entirely. You can modify this behavior by specifying an alternative value for the **On failure** property of the actions in your runbook. If you want the workflow to continue to the next step in the runbook, choose **Continue**. If you want the workflow to jump to a different subsequent step in the runbook, choose **Step** and then enter the name of the step.

Canceled actions

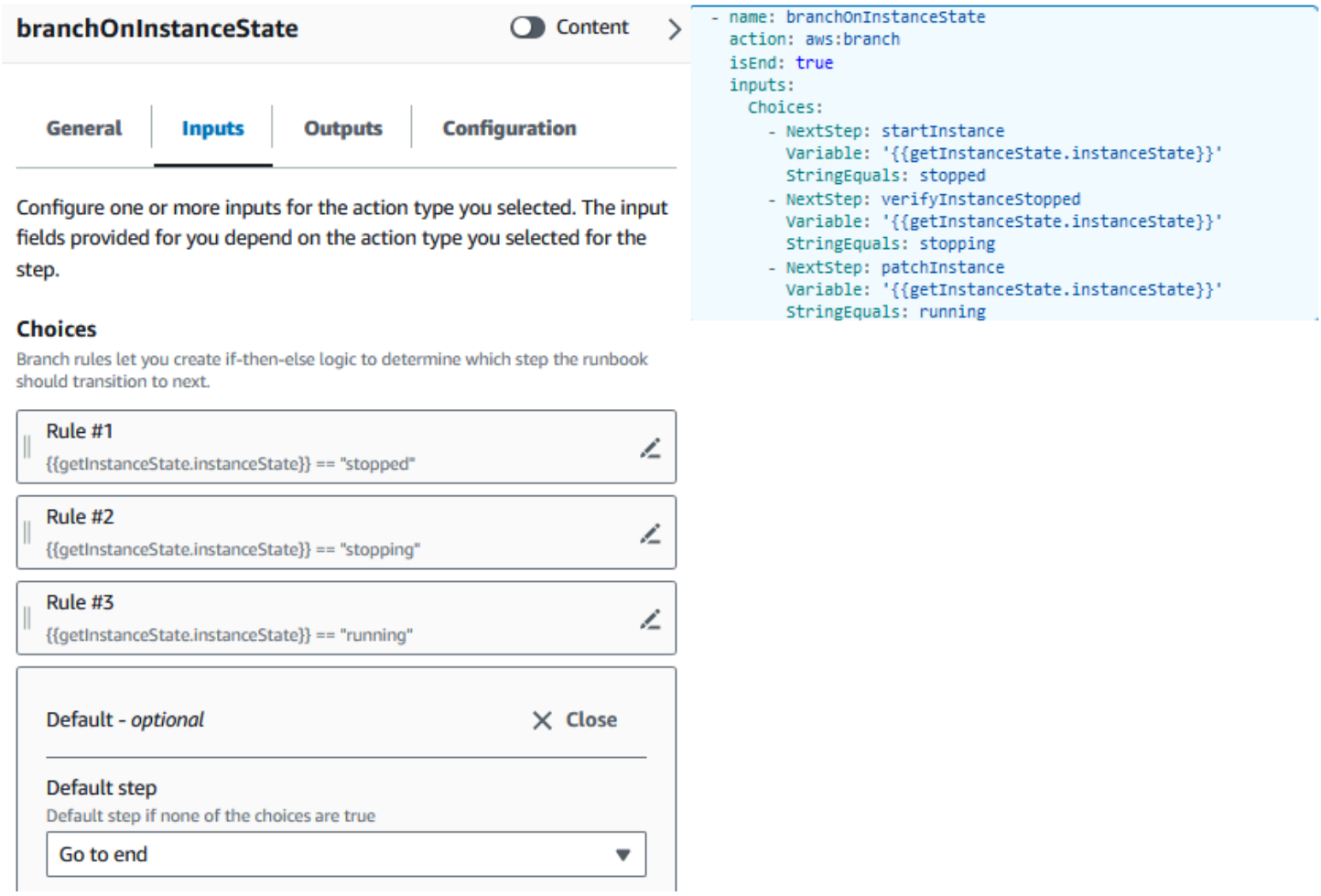
By default, when an action is canceled by a user, Automation stops the runbook's workflow entirely. You can modify this behavior by specifying an alternative value for the **On cancel** property of the actions in your runbook. If you want the workflow to jump to a different subsequent step in the runbook, choose **Step** and then enter the name of the step.

Critical actions

You can designate an action as *critical*, meaning it determines the overall reporting status of your automation. If a step with this designation fails, Automation reports the final status as `Failed` regardless of the success of other actions. To configure an action as critical, leave the default value as **True** for the **Is critical** property.

Ending actions

The **Is end** property stops an automation at the end of the specified action. The default value for this property is `false`. If you configure this property for an action, the automation stops whether the action succeeds or fails. This property is most often used with `aws:branch` actions to handle unexpected or undefined input values. The following example shows a runbook that is expecting an instance state of either `running`, `stopping`, or `stopped`. If an instance is in a different state, the automation ends.



Tutorial: Create a runbook using the visual design experience

In this tutorial, you will learn the basics of working with the visual design experience provided by Systems Manager Automation. In the visual design experience, you can create a runbook that uses multiple actions. You use the drag and drop feature to arrange actions on the canvas. You also search for, select, and configure these actions. Then, you can view the auto-generated YAML code for your runbook's workflow, exit the visual design experience, run the runbook, and review the execution details.

This tutorial also shows you how to update the runbook and view the new version. At the end of the tutorial, you perform a clean-up step and delete your runbook.

After you complete this tutorial, you'll know how to use the visual design experience to create a runbook. You'll also know how to update, run, and delete your runbook.

> **Note**
>
> Before you start this tutorial, make sure to complete Setting up Automation.

Topics

- Step 1: Navigate to the visual design experience
- Step 2: Create a workflow
- Step 3: Review the auto-generated code
- Step 4: Run your new runbook
- Step 5: Clean up

Step 1: Navigate to the visual design experience

1. Sign in to the Systems Manager Automation console.
2. Choose **Create automation runbook**.

Step 2: Create a workflow

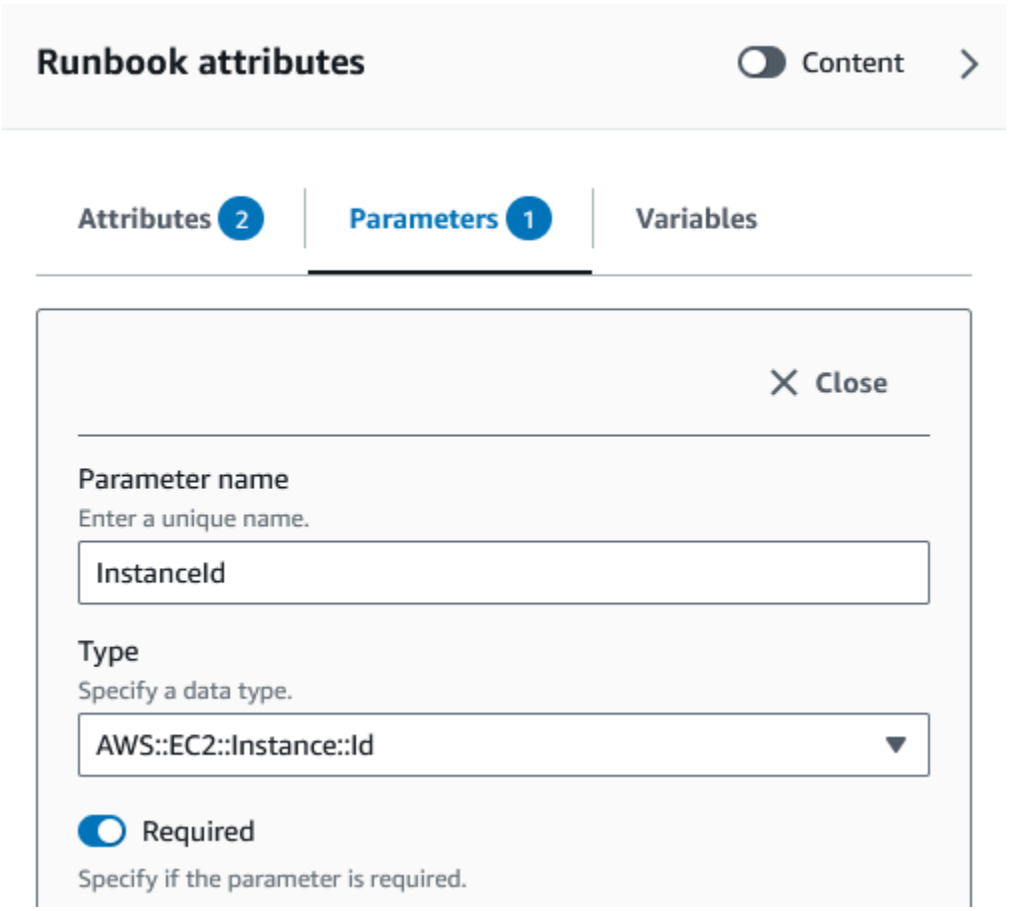
In the visual design experience, a workflow is a graphical representation of your runbook on the canvas. You can use the visual design experience to define, configure, and examine the individual actions of your runbook.

To create a workflow

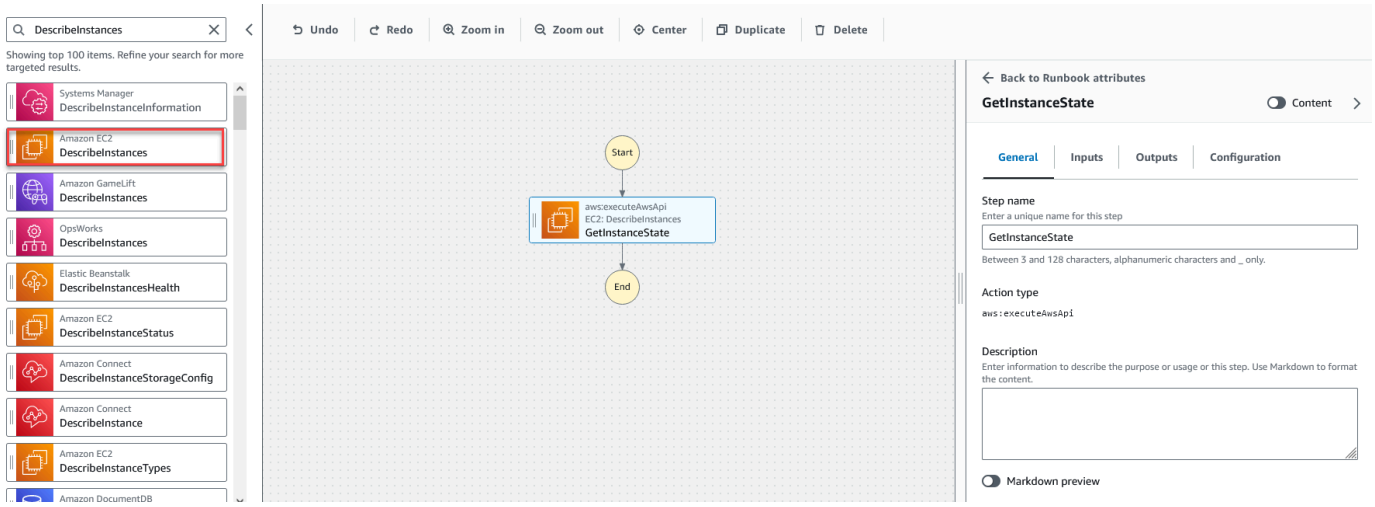
1. Next to the **Design** and **Code** toggle, select the pencil icon and enter a name for your runbook. For this tutorial, enter **`VisualDesignExperienceTutorial`**.



2. In the **Document attributes** section of the **Form** panel, expand the **Input parameters** dropdown, and select **Add a parameter**.
   a. In the **Parameter name** field, enter **`InstanceId`**.
   b. In the **Type** dropdown, choose **AWS::EC2::Instance**.
   c. Select the **Required** toggle.



3. In the **AWS APIs** browser, enter **`DescribeInstances`** in the search bar.
4. Drag an **Amazon EC2 – DescribeInstances** action to the empty canvas.
5. For **Step name**, enter a value. For this tutorial, you can use the name **`GetInstanceState`**.



a. Expand the **Additional inputs** dropdown, and in the **Input name** field, enter **`InstanceIds`**.

b. Choose the **Inputs** tab.

c. In the **Input value** field, choose the **`InstanceId`** document input. This references the value of the input parameter that you created at the beginning of the procedure. Since the **InstanceIds** input for the `DescribeInstances` action accepts `StringList` values, you must wrap the **InstanceId** input in square brackets. The YAML for the **Input value** should match the following: **`['{{ InstanceId }}']`**.

d. In the **Outputs** tab, select **Add an output** and enter **`InstanceState`** in the **Name** field.

e. In the **Selector** field, enter **`$.Reservations[0].Instances[0].State.Name`**.

f. In the **Type** dropdown, choose **String**.

6. Drag a **Branch** action from the **Actions** browser, and drop it below the **`GetInstanceState`** step.

7. For **Step name**, enter a value. For this tutorial, use the name **`BranchOnInstanceState`**.

   To define the branching logic, do the following:

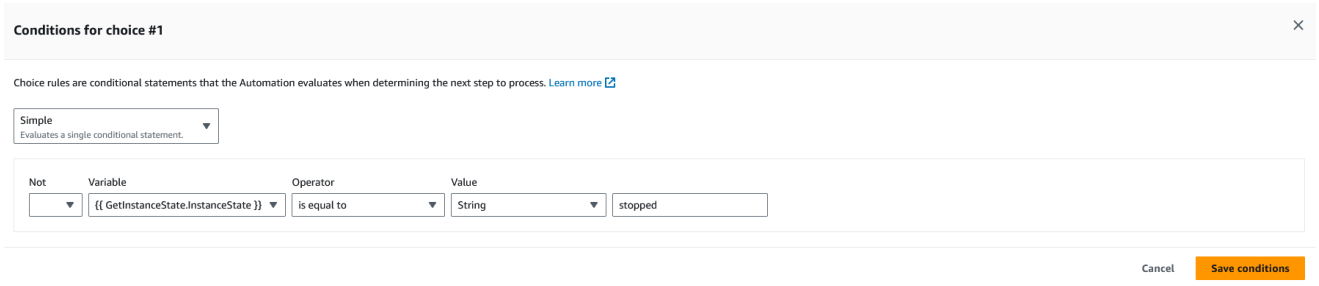
   a. Choose the **`Branch`** state on the canvas. Then, under **Inputs** and **Choices**, select the pencil icon to edit **Rule #1**.

   b. Choose **Add conditions**.

   c. In the **Conditions for rule #1** dialog box, choose the **`GetInstanceState.InstanceState`** step output from the **Variable** dropdown.

   d. For **Operator**, choose **is equal to**.

   e. For **Value**, choose **String** from the dropdown list. Enter **`stopped`**.



   f. Select **Save conditions**.

   g. Choose **Add new choice rule**.

   h. Choose **Add conditions** for **Rule #2**.

i. In the **Conditions for rule #2** dialog box, choose the **`GetInstanceState.InstanceState`** step output from the **Variable** dropdown.

j. For **Operator**, choose **is equal to**.

k. For **Value**, choose **String** from the dropdown list. Enter **`stopping`**.

l. Select **Save conditions**.

m. Choose **Add new choice rule**.

n. For **Rule #3**, choose **Add conditions**.

o. In the **Conditions for rule #3** dialog box, choose the **`GetInstanceState.InstanceState`** step output from the **Variable** dropdown.

p. For **Operator**, choose **is equal to**.

q. For **Value**, choose **String** from the dropdown list. Enter **`running`**.

r. Select **Save conditions**.

s. In the **Default rule**, choose **Go to end** for the **Default step**.

8. Drag a **Change an instance state** action to the empty **Drag action here** box under the **{{ GetInstanceState.InstanceState }} == "stopped"** condition.

   a. For the **Step name**, enter **`StartInstance`**.

   b. In the **Inputs** tab, under **Instance IDs**, choose the **InstanceId** document input value from the dropdown.

   c. For the **Desired state**, specify **`running`**.

9. Drag a **Wait on AWS resource** action to the empty **Drag action here** box under the **{{ GetInstanceState.InstanceState }} == "stopping"** condition.

10. For **Step name**, enter a value. For this tutorial, use the name **`WaitForInstanceStop`**.

    a. For the **Service** field, choose **Amazon EC2**.

    b. For the **API** field, choose **DescribeInstances**.

    c. For the **Property selector** field, enter **`$.Reservations[0].Instances[0].State.Name`**.

    d. For the **Desired values** parameter, enter **`["stopped"]`**.

    e. In the **Configuration** tab of the **WaitForInstanceStop** action, choose **StartInstance** from the **Next step** dropdown.

11. Drag a **Run command on instances** action to the empty **Drag action here** box under the **{{ GetInstanceState.InstanceState }} == "running"** condition.

12. For the **Step name**, enter **`SayHello`**.

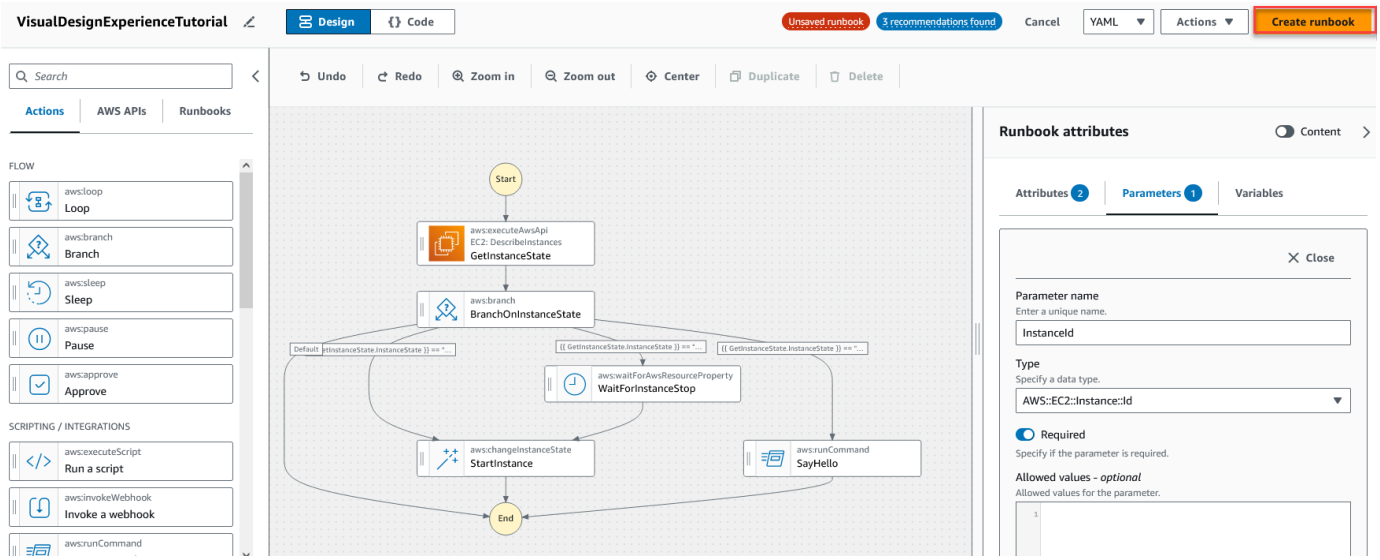
    a. In the **Inputs** tab, enter **`AWS-RunShellScript`** for the **Document name** parameter.

    b. For **InstanceIds**, choose the **InstanceId** document input value from the dropdown.

    c. Expand the **Additional inputs** dropdown, and in the **Input name** dropdown, choose **Parameters**.

    d. In the **Input value** field, enter **`{"commands": "echo 'Hello World'"}`**.

13. Review the completed runbook in the canvas and select **Create runbook** to save the tutorial runbook.



Step 3: Review the auto-generated code

As you drag and drop actions from the **Actions** browser onto the canvas, the visual design experience automatically composes the YAML or JSON content of your runbook in real-time. You can view and edit this code. To view the auto-generated code, select **Code** for the **Design** and **Code** toggle.

Step 4: Run your new runbook

After creating your runbook, you can run the automation.

To run your new automation 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 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 **Execute automation runbook** 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. (Optional) Choose an Amazon CloudWatch alarm to apply to your automation for monitoring. To attach a CloudWatch alarm to your automation, the IAM principal that starts the automation must have permission for the `iam:createServiceLinkedRole` action. For more information about CloudWatch alarms, see Using Amazon CloudWatch alarms. If your alarm activates, the automation is stopped. If you use AWS CloudTrail, you will see the API call in your trail.
9. Choose **Execute**.

Step 5: Clean up

To delete your runbook

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Documents**.
3. Select the **Owned by me** tab.
4. Locate the **VisualDesignExperienceTutorial** runbook.
5. Select the button on the document card page, and then choose **Delete document** from the **Actions** dropdown.

Authoring Automation runbooks

Each runbook in Automation, a tool in 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 the visual designer, or learning more about either markup language before attempting to author your own runbook. For more information about the visual designer, see Visual design experience for Automation runbooks.

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. 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 Additional runbook examples.

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” – 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” – 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 Run automated operations at scale. For more information about the automation actions used in this example, see the Systems Manager Automation actions reference.

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"
     }
   }
},
```

2. 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
        - NextStep: verifyInstanceStopped
          Variable: '{{getInstanceState.instanceState}}'
          StringEquals: stopping
        - 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}}"
            }
         }
      },
```

4. 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}}"
            ]
         }
      }
   ]
}
```

5. 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.

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."
      }
   }
},
```

2. 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"
         }
      }
   ]
}
```

3. 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.
  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'
```

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.

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, 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']
              originalInstanceStates = {}
              for results in queryResponse:
                  instanceSet = results['Instances']
                  for instance in instanceSet:
                      instanceId = instance['InstanceId']
                      originalInstanceStates[instanceId] = instance['State']
['Name']
              return originalInstanceStates
    outputs:
      - Name: originalInstanceStates
        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"
      },
```

3. 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"
      },
```

4. 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.originalInstanceStates}}"
```

```
            },
            "Script":"..."
         },
         "nextStep":"returnPrimaryTagKey"
      },
```

5. 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":{
               "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"
      },
```

6. 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.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
    nextStep: getSecondaryInstanceState
```

JSON

```
{
         "name":"returnPrimaryToOriginalState",
         "action":"aws:executeScript",
         "timeoutSeconds":600,
         "onFailure":"Abort",
         "inputs":{
            "Runtime":"python3.7",
            "Handler":"returnToOriginalState",
            "InputPayload":{

 "targetInstances":"{{getPrimaryInstanceState.originalInstanceStates}}"
            },
            "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']
              originalInstanceStates = {}
              for results in queryResponse:
                  instanceSet = results['Instances']
                  for instance in instanceSet:
                      instanceId = instance['InstanceId']
                      originalInstanceStates[instanceId] = instance['State']
['Name']
              return originalInstanceStates
    outputs:
      - Name: originalInstanceStates
```

```
        Selector: $.Payload
        Type: StringMap
    nextStep: verifySecondaryInstancesRunning
  - name: verifySecondaryInstancesRunning
    action: 'aws:executeScript'
    timeoutSeconds: 600
    onFailure: Abort
    inputs:
      Runtime: python3.7
      Handler: verifyInstancesRunning
      InputPayload:
        targetInstances: '{{getSecondaryInstanceState.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: waitForSecondaryRunningInstances
  - name: waitForSecondaryRunningInstances
    action: 'aws:executeScript'
    timeoutSeconds: 300
    onFailure: Abort
```

```
    inputs:
      Runtime: python3.7
      Handler: waitForRunningInstances
      InputPayload:
        targetInstances: '{{getSecondaryInstanceState.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: 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

```
{
         "name":"getSecondaryInstanceState",
         "action":"aws:executeScript",
         "timeoutSeconds":120,
         "onFailure":"Abort",
         "inputs":{
            "Runtime":"python3.7",
            "Handler":"getInstanceStates",
            "InputPayload":{
               "secondaryTag":"{{SecondaryPatchGroupTag}}"
            },
            "Script":"..."
         },
         "outputs":[
            {
               "Name":"originalInstanceStates",
               "Selector":"$.Payload",
               "Type":"StringMap"
            }
         ],
         "nextStep":"verifySecondaryInstancesRunning"
      },
      {
```

```
        "name":"verifySecondaryInstancesRunning",
        "action":"aws:executeScript",
        "timeoutSeconds":600,
        "onFailure":"Abort",
        "inputs":{
           "Runtime":"python3.7",
           "Handler":"verifyInstancesRunning",
           "InputPayload":{

"targetInstances":"{{getSecondaryInstanceState.originalInstanceStates}}"
           },
           "Script":"..."
        },
        "nextStep":"waitForSecondaryRunningInstances"
     },
     {
        "name":"waitForSecondaryRunningInstances",
        "action":"aws:executeScript",
        "timeoutSeconds":300,
        "onFailure":"Abort",
        "inputs":{
           "Runtime":"python3.7",
           "Handler":"waitForRunningInstances",
           "InputPayload":{

"targetInstances":"{{getSecondaryInstanceState.originalInstanceStates}}"
           },
           "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.originalInstanceStates}}"
            },
            "Script":"..."
         }
      }
   ]
}
```

8. 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']
              originalInstanceStates = {}
              for results in queryResponse:
                  instanceSet = results['Instances']
                  for instance in instanceSet:
                      instanceId = instance['InstanceId']
                      originalInstanceStates[instanceId] = instance['State']
['Name']
              return originalInstanceStates
    outputs:
      - Name: originalInstanceStates
        Selector: $.Payload
```

```
        Type: StringMap
    nextStep: verifyPrimaryInstancesRunning
  - 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
  - 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
  - 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.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
    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']
              originalInstanceStates = {}
              for results in queryResponse:
                  instanceSet = results['Instances']
                  for instance in instanceSet:
                      instanceId = instance['InstanceId']
                      originalInstanceStates[instanceId] = instance['State']
['Name']
              return originalInstanceStates
    outputs:
      - Name: originalInstanceStates
        Selector: $.Payload
        Type: StringMap
    nextStep: verifySecondaryInstancesRunning
  - name: verifySecondaryInstancesRunning
    action: 'aws:executeScript'
    timeoutSeconds: 600
    onFailure: Abort
    inputs:
      Runtime: python3.7
      Handler: verifyInstancesRunning
      InputPayload:
        targetInstances: '{{getSecondaryInstanceState.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: waitForSecondaryRunningInstances
- name: waitForSecondaryRunningInstances
  action: 'aws:executeScript'
  timeoutSeconds: 300
  onFailure: Abort
  inputs:
    Runtime: python3.7
    Handler: waitForRunningInstances
    InputPayload:
      targetInstances: '{{getSecondaryInstanceState.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: 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":"originalInstanceStates",
               "Selector":"$.Payload",
               "Type":"StringMap"
            }
         ],
         "nextStep":"verifyPrimaryInstancesRunning"
      },
      {
         "name":"verifyPrimaryInstancesRunning",
         "action":"aws:executeScript",
         "timeoutSeconds":600,
         "onFailure":"Abort",
         "inputs":{
            "Runtime":"python3.7",
            "Handler":"verifyInstancesRunning",
            "InputPayload":{

 "targetInstances":"{{getPrimaryInstanceState.originalInstanceStates}}"
            },
            "Script":"..."
         },
         "nextStep":"waitForPrimaryRunningInstances"
      },
      {
         "name":"waitForPrimaryRunningInstances",
         "action":"aws:executeScript",
         "timeoutSeconds":300,
         "onFailure":"Abort",
         "inputs":{
```

```
            "Runtime":"python3.7",
            "Handler":"waitForRunningInstances",
            "InputPayload":{

"targetInstances":"{{getPrimaryInstanceState.originalInstanceStates}}"
            },
            "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.originalInstanceStates}}"
            },
            "Script":"..."
         },
         "nextStep":"getSecondaryInstanceState"
      },
      {
         "name":"getSecondaryInstanceState",
         "action":"aws:executeScript",
         "timeoutSeconds":120,
         "onFailure":"Abort",
         "inputs":{
            "Runtime":"python3.7",
            "Handler":"getInstanceStates",
            "InputPayload":{
               "secondaryTag":"{{SecondaryPatchGroupTag}}"
            },
            "Script":"..."
         },
         "outputs":[
            {
               "Name":"originalInstanceStates",
               "Selector":"$.Payload",
               "Type":"StringMap"
            }
         ],
         "nextStep":"verifySecondaryInstancesRunning"
      },
      {
         "name":"verifySecondaryInstancesRunning",
         "action":"aws:executeScript",
         "timeoutSeconds":600,
         "onFailure":"Abort",
```

```
        "inputs":{
           "Runtime":"python3.7",
           "Handler":"verifyInstancesRunning",
           "InputPayload":{

"targetInstances":"{{getSecondaryInstanceState.originalInstanceStates}}"
           },
           "Script":"..."
        },
        "nextStep":"waitForSecondaryRunningInstances"
     },
     {
        "name":"waitForSecondaryRunningInstances",
        "action":"aws:executeScript",
        "timeoutSeconds":300,
        "onFailure":"Abort",
        "inputs":{
           "Runtime":"python3.7",
           "Handler":"waitForRunningInstances",
           "InputPayload":{

"targetInstances":"{{getSecondaryInstanceState.originalInstanceStates}}"
           },
           "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.originalInstanceStates}}"
            },
            "Script":"..."
         }
      }
   ]
}
```

For more information about the automation actions used in this example, see the Systems Manager Automation actions reference.

Additional runbook examples

The following example runbook demonstrate how you can use AWS Systems Manager automation actions to automate common deployment, troubleshooting, and maintenance tasks.

> **Note**
>
> The example runbooks 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.

Examples

- Deploy VPC architecture and Microsoft Active Directory domain controllers
- Restore a root volume from the latest snapshot
- Create an AMI and cross-Region copy

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 tool in AWS Systems Manager.

The following example AWS Systems Manager runbook performs these actions:

- Retrieves the latest Windows Server 2016 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 tool in 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 Example
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-2016-English-Full-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: '$.InstanceStatuses..InstanceState.Name'
          DesiredValues:
            - running
        nextStep: verifyInstancesOnlineSSM
      - name: verifyInstancesOnlineSSM
        action: aws:waitForAwsResourceProperty
        timeoutSeconds: 600
        inputs:
          Service: ssm
          Api: DescribeInstanceInformation
          InstanceInformationFilterList:
            - key: InstanceIds
              valueSet:
                - '{{ launchDC1.pdcInstanceId }}'
                - '{{ launchDC2.adcInstanceId }}'
          PropertySelector: '$.InstanceInformationList..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 Example",
      "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-2016-English-
Full-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",
            "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 tool in AWS Systems Manager.

The following example 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 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 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: python3.8
        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 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": "$.Reservations[0].Instances[0].RootDeviceName",
               "Type": "String"
            }
         ],
         "nextStep": "getRootVolumeId"
      },
      {
         "name": "getRootVolumeId",
         "action": "aws:executeAwsApi",
         "onFailure": "Abort",
         "inputs": {
            "Service": "ec2",
            "Api": "DescribeVolumes",
            "Filters": [
               {
                  "Name": "attachment.device",
                  "Values": [
                     "{{ getInstanceDetails.rootDeviceName }}"
                  ]
               },
               {
                  "Name": "attachment.instance-id",
                  "Values": [
                     "{{ InstanceId }}"
                  ]
               }
            ]
         },
         "outputs": [
            {
               "Name": "rootVolumeId",
               "Selector": "$.Volumes[0].VolumeId",
               "Type": "String"
            }
         ],
         "nextStep": "getSnapshotsByStartTime"
      },
      {
         "name": "getSnapshotsByStartTime",
         "action": "aws:executeScript",
         "timeoutSeconds": 45,
         "onFailure": "Continue",
```

```
            "inputs": {
               "Runtime": "python3.8",
               "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 tool in AWS Systems Manager.

The following example 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 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 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: python3.8
      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 Example",
      "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": "python3.8",
            "Handler": "crossRegionImageCopy",
            "InputPayload": {
               "newImageId": "{{ createImage.newImageId }}"
            },
            "Attachment": "crossRegionImageCopy.py"
```

```
            }
         }
      ],
      "files": {
           "crossRegionImageCopy.py": {
               "checksums": {
                   "sha256": "sampleETagValue"
               }
           }
       }
   }
```

Creating input parameters that populate AWS resources

Automation, a tool in 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}|[a-z0-9]{17})$`
- `List<AWS::EC2::Instance::Id>` - `^m?i-([a-z0-9]{8}|[a-z0-9]{17})$`
- `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
      }
   ]
}
```

Using Document Builder to create runbooks

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 Document Builder in the Systems Manager Automation console to build a custom runbook.

Using 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.

Create a runbook using Document Builder

Before you begin

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.

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**.
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.

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.
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.
    - 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 for the action type you selected. For example: `aws:executeStateMachine` – Run an AWS Step Functions state machine.
    - (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.

> **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.

Create a runbook that runs scripts

The following procedure shows how to use Document Builder in the AWS Systems Manager Automation console to create a custom runbook that runs a script.

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

Make sure you have completed the following steps:

- 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.
- 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.

  For information about the IAM service role requirement for running `aws:executeScript`, see Permissions for using runbooks.
- Verify that you have permission to launch EC2 instances.

  For information, see IAM and Amazon EC2 in the *Amazon EC2 User Guide*.

To create a custom runbook that runs scripts 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**.
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 2023 AMI ID available. | {{ ssm:/
aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-arm64 }}
```

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.

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/al2023-ami-kernel-6.1-arm64 }}
```

> **Note**
>
> This value launches an Amazon EC2 instance using the latest Amazon Linux 2023 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 2023 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: **`WaitForInstanceStatusOk`**.
2. For **Action type**, choose **Run a script** (**`aws:executeScript`**).
3. For **Description**, enter a description for the automation step, such as the following.

```
**About This Step**

   The script continuously polls the instance status check value for the
 instance launched in Step 1 until the ```ok``` status is returned.
```

4. For **Runtime**, choose the runtime language to be used for executing the provided script.
5. For **Handler**, enter **`poll_instance`**. This is the function name declared in the following script.

> **Note**
>
> This is not required for PowerShell.

6. For **Script**, replace the default contents with the following. Be sure to match the script with the corresponding runtime value.

   Python

```
def poll_instance(events, context):
      import boto3
      import time

      ec2 = boto3.client('ec2')

      instance_id = events['InstanceId']

      print('[INFO] Waiting for instance status check to report ok',
 instance_id)

      instance_status = "null"

      while True:
        res = ec2.describe_instance_status(InstanceIds=[instance_id])

        if len(res['InstanceStatuses']) == 0:
          print("Instance status information is not available yet")
          time.sleep(5)
          continue

        instance_status = res['InstanceStatuses'][0]['InstanceStatus']
['Status']

        print('[INFO] Polling to get status of the instance', instance_status)

        if instance_status == 'ok':
          break

        time.sleep(10)

      return {'Status': instance_status, 'InstanceId': instance_id}
```

   PowerShell

```
    Install-Module AWS.Tools.EC2 -Force
```

```
    $inputPayload = $env:InputPayload | ConvertFrom-Json

    $instanceId = $inputPayload.payload.InstanceId

    $status = Get-EC2InstanceStatus -InstanceId $instanceId

    while ($status.Status.Status -ne 'ok'){
       Write-Host 'Polling get status of the instance', $instanceId

       Start-Sleep -Seconds 5

       $status = Get-EC2InstanceStatus -InstanceId $instanceId
    }

    return @{Status = $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.

Using scripts in runbooks

Automation runbooks support running scripts as part of the automation. Automation is a tool in 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.

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 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.

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. For more information about the `aws:executeScript` action and how to use attachments for scripts, see [`aws:executeScript` – Run a script].

Script constraints for runbooks

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.

Using conditional statements in runbooks

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 Example `aws:branch` runbooks.

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, a variable defined in the `Variables` section, or specify an output object from a previous step.

  Specify variable values by using the following form.

  `Variable: "{{variable name}}"`

  Specify parameter values 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.

- **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 conditional 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 on`Failure: 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:
      - msiexec /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:
      - msiexec /i {{packageName}}
  nextStep: TestInstall
- name: TestInstall
  action: aws:invokeLambdaFunction
  maxAttempts: 1
  timeoutSeconds: 500
  inputs:
    FunctionName: TestLambdaFunction
  isEnd: true
- name: PostFailure
  action: aws:invokeLambdaFunction
  maxAttempts: 1
  timeoutSeconds: 500
  inputs:
    FunctionName: PostFailureRecoveryLambdaFunction
  isEnd: true
...
```

Note

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 (on`Failure: 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:
    - msiexec /i {{packageName}}
nextStep: TestPackage
...
```

Using action outputs as inputs

Several automation actions return pre-defined outputs. You can pass these outputs as inputs to later steps in your runbook using the format {{*stepName*.*outputName*}}. You can also define custom outputs for automation actions in your runbooks. This allows you to run scripts, or invoke API operations for other AWS services once so you can reuse the values as inputs in later actions. Parameter types in runbooks are static. This means the parameter type can't be changed after it's defined. To define a step output provide the following fields:

- Name: (Required) The output name which is used to reference the output value in later steps.
- Selector: (Required) The JSONPath expression that is used to determine the output value.
- Type: (Optional) The data type of the value returned by the selector field. Valid type values are `String`, `Integer`, `Boolean`, `StringList`, `StringMap`, `MapList`. The default value is `String`.

If the value of an output doesn't match the data type you've specified, Automation tries to convert the data type. For example, if the value returned is an `Integer`, but the `Type` specified is `String`, the final output value is a `String` value. The following type conversions are supported:

- `String` values can be converted to `StringList`, `Integer` and `Boolean`.
- `Integer` values can be converted to `String` and `StringList`.
- `Boolean` values can be converted to `String` and `StringList`.

- `StringList`, `IntegerList`, or `BooleanList` values containing one element can be converted to `String`, `Integer`, or `Boolean`.

When using parameters or outputs with automation actions, the data type can't be dynamically changed within an action's input.

Here is an example runbook that demonstrates how to define action outputs, and reference the value as input for a later action. The runbooks 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
  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"
...
```

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
- `aws:executeAwsApi` – Call and run AWS API operations
- `aws:waitForAwsResourceProperty` – Wait on an AWS resource property

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 runbooks

A JSONPath expression is a string beginning with "$." that is used to select one of more components within a JSON element. The following list includes information about JSONPath operators that are supported by Systems Manager Automation:

- **Dot-notated child (.)**: Use with a JSON object. This operator selects the value of a specific key.
- **Deep-scan (..)**: Use with a JSON element. This operator scans the JSON element level by level and selects a list of values with the specific key. 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.
- **Filter ([?(*expression*)])**: Use with a JSON array. This operator filters JSON array values that match the criteria defined in the filter expression. Filter expressions can only use the following operators: ==, !=, >, <, >=, or <=. Combining multiple filter expressions with AND (&&) or OR (||) is not supported. The return type of this operator is always a JSON array.

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
```

JSONPath Example 7: Filter for instances in the `running` state

```
JSONPath:
$.Reservations..Instances[?(@.State.Name == 'running')]

Returns:
[
  {
    "ImageId": "ami-12345678",
    "BlockDeviceMappings": [
```

```
      {
        "Ebs": {
          "DeleteOnTermination": true,
          "Status": "attached",
          "VolumeId": "vol-000000000000"
        },
        "DeviceName": "/dev/xvda"
      }
    ],
    "State": {
      "Code": 16,
      "Name": "running"
    }
  }
]

Type: MapList
```

JSONPath Example 8: Return the `ImageId` of instances which aren't in the `running` state

```
JSONPath:
$.Reservations..Instances[?(@.State.Name != 'running')].ImageId

Returns:
[
  "ami-12345678"
]

Type: StringList | String
```

Creating webhook 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. To learn more about the `aws:invokeWebhook` action, see `aws:invokeWebhook` – Invoke an Automation webhook integration.

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 tool in 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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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-SSMParameter `
      -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.

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, Support, and AWS Config. For more information, see Systems Manager Automation Runbook Reference.

Tutorials

The following tutorials help you to use AWS Systems Manager Automation to address common use cases. These tutorials demonstrate how to use your own runbooks, predefined runbooks provided by Automation, and other Systems Manager tools with other AWS services.

Contents

- Updating AMIs
  - Update a Linux AMI
  - Update a Linux AMI (AWS CLI)
  - Update a Windows Server AMI
  - Update a golden AMI using Automation, AWS Lambda, and Parameter Store
    - Task 1: Create a parameter in Systems Manager Parameter Store
    - Task 2: Create an IAM role for AWS Lambda
    - Task 3: Create an AWS Lambda function
    - Task 4: Create a runbook and patch the AMI
  - Updating AMIs using Automation and Jenkins
  - Updating AMIs for Auto Scaling groups
    - Create the PatchAMIAndUpdateASG runbook
- Using AWS Support self-service runbooks
  - Run the EC2Rescue tool on unreachable instances
    - How it works
    - Before you begin
      - Granting AWSSupport-EC2Rescue permissions to perform actions on your instances
        - Granting permissions by using IAM policies
        - Granting permissions by using an AWS CloudFormation template
    - Running the Automation
  - Reset passwords and SSH keys on EC2 instances
    - How it works
    - Before you begin
      - Granting AWSSupport-EC2Rescue permissions to perform actions on your instances
        - Granting permissions by using IAM policies

        - Granting permissions by using an AWS CloudFormation template
    - Running the Automation
- Passing data to Automation using input transformers

Updating AMIs

The following tutorials explain how to update Amazon Machine Image (AMIs) to include the latest patches.

Topics

- Update a Linux AMI
- Update a Linux AMI (AWS CLI)
- Update a Windows Server AMI
- Update a golden AMI using Automation, AWS Lambda, and Parameter Store
- Updating AMIs using Automation and Jenkins
- Updating AMIs for Auto Scaling groups

Update a Linux AMI

This Systems Manager Automation walkthrough shows you how to use the console or AWS CLI and the `AWS-UpdateLinuxAmi` runbook to update a Linux AMI with the latest patches of packages that you specify. Automation is a tool in 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 Server, Red Hat Enterprise Linux (RHEL), or Amazon Linux AMIs. For a full list of supported Linux versions, see Patch Manager prerequisites.

The `AWS-UpdateLinuxAmi` runbook allows 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, or Ubuntu Server Amazon Machine Image (AMI). This is the default runbook behavior.
- Install AWS Systems Manager SSM Agent on an existing image to enable Systems Manager tools, 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. 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 Configure instance permissions required for Systems Manager.

The `AWS-UpdateLinuxAmi` runbook accepts the following input parameters.

Parameter	Type	Description
SourceAmiId	String	(Required) The source AMI ID.
IamInstanceProfileName	String	(Required) The name of the IAM instance profile role you created in Configure instance permissions required for Systems Manager. 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.
AutomationAssumeRole	String	(Required) The name of the IAM service role you created in Setting up Automation. The service role (also called an assume role) gives Automatio

Parameter	Type	Description
		n permission to assume your IAM role and perform actions on your behalf. For example, the service role allows Automation to create a new AMI when running the `aws:createImage` action in 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 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.

Parameter	Type	Description
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 tools 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.

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**.
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.

Update 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 tool in 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 Server, Red Hat Enterprise Linux (RHEL), or Amazon Linux AMIs. For a full list of supported Linux versions, see Patch Manager prerequisites.

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, RHEL, or Ubuntu Server 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. 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 Configure instance permissions required for Systems Manager.

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 *public* parameter. For more information, see Query for the latest Amazon Linux AMI IDs using AWS Systems Manager Parameter Store.
IamInstanceProfileName	String	(Required) The name of the IAM instance profile role you created in Configure instance permissions required for Systems Manager. The instance profile role gives Automation permission to perform actions on your

Parameter	Type	Description
		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 Setting up Automation. The service role (also called an assume role) gives Automation permission to assume your IAM role and perform actions on your behalf. For example, the service role allows Automation to create a new AMI when running the `aws:createImage` action in 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 t2.micro.

Parameter	Type	Description
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 tools 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.

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 Installing or updating the latest version of the AWS CLI.

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.

Update 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 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 tool in AWS Systems Manager. For more information, see Setting up Automation. 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 Configure instance permissions required for Systems Manager.

> **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 Systems Manager Parameter Store *public* parameter. For more information, see Query for the latest Windows AMI IDs using AWS Systems Manager Parameter Store.
SubnetId	String	(Optional) The subnet you want to launch the temporary instance into. You must specify a value for this parameter if you've deleted your default VPC.
IamInstanceProfileName	String	(Required) The name of the IAM instance profile role you created in Configure instance permissions required for Systems Manager. 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

Parameter	Type	Description
		in Setting up Automation. The service role (also called an assume role) gives Automatio n permission to assume your IAM role and perform actions on your behalf. For example, the service role allows Automation to create a new AMI when running the `aws:createImage` action in 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 t2.medium.
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.

Parameter	Type	Description
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-ConfigureAWSPackage` runbook.

Step 6: updateAwsEnaNetworkDriver (`aws:runCommand` action)

This step updates AWS ENA Network drivers by using the `AWS-ConfigureAWSPackage` 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.

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 Installing or updating the latest version of the AWS CLI.

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.

Update a golden AMI using Automation, AWS Lambda, and Parameter Store

The following 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 an 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 tools of AWS Systems Manager.

Before you begin

Configure Automation roles and, optionally, Amazon EventBridge for Automation. For more information, see Setting up Automation.

Contents

- Task 1: Create a parameter in Systems Manager Parameter Store
- Task 2: Create an IAM role for AWS Lambda
- Task 3: Create an AWS Lambda function
- Task 4: Create a runbook and patch the AMI

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**: An AMI ID. For example: `ami-188d6e0e`.

For information about how to create a Parameter Store string parameter, see Creating Parameter Store parameters in Systems Manager.

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 the default contents with the following policy. Replace each *`example resource placeholder`* with your own information.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:us-east-1:111122223333:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
```

```
                "arn:aws:logs:us-east-1:111122223333:log-group:/aws/
lambda/function name:*"
            ]
        }
    ]
}
```

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, such as **amiLambda**.
9. Choose **Create policy**.
10. Repeat steps 2 and 3.
11. Paste the following policy. Replace each *example resource placeholder* with your own information.

    JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:PutParameter",
            "Resource": "arn:aws:ssm:us-
east-1:111122223333:parameter/latestAmi"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeParameters",
            "Resource": "*"
        }
    ]
}
```

12. Choose **Next: Tags**.
13. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this policy.

14. Choose **Next: Review**.
15. On the **Review policy** page, for **Name**, enter a name for the inline policy, such as **`amiParameter`**.
16. Choose **Create policy**.
17. In the navigation pane, choose **Roles**, and then choose **Create role**.
18. Immediately under **Use case**, choose **Lambda**, and then choose **Next**.
19. On the **Add permissions** page, use the **Search** field to locate the two policies you created earlier.
20. Select the check box next to the policies, and then choose **Next**.
21. 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.

22. (Optional) Add one or more tag key-value pairs to organize, track, or control access for this role, and then 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**, enter **`Automation-UpdateSsmParam`**.
5. For **Runtime**, choose **Python 3.8**.
6. For **Architecture**, select the type of computer processor for Lambda to use to run the function, **x86_64** or **arm64**,

7. In the **Permissions** section, expand **Change default execution role**.
8. Choose **Use an existing role**, and then choose the service role for Lambda that you created in Task 2.
9. Choose **Create function**.
10. In the **Code source** 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
        )

    responseString = 'Updated parameter %s with value %s.' %
 (event['parameterName'], event['parameterValue'])

    return responseString
```

11. Choose **File, Save**.
12. To test the Lambda function, from the **Test** menu, choose **Configure test event**.
13. For **Event name**, enter a name for the test event, such as **`MyTestEvent`**.
14. 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"
}
```

15. Choose **Save**.
16. Choose **Test** to test the function. On the **Execution result** tab, the status should be reported as **Succeeded**, along with other details about the update.

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 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**.
3. For **Create document**, choose **Automation**.
4. For **Name**, enter **`UpdateMyLatestWindowsAmi`**.
5. Choose the **Editor** tab, and then choose **Edit**.
6. Choose **OK** when prompted.
7. In the **Document editor** field, replace the default content with the following YAML sample runbook content.

```
---
description: Systems Manager Automation Demo - Patch AMI and Update ASG
schemaVersion: '0.3'
assumeRole: '{{ AutomationAssumeRole }}'
parameters:
  AutomationAssumeRole:
    type: String
    description: '(Required) The ARN of the role that allows Automation to perform
 the actions on your behalf. If no role is specified, Systems Manager Automation
 uses your IAM permissions to execute this document.'
    default: ''
  SourceAMI:
    type: String
    description: The ID of the AMI you want to patch.
    default: '{{ ssm:latestAmi }}'
  SubnetId:
    type: String
    description: The ID of the subnet where the instance from the SourceAMI
 parameter is launched.
  SecurityGroupIds:
    type: StringList
    description: The IDs of the security groups to associate with the instance
 that's launched from the SourceAMI parameter.
  NewAMI:
    type: String
```

```
    description: The name of of newly patched AMI.
    default: 'patchedAMI-{{global:DATE_TIME}}'
  InstanceProfile:
    type: String
    description: The name of the IAM instance profile you want the source instance
 to use.
  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: startInstances
    action: 'aws:runInstances'
    timeoutSeconds: 1200
    maxAttempts: 1
    onFailure: Abort
    inputs:
      ImageId: '{{ SourceAMI }}'
      InstanceType: m5.large
      MinInstanceCount: 1
      MaxInstanceCount: 1
      IamInstanceProfileName: '{{ InstanceProfile }}'
      SubnetId: '{{ SubnetId }}'
      SecurityGroupIds: '{{ SecurityGroupIds }}'
  - name: verifyInstanceManaged
```

```
    action: 'aws:waitForAwsResourceProperty'
    timeoutSeconds: 600
    inputs:
      Service: ssm
      Api: DescribeInstanceInformation
      InstanceInformationFilterList:
        - key: InstanceIds
          valueSet:
            - '{{ startInstances.InstanceIds }}'
      PropertySelector: '$.InstanceInformationList[0].PingStatus'
      DesiredValues:
        - Online
    onFailure: 'step:terminateInstance'
  - name: installPatches
    action: 'aws:runCommand'
    timeoutSeconds: 7200
    onFailure: Abort
    inputs:
      DocumentName: AWS-RunPatchBaseline
      Parameters:
        SnapshotId: '{{SnapshotId}}'
        RebootOption: '{{RebootOption}}'
        Operation: '{{Operation}}'
      InstanceIds:
        - '{{ startInstances.InstanceIds }}'
  - 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: '{{ NewAMI }}'
      NoReboot: false
      ImageDescription: Patched AMI created by 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
```

8. Choose **Create automation**.
9. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
10. In the **Choose document** page, choose the **Owned by me** tab.
11. Search for the **UpdateMyLatestWindowsAmi** runbook, and select the button in the **UpdateMyLatestWindowsAmi** card.
12. Choose **Next**.
13. Choose **Simple execution**.
14. Specify values for the input parameters.
15. Choose **Execute**.
16. 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.

Updating AMIs using Automation and 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 tool in 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 IAM policy that you create in the example and attach to your instance profile.

> **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 Update a golden AMI using Automation, AWS Lambda, and Parameter Store 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.

To create an IAM policy 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.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:StartAutomationExecution",
            "Resource": [
```

```
                "arn:aws:ssm:us-
east-1:111122223333:document/UpdateMyLatestWindowsAmi",
                "arn:aws:ssm:us-east-1:111122223333: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 **Roles**.
9. Choose the instance profile that's attached to your Jenkins server.
10. In the **Permissions** tab, select **Add permissions** and choose **Attach policies**.
11. In the **Other permissions policies** section, enter the name of policy you created in the previous steps. For example, **JenkinsPolicy**.
12. Select the check box next to your policy, and choose **Attach policies**.

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 **Manage nodes and clouds**.
10. In the **Configure Clouds** section, select **Add a new cloud**, and then choose **Amazon EC2**.
11. Enter your information in the remaining fields. Make sure you select the **Use EC2 instance profile to obtain credentials** option.

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 Update a golden AMI using Automation, AWS Lambda, and Parameter Store.

```
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.



6. In the Jenkins project, choose **Build Now**. Jenkins returns output similar to the following example.

```
Console Output

Started by user admin
Building in workspace /var/lib/jenkins/workspace/Build 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
```

Updating AMIs for Auto Scaling groups

The following example updates an Auto Scaling group with a 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 a Python function to create a new launch template that uses the newly patched AMI. Then the Auto Scaling group is updated to use the new launch template. 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 tool in 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.

Create the PatchAMIAndUpdateASG runbook

Use the following procedure to create the **PatchAMIAndUpdateASG** runbook that patches the AMI you specify for the **SourceAMI** parameter. The runbook also updates an 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**.
3. In the **Create document** dropdown, choose **Automation**.
4. In the **Name** field, enter **`PatchAMIAndUpdateASG`**.
5. Choose the **Editor** tab, and choose the **Edit**.
6. Choose **OK** when prompted, and delete the content in the **Document editor** field.
7. In the **Document editor** field, paste the following YAML sample runbook content.

```
---
description: Systems Manager Automation Demo - Patch AMI and Update ASG
schemaVersion: '0.3'
assumeRole: '{{ AutomationAssumeRole }}'
parameters:
  AutomationAssumeRole:
    type: String
    description: '(Required) The ARN of the role that allows Automation to perform
 the actions on your behalf. If no role is specified, Systems Manager Automation
 uses your IAM permissions to execute this document.'
    default: ''
  SourceAMI:
    type: String
    description: '(Required) The ID of the AMI you want to patch.'
  SubnetId:
    type: String
    description: '(Required) The ID of the subnet where the instance from the
 SourceAMI parameter is launched.'
```

```
  SecurityGroupIds:
    type: StringList
    description: '(Required) The IDs of the security groups to associate with the
 instance launched from the SourceAMI parameter.'
  NewAMI:
    type: String
    description: '(Optional) The name of of newly patched AMI.'
    default: 'patchedAMI-{{global:DATE_TIME}}'
  TargetASG:
    type: String
    description: '(Required) The name of the Auto Scaling group you want to
 update.'
  InstanceProfile:
    type: String
    description: '(Required) The name of the IAM instance profile you want the
 source instance to use.'
  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: startInstances
    action: 'aws:runInstances'
    timeoutSeconds: 1200
    maxAttempts: 1
```

```
    onFailure: Abort
    inputs:
      ImageId: '{{ SourceAMI }}'
      InstanceType: m5.large
      MinInstanceCount: 1
      MaxInstanceCount: 1
      IamInstanceProfileName: '{{ InstanceProfile }}'
      SubnetId: '{{ SubnetId }}'
      SecurityGroupIds: '{{ SecurityGroupIds }}'
  - name: verifyInstanceManaged
    action: 'aws:waitForAwsResourceProperty'
    timeoutSeconds: 600
    inputs:
      Service: ssm
      Api: DescribeInstanceInformation
      InstanceInformationFilterList:
        - key: InstanceIds
          valueSet:
            - '{{ startInstances.InstanceIds }}'
      PropertySelector: '$.InstanceInformationList[0].PingStatus'
      DesiredValues:
        - Online
    onFailure: 'step:terminateInstance'
  - name: installPatches
    action: 'aws:runCommand'
    timeoutSeconds: 7200
    onFailure: Abort
    inputs:
      DocumentName: AWS-RunPatchBaseline
      Parameters:
        SnapshotId: '{{SnapshotId}}'
        RebootOption: '{{RebootOption}}'
        Operation: '{{Operation}}'
      InstanceIds:
        - '{{ startInstances.InstanceIds }}'
  - 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: '{{ NewAMI }}'
      NoReboot: false
      ImageDescription: Patched AMI created by Automation
  - name: terminateInstance
    action: 'aws:changeInstanceState'
    maxAttempts: 1
    onFailure: Continue
    inputs:
      InstanceIds:
        - '{{ startInstances.InstanceIds }}'
      DesiredState: terminated
  - name: updateASG
    action: 'aws:executeScript'
    timeoutSeconds: 300
    maxAttempts: 1
    onFailure: Abort
    inputs:
      Runtime: python3.8
      Handler: update_asg
      InputPayload:
        TargetASG: '{{TargetASG}}'
        NewAMI: '{{createImage.ImageId}}'
      Script: |-
        from __future__ import print_function
        import datetime
        import json
        import time
        import boto3

        # create auto scaling and ec2 client
        asg = boto3.client('autoscaling')
        ec2 = boto3.client('ec2')

        def update_asg(event, context):
            print("Received event: " + json.dumps(event, indent=2))

            target_asg = event['TargetASG']
            new_ami = event['NewAMI']
```

```
            # get object for the ASG we're going to update, filter by name of
 target ASG
            asg_query =
 asg.describe_auto_scaling_groups(AutoScalingGroupNames=[target_asg])
            if 'AutoScalingGroups' not in asg_query or not
 asg_query['AutoScalingGroups']:
                return 'No ASG found matching the value you specified.'

            # gets details of an instance from the ASG that we'll use to model the
 new launch template after
            source_instance_id = asg_query.get('AutoScalingGroups')[0]['Instances']
[0]['InstanceId']
            instance_properties = ec2.describe_instances(
                InstanceIds=[source_instance_id]
            )
            source_instance = instance_properties['Reservations'][0]['Instances']
[0]

            # create list of security group IDs
            security_groups = []
            for group in source_instance['SecurityGroups']:
                security_groups.append(group['GroupId'])

            # create a list of dictionary objects for block device mappings
            mappings = []
            for block in source_instance['BlockDeviceMappings']:
                volume_query = ec2.describe_volumes(
                    VolumeIds=[block['Ebs']['VolumeId']]
                )
                volume_details = volume_query['Volumes']
                device_name = block['DeviceName']
                volume_size = volume_details[0]['Size']
                volume_type = volume_details[0]['VolumeType']
                device = {'DeviceName': device_name, 'Ebs': {'VolumeSize':
 volume_size, 'VolumeType': volume_type}}
                mappings.append(device)

            # create new launch template using details returned from instance in
 the ASG and specify the newly patched AMI
            time_stamp = time.time()
            time_stamp_string =
 datetime.datetime.fromtimestamp(time_stamp).strftime('%m-%d-%Y_%H-%M-%S')
            new_template_name = f'{new_ami}_{time_stamp_string}'
            try:
```

```
                ec2.create_launch_template(
                    LaunchTemplateName=new_template_name,
                    LaunchTemplateData={
                        'BlockDeviceMappings': mappings,
                        'ImageId': new_ami,
                        'InstanceType': source_instance['InstanceType'],
                        'IamInstanceProfile': {
                            'Arn': source_instance['IamInstanceProfile']['Arn']
                        },
                        'KeyName': source_instance['KeyName'],
                        'SecurityGroupIds': security_groups
                    }
                )
            except Exception as e:
                return f'Exception caught: {str(e)}'
            else:
                # update ASG to use new launch template
                asg.update_auto_scaling_group(
                    AutoScalingGroupName=target_asg,
                    LaunchTemplate={
                        'LaunchTemplateName': new_template_name
                    }
                )
                return f'Updated ASG {target_asg} with new launch template
 {new_template_name} which uses AMI {new_ami}.'
outputs:
  - createImage.ImageId
```

8. Choose **Create automation**.
9. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
10. In the **Choose document** page, choose the **Owned by me** tab.
11. Search for the **PatchAMIAndUpdateASG** runbook, and select the button in the **PatchAMIAndUpdateASG** card.
12. Choose **Next**.
13. Choose **Simple execution**.
14. Specify values for the input parameters. Be sure the `SubnetId` and `SecurityGroupIds` you specify allow access to the public Systems Manager endpoints, or your interface endpoints for Systems Manager.
15. Choose **Execute**.

16. After automation completes, in the Amazon EC2 console, choose **Auto Scaling**, and then choose **Launch Templates**. Verify that you see the new launch template, and that it uses the new AMI.
17. Choose **Auto Scaling**, and then choose **Auto Scaling Groups**. Verify that the Auto Scaling group uses the new launch template.
18. Terminate one or more instances in your Auto Scaling group. Replacement instances will be launched using the new AMI.

Using AWS Support self-service runbooks

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

- Run the EC2Rescue tool on unreachable instances
- Reset passwords and SSH keys on EC2 instances

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 tool in 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. Instances with encypted root volumes are not supported. 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.

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
- Granting permissions by using an AWS CloudFormation template

Granting permissions by using IAM policies

You can either attach the following IAM policy to your user, group, or role as an inline policy; or, you can create a new IAM managed policy and attach it to your user, group, or role. For more information about adding an inline policy to your user, 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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunction"
            ],
            "Resource": "arn:aws:lambda:*:111122223333:function:AWSSupport-
EC2Rescue-*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::awssupport-ssm.*/*.template",
                "arn:aws:s3:::awssupport-ssm.*/*.zip"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:CreateRole",
                "iam:CreateInstanceProfile",
                "iam:GetRole",
```

```
                "iam:GetInstanceProfile",
                "iam:PutRolePolicy",
                "iam:DetachRolePolicy",
                "iam:AttachRolePolicy",
                "iam:PassRole",
                "iam:AddRoleToInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy",
                "iam:DeleteInstanceProfile"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/AWSSupport-EC2Rescue-*",
                "arn:aws:iam::111122223333: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*",
                "autoscaling:DescribeAutoScalingInstances"
            ],
            "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**. The is the ARN of the AssumeRole. You specify this ARN when you run the Automation in the next procedure, Running the Automation.

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**.
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.medium`.

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.

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 tool in 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*.

Related information

Connect to your Linux instance from Windows using PuTTY in the *Amazon EC2 User Guide*

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.

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
- Granting permissions by using an AWS CloudFormation template

Granting permissions by using IAM policies

You can either attach the following IAM policy to your user, group, or role as an inline policy; or, you can create a new IAM managed policy and attach it to your user, group, or role. For more information about adding an inline policy to your user, 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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunction"
            ],
            "Resource": "arn:aws:lambda:*:111122223333:function:AWSSupport-
EC2Rescue-*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::awssupport-ssm.*/*.template",
                "arn:aws:s3:::awssupport-ssm.*/*.zip"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:CreateRole",
                "iam:CreateInstanceProfile",
                "iam:GetRole",
                "iam:GetInstanceProfile",
                "iam:PutRolePolicy",
                "iam:DetachRolePolicy",
                "iam:AttachRolePolicy",
                "iam:PassRole",
                "iam:AddRoleToInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy",
                "iam:DeleteInstanceProfile"
            ],
            "Resource": [
```

```
                "arn:aws:iam::111122223333:role/AWSSupport-EC2Rescue-*",
                "arn:aws:iam::111122223333: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**. The is the ARN of the AssumeRole. You will specify this ARN when you run the Automation.

Running the Automation

The following procedure describes how to run the `AWSSupport-ResetAccess` 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*.

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**.
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.medium`.

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**.

Passing data to Automation using input transformers

This AWS Systems Manager Automation tutorial 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 tool in 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**.
3. Choose **Create rule**.
4. 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.
5. For **Event bus**, 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 **default**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
6. For **Rule type**, choose **Rule with an event pattern**.
7. Choose **Next**.
8. For **Event source**, choose **AWS events or EventBridge partner events**.
9. In the **Event pattern** section, choose **Event pattern form**.
10. For **Event source**, choose **AWS services**.
11. For **AWS service**, choose **EC2**.
12. For **Event type**, choose **EC2 Instance State-change Notification**.
13. For **Specific state(s)**, choose **stopped**.
14. Choose **Next**.
15. For **Target types**, choose **AWS service**.
16. For **Select a target**, choose **Systems Manager Automation**.

17. For **Document**, choose **AWS-CreateImage**.
18. In the **Configure automation parameter(s)** section, choose **Input Transformer**.
19. For **Input path**, enter **`{"instance":"$.detail.instance-id"}`**.
20. For **Template**, enter **`{"InstanceId":[<instance>]}`**.
21. For **Execution role**, choose **Use existing role** and choose your Automation service role.
22. Choose **Next**.
23. (Optional) Enter one or more tags for the rule. For more information, see Tagging Your Amazon EventBridge Resources in the *Amazon EventBridge User Guide*.
24. Choose **Next**.
25. Review the details of the rule and choose **Create rule**.

Learn about statuses returned by Systems Manager Automation

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 tool in 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-SSMAutomationStepExecution or Get-SSMAutomationExecution.
- Configure Amazon EventBridge to respond to action or automation status changes.

For more information about handling timeouts in an automation, see Handling timeouts in runbooks.

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.
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.
Exited	Only returned by the `aws:loop` action. The loop didn't fully complete. A step inside the loop moved to an outside step using the `nextStep`, `onCancel`, or `onFailure` properties.

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 tool in AWS Systems Manager. This topic includes specific tasks to resolve issues based on Automation error messages.

Topics

- Common Automation errors
- Automation execution failed to start
- Execution started, but status is failed
- Execution started, but timed out

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) 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 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 that was used to start the automation.
- Possible cause 2: The 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 that was used to start the automation.

Access denied due to 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 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 user attempting to start the automation. For more information, see Task 2: Attach the iam:PassRole policy to your Automation role.

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”. Specific details for creating this role are described in the following topic, Task 1: Create a service role for Automation.
- 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 lacks 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: xxxxxx (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 Create the service roles for Automation using the console.

Unexpected State

Error message: `Step fails when it's verifying launched instance(s) are ready to be used. Instance i-xxxxxxxxx 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 services invoked by this action	For information about this service	Troubleshooting content
`aws:runInstances`	Amazon EC2	Amazon EC2 User Guide	Troubleshooting EC2 Instances
`aws:changeInstanceState`	Amazon EC2	Amazon EC2 User Guide	Troubleshooting EC2 instances
`aws:runCommand`	Systems Manager	AWS Systems Manager Run Command	Troubleshooting Systems Manager Run Command

Action	AWS services invoked by this action	For information about this service	Troubleshooting content
aws:createImage	Amazon EC2	Amazon Machine Images	
aws:createStack	AWS CloudFormation	AWS CloudFormation User Guide	Troubleshooting AWS CloudFormation
aws:deleteStack	AWS CloudFormation	AWS CloudFormation User Guide	Troubleshooting AWS CloudFormation
aws:deleteImage	Amazon EC2	Amazon Machines Images	
aws:copyImage	Amazon EC2	Amazon Machine Images	
aws:createTag	Amazon EC2, Systems Manager	EC2 Resource and Tags	
aws:invokeLambdaFunction	AWS Lambda	AWS Lambda Developer Guide	Troubleshooting Lambda

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 tool in AWS Systems Manager, allows you to set up date and time ranges when actions you specify (for example, in Systems Manager Automation 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 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.

You can choose to have all scheduled Automation workflows, maintenance windows, and State Manager associations added automatically to a calendar. You can also remove any of those individual types from the calendar display.

Who should use Change Calendar?

- AWS customers who perform the following action types:

- Create or run Automation runbooks.
- Create change requests in Change Manager.
- Run maintenance windows.
- Create associations in State Manager.

  Automation, Change Manager, Maintenance Windows, and State Manager are all tools AWS Systems Manager. By integrating these tools with Change Calendar, you can allow or block these 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.

> **Note**
>
> The `GetCalendarState` API has a quota of 10 requests per second. For more information about Systems Manager quotas, see Systems Manager service quotas in the *Amazon Web Services General Reference*.

- **EventBridge support**

  This Systems Manager tool is supported as an *event* type in Amazon EventBridge rules. For information, see Monitoring Systems Manager events with Amazon EventBridge and Reference: Amazon EventBridge event patterns and types for Systems Manager.

Topics

- Setting up Change Calendar
- Working with Change Calendar
- Adding Change Calendar dependencies to Automation runbooks
- Troubleshooting Change Calendar

Setting up Change Calendar

Complete the following before using Change Calendar, a tool in 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 Installing, updating, and

Requirement	Description
	uninstalling the AWS CLI in the *AWS Command Line Interface User Guide*.
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 Installing the AWS Tools for PowerShell in the *AWS Tools for PowerShell User Guide*.

Set up permissions

If your user, group, or role is assigned administrator permissions, then you have full access to Change Calendar. If you don't have administrator permissions, then an administrator must give you permission by either assigning the `AmazonSSMFullAccess` managed policy, or assigning a policy that provides the necessary permissions to your user, group, or role.

The following permissions are required to work with Change Calendar.

Change Calendar entries

To create, update, or delete a Change Calendar entry, including adding and removing events from the entry, a policy attached to your user, group, or 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`

Calendar state

To get information about the current or upcoming state of the calendar, a policy attached to your user, group, or role must allow the following action:

- `ssm:GetCalendarState`

Operational events

To view operational events, such as maintenance windows, associations, and planned automations, the policy attached to your user, group, or role must allow the following actions:

- `ssm:DescribeMaintenanceWindows`
- `ssm:DescribeMaintenanceWindowExecution`
- `ssm:DescribeAutomationExecutions`
- `ssm:ListAssociations`

> **Note**
>
> 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. Maintenance windows, State Manager associations, and automations aren't shared.

Working with Change Calendar

You can use the AWS Systems Manager console to add, manage, or delete entries in Change Calendar, a tool in 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
- Creating and managing events in Change Calendar
- Importing and managing events from third-party calendars

- Updating a change calendar
- Sharing a change calendar
- Deleting a change calendar
- Getting the state of a change calendar

Creating a change calendar

When you create an entry in Change Calendar, a tool in 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.
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. (Optional) In **Change management events**, select **Add change management events to the calendar**. This selection displays all scheduled maintenance windows, State Manager associations, Automation workflows, and Change Manager change requests in your monthly calendar display.

   > **Tip**
   >
   > If later you want to permanently remove these events types from the calendar display, edit the calendar, clear this check box, and then choose **Save**.

9. 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. For information about importing events, see Importing events from third-party calendar providers.

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 tool in 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.

Topics

- Creating a Change Calendar event
- Updating a Change Calendar event

- Deleting a Change Calendar event

Creating a Change Calendar event

When you add an event to an entry in Change Calendar, a tool 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 tool in 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 tool in AWS Systems Manager, by using the AWS Management Console.

> **Tip**
>
> If you selected **Add change management events to the calendar** when you created the calendar, you can do the following:
>
> - To *temporarily* hide a change management event type from the calendar display, choose the **X** for the type at the top of the monthly preview.
> - To *permanently* remove these types from the calendar display, edit the calendar, clear the **Add change management events to the calendar** check box, and then choose **Save**. Removing the types from the calendar display doesn't delete them from your account.

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 tool in 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
- Updating all events from a third-party calendar provider
- Deleting all events imported from a third-party calendar

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 tool in 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. 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.

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.

   -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. Navigate 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 tool in 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 navigate 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 tool in 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.

Sharing a change calendar

You can share a calendar in Change Calendar, a tool in AWS Systems Manager, with other AWS accounts by using the AWS Systems Manager console. When you share a calendar, the calendar is read-only to users in the shared account. Maintenance windows, State Manager associations, and automations aren't shared.

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 tool in 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 tool in AWS Systems Manager. You can also show the next time that the calendar state changes from `OPEN` to `CLOSED`, or the reverse.

> **Note**
>
> For information about integrating Change Calendar with Amazon EventBridge for automated monitoring of calendar state changes, see the section called "Change Calendar integration with Amazon EventBridge". EventBridge integration provides event-driven notifications when calendar states transition, complementing the polling-based approach of the `GetCalendarState` API action.

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. Replace each *`example resource placeholder`* with your own information.

  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 tool in AWS Systems Manager, add a step in an Automation runbook that uses the `aws:assertAwsResourceProperty` 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: python3.8
...
```

Troubleshooting Change Calendar

Use the following information to help you troubleshoot problems with Change Calendar, a tool in AWS Systems Manager.

Topics

- 'Calendar import failed' error

'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.
- **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 Change Manager

Change Manager, a tool in 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 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 tool in 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 IAM Identity Center 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 in 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 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. 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 you 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.

More info

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 IAM Identity Center and Active Directory support**

With IAM Identity Center integration, members of your organization can access AWS accounts and manage their resources using Systems Manager based on a common user identity. Using IAM Identity Center, 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 CloudTrail Lake**

  By creating an event data store in AWS CloudTrail Lake, you can view auditable information about the changes made by change requests that run in your account or organization. The event information stored includes such details as the following:

  - The API actions that were run
  - Tthe request parameters included for those actions
  - The user that ran the action
  - The resources that were updated during the process

- **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. Although 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.

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 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
- Working with Change Manager
- Auditing and logging Change Manager activity
- Troubleshooting Change Manager

Setting up Change Manager

You can use Change Manager, a tool in 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), and then proceed to Configuring Change Manager options and best practices.

If you're using Change Manager with a single account, proceed directly to Configuring Change Manager options and best practices.

> **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)
- Configuring Change Manager options and best practices
- Configuring roles and permissions for Change Manager
- Controlling access to auto-approval runbook workflows

Setting up Change Manager for an organization (management account)

The tasks in this topic apply if you're using Change Manager, a tool in 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.

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 tool in 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**.
3. On the **Change Manager** card, choose **Create**.
4. 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. Although you can make changes from other accounts in the organization, those changes won't be reported in or viewable from the delegated administrator account.

5. 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 and Security best practices for Systems Manager.

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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:CreateDocument",
                "ssm:DescribeDocument",
                "ssm:DescribeDocumentParameters",
```

```
                "ssm:DescribeDocumentPermission",
                "ssm:GetDocument",
                "ssm:ListDocumentVersions",
                "ssm:ModifyDocumentPermission",
                "ssm:UpdateDocument",
                "ssm:UpdateDocumentDefaultVersion"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:*:document/AWS-CreateDynamoDbBackup",
                "arn:aws:ssm:us-east-1:*:document/AWS-AWS-
DeleteDynamoDbBackup",
                "arn:aws:ssm:us-east-1:*:document/AWS-
DeleteDynamoDbTableBackups",
                "arn:aws:ssm:us-east-1:*:document/AWSConfigRemediation-
DeleteDynamoDbTable",
                "arn:aws:ssm:us-east-1:*:document/AWSConfigRemediation-
EnableEncryptionOnDynamoDbTable",
                "arn:aws:ssm:us-east-1:*:document/AWSConfigRemediation-
EnablePITRForDynamoDbTable",
                "arn:aws:ssm:us-east-1:111122223333:document/
MyFirstDBChangeTemplate",
                "arn:aws:ssm:us-east-1:111122223333:document/
MySecondDBChangeTemplate"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ssm:ListDocuments",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:StartChangeRequestExecution",
            "Resource": "arn:aws:ssm:us-east-1:111122223333:automation-
definition/*:*"
        }
    ]
}
```

For more information about IAM policies, see Access management for AWS resources and Creating IAM policies in the *IAM User Guide*.

6. 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.
7. In the **Target OUs** section, select the check boxes of the organizational units to use with Change Manager.
8. 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 role that was created for the job function you configured. For example, `AWS-QuickSetup-SSMChangeMgr-DBAdminInvocationRole`.

> **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.

Configuring Change Manager options and best practices

The tasks in this section must be performed whether you're using Change Manager, a tool in 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
- Task 2: Configuring Change Manager change freeze event approvers and best practices
- Configuring Amazon SNS topics for Change Manager notifications

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 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**.
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 actions in Change Manager by using your existing user, groups, and roles.
   - **AWS IAM Identity Center (IAM Identity Center)** – Allow IAM Identity Center 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.

   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.

   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.

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, 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**.
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.

   - 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.

     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 tool in 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
- Task 2: Update the Amazon SNS access policy
- Task 3: (Optional) Update the AWS Key Management Service access policy

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. In the navigation pane, choose **Customer managed keys**.
3. Choose the ID of the customer managed key you chose when you created the topic.
4. In the **Key policy** section, choose **Switch to policy view**.
5. Choose **Edit**.
6. Enter the following `Sid` block after one of the existing `Sid` blocks in the existing policy. 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"
            ]
        }
    }
}
```

7. Now enter the following `Sid` block after one of the existing `Sid` blocks in the resource policy to help prevent the cross-service confused deputy problem.

This block uses the aws:SourceArn and aws:SourceAccount global condition context keys to limit the permissions that Systems Manager gives another service to the resource.

Replace each *user input placeholder* with your own information.

JSON

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "Configure confused deputy protection for AWS KMS keys
 used in Amazon SNS topic when called from Systems Manager",
            "Effect": "Allow",
            "Principal": {
                "Service": "ssm.amazonaws.com"
            },
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:111122223333:topic-name",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:ssm:us-east-1:111122223333:*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

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
- Task 2: Creating an assume role for Change Manager
- Task 3: Attaching the iam:PassRole policy to other roles
- Task 4: Adding inline policies to an assume role to invoke other AWS services
- Task 5: Configuring user access to Change Manager

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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333: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:us-east-1:111122223333:automation-
definition/template-name:$DEFAULT",
                "arn:aws:ssm:us-east-1::document/template-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:ListOpsItemEvents",
                "ssm:GetOpsItem",
                "ssm:ListDocuments",
                "ssm:DescribeOpsItems"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

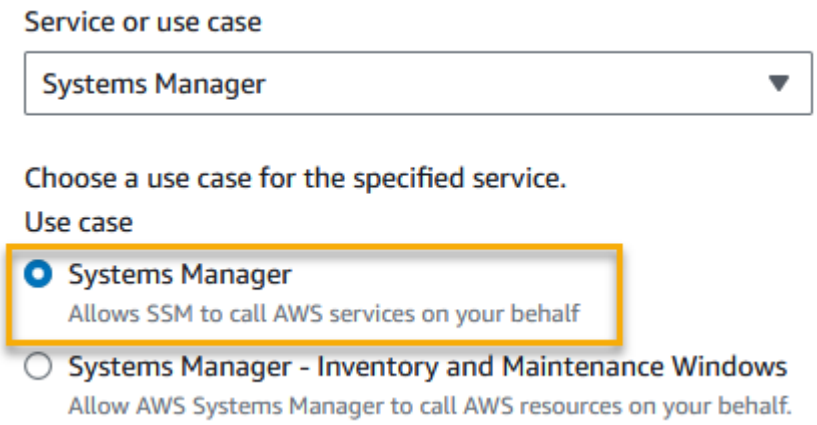
4. For the `iam:PassRole` action, update the `Resource` value to include the ARNs of all job functions defined for your organization that you want to grant permissions to initiate runbook workflows.
5. Replace the *`region`*, *`account-id`*, *`template-name`*, *`delegated-admin-account-id`*, and *`job-function`* 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.
9. Choose **Next: Review**.
10. On the **Review policy** page, enter a name in the **Name** box, such as **`MyChangeManagerAssumeRole`**, and then enter an optional description.
11. Choose **Create policy**, and continue to Task 2: Creating an assume role for Change Manager.

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. For **Select trusted entity**, make the following choices:
    1. For **Trusted entity type**, choose **AWS service**
    2. For **Use cases for other AWS services**, choose **Systems Manager**
    3. Choose **Systems Manager**, as shown in the following image.



4. Choose **Next**.
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], such as **`MyChangeManagerAssumeRole`**.
6. Select the check box next to the assume role policy name, and then choose **Next: Tags**.
7. For **Role name**, enter a name for your new instance profile, such as **`MyChangeManagerAssumeRole`**.
8. (Optional) For **Description**, update the description for this instance role.
9. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this role.
10. Choose **Next: Review**.
11. (Optional) For **Tags**, 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.
12. Choose **Create role**. The system returns you to the **Roles** page.
13. 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 non-EC2 managed nodes in a hybrid and multicloud environment, 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 tools when running runbook workflows.

To attach the `iam:PassRole` policy to an IAM instance profile or service role

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 permissions, Create 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. For **Name**, enter a name to identify this policy, and then choose **Create policy**.

More info

- Configure instance permissions required for Systems Manager
- Create the IAM service role required for Systems Manager in hybrid and multicloud environments

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 permissions, Create inline policy**.
6. Choose the **JSON** tab.
7. Enter a JSON policy document for the AWS services you want to invoke. Here are two example JSON policy documents.

   Amazon S3 `PutObject` and `GetObject` example

   JSON

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

   Amazon EC2 `CreateSnapshot` and `DescribeSnapShots` example

JSON

```
{
   "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. For **Name**, enter a name to identify 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 user, 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 user, group, or role.

Use the following procedure to configure a user to use Change Manager. The user you choose will have permission to configure and run Change Manager.

Depending on the identity application that you are using in your organization, you can select any of the three options available to configure user access. While configuring the user access, assign or add the following:

1. Assign the `AmazonSSMFullAccess` policy or a comparable policy that gives permission to access Systems Manager.
2. Assign the `iam:PassRole` policy.
3. Add the ARN for the Change Manager assume role you copied at the end of Task 2: Creating an assume role for Change Manager.

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.
- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

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.

Controlling access to auto-approval runbook workflows

In each change template created for your organization or account, you can specify whether change requests created from that template can run as auto-approved change requests, meaning that they run automatically without a review step (with the exception of change freeze events).

However, you might want to prevent certain users, groups, or AWS Identity and Access Management (IAM) roles from running auto-approved change requests even if a change template

allows it. You can do this through the use of the `ssm:AutoApprove` condition key for the `StartChangeRequestExecution` operation in an IAM policy assigned to the user, group, or IAM role.

You can add the following policy as an inline policy, where the condition is specified as `false`, to prevent users from running auto-approvable change requests.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
            {
            "Effect": "Allow",
            "Action": "ssm:StartChangeRequestExecution",
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "ssm:AutoApprove": "false"
                }
            }
        }
    ]
}
```

For information about specifying inline policies, see Inline policies and Adding and removing IAM identity permissions in the *IAM User Guide*.

For more information about condition keys for Systems Manager policies, see Condition keys for Systems Manager.

Working with Change Manager

With Change Manager, a tool in 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.

Topics

- Working with change templates
- Working with change requests
- Reviewing change request details, tasks, and timelines (console)
- Viewing aggregated counts of change requests (command line)

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 change template you can use to try out Change Manager, a tool in 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
- Creating change templates
- Reviewing and approving or rejecting change templates
- Deleting change templates

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 tool in 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).
- Configure Change Manager for your delegated administrator account or single account, as described in Configuring Change Manager options and best practices.

> **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**.
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.

Next steps

For information about approving change requests, see Reviewing and approving or rejecting change requests.

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 tool in AWS Systems Manager, using the console, which includes Builder and Editor options, or command line tools.

Topics

- About approvals in your change templates
- Creating change templates using Builder
- Creating change templates using Editor
- Creating change templates using command line tools

About approvals in your change templates

For each change template that you create, you can specify up to five approval *levels* for change requests created from it. For each of those levels, you can designate up to five potential *approvers*. An approver isn't limited to a single user. You can also specify an IAM group or IAM role as an individual approver. For IAM groups and IAM roles, one or more users belonging to the group or role can provide approvals toward receiving the total number of approvals required for a change request. You can also specify more approvers than your change template requires.

Change Manager supports two main approaches to approvals: *per-level approvals* and *per-line approvals*. A combination of the two types is also possible in some situations. We recommend using only per-level approvals in your Change Manager operations.

Per-level approvals

Recommended. As of January 23, 2023, Change Manager supports per-level approvals. In this model, for each approval level in your change template, you first specify how many approvals are required for that level. Then you specify at least that many approvers for the level and can specify more approvers. However, only the number of per-level approvers that you specify need to approve the change request. For example, you could specify five approvers but require three approvals.

For console-view and JSON samples of this approval type, see the section called “Sample per-level approval configuration”.

Per-line approvals

Supported for backward compatibility. The original release of Change Manager supported only per-line approvals. In this model, every approver specified for an approval level is represented as an approval line. Each approver had to approve a change request for it to be approved at that level. Prior to January 23, 2023, this was the only supported model for approvals. Change templates created before this date continue to support per-line approvals, but we recommend using per-level approvals instead.

For console-view and JSON samples of this approval type, see the section called “Sample per-line approval configuration”.

Combined per-line and per-level approvals

Not recommended. In the console, the **Builder** tab no longer supports adding per-line approvals. However, in some cases you might end up with both per-line and per-level approvals in a change template. This can occur if you update a change template that was created before January 23, 2023, or if you create or update a change template by editing its YAML content manually,

For console-view and JSON samples of this approval type, see the section called “Sample combined per-level and per-line approval configuration”.

Important

Although it's possible to create a change template that combines per-line and per-level approvals, this configuration isn't recommended or necessary. Whichever approval type requires more approvals (per-line or per-level approvals) takes precedence. For example:

- If a change template specifies three per-level approvals but five per-line approvals, then five approvals are required.
- If a change template specifies four per-level approvals but two per-line approvals, then four approvals are required.

You can create a level that includes both per-line and per-level approvals by editing the YAML or JSON content manually. Then, the **Builder** tab displays controls for specifying the required number of approvals for both the level and for individual lines. However, new levels that you add using the console still support only per-level approval configurations.

Change request notifications and rejections

Amazon SNS notifications

When a change request is created using your change template, notifications are sent to subscribers of the Amazon Simple Notification Service (Amazon SNS) topic that has been designated for approval notifications at that level. You can specify the notification topic in the change template or allow the user creating the change request to specify one.

After the minimum number of required approvals is received at one level, notifications are sent to approvers subscribed to the Amazon SNS topic for the next level, and so on.

> **Important**
>
> Ensure that the IAM roles, groups, and users you designate together provide enough approvers to meet the required number of approvals you specify. For example, if you designate only a single IAM group as an approver that contains three users, you can't specify that five approvals are mandatory at that level, only three or less.

Change request rejections

No matter how many approval levels and approvers you specify, only one rejection to a change request is required to prevent the runbook workflow for that request from occurring.

Change Manager approval type examples

The following samples demonstrate the console view and JSON content for the three types of approval types in Change Manager.

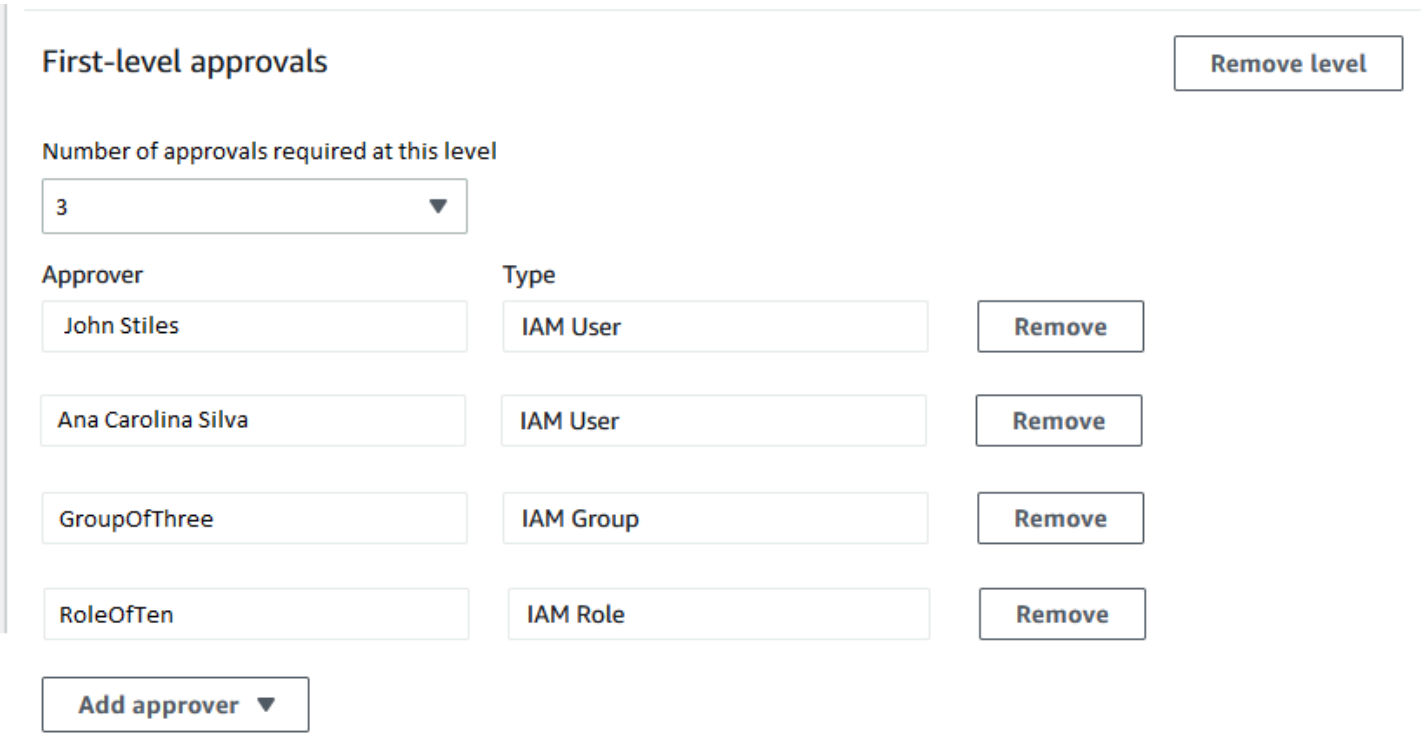
Topics

- Sample per-level approval configuration
- Sample per-line approval configuration
- Sample combined per-level and per-line approval configuration

Sample per-level approval configuration

In the per-level approval level setup shown in the following image, three approvals are required. Those approvals can come from any combination of IAM users, groups, and roles that are specified as approvers. Specified approvers include two IAM users (John Stiles and Ana Carolina Silva), a user group that contains three members (`GroupOfThree`), and a user role that represents ten users (`RoleOfTen`).

If all three users in the `GroupOfThree` group approve the change request, it is approved for that level. It's not necessary to receive an approval from each user, group, or role. The minimum number of approvals can come from any combination of specified approvers. We recommend per-level approvals for your Change Manager operations.



The following sample illustrates part of the YAML code for this configuration.

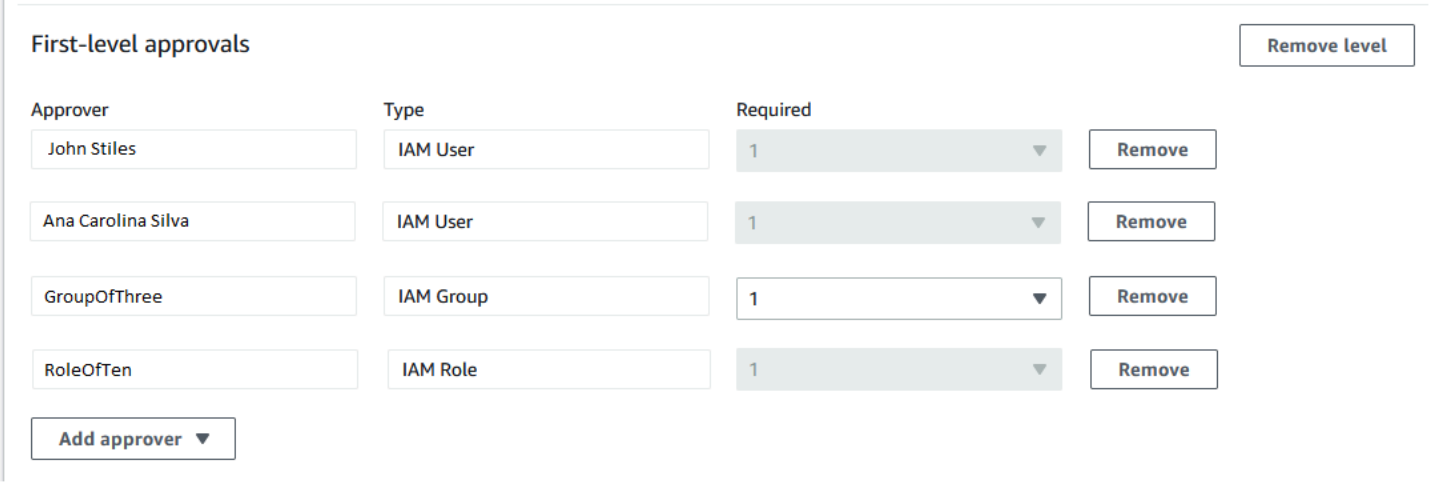
> **Note**
>
> This version of the YAML code include an additional input, `MinRequiredApprovals` (with an initial capital `M`). The value for this input indicates how many approvals are required from among all available reviewers. Note also that the `minRequiredApprovals` (lowercase initial `m`) value for each approver in the `Approvers` list is `0` (zero). This indicates that the approver can contribute to the overall approvals but is not required to do so.

```
schemaVersion: "0.3"
emergencyChange: false
autoApprovable: false
mainSteps:
  - name: ApproveAction1
    action: aws:approve
    timeoutSeconds: 604800
    inputs:
      Message: Please approve this change request
      MinRequiredApprovals: 3
```

```
      EnhancedApprovals:
        Approvers:
          - approver: John Stiles
            type: IamUser
            minRequiredApprovals: 0
          - approver: Ana Carolina Silva
            type: IamUser
            minRequiredApprovals: 0
          - approver: GroupOfThree
            type: IamGroup
            minRequiredApprovals: 0
          - approver: RoleOfTen
            type: IamRole
            minRequiredApprovals: 0
templateInformation: >
  #### What is the purpose of this change?
    //truncated
```

Sample per-line approval configuration

In the approval level setup shown in the following image, four approvers are specified. These include two IAM users (John Stiles and Ana Carolina Silva), a user group that contains three members (`GroupOfThree`), and a user role that represents ten users (`RoleOfTen`). Per-line approvals are supported for backwards compatibility but not recommended.



For the change request to be approved in this per-line approval configuration, it must be approved by all approver lines: John Stiles, Ana Carolina Silva, one member of the `GroupOfThree` group, and one member of the `RoleOfTen` role.

The following sample illustrates part of the YAML code for this configuration.

Note

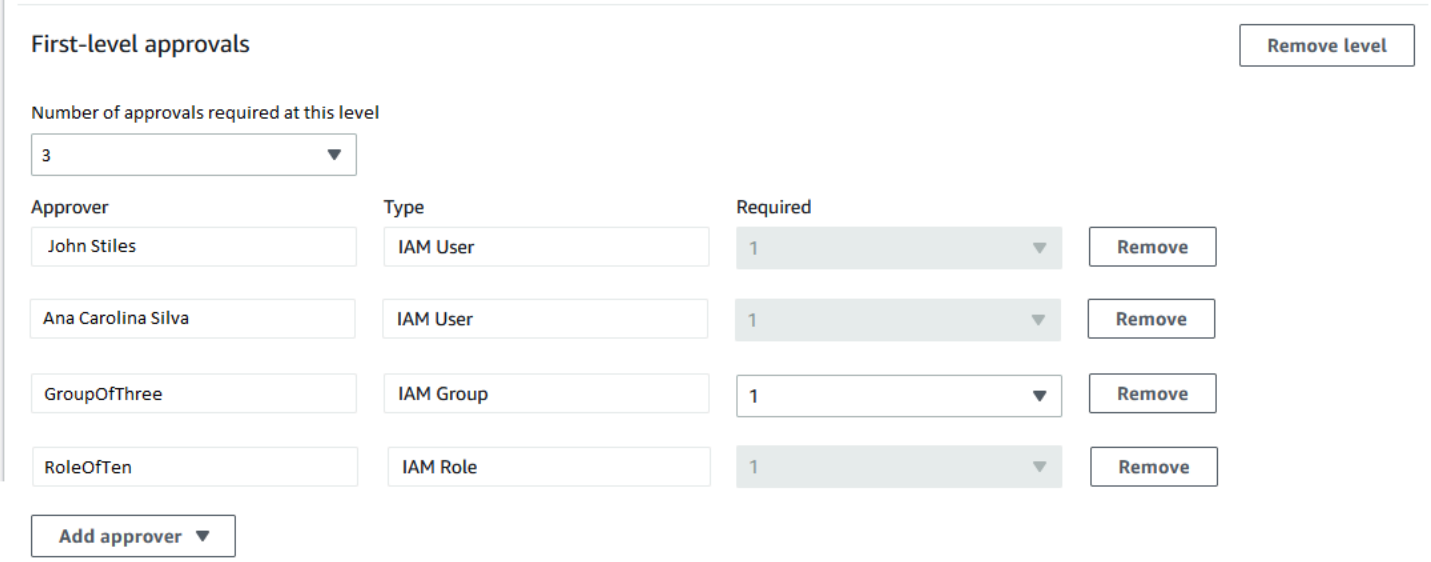
Observe that the value for each `minRequiredApprovals` approver is 1. This indicates that one approval is required from each approver.

```
schemaVersion: "0.3"
emergencyChange: false
autoApprovable: false
mainSteps:
  - name: ApproveAction1
    action: aws:approve
    timeoutSeconds: 10000
    inputs:
      Message: Please approve this change request
      EnhancedApprovals:
        Approvers:
          - approver: John Stiles
            type: IamUser
            minRequiredApprovals: 1
          - approver: Ana Carolina Silva
            type: IamUser
            minRequiredApprovals: 1
          - approver: GroupOfThree
            type: IamGroup
            minRequiredApprovals: 1
          - approver: RoleOfTen
            type: IamRole
            minRequiredApprovals: 1
executableRunBooks:
  - name: AWS-HelloWorld
    version: $DEFAULT
templateInformation: >
  #### What is the purpose of this change?
    //truncated
```

Sample combined per-level and per-line approval configuration

In the combined per-level and per-line approval setup shown in the following image, three approvals are specified for the level, but four approvals are specified for the line-item approvals. Whichever approval type requires more approvals takes precedence over the other, so four

approvals are required by this configuration. Combined per-level and per-line approval are not recommended.




```
schemaVersion: "0.3"
emergencyChange: false
autoApprovable: false
mainSteps:
  - name: ApproveAction1
    action: aws:approve
    timeoutSeconds: 604800
    inputs:
      Message: Please approve this change request
      MinRequiredApprovals: 3
      EnhancedApprovals:
        Approvers:
          - approver: John Stiles
            type: IamUser
            minRequiredApprovals: 1
          - approver: Ana Carolina Silva
            type: IamUser
            minRequiredApprovals: 1
          - approver: GroupOfThree
            type: IamGroup
            minRequiredApprovals: 1
          - approver: RoleOfTen
            type: IamRole
            minRequiredApprovals: 1
templateInformation: >
```

```
#### What is the purpose of this change?
  //truncated
```

Topics

- Creating change templates using Builder
- Creating change templates using Editor
- Creating change templates using command line tools

Creating change templates using Builder

Using the Builder for change templates in Change Manager, a tool in 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**.
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). If you want to restrict a particular user, group, or IAM role from submitting auto-approval requests, you can use a condition in an IAM policy for this purpose. For more information, see Controlling access to auto-approval runbook workflows.

- For **Number of approvals required at this level**, choose the number of approvals that change requests created from this change template must receive for this level.
- 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.

> **Important**
>
> Prior to January 23, 2023, the **Builder** tab supported specifying per-line approvals only. New change templates and new levels you add to existing change templates using the **Builder** tab support per-level approvals only. We recommend using only per-level approvals in your Change Manager operations.
> For more information, see About approvals in your change templates.

- 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.

  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.

    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`

12. Choose **Save and preview**.
13. Review the details of the change template you're creating.

    If you want to make change 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 tool in 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**.
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.
For more information about approvals for change templates, see About approvals in your change templates.

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
autoApprovable: 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, SSOUser\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 SSOUser."
      }
   },
   "executableRunBooks": [
      {
         "name": "AWS-HelloWorld",
         "version": "1"
      }
   ],
```

```
  "emergencyChange": false,
  "autoApprovable": 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 change 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 Server) or AWS Tools for Windows PowerShell to create a change request in Change Manager, a tool in 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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.
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.
For more information about approvals for change templates, see About approvals in your change templates.

```
{
   "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, SSOUser\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 SSOUser."
     }
  },
  "executableRunBooks": [
     {
        "name": "AWS-HelloWorld",
        "version": "1"
     }
  ],
  "emergencyChange": false,
  "autoApprovable": 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
                }
            ]
        }
    }
  ]
}
```

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 tool in 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.

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**.
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 tool in 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**.
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 tool in 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

- Reviewing and approving or rejecting change requests

Creating change requests

When you create a change request in Change Manager, a tool in 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. Although you can make changes from other accounts in the organization, those changes won't be reported in or viewable from the delegated administrator account.

Topics

- About change request approvals
- Creating change requests (console)
- Creating change requests (AWS CLI)

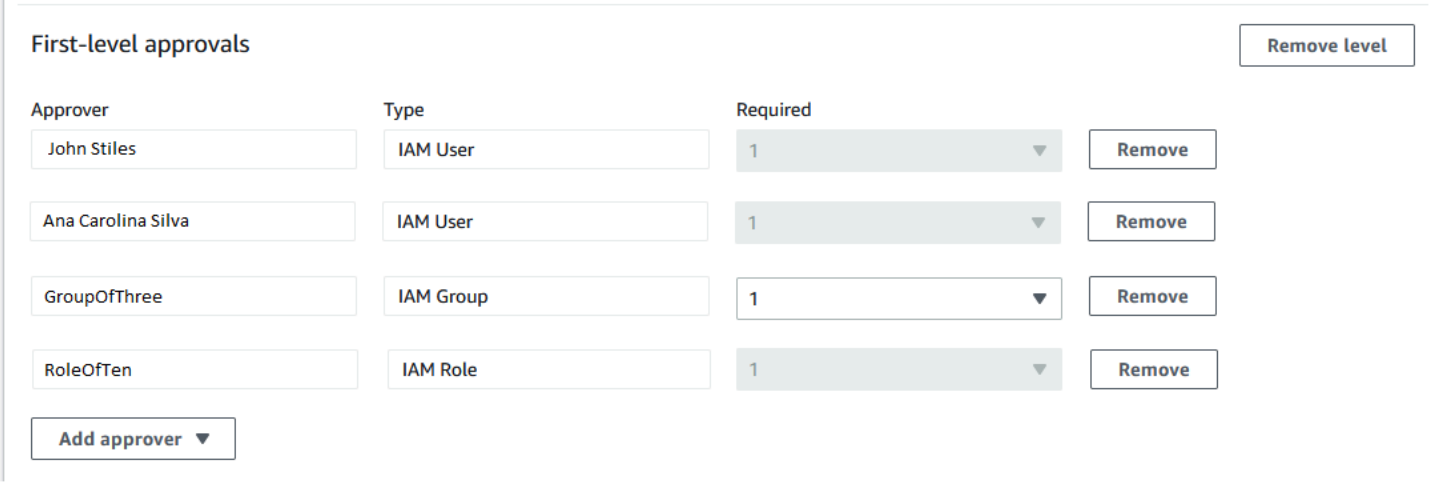
About change request approvals

Depending on the requirements specified in a change template, change requests that you create from it can require approvals from up to five *levels* before the runbook workflow for the request can occur. For each of those levels, the template creator could specify up to five potential *approvers*. An approver isn't limited to a single user. An approver in this sense can also be an IAM group or IAM role. For IAM groups and IAM roles, one or more users belonging to the group or

role can provide approvals toward receiving the total number of approvals required for a change request. Template creators can also specify more approvers than the change template requires.

Original approval workflows and updated and/or approvals

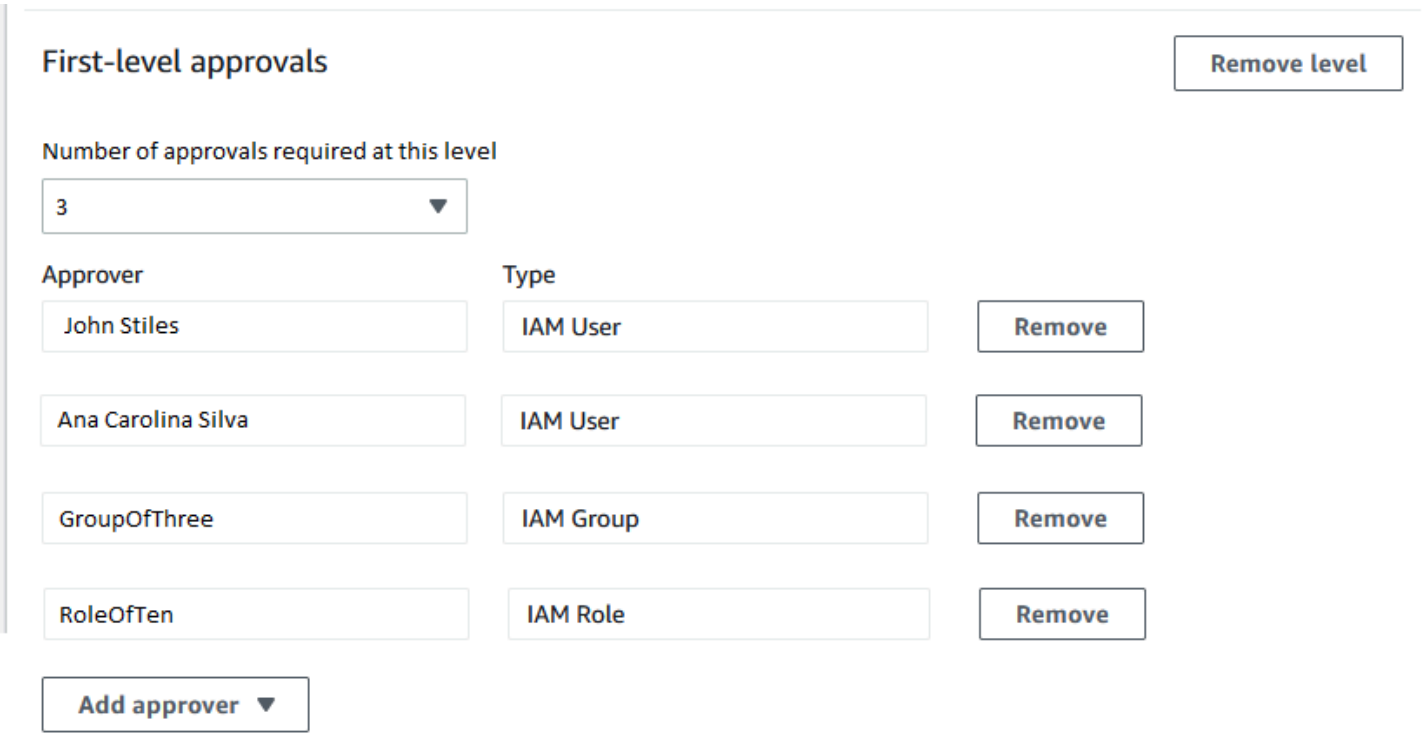
Using change templates created before January 23, 2023, an approval must be received from each specified approver for the change request to be approved at that level. For example, in the approval level setup shown in the following image, four approvers are specified. Specified approvers include two users (John Stiles and Ana Carolina Silva), a user group that contains three members (GroupOfThree), and a user role that represents ten users (RoleOfTen).



For the change request to be approved at this level, it must be approved by John Stiles, Ana Carolina Silva, one member of the `GroupOfThree` group, and one member of the `RoleOfTen` role.

Using change templates created on or after January 23, 2023, for each approval level, template creators can specify an overall total number of required approvals. Those approvals can come from any combination of users, groups, and roles that have been specified as approvers. A change template could require only one approval for a level but specify, for example, two individual users, two groups, and one role as potential approvers.

For example, in the approval level area shown in the following image, three approvals are required. The template-specified approvers include two users (John Stiles and Ana Carolina Silva), a user group that contains three members (`GroupOfThree`), and a user role that represents ten users (`RoleOfTen`).



If all three users in the `GroupOfThree` group approve your change request, it is approved for that level. It's not necessary to receive an approval from each user, group, or role. The minimum number of approvals can come from any combination of potential approvers.

When your change request is created, notifications are sent to subscribers of the Amazon SNS topic that has been specified for approval notifications at that level. The change template creator might have specified the notification topic that must be used or allowed you to specify one.

After the minimum number of required approvals is received at one level, notifications are sent to approvers that are subscribed to the Amazon SNS topic for the next level, and so on.

No matter how many approval levels and approvers are specified, only one rejection to a change request is required to prevent the runbook workflow for that request from occurring.

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**.
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

available, the template you have chosen doesn't allow additional reviewers to be added to requests.

For more information about approvals for change requests, see About change request approvals.

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.

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.
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-RetartEC2Instance`, 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-RetartEC2Instance` 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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.
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
   > For more information about approvals for change requests, see About change request approvals.

```
{
    "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 tool in 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.

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.
> For more information about approvals for change requests, see About change request approvals.

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**.
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 Server) 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 Installing or updating the latest version of the AWS CLI.

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
```

4. 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 tool in 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

1. In the navigation pane, choose **Change Manager**.
2. Choose the **Requests** tab.
3. 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:

   1. 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.
- **Associated events** – View auditable details about change requests that are recorded in AWS CloudTrail Lake. Details include which API actions were run, the request parameters included for those actions, the user account that ran the action, the resources updated during the process, and more.

  When you enable CloudTrail Lake event tracking, CloudTrail Lake creates an event data store for events related to your change requests. The event details are available for the account or organization where the change request ran. You can turn on CloudTrail Lake event tracking from any change request in your account or organization. For information about enabling CloudTrail Lake integration and creating an event data store, see Monitoring your change request events.

  > **Note**
  >
  > There is a charge to use **CloudTrail Lake**. For details, see AWS CloudTrail pricing.

Viewing aggregated counts of change requests (command line)

You can view aggregated counts of change requests in Change Manager, a tool in 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 Creating a resource data sync for Inventory.

The following procedure describes how to use the AWS Command Line Interface (AWS CLI) (on Linux, macOS, or Windows Server) 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 Installing or updating the latest version of the AWS CLI.

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","Aggregat
[{"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","Aggregat
[{"AggregatorType":"count","TypeName":"AWS:OpsItem","AttributeName":"SourceRegion"}]}]'
```

The call returns information like the following.

```
{
        "Entities": [
            {
                "Data": {
                    "AWS:OpsItem": {
                        "Content": [
                            {
                                "Count": "38",
                                "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","Aggregat
[{"AggregatorType":"count","TypeName":"AWS:OpsItem","AttributeName":"SourceAccountId","A
[{"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","Aggregat
[{"AggregatorType":"count","TypeName":"AWS:OpsItem","AttributeName":"SourceAccountId","A
[{"AggregatorType":"count","TypeName":"AWS:OpsItem","AttributeName":"SourceRegion"}]}]}]
```

The call returns information like the following.

```
{
    "Entities": [
        {
            "Data": {
                "AWS:OpsItem": {
                    "Content": [
```

```
                        {
                            "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 tool in AWS Systems Manager, by using Amazon CloudWatch and AWS CloudTrail alarms.

For more information about auditing and logging options for Systems Manager, see Logging and monitoring in AWS Systems Manager.

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.

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.

Troubleshooting Change Manager

Use the following information to help you troubleshoot problems with Change Manager, a tool in AWS Systems Manager.

Topics

- “Group {GUID} not found” error during change request approvals when using Active Directory (groups

"Group *{GUID}* not found" error during change request approvals when using Active Directory (groups

Problem: When AWS IAM Identity Center (IAM Identity Center) 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 IAM Identity Center for access to the AWS Management Console, the system schedules a periodic synchronization that copies information from those Active Directory groups into IAM Identity Center. 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 IAM Identity Center User Guide*.

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.

For enhanced security, as of July 14th, 2025, SSM documents support environment variable interpolation when processing parameters. This feature, available in schema version 2.2 and with SSM Agent version 3.3.2746.0 or higher, helps prevent command injection attacks.

To get started with SSM documents, open the Systems Manager console. In the navigation pane, choose **Documents**.

> **Important**
>
> In Systems Manager, an *Amazon-owned* SSM document is a document created and managed by Amazon Web Services itself. *Amazon-owned* documents include a prefix like `AWS-*` in the document name. The owner of the document is considered to be Amazon, not a specific user account within AWS. These documents are publicly available for all to use.

How can the Documents tool benefit my organization?

Documents, a tool in AWS Systems Manager, 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 document content.

- **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.

- **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.

Who should use Documents?

- Any AWS customer who wants to use Systems Manager tools 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 ApplicationConfigurationSchema	AWS AppConfig	AWS AppConfig, a tool in 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 Freeform configurations in the *AWS AppConfig User Guide*. If you create a configuration in an SSM document, then

Type	Use with	Details
		you must specify a correspon ding JSON Schema. The schema uses the `Applicati onConfigurationSch ema` document type and, like a set of rules, defines the allowable properties for each application configuration setting. For more information, see About validators in the *AWS AppConfig User Guide*.
Automation runbook	Automation State Manager Maintenance Windows	Use Automation runbooks when performing common maintenance and deploymen t 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. Maintenan ce Windows uses Automatio n runbooks to perform common maintenance and deployment tasks based on the specified schedule. All Automation runbooks that are supported for Linux-bas ed operating systems are also supported on EC2 instances for macOS.

Type	Use with	Details
Change Calendar document	Change Calendar	Change Calendar, a tool in AWS Systems Manager, uses the `ChangeCalendar` document type. A Change Calendar document stores a calendar entry and associated events that can allow or prevent Automation actions from changing your environment. In Change Calendar, a document stores iCalendar 2.0 data in plaintext format. Change Calendar isn't supported on EC2 instances for macOS.

Type	Use with	Details
AWS CloudFormation template	AWS CloudFormation	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. You can create and edit CloudFormation templates and stacks by using Application Manager, a tool in Systems Manager. For more information, see Working with AWS CloudFormation templates and stacks in Application Manager.

Type	Use with	Details
Command document	Run Command State Manager Maintenance Windows	Run Command, a tool in AWS Systems Manager, uses Command documents to run commands. State Manager, a tool in 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 tool in AWS Systems Manager, uses Command documents to apply a configuration based on the specified schedule. 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: • `AWS-ConfigureAWSPackage` • `AWS-RunPatchBaseline` • `AWS-RunPatchBaselineAssociation` • `AWS-RunShellScript`

Type	Use with	Details
AWS Config conformance pack template	AWS Config	AWS Config conformance pack templates are YAML formatted documents used to create conformance packs that contains the list of AWS Config managed or custom rules and remediation actions. For more information, see Conformance Packs.
Package document	Distributor	In Distributor, a tool in 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. Distributor isn't supported on Oracle Linux and macOS managed instances.

Type	Use with	Details
Policy document	State Manager	Inventory, a tool in AWS Systems Manager, uses the `AWS-GatherSoftwareInventory` Policy document with a State Manager association to collect inventory data from managed instances. When creating your own SSM documents, Automation runbooks and Command documents are the preferred method for enforcing a policy on a managed instance. Systems Manager Inventory and the `AWS-GatherSoftwareInventory` Policy document are supported on all operating systems supported by Systems Manager.
Post-incident analysis template	Incident Manager post-incident analysis	Incident Manager uses the post-incident analysis template to create an analysis based on AWS operations management best practices. Use the template to create an analysis that your team can use to identify improvements to your incident response.

Type	Use with	Details
Session document	Session Manager	Session Manager, a tool in 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 interacti ve command, or a session to create an SSH tunnel. 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: • `AWS-PasswordReset` • `AWS-StartInteracti veCommand` • `AWS-StartPortForwa rdingSession` • `AWS-StartPortForwa rdingSessionToSock et` • `AWS-StartSSHSession`

SSM document quotas

For information about SSM document quotas, see Systems Manager service quotas in the *Amazon Web Services General Reference*.

Topics

- Document components
- Creating SSM document content
- Working with documents
- Troubleshooting parameter handling issues

Document components

This section includes information about the components that make up SSM documents.

Contents

- Schemas, features, and examples
- Data elements and parameters
- Command document plugin reference

Schemas, 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.
- Documents of type `Session` must use schema version 1.0.
- You can create documents in JSON or YAML.

For more information about `Session` document schema, see Session document schema.

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 *steps* 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 `precondition` parameter within a step.
Parameter interpolation	Interpolation means to insert or substitute a variable value into a string. Think of it as filling in a blank space with actual values before the string is used. In the context of SSM documents, parameter interpolation allows string parameters to be interpolated into environment variables before command execution, providing better security against command injections. When set to `ENV_VAR`, the agent creates an environment variable named `SSM_`*`parameter-name`* that contains the parameter's value.

> **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 Updating the SSM Agent using Run Command.

The following table lists the differences between major schema versions.

Version 1.2	Version 2.2 (latest version)	Details
runtimeConfig	mainSteps	In version 2.2, the `mainSteps` section replaces `runtimeConfig`. The `mainSteps` section allows Systems Manager to run steps in sequence.
properties	inputs	In version 2.2, the `inputs` section replaces the `properties` section. The `inputs` section accepts parameters for steps.
commands	runCommand	In version 2.2, the `inputs` section takes the `runCommand` 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"
    allowedValues:
    - "Hello World"
mainSteps:
  - action: "aws:runPowerShellScript"
    name: "example"
    inputs:
      timeoutSeconds: '60'
      runCommand:
      - "Write-Output {{Message}}"
```

JSON

```
{
   "schemaVersion": "2.2",
   "description": "Example document",
   "parameters": {
      "Message": {
         "type": "String",
         "description": "Example parameter",
         "default": "Hello World",
         "allowedValues": ["Hello World"]
      }
   },
   "mainSteps": [
      {
         "action": "aws:runPowerShellScript",
         "name": "example",
         "inputs": {
            "timeoutSeconds": "60",
            "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:
    - cmds
- action: aws:runShellScript
  name: PatchLinux
  precondition:
    StringEquals:
    - platformType
    - Linux
  inputs:
    runCommand:
    - cmds
```

JSON

```
{
   "schemaVersion": "2.2",
   "description": "cross-platform sample",
   "mainSteps": [
      {
         "action": "aws:runPowerShellScript",
         "name": "PatchWindows",
```

```
         "precondition": {
            "StringEquals": [
               "platformType",
               "Windows"
            ]
         },
         "inputs": {
            "runCommand": [
               "cmds"
            ]
         }
      },
      {
         "action": "aws:runShellScript",
         "name": "PatchLinux",
         "precondition": {
            "StringEquals": [
               "platformType",
               "Linux"
            ]
         },
         "inputs": {
            "runCommand": [
               "cmds"
            ]
         }
      }
   ]
}
```

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 interpolation example with SSM Agent versions before 3.3.2746.0

On SSM Agent versions prior to 3.3.2746.0, the agent ignores the `interpolationType` parameter and instead performs a raw string substitution. If you are referencing `SSM_`*`parameter-name`* explicitly, you must set this explicitly. In the following example for Linux, the `SSM_Message` environment variable is referenced explicitly.

```
{
    "schemaVersion": "2.2",
    "description": "An example document",
    "parameters": {
        "Message": {
            "type": "String",
            "description": "Message to be printed",
            "default": "Hello",
            "interpolationType" : "ENV_VAR",
      "allowedPattern: "^[^"]*$"

        }
    },
    "mainSteps": [{
        "action": "aws:runShellScript",
        "name": "printMessage",
        "inputs": {
            "runCommand": [
              "if [ -z "${SSM_Message+x}" ]; then",
              "    export SSM_Message=\"{{Message}}\"",
              "fi",
              "",
              "echo $SSM_Message"
            ]
        }
    }
}
```

Note

`allowedPattern` isn't technically required if an SSM document doesn't use double braces: `{{ }}`

Schema version 2.2 State Manager example

You can use the following SSM document with State Manager, a tool in 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."
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."
      }
   },
   "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 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": {
        "parameter1": {
            "type": "String",
            "description": "parameter-1-description",
            "default": ""
        },
        "parameter2": {
            "type": "String",
            "description": "parameter-2-description",
            "default": ""
        }
    },
    "variables": {
        "variable1": {
            "type": "StringMap",
            "description": "variable-1-description",
            "default": {}
        },
        "variable2": {
            "type": "String",
            "description": "variable-2-description",
            "default": "default-value"
        }
    },
    "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/al2023-ami-kernel-6.1-x86_64 }}
schemaVersion: '0.3'
assumeRole: 'arn:aws:iam::111122223333::role/AutomationServiceRole'
parameters:
  imageId:
    type: String
    default: '{{ ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-
x86_64 }}'
    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: python3.8
      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: python3.8
      Handler: poll_instance
      Script: |-
        def poll_instance(events, context):
          import boto3
          import time

          ec2 = boto3.client('ec2')

          instance_id = events['InstanceId']

          print('[INFO] Waiting for instance status check to report ok', instance_id)

          instance_status = "null"

          while True:
            res = ec2.describe_instance_status(InstanceIds=[instance_id])

            if len(res['InstanceStatuses']) == 0:
              print("Instance status information is not available yet")
              time.sleep(5)
              continue

            instance_status = res['InstanceStatuses'][0]['InstanceStatus']['Status']

            print('[INFO] Polling to get status of the instance', instance_status)

            if instance_status == 'ok':
              break

            time.sleep(10)

          return {'Status': instance_status, 'InstanceId': instance_id}
      InputPayload: '{{ LaunchEc2Instance.payload }}'
    description: >-
```

```
      **About This Step**


      The python script continuously polls the instance status check value for
      the instance launched in Step 1 until the ```ok``` status is returned.
files:
  launch.py:
    checksums:
      sha256: 18871b1311b295c43d0f...[truncated]...772da97b67e99d84d342ef4aEXAMPLE
```

Secure parameter handling examples

The following examples demonstrate secure parameter handling using environment variable `interpolationType`.

Basic secure command execution

This example shows how to securely handle a command parameter:

> **Note**
>
> `allowedPattern` isn't technically required in SSM documents that don't use double braces: `{{ }}`

YAML

```
---

schemaVersion: '2.2'
description: An example document.
parameters:
  Message:
    type: String
    description: "Message to be printed"
    default: Hello
    interpolationType: ENV_VAR
    allowedPattern: "^[^"]*$"
mainSteps:
  - action: aws:runShellScript
    name: printMessage
    precondition:
      StringEquals:
```

```
        - platformType
        - Linux
    inputs:
      runCommand:
        - echo {{Message}}
```

JSON

```
{
    "schemaVersion": "2.2",
    "description": "An example document.",
    "parameters": {
        "Message": {
            "type": "String",
            "description": "Message to be printed",
            "default": "Hello",
            "interpolationType": "ENV_VAR",
            "allowedPattern": "^[^"]*$"
        }
    },
    "mainSteps": [{
        "action": "aws:runShellScript",
        "name": "printMessage",
        "precondition": {
           "StringEquals": ["platformType", "Linux"]
        },
        "inputs": {
            "runCommand": [
              "echo {{Message}}"
            ]
        }
    }]
}
```

Using parameters in interpreted languages

This example demonstrates secure parameter handling in Python:

YAML

```
---
schemaVersion: '2.2'
```

```
description: 'Secure Python script execution'
parameters:
  inputData:
    type: String
    description: 'Input data for processing'
    interpolationType: 'ENV_VAR'
mainSteps:
  - action: aws:runPowerShellScript
    name: runPython
    inputs:
      runCommand:
        - |
          python3 -c '
          import os
          import json

          # Safely access parameter through environment variable
          input_data = os.environ.get("SSM_inputData", "")

          # Process the data
          try:
              processed_data = json.loads(input_data)
              print(f"Successfully processed: {processed_data}")
          except json.JSONDecodeError:
              print("Invalid JSON input")
          '
```

Backwards compatibility example

This example shows how to handle parameters securely while maintaining backwards compatibility:

YAML

```
---
schemaVersion: '2.2'
description: 'Backwards compatible secure parameter handling'
parameters:
  userInput:
    type: String
    description: 'User input to process'
    interpolationType: 'ENV_VAR'
```

```
    allowedPattern: '^[^"]*$'

mainSteps:
  - action: aws:runShellScript
    name: processInput
    inputs:
      runCommand:
        - |
          # Handle both modern and legacy agent versions
          if [ -z "${SSM_userInput+x}" ]; then
              # Legacy agent - fall back to direct parameter reference
              export SSM_userInput="{{userInput}}"
          fi

          # Process the input securely
          echo "Processing input: $SSM_userInput"
```

> **Note**
>
> `allowedPattern` isn't technically required in SSM documents that don't use double braces: `{{ }}`

Parameter security best practices

Follow these best practices when handling parameters in SSM documents:

- **Use environment variable interpolation** - Always use `interpolationType: "ENV_VAR"` for string parameters that will be used in command execution.
- **Implement input validation** - Use `allowedPattern` to restrict parameter values to safe patterns.
- **Handle legacy systems** - Include fallback logic for older SSM Agent versions that don't support environment variable interpolation.
- **Escape special characters** - When using parameter values in commands, properly escape special characters to prevent interpretation by the shell.
- **Limit parameter scope** - Use the most restrictive parameter patterns possible for your use case.

Data elements and parameters

This topic describes the data elements used in SSM documents. The schema version used to create a document defines the syntax and data elements that the document accepts. 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 the Console in the *AWS Management Console Getting Started Guide*.

The following section describes the data elements that you can include in a SSM document.

Top-level data 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 enhanced security when handling string parameters, you can use environment variable interpolation by specifying the `interpolationType` property. When set to `ENV_VAR`, the system creates an environment variable named `SSM_`*`parameter-name`* containing the parameter value.

The following includes an example of a parameter using environment variable `interpolationType`:

```
{
    "schemaVersion": "2.2",
    "description": "An example document.",
    "parameters": {
        "Message": {
            "type": "String",
            "description": "Message to be printed",
            "default": "Hello",
            "interpolationType" : "ENV_VAR",
            "allowedPattern": "^[^"]*$"

        }
    },
    "mainSteps": [{
        "action": "aws:runShellScript",
        "name": "printMessage",
        "precondition" : {
           "StringEquals" : ["platformType", "Linux"]
        },
        "inputs": {
            "runCommand": [
              "echo {{Message}}"
            ]
        }
    }
}
```

> **Note**
>
> `allowedPattern` isn't technically required in SSM documents that don't use double braces: `{{ }}`

For parameters that you use often, we recommend that you store those parameters in Parameter Store, a tool in AWS Systems Manager. Then, you can define parameters in your document that reference Parameter Store parameters as their default value. To reference a Parameter Store parameter, use the following syntax.

```
{{ssm:parameter-name}}
```

You can use a parameter that references a Parameter Store parameter the same way as any other document parameters. In the following example, the default value for the `commands` parameter is the Parameter Store parameter `myShellCommands`. By specifying the `commands` parameter as a `runCommand` string, the document runs the commands stored in the `myShellCommands` parameter.

YAML

```
---
schemaVersion: '2.2'
description: runShellScript with command strings stored as Parameter Store
 parameter
parameters:
  commands:
    type: StringList
    description: "(Required) The commands to run on the instance."
    default: ["{{ ssm:myShellCommands }}"],
            interpolationType : 'ENV_VAR'
            allowedPattern: '^[^"]*$'

mainSteps:
- action: aws:runShellScript
  name: runShellScriptDefaultParams
  inputs:
    runCommand:"{{ commands }}"
```

JSON

```
{
    "schemaVersion": "2.2",
    "description": "runShellScript with command strings stored as Parameter Store
 parameter",
    "parameters": {
      "commands": {
        "type": "StringList",
        "description": "(Required) The commands to run on the instance.",
        "default": ["{{ ssm:myShellCommands }}"],
        "interpolationType" : "ENV_VAR"
      }
    },
```

```
    "mainSteps": [
      {
        "action": "aws:runShellScript",
        "name": "runShellScriptDefaultParams",
        "inputs": {
            "runCommand": [
              "{{ commands }}"
          ]
        }
      }
    ]
  }
```

> **Note**
>
> You can reference `String` and `StringList` Parameter Store parameters in the `parameters` section of your document. You can't reference `SecureString` Parameter Store parameters.

For more information about Parameter Store, see AWS Systems Manager Parameter Store.

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 in the next section.

  > **Note**
  >
  > Command type documents only support the `String` and `StringList` parameter types.

- `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-f0-9]{8}|[a-f0-9]{17})$"
  default: ''
```

JSON

```
"InstanceId": {
  "type": "String",
  "description": "(Required) The instance ID to target.",
  "allowedPattern": "^i-(?:[a-f0-9]{8}|[a-f0-9]{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.
- `interpolationType`: (Optional) Defines how parameter values are processed before command execution. When set to `ENV_VAR`, the parameter value is made available as an environment variable named `SSM_`*`parameter-name`*. This feature helps prevent command injection by treating parameter values as literal strings.

  Type: String

  Valid values: `ENV_VAR`

Required: No

variables

(Schema version 0.3 only) Values you can reference or update throughout the steps in an Automation runbook. Variables are similar to parameters, but differ in a very important way.

Parameter values are static in the context of a runbook, but the values of variables can be changed in the context of the runbook. When updating the value of a variable, the data type must match the defined data type. For information about updating variables values in an automation, see [aws:updateVariable – Updates a value for a runbook variable]

Type: Boolean | Integer | MapList | String | StringList | StringMap

Required: No

YAML

```
variables:
    payload:
        type: StringMap
        default: "{}"
```

JSON

```
{
    "variables": [
        "payload": {
            "type": "StringMap",
            "default": "{}"
        }
    ]
}
```

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 Using action outputs as inputs.

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.

To learn about the runtimes supported by Automation runbooks, see `aws:executeScript` – Run a script. For more information about including scripts in Automation runbooks, see Using scripts in runbooks and Visual design experience for Automation runbooks.

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 SSM documents 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.

String parameters can include an optional `interpolationType` field with the value `ENV_VAR` to enable environment variable interpolation for improved security.

YAML

```
---
InstanceId:
  type: String
  description: "(Optional) The target EC2 instance ID."
  interpolationType: ENV_VAR
```

JSON

```
"InstanceId":{
  "type":"String",
  "description":"(Optional) The target EC2 instance ID.",
  "interpolationType": "ENV_VAR"
}
```

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."
  default: ""
  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. Keys and values must be strings. 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 objects.

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
}
```

Viewing SSM Command document content

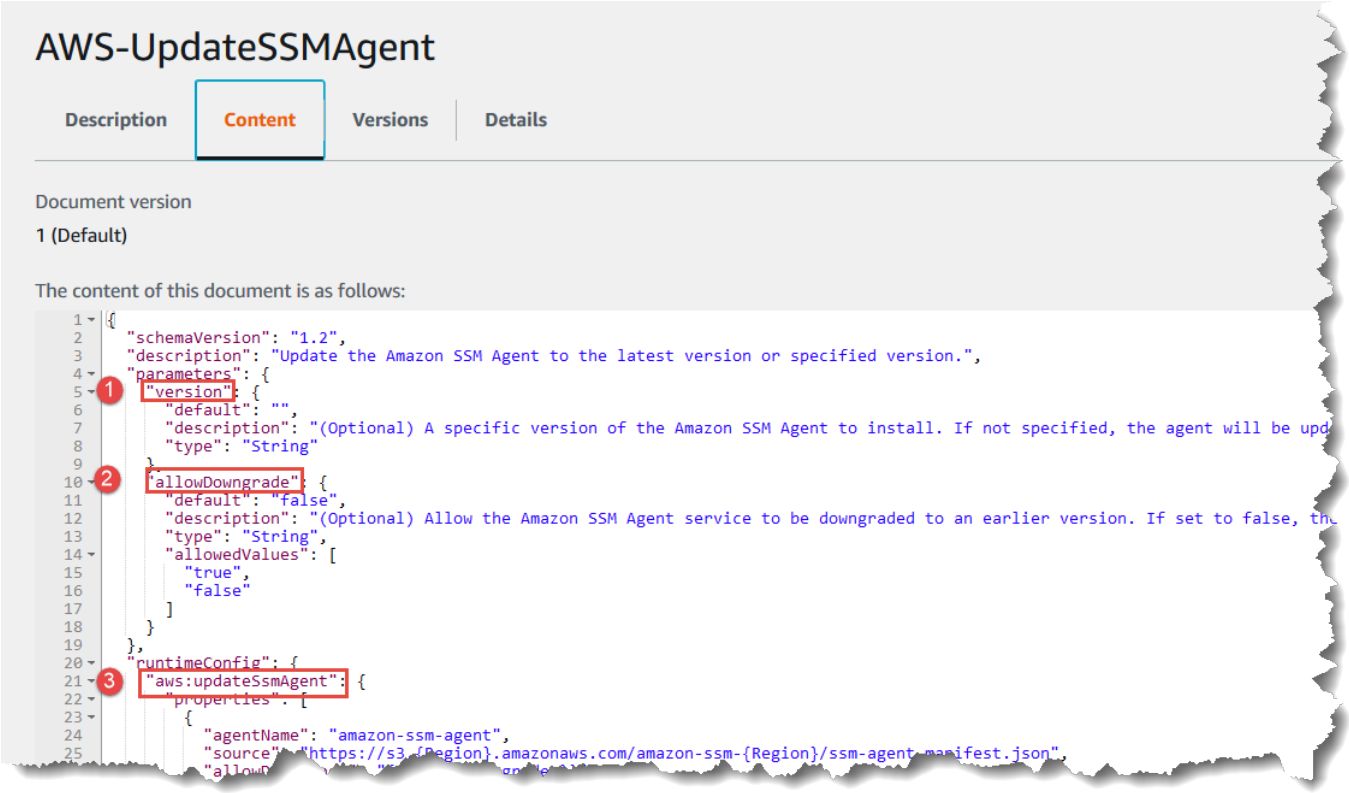
To preview the required and optional parameters for an AWS Systems Manager (SSM) Command document, in addition to the actions the document runs, you can view the content of the document in the Systems Manager console.

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**.

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`.



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.

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.

For plugins that accept String parameters, such as `aws:runShellScript` and `aws:runPowerShellScript`, the `interpolationType` parameter can be used to enhance security by treating parameter inputs as string literals rather than potentially executable commands. For example:

```
{
    "schemaVersion": "2.2",
    "description": "runShellScript with command strings stored as Parameter Store
 parameter",
    "parameters": {
      "commands": {
        "type": "StringList",
        "description": "(Required) The commands to run on the instance.",
        "default": ["{{ ssm:myShellCommands }}"],
        "interpolationType" : "ENV_VAR"
      }
    },
    //truncated
 }
```

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

- aws:applications
- aws:cloudWatch
- aws:configureDocker
- aws:configurePackage
- aws:domainJoin
- aws:downloadContent
- aws:psModule
- aws:refreshAssociation
- aws:runDockerAction
- aws:runDocument
- aws:runPowerShellScript
- aws:runShellScript
- aws:softwareInventory
- aws:updateAgent
- aws:updateSsmAgent

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.

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 Collecting metrics, logs, and traces with the CloudWatch agent in the *Amazon CloudWatch User Guide*.

> **Important**
>
> The unified CloudWatch agent has replaced SSM Agent as the tool for sending log data to Amazon CloudWatch Logs. The SSM Agent aws:cloudWatch plugin is not supported. We recommend using only the unified CloudWatch agent for your log collection processes. For more information, see the following topics:
>
> - Sending node logs to unified CloudWatch Logs (CloudWatch agent)
> - Migrate Windows Server node log collection to the CloudWatch agent
> - Collecting metrics, logs, and traces 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 Microsoft Learn 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`*n* | `CloudWatchLogs`*n*

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 information 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\\W3SVC*n***). 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 most Linux variants and Windows Server operating systems.

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`.

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
  ssmParameter:
    description: "(Required) Argument stored in Parameter Store."
    type: String
    default: "{{ ssm:parameter_store_arg }}"
```

```
mainSteps:
- action: aws:configurePackage
  name: configurePackage
  inputs:
    name: "{{ name }}"
    action: "{{ action }}"
    additionalArguments:
      "{\"SSM_parameter_store_arg\": \"{{ ssmParameter }}\", \"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"
         ]
      },
      "ssmParameter": {
         "description": "(Required) Argument stored in Parameter Store.",
         "type": "String",
         "default": "{{ ssm:parameter_store_arg }}"
      }
   },
   "mainSteps": [
      {
         "action": "aws:configurePackage",
         "name": "configurePackage",
         "inputs": {
            "name": "{{ name }}",
            "action": "{{ action }}",
```

```
            "additionalArguments": "{\"SSM_parameter_store_arg\":
 \"{{ ssmParameter }}\", \"SSM_custom_arg\": \"myValue\"}"
         }
      }
   ]
}
```

Inputs

name

The name of the AWS package to install or uninstall. Available packages include the following: `AWSPVDriver`, `AwsEnaNetworkDriver`, `AwsVssComponents`, and `AmazonCloudWatchAgent`.

Type: String

Required: Yes

action

Install or uninstall a package.

Type: Enum

Valid values: `Install` | `Uninstall`

Required: Yes

installationType

The type of installation to perform. If you specify `Uninstall and reinstall`, the package is completely uninstalled, and then reinstalled. The application is unavailable until the reinstallation 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

A JSON string of 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. This plugin changes the hostname for Linux instances to the format EC2AMAZ-*XXXXXXX*. 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*.

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
  directoryOU:
    description: "(Optional) The organizational unit to assign the computer object
 to."
    type: String
  dnsIpAddresses:
    description: "(Required) The IP addresses of the DNS servers for your
 directory."
    type: StringList
  hostname:
    description: "(Optional) The hostname you want to assign to the node."
    type: String
mainSteps:
- action: aws:domainJoin
  name: domainJoin
  inputs:
    directoryId: "{{ directoryId }}"
    directoryName: "{{ directoryName }}"
    directoryOU: "{{ directoryOU }}"
    dnsIpAddresses: "{{ dnsIpAddresses }}"
    hostname: "{{ hostname }}"
```

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"
```

```
    },
    "directoryOU": {
        "description": "(Optional) The organizational unit to assign the computer
 object to.",
        "type": "String"
      },
    "dnsIpAddresses": {
      "description": "(Required) The IP addresses of the DNS servers for your
 directory.",
      "type": "StringList"
    },
    "hostname": {
        "description": "(Optional) The hostname you want to assign to the node.",
        "type": "String"
      }
  },
  "mainSteps": [
    {
      "action": "aws:domainJoin",
      "name": "domainJoin",
      "inputs": {
        "directoryId": "{{ directoryId }}",
        "directoryName": "{{ directoryName }}",
        "directoryOU":"{{ directoryOU }}",
        "dnsIpAddresses":"{{ dnsIpAddresses }}",
        "hostname":"{{ hostname }}"
      }
    }
  ]
}
```

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"]

hostname

The hostname you want to assign to the node. If not provided, there is no name change for Windows Server instances, while Linux instances will use the default naming pattern. If provided, Windows Server instances will use the exact provided value, while for Linux instances it serves as a prefix (unless `keepHostName` is set to "true").

Type: String

Required: No

keepHostName

Determines whether the hostname is changed for Linux instances when joined to the domain. This is a Linux-only parameter. By default (without inputs to `hostname`, `hostnameNumAppendDigits`, and with `keepHostName` as "false"), Linux hosts will be renamed to the pattern EC2AMAZ-XXXXXX. When set to "true", it keeps the original hostname and ignores inputs to `hostname` and `hostnameNumAppendDigits`.

Type: Boolean

Required: No

hostnameNumAppendDigits

Defines the number of random numeric digits to append after the hostname value. This is a Linux-only parameter and is used together with the `hostname` parameter. It is ignored if `hostname` is not provided.

Type: String

Allowed values: 1 to 5

Required: No

Examples

For examples, see Join an Amazon EC2 Instance to your AWS Managed Microsoft AD in the *AWS Directory Service Administration Guide*.

`aws:downloadContent`

(Schema version 2.0 or later) Download SSM documents and scripts from remote locations. GitHub Enterprise repositories are not supported. 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/heads/*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:bbc1ddb94...b76d3bEXAMPLE",
    ```
- tokenInfo: The Systems Manager parameter (a SecureString parameter) where you store your GitHub access token information, in the format `{{ssm-secure:`*secure-string-token-name*`}}`.

  > **Note**
  >
  > This `tokenInfo` field is the only SSM document plugin field that supports a SecureString parameter. SecureString parameters 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:refs/heads/*branch_name*

    The default is `master`.

    `"branch"` is required only if your SSM document is stored in a branch other than `master`. For example:

    ```
    "getOptions": "branch:refs/heads/main"
    ```

  - 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/amzn-s3-demo-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.

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.

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.

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 }}"
    workingDirectory: "{{ workingDirectory }}"
    timeoutSeconds: "{{ timeoutSeconds }}"
```

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 }}",
            "workingDirectory": "{{ workingDirectory }}",
```

```
            "timeoutSeconds": "{{ timeoutSeconds }}"
         }
      }
   ]
}
```

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

workingDirectory

The path to the working directory on your managed node.

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

`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` 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.

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.

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.

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.

Type: String

Required: No

networkConfig

(Optional) Collect metadata for network configurations.

Type: String

Required: No

billingInfo

(Optional) Collect metadata for platform details associated with the billing code of the AMI.

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.

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

Type: String

Required: No

customInventoryDirectory

(Optional) Collect custom inventory data from the specified directory. For more information about custom inventory, see Working with custom inventory

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 (legacy) in the *Amazon EC2 User Guide*.

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.

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

Creating SSM document content

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 creating or cloning a document, the content of the document must not exceed 64KB. This quota also includes the content specified for input parameters at runtime. 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
- Data elements and parameters
- Schemas, features, and examples
- Command document plugin reference
- Systems Manager Automation actions reference
- Automation system variables
- Additional runbook examples
- Working with Systems Manager Automation runbooks using the AWS Toolkit for Visual Studio Code
- Visual design experience for Automation runbooks
- Using scripts in runbooks

AWS pre-defined SSM documents might perform some of the actions you require.
You can call these documents by using the `aws:runDocument`, `aws:runCommand`, or `aws:executeAutomation` plugins within your custom SSM document, depending on the document type. You can also copy portions of those documents into a custom SSM document, and edit the content to meet your requirements.

Tip

When creating SSM document content, you might change the content and update your SSM document several times while testing. The following commands update the SSM document with your latest content, and update the document's default version to the latest version of the document.

Note

The Linux and Windows commands use the `jq` command line tool to filter the JSON response data.

Linux & macOS

```
latestDocVersion=$(aws ssm update-document \
    --content file://path/to/file/documentContent.json \
    --name "ExampleDocument" \
    --document-format JSON \
    --document-version '$LATEST' \
    | jq -r '.DocumentDescription.LatestVersion')

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 '.DocumentDescription.LatestVersion')

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
```

Security best practices for SSM documents

When creating SSM documents, follow these security best practices to help prevent command injection and ensure secure parameter handling:

- Use environment variable interpolation for string parameters that will be used in commands or scripts. Add the `interpolationType` property with value `ENV_VAR` to your string parameters:

```
{
    "command": {
        "type": "String",
        "description": "Command to execute",
        "interpolationType": "ENV_VAR"
    }
}
```

  You can further improve the security of your SSM documents by specifying that double-quote marks aren't accepted in values delivered by interpolation:

```
{
    "command": {
        "type": "String",
        "description": "Command to execute",
        "interpolationType": "ENV_VAR",
```

```
            "allowedPattern": "^[^"]*$"
    }
}
```

- When using interpreted languages like Python, Ruby, or Node.js, reference parameters using the appropriate environment variable syntax:

```
# Python example
import os
command = os.environ['SSM_Message']
```

- For backwards compatibility with older SSM Agent versions (prior to version 3.3.2746.0), include fallback logic for environment variables:

```
if [ -z "${SSM_command+x}" ]; then
    export SSM_command="{{command}}"
fi
```

- Combine environment variable interpolation with `allowedPattern` for additional input validation. In the following example, the `allowedPattern` value `^[^"]*$` specifically prevent double-quotes in the string value:

```
{
    "command": {
        "type": "String",
        "interpolationType": "ENV_VAR",
        "allowedPattern": "^[a-zA-Z0-9_-]+$"
    }
}
```

- Before implementing your SSM document, verify the following security considerations:
  - All string parameters that accept user input use environment variable interpolation when appropriate.
  - Input validation is implemented using `allowedPattern` where possible.
  - The document includes appropriate error handling for parameter processing.
  - Backwards compatibility is maintained for environments using older SSM Agent versions.

For information about AWS service-owned resources that Systems Manager accesses and how to configure data perimeter policies, see Data perimeters in AWS Systems Manager.

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. You can't clone a document larger than 64KB.

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**.
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.

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

- Creating composite documents

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. Installs all patches in the allow list.
2. Installs antivirus software.
3. 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-RunPatchBaseline` document, which installs all allow listed patches.
2. The `aws:runDocument` plugin to run the `AWS-InstallApplication` document, which installs the antivirus software.
3. 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.

To create a composite document, add the `aws:runDocument` 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` 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": {
        "documentType": "LocalPath",
        "documentPath": "bootstrap",
        "documentParameters": "{}"
      }
    },
    {
      "action": "aws:runDocument",
      "name": "configureDocker",
      "inputs": {
        "documentType": "SSMDocument",
        "documentPath": "AWS-ConfigureDocker",
        "documentParameters": "{\"action\":\"Install\"}"
      }
    }
  ]
}
```

More info

- For information about rebooting servers and instances when using Run Command to call scripts, see Handling reboots when running commands.
- For more information about the plugins you can add to a custom SSM document, see Command document plugin reference.
- If you simply want to run a document from a remote location (without creating a composite document), see Running documents from remote locations.

Working with documents

This section includes information about how to use and work with SSM documents.

Topics

- Compare SSM document versions
- Create an SSM document
- Deleting custom SSM documents
- Running documents from remote locations
- Sharing SSM documents
- Searching for SSM documents

Compare 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**.
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.

Create an SSM document

After you create the content for your custom SSM document, as described in Writing SSM document content, you can use the Systems Manager console to create an SSM document using your content.

To create 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**.
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`
9. Choose **Create document** to save the document.

Deleting custom SSM documents

If you no longer want to use a custom SSM document, you can delete it using the AWS Systems Manager console.

To delete 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**.
3. Select the document you want to delete.
4. Select **Delete**. When prompted to delete the document, select **Delete**.

For examples using command line tools or SDKs to delete SSM documents, see Use `DeleteDocument` with an AWS SDK or CLI.

Running 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:

- Public and private GitHub repositories (GitHub Enterprise is not supported)
- Amazon S3 buckets
- Systems Manager

While you can also run remote documents by using State Manager or Automation, tools in 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`.

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 document content
- 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 Parameter Store parameters in Systems Manager.

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**.
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:bbc1ddb94...b76d3bEXAMPLE",
```

- If you choose **S3**, specify **Source Info** information in the following format:

```
{"path":"URL_to_document_in_S3"}
```

For example:

```
{"path":"https://s3.amazonaws.com/amzn-s3-demo-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.

> **Tip**
>
> If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

12. Choose **Run**.

> **Note**
>
> For information about rebooting servers and instances when using Run Command to call scripts, see Handling reboots when running commands.

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`. Documents can't be simultaneously shared publicly and privately.

> **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.

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.
- You can share a document with a maximum of 1000 AWS accounts. You can request an increase to this limit in the 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 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.

> **Important**
>
> In Systems Manager, an *Amazon-owned* SSM document is a document created and managed by Amazon Web Services itself. *Amazon-owned* documents include a prefix like `AWS-*` in the document name. The owner of the document is considered to be Amazon, not a specific user account within AWS. These documents are publicly available for all to use.

For more information about Systems Manager service quotas, see AWS Systems Manager Service Quotas.

Contents

- Best practices for shared SSM documents
- Block public sharing for SSM documents
- Share an SSM document
- Modify permissions for a shared SSM document
- Using shared SSM documents

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

Review all publicly shared SSM documents in your account and confirm whether you want to continue sharing them. To stop sharing a document with the public, you must modify the document permission setting as described in the Modify permissions for a shared SSM document section of this topic. Turning on the block public sharing setting doesn't affect any documents you're currently sharing with the public. Unless your use case requires you to share documents with the public, we recommend turning on the block public sharing setting for your SSM documents in the **Preferences** section of the Systems Manager Documents console. 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.

Restrict Run Command actions using an IAM trust policy

Create a restrictive AWS Identity and Access Management (IAM) 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 Customer managed policy examples.

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.

Use the interpolation parameter to improve security

For `String` type parameters in your SSM documents, use the parameter and value `interpolationType": "ENV_VAR` to improve security against command injection attacks by treating parameter inputs as string literals rather than potentially executable commands. In this case, the agent creates an environment variable named `SSM_`*`parameter-name`* with the parameter's value. We recommend updating all your existing SSM documents that include `String` type parameters to include `"interpolationType": "ENV_VAR"`. For more information, see the section called “Writing SSM document content”.

Block public sharing for SSM documents

Before you begin, review all publicly shared SSM documents in your AWS account and confirm whether you want to continue sharing them. To stop sharing an SSM document with the public, you must modify the document permission setting as described in the Modify permissions for a shared SSM document section of this topic. Turning on the block public sharing setting doesn't affect any SSM documents you're currently sharing with the public. With the block public sharing setting enabled, you won’t be able to share any additional SSM documents with the public.

Unless your use case requires you to share documents with the public, we recommend turning on the block public sharing setting for your 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 any SSM documents you're not currently sharing.

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**.

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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:UpdateServiceSetting",
            "Resource": "arn:aws:ssm:*:444455556666: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. When sharing documents from the console, only the default version of the document can be shared. 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)

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 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. Optionally, for documents shared with specific AWS account IDs, you can specify a version of the document you want to share using the `--shared-document-version` parameter. If you don't specify a version, the system shares the `Default` version of the document. If you share a document publicly (with `all`), all versions of the specified document are shared by default. The following example 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.

> **Note**
>
> If you share a document publicly (with `all`), all versions of the specified document are shared by default.

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**.
3. In the documents list, choose the document you want to stop sharing, and then choose the **Details**. In the **Permissions** section, 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. You can also favorite documents to help you find frequently used SSM documents. The following sections describes how to use these features.

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**.
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 Installing or updating the latest version of the AWS CLI.

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
```

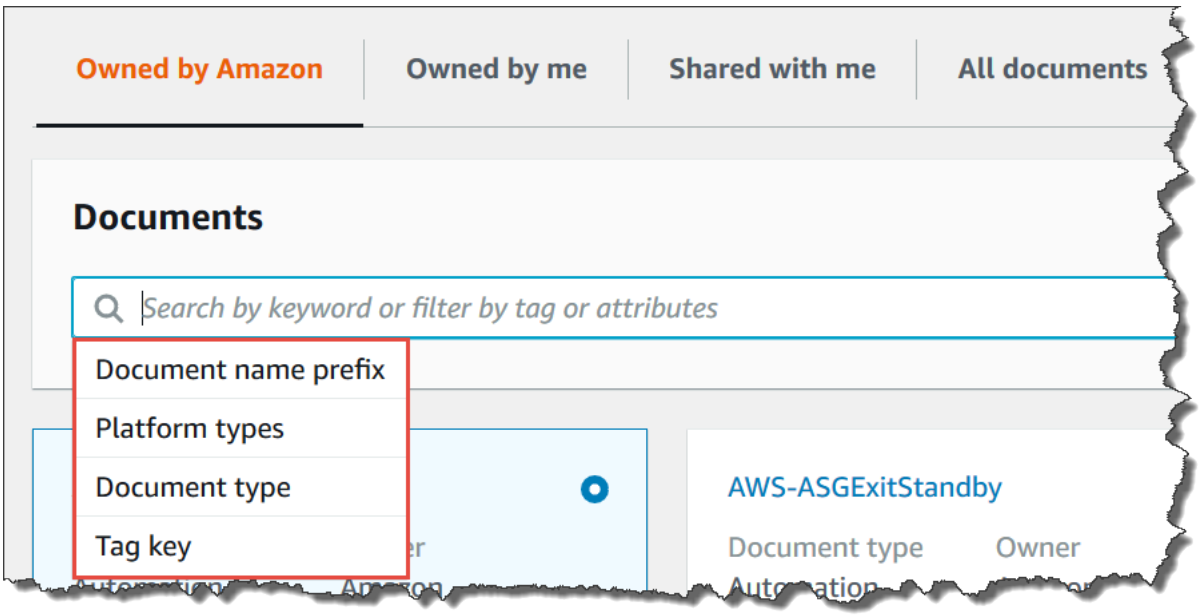
Here's an example.

```
aws ssm list-documents --filters Key="SearchKeyword",Values="aws-asg","aws-
ec2","restart" --region us-east-2
```

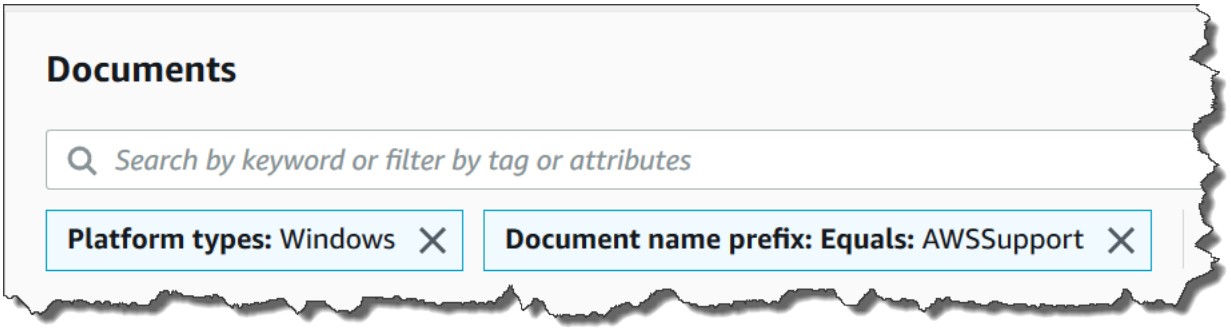
Using filters

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.

Adding documents to your favorites

To help you find frequently used SSM documents, add documents to your favorites. You can favorite up to 20 documents per document type, per AWS account and AWS Region. You can choose, modify, and view your favorites from the documents AWS Management Console. The following procedures describe how to choose, modify, and view your favorites.

To favorite 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**.
3. Select the star icon next to the document name you want to favorite.

To remove an SSM document from your favorites

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Documents**.
3. Deselect the star icon next to the document name you want to remove from your favorites.

To view your favorites from the documents AWS Management Console

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Documents**.
3. Select the **Favorites** tab.

Troubleshooting parameter handling issues

Common parameter handling issues

Environment variables not available during execution

Problem: Commands fail because environment variables (SSM_*parameter-name*) are not found.

Possible causes:

- SSM Agent version doesn't support environment variable interpolation
- `interpolationType` is not set to `ENV_VAR`
- Parameter name doesn't match the expected environment variable name

Solution:

- Verify SSM Agent version is 3.3.2746.0 or later
- Add fallback logic for older agent versions:

```
if [ -z "${SSM_parameterName+x}" ]; then
    export SSM_parameterName="{{parameterName}}"
fi
```

Parameter values containing special characters

Problem: Commands fail when parameter values contain spaces, quotes, or other special characters.

Solution:

- Use proper quoting when referencing environment variables:

```
# Correct
echo "$SSM_parameter-name"

# Incorrect
echo $SSM_parameter-name
```

- Add input validation using `allowedPattern` to restrict special characters

Inconsistent behavior across platforms

Problem: Parameter handling works differently on Linux and Windows Server systems.

Solution:

- Use platform-specific environment variable syntax:

```
# PowerShell
$env:SSM_parameter-name

# Bash
$SSM_parameter-name
```

- Use platform-specific precondition checks in your document

Parameter values not properly escaped

Problem: Command injection vulnerabilities despite using environment variable interpolation.

Solution:

- Always use proper escaping when including parameter values in commands:

```
# Correct
mysql_command="mysql -u \"$SSM_username\" -p\"$SSM_password\""

# Incorrect
mysql_command="mysql -u $SSM_username -p$SSM_password"
```

Parameter validation tips

Use these techniques to validate your parameter handling:

1. Test environment variable availability:

```
#!/bin/bash
# Print all SSM_ environment variables
env | grep ^SSM_

# Test specific parameter
if [ -n "$SSM_parameter" ]; then
    echo "Parameter is available"
else
    echo "Parameter is not available"
fi
```

2. Verify parameter patterns:

```
parameters:
  myParameter:
    type: String
    allowedPattern: "^[a-zA-Z0-9_-]+$"
    description: "Test this pattern with sample inputs"
```

3. Include error handling:

```
if [[ ! "$SSM_parameter" =~ ^[a-zA-Z0-9_-]+$ ]]; then
    echo "Parameter validation failed"
    exit 1
fi
```

AWS Systems Manager Maintenance Windows

Maintenance Windows, a tool in 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.

> **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.

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**.

Each maintenance window has a schedule, a maximum duration, a set of registered targets (the managed 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.

For more information about working with the `--schedule` option, see Reference: Cron and rate expressions for Systems Manager.

Supported task types

With maintenance windows, you can run four types of tasks:

- Commands in Run Command, a tool in Systems Manager

  For more information about Run Command, see AWS Systems Manager Run Command.
- Workflows in Automation, a tool in Systems Manager

  For more information about Automation workflows, see AWS Systems Manager Automation.
- 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

> **Note**
>
> Maintenance window tasks support Step Functions Standard state machine workflows only. They don't support Express state machine workflows. For information about state machine workflow types, see Standard vs. Express Workflows in the *AWS Step Functions Developer Guide*.

For more information about Step Functions, see the *AWS Step Functions Developer Guide*.

This means you can use maintenance windows to perform tasks such as 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.

> **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.

EventBridge support

This Systems Manager tool is supported as an *event* type in Amazon EventBridge rules. For information, see Monitoring Systems Manager events with Amazon EventBridge and Reference: Amazon EventBridge event patterns and types for Systems Manager.

Contents

- Setting up Maintenance Windows
- Create and manage maintenance windows using the console
- Tutorials

- Using pseudo parameters when registering maintenance window tasks
- Maintenance window scheduling and active period options
- Registering maintenance window tasks without targets
- Troubleshooting maintenance windows

Setting up Maintenance Windows

Before users in your AWS account can create and schedule maintenance window tasks using Maintenance Windows, a tool in AWS Systems Manager, they must be granted the necessary permissions. In addition, you must create an IAM service role for maintenance windows and the IAM policy to attach to it.

Before you begin

In addition to the permissions you configure in this section, the IAM Entities (users, roles, or groups that will work with maintenance windows should already have general maintenance window permissions. You can grant these permissions by assigning the IAM policy `AmazonSSMFullAccess` to the Entities, or assigning a custom IAM policy that provides a smaller set of access permissions for Systems Manager that covers maintenance window tasks.

Topics

- Control access to maintenance windows using the console
- Control access to maintenance windows using the AWS CLI

Control access to maintenance windows using the console

The following procedures describe how to use the AWS Systems Manager console to create the required permissions and roles for maintenance windows.

Topics

- Task 1: Create a custom policy for your maintenance window service role using the console
- Task 2: Create a custom service role for maintenance windows using the console
- Task 3: Grant permissions to specified users to register maintenance window tasks using the console
- Task 4: Prevent specified users from registering maintenance window tasks using the console

Task 1: Create a custom policy for your maintenance window service role using the console

Maintenance window tasks require an IAM role to provide the permissions required to run on the target resources. The permissions are provided through an IAM policy attached to the role. The types of tasks you run and your other operational requirements determine the contents of this policy. We provide a base policy you can adapt to your needs. Depending on the tasks and types of tasks your maintenance windows run, you might not need all the permissions in this policy, and you might need to include additional permissions. You attach this policy to the role that you create later in Task 2: Create a custom service role for maintenance windows using the console.

To create a custom policy using the console

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. In the **Policy editor** area, choose **JSON**.
4. Replace the default contents with the following:

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand",
                "ssm:CancelCommand",
                "ssm:ListCommands",
                "ssm:ListCommandInvocations",
                "ssm:GetCommandInvocation",
                "ssm:GetAutomationExecution",
                "ssm:StartAutomationExecution",
                "ssm:ListTagsForResource",
                "ssm:DescribeInstanceInformation",
                "ssm:GetParameters"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "states:DescribeExecution",
                "states:StartExecution"
            ],
            "Resource": [
                "arn:aws:states:*:*:execution:*:*",
                "arn:aws:states:*:*:stateMachine:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:function:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "resource-groups:ListGroups",
                "resource-groups:ListGroupResources"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/maintenance-window-
role-name",
            "Condition": {
                "StringEquals": {
```

```
                    "iam:PassedToService": [
                        "ssm.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

5. Modify the JSON content as needed for the maintenance tasks that you run in your account. The changes you make are specific to your planned operations.

   For example:

   - You can provide Amazon Resource Names (ARNs) for specific functions and state machines instead of using wildcard (*) qualifiers.
   - If you don't plan to run AWS Step Functions tasks, you can remove the `states` permissions and (ARNs).
   - If you don't plan to run AWS Lambda tasks, you can remove the `lambda` permissions and ARNs.
   - If you don't plan to run Automation tasks, you can remove the `ssm:GetAutomationExecution` and `ssm:StartAutomationExecution` permissions.
   - Add additional permissions that might be needed for the tasks to run. For example, some Automation actions work with AWS CloudFormation stacks. Therefore, the permissions `cloudformation:CreateStack`, `cloudformation:DescribeStacks`, and `cloudformation:DeleteStack` are required.

     For another example, the Automation runbook `AWS-CopySnapshot` requires permissions to create an Amazon Elastic Block Store (Amazon EBS) snapshot. Therefore, the service role needs the permission `ec2:CreateSnapshot`.

     For information about the role permissions needed by Automation runbooks, see the runbook descriptions in the AWS Systems Manager Automation Runbook Reference.

6. After completing the policy revisions, choose **Next**.
7. For **Policy name**, enter a name that identifies this as the policy attached to the service role you create. For example: **`my-maintenance-window-role-policy`**.
8. (Optional) In the **Add tags** area, add one or more tag-key value pairs to organize, track, or control access for this policy.

9. Choose **Create policy**.

   Make a note of the name you specified for the policy. You refer to it in the next procedure, Task 2: Create a custom service role for maintenance windows using the console.

Task 2: Create a custom service role for maintenance windows using the console

The policy you created in the previous task is attached to the maintenance window service role you create in this task. When users register a maintenance window task, they specify this IAM role as part of the task configuration. The permissions in this role allow Systems Manager to run tasks in maintenance windows on your behalf.

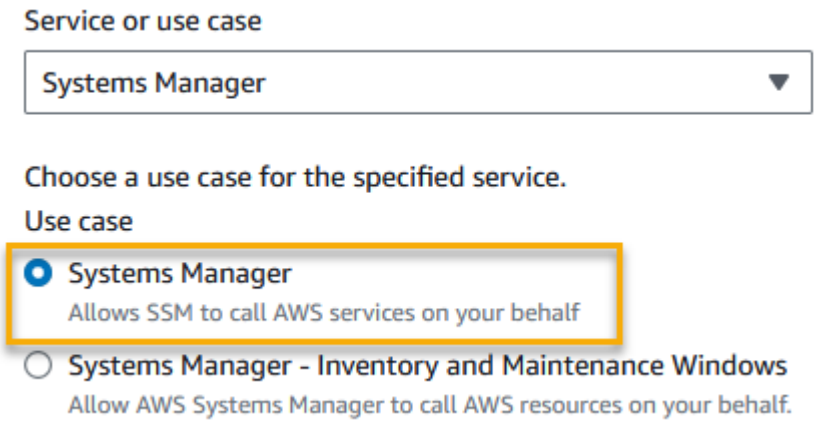
> **Important**
>
> Previously, the Systems Manager console provided you with the ability to choose the AWS managed IAM service-linked role `AWSServiceRoleForAmazonSSM` to use as the maintenance role for your tasks. Using this role and its associated policy, `AmazonSSMServiceRolePolicy`, for maintenance window tasks is no longer recommended. If you're using this role for maintenance window tasks now, we encourage you to stop using it. Instead, create your own IAM role that enables communication between Systems Manager and other AWS services when your maintenance window tasks run.

Use the following procedure to create a custom service role for Maintenance Windows, so that Systems Manager can run Maintenance Windows tasks on your behalf. You attach the policy you created in the previous task to the custom service role you create.

To create a custom service role for maintenance windows using the 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. For **Select trusted entity**, make the following choices:
   1. For **Trusted entity type**, choose **AWS service**.
   2. For **Use case**, choose **Systems Manager**
   3. Choose **Systems Manager**.

The following image highlights the location of the Systems Manager option.



4. Choose **Next**.
5. In the **Permissions policies** area, in the search box, enter the name of the policy you created in [Task 1: Create a custom policy for your maintenance window service role using the console], select the box next to its name, and then choose **Next**.
6. For **Role name**, enter a name that identifies this role as a Maintenance Windows role. For example: **`my-maintenance-window-role`**.
7. (Optional) Change the default role description to reflect the purpose of this role. For example: **`Performs maintenance window tasks on your behalf`**.
8. For **Step 1: Select trusted entities**, verify that the following policy is displayed in the **Trusted policy** box.

   JSON

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

```
}
```

9. For **Step 2: Add permissions**, verify that policy you created in Task 1: Create a custom policy for your maintenance window service role using the console is present.
10. (Optional) In **Step 3: Add tags**, add one or more tag-key value pairs to organize, track, or control access for this role.
11. Choose **Create role**. The system returns you to the **Roles** page.
12. Choose the name of the IAM role you just created.
13. Copy or make a note of the role name and the **ARN** value in the **Summary** area. Users in your account specify this information when they create maintenance windows.

Task 3: Grant permissions to specified users to register maintenance window tasks using the console

Providing users with permissions to access the custom service role for maintenance windows lets them use it with their maintenance windows tasks. This is in addition to permissions that you've already given them to work with the Systems Manager API commands for the Maintenance Windows tool. This IAM role conveys the permissions need to run a maintenance window task. As a result, a user can't register tasks with a maintenance window using your custom service role without the ability to pass these IAM permissions.

When you register a task with a maintenance window, you specify a service 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 entity (such as a user or group). This allows the IAM entity to specify, as part of registering those tasks with the maintenance window, the role that should be used when running tasks. For information, see Grant a user permissions to pass a role to an AWS service in the *IAM User Guide*.

To configure permissions to allow users to register maintenance window tasks

If an IAM entity (user, role, or group) is set up with administrator permissions, then the IAM user or role has access to Maintenance Windows. For IAM entities without administrator permissions, an administrator must grant the following permissions to the IAM entity. These are the minimum permissions required to register tasks with a maintenance window:

- The `AmazonSSMFullAccess` managed policy, or a policy that provides comparable permissions.
- The following `iam:PassRole` and `iam:ListRoles`permissions.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/my-maintenance-window-
role"
        },
        {
            "Effect": "Allow",
            "Action": "iam:ListRoles",
            "Resource": "arn:aws:iam::111122223333:role/"
        },
        {
            "Effect": "Allow",
            "Action": "iam:ListRoles",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
ssm.amazonaws.com/"
        }
    ]
}
```

my-maintenance-window-role represents the name of the custom maintenance window service 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.

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.

- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.

- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

To configure permissions for groups that are allowed to register maintenance window tasks using the 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, or first create a new group if necessary
4. On the **Permissions** tab, choose **Add permissions, Create inline policy**.
5. In the **Policy editor** area, choose **JSON**, and replace the default contents of the box with the following.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/my-maintenance-
window-role"
        },
        {
            "Effect": "Allow",
            "Action": "iam:ListRoles",
            "Resource": "arn:aws:iam::111122223333:role/"
        },
```

```
        {
            "Effect": "Allow",
            "Action": "iam:ListRoles",
            "Resource": "arn:aws:iam::111122223333: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 **Next**.
7. On the **Review and create** page, enter a name in the **Policy name** box to identify this `PassRole` policy, such as **`my-group-iam-passrole-policy`**, and then choose **Create policy**.

Task 4: Prevent specified users from registering maintenance window tasks using the console

You can deny the `ssm:RegisterTaskWithMaintenanceWindow` permission for the users in your AWS account who you don't want to register tasks with maintenance windows. This provides an extra layer of prevention for users who shouldn't register maintenance window tasks.

To configure permissions for groups that aren't allowed to register maintenance window tasks using the 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, or first create a new group if necessary.

4. On the **Permissions** tab, choose **Add permissions, Create inline policy**.
5. In the **Policy editor** area, choose **JSON**, and then replace the default contents of the box with the following.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:RegisterTaskWithMaintenanceWindow",
            "Resource": "*"
        }
    ]
}
```

6. Choose **Next**.
7. On the **Review and create** page, for **Policy 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 using the AWS CLI

The following procedures describe how to use the AWS Command Line Interface (AWS CLI) to create the required permissions and roles for Maintenance Windows, a tool in AWS Systems Manager.

Topics

- Task 1: Create trust policy and customer managed policy files in JSON format
- Task 2: Create and verify a custom service role for maintenance windows using the AWS CLI
- Task 3: Grant permissions to specified users to register maintenance window tasks using the AWS CLI
- Task 4: Prevent specified users from registering maintenance window tasks using the AWS CLI

Task 1: Create trust policy and customer managed policy files in JSON format

Maintenance window tasks require an IAM role to provide the permissions required to run on the target resources. The permissions are provided through an IAM policy attached to the role. The

types of tasks you run and your other operational requirements determine the contents of this policy. We provide a base policy you can adapt to your needs. Depending on the tasks and types of tasks your maintenance windows run, you might not need all the permissions in this policy, and you might need to include additional permissions.

In this task, you specify the permissions needed for your custom maintenance window role in a pair of JSON files. You attach this policy to the role that you create later in Task 2: Create and verify a custom service role for maintenance windows using the AWS CLI.

To create trust policy and customer managed policy files

1. Copy and paste the following trust policy into a text file. Save this file with the following name and file extension: **`mw-role-trust-policy.json`**.

   JSON

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

2. Copy and paste the following JSON policy into a different text file. In the same directory where you created the first file, save this file with the following name and file extension: **`mw-role-custom-policy.json`**.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ssm:SendCommand",
                "ssm:CancelCommand",
                "ssm:ListCommands",
                "ssm:ListCommandInvocations",
                "ssm:GetCommandInvocation",
                "ssm:GetAutomationExecution",
                "ssm:StartAutomationExecution",
                "ssm:ListTagsForResource",
                "ssm:GetParameters"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "states:DescribeExecution",
                "states:StartExecution"
            ],
            "Resource": [
                "arn:aws:states:*:*:execution:*:*",
                "arn:aws:states:*:*:stateMachine:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:function:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "resource-groups:ListGroups",
                "resource-groups:ListGroupResources"
            ],
            "Resource": [
                "*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/maintenance-window-
role-name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ssm.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

3. Modify the the content of `mw-role-custom-policy.json` as needed for the maintenance tasks that you run in your account. The changes you make are specific to your planned operations.

   For example:

   - You can provide Amazon Resource Names (ARNs) for specific functions and state machines instead of using wildcard (*) qualifiers.
   - If you don't plan to run AWS Step Functions tasks, you can remove the `states` permissions and (ARNs).
   - If you don't plan to run AWS Lambda tasks, you can remove the `lambda` permissions and ARNs.
   - If you don't plan to run Automation tasks, you can remove the `ssm:GetAutomationExecution` and `ssm:StartAutomationExecution` permissions.
   - Add additional permissions that might be needed for the tasks to run. For example, some Automation actions work with AWS CloudFormation stacks. Therefore, the

permissions `cloudformation:CreateStack`, `cloudformation:DescribeStacks`, and `cloudformation:DeleteStack` are required.

For another example, the Automation runbook `AWS-CopySnapshot` requires permissions to create an Amazon Elastic Block Store (Amazon EBS) snapshot. Therefore, the service role needs the permission `ec2:CreateSnapshot`.

For information about the role permissions needed by Automation runbooks, see the runbook descriptions in the AWS Systems Manager Automation Runbook Reference.

Save the file again after making any needed changes.

Task 2: Create and verify a custom service role for maintenance windows using the AWS CLI

The policy you created in the previous task is attached to the maintenance window service role you create in this task. When users register a maintenance window task, they specify this IAM role as part of the task configuration. The permissions in this role allow Systems Manager to run tasks in maintenance windows on your behalf.

> **Important**
>
> Previously, the Systems Manager console provided you with the ability to choose the AWS managed IAM service-linked role `AWSServiceRoleForAmazonSSM` to use as the maintenance role for your tasks. Using this role and its associated policy, `AmazonSSMServiceRolePolicy`, for maintenance window tasks is no longer recommended. If you're using this role for maintenance window tasks now, we encourage you to stop using it. Instead, create your own IAM role that enables communication between Systems Manager and other AWS services when your maintenance window tasks run.

In this task, you run CLI commands to create your maintenance windows service role, adding the policy content from the JSON files you created.

Create a custom service role for maintenance windows using the AWS CLI

1. Open the AWS CLI and run the following command in the directory where you placed `mw-role-custom-policy.json` and `mw-role-trust-policy.json`. The command creates a

maintenance window service role called `my-maintenance-window-role` and attaches the *trust policy* to it.

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 similar to the following.

```
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": "sts:AssumeRole",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ssm.amazonaws.com"
                    }
                }
            ]
        },
        "RoleId": "AROAIIZKPBKS2LEXAMPLE",
        "CreateDate": "2024-08-19T03:40:17.373Z",
        "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 include them in the next command.

2. Run the following command to attach the *customer managed policy* to the role. Replace the *`account-id`* placeholder with your own AWS account ID

   Linux & macOS

   ```
   aws iam attach-role-policy \
       --role-name "my-maintenance-window-role" \
       --policy-arn "arn:aws:iam::account-id:policy/mw-role-custom-policy.json"
   ```

   Windows

   ```
   aws iam attach-role-policy ^
       --role-name "my-maintenance-window-role" ^
       --policy-arn "arn:aws:iam::account-id:policy/mw-role-custom-policy.json"
   ```

3. Run the following command to verify that your role has been created, and that the trust policy has been attached.

   ```
   aws iam get-role --role-name my-maintenance-window-role
   ```

   The command returns information similar to the following:

   ```
   {
       "Role": {
           "Path": "/",
           "RoleName": "my-maintenance-window-role",
           "RoleId": "AROA123456789EXAMPLE",
           "Arn": "arn:aws:iam::123456789012:role/my-maintenance-window-role",
           "CreateDate": "2024-08-19T14:13:32+00:00",
           "AssumeRolePolicyDocument": {
               "Version": "2012-10-17",
               "Statement": [
                   {
                       "Effect": "Allow",
                       "Principal": {
   ```

```
                        "Service": "ssm.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "MaxSessionDuration": 3600,
        "RoleLastUsed": {
            "LastUsedDate": "2024-08-19T14:30:44+00:00",
            "Region": "us-east-2"
        }
    }
}
```

4. Run the following command to verify that the customer managed policy has been attached to the role.

```
aws iam list-attached-role-policies --role-name my-maintenance-window-role
```

The command returns information similar to the following:

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "mw-role-custom-policy",
            "PolicyArn": "arn:aws:iam::123456789012:policy/mw-role-custom-policy"
        }
    ]
}
```

Task 3: Grant permissions to specified users to register maintenance window tasks using the AWS CLI

Providing users with permissions to access the custom service role for maintenance windows lets them use it with their maintenance windows tasks. This is in addition to permissions that you've already given them to work with the Systems Manager API commands for the Maintenance Windows tool. This IAM role conveys the permissions need to run a maintenance window task. As a result, a user can't register tasks with a maintenance window using your custom service role without the ability to pass these IAM permissions.

When you register a task with a maintenance window, you specify a service 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 entity (such as a user or group). This allows the IAM entity to specify, as part of registering those tasks with the maintenance window, the role that should be used when running tasks. For information, see Grant 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 using the AWS CLI

1. 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`.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/my-maintenance-
window-role"
        },
        {
            "Effect": "Allow",
            "Action": "iam:ListRoles",
            "Resource": "arn:aws:iam::111122223333:role/"
        },
        {
            "Effect": "Allow",
            "Action": "iam:ListRoles",
            "Resource": "arn:aws:iam::111122223333: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 entity (user or group), run one of the following commands.

   - **For an IAM entity:**

     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 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 (or an IAM policy that provides a smaller set of access permissions for Systems Manager that covers maintenance window tasks). Run the following command to assign the `AmazonSSMFullAccess` policy to your user.

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 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 4: Prevent specified users from registering maintenance window tasks using the AWS CLI

You can deny the `ssm:RegisterTaskWithMaintenanceWindow` permission for the users in your AWS account who you don't want to register tasks with maintenance windows. This provides an extra layer of prevention for users who shouldn't register maintenance window tasks.

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 using the 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`**.

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 entity (user or group), run one of the following commands.

   - **For a 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 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 a 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 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`

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"
```

Create and manage maintenance windows using the 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 tool in AWS Systems Manager. For more information, see Setting up Maintenance Windows.

Topics

- Create a maintenance window using the console
- Assign targets to a maintenance window using the console
- Assign tasks to a maintenance window using the console
- Disable or enable a maintenance window using the console
- Update or delete maintenance window resources using the console

Create a maintenance window using the console

In this procedure, you create a maintenance window in Maintenance Windows, a tool in 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. For more information about working with the `--schedule` option, see Reference: Cron and rate expressions for Systems Manager.

To create a maintenance window using the 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.
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.

    > **Note**
    >
    > You can't specify a start date and time that occurs in the past.

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 using the 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 using the 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.
8. In the **Targets** area, choose one of the options described in the following table.

Option	Description
Specify instance tags	For the **Specify instance tags** 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 *all* the tag keys and values you specify to be included in the target group.

Option	Description
Choose instances manually	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 Troubleshooting managed node availability for troubleshooting tips. For edge devices and on-premises servers and virtual machines (VMs), see Managing nodes in hybrid and multicloud environments with Systems Manager

Option	Description
Choose a resource group	For **Resource group**, choose the name of an existing resource group in your account from the list. For information about creating and working with resource groups, see the following topics: • What are resource groups? in the *AWS Resource Groups User Guide* • Resource Groups and Tagging for AWS in the *AWS News Blog* (Optional) For **Resource types**, select up to five available resource types, or choose **All resource types**. 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. For example, suppose you add the following resource types to this target: • `AWS::S3::Bucket` • `AWS::DynamoDB::Table` • `AWS::EC2::Instance` 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

Option	Description
	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.

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 using the 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 Step Functions tasks
- 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`.

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**

   > **Note**
   >
   > Maintenance window tasks support Step Functions Standard state machine workflows only. They don't support Express state machine workflows. For information about state machine workflow types, see Standard vs. Express Workflows in the *AWS Step Functions Developer Guide*.

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. For this step, follow the sub-steps for your selected task type.

Run Command

1. In the **Command document** list, choose the Systems Manager Command document (SSM document) that defines the tasks to run.
2. For **Document version**, choose the document version to use.
3. 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.

Automation

1. In the **Automation document** list, choose the Automation runbook that defines the tasks to run.
2. For **Document version**, choose the runbook version to use.
3. 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.

Lambda

1. In the **Lambda parameters** area, choose a Lambda function from the list.
2. (Optional) Provide any content for **Payload**, **Client Context**, or **Qualifier** that you want to include.

   > **Note**
   >
   > In some cases, you can use a *pseudo parameter* as part of your `Payload` value. Then when the maintenance window task runs, it passes the correct values instead of the pseudo parameter placeholders. For information, see Using pseudo parameters when registering maintenance window tasks.

3. 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.

Step Functions

1. In the **Step Functions parameters** area, choose a state machine from the list.
2. (Optional) Provide a name for the state machine execution and any content for **Input** that you want to include.

   > **Note**
   >
   > In some cases, you can use a *pseudo parameter* as part of your `Input` value. Then when the maintenance window task runs, it passes the correct values instead of the pseudo parameter placeholders. For information, see Using pseudo parameters when registering maintenance window tasks.

3. 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.

9. 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 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.

> **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.

10. *Automation tasks only:*

    In the **Input parameters** area, provide values for any required or optional parameters needed to run your task.

    > **Note**
    >
    > In some cases, you can use a *pseudo parameter* for certain input parameter values. Then when the maintenance window task runs, it passes the correct values instead of the pseudo parameter placeholders. For information, see Using pseudo parameters when registering maintenance window tasks.

11. 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.

12. (Optional) For **IAM service role**, choose a role to provide permissions for Systems Manager to assume when running a maintenance window task.

    If you don't specify a service role ARN, Systems Manager uses a service-linked role in your account. This role is not listed in the drop-down menu. If no appropriate service-linked role for Systems Manager exists in your account, it's created when the task is registered successfully.

    > **Note**
    >
    > For an improved security posture, we strongly recommend creating a custom policy and custom service role for running your maintenance window tasks. The policy can be crafted to provide only the permissions needed for your particular maintenance window tasks. For more information, see Setting up Maintenance Windows.

13. *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.
    - Select the **CloudWatch output** check box to write complete output to Amazon CloudWatch Logs. Enter the name of a CloudWatch Logs log group.

    > **Note**
    >
    > The permissions that grant the ability to write data to an S3 bucket or CloudWatch Logs are those of the instance profile assigned to the node, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager. In addition, if the specified S3 bucket or log group is in a different AWS account, verify that the instance profile associated with the node has the necessary permissions to write to that bucket.

14. *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.

15. *Run Command tasks only:*

    In the **Parameters** area, specify parameters for the document.

    > **Note**
    >
    > In some cases, you can use a *pseudo parameter* for certain input parameter values. Then when the maintenance window task runs, it passes the correct values instead of the pseudo parameter placeholders. For information, see Using pseudo parameters when registering maintenance window tasks.

16. *Run Command and Automation tasks only:*

    (Optional) In the **CloudWatch alarm** area, for **Alarm name**, choose an existing CloudWatch alarm to apply to your task for monitoring.

    If the alarm activates, the task is stopped.

    > **Note**
    >
    > To attach a CloudWatch alarm to your task, the IAM principal that runs the task must have permission for the `iam:createServiceLinkedRole` action. For more information about CloudWatch alarms, see Using Amazon CloudWatch alarms.

17. Depending on your task type, choose one of the following:

    - **Register Run command task**
    - **Register Automation task**
    - **Register Lambda task**
    - **Register Step Functions task**

Disable or enable a maintenance window using the console

You can disable or enable a maintenance window in Maintenance Windows, a tool in AWS Systems Manager. You can choose one maintenance window at a time to either disable or enable the

maintenance window from running. You can also select multiple or all maintenance windows to enable and disable.

This section describes how to disable or enable a maintenance window 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 using the AWS CLI.

Topics

- Disable a maintenance window using the console
- Enable a maintenance window using the console

Disable a maintenance window using the console

You can disable a maintenance window to pause a task for a specified period, and it will remain available to enable again later.

To disable 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. Using the check box next to the maintenance window that you want to disable, select one or more maintenance windows.
4. Choose **Disable maintenance window** in the **Actions** menu. The system prompts you to confirm your actions.

Enable a maintenance window using the console

You can enable a maintenance window to resume a task.

> **Note**
>
> If the maintenance window uses a rate schedule and the start date is currently set to a past date and time, the current date and time is used as the start date for the maintenance window. You can change the start date of the maintenance window before or after enabling it. For information, see Update or delete maintenance window resources using the console.

To enable 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 check box next to the maintenance window to enable.
4. Choose **Actions, Enable maintenance window**. The system prompts you to confirm your actions.

Update or delete maintenance window resources using the console

You can update or delete a maintenance window in Maintenance Windows, a tool in 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 using the AWS CLI.

Topics

- Updating or deleting a maintenance window using the console
- Updating or deregistering maintenance window targets using the console
- Updating or deregistering maintenance window tasks using the console

Updating or deleting a maintenance window using the 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 using the console.

Updating or deregistering maintenance window targets using the 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 using the 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 using the 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**.

Tutorials

The tutorials in this section show you how to perform common tasks when working with maintenance windows.

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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.
- **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.
- **Create or configure an instance that is compatible with Systems Manager** – You need at least one Amazon Elastic Compute Cloud (Amazon EC2) instance that is configured for use with Systems Manager 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 Find AMIs with the SSM Agent preinstalled.

For information about installing SSM Agent on an instance, see the following topics:

- Manually installing and uninstalling SSM Agent on EC2 instances for Windows Server
- Manually installing and uninstalling SSM Agent on EC2 instances for Linux

For information about configuring IAM permissions for Systems Manager to your instance, see Configure instance permissions required for Systems Manager.

- **Create additional resources as needed** – Run Command, a tool in 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 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 tool supports running the following 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.

Tutorials

- Tutorials: Create and manage maintenance windows using the AWS CLI
- Tutorial: Create a maintenance window for patching using the console

Tutorials: Create and manage maintenance windows using the 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

Keep track of resource IDs

As you complete the tasks in these AWS CLI tutorials, 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 the following 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 that you plan to use in the tutorials. For example: `i-02573cafcfEXAMPLE`

Tutorials

- Tutorial: Create and configure a maintenance window using the AWS CLI
- Tutorial: View information about maintenance windows using the AWS CLI
- Tutorial: View information about tasks and task executions using the AWS CLI
- Tutorial: Update a maintenance window using the AWS CLI
- Tutorial: Delete a maintenance window using the AWS CLI

Tutorial: Create and configure a maintenance window using the 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-02573cafcfEXAMPLE* with IDs of resources you create.

Contents

- Step 1: Create the maintenance window using the AWS CLI
- Step 2: Register a target node with the maintenance window using the AWS CLI
- Step 3: Register a task with the maintenance window using the AWS CLI

Step 1: Create the maintenance window using the 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.
> For more information about working with the `--schedule` option, see Reference: Cron and rate expressions for Systems Manager.

To create a maintenance window using the 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 similar to 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 similar to 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 using the AWS CLI.

Step 2: Register a target node with the maintenance window using the 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.

> **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.

To register a target node with a maintenance window using the AWS CLI

1. Run the following command on your local machine. Replace each *example resource placeholder* with your own information.

   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 similar to the following.

```
{
   "WindowTargetId":"e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
}
```

2. Now run the following command on your local machine to view details about your maintenance window target.

Linux & 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 similar to the following.

```
{
    "Targets": [
        {
            "WindowId": "mw-0c50858d01EXAMPLE",
            "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE",
            "ResourceType": "INSTANCE",
            "Targets": [
```

```
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-02573cafcfEXAMPLE"
                    ]
                }
            ]
        }
    ]
}
```

Continue to Step 3: Register a task with the maintenance window using the AWS CLI.

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 using the AWS CLI. 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 tool in 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. Replace each *example resource placeholder* with your own information.

Linux & macOS

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-0c50858d01EXAMPLE" \
    --resource-type "INSTANCE" \
    --target
 "Key=InstanceIds,Values=i-02573cafcfEXAMPLE,i-0471e04240EXAMPLE,i-07782c72faEXAMPLE"
```

Windows

```
aws ssm register-target-with-maintenance-window ^
    --window-id "mw-0c50858d01EXAMPLE ^
    --resource-type "INSTANCE" ^
    --target
 "Key=InstanceIds,Values=i-02573cafcfEXAMPLE,i-0471e04240EXAMPLE,i-07782c72faEXAMPLE
```

Recommended use: Most useful when registering a unique group of 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. Replace each *example resource placeholder* with your own information.

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` or `PatchGroup` and assign the nodes a common key value, such as `my-patch-group`. (You must use `PatchGroup`, without a space, if you have allowed tags in EC2 instance metadata.) Patch Manager, a tool in Systems Manager, evaluates the `Patch Group` or `PatchGroup` 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` or `PatchGroup` key-value pair when you register targets with a maintenance window. For example: `--target "Key=tag:PatchGroup,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 Patch groups.

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. Replace each *example resource placeholder* with your own information.

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. Replace *mw-0c50858d01EXAMPLE* with your own information. 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. Replace *mw-0c50858d01EXAMPLE* with your own information. 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 using the 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. 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. Replace each *example resource placeholder* with your own information. 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-02573cafcfEXAMPLE" ^
    --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"
}
```

2. 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
   ```

3. 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-02573cafcfEXAMPLE"
```

```
                    ]
                }
            ],
            "TaskParameters": {},
            "Priority": 10,
            "ServiceRoleArn": "arn:aws:iam::123456789012:role/
MyMaintenanceWindowServiceRole",
            "MaxConcurrency": "1",
            "MaxErrors": "1"
        }
    ]
}
```

4. Wait until the task has had time to run, based on the schedule you specified in Step 1: Create the maintenance window using the AWS CLI. 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. Replace *mw-0c50858d01EXAMPLE* with your own information.

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 and Maintenance window scheduling and active period options.
For information about using the AWS Command Line Interface (AWS CLI) to modify a maintenance window, see Tutorial: Update a maintenance window using the AWS CLI.

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 using the AWS CLI.

Accessing 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 tool in AWS Systems Manager, with a maintenance window using the AWS Command Line Interface (AWS CLI), as demonstrated in Register tasks with the maintenance window. 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.

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
   ```

Pseudo parameters in maintenance window tasks

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 Using pseudo parameters when registering maintenance window tasks.

More info

- Parameter options for the register-task-with-maintenance-windows command.
- 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-02573cafcfEXAMPLE" \
    --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-02573cafcfEXAMPLE" ^
    --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"
            },
            "OutputS3BucketName": "amzn-s3-demo-bucket",
            "OutputS3KeyPrefix": "S3-PREFIX",
            "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": "ew0KICAi--truncated--0KIEXAMPLE",
            "Payload": "{ \"instanceId\": \"{{RESOURCE_ID}}\", \"targetType\":
 \"{{TARGET_TYPE}}\" }",
            "Qualifier": "$LATEST"
        }
    },
    "Name": "My-Lambda-Task",
    "Description": "A description for my LAMBDA task",
    "Priority": 5
}
```

Register a Step Functions task

The following examples demonstrate how to register Step Functions state machine tasks with a maintenance window using the AWS CLI.

> **Note**
>
> Maintenance window tasks support Step Functions Standard state machine workflows only. They don't support Express state machine workflows. For information about state machine

workflow types, see Standard vs. Express Workflows in the *AWS Step Functions Developer Guide*.

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
}
```

Parameter options for the register-task-with-maintenance-windows command

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*.

Command option: `--task-arn`

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
`RUN_COMMAND` and **`AUTOMATION`**	`TaskArn` is the SSM document name or Amazon Resource Name (ARN). For example: `AWS-RunBatchShellScript` -or- `arn:aws:ssm:` *`region`*`:11112222 3333:document/My-Document` .
`LAMBDA`	`TaskArn` is the function name or ARN. For example: `SSMMy-Lambda-Function` -or- `arn:aws:lambda:` *`region`*`:11112222 3333:function:SSMMyLambdaFu nction` . **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` ,

Maintenance window task type	TaskArn value
	change it to `SSMMyLambdaFunctio n` .
`STEP_FUNCTIONS`	`TaskArn` is the state machine ARN. For example: `arn:aws:states:us-east-2:11 1122223333:stateMachine:SSM MyStateMachine` . **Important** The IAM policy for maintenance windows requires that you prefix Step Functions state machine names with `SSM`. Before you register this type of task, you must update its name in AWS Step Functions to include `SSM`. For example, if your state machine name is `MyStateMachine` , change it to `SSMMyStateMachine` .

Command option: `--service-role-arn`

The role for AWS Systems Manager to assume when running the maintenance window task.

For more information, see Setting up Maintenance Windows

Command option: `--task-invocation-parameters`

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 Using pseudo parameters when registering maintenance window tasks.

Task invocation parameters options for maintenance window tasks

Maintenance window task type	Available parameters	Example

RUN_COMMAND	Comment
	DocumentHash
	DocumentHashType
	NotificationConfig
	OutputS3BucketName
	OutPutS3KeyPrefix
	Parameters
	ServiceRoleArn
	TimeoutSeconds
		<pre>"TaskInvocationPar
		ameters": {
		"RunCommand": {
		"Comment"
		: "My Run Command task
		comment",
		"Document
		Hash": "6554ed3d--
		truncated--5EXAMPLE",
		"Document
		HashType": "Sha256",
		"Notifica
		tionConfig": {
		"Notifica
		tionArn": "arn:aws:
		sns: region:12345678
		9012:my-sns-topic-
		name",
		"NotificationEvents":
		[
		"FAILURE"
		],
		"NotificationType":
		"Invocation"
		},</pre>

Maintenance window task type	Available parameters	Example
		"OutputS3BucketName": "amzn-s3-demo-bucket", "OutputS3KeyPrefix": " *S3-PREFIX* ", "Parameters": { "commands": ["Get-ChildItem$env:temp-Recurse|Remove-Item-Recurse-force"] }, "ServiceRoleArn": "arn:aws:iam::123456789012:role/MyMaintenanceWindowServiceRole", "TimeoutSeconds": 3600 } }

Maintenance window task type	Available parameters	Example

AUTOMATION	DocumentVersion
	
	Parameters
		"TaskInvocationParameters": {
		"Automation": {
		"DocumentVersion": "3",
		"Parameters": {
		"instanceid": [
		"{{TARGET_ID}}"
		]
		}
		}
		}

LAMBDA	ClientContext
	
	Payload
	
	Qualifier
		"TaskInvocationParameters": {
		"Lambda": {
		"ClientContext": "ew0KICAi--truncated--0KIEXAMPLE",
		"Payload": "{ \"targetId\": \"{{TARGET_ID}}\", \"targetType\": \"{{TARGET_TYPE}}\" }",
		"Qualifier": "$LATEST"
		}
		}

Maintenance window task type	Available parameters	Example

STEP_FUNCTIONS	Input
	
	Name
		"TaskInvocationParameters": {
		"StepFunctions": {
		"Input": "{ \"targetId\": \"{{TARGET_ID}}\" }",
		"Name": "{{INVOCATION_ID}}"
		}
		}

Tutorial: View information about maintenance windows using the 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'
- Examples for 'describe-maintenance-window-targets'
- Examples for 'describe-maintenance-window-tasks'
- Examples for 'describe-maintenance-windows-for-target'
- Examples for 'describe-maintenance-window-executions'
- Examples for 'describe-maintenance-window-schedule'

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 similar to 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 similar to 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 similar to 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 similar to 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"
        },
        {
            "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 similar to the following.

```
{
    "Targets": [
        {
            "WindowId": "mw-0c50858d01EXAMPLE",
            "WindowTargetId": "e32eecb2-646c-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 similar to 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-02573cafcfEXAMPLE",
               "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-02573cafcfEXAMPLE",
               "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 similar to 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-02573cafcfEXAMPLE",
               "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 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-02573cafcfEXAMPLE"
                    ]
                }
            ],
            "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-02573cafcfEXAMPLE" \
    --max-results 10
```

Windows

```
aws ssm describe-maintenance-windows-for-target ^
    --resource-type INSTANCE ^
    --targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE" ^
    --max-results 10
```

The system returns information similar to 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 similar to 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 similar to 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 similar to 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 similar to 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 similar to 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 using the 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 using the AWS CLI, 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 using the 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 similar to 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 similar to 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 similar to 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 similar to 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-02573cafcfEXAMPLE"
                ]
            },
            "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 similar to 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-02573cafcfEXAMPLE\"],\"maxConcurrency\":\"1\",\"maxErrors\":\"1\",
   \"parameters\":{\"commands\":[\"df\"]}}",
               "Status": "SUCCESS",
               "StatusDetails": "Success",
               "StartTime": 1557593493.222,
               "EndTime": 1557593498.466
           }
       ]
   }
   ```

Tutorial: Update a maintenance window using the 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 Update or delete maintenance window resources using the console.

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 update a maintenance window using the 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 similar to the following.

```
{
    "WindowId": "mw-0c50858d01EXAMPLE",
    "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE",
    "Targets": [
        {
            "Key": "InstanceIds",
            "Values": [
                "i-02573cafcfEXAMPLE"
            ]
        }
    ],
    "Name": "My-Maintenance-Window-Target",
    "Description": "Description for my maintenance window target"
}
```

2. Run the following command to use the `replace` option to remove the description field and add an additional target. The description field is removed, because the update 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-02573cafcfEXAMPLE,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-02573cafcfEXAMPLE,i-0471e04240EXAMPLE" ^
       --name "My-Maintenance-Window-Target" ^
   ```

```
    --replace
```

The system returns information similar to the following.

```
{
    "WindowId": "mw-0c50858d01EXAMPLE",
    "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE",
    "Targets": [
        {
            "Key": "InstanceIds",
            "Values": [
                "i-02573cafcfEXAMPLE",
                "i-0471e04240EXAMPLE"
            ]
        }
    ],
    "Name": "My-Maintenance-Window-Target"
}
```

3. The `start-date` option allows you to delay activation of a maintenance window until a specified future date. The `end-date` option allows you to set a date and time in the future after which the maintenance window no longer runs. Specify the options in ISO-8601 Extended format.

   Run the following command to specify a date and time range for regularly scheduled maintenance window executions.

   Linux & 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 similar to the following.

```
{
```

```
    "WindowId": "mw-0c50858d01EXAMPLE",
    "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
    "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"
}
```

5. 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 similar to 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"
}
```

6. 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 similar to 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"
}
```

7. Run the following command to unregister a target from a maintenance window. This example uses the `safe` parameter to determine if the target is referenced by any tasks and therefore safe to unregister.

   Linux & 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 similar to the following.

```
An error occurred (TargetInUseException) when calling the
 DeregisterTargetFromMaintenanceWindow operation:
This Target cannot be deregistered because it is still referenced in Task:
 4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE
```

8. Run the following command to unregister a target from a maintenance window even if the target is referenced by a task. You can force the unregister operation by using the `no-safe` parameter.

   Linux & 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 similar to the following.

   ```
   {
       "WindowId": "mw-0c50858d01EXAMPLE",
       "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
   }
   ```

9. Run the following command to update a Run Command task. This example uses a Systems Manager Parameter Store parameter called `UpdateLevel`, which is formatted as follows: `'{{ssm:UpdateLevel}}'`

Linux & macOS

```
aws ssm update-maintenance-window-task \
    --window-id "mw-0c50858d01EXAMPLE" \
    --window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" \
    --targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE"  \
    --task-invocation-parameters "RunCommand={Comment=A comment for my task
 update,Parameters={UpdateLevel='{{ssm:UpdateLevel}}'}}"
```

Windows

```
aws ssm update-maintenance-window-task ^
    --window-id "mw-0c50858d01EXAMPLE" ^
    --window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" ^
    --targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE"  ^
    --task-invocation-parameters "RunCommand={Comment=A comment for my task
 update,Parameters={UpdateLevel='{{ssm:UpdateLevel}}'}}"
```

The system returns information similar to the following.

```
{
    "WindowId": "mw-0c50858d01EXAMPLE",
    "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
    "Targets": [
        {
            "Key": "InstanceIds",
            "Values": [
                "i-02573cafcfEXAMPLE"
            ]
        }
    ],
    "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"
}
```

10. Run the following command to update an Automation task to specify `WINDOW_ID` and `WINDOW_TASK_ID` parameters for the `task-invocation-parameters` parameter:

    Linux & 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 similar to 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 using the 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 similar to the following.

```
{
   "WindowId":"mw-0c50858d01EXAMPLE"
}
```

Tutorial: Create a maintenance window for patching using the console

> **Important**
>
> You can continue to use this legacy topic to create a maintenance window for patching. However, we recommend that you use a patch policy instead. For more information, see Patch policy configurations in Quick Setup and Configure patching for instances in an organization using a Quick Setup patch policy.

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.

You must configure roles and permissions for Maintenance Windows, a tool in AWS Systems Manager, before beginning this procedure. For more information, see Setting up Maintenance Windows.

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**.
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. (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. 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.

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 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 this maintenance window to identify the type of tasks it runs. In this case, you could specify the following key name/value pair:

- `Key=TaskType,Value=Patching`

14. Choose **Create maintenance window**.
15. In the maintenance windows list, choose the maintenance window you just created, and then choose **Actions**, **Register targets**.
16. (Optional) In the **Maintenance window target details** section, provide a name, a description, and owner information (your name or alias) for this target.
17. For **Target selection**, choose **Specify instance tags**.
18. For **Specify instance tags**, enter a tag key and a tag value to identify the nodes to register with the maintenance window, and then choose **Add**.
19. Choose **Register target**. The system creates a maintenance window target.
20. In the details page of the maintenance window you created, choose **Actions**, **Register Run command task**.
21. (Optional) For **Maintenance window task details**, provide a name and description for this task.
22. For **Command document**, choose `AWS-RunPatchBaseline`.
23. For **Task priority**, choose a priority. Zero (`0`) is the highest priority.
24. For **Targets**, under **Target by**, choose the maintenance window target you created earlier in this procedure.
25. 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.

26. (Optional) For **IAM service role**, choose a role to provide permissions for Systems Manager to assume when running a maintenance window task.

    If you don't specify a service role ARN, Systems Manager uses a service-linked role in your account. If no appropriate service-linked role for Systems Manager exists in your account, it's created when the task is registered successfully.

    > **Note**
    >
    > For an improved security posture, we strongly recommend creating a custom policy and custom service role for running your maintenance window tasks. The policy can be crafted to provide only the permissions needed for your particular maintenance window tasks. For more information, see Setting up Maintenance Windows.

27. (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 Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

28. 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.

29. 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`.
    - For **RebootOption**, 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`.
    - (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.

30. Choose **Register Run command task**.

After the maintenance window task is complete, you can view patch compliance details in the Systems Manager console in the Fleet Manager tool.

You can also view compliance information in the Patch Manager tool, on the **Compliance reporting** tab.

You can also use the DescribePatchGroupState and DescribeInstancePatchStatesForPatchGroup APIs to view compliance details. For information about patch compliance data, see About patch compliance.

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` or `PatchGroup` tag for the tag name, and any value for which you have defined Amazon Elastic

Compute Cloud (Amazon EC2) tags, for example, "web servers" or "US-EAST-PROD. (You must use `PatchGroup`, without a space, if you have allowed tags in EC2 instance metadata.

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 tool in 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`.)

Using pseudo parameters when registering maintenance window tasks

When you register a task in Maintenance Windows, a tool in AWS Systems Manager, you specify the parameters that are unique to each of the four task types. (In CLI commands, these are provided using the `--task-invocation-parameters` option.)

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 later in this topic in Supported pseudo parameters.

> **Important**
>
> For the target type `RESOURCE_GROUP`, depending on the ID format needed for the task, you can choose between using `{{TARGET_ID}}` and `{{RESOURCE_ID}}` to reference

the resource when your task runs. {{TARGET_ID}} returns the full ARN of the resource. {{RESOURCE_ID}} returns only a shorter name or ID of the resource, as shown in these examples.

- {{TARGET_ID}} format: arn:aws:ec2:us-east-1:123456789012:instance/i-02573cafcfEXAMPLE
- {{RESOURCE_ID}} format: i-02573cafcfEXAMPLE

For target type INSTANCE, both the {{TARGET_ID}} and {{RESOURCE_ID}} parameters yield the instance ID only. For more information, see Supported pseudo parameters. {{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": "ew0KICAi--truncated--0KIEXAMPLE",
            "Payload": "{ \"instanceId\": \"{{RESOURCE_ID}}\" }",
            "Qualifier": "$LATEST"
        }
    }
```

If your Lambda task is intended to run against another supported target type in addition to 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": "ew0KICAi--truncated--0KIEXAMPLE",
            "Payload": "{ \"tableArn\": \"{{TARGET_ID}}\" }",
            "Qualifier": "$LATEST"
        }
    }
```

The same syntax works for targeting instances or other resource types. When multiple resource types have been added to a resource group, the task runs against each of the appropriate resources.

> **Important**
>
> Not all resource types that might be included in a resource group yield a value for the `{{RESOURCE_ID}}` parameter. For a list of supported resource types, see Supported pseudo parameters.

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::CloudWatch::Alarm	arn:aws:cloudwatch:us-east-1:123456789012:alarm:MyCloudWatchAlarm i-078a280217EXAMPLE
AWS::DynamoDB::Table	arn:aws:dynamodb:us-east-1:123456789012:table/MyTable
AWS::EC2::Instance	arn:aws:ec2:us-east-1:123456789012:instance/ i-078a280217EXAMPLE
AWS::EC2::Image	arn:aws:ec2:us-east-1:123456789012:image/ami-02250b3732EXAMPLE
AWS::EC2::SecurityGroup	arn:aws:ec2:us-east-1:123456789012:security-group/sg-cEXAMPLE
AWS::EC2::Snapshot	arn:aws:ec2:us-east-1:123456789012:snapshot/snap-03866bf003EXAMPLE

Resource type	Example TARGET_ID
AWS::EC2::Volume	arn:aws:ec2:us-east-1:123456789012:volume/vol-0912e04d78EXAMPLE
AWS::ECS::Service	arn:aws:ecs:us-east-1:123456789012:service/my-ecs-service
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:::amzn-s3-demo-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::CloudWatch::Alarm	MyCloudWatchAlarm	
AWS::DynamoDB::Table	MyTable	
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::ECS::Service	my-ecs-service	
AWS::RDS::DBCluster	My-Cluster	
AWS::RDS::DBInstance	My-SQL-Instance	
AWS::S3::Bucket	amzn-s3-demo-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.

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. 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
- Example 2: Specify a maintenance window start date and end date
- Example 3: Create a maintenance window that runs only once
- Example 4: Specify the number of schedule offset days for a maintenance window

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.

More info

- Reference: Cron and rate expressions for Systems Manager
- Create a maintenance window using the console
- Tutorial: Create and configure a maintenance window using the AWS CLI
- 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.

> **Note**
>
> When a task has registered targets, Automation, AWS Lambda, and AWS Step Functions tasks resolve the targets from resource groups and tags and send one invocation per resolved resource, which results in multiple task invocations. But say, for example, that you want only one invocation for a Lambda task that's registered with a resource group containing more than one instance. In this case, if you are working in the AWS Management Console, choose the option **Task target not required** in the **Register Lambda task** or **Edit Lambda task** page. If you are using the AWS CLI command, do not specify targets using the `--targets` parameter when running the register-task-with-maintenance-window command or update-maintenance-window-task command.

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 `AWS-RestartEC2Instance` runbook.

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-02573cafcfEXAMPLE\"]}}}" \
    --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-02573cafcfEXAMPLE\"]}}}" ^
    --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.

More info

- Error messages: "Maintenance window tasks without targets don't support MaxConcurrency values" and "Maintenance window tasks without targets don't support MaxErrors values"

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."
- Not all maintenance window targets are updated
- Task fails with task invocation status: "The provided role does not contain the correct SSM permissions."
- Task fails with error message: "Step fails when it is validating and resolving the step inputs"
- Error messages: "Maintenance window tasks without targets don't support MaxConcurrency values" and "Maintenance window tasks without targets don't support MaxErrors values"

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 fix: 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.

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 fix: Select the correct name of the IAM maintenance window role you want to use, or create a new one to specify for the task.

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 task invocation status: "The provided role does not contain the correct SSM permissions."

Problem: You have specified a maintenance window service role for a task, but the task fails to run successfully and the task invocation status reports that "The provided role does not contain the correct SSM permissions."

- **Solution**: In Task 1: Create a custom policy for your maintenance window service role using the console, we provide a basic policy you can attach to your custom maintenance window service role. The policy includes the permissions needed for many task scenarios. However, due to the wide variety of tasks you can run, you might need to provide additional permissions in the policy for your maintenance window role.

For example, some Automation actions work with AWS CloudFormation stacks. Therefore, you might need to add the additional permissions `cloudformation:CreateStack`, `cloudformation:DescribeStacks`, and `cloudformation:DeleteStack` to the policy for your maintenance window service role.

For another example, the Automation runbook `AWS-CopySnapshot` requires permissions to create an Amazon Elastic Block Store (Amazon EBS) snapshot. Therefore, you might need to add the permission `ec2:CreateSnapshot`.

For information about the role permissions needed by an AWS managed Automation runbook, see the runbook descriptions in the AWS Systems Manager Automation Runbook Reference.

For information about the role permissions needed by an AWS managed SSM document, review the content of the document in the Documents section Systems Manager console.

For information about the role permissions needed for Step Functions tasks, Lambda tasks, and custom Automation runbooks and SSM documents, verify permission requirements with the author of those resources.

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 Using pseudo parameters when registering maintenance window tasks and Task registration examples.

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 in the *AWS Systems Manager User Guide*.

AWS Systems Manager Quick Setup

Use Quick Setup, a tool in 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. There is no cost to use Quick Setup. However, costs can be incurred based on the type of services you set up and the usage limits with no fees for the services used to set up your service. To get started with Quick Setup, open the Systems Manager console. In the navigation pane, choose **Quick Setup**.

> **Note**
>
> If you were directed to Quick Setup to help you configure your instances to be managed by Systems Manager, complete the procedure in Set up Amazon EC2 host management using 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.

Availability of Quick Setup in AWS Regions

In the following AWS Regions, you can use all Quick Setup configuration types for an entire organization, as configured in AWS Organizations, or for only the organizational accounts and Regions you choose. You can also use Quick Setup with just a single account in these Regions.

- 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 (Stockholm)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- South America (São Paulo)

In the following Regions, only the Host Management configuration type is available for individual accounts:

- Europe (Milan)
- Asia Pacific (Hong Kong)
- Middle East (Bahrain)
- China (Beijing)
- China (Ningxia)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

For a list of all supported Regions for Systems Manager, see the **Region** column in Systems Manager service endpoints in the *Amazon Web Services General Reference*.

Getting started with Quick Setup

Use the information in this topic to help you prepare to use Quick Setup.

Topics

- IAM roles and permissions for Quick Setup onboarding
- Manual onboarding for working with Quick Setup API programmatically

IAM roles and permissions for Quick Setup onboarding

Quick Setup launched a new console experience and a new API. Now you can interact with this API using the console, AWS CLI, AWS CloudFormation, and SDKs. If you opt in to the new experience, your existing configurations are recreated using the new API. Depending on the number of existing configurations in your account, this process can take several minutes.

To use the new Quick Setup console, you must have permissions for the following actions:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm-quicksetup:*",
                "cloudformation:DescribeStackSetOperation",
                "cloudformation:ListStacks",
                "cloudformation:DescribeStacks",
                "cloudformation:DescribeStackResources",
                "cloudformation:ListStackSetOperations",
                "cloudformation:ListStackInstances",
                "cloudformation:DescribeStackSet",
                "cloudformation:ListStackSets",
                "cloudformation:DescribeStackInstance",
                "cloudformation:DescribeOrganizationsAccess",
                "cloudformation:ActivateOrganizationsAccess",
                "cloudformation:GetTemplate",
                "cloudformation:ListStackSetOperationResults",
                "cloudformation:DescribeStackEvents",
                "cloudformation:UntagResource",
                "ec2:DescribeInstances",
                "ssm:DescribeAutomationExecutions",
                "ssm:GetAutomationExecution",
                "ssm:ListAssociations",
                "ssm:DescribeAssociation",
                "ssm:GetDocument",
                "ssm:ListDocuments",
                "ssm:DescribeDocument",
                "ssm:ListResourceDataSync",
```

```
                "ssm:DescribePatchBaselines",
                "ssm:GetPatchBaseline",
                "ssm:DescribeMaintenanceWindows",
                "ssm:DescribeMaintenanceWindowTasks",
                "ssm:GetOpsSummary",
                "organizations:DeregisterDelegatedAdministrator",
                "organizations:DescribeAccount",
                "organizations:DescribeOrganization",
                "organizations:ListDelegatedAdministrators",
                "organizations:ListRoots",
                "organizations:ListParents",
                "organizations:ListOrganizationalUnitsForParent",
                "organizations:DescribeOrganizationalUnit",
                "organizations:ListAWSServiceAccessForOrganization",
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "resource-groups:ListGroups",
                "iam:ListRoles",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:CreatePolicy",
                "organizations:RegisterDelegatedAdministrator",
                "organizations:EnableAWSServiceAccess",
                "cloudformation:TagResource"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:RollbackStack",
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack",
                "cloudformation:DeleteStack"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
                "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
                "arn:aws:cloudformation:*:*:type/resource/*",
                "arn:aws:cloudformation:*:*:stack/StackSet-SSMQuickSetup"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStackSet",
                "cloudformation:UpdateStackSet",
                "cloudformation:DeleteStackSet",
                "cloudformation:DeleteStackInstances",
                "cloudformation:CreateStackInstances",
                "cloudformation:StopStackSetOperation"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
                "arn:aws:cloudformation:*:*:stackset/SSMQuickSetup",
                "arn:aws:cloudformation:*:*:type/resource/*",
                "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "iam:GetRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/AWS-QuickSetup-*",
                "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/AWS-QuickSetup-*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ssm-quicksetup.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ssm:DeleteAssociation",
                "ssm:CreateAssociation",
                "ssm:StartAssociationsOnce"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:StartAutomationExecution",
            "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-
EnableExplorer:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetOpsSummary",
                "ssm:CreateResourceDataSync",
                "ssm:UpdateResourceDataSync"
            ],
            "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "accountdiscovery.ssm.amazonaws.com",
                        "ssm.amazonaws.com",
                        "ssm-quicksetup.amazonaws.com",
                        "stacksets.cloudformation.amazonaws.com"
                    ]
                }
            },
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
```

```
            "Resource": "arn:aws:iam::*:role/aws-
service-role/stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin"
        }
    ]
}
```

To restrict users to read-only permissions, only allow `ssm-quicksetup:List*` and `ssm-quicksetup:Get*` operations for the Quick Setup API.

During onboarding, Quick Setup creates the following AWS Identity and Access Management (IAM) roles on your behalf:

- `AWS-QuickSetup-LocalExecutionRole` – Grants AWS CloudFormation permissions to use any template, excluding the patch policy template, and create the necessary resources.
- `AWS-QuickSetup-LocalAdministrationRole` – Grants permissions to AWS CloudFormation to assume `AWS-QuickSetup-LocalExecutionRole`.
- `AWS-QuickSetup-PatchPolicy-LocalExecutionRole` – Grants permissions to AWS CloudFormation to use the patch policy template, and create the necessary resources.
- `AWS-QuickSetup-PatchPolicy-LocalAdministrationRole` – Grants permissions to AWS CloudFormation to assume `AWS-QuickSetup-PatchPolicy-LocalExecutionRole`.

If you're onboarding a management account—the account that you use to create an organization in AWS Organizations—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 tool in 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 Using roles to collect AWS account information for OpsCenter and Explorer.

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.

For information about AWS Organizations account types, see AWS Organizations terminology and concepts in the *AWS Organizations User Guide*.

> **Note**
>
> Quick Setup uses AWS CloudFormation StackSets to deploy your configurations across AWS accounts and Regions. If the number of target accounts multiplied by the number of Regions exceeds 10,000, the configuration fails to deploy. We recommend reviewing your use case and creating configurations that use fewer targets to accommodate the growth of your organization. Stack instances aren't deployed to your organization's management account. For more information, see Considerations when creating a stack set with service-managed permissions.

Manual onboarding for working with Quick Setup API programmatically

If you use the console to work with Quick Setup, the service handles onboarding steps for you. If you plan to use SDKs or the AWS CLI to work with the Quick Setup API, you can still use the console to complete onboarding steps for you so you don't have to perform them manually. However, some customers need to complete onboarding steps for Quick Setup programmatically without interacting with the console. If this method fits your use case, you must complete the following steps. All of these steps must be completed from your AWS Organizations management account.

To complete manual onboarding for Quick Setup

1. Activate trusted access for AWS CloudFormation with Organizations. This provides the management account with the permissions needed to create and manage StackSets for your organization. You can use AWS CloudFormation's `ActivateOrganizationsAccess` API action to complete this step. For more information, see ActivateOrganizationsAccess in the *AWS CloudFormation API Reference*.
2. Enable the integration of Systems Manager with Organizations. This allows Systems Manager to create a service-linked role in all the accounts in your organization. This also allows Systems

Manager to perform operations on your behalf in your organization and its accounts. You can use AWS Organizations's `EnableAWSServiceAccess` API action to complete this step. The service principal for Systems Manager is `ssm.amazonaws.com`.For more information, see EnableAWSServiceAccess in the *AWS Organizations API Reference*.

3. Create the required IAM role for Explorer. This allows Quick Setup to create dashboards for your configurations so you can view deployment and association statuses. Create an IAM role and attach the `AWSSystemsManagerEnableExplorerExecutionPolicy` managed policy. Modify the trust policy for the role to match the following. Replace each *`account ID`* with your information.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ssm.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:*:ssm:*:111122223333:automation-
execution/*"
                }
            }
        }
    ]
}
```

4. Update the Quick Setup service setting for Explorer. You can use Quick Setup's `UpdateServiceSettings` API action to complete this step. Specify the ARN for the IAM role you created in the previous step for the `ExplorerEnablingRoleArn` request parameter. For more information, see UpdateServiceSettings in the *Quick Setup API Reference*.

5. Create the required IAM roles for AWS CloudFormation StackSets to use. You must create an *execution* role and an *administration* role.

   a. Create the execution role. The execution role should have at least one of the `AWSQuickSetupDeploymentRolePolicy` or `AWSQuickSetupPatchPolicyDeploymentRolePolicy` managed policies attached. If you're only creating patch policy configurations, you can use `AWSQuickSetupPatchPolicyDeploymentRolePolicy` managed policy. All other configurations use the `AWSQuickSetupDeploymentRolePolicy` policy. Modify the trust policy for the role to match the following. Replace each *`account ID`* and *`administration role name`* with your information.

      JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/administration
 role name"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   b. Create the administration role. The permissions policy must match the following. Replace each *`account ID`* and *`execution role name`* with your information.

      JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
```

```
            "Resource": "arn:*:iam::111122223333:role/execution role
 name",
            "Effect": "Allow"
        }
    ]
}
```

Modify the trust policy for the role to match the following. Replace each *account ID* with your information.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudformation.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn":
 "arn:aws:cloudformation:*:111122223333:stackset/AWS-QuickSetup-*"
                }
            }
        }
    ]
}
```

Using a delegated administrator for Quick Setup

After you register a delegated administrator account for Quick Setup, users with the appropriate permissions in that account can create, update, view, and delete configuration managers that target organizational units within your AWS Organizations structure. This delegated administrator

account can also manage configuration managers previously created by your organization's management account.

The management account in Organizations can designate one account within your organization as a delegated administrator. When you register an account as a delegated administrator for Quick Setup, this account automatically becomes a delegated administrator for AWS CloudFormation StackSets and Systems Manager Explorer as well, since these services are required to deploy and monitor Quick Setup configurations.

> **Note**
>
> At this time, the patch policy configuration type isn't supported by the delegated administrator for Quick Setup. Patch policy configurations for an organization must be created and maintained in the management account for an organization. For more information, see the section called “Creating a patch policy”.

The following topics describe how to register and deregister a delegated administrator for Quick Setup.

Topics

- Register a delegated administrator for Quick Setup
- Deregister a delegated administrator for Quick Setup

Register a delegated administrator for Quick Setup

Use the following procedure to register a delegated administrator for Quick Setup.

To register a Quick Setup 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 **Quick Setup**.
4. Choose **Settings**.

5. In the **Delegated administrator for Quick Setup** 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**.

Deregister a delegated administrator for Quick Setup

Use the following procedure to deregister a delegated administrator for Quick Setup.

To deregister a Quick Setup 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 **Quick Setup**.
4. Choose **Settings**.
5. In the **Delegated administrator for Quick Setup** section, choose **Deregister** from the **Actions** dropdown.
6. Select **Confirm**.

Learn Quick Setup terminology and details

Quick Setup, a tool in AWS Systems Manager, displays the results of all configuration managers you've created across all AWS Regions in the **Configuration managers** 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 **Configuration managers** table contains the following information:

- **Name** – The name of the configuration manager if provided when created.
- **Configuration type** – The configuration type chosen when creating the configuration.
- **Version** – The version of the configuration type currently deployed.
- **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. The management account is the account that you use to create an organization in AWS Organizations.

- **Deployment type** – Indicates whether the deployment applies to the entire organization (`Organizational`) or only your account (`Local`).
- **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 tool in AWS Systems Manager.

To view configurations deployed to the Region you're currently browsing, select the **Configurations** tab.

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.

Depending on the type of configuration, one or more of the following status graphs are displayed:

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.

Setup status

Displays the number of actions performed by the configuration type and their current statuses.

Resource compliance

Displays the number of resources that are compliant to the configurations specified policy.

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.

Periodically, Quick Setup releases new versions of configurations. You can select the **Upgrade configuration** option to upgrade your configuration to the latest version.

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 tools in AWS Systems Manager. To learn more about compliance, see Learn details about Compliance. To learn more about viewing compliance in Explorer, see AWS Systems Manager Explorer.

Using the Quick Setup API to manage configurations and deployments

You can use the API provided by Quick Setup to create and manage configurations and deployments using the AWS CLI or your preferred SDK. You can also use AWS CloudFormation to create a configuration manager resource that deploys configurations. Using the API, you create configuration managers that deploy configuration *definitions*. Configuration definitions contain all of the necessary information to deploy a particular configuration type. For more information about the Quick Setup API, see the Quick Setup API Reference.

The following examples demonstrate how to create configuration managers using the AWS CLI and AWS CloudFormation.

AWS CLI

```
aws ssm-quicksetup create-configuration-manager \
--name configuration manager name \
--description Description of your configuration manager
--configuration-definitions JSON string containing configuration defintion
```

The following is an example JSON string containing a configuration definition for patch policy.

```
'{"Type":"AWSQuickSetupType-
PatchPolicy","LocalDeploymentAdministrationRoleArn":"arn:aws:iam::123456789012:role/
AWS-QuickSetup-StackSet-Local-
AdministrationRole","LocalDeploymentExecutionRoleName":"AWS-
QuickSetup-StackSet-Local-ExecutionRole","Parameters":
{"ConfigurationOptionsInstallNextInterval":"true","ConfigurationOptionsInstallValue":"cron(0
 2 ? * SAT#1
 *)","ConfigurationOptionsPatchOperation":"ScanAndInstall","ConfigurationOptionsScanNextInte
 1 * * ?
 *)","HasDeletedBaseline":"false","IsPolicyAttachAllowed":"true","OutputBucketRegion":"","Ou
```

```
east-1","PatchBaselineUseDefault":"custom","PatchPolicyName":"dev-patch-
policy","RateControlConcurrency":"5","RateControlErrorThreshold":"0%","RebootOption":"Reboot
\":{\"value\":\"arn:aws:ssm:us-east-1:123456789012:patchbaseline/
pb-0cb0c4966f86b059b\",\"label\":\"AWS-AlmaLinuxDefaultPatchBaseline\",\"description
\":\"Default Patch Baseline for Alma Linux Provided by AWS.\",\"disabled\":false},
\"AMAZON_LINUX_2\":{\"value\":\"arn:aws:ssm:us-east-1:123456789012:patchbaseline/
pb-0be8c61cde3be63f3\",\"label\":\"AWS-AmazonLinux2DefaultPatchBaseline\",
\"description\":\"Baseline containing all Security and Bugfix updates approved
 for Amazon Linux 2 instances\",\"disabled\":false},\"AMAZON_LINUX_2023\":{\"value
\":\"arn:aws:ssm:us-east-1:123456789012:patchbaseline/pb-05c9c9bf778d4c4d0\",
\"label\":\"AWS-AmazonLinux2023DefaultPatchBaseline\",\"description\":\"Default
 Patch Baseline for Amazon Linux 2023 Provided by AWS.\",\"disabled\":false},
\"DEBIAN\":{\"value\":\"arn:aws:ssm:us-east-1:123456789012:patchbaseline/
pb-09a5f8eb62bde80b1\",\"label\":\"AWS-DebianDefaultPatchBaseline\",\"description
\":\"Default Patch Baseline for Debian Provided by AWS.\",\"disabled\":false},
\"MACOS\":{\"value\":\"arn:aws:ssm:us-east-1:123456789012:patchbaseline/
pb-0ee4f94581368c0d4\",\"label\":\"AWS-MacOSDefaultPatchBaseline\",\"description
\":\"Default Patch Baseline for MacOS Provided by AWS.\",\"disabled\":false},
\"ORACLE_LINUX\":{\"value\":\"arn:aws:ssm:us-east-1:123456789012:patchbaseline/
pb-06bff38e95fe85c02\",\"label\":\"AWS-OracleLinuxDefaultPatchBaseline\",
\"description\":\"Default Patch Baseline for Oracle Linux Server Provided
 by AWS.\",\"disabled\":false},\"REDHAT_ENTERPRISE_LINUX\":{\"value\":
\"arn:aws:ssm:us-east-1:123456789012:patchbaseline/pb-0cbb3a633de00f07c\",
\"label\":\"AWS-RedHatDefaultPatchBaseline\",\"description\":\"Default Patch
 Baseline for Redhat Enterprise Linux Provided by AWS.\",\"disabled\":false},
\"ROCKY_LINUX\":{\"value\":\"arn:aws:ssm:us-east-1:123456789012:patchbaseline/
pb-03ec98bc512aa3ac0\",\"label\":\"AWS-RockyLinuxDefaultPatchBaseline\",
\"description\":\"Default Patch Baseline for Rocky Linux Provided by AWS.\",
\"disabled\":false},\"UBUNTU\":{\"value\":\"pb-06e3563bd35503f2b\",\"label\":
\"custom-UbuntuServer-Blog-Baseline\",\"description\":\"Default Patch Baseline
 for Ubuntu Provided by AWS.\",\"disabled\":false},\"WINDOWS\":{\"value\":
\"pb-016889927b2bb8542\",\"label\":\"custom-WindowsServer-Blog-Baseline\",
\"disabled\":false}}","TargetInstances":"","TargetOrganizationalUnits":"ou-9utf-
example","TargetRegions":"us-east-1,us-
east-2","TargetTagKey":"Patch","TargetTagValue":"true","TargetType":"Tags"}}' \
```

AWS CloudFormation

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
SSMQuickSetupTestConfigurationManager:
Type: "AWS::SSMQuickSetup::ConfigurationManager"
Properties:
```

```
    Name: "MyQuickSetup"
    Description: "Test configuration manager"
    ConfigurationDefinitions:
    - Type: "AWSQuickSetupType-CFGRecording"
      Parameters:
        TargetAccounts:
            Ref: AWS::AccountId
        TargetRegions:
            Ref: AWS::Region
        LocalDeploymentAdministrationRoleArn: !Sub "arn:aws:iam::
${AWS::AccountId}:role/AWS-QuickSetup-StackSet-ContractTest-AdministrationRole"
        LocalDeploymentExecutionRoleName: "AWS-QuickSetup-StackSet-ContractTest-
ExecutionRole"
    Tags:
        foo1: "bar1"
```

Supported Quick Setup configuration types

Supported configuration types

Quick Setup walks you through configuring operational best practices for a number of Systems Manager and other AWS services, and automatically deploying those configurations. The Quick Setup dashboard displays a real-time view of your configuration deployment status.

You can use Quick Setup in an individual AWS account or across multiple AWS accounts and Regions by integrating with AWS Organizations. Using Quick Setup across multiple accounts helps to ensure that your organization maintains consistent configurations.

Quick Setup provides support for the following configuration types.

- Set up Amazon EC2 host management using Quick Setup
- Set up the Default Host Management Configuration for an organization using Quick Setup
- Create an AWS Config configuration recorder using Quick Setup
- Deploy AWS Config conformance pack using Quick Setup
- Configure patching for instances in an organization using a Quick Setup patch policy
- Change Manager organization setup
- Set up DevOps Guru using Quick Setup
- Deploy Distributor packages using Quick Setup

- Stop and start EC2 instances automatically on a schedule using Quick Setup
- OpsCenter organization setup
- Configure AWS Resource Explorer using Quick Setup

Set up Amazon EC2 host management using Quick Setup

Use Quick Setup, a tool in AWS Systems Manager, to quickly configure required security roles and commonly used Systems Manager tools 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 tools 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?.

Important

Quick Setup might not be the right tool to use for EC2 management if either of the following applies to you:

- You're trying to create an EC2 instance for the first time to try out AWS capabilities.
- You're still new to EC2 instance management.

Instead, we recommend that you explore the following content:

- Getting Started with Amazon EC2
- Launch an instance using the new launch instance wizard in the *Amazon EC2 User Guide*
- Tutorial: Get started with Amazon EC2 Linux instances in the *Amazon EC2 User Guide*

If you're already familiar with EC2 instance management and want to streamline configuration and management for multiple EC2 instances, use Quick Setup. Whether your organization has dozens, thousands, or millions of EC2 instances, use the following Quick Setup procedure to configure multiple options for them, all at once.

> **Note**
>
> This configuration type lets you set multiple options for an entire organization defined in AWS Organizations, only some organizational accounts and Regions, or a single account. One of these options is to check for and apply updates to SSM Agent every two weeks. If you are an organization administrator, you can also choose to update *all* EC2 instances in your organization with agent updates every two weeks using the Default Host Management Configuration type. For information, see Set up the Default Host Management Configuration for an organization using Quick Setup.

Configuring host management options for EC2 instances

To set up host management, perform the following tasks in the AWS Systems Manager Quick Setup console.

To open the Host Management configuration page

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Quick Setup**.
3. On the **Host Management** card, choose **Create**.

> **Tip**
>
> If you already have one or more configurations in your account, first choose the **Library** tab or the **Create** button in the **Configurations** section to view the cards.

To configure Systems Manager host management options

- To configure Systems Manager functionality, in the **Configuration options** section, choose the options in the **Systems Manager** group that you want to enable 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. Quick Setup doesn't install the agent on instances where it's not already present. For information about which AMIs have SSM Agent preinstalled, see Find AMIs with the SSM Agent preinstalled.

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.

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 tool in AWS Systems Manager, see AWS Systems Manager Inventory.

> **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 tool in 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 information about patching operations and patch baselines, see AWS Systems Manager Patch Manager.

For information about patch compliance, see the Systems Manager Compliance page.

For information about patching managed nodes in multiple accounts and Regions in one configuration, see Patch policy configurations in Quick Setup and Configure patching for instances in an organization using a Quick Setup patch policy.

> **Important**
>
> Systems Manager supports several methods for scanning managed nodes for patch compliance. If you implement more than one of these methods at a time, the patch compliance information you see is always the result of the most recent scan. Results from previous scans are overwritten. If the scanning methods use different patch baselines, with different approval rules, the patch compliance information can change unexpectedly. For more information, see Avoiding unintentional patch compliance data overwrites.

To configure Amazon CloudWatch host management options

- To configure CloudWatch functionality, in the **Configuration options** section, choose the options in the **Amazon CloudWatch** group that you want to enable for your configuration:

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.

To configure Amazon EC2 Launch Agent host management options

- To configure Amazon EC2 Launch Agent functionality, in the **Configuration options** section, choose the options in the **Amazon EC2 Launch Agent** group that you want to enable for your configuration:

Update the EC2 launch agent once every 30 days

Enables Systems Manager to check every 30 days for a new version of the launch agent installed on your instance. If a new version is available, 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 applicable launch agent. For Amazon EC2 Windows instances, this option supports EC2Launch, EC2Launch v2, and EC2Config. For Amazon EC2 Linux instances, this option supports `cloud-init`. For Amazon EC2 Mac instances, this option supports `ec2-macos-init`. Quick Setup doesn't support updating launch agents that are installed on operating systems not supported by the launch agent, or on AL2023.

For more information about these initialization agents see the following topics:

- Configure a Windows instance using EC2Launch v2
- Configure a Windows instance using EC2Launch
- Configure a Windows instance using the EC2Config service
- cloud-init Documentation
- ec2-macos-init

To select the EC2 instances to be updated by the host management configuration

- In the **Targets** section, choose the method to determine the accounts and Regions where the configuration is to be deployed:

  > **Note**
  >
  > You can't create multiple Quick Setup Host Management configurations that target the same AWS Region.

  Entire organization

  Your configuration is deployed to all organizational units (OUs) and AWS Regions in your organization.

  > **Note**
  >
  > The **Entire organization** option is only available if you're configuring host management from your organization's management account.

  Custom

  1. In the **Target OUs** section, select the OUs where you want to deploy this host management configuration.
  2. In the **Target Regions** section, select the Regions where you want to deploy this host management configuration.

  Current account

  Choose one of the Region options and follow the steps for that option.

  Current Region

  Choose how to target instances in the current Region only:

- **All instances** – The host management configuration automatically targets every EC2 in the current Region.
- **Tag** – Choose **Add** and enter the key and optional value that is added to the instances to be targeted.
- **Resource group** – For **Resource group**, select an existing resource group that contains the EC2 instances to be targeted.
- **Manual** – In the **Instances** section, select the check box of each EC2 instance to be targeted.

Choose Regions

Choose how to target instances in the Region you specify by choosing one of the following:

- **All instances** – All instances in the Regions you specify are targeted.
- **Tag** – Choose **Add** and enter the key and optional value that has been added to the instances to be targeted.

In the **Target Regions** section, select the Regions where you want to deploy this host management configuration.

To specify an instance profile option

- ***Entire organization*** *and* ***Custom*** *targets only.*

  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.

After specifying all your configuration choices, choose **Create**.

Set up the Default Host Management Configuration for an organization using Quick Setup

With Quick Setup, a tool in AWS Systems Manager, you can activate Default Host Management Configuration for all accounts and Regions that have been added to your organization in AWS Organizations. This ensures that SSM Agent is kept up to date on all Amazon Elastic Compute Cloud (EC2) instances in the organization, and that they can connect to Systems Manager.

Before you begin

Ensure that the following requirements are met before enabling this setting.

- The latest version of SSM Agent is already installed on all EC2 instances to be managed in your organization.
- Your EC2 instances to be managed are using Instance Metadata Service Version 2 (IMDSv2).
- You are signed in to the management account for your organization, as specified in AWS Organizations, using an AWS Identity and Access Management (IAM) identity (user, role, or group) with administrator permissions.

Using the default EC2 instance management role

Default Host Management Configuration makes use of the `default-ec2-instance-management-role` service setting for Systems Manager. This is a role with permissions that you want made available to all accounts in your organization to allow communication between SSM Agent on the instance and the Systems Manager service in the cloud.

If you have already set this role using the update-service-setting CLI command, Default Host Management Configuration uses that role. If you have not set this role yet, Quick Setup will create and apply the role for you.

To check whether this role has already been specified for your organization, use the get-service-setting command.

Enable automatic updates of SSM Agent every two weeks

Use the following procedure to enable the Default Host Management Configuration option for your entire AWS Organizations organization.

To enable automatic updates of SSM Agent every two weeks

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Quick Setup**.
3. On the **Default Host Management Configuration** card, choose **Create**.

> **Tip**
>
> If you already have one or more configurations in your account, first choose the **Library** tab or the **Create** button in the **Configurations** section to view the cards.

4. In the **Configuration options** section, select **Enable automatic updates of SSM Agent every two weeks**.
5. Choose **Create**

Create an AWS Config configuration recorder using Quick Setup

With Quick Setup, a tool in 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**
>
> If you've already created a configuration recorder, Quick Setup doesn't stop recording or make any changes to resource types that you're already recording. If you choose to record additional resource types using Quick Setup, the service appends them to your existing recorder groups. 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**.
3. On the **Config Recording** card, choose **Create**.

   > **Tip**
   >
   > If you already have one or more configurations in your account, first choose the **Library** tab or the **Create** button in the **Configurations** section to view the cards.

4. In the **Configuration options** section, do the following:

   a. For **Choose the AWS resource types to record**, specify whether to record all supported resources or only the resource types you choose.

   b. For **Delivery settings, specify whether to create a new Amazon Simple Storage Service (Amazon S3) bucket, or choose an existing bucket to send configuration snapshots to.**

   c. For **Notification options**, 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.

5. 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**.
6. In the **Targets** section, choose one of the following to identify the accounts and Regions for recording.

> **Note**
>
> If you are working in a single account, options for working with organizations and organizational units (OUs) are not available. You can choose whether to apply this configuration to all AWS Regions in your account or only the Regions you select.

- **Entire organization** – All accounts and Regions in your organization.
- **Custom** – Only the OUs and Regions that you specify.
  - In the **Target OUs** section, select the OUs where you want to allow recording.
  - In the **Target Regions** section, select the Regions where you want to allow recording.
- **Current account** – Only the Regions you specify in the account you are currently signed into are targeted. Choose one of the following:
  - **Current Region** – Only managed nodes in the Region selected in the console are targeted.
  - **Choose Regions** – Choose the individual Regions to apply the recording configuration to.

7. Choose **Create**.

Deploy AWS Config conformance pack using Quick Setup

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**.
3. On the **Conformance Packs** card, choose **Create**.

   > **Tip**
   >
   > If you already have one or more configurations in your account, first choose the **Library** tab or the **Create** button in the **Configurations** section to view the cards.

4. In the **Choose conformance packs** section, choose the conformance packs you want to deploy.
5. 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**.
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 patching for instances in an organization using a Quick Setup patch policy

With Quick Setup, a tool in AWS Systems Manager, you can create patch policies powered by Patch Manager. A patch policy defines the schedule and baseline to use when automatically patching your Amazon Elastic Compute Cloud (Amazon EC2) instances and other managed nodes. Using a single patch policy configuration, you can define patching for all accounts in multiple AWS Regions in your organization, for only the accounts and Regions you choose, or for a single account-Region pair. For more information about patch policies, see Patch policy configurations in Quick Setup.

Prerequisite

To define a patch policy for a node using Quick Setup, the node must be a *managed node*. For more information about managing your nodes, see Setting up Systems Manager unified console for an organization.

Important

Patch compliance scanning methods – Systems Manager supports several methods for scanning managed nodes for patch compliance. If you implement more than one of these methods at a time, the patch compliance information you see is always the result of the most recent scan. Results from previous scans are overwritten. If the scanning methods use different patch baselines, with different approval rules, the patch compliance information can change unexpectedly. For more information, see Avoiding unintentional patch compliance data overwrites.

Association compliance status and patch policies – The patching status for a managed node that's under a Quick Setup patch policy matches the status of the State Manager association execution for that node. If the association execution status is `Compliant`, the patching status for the managed node is also marked `Compliant`. If the association execution status is `Non-Compliant`, the patching status for the managed node is also marked `Non-Compliant`.

Supported Regions for patch policy configurations

Patch policy configurations in Quick Setup are currently supported in the following Regions:

- US East (Ohio) (us-east-2)
- US East (N. Virginia) (us-east-1)
- US West (N. California) (us-west-1)
- US West (Oregon) (us-west-2)
- Asia Pacific (Mumbai) (ap-south-1)
- Asia Pacific (Seoul) (ap-northeast-2)
- Asia Pacific (Singapore) (ap-southeast-1)
- Asia Pacific (Sydney) (ap-southeast-2)
- Asia Pacific (Tokyo) (ap-northeast-1)
- Canada (Central) (ca-central-1)
- Europe (Frankfurt) (eu-central-1)

- Europe (Ireland) (eu-west-1)
- Europe (London) (eu-west-2)
- Europe (Paris) (eu-west-3)
- Europe (Stockholm) (eu-north-1)
- South America (São Paulo) (sa-east-1)

Permissions for the patch policy S3 bucket

When you create a patch policy, Quick Setup creates an Amazon S3 bucket that contains a file named `baseline_overrides.json`. This file stores information about the patch baselines that you specified for your patch policy.

The S3 bucket is named in the format `aws-quicksetup-patchpolicy-`*`account-id`*`-`*`quick-setup-configuration-id`*.

For example: `aws-quicksetup-patchpolicy-123456789012-abcde`

If you're creating a patch policy for an organization, the bucket is created in your organization's management account.

There are two use cases when you must provide other AWS resources with permission to access this S3 bucket using AWS Identity and Access Management (IAM) policies:

- Case 1: Use your own instance profile or service role with your managed nodes instead of one provided by Quick Setup
- Case 2: Use VPC endpoints to connect to Systems Manager

The permissions policy you need in either case is located in the section below, Policy permissions for Quick Setup S3 buckets.

Case 1: Use your own instance profile or service role with your managed nodes instead of one provided by Quick Setup

Patch policy configurations include an option to **Add required IAM policies to existing instance profiles attached to your instances**.

If you don't choose this option but want Quick Setup to patch your managed nodes using this patch policy, you must ensure that the following are implemented:

- The IAM managed policy `AmazonSSMManagedInstanceCore` must be attached to the IAM instance profile or IAM service role that's used to provide Systems Manager permissions to your managed nodes.
- You must add permissions to access your patch policy bucket as an inline policy to the IAM instance profile or IAM service role. You can provide wildcard access to all `aws-quicksetup-patchpolicy` buckets or only the specific bucket created for your organization or account, as shown in the earlier code samples.
- You must tag your IAM instance profile or IAM service role with the following key-value pair.

  `Key: QSConfigId-`*`quick-setup-configuration-id`*`, Value:` *`quick-setup-configuration-id`*

  `quick-setup-configuration-id` represents the value of the parameter applied to the AWS CloudFormation stack that is used in creating your patch policy configuration. To retrieve this ID, do the following:

  1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
  2. Select the name of the stack that is used to create your patch policy. The name is in a format such as `StackSet-AWS-QuickSetup-PatchPolicy-LA-q4bkg-52cd2f06-d0f9-499e-9818-d887cEXAMPLE`.
  3. Choose the **Parameters** tab.
  4. In the **Parameters** list, in the **Key** column, locate the key **QSConfigurationId**. In the **Value** column for its row, locate the configuration ID, such as `abcde`.

     In this example, for the tag to apply to your instance profile or service role, the key is `QSConfigId-abcde`, and the value is `abcde`.

For information about adding tags to an IAM role, see Tagging IAM roles and Managing tags on instance profiles (AWS CLI or AWS API) in the *IAM User Guide*.

Case 2: Use VPC endpoints to connect to Systems Manager

If you use VPC endpoints to connect to Systems Manager, your VPC endpoint policy for S3 must allow access to your Quick Setup patch policy S3 bucket.

For information about adding permissions to a VPC endpoint policy for S3, see Controlling access from VPC endpoints with bucket policies in the *Amazon S3 User Guide*.

Policy permissions for Quick Setup S3 buckets

You can provide wildcard access to all `aws-quicksetup-patchpolicy` buckets or only the specific bucket created for your organization or account. To provide the necessary permissions for the two cases described below, use either format.

All patch policy buckets

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessToAllPatchPolicyRelatedBuckets",
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::aws-quicksetup-patchpolicy-*"
    }
  ]
}
```

Specific patch policy bucket

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessToMyPatchPolicyRelatedBucket",
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::aws-quicksetup-patchpolicy-111122223333-quick-
setup-configuration-id"
    }
  ]
}
```

> **Note**
>
> After the patch policy configuration is created, you can locate the full name of your bucket in the S3 console. For example: `aws-quicksetup-patchpolicy-123456789012-abcde`

Random patch baseline IDs in patch policy operations

Patching operations for patch policies utilize the `BaselineOverride` parameter in the `AWS-RunPatchBaseline` SSM Command document.

When you use `AWS-RunPatchBaseline` for patching *outside* of a patch policy, you can use `BaselineOverride` to specify a list of patch baselines to use during the operation that are different from the specified defaults. You create this list in a file named `baseline_overrides.json` and manually add it to an Amazon S3 bucket that you own, as explained in Using the BaselineOverride parameter.

For patching operations based on patch policies, however, Systems Manager automatically creates an S3 bucket and adds a `baseline_overrides.json` file to it. Then, every time Quick Setup runs a patching operation (using the Run Command tool, the system generates a random ID for each patch baseline. This ID is different for every patch policy patching operation, and the patch baseline it represents is not stored or accessible to you in your account.

As a result, you will not see the ID of the patch baseline selected in your configuration in patching logs. This applies to both AWS managed patch baselines and custom patch baselines you might have selected. The baseline ID reported in the log is instead that one that was generated for that specific patching operation.

In addition, if you attempt to view details in Patch Manager about a patch baseline that was generated with a random ID, the system reports that the patch baseline doesn't exist. This is expected behavior and can be ignored.

Creating a patch policy

To create a patch policy, perform the following tasks in the Systems Manager console.

To create a patch policy with Quick Setup

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.

   If you are setting up patching for an organization, make sure you are signed in to the management account for the organization. You can't set up the policy using the delegated administrator account or a member account.

2. In the navigation pane, choose **Quick Setup**.
3. On the **Patch Manager** card, choose **Create**.

   > **Tip**
   >
   > If you already have one or more configurations in your account, first choose the **Library** tab or the **Create** button in the **Configurations** section to view the cards.

4. For **Configuration name**, enter a name to help identify the patch policy.
5. In the **Scanning and installation** section, under **Patch operation**, choose whether the patch policy will **Scan** the specified targets or **Scan and install** patches on the specified targets.
6. Under **Scanning schedule**, choose **Use recommended defaults** or **Custom scan schedule**. The default scan schedule will scan your targets daily at 1:00 AM UTC.

   - If you choose **Custom scan schedule**, select the **Scanning frequency**.
   - If you choose **Daily**, enter the time, in UTC, that you want to scan your targets.
   - If you choose **Custom CRON Expression**, enter the schedule as a **CRON expression**. For more information about formatting CRON expressions for Systems Manager, see Reference: Cron and rate expressions for Systems Manager.

     Also, select **Wait to scan targets until first CRON interval**. By default, Patch Manager immediately scans nodes as they become targets.

7. If you chose **Scan and install**, choose the **Installation schedule** to use when installing patches to the specified targets. If you choose **Use recommended defaults**, Patch Manager will install weekly patches at 2:00 AM UTC on Sunday.

   - If you choose **Custom install schedule**, select the **Installation Frequency**.
   - If you choose **Daily**, enter the time, in UTC, that you want to install updates on your targets.

- If you choose **Custom CRON expression**, enter the schedule as a **CRON expression**. For more information about formatting CRON expressions for Systems Manager, see Reference: Cron and rate expressions for Systems Manager.

  Also, clear **Wait to install updates until first CRON interval** to immediately install updates on nodes as they become targets. By default, Patch Manager waits until the first CRON interval to install updates.

- Choose **Reboot if needed** to reboot the nodes after patch installation. Rebooting after installation is recommended but can cause availability issues.

8. In the **Patch baseline** section, choose the patch baselines to use when scanning and updating your targets.

   By default, Patch Manager uses the predefined patch baselines. For more information, see Predefined baselines.

   If you choose **Custom patch baseline**, change the selected patch baseline for operating systems that you don't want to use a predefined AWS patch baseline.

   > **Note**
   >
   > If you use VPC endpoints to connect to Systems Manager, make sure your VPC endpoint policy for S3 allows access to this S3 bucket. For more information, see Permissions for the patch policy S3 bucket.

   > **Important**
   >
   > If you are using a patch policy configuration in Quick Setup, updates you make to custom patch baselines are synchronized with Quick Setup once an hour.
   > If a custom patch baseline that was referenced in a patch policy is deleted, a banner displays on the Quick Setup **Configuration details** page for your patch policy. The banner informs you that the patch policy references a patch baseline that no longer exists, and that subsequent patching operations will fail. In this case, return to the Quick Setup **Configurations** page, select the Patch Manager configuration , and choose **Actions**, **Edit configuration**. The deleted patch baseline name is highlighted, and you must select a new patch baseline for the affected operating system.

9. (Optional) In the **Patching log storage** section, select **Write output to S3 bucket** to store patching operation logs in an Amazon S3 bucket.

> **Note**
>
> If you are setting up a patch policy for an organization, the management account for your organization must have at least read-only permissions for this bucket. All organization units included in the policy must have write-access to the bucket. For information about granting bucket access to different accounts, see Example 2: Bucket owner granting cross-account bucket permissions in the *Amazon Simple Storage Service User Guide*.

10. Choose **Browse S3** to select the bucket that you want to store patch log output in. The management account must have read access to this bucket. All non-management accounts and targets configured in the **Targets** section must have write access to the provided S3 bucket for logging.
11. In the **Targets** section, choose one of the following to identify the accounts and Regions for this patch policy operation.

> **Note**
>
> If you are working in a single account, options for working with organizations and organizational units (OUs) are not available. You can choose whether to apply this configuration to all AWS Regions in your account or only the Regions you select. If you previously specified a Home Region for you account and haven't onboarded to the new Quick Setup console experience, you can't exclude that Region from the **Targets** configuration.

- **Entire organization** – All accounts and Regions in your organization.
- **Custom** – Only the OUs and Regions that you specify.
  - In the **Target OUs** section, select the OUs where you want to set up the patch policy.
  - In the **Target Regions** section, select the Regions where you want to apply the patch policy.
- **Current account** – Only the Regions you specify in the account you are currently signed into are targeted. Choose one of the following:

- **Current Region** – Only managed nodes in the Region selected in the console are targeted.
- **Choose Regions** – Choose the individual Regions to apply the patch policy to.

12. For **Choose how you want to target instances**, choose one of the following to identify the nodes to patch:

    - **All managed nodes** – All managed nodes in the selected OUs and Regions.
    - **Specify the resource group** – Choose the name of a resource group from the list to target its associated resources.

      > **Note**
      >
      > Currently, selecting resource groups is supported only for single account configurations. To patch resources in multiple accounts, choose a different targeting option.

    - **Specify a node tag** – Only nodes tagged with the key-value pair that you specify are patched in all accounts and Regions you have targeted.
    - **Manual** – Choose managed nodes from all specified accounts and Regions manually from a list.

      > **Note**
      >
      > This option currently supports only Amazon EC2 instances. You can add a maximum of 25 instances manually in a patch policy configuration.

13. In the **Rate control** section, do the following:

    - For **Concurrency**, enter a number or percentage of nodes to run the patch policy on at the same time.
    - For **Error threshold**, enter the number or percentage of nodes that can experience an error before the patch policy fails.

14. (Optional) Select the **Add required IAM policies to existing instance profiles attached to your instances** check box.

    This selection applies the IAM policies created by this Quick Setup configuration to nodes that already have an instance profile attached (EC2 instances) or a service role attached (hybrid-activated nodes). We recommend this selection when your managed nodes already have an

instance profile or service role attached, but it doesn't contain all the permissions required for working with Systems Manager.

Your selection here is applied to managed nodes created later in the accounts and Regions that this patch policy configuration applies to.

> **Important**
>
> If you don't select this check box but want Quick Setup to patch your managed nodes using this patch policy, you must do the following:
> Add permissions to your IAM instance profile or IAM service role to access the S3 bucket created for your patch policy
> Tag your IAM instance profile or IAM service role with a specific key-value pair.
> For information, see Case 1: Use your own instance profile or service role with your managed nodes instead of one provided by Quick Setup.

15. Choose **Create**.

To review patching status after the patch policy is created, you can access the configuration from the Quick Setup page.

Set up DevOps Guru using 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**.
3. On the **DevOps Guru** card, choose **Create**.

   > **Tip**
   >
   > If you already have one or more configurations in your account, first choose the **Library** tab or the **Create** button in the **Configurations** section to view the cards.

4. 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.

5. 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**.
6. 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.
7. In the **Target OUs** and **Target Regions** sections, select the check boxes of the OUs and Regions where you want to use DevOps Guru.
8. Choose the Regions where you want to use DevOps Guru in the current account.
9. Choose **Create**.

Deploy Distributor packages using Quick Setup

Distributor is a tool in 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 EC2Launch v2 agent, Amazon Elastic File System (Amazon EFS) utilities package and Amazon CloudWatch agent can be deployed with Quick Setup. For more information about Distributor, see AWS Systems Manager Distributor.

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**.

3. On the **Distributor** card, choose **Create**.

> **Tip**
>
> If you already have one or more configurations in your account, first choose the **Library** tab or the **Create** button in the **Configurations** section to view the cards.

4. In the **Configuration options** section, choose the package you want to deploy.
5. 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.
6. In the **Target OUs** section, select the check boxes of the OUs and Regions you want to deploy the package to.
7. Choose **Create**.

Stop and start EC2 instances automatically on a schedule using Quick Setup

With Quick Setup, a tool in AWS Systems Manager, you can configure Resource Scheduler to automate the starting and stopping of Amazon Elastic Compute Cloud (Amazon EC2) instances.

This Quick Setup configuration helps you reduce operational costs by starting and stopping instances according to the schedule that you specify. This tool helps you avoid incurring unnecessary costs for running instances when they're not needed.

For example, you currently might leave your instances running constantly, even though they're only used 10 hours a day, 5 days a week. Instead, you can schedule your instances to stop every day after business hours. As a result, there would be 70 percent savings for those instances because the running time is reduced from 168 hours to 50 hours. There is no cost to use Quick Setup. However, costs can be incurred by the resources you set up and the usage limits with no fees for the services used to set up your configuration.

Using Resource Scheduler, you can choose to automatically stop and start instances across multiple AWS Regions and AWS accounts according to a schedule you define. The Quick Setup configuration targets Amazon EC2 instances using the tag key and value that you specify. Only the instances with a tag matching the value that you specify in your configuration are stopped or started by Resource Scheduler. Note that if the Amazon EBS volumes attached to the instance are encrypted, you must

add the required permissions for the AWS KMS key to the IAM role for Resource Scheduler to start the instance.

Maximum instances per configuration

An individual configuration supports scheduling up to 5,000 instances per Region. If your case requires more than 5,000 instances to be scheduled in a given Region, you must create multiple configurations. Tag your instances accordingly so each configuration is managing up to 5,000 instances. When creating multiple Resource Scheduler Quick Setup configurations, you must specify different tag key values. For example, one configuration can use the tag key `Environment` with the value `Production`, while another uses `Environment` and `Development`.

Scheduling behaviors

The following points describe certain behaviors of schedule configurations:

- Resource Scheduler starts the tagged instances only if they are in the `Stopped` state. Similarly, instances are only stopped if they are in the `running` state. Resource Scheduler operates on an event driven model and only starts or stops instances at the times that you specify. For example, you create a schedule that starts instances at 9 AM. Resource Scheduler starts all instances associated with the tag you specify that are in the `Stopped` state at 9 AM. If the instances are manually stopped at a later time, Resource Scheduler will not start them again to maintain the `Running` state. Similarly, if an instance is started manually after it was stopped according to your schedule, Resource Scheduler will not stop the instance again.
- If you create a schedule with a start time that is later in a 24-hour day than the stop time, Resource Scheduler assumes your instances are to run overnight. For example, you create a schedule that starts instances at 9 PM, and stops instances at 7 AM. Resource Scheduler starts all instances associated with the tag you specify that are in the `Stopped` state at 9 PM, and stops them at 7 AM the following day. For overnight schedules, the start time applies to the days you select for your schedule. However, the stop time applies to the following day in your schedule.
- When you create a schedule configuration, the current state of your instances might be changed to match the requirements of the schedule.

  For example, say that today is a Wednesday, and you specify a schedule for your managed instances to start at 9 AM and stop at 5 PM on Tuesdays and Thursdays *only*. Because your current time is outside of the prescribed hours for the instances to be running, they will be stopped after the configuration is created. The instances won't run again until the next prescribed hour, 9 AM on Thursday.

If your instances are currently in a `Stopped` state, and you specify a schedule in which they would be running at the current time, Resource Scheduler starts them after the configuration is created.

If you delete your configuration, instances are no longer stopped and started according to the previously defined schedule. In rare cases, instances might not successfully stop or start due to API operation failures.

To set up scheduling for Amazon EC2 instances, perform the following tasks in the AWS Systems Manager Quick Setup console.

To set up instance scheduling 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**.
3. On the **Resource Scheduler** card, choose **Create**.

   > **Tip**
   >
   > If you already have one or more configurations in your account, first choose the **Library** tab or the **Create** button in the **Configurations** section to view the cards.

4. In the **Instance tag** section, specify the tag key and value applied to the instances you want to associate with your schedule.
5. In the **Schedule options** section, specify the time zone, days, and times you want to start and stop your instances.
6. In the **Targets** section, choose whether to set scheduling for a **Custom** group of organizational units (OUs), or the **Current account** you're signed in to:

   - **Custom** – In the **Target OUs** section, select the OUs where you want to set up scheduling. Next, in the **Target Regions** section, select the Regions where you want to set up scheduling.
   - **Current account** – Select **Current Region** or **Choose Regions**. If you selected **Choose Regions**, choose the **Target Regions** where you want to set up scheduling.

7. Verify the schedule information in the **Summary** section.
8. Choose **Create**.

Configure AWS Resource Explorer using Quick Setup

With Quick Setup, a tool in AWS Systems Manager, you can quickly configure AWS Resource Explorer to search and discover resources in your AWS account or across an entire AWS organization. You can search for your resources using metadata like names, tags, and IDs. AWS Resource Explorer provides fast responses to your search queries by using *indexes*. Resource Explorer creates and maintains indexes using a variety of data sources to gather information about resources in your AWS account.

Quick Setup for Resource Explorer automates the index configuration process. For more information about AWS Resource Explorer, see What is AWS Resource Explorer? in the AWS Resource Explorer User Guide.

During Quick Setup, Resource Explorer does the following:

- Creates an index in every AWS Region in your AWS account.
- Updates the index in the Region you specify to be the aggregator index for the account.
- Creates a default view in the aggregator index Region. This view has no filters so it returns all resources found in the index.

Minimum permissions

To perform the steps in the following procedure, you must have the following permissions:

- **Action**: `resource-explorer-2:*` – **Resource**: no specific resource (*)
- **Action**: `iam:CreateServiceLinkedRole` – **Resource**: no specific resource (*)

To configure Resource Explorer

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Quick Setup**.
3. On the **Resource Explorer** card, choose **Create**.
4. In the **Aggregator Index Region** section, choose which Region you want to contain the **aggregator index**. You should select the Region that is appropriate for the geographic location for your users.

5. (Optional) Select the **Replace existing aggregator indexes in Regions other than the one selected above** check box.
6. In the **Targets** section, choose the target **organization** or specific **Organizational Units (OUs)** containing the resources you want to discover.
7. In the **Regions** section, choose which **Regions** to include in the configuration.
8. Review the configuration summary, and then choose **Create**.

On the **Resource Explorer** page, you can monitor the configuration status.

Troubleshooting Quick Setup results

Use the following information to help you troubleshoot problems with Quick Setup, a tool in AWS Systems Manager. This topic includes specific tasks to resolve issues based on the type of Quick Setup issue.

Issue: 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.

Issue: 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.

Issue: 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.

AWS Systems Manager application tools

Application tools are a suite of capabilities that help you manage your applications running in AWS.

Topics

- AWS AppConfig
- AWS Systems Manager Application Manager
- AWS Systems Manager Parameter Store

AWS AppConfig

Information about AWS AppConfig has been moved into separate guides. For more information, see the following:

- AWS AppConfig User Guide
- AWS AppConfig API Reference

AWS Systems Manager Application Manager

Application Manager, a tool in 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 tools 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.

The first time you open Application Manager, the **What Application Manager can do for you** page displays. When you choose **Get started**, Application Manager automatically imports metadata about your resources that were created in other AWS services or Systems Manager tools. Application Manager then displays those resources in a list grouped by predefined categories.

For **Applications**, the list includes the following:

- AWS CloudFormation stacks
- Custom applications
- AWS Launch Wizard applications
- AppRegistry applications
- AWS SAP Enterprise Workload applications
- Amazon ECS clusters
- Amazon EKS clusters

After you set up and configure AWS services and Systems Manager tools, Application Manager displays the following types of information about your resources:

- Information about the current state, status, and Amazon EC2 Auto Scaling health of the Amazon Elastic Compute Cloud (Amazon EC2) instances in your application
- 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 information about your instances in the context of an application**

  Application Manager integrates with Amazon Elastic Compute Cloud (Amazon EC2) to display information about your instances in the context of an application. Application Manager displays instance state, status, and Amazon EC2 Auto Scaling health for a selected application in a graphical format. The **Instances** tab also includes a table with the following information for each instance in your application.

  - Instance state (Pending, Stopping, Running, Stopped)

  - Ping status for SSM Agent
  - Status and name of the last Systems Manager Automation runbook processed on the instance
  - A count of Amazon CloudWatch Logs alarms per state.
    - `ALARM` – The metric or expression is outside of the defined threshold.
    - `OK` – The metric or expression is within the defined threshold.
    - `INSUFFICIENT_DATA` – The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.
  - Auto Scaling group health for the parent and individual autoscaling groups
- **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 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 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 Cost Explorer, a feature of AWS Billing and Cost Management, through the **Cost** widget. After you enable Cost Explorer in the Billing and Cost Management console, the **Cost** 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 execute a runbook against all of the applicable resources in an application without having to leave Application Manager.

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 tools in the Systems Manager service quotas in the *Amazon Web Services General Reference*. Unless otherwise noted, each quota is Region specific.

Topics

- Setting up related services
- Configuring permissions for Systems Manager Application Manager
- Adding applications and container clusters to Application Manager
- Working with applications

Setting up related services

Application Manager, a tool in AWS Systems Manager, displays resources and information from other AWS services and Systems Manager tools. To maximize the amount of operations information displayed in Application Manager, we recommend that you set up and configure these other services or tools *before* you use Application Manager.

Topics

- Set up tasks for importing resources
- Set up tasks for viewing operations information about resources

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.

(Recommended) Verify runbook permissions

You can remediate issues with AWS resources from Application Manager by using Systems Manager Automation runbooks. To use this remediation tool, you must configure or verify permissions. For pricing information, see AWS Systems Manager Pricing.

(Optional) Enable Cost Explorer

AWS Cost Explorer is a feature of AWS Cost Management that you can use to visualize your cost data for further analysis. When you enable Cost Explorer, you can view cost information, cost history, and cost optimization for your application's resources in the Application Manager console.

(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 multicloud 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.

Configuring permissions for Systems Manager Application Manager

You can use all features of Application Manager, a tool in AWS Systems Manager, if your AWS Identity and Access Management (IAM) entity (such as a 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
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
```

Resource details only

```
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

```
applicationinsights:CreateApplication
applicationinsights:DescribeApplication
applicationinsights:ListProblems
ce:GetCostAndUsage
ce:GetTags
ce:ListCostAllocationTags
ce:UpdateCostAllocationTagsStatus
cloudformation:CreateStack
cloudformation:DeleteStack
cloudformation:DescribeStackDriftDetectionStatus
cloudformation:DescribeStackEvents
cloudformation:DescribeStacks
cloudformation:DetectStackDrift
cloudformation:GetTemplate
cloudformation:GetTemplateSummary
cloudformation:ListStacks
cloudformation:UpdateStack
cloudwatch:DescribeAlarms
cloudwatch:DescribeInsightRules
cloudwatch:DisableAlarmActions
cloudwatch:EnableAlarmActions
cloudwatch:GetMetricData
cloudwatch:ListTagsForResource
cloudwatch:PutMetricAlarm
config:DescribeComplianceByConfigRule
config:DescribeComplianceByResource
config:DescribeConfigRules
config:DescribeRemediationConfigurations
config:GetComplianceDetailsByConfigRule
config:GetComplianceDetailsByResource
config:GetResourceConfigHistory
config:ListDiscoveredResources
config:PutRemediationConfigurations
config:SelectResourceConfig
config:StartConfigRulesEvaluation
config:StartRemediationExecution
ec2:DescribeInstances
ecs:DescribeCapacityProviders
ecs:DescribeClusters
ecs:DescribeContainerInstances
```

Application actions and details

```
ecs:ListClusters
ecs:ListContainerInstances
ecs:TagResource
eks:DescribeCluster
eks:DescribeFargateProfile
eks:DescribeNodegroup
eks:ListClusters
eks:ListFargateProfiles
eks:ListNodegroups
eks:TagResource
iam:CreateServiceLinkedRole
iam:ListRoles
logs:DescribeLogGroups
resource-groups:CreateGroup
resource-groups:DeleteGroup
resource-groups:GetGroup
resource-groups:GetGroupQuery
resource-groups:GetTags
resource-groups:ListGroupResources
resource-groups:ListGroups
resource-groups:Tag
resource-groups:Untag
resource-groups:UpdateGroup
s3:ListAllMyBuckets
s3:ListBucket
s3:ListBucketVersions
servicecatalog:GetApplication
servicecatalog:ListApplications
sns:CreateTopic
sns:ListSubscriptionsByTopic
sns:ListTopics
sns:Subscribe
ssm:AddTagsToResource
ssm:CreateDocument
ssm:CreateOpsMetadata
ssm:DeleteDocument
ssm:DeleteOpsMetadata
ssm:DescribeAssociation
ssm:DescribeAutomationExecutions
ssm:DescribeDocument
ssm:DescribeDocumentPermission
ssm:GetDocument
```

Application actions and details

```
ssm:GetInventory
ssm:GetOpsMetadata
ssm:GetOpsSummary
ssm:GetServiceSetting
ssm:ListAssociations
ssm:ListComplianceItems
ssm:ListDocuments
ssm:ListDocumentVersions
ssm:ListOpsMetadata
ssm:ListResourceComplianceSummaries
ssm:ListTagsForResource
ssm:ModifyDocumentPermission
ssm:RemoveTagsFromResource
ssm:StartAssociationsOnce
ssm:StartAutomationExecution
ssm:UpdateDocument
ssm:UpdateDocumentDefaultVersion
ssm:UpdateOpsItem
ssm:UpdateOpsMetadata
ssm:UpdateServiceSetting
tag:GetTagKeys
tag:GetTagValues
tag:TagResources
tag:UntagResources
```

Example policy for all Application Manager permissions

To configure Application Manager permissions for an IAM entity (such as a user, group, or role), create an IAM policy using the following example. This policy example includes all API operations used by Application Manager.

JSON

```
                {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "acm:DescribeCertificate",
                "acm:ListTagsForCertificate",
                "applicationinsights:CreateApplication",
                "applicationinsights:DescribeApplication",
                "applicationinsights:ListProblems",
                "autoscaling:DescribeAutoScalingGroups",
                "ce:GetCostAndUsage",
                "ce:GetTags",
                "ce:ListCostAllocationTags",
                "ce:UpdateCostAllocationTagsStatus",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackDriftDetectionStatus",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStacks",
                "cloudformation:DetectStackDrift",
                "cloudformation:GetTemplate",
                "cloudformation:GetTemplateSummary",
                "cloudformation:ListStacks",
                "cloudformation:ListStackResources",
                "cloudformation:UpdateStack",
                "cloudfront:GetDistribution",
                "cloudfront:ListTagsForResource",
                "cloudtrail:DescribeTrails",
                "cloudtrail:ListTags",
                "cloudtrail:LookupEvents",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeInsightRules",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "cloudwatch:GetMetricData",
                "cloudwatch:ListTagsForResource",
                "cloudwatch:PutMetricAlarm",
                "codebuild:BatchGetProjects",
                "codepipeline:GetPipeline",
                "codepipeline:ListTagsForResource",
                "config:DescribeComplianceByConfigRule",
                "config:DescribeComplianceByResource",
                "config:DescribeConfigRules",
                "config:DescribeRemediationConfigurations",
                "config:GetComplianceDetailsByConfigRule",
                "config:GetComplianceDetailsByResource",
                "config:GetResourceConfigHistory",
```

```
                "config:ListDiscoveredResources",
                "config:PutRemediationConfigurations",
                "config:SelectResourceConfig",
                "config:StartConfigRulesEvaluation",
                "config:StartRemediationExecution",
                "dynamodb:DescribeTable",
                "dynamodb:ListTagsOfResource",
                "ec2:DescribeAddresses",
                "ec2:DescribeCustomerGateways",
                "ec2:DescribeHosts",
                "ec2:DescribeInstances",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeNetworkAcls",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVolumes",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpnConnections",
                "ec2:DescribeVpnGateways",
                "ecs:DescribeCapacityProviders",
                "ecs:DescribeClusters",
                "ecs:DescribeContainerInstances",
                "ecs:ListClusters",
                "ecs:ListContainerInstances",
                "ecs:TagResource",
                "eks:DescribeCluster",
                "eks:DescribeFargateProfile",
                "eks:DescribeNodegroup",
                "eks:ListClusters",
                "eks:ListFargateProfiles",
                "eks:ListNodegroups",
                "eks:TagResource",
                "elasticbeanstalk:DescribeApplications",
                "elasticbeanstalk:ListTagsForResource",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTags",
                "iam:CreateServiceLinkedRole",
                "iam:GetGroup",
                "iam:GetPolicy",
                "iam:GetRole",
```

```
                "iam:GetUser",
                "iam:ListRoles",
                "lambda:GetFunction",
                "logs:DescribeLogGroups",
                "rds:DescribeDBClusters",
                "rds:DescribeDBInstances",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSnapshots",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEventSubscriptions",
                "rds:ListTagsForResource",
                "redshift:DescribeClusterParameters",
                "redshift:DescribeClusters",
                "redshift:DescribeClusterSecurityGroups",
                "redshift:DescribeClusterSnapshots",
                "redshift:DescribeClusterSubnetGroups",
                "resource-groups:CreateGroup",
                "resource-groups:DeleteGroup",
                "resource-groups:GetGroup",
                "resource-groups:GetGroupQuery",
                "resource-groups:GetTags",
                "resource-groups:ListGroupResources",
                "resource-groups:ListGroups",
                "resource-groups:Tag",
                "resource-groups:Untag",
                "resource-groups:UpdateGroup",
                "s3:GetBucketTagging",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:ListBucketVersions",
                "servicecatalog:GetApplication",
                "servicecatalog:ListApplications",
                "sns:CreateTopic",
                "sns:ListSubscriptionsByTopic",
                "sns:ListTopics",
                "sns:Subscribe",
                "ssm:AddTagsToResource",
                "ssm:CreateDocument",
                "ssm:CreateOpsMetadata",
                "ssm:DeleteDocument",
                "ssm:DeleteOpsMetadata",
                "ssm:DescribeAssociation",
                "ssm:DescribeAutomationExecutions",
                "ssm:DescribeDocument",
```

```
                "ssm:DescribeDocumentPermission",
                "ssm:GetDocument",
                "ssm:GetInventory",
                "ssm:GetOpsMetadata",
                "ssm:GetOpsSummary",
                "ssm:GetServiceSetting",
                "ssm:ListAssociations",
                "ssm:ListComplianceItems",
                "ssm:ListDocuments",
                "ssm:ListDocumentVersions",
                "ssm:ListOpsMetadata",
                "ssm:ListResourceComplianceSummaries",
                "ssm:ListTagsForResource",
                "ssm:ModifyDocumentPermission",
                "ssm:RemoveTagsFromResource",
                "ssm:StartAssociationsOnce",
                "ssm:StartAutomationExecution",
                "ssm:UpdateDocument",
                "ssm:UpdateDocumentDefaultVersion",
                "ssm:UpdateOpsMetadata",
                "ssm:UpdateOpsItem",
                "ssm:UpdateServiceSetting",
                "tag:GetResources",
                "tag:GetTagKeys",
                "tag:GetTagValues",
                "tag:TagResources",
                "tag:UntagResources"
            ],
            "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. The following are all of the APIs that allow users to make changes to the application or any other related resources.

```
applicationinsights:CreateApplication
ce:UpdateCostAllocationTagsStatus
cloudformation:CreateStack
cloudformation:DeleteStack
cloudformation:UpdateStack
cloudwatch:DisableAlarmActions
cloudwatch:EnableAlarmActions
cloudwatch:PutMetricAlarm
config:PutRemediationConfigurations
config:StartConfigRulesEvaluation
config:StartRemediationExecution
ecs:TagResource
eks:TagResource
iam:CreateServiceLinkedRole
resource-groups:CreateGroup
resource-groups:DeleteGroup
resource-groups:Tag
resource-groups:Untag
resource-groups:UpdateGroup
sns:CreateTopic
sns:Subscribe
ssm:AddTagsToResource
ssm:CreateDocument
ssm:CreateOpsMetadata
ssm:DeleteDocument
ssm:DeleteOpsMetadata
ssm:ModifyDocumentPermission
ssm:RemoveTagsFromResource
ssm:StartAssociationsOnce
ssm:StartAutomationExecution
ssm:UpdateDocument
ssm:UpdateDocumentDefaultVersion
ssm:UpdateOpsMetadata
ssm:UpdateOpsItem
ssm:UpdateServiceSetting
tag:TagResources
tag:UntagResources
```

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 entity (such as a user, group, or role), see Adding and removing IAM identity permissions.

Adding applications and container 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.

The first time you open Application Manager, the **What Application Manager can do for you** page displays. When you choose **Get started**, Application Manager automatically imports metadata about your resources that were created in other AWS services or Systems Manager tools. Application Manager then displays those resources in a list grouped by predefined categories.

For **Applications**, the list includes the following:

- AWS CloudFormation stacks
- Custom applications
- AWS Launch Wizard applications
- AppRegistry applications
- AWS SAP Enterprise Workload applications
- 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 choose **Create application**.
4. Choose an option from the drop-down list and complete the fields in the page that appears.

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

- Application overview in Application Manager
- Managing your application EC2 instances
- Viewing resources associated with your application
- Managing your application compliance
- Using CloudWatch Application Insights to monitor an application
- Viewing OpsItems for an application
- Managing your application logs
- Use Automation runbooks to remediate application issues
- Tag resources in Application Manager
- Working with AWS CloudFormation templates and stacks in Application Manager
- Working with clusters in Application Manager

Application overview in Application Manager

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*.

About Cost Explorer

Application Manager is integrated with AWS Cost Explorer, a feature of AWS Cost Management, through the **Cost** widget and **Cost** tab. After you enable Cost Explorer in the Cost Management console, the **Cost** widget and **Cost** tab in Application Manager shows cost data for a specific non-container application or application component. You can use filters in the widget, or tab, to view cost data according to different time periods, levels of granularity, and cost types in either a bar or line chart.

You can enable this feature by choosing the **Go to AWS Cost Management console** button. By default, the data is filtered to the past three months. For a non-container application, if you choose the **View all** button, Application Manager opens the **Resources** tab. For container applications, the **View all** button opens the AWS Cost Explorer console.

Actions you can perform on this page

You can turn on and access information about the following widgets on the **Overview** tab on this page. When a widget is enabled, choose its **View all** to see relevant application details for that area.

- In the **Insights and Alarms** section, choose the number for a severity to open the **Monitoring**, tab, where you can view more details about alarms of the chosen severity.
- In the **Cost** section, choose **View all** to open the **Resources** tab, where you can view cost data for a specific application or application component.
- In the **Compliance** section, choose **View all** to open the **Compliance** tab, where you can view compliance information from AWS Config and State Manager associations.

> **Note**
>
> To view patch compliance details, choose the **Compliance** tab directly. Then you can view patch compliance details for the managed nodes used by the selected application.

- 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.
- In the **OpsItems** section, choose a severity to open the **OpsItems** tab where you can view all OpsItems of the chosen severity.
- 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

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.

Managing your application EC2 instances

Application Manager integrates with Amazon Elastic Compute Cloud (Amazon EC2) to display information about your instances in the context of an application. Application Manager displays instance state, status, and Amazon EC2 Auto Scaling health for a selected application in a graphical format. The **Instances** tab also includes a table with the following information for each instance in your application:

- Instance state (Pending, Stopping, Running, Stopped)
- Ping status for SSM Agent
- Status and name of the most recent Systems Manager Automation runbook processed on the instance
- A count of Amazon CloudWatch Logs alarms per state.
  - `ALARM` – The metric or expression is outside of the defined threshold.

- OK – The metric or expression is within the defined threshold.
- `INSUFFICIENT_DATA` – The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.
- Auto Scaling group health for the parent and individual autoscaling groups

If you choose an instance in the **All instances** table, Application Manager displays information about that instance on four tabs:

- **Details** – All of the instance details from Amazon EC2, including the Amazon Machine Image (AMI), DNS information, IP address information, and more.
- **Health** – The current status as provided by EC2 system and instance status checks.
- **Execution history** – Execution logs for Systems Manager Automation runbooks and API calls processed by the instance.
- **CloudWatch alarms** – The name, state, and more for any CloudWatch alarms raised by the instance.

Actions you can perform on this page

You can perform the following actions on this page:

- Start, stop, and terminate instances.
- Apply a Chef recipe.
- Attach instances to, or detach instances from, an Auto Scaling group.
- Enable automated updates for SSM Agent.

To open the Instances 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 that 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 **Instances** tab.

To view details for your application instances

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 that 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 **Instances** tab.
6. Select the button next to the instance whose details you want to view.
7. Review the instance details at the bottom of the page.

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 **Application Manager**.
3. In the **Applications** section, choose a category. If you want to open an application that 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 **Instances** tab.
6. In the **Agent actions** dropdown, choose **Configure SSM Agent update**.
7. Choose **All instances** to configure automatic SSM Agent updates for all managed instances. Alternatively, choose **Instance** to configure automation SSM Agent updates for a single instance in your application.
8. Select the **Enable automatic update** toggle.
9. In the **Specify schedule** dropdown, choose the schedule you want to use for SSM Agent updates.
10. Select **Configure**.

Viewing resources associated with your application

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 AWS Cost Management console** button. Use the filters in this section to view cost information about your application.

To open the Resources 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 **Resources** tab.

Managing your application compliance

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.

Using CloudWatch Application Insights to monitor an application

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.

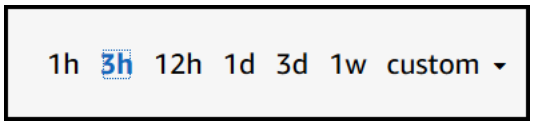
About Application Insights

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*.

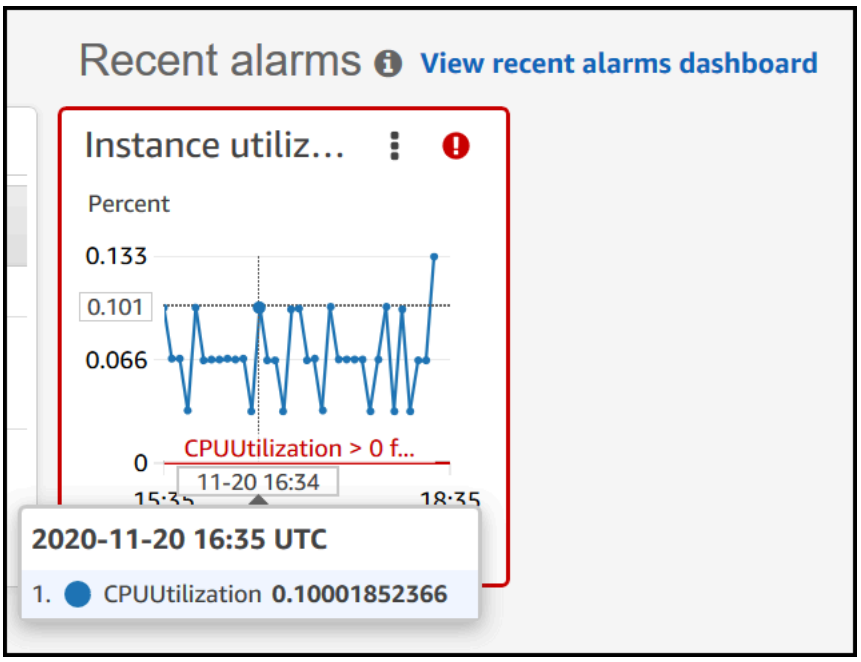
Actions you can perform on this page

You can perform the following actions on this page:

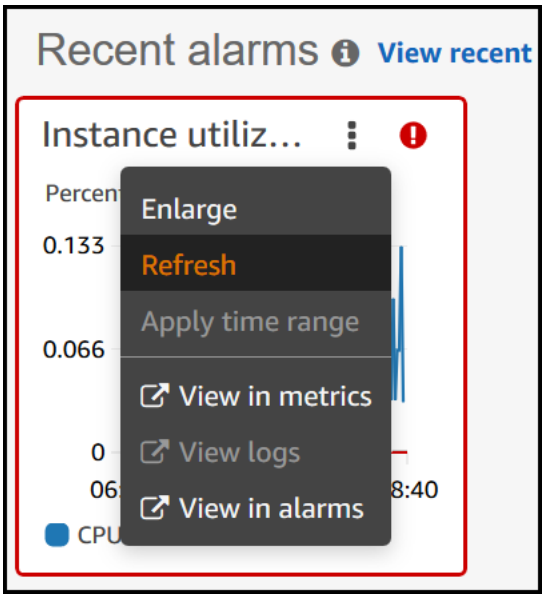
- Choose a service name in the **Alarms by AWS service** section to open CloudWatch to the selected service and alarm.
- Adjust the time period for data displayed in widgets in the **Recent alarms** section by selecting one of the predefined time period values. You can choose **custom** to define your own time period.



- Hover your cursor over a widget in the **Recent alarms** section to view a data pop-up for a specific time.



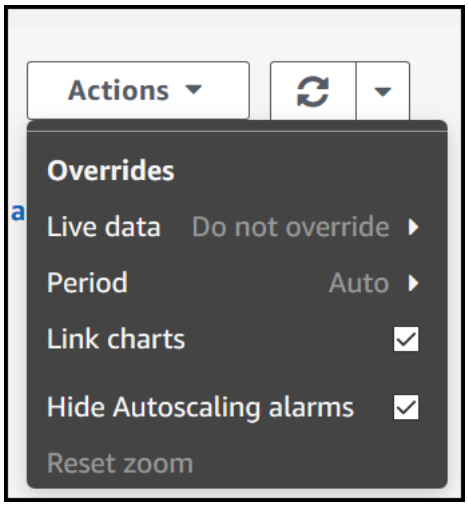
- Choose the options menu in a widget to view display options. Choose **Enlarge** to expand a widget. Choose **Refresh** to update the data in a widget. Click and drag your cursor in a widget data display to select a specific range. You can then choose **Apply time range**.



- 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.

Managing your application logs

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.

Use Automation runbooks to remediate application issues

You can remediate issues with AWS resources from Application Manager, a tool in AWS Systems Manager, by using Automation 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 tool in 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.

When you choose **Start runbook** from an Application Manager application or cluster, the system displays a filtered list of available 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.

- Review the Automation procedure documentation about starting runbooks. For more information, see Run an automated operation powered by Systems Manager Automation.

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 **Automation widget** pop up. For information about the options in the **Automation widget**, see Run an automated operation powered by Systems Manager Automation.

Tag resources in Application Manager

You can quickly add or delete tags on applications and AWS resources in Application Manager. 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 tool in 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.

About Cost Explorer

Application Manager is integrated with AWS Cost Explorer, a feature of AWS Cost Management, through the **Cost** widget. After you enable Cost Explorer in the Cost Management console, the **Cost** 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.

You can enable this feature by choosing the **Go to AWS Cost Management console** button. By default, the data is filtered to the past three months. For a non-container application, if you choose the **View all** button, Application Manager opens the **Resources** tab. For container applications, the **View all** button opens the AWS Cost Explorer console.

Note

Cost Explorer uses tags to track your application costs. If your AWS CloudFormation stack-based application isn't configured with the `AppManagerCFNStackKey` tag key, Cost Explorer fails to present accurate cost data in Application Manager. When the `AppManagerCFNStackKey` tag key is not detected, you will be prompted in the console to add the tag to your CloudFormation stack to enable cost tracking. Adding it maps the tag key to the Amazon Resource Name (ARN) of your stack and enables the **Cost** widget to display accurate cost data.

Important

Adding the `AppManagerCFNStackKey` tag will trigger a stack update. Any manual configurations that were performed after the stack was originally deployed will not be reflected after the user tag is added. For more information about resource update behaviors, see Update behaviors of stack resources in the *AWS CloudFormation User Guide*

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

- Using Application Manager to manage AWS CloudFormation templates
- Using Application Manager to manage AWS CloudFormation stacks

Using Application Manager to manage AWS CloudFormation templates

Application Manager, a tool in 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
- Creating a template
- Editing a template

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
- Sharing SSM documents
- Best practices for shared SSM documents

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. Choose **CloudFormation 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 Infrastructure Composer, a tool for visually creating and modifying templates. For more information, see Create templates visually with Infrastructure Composer in the *AWS CloudFormation User Guide*. For information about the structure and syntax of a template, see AWS CloudFormation template sections in the *AWS CloudFormation User Guide*.

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. Choose **CloudFormation Template Library**, and then either choose **Create template** or choose an existing template and then choose **Actions**, **Clone**.
4. For **Name**, enter a name for the template that helps you identify the resources it creates or the purpose of the stack.

5. (Optional) For **Version name**, enter a name or a number to identity the template version.
6. In the **Code editor** section, choose either **YAML** or **JSON** and then either enter or copy and paste your template code.
7. (Optional) In the **Tags** section, apply one or more tag key name/value pairs to the template.
8. (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.
9. 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. Replace each *example resource placeholder* with your own information.

Before you begin

Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already. For information, see Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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. Choose **CloudFormation Template Library**
4. Choose a template, and then choose **Actions**, **Edit**. You can't change the name of a template, but you can change all other details.
5. Choose **Save**. The template is saved in the Systems Manager Document service.

Using Application Manager to manage AWS CloudFormation stacks

Application Manager, a tool in 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
- Updating a stack

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. Choose **Create Application, CloudFormation Stack**.
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. Select the button beside a template name, and then choose **Next**.
6. 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.
   - Choose **Next**.
7. 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**.

8. On the **Review and provision** page, review all the details of the stack. Choose an **Edit** button on this page to make change.
9. 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. Replace each `example resource placeholder` with your own information. 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 Installing or updating the latest version of the AWS CLI and Installing the AWS Tools for PowerShell.

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-CFNStack `
    -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. Select the radio button next to an application name, and then choose **Actions**, **Update stack**.
4. 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.
5. After you finish making changes to the template, choose **Next**.
6. 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**.
7. 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
- Working with Amazon EKS in Application Manager
- Working with runbooks for clusters

Working with Amazon ECS in Application Manager

With Application Manager, a tool in AWS Systems Manager, you can view and manage your Amazon Elastic Container Service (Amazon ECS) cluster infrastructure. Application Manager applies a tag to your Amazon ECS cluster using the Amazon Resource Name (ARN) of the cluster as the tag value. Application Manager provides 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 containers in Application Manager. You can only manage and view operations information about the infrastructure hosting your Amazon ECS 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 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 **Manage 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 tool in AWS Systems Manager, integrates with Amazon Elastic Kubernetes Service (Amazon EKS) to provide information about the health of your Amazon EKS cluster infrastructure. Application Manager applies a tag to your Amazon EKS cluster using the Amazon Resource Name (ARN) of the cluster as the tag value. Application Manager provides 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 **Manage 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 tool in 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.
- Review the Automation procedure documentation about starting runbooks. For more information, see Run an automated operation powered by Systems Manager Automation.
- 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 Run automated operations at scale.

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 Run an automated operation powered by Systems Manager Automation.

AWS Systems Manager Parameter Store

Parameter Store, a tool in 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.

> **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 triggering actions based on Parameter Store events.
- **Organize parameters**

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 or departments.

- **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.

- **Share parameters with other accounts**

  You can optionally centralize configuration data in a single AWS account and share parameters with other accounts that need to access them.

- **Accessible from other AWS services**

  You can use Parameter Store parameters with other Systems Manager tools and AWS services to retrieve secrets and configuration data from a central store. Parameters work with Systems Manager tools such as Run Command, Automation, and State Manager, tools in 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 rules for parameters and parameter policies.
  - Amazon EventBridge: For more information, see Monitoring Systems Manager status changes using Amazon SNS notifications and Reference: Amazon EventBridge event patterns and types for Systems Manager.
  - AWS CloudTrail: For more information, see Logging AWS Systems Manager API calls with AWS CloudTrail.

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`.

Parameter type: String

By default, the value of a `String` parameter consists of any block of text you enter. For example:

- `abc123`
- `Example Corp`
- `<img src="images/bannerImage1.png"/>`

Parameter type: StringList

The values of `StringList` parameters contain a comma-separated list of values, as shown in the following examples.

`Monday,Wednesday,Friday`

`CSV,TSV,CLF,ELF,JSON`

Parameter type: SecureString

The value of 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 Creating a SecureString parameter using the AWS CLI.

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.)

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.

```
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 keys 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
> ```

More info

- Creating a SecureString parameter in Parameter Store and joining a node to a Domain (PowerShell)
- Use Parameter Store to Securely Access Secrets and Config Data in CodeDeploy
- Interesting Articles on Amazon EC2 Systems Manager Parameter Store

Parameter size limits

Parameter Store has different size limits for parameter values depending on the parameter tier you use:

- **Standard parameters**: Maximum value size of 4 KB
- **Advanced parameters**: Maximum value size of 8 KB

If you need to store parameter values larger than 4 KB, you must use the advanced parameter tier. Advanced parameters provide additional capabilities but incur charges on your AWS account. For more information about parameter tiers and their features, see Managing parameter tiers.

For a complete list of Parameter Store quotas and limits, see AWS Systems Manager endpoints and quotas in the *AWS General Reference*.

Setting up Parameter Store

Before setting up parameters in Parameter Store, a tool in 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 Parameter Store parameters using IAM policies
- Managing parameter tiers
- Increasing or resetting Parameter Store throughput
- Setting up notifications or triggering actions based on Parameter Store events

Restricting access to Parameter Store 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-*`.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeParameters"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
```

```
            "Resource": "arn:aws:ssm:us-east-1:111122223333: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-*`.

JSON

```
{
    "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-1:111122223333:parameter/dbserver-
prod-*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeParameters",
```

```
            "Resource": "*"
        }
    ]
}
```

Denying 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-*`.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ssm:GetParameter*"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333: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-*`.

JSON

```
{
    "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-1:111122223333: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 Managing EC2 instances with Systems Manager.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:parameter/prod-*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:111122223333: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.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:PutParameter",
                "ssm:GetParameter",
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:parameter/prod-*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-12345EXAMPLE"
            ]
        }
    ]
}
```

> **Note**
>
> 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 identified with `aws/ssm` in the **Alias** column.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
```

```
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:111122223333: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 tool in 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 Assigning parameter policies in Parameter Store.
Cost	No additional charge	Charges apply For more information, see AWS Systems Manager Pricing for Parameter Store.

Topics

- Specifying a default parameter tier
- Changing a standard parameter to an advanced parameter

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. To share a parameter, it must be in the advanced parameter tier. There is a charge to use the advanced-parameter tier. For more information, see AWS Systems Manager Pricing for Parameter Store. 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 entity (such as 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

Grant the following permissions to the IAM entity to allow a user to view and change the default tier setting for parameters in a specific AWS Region in an AWS account.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetServiceSetting"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:UpdateServiceSetting"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:servicesetting/ssm/
parameter-store/default-parameter-tier"
        }
    ]
}
```

Administrators can specify read-only permission by assigning the following permissions.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetServiceSetting"
            ],
            "Resource": "*"
        },
```

```
        {
            "Effect": "Deny",
            "Action": [
                "ssm:ResetServiceSetting",
                "ssm:UpdateServiceSetting"
            ],
            "Resource": "*"
        }
    ]
}
```

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.

- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.

- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

Specifying or changing the Parameter Store default tier using the 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 using the AWS CLI and Specifying or changing the Parameter Store default tier (PowerShell).

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**.
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.
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 using the 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.

There is no output if the command succeeds.

2. Run the following command to view the current default parameter tier 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

1. Change the Parameter Store default tier 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/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.

   There is no output if the command succeeds.

2. Run the following command to view the current default parameter tier 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/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 Parameter Store parameters in Systems Manager.

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**.
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 or resetting Parameter Store throughput

Increasing Parameter Store throughput increases the maximum number of transactions per second (TPS) that Parameter Store, a tool in 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.

The Parameter Store throughput setting applies to all transactions created by all IAM users in the current AWS account and AWS Region. The throughput setting applies to standard and advanced parameters.

> **Note**
>
> Typically, updates are immediately visible in Service Quotas. In rare cases, it can take up to 24 hours for an update to be reflected.

For more information about max throughput default and maximum limits, 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.

Topics

- Configuring permissions to change Parameter Store throughput
- Increasing or resetting throughput using the console
- Increasing or resetting throughput using the AWS CLI
- Increasing or resetting throughput (PowerShell)

Configuring permissions to change Parameter Store throughput

Verify that you have permission in IAM to change Parameter Store throughput by doing one of the following:

- Make sure that the `AdministratorAccess` policy is attached to your IAM entity (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

Grant the following permissions to the IAM entity to allow a user to view and change the parameter-throughput setting for parameters in a specific AWS Region in an AWS account.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetServiceSetting"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:UpdateServiceSetting"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:servicesetting/ssm/
parameter-store/high-throughput-enabled"
        }
    ]
}
```

Administrators can specify read-only permission by assigning the following permissions.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetServiceSetting"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "ssm:ResetServiceSetting",
                "ssm:UpdateServiceSetting"
            ],
            "Resource": "*"
        }
    ]
}
```

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.

- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.

- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

Increasing or resetting throughput using the 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. It also shows how to revert to the standard settings if you no longer need increased throughput or no longer want to incur charges.

To increase or reset Parameter Store throughput using the console

> **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 or resetting throughput using the AWS CLI and Increasing or resetting throughput (PowerShell).

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Parameter Store**.
3. Choose the **Settings** tab.
4. To increase throughput, choose **Set limit**.

-or-

To revert to the default limit, choose **Reset limit**.

5. If you are increasing the limit, do the following:

   - Select the check box for **I accept that changing this setting incurs charges on my AWS account**.
   - Choose **Set limit**.

   -or-

   If you are resetting the limit to the default, do the following:

   - Select the check box for **I accept that resetting to the default throughput limit causes Parameter Store to process fewer transactions per second**.
   - Choose **Reset limit**.

Increasing or resetting throughput using the 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. You can also revert to the default limit.

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.

2. 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 or resetting 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. You can also revert to the default limit.

To increase Parameter Store throughput using PowerShell

1. 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 systems 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 triggering 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 in Parameter Store and Managing parameter tiers.

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 tools in 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*.

Configuring EventBridge rules for parameters and parameter policies

This topic explains the following:

- How to create an EventBridge rule that invokes a target based on events that happen to one or more parameters in your AWS account.

- 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 in Parameter Store and Managing parameter tiers.

To configure an EventBridge rule for a Systems Manager parameter or parameter policy

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 **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 **default** . When an AWS service in your account emits an event, it always goes to your account's default event bus.
5. For **Rule type**, leave the default **Rule with an event pattern** selected.
6. Choose **Next**.
7. For **Event source**, leave the default **AWS events or EventBridge partner events** selected. You can skip the **Sample event** section.
8. For **Event pattern**, do the following:

   - Choose **Custom patterns (JSON editor)**.
   - For **Event pattern**, paste one of the following content in the box, depending on whether you are creating a rule for a parameter or a parameter policy:

     Parameter

```
{
    "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"
        ]
    }
}
```

Parameter policy

```
{
    "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"
        ]
    }
}
```

- Modify the contents for the parameters and the operations you want to act on, as shown in the following samples.

  Parameter

  With this example, 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"
        ]
    }
}
```

  Parameter policy

  With this example, 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"
        ]
    }
}
```

9. Choose **Next**.
10. For **Target 1**, choose a target type and a supported resource. For example, if you choose **SNS topic**, make a selection for **Topic**. If you choose **CodePipeline**, enter a pipeline ARN for **Pipeline ARN**. Provide additional configuration values as required.

    > **Tip**
    >
    > Choose **Add another target** if you require additional targets for the rule.

11. Choose **Next**.
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 **Next**.
14. Choose **Create rule**.

More info

- Use parameter labels for easy configuration update across environments
- Tutorial: Use EventBridge to relay events to AWS Systems Manager Run 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 AWS Command Line Interface (AWS CLI), the AWS Tools for PowerShell, and the AWS SDKs to create and work with parameters. For more information about parameters, see What is a parameter?.

Topics

- Creating Parameter Store parameters in Systems Manager
- Searching for Parameter Store parameters in Systems Manager
- Assigning parameter policies in Parameter Store
- Working with parameter hierarchies in Parameter Store
- Preventing access to Parameter Store API operations
- Working with parameter labels in Parameter Store
- Working with parameter versions in Parameter Store
- Working with shared parameters in Parameter Store
- Working with parameters in Parameter Store using Run Command commands
- Using native parameter support in Parameter Store for Amazon Machine Image IDs
- Deleting parameters from Parameter Store

Creating Parameter Store parameters in Systems Manager

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 tools in AWS services. For more information, see What is a parameter?

Understanding 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 specify is 1011 characters. This count of 1011 characters includes the characters in the ARN that precede the name you specify, such as the 45 characters in `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

- Creating a Parameter Store parameter using the console
- Creating a Parameter Store parameter using the AWS CLI
- Creating a Parameter Store parameter using Tools for Windows PowerShell

Creating a Parameter Store parameter using the 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`, and `SecureString` 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**.
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 in Parameter Store.
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.
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 keys 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.

> **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.

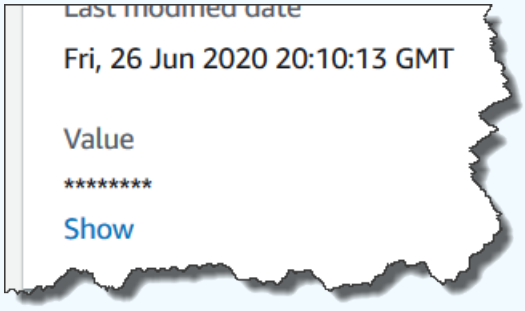
```
arn:aws:kms:us-east-2:123456789012:alias/abcd1234-ab12-cd34-ef56-abcdeEXAMPLE
```

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, as shown in the following illlustration. 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.

Creating a Parameter Store parameter using the 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

- Creating a String parameter using the AWS CLI
- Creating a StringList parameter using the AWS CLI
- Creating a SecureString parameter using the AWS CLI
- Creating a multi-line parameter using the AWS CLI

Creating a `String` parameter using the AWS CLI

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

   For information, see Installing or updating the latest version of the AWS CLI.

2. Run the following command to create a `String`-type parameter. Replace each *example resource placeholder* with your own information.

   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 in Parameter Store.

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 Using native parameter support in Parameter Store for Amazon Machine Image IDs.

> **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 Using native parameter support in Parameter Store for Amazon Machine Image IDs.

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 in Parameter Store.

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"
```

3. 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": []
        }
```

```
    ]
}
```

Creating a StringList parameter using the AWS CLI

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

   For information, see Installing or updating the latest version of the AWS CLI.

2. Run the following command to create a parameter. Replace each *example resource placeholder* with your own information.

   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 /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.

3. Run the `get-parameters` command to verify the details of the parameter. For example:

```
aws ssm get-parameters --name "/IAD/ERP/Oracle/addUsers"
```

Creating a SecureString parameter using the AWS CLI

Use the following procedure to create a `SecureString` parameter. Replace each *example resource placeholder* with your own information.

> **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 Command Line Interface (AWS CLI), if you haven't already.

   For information, see Installing or updating the latest version of the AWS CLI.

2. 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 key 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.

Creating a multi-line parameter using the 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 Installing or updating the latest version of the AWS CLI.
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>
```

Creating a Parameter Store parameter using 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

- Creating a String parameter (Tools for Windows PowerShell)
- Creating a StringList parameter (Tools for Windows PowerShell)
- Creating a SecureString parameter (Tools for Windows PowerShell)

Creating a String parameter (Tools for Windows PowerShell)

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

   For information, see Installing the AWS Tools for PowerShell.

2. Run the following command to create a parameter that contains a plain text value. Replace each *example resource placeholder* with your own information.

```
Write-SSMParameter `
    -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-SSMParameter `
    -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 Using native parameter support in Parameter Store for Amazon Machine Image IDs.

Here is an example that uses a parameter hierarchy.

```
Write-SSMParameter `
    -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-SSMParameterValue -Name "the-parameter-name-you-specified").Parameters
```

Creating a StringList parameter (Tools for Windows PowerShell)

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

   For information, see Installing the AWS Tools for PowerShell.

2. Run the following command to create a StringList parameter. Replace each *example resource placeholder* with your own information.

> **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-SSMParameter `
```

```
    -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-SSMParameter `
    -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-SSMParameterValue -Name "the-parameter-name-you-specified").Parameters
```

Creating 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 Creating a SecureString parameter using the AWS CLI.

> **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 (Tools for Windows PowerShell), if you haven't already.

   For information, see Installing the AWS Tools for PowerShell.

2. Run the following command to create a parameter. Replace each *example resource placeholder* with your own information.

   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-SSMParameter `
       -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 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 key.

```
Write-SSMParameter `
    -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.

Searching for Parameter Store parameters in Systems Manager

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

- Searching for a parameter using the console
- Searching for a parameter using the AWS CLI

Searching for a parameter 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**.
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.

> **Note**
>
> Your search might contain more results than are displayed on the first page of results. Use the right arrow (**>**) at the topic of the parameter list (if available) to view the next set of results.

Searching for a parameter using the 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 Installing or updating the latest version of the AWS CLI.

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 Creating a SecureString parameter using the AWS CLI.

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": []
        }
    ]
}
```

Assigning parameter policies in Parameter Store

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 tool in 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.

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
Expiration	This policy deletes the parameter. You can specify a specific date and time by using either the `ISO_INSTANT` format or the `ISO_OFFSET_DATE_TIME` format. To change when you want the parameter to be deleted, update the policy. Updating a *parameter* 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. **Note** This example uses the `ISO_INSTANT` format. You can also specify a date and time by using the `ISO_OFFSE`	`{ "Type": "Expiration", "Version": "1.0", "Attributes": { "Timestamp": "2018-12-02T21:34:33.000Z" } }`

Policy	Details	Examples
	`T_DATE_TIME` format. Here is an example: `2019-11-01T22:13:48.87+10:30:00` .	
ExpirationNotification	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.	`{ "Type": "ExpirationNotification", "Version": "1.0", "Attributes": { "Before": "15", "Unit": "Days" } }`
NoChangeNotification	This policy initiates an event in EventBridge if a parameter has *not* been modified for a specified period of time. This policy type is useful when, for example, a password needs to be changed within a period of time. This policy determines when to send a notification by reading the `LastModifiedTime` attribute of the parameter. If you change or edit a parameter, the system resets the notification time period based on the new value of `LastModifiedTime` .	`{ "Type": "NoChangeNotification", "Version": "1.0", "Attributes": { "After": "20", "Unit": "Days" } }`

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 Parameter Store parameters in Systems Manager.

Topics

- Adding policies to an existing parameter using the console
- Adding policies to an existing parameter using the AWS CLI
- Adding policies to an existing parameter (Tools for Windows PowerShell)

Adding policies to an existing parameter using the 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**.
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.

Adding policies to an existing parameter using the 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 Installing or updating the latest version of the AWS CLI.

2. Run the following command to add policies to an existing parameter. Replace each *example resource placeholder* with your own information.

   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 (3l3vat3131) 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/3l3vat3131" \
    --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/3l3vat3131" ^
    --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\"}}]"
```

3. Run the following command to verify the details of the parameter. Replace *parameter name* with your own information.

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. Replace each *example resource placeholder* with your own information. 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.

Adding 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. Replace each *example resource placeholder* with your own information.

To add policies to an existing parameter

1. 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).

   ```
   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. Replace each *example resource placeholder* with your own information.

   ```
   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@sSwW)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. Replace *parameter name* with your own information.

```
(Get-SSMParameterValue -Name "parameter name").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.

Working with parameter hierarchies in Parameter Store

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

- Understanding parameter hierarchy through examples
- Querying parameters in a hierarchy
- Managing parameters using hierarchies using the AWS CLI

Understanding parameter hierarchy through 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.

`/DevTest/MyApp/database/my-password`

You could then create a unique password for your production environment, as shown in the following example:

`/prod/MyApp/database/my-password`

You 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 systems 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 *under* a certain level in 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 under the `Oncall` level:

```
aws ssm get-parameters-by-path --path /Dev/Web/Oncall
```

Sample output:

```
{
    "Parameters": [
        {
            "Name": "/Dev/Web/Oncall/Week1",
            "Type": "String",
            "Value": "John Doe",
            "Version": 1,
            "LastModifiedDate": "2024-11-22T07:18:53.510000-08:00",
            "ARN": "arn:aws:ssm:us-east-2:123456789012:parameter/Dev/Web/Oncall/Week1",
            "DataType": "text"
        },
        {
            "Name": "/Dev/Web/Oncall/Week2",
            "Type": "String",
            "Value": "Mary Major",
            "Version": 1,
            "LastModifiedDate": "2024-11-22T07:21:25.325000-08:00",
            "ARN": "arn:aws:ssm:us-east-2:123456789012:parameter/Dev/Web/Oncall/Week2",
            "DataType": "text"
        }
    ]
}
```

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
```

Managing parameters using hierarchies using the 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 Installing or updating the latest version of the AWS CLI.

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 "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 under 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"
   ```

Preventing access to Parameter Store API operations

Using service-specific *conditions* supported by Systems Manager for AWS Identity and Access Management (IAM) policies, you can explicity allow or deny access to Parameter Store API operations and content. By using these conditions, you can allow only certain IAM Entities (users and roles) in your organization to call certain API actions, or prevent certain IAM Entities from running them. This includes actions run through the Parameter Store console, the AWS Command Line Interface (AWS CLI), and SDKs.

Systems Manager currently supports three conditions that are specific to Parameter Store.

Topics

- Preventing changes to existing parameters using ssm:Overwrite
- Preventing creation or updates to parameters that use a parameter policy using ssm:Policies
- Preventing access to levels in a hierarchical parameter using ssm:Recursive

Preventing changes to existing parameters using `ssm:Overwrite`

Use the `ssm:Overwrite` condition to control whether IAM Entities can update existing parameters.

In the following sample policy, the `"Allow"` statement grants permission to create parameters by running the `PutParameter` API operation in the AWS account 123456789012 in the US East (Ohio) Region (us-east-2).

However, the `"Deny"` statement prevents Entities from changing values of *existing* parameters because the `Overwrite` option is explicitly denied for the `PutParameter` operation. Therefore, Entities that are assigned this policy can create parameters, but not make changes to existing parameters.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:PutParameter"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "ssm:PutParameter"
            ],
            "Condition": {
                "StringEquals": {
                    "ssm:Overwrite": [
                        "true"
                    ]
                }
            },
            "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/*"
        }
    ]
}
```

Preventing creation or updates to parameters that use a parameter policy using `ssm:Policies`

User the `ssm:Policies` condition to control whether Entities can create parameters that include a parameter policy and update existing parameters that include a parameter policy.

In the following policy example, the `"Allow"` statement grants general permission to create parameters, but the `"Deny"` statement prevents Entities from creating or updating parameters that include a parameter policy in the the AWS account 123456789012 in the US East (Ohio) Region (us-east-2). Entities can still create or update parameters that aren't assigned a parameter policy.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:PutParameter"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "ssm:PutParameter"
            ],
            "Condition": {
                "StringEquals": {
                    "ssm:Policies": [
                        "true"
                    ]
                }
            },
            "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/*"
        }
    ]
}
```

Preventing access to levels in a hierarchical parameter using `ssm:Recursive`

Use the `ssm:Recursive` condition to control whether IAM Entities can view or reference levels in a hierarchical parameter. You can provide or restrict access to all parameters beyond a specific level of a hierarchy.

In the following example policy, the `"Allow"` statement provides access to Parameter Store operations on all parameters in the path `/Code/Departments/Finance/*` for the AWS account 123456789012 in the US East (Ohio) Region (us-east-2).

After this, the `"Deny"` statement prevents IAM Entities from viewing or retrieving parameter data at or below the level of `/Code/Departments/*`. Entities can still, however, still create or update parameters in that path. The example has been constructed to illustrate that recursively denying access below a certain level in a parameter hierarchy takes precedence over more permissive access in the same policy.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:*"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "ssm:GetParametersByPath"
            ],
            "Condition": {
                "StringEquals": {
                    "ssm:Recursive": [
                        "true"
                    ]
                }
            },
            "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/Code/
Departments/*"
```

```
        }
    ]
}
```

> **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. This is true unless the user has explicitly been denied access in IAM for parameter /b, as illustrated above.

Working with parameter labels in Parameter Store

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.

> **Note**
>
> If you run a command that specifies a version of a parameter that doesn't exist, the command fails. It doesn't fall back to the latest or default value of the parameter.

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 using the console
- Working with parameter labels using the AWS CLI

Working with parameter labels using the console

This section describes how to perform the following tasks by using the Systems Manager console.

- Creating a parameter label using the console
- Viewing labels attached to a parameter using the console

- Moving a parameter label using the console
- Deleting parameter labels using the console

Creating a parameter label using the 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**.
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**.

Viewing labels attached to a parameter using the 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**.
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.

Moving a parameter label using the 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**.
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**.

Deleting parameter labels using the 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**.
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 using the AWS CLI

This section describes how to perform the following tasks by using the AWS Command Line Interface (AWS CLI).

- Creating a new parameter label using the AWS CLI
- Viewing labels for a parameter using the AWS CLI
- Viewing a list of parameters that are assigned a label using the AWS CLI
- Moving a parameter label using the AWS CLI
- Deleting parameter labels using the AWS CLI

Creating a new parameter label using the 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 Installing or updating the latest version of the AWS CLI.
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.

Viewing labels for a parameter using the 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"
        }
```

```
    ]
}
```

Viewing a list of parameters that are assigned a label using the 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. Replace each *example resource placeholder* with your own information.

```
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"
        }
    ]
```

```
}
```

Moving a parameter label using the 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. Replace *parameter name* with your own information.

   ```
   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. Replace each *example resource placeholder* with your own information.

   ```
   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.

Deleting parameter labels using the 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. Replace *parameter name* with your own information.

   ```
   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.

2. Run the following command to delete the labels you choose from that parameter. Replace each *example resource placeholder* with your own information.

```
aws ssm unlabel-parameter-version \
    --name parameter name \
    --parameter-version version \
    --labels label 1,label 2,label 3
```

The system returns information like the following.

```
{
    "InvalidLabels": ["invalid"],
    "DeletedLabels" : ["Prod"]
 }
```

Working with parameter versions in Parameter Store

Each time you edit the value of a parameter, Parameter Store, a tool in 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 increments 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. If you specify the number for a version that doesn't exist, the system returns an error rather than falling back to the latest or default version of the parameter.

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 Moving a parameter label using the console and Moving a parameter label using the AWS CLI.

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

- Creating a new version of a parameter using the console
- Referencing a parameter version

Creating a new version of a parameter using the 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**.
3. Choose the name of a parameter that you created earlier. For information about creating a new parameter, see Creating Parameter Store parameters in Systems Manager.
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.

Referencing 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 Using native parameter support in Parameter Store for Amazon Machine Image IDs.

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}}"
                ]
            }
        }
    ]
}
```

Working with shared parameters in Parameter Store

Sharing advanced parameters simplifies configuration data management in a multi-account environment. You can centrally store and manage your parameters and share them with other AWS accounts that need to reference them.

Parameter Store integrates with AWS Resource Access Manager (AWS RAM) to enable advanced parameter sharing. AWS RAM is a service that enables you to share resources with other AWS accounts or through AWS Organizations.

With AWS RAM, you share resources that you own by creating a resource share. A resource share specifies the resources to share, permissions to grant, and the consumers with whom to share. Consumers can include:

- Specific AWS accounts inside or outside of its organization in AWS Organizations
- An organizational unit inside its organization in AWS Organizations
- Its entire organization in AWS Organizations

For more information about AWS RAM, see the *AWS RAM User Guide*.

This topic explains how to share parameters that you own, and how to use parameters that are shared with you.

Contents

- Prerequisites for sharing parameters
- Sharing a parameter
- Stop sharing a shared parameter
- Identifying shared parameters
- Accessing shared parameters
- Permissions sets for sharing parameters
- Maximum throughput for shared parameters
- Pricing for shared parameters
- Cross-account access for closed AWS accounts

Prerequisites for sharing parameters

The following prerequisites must be met before you can share parameters from your account:

- To share a parameter, you must own it in your AWS account. You can't share a parameter that has been shared with you.
- To share a parameter, it must be in the advanced parameter tier. For information about parameter tiers, see Managing parameter tiers. For information about changing an existing

standard parameter to an advanced parameter, see Changing a standard parameter to an advanced parameter.

- To share a `SecureString` parameter, it must be encrypted with a customer managed key, and you must share the key separately through AWS Key Management Service. AWS managed keys cannot be shared. Parameters encrypted with the default AWS managed key can be updated to use a customer managed key instead. For AWS KMS key definitions, see AWS KMS concepts in the *AWS Key Management Service Developer Guide*.
- To share a parameter with your organization or an organizational unit in AWS Organizations, you must enable sharing with AWS Organizations. For more information, see Enable Sharing with AWS Organizations in the *AWS RAM User Guide*.

Sharing a parameter

To share a parameter, you must add it to a resource share. A resource share is an AWS RAM resource that lets you share your resources across AWS accounts. A resource share specifies the resources to share, and the consumers with whom they are shared.

When you share a parameter that you own with other AWS accounts, you can choose from two AWS managed permissions to grant the consumers. For more information, see Permissions sets for sharing parameters.

If you are part of an organization in AWS Organizations and sharing within your organization is enabled, you can grant consumers in your organization access from the AWS RAM console to the shared parameter. Otherwise, consumers receive an invitation to join the resource share and are granted access to the shared parameter after accepting the invitation.

You can share a parameter that you own using the AWS RAM console, or the AWS CLI.

> **Note**
>
> While you can share a parameter using the Systems Manager PutResourcePolicy API operation, we recommend using AWS Resource Access Manager (AWS RAM) instead. This is because using `PutResourcePolicy` requires the extra step of promoting the parameter to a standard Resource Share using the AWS RAM PromoteResourceShareCreatedFromPolicy API operation. Otherwise, the parameter won't be returned by the Systems Manager DescribeParameters API operation using the `--shared` option.

To share a parameter that you own using the AWS RAM console

See Creating a resource share in AWS RAM in the *AWS RAM User Guide*.

Make the following selections as you complete the procedure:

- In the Step 1 page, for **Resources**, select `Parameter Store Advanced Parameter`, and then select the box of each parameter in the advanced parameter tier that you want to share.
- In the Step 2 page, for **Managed permissions**, choose the permission to grant consumers, as described in Permissions sets for sharing parameters later in this topic.

Choose other options based on your parameter sharing objectives.

To share a parameter that you own using the AWS CLI

Use the create-resource-share command to add parameters to a new resource share.

Use the associate-resource-share command to add parameters to an existing resource share.

The following example creates a new resource share to share parameters with consumers in an organization and in an individual account.

```
aws ram create-resource-share \
    --name "MyParameter" \
    --resource-arns "arn:aws:ssm:us-east-2:123456789012:parameter/MyParameter" \
    --principals "arn:aws:organizations::123456789012:ou/o-63bEXAMPLE/ou-46xi-rEXAMPLE"
 "987654321098"
```

Stop sharing a shared parameter

When you stop sharing a shared parameter, the consumer account can no longer access the parameter.

To stop sharing a parameter that you own, you must remove it from the resource share. You can do this using the Systems Manager console, AWS RAM console, or the AWS CLI.

To stop sharing a parameter that you own using the AWS RAM console

See Update a resource share in AWS RAM in the *AWS RAM User Guide*.

To stop sharing a parameter that you own using the AWS CLI

Use the disassociate-resource-share command.

Identifying shared parameters

Owners and consumers can identify shared parameters using the AWS CLI.

To identify shared parameters using the AWS CLI

To identify shared parameters using the AWS CLI, you can choose from the Systems Manager `describe-parameters` command and the AWS RAM `list-resources` command.

When you use the `--shared` option with `describe-parameters`, the command returns the parameters that are shared with you.

The following is an example:

```
aws ssm describe-parameters --shared
```

Accessing shared parameters

Consumers can access shared parameters using the AWS command line tools, and AWS SDKs. For consumer accounts, parameters shared with that account aren't included in the **My parameters** page.

CLI Example: Accessing shared parameter details using the AWS CLI

To access shared parameter details using the AWS CLI, you can use the get-parameter or get-parameters commands. You must specify the full parameter ARN as the `--name` in order to retrieve the parameter from another account.

The following is an example.

```
aws ssm get-parameter \
    --name arn:aws:ssm:us-east-2:123456789012:parameter/MySharedParameter
```

Supported and unsupported integrations for shared parameters

Currently, you can use shared parameters in the following integration scenarios:

- AWS CloudFormation template parameters
- The AWS Parameters and Secrets Lambda extension
- Amazon Elastic Compute Cloud (EC2) launch templates
- Values for `ImageID` with the EC2 RunInstances command to create instances from an Amazon Machine Image (AMI)

- Retrieving parameter values in runbooks for Automation, a tool in Systems Manager

The following scenarios and integrated services do not currently support the use of shared parameters:

- Parameters in commands in Run Command, a tool in Systems Manager
- AWS CloudFormation dynamic references
- The values of environment variables in AWS CodeBuild
- The values of environment variables in AWS App Runner
- The value of a secret in Amazon Elastic Container Service

Permissions sets for sharing parameters

Consumer accounts receive read-only access to the parameters you share with them. The consumer can't update or delete the parameter. The consumer can't share the parameter with a third account.

When you create a resource share in AWS Resource Access Manager for sharing your parameters, you can choose from two AWS managed permission sets to grant this read-only access:

AWSRAMDefaultPermissionSSMParameterReadOnly

Allowed actions: `DescribeParameters`, `GetParameter`, `GetParameters`

AWSRAMPermissionSSMParameterReadOnlyWithHistory

Allowed actions: `DescribeParameters`, `GetParameter`, `GetParameters`, `GetParameterHistory`

When you folllow the steps in Creating a resource share in AWS RAM in the *AWS RAM User Guide*, choose `Parameter Store Advanced Parameters` as the resource type and either of these managed permissions, depending on whether you want users to view parameter history or not.

> **Note**
>
> If you're retrieving shared parameters programmatically (for example, using AWS Lambda) you might need to add the `ssm:GetResourcePolicies` and `ssm:PutResourcePolicy` permissions to any IAM roles calling AWS Resource Access Manager API actions.

Maximum throughput for shared parameters

Systems Manager limits the maximum throughput (transactions per second) for the GetParameter and GetParameters. operations. Throughput is enforced at the individual account level. Therefore, each account that consumes a shared parameter can use its maximum allowed throughput without being affected by other accounts. For more information about maximum throughput for parameters, see the following topics:

- Increasing Parameter Store throughput
- Systems Manager Service quotas in the *Amazon Web Services General Reference*.

Pricing for shared parameters

Cross-account sharing is only available in the advanced parameter tier. For advanced parameters, charges are incurred at the current price for the *storage* and *API usage* for each advanced parameter. The owning account is charged for storage of the advanced parameter. Any consuming account that makes an API call to a shared advanced parameter is charged for the parameter usage.

For example, if Account A creates an advanced parameter, `MyAdvancedParameter`, that account is charged USD 0.05 per month to store the parameter.

Account A then shares `MyAdvancedParameter` with Account B and Account C. During a month, the three accounts make calls to `MyAdvancedParameter`. The following table illustrates the charges they would incur for the number of calls each makes.

> **Note**
>
> The charges in the following table are for illustration only. To verify current pricing, see AWS Systems Manager Pricing for Parameter Store.

Account	Number of calls	Charges
Account A (owning account)	10,000 calls	• One month advanced parameter storage: USD 0.05

Account	Number of calls	Charges
		• 10,000 calls to `MyAdvanceParameter` : USD 0.05 • **Total: USD 0.10**
Account B (consuming account)	20,000 calls	• 20,000 calls to `MyAdvanceParameter` : USD 0.10 • **Total: USD 0.10**
Account C (consuming account)	30,000 calls	• 30,000 calls to `MyAdvanceParameter` : USD 0.15 • **Total: USD 0.15**

Cross-account access for closed AWS accounts

If the AWS account that owns a shared parameter is closed, all consuming accounts lose access to the shared parameter. If the owning account is reopened within 90 days after the account is closed, consuming accounts regain access to the previously shared parameters. For more information about reopening an account during the Post-Closure Period, see Accessing your AWS account after you close it in the *AWS Account Management Reference Guide*.

Working with parameters in Parameter Store using Run Command commands

You can work with parameters in Run Command, a tool in AWS Systems Manager. For more information, see AWS Systems Manager Run Command.

Running a String parameter using the 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**.
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.

   > **Tip**
   >
   > If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.
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`**.

Running a parameter using the AWS CLI

Example 1: Simple command

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-02573cafcfEXAMPLE" \
    --parameters "commands='echo {{ssm:DNS-IP}}'" \
    --timeout-seconds 600 \
```

```
    --max-concurrency "50" \
    --max-errors "0" \
    --region us-east-2
```

Windows

```
aws ssm send-command ^
    --document-name "AWS-RunPowerShellScript" ^
    --document-version "1" ^
    --targets "Key=instanceids,Values=i-02573cafcfEXAMPLE" ^
    --parameters "commands='echo {{ssm:DNS-IP}}'" ^
    --timeout-seconds 600 ^
    --max-concurrency "50" ^
    --max-errors "0" ^
    --region us-east-2
```

The command returns information like the following.

```
{
    "Command": {
        "CommandId": "c70a4671-8098-42da-b885-89716EXAMPLE",
        "DocumentName": "AWS-RunShellScript",
        "DocumentVersion": "1",
        "Comment": "",
        "ExpiresAfter": "2023-12-26T15:19:17.771000-05:00",
        "Parameters": {
            "commands": [
                "echo {{ssm:DNS-IP}}"
            ]
        },
        "InstanceIds": [],
        "Targets": [
            {
                "Key": "instanceids",
                "Values": [
                    "i-02573cafcfEXAMPLE"
                ]
            }
        ],
        "RequestedDateTime": "2023-12-26T14:09:17.771000-05:00",
        "Status": "Pending",
        "StatusDetails": "Pending",
```

```
        "OutputS3Region": "us-east-2",
        "OutputS3BucketName": "",
        "OutputS3KeyPrefix": "",
        "MaxConcurrency": "50",
        "MaxErrors": "0",
        "TargetCount": 0,
        "CompletedCount": 0,
        "ErrorCount": 0,
        "DeliveryTimedOutCount": 0,
        "ServiceRole": "",
        "NotificationConfig": {
            "NotificationArn": "",
            "NotificationEvents": [],
            "NotificationType": ""
        },
        "CloudWatchOutputConfig": {
            "CloudWatchLogGroupName": "",
            "CloudWatchOutputEnabled": false
        },
        "TimeoutSeconds": 600,
        "AlarmConfiguration": {
            "IgnorePollAlarmFailure": false,
            "Alarms": []
        },
        "TriggeredAlarms": []
    }
}
```

After a command execution completes, you can view more information about it using the following commands:

- get-command-invocation – View detailed information about the command execution.
- list-command-invocations – View the command execution status on a specific managed node.
- list-commands – View the command execution status across managed nodes.

Example 2: Decrypt a `SecureString` parameter value

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-02573cafcfEXAMPLE" \
        --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-02573cafcfEXAMPLE" ^
        --parameters "commands=['$secure = (Get-SSMParameterValue -Names
 SecurePassword -WithDecryption $True).Parameters[0].Value','net user administrator
 $secure']" ^
        --timeout-seconds 600 ^
        --max-concurrency "50" ^
        --max-errors "0" ^
        --region us-east-2
```

Example 3: Reference a parameter in an SSM document

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
```

Using native parameter support in Parameter Store 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*.

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.

Granting 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.

To create an `aws:ec2:image` data type parameter, the user must have both `ssm:PutParameter` and `ec2:DescribeImages` permissions.

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.

JSON

```
{
    "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 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 triggering actions based on Parameter Store events.

Deleting parameters from Parameter Store

This topic describes how to delete parameters that you have created in Parameter Store, a tool in AWS Systems Manager.

> **Warning**
>
> Deleting a parameter removes all versions of it. Once deleted, the parameter and its versions can't be restored.

To delete a parameter 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**.
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**.

To delete a parameter using the AWS CLI

- Run the following command:

```
aws ssm delete-parameter --name "my-parameter"
```

Replace *my-parameter* with the name of your parameter to be deleted.

For information about all options available for use with the `delete-parameter` command, see delete-parameter in the *AWS Systems Manager section of the AWS CLI Command Reference*.

Working with public parameters in Parameter Store

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.

Topics in this guide

- Discovering public parameters in Parameter Store
- Calling AMI public parameters in Parameter Store
- Calling the ECS optimized AMI public parameter in Parameter Store
- Calling the EKS optimized AMI public parameter in Parameter Store
- Calling public parameters for AWS services, Regions, endpoints, Availability Zones, local zones, and Wavelength Zones in Parameter Store

Related AWS blog posts

- Query for AWS Regions, Endpoints, and More Using AWS Systems Manager Parameter Store
- Query for the latest Amazon Linux AMI IDs using AWS Systems Manager Parameter Store
- Query for the Latest Windows AMI Using AWS Systems Manager Parameter Store

Discovering public parameters in Parameter Store

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 partial list of AWS services and other resources that provide public parameters:

- ami-amazon-linux-latest
- ami-windows-latest
- ec2-macos
- appmesh
- aws-for-fluent-bit
- aws-sdk-pandas
- bottlerocket
- canonical
- cloud9
- datasync
- deeplearning
- ecs
- eks
- fis
- freebsd
- global-infrastructure
- marketplace
- neuron
- powertools
- sagemaker-distribution
- storagegateway

Not all public parameters are published to every AWS Region.

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**.
3. Choose the **Public parameters** tab.
4. Choose the **Select a service** dropdown. Choose the service whose parameters you want to use.
5. (Optional) 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.

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 command returns information like the following. This example 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).

> **Note**
>
> The `/aws/service/global-infrastructure` path is not supported for queries in all Regions. For information, see Calling public parameters for AWS services, Regions, endpoints, Availability Zones, local zones, and Wavelength Zones in Parameter Store.

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`. The following is an example.

```
{
    "Parameters": [
        {
            "Name": "/aws/service/global-infrastructure/current-region",
            "Type": "String",
            "LastModifiedDate": "2019-06-21T05:15:34.252000-07:00",
            "Version": 1,
            "Tier": "Standard",
            "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.

Calling AMI public parameters in Parameter Store

Amazon Elastic Compute Cloud (Amazon EC2) Amazon Machine Image (AMI) public parameters are available for Amazon Linux 2, Amazon Linux 2023 (AL2023), macOS, and Windows Server from the following paths:

- Amazon Linux 2, and Amazon Linux 2023: `/aws/service/ami-amazon-linux-latest`

- macOS: `/aws/service/ec2-macos`
- Windows Server: `/aws/service/ami-windows-latest`

Calling AMI public parameters for Amazon Linux 2 and Amazon Linux 2023

You can view a list of all Amazon Linux 2 and Amazon Linux 2023 (AL2023) 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/al2023-ami-kernel-6.1-arm64",
    "/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64",
    "/aws/service/ami-amazon-linux-latest/al2023-ami-minimal-kernel-6.1-arm64",
    "/aws/service/ami-amazon-linux-latest/al2023-ami-minimal-kernel-6.1-x86_64",
    "/aws/service/ami-amazon-linux-latest/al2023-ami-minimal-kernel-default-arm64",
    "/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/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-kernel-5.10-hvm-x86_64-ebs",
    "/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-arm64",
    "/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64",
    "/aws/service/ami-amazon-linux-latest/al2023-ami-minimal-kernel-default-x86_64",
    "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-ebs",
    "/aws/service/ami-amazon-linux-latest/amzn-ami-minimal-hvm-x86_64-ebs",
    "/aws/service/ami-amazon-linux-latest/amzn-ami-minimal-hvm-x86_64-s3",
```

```
    "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2",
    "/aws/service/ami-amazon-linux-latest/amzn2-ami-kernel-5.10-hvm-arm64-gp2",
    "/aws/service/ami-amazon-linux-latest/amzn2-ami-kernel-5.10-hvm-x86_64-gp2",
    "/aws/service/ami-amazon-linux-latest/amzn2-ami-minimal-hvm-arm64-ebs",
    "/aws/service/ami-amazon-linux-latest/amzn2-ami-minimal-hvm-x86_64-ebs"
]
```

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/al2023-ami-kernel-6.1-arm64",
            "Type": "String",
            "Value": "ami-0b1b8b24a6c8e5d8b",
            "Version": 69,
            "LastModifiedDate": "2024-03-13T14:05:09.583000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-
latest/al2023-ami-kernel-6.1-arm64",
            "DataType": "text"
        },
```

```
        {
            "Name": "/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-
x86_64",
            "Type": "String",
            "Value": "ami-0e0bf53f6def86294",
            "Version": 69,
            "LastModifiedDate": "2024-03-13T14:05:09.890000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-
latest/al2023-ami-kernel-6.1-x86_64",
            "DataType": "text"
        },
        {
            "Name": "/aws/service/ami-amazon-linux-latest/al2023-ami-minimal-
kernel-6.1-arm64",
            "Type": "String",
            "Value": "ami-09951bb66f9e5b5a5",
            "Version": 69,
            "LastModifiedDate": "2024-03-13T14:05:10.197000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-
latest/al2023-ami-minimal-kernel-6.1-arm64",
            "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/al2023-ami-kernel-6.1-arm64 \
    --region us-east-2
```

Windows

```
aws ssm get-parameters ^
    --names /aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-arm64 ^
    --region us-east-2
```

The command returns the following information.

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-arm64",
            "Type": "String",
            "Value": "ami-0b1b8b24a6c8e5d8b",
            "Version": 69,
            "LastModifiedDate": "2024-03-13T14:05:09.583000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-
latest/al2023-ami-kernel-6.1-arm64",
            "DataType": "text"
        }
    ],
    "InvalidParameters": []
}
```


Calling AMI public parameters for macOS

You can view a list of all macOS 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/ec2-macos\
    --query 'Parameters[].Name'
```

Windows

```
aws ssm get-parameters-by-path ^
    --path /aws/service/ec2-macos ^
    --query Parameters[].Name
```

The command returns information like the following.


```
[
    "/aws/service/ec2-macos/sonoma/x86_64_mac/latest/image_id",
    "/aws/service/ec2-macos/ventura/x86_64_mac/latest/image_id",
    "/aws/service/ec2-macos/sonoma/arm64_mac/latest/image_id",
    "/aws/service/ec2-macos/ventura/arm64_mac/latest/image_id"
```

```
]
```

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/ec2-macos" \
    --region region
```

Windows

```
aws ssm get-parameters-by-path ^
    --path "/aws/service/ec2-macos" ^
    --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": [
        ...sample results pending...
    ]
}
```

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/ec2-macos/...pending... \
    --region us-east-2
```

Windows

```
aws ssm get-parameters ^
     --names /aws/service/ec2-macos/...pending... ^
    --region us-east-2
```

The command returns the following information.

```
{
    "Parameters": [
         ...sample results pending...
    ],
    "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/EC2LaunchV2-Windows_Server-2016-English-Full-
Base",
```

```
    "/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-
SQL_2014_SP3_Enterprise",
    "/aws/service/ami-windows-latest/Windows_Server-2016-German-Full-Base",
    "/aws/service/ami-windows-latest/Windows_Server-2016-Japanese-Full-
SQL_2016_SP3_Standard",
    "/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-1.25",
    "/aws/service/ami-windows-latest/Windows_Server-2019-Italian-Full-Base",
    "/aws/service/ami-windows-latest/Windows_Server-2022-Japanese-Full-
SQL_2019_Enterprise",
    "/aws/service/ami-windows-latest/Windows_Server-2022-Portuguese_Brazil-Full-Base",
    "/aws/service/ami-windows-latest/amzn2-ami-hvm-2.0.20191217.0-x86_64-gp2-mono",
    "/aws/service/ami-windows-latest/Windows_Server-2016-English-Deep-Learning",
    "/aws/service/ami-windows-latest/Windows_Server-2016-Japanese-Full-
SQL_2016_SP3_Web",
    "/aws/service/ami-windows-latest/Windows_Server-2016-Korean-Full-Base",
    "/aws/service/ami-windows-latest/Windows_Server-2019-English-STIG-Core",
    "/aws/service/ami-windows-latest/Windows_Server-2019-French-Full-Base",
    "/aws/service/ami-windows-latest/Windows_Server-2019-Japanese-Full-
SQL_2017_Enterprise",
    "/aws/service/ami-windows-latest/Windows_Server-2019-Korean-Full-Base",
    "/aws/service/ami-windows-latest/Windows_Server-2022-English-Full-SQL_2022_Web",
    "/aws/service/ami-windows-latest/Windows_Server-2022-Italian-Full-Base",
    "/aws/service/ami-windows-latest/amzn2-x86_64-SQL_2019_Express",
    "/aws/service/ami-windows-latest/EC2LaunchV2-Windows_Server-2016-English-Core-
Base",
    "/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-
SQL_2019_Enterprise",
    "/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-
SQL_2019_Standard",
    "/aws/service/ami-windows-latest/Windows_Server-2016-Portuguese_Portugal-Full-
Base",
    "/aws/service/ami-windows-latest/Windows_Server-2019-English-Core-
EKS_Optimized-1.24",
    "/aws/service/ami-windows-latest/Windows_Server-2019-English-Deep-Learning",
    "/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-SQL_2017_Web",
    "/aws/service/ami-windows-latest/Windows_Server-2019-Hungarian-Full-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/EC2LaunchV2-Windows_Server-2016-
English-Full-Base",
            "Type": "String",
            "Value": "ami-0a30b2e65863e2d16",
            "Version": 36,
            "LastModifiedDate": "2024-03-15T15:58:37.976000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-windows-latest/
EC2LaunchV2-Windows_Server-2016-English-Full-Base",
            "DataType": "text"
        },
        {
            "Name": "/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-
SQL_2014_SP3_Enterprise",
            "Type": "String",
            "Value": "ami-001f20c053dd120ce",
            "Version": 69,
            "LastModifiedDate": "2024-03-15T15:53:58.905000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-windows-latest/
Windows_Server-2016-English-Full-SQL_2014_SP3_Enterprise",
```

```
            "DataType": "text"
        },
        {
            "Name": "/aws/service/ami-windows-latest/Windows_Server-2016-German-Full-
Base",
            "Type": "String",
            "Value": "ami-063be4935453e94e9",
            "Version": 102,
            "LastModifiedDate": "2024-03-15T15:51:12.003000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-windows-latest/
Windows_Server-2016-German-Full-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/EC2LaunchV2-Windows_Server-2016-English-
Full-Base \
    --region us-east-2
```

Windows

```
aws ssm get-parameters ^
    --names /aws/service/ami-windows-latest/EC2LaunchV2-Windows_Server-2016-English-
Full-Base ^
    --region us-east-2
```

The command returns the following information.

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ami-windows-latest/EC2LaunchV2-Windows_Server-2016-
English-Full-Base",
            "Type": "String",
```

```
            "Value": "ami-0a30b2e65863e2d16",
            "Version": 36,
            "LastModifiedDate": "2024-03-15T15:58:37.976000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-windows-latest/
EC2LaunchV2-Windows_Server-2016-English-Full-Base",
            "DataType": "text"
        }
    ],
    "InvalidParameters": []
}
```

Calling the ECS optimized AMI public parameter in Parameter Store

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 in Parameter Store

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 in Parameter Store

You can call the AWS Region, service, endpoint, Availability, and Wavelength Zones of public parameters by using the following path.

`/aws/service/global-infrastructure`

> **Note**
>
> Currently, the path `/aws/service/global-infrastructure` is supported for queries in the following AWS Regions only:
>
> - US East (N. Virginia) (us-east-1)
> - US East (Ohio) (us-east-2)
> - US West (N. California) (us-west-1)
> - US West (Oregon) (us-west-2)
> - Asia Pacific (Hong Kong) (ap-east-1)
> - Asia Pacific (Mumbai) (ap-south-1)
> - Asia Pacific (Seoul) (ap-northeast-2)
> - Asia Pacific (Singapore) (ap-southeast-1)
> - Asia Pacific (Sydney) (ap-southeast-2)
> - Asia Pacific (Tokyo) (ap-northeast-1)
> - Canada (Central) (ca-central-1)

- Europe (Frankfurt) (eu-central-1)
- Europe (Ireland) (eu-west-1)
- Europe (London) (eu-west-2)
- Europe (Paris) (eu-west-3)
- Europe (Stockholm) (eu-north-1)
- South America (São Paulo) (sa-east-1)

If you are working in a different commercial Region, you can specify a supported Region in your query to view results. For example, if you are working in the Canada West (Calgary) (ca-west-1) Region, you could specify Canada (Central) (ca-central-1) in your query:

```
aws ssm get-parameters-by-path \
    --path /aws/service/global-infrastructure/regions \
    --region ca-central-1
```

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-south-2",
    "/aws/service/global-infrastructure/regions/ca-central-1",
    "/aws/service/global-infrastructure/regions/eu-central-2",
    "/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-southeast-1",
    "/aws/service/global-infrastructure/regions/ap-southeast-2",
    "/aws/service/global-infrastructure/regions/ap-southeast-3",
    "/aws/service/global-infrastructure/regions/cn-north-1",
    "/aws/service/global-infrastructure/regions/cn-northwest-1",
    "/aws/service/global-infrastructure/regions/eu-south-1",
    "/aws/service/global-infrastructure/regions/eu-south-2",
    "/aws/service/global-infrastructure/regions/us-east-2",
    "/aws/service/global-infrastructure/regions/us-west-1",
    "/aws/service/global-infrastructure/regions/ap-northeast-1",
    "/aws/service/global-infrastructure/regions/ap-south-1",
    "/aws/service/global-infrastructure/regions/ap-southeast-4",
    "/aws/service/global-infrastructure/regions/ca-west-1",
    "/aws/service/global-infrastructure/regions/eu-central-1",
    "/aws/service/global-infrastructure/regions/il-central-1",
    "/aws/service/global-infrastructure/regions/me-central-1",
    "/aws/service/global-infrastructure/regions/me-south-1",
    "/aws/service/global-infrastructure/regions/sa-east-1",
    "/aws/service/global-infrastructure/regions/us-gov-east-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. This example has been truncated for space.

```
[
     "/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/amazonlocationservice",
    "/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",
    "/aws/service/global-infrastructure/services/appstream",
    "/aws/service/global-infrastructure/services/appsync",
    "/aws/service/global-infrastructure/services/aps",
    "/aws/service/global-infrastructure/services/arc-zonal-shift",
    "/aws/service/global-infrastructure/services/artifact",
    "/aws/service/global-infrastructure/services/athena",
    "/aws/service/global-infrastructure/services/auditmanager",
    "/aws/service/global-infrastructure/services/augmentedairuntime",
    "/aws/service/global-infrastructure/services/aurora",
    "/aws/service/global-infrastructure/services/autoscaling",
```

```
    "/aws/service/global-infrastructure/services/aws-appfabric",
    "/aws/service/global-infrastructure/services/awshealthdashboard",
```

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-south-1",
    "eu-central-1",
    "eu-central-2",
    "eu-west-1",
    "eu-west-2",
    "eu-west-3",
    "il-central-1",
    "me-south-1",
    "us-east-2",
    "us-gov-west-1",
    "af-south-1",
    "ap-northeast-3",
    "ap-southeast-1",
    "ap-southeast-4",
    "ca-central-1",
    "ca-west-1",
    "cn-north-1",
    "eu-north-1",
```

```
    "eu-south-2",
    "us-west-1",
    "ap-east-1",
    "ap-northeast-1",
    "ap-northeast-2",
    "ap-southeast-2",
    "ap-southeast-3",
    "cn-northwest-1",
    "eu-south-1",
    "me-central-1",
    "us-gov-east-1",
    "us-west-2",
    "ap-south-2",
    "sa-east-1",
    "us-east-1"
]
```

View the Regional endpoint for a service

You can view a Regional endpoint for a service by using the following command. This command queries the US East (Ohio) (us-east-2) Region.

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/afs1-az3",
            "Type": "String",
            "Value": "afs1-az3",
            "Version": 1,
            "LastModifiedDate": "2020-04-21T12:05:35.375000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
availability-zones/afs1-az3",
            "DataType": "text"
        },
        {
            "Name": "/aws/service/global-infrastructure/availability-zones/aps1-az2",
            "Type": "String",
            "Value": "aps1-az2",
            "Version": 1,
            "LastModifiedDate": "2020-04-03T16:13:57.351000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
availability-zones/aps1-az2",
            "DataType": "text"
        },
        {
            "Name": "/aws/service/global-infrastructure/availability-zones/apse3-az1",
            "Type": "String",
            "Value": "apse3-az1",
            "Version": 1,
            "LastModifiedDate": "2021-12-13T08:51:38.983000-05:00",
```

```
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
availability-zones/apse3-az1",
            "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-2/availability-zones \
    --query 'Parameters[].Name | sort(@)'
```

Windows

```
aws ssm get-parameters-by-path ^
    --path /aws/service/global-infrastructure/regions/us-east-2/availability-zones ^
    --query "Parameters[].Name | sort(@)"
```

The command returns information like the following.

```
[
    "/aws/service/global-infrastructure/regions/us-east-2/availability-zones/use2-az1",
    "/aws/service/global-infrastructure/regions/us-east-2/availability-zones/use2-az2",
    "/aws/service/global-infrastructure/regions/us-east-2/availability-zones/use2-az3"
```

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/apne1-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/afs1-los1-az1",
            "Type": "String",
            "Value": "afs1-los1-az1",
            "Version": 1,
            "LastModifiedDate": "2023-01-25T11:53:11.690000-05:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
local-zones/afs1-los1-az1",
```

```
            "DataType": "text"
        },
        {
            "Name": "/aws/service/global-infrastructure/local-zones/apne1-tpe1-az1",
            "Type": "String",
            "Value": "apne1-tpe1-az1",
            "Version": 1,
            "LastModifiedDate": "2024-03-15T12:35:41.076000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
local-zones/apne1-tpe1-az1",
            "DataType": "text"
        },
        {
            "Name": "/aws/service/global-infrastructure/local-zones/aps1-ccu1-az1",
            "Type": "String",
            "Value": "aps1-ccu1-az1",
            "Version": 1,
            "LastModifiedDate": "2022-12-19T11:34:43.351000-05:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
local-zones/aps1-ccu1-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": "2020-12-15T17:16:04.715000-05:00",
            "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/apne2-wl1-sel-
wlz1",
            "Type": "String",
            "Value": "apne2-wl1-sel-wlz1",
            "Version": 1,
            "LastModifiedDate": "2022-05-25T12:29:13.862000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
wavelength-zones/apne2-wl1-sel-wlz1",
            "DataType": "text"
        },
        {
            "Name": "/aws/service/global-infrastructure/wavelength-zones/cac1-wl1-yto-
wlz1",
            "Type": "String",
            "Value": "cac1-wl1-yto-wlz1",
            "Version": 1,
            "LastModifiedDate": "2022-04-26T09:57:44.495000-04:00",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/
wavelength-zones/cac1-wl1-yto-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. This example has been truncated for space.

```
{
    "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"
]
```

Parameter Store walkthroughs

The walkthrough in this section shows you how to create, store, and run parameters with Parameter Store, a tool in AWS Systems Manager, in a test environment. This walkthrough shows you how to use Parameter Store with other Systems Manager tools. You can also use Parameter Store with other AWS services. For more information, see What is a parameter?.

Contents

- Creating a SecureString parameter in Parameter Store and joining a node to a Domain (PowerShell)

Creating a SecureString parameter in Parameter Store and joining 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.

   In the following commands, replace each *`user input placeholder`* with your own information.

   ```
   Write-SSMParameter -Name "domainName" -Value "DOMAIN-NAME" -Type String
   Write-SSMParameter -Name "domainJoinUserName" -Value "DOMAIN\USERNAME" -Type String
   Write-SSMParameter -Name "domainJoinPassword" -Value "PASSWORD" -Type SecureString
   ```

   > **Important**
   >
   > Only the *value* of a `SecureString` parameter is encrypted. Parameter names, descriptions, and other properties 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

  > **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.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "ssm:CreateDocument"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:111122223333: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.
 Restarting\"; 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.
```

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. For more information about auditing and logging options for Systems Manager, see Logging and monitoring in AWS Systems Manager.

Troubleshooting Parameter Store

Use the following information to help you troubleshoot problems with Parameter Store, a tool in 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.

No permission to create an instance

Problem: You try to create an instance using an `aws:ec2:image` parameter but receive an error message such as "You are not authorized to perform this operation."

- **Solution**: You do not have all the permissions needed to create an EC2 instance using a parameter value, such as permissions for `ec2:RunInstances`, `ec2:DescribeImages`, and `ssm:GetParameter`, among others. Contact a user with administrator permissions in your organzation to request the necessary permissions.

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 do not have all the permissions needed for the `ec2:DescribeImages` API operation, or you lack permission to access the specific image referenced in the parameter. Contact a 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 triggering actions based on Parameter Store events.
  - 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 operations tools

Operations tools are a suite of capabilities that help you manage your AWS resources.

- Amazon CloudWatch dashboards hosted by Systems Manager are customizable home pages in the CloudWatch console that you can use to monitor your resources across AWS Regions in a single view.
- 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.
- Incident Manager helps you mitigate and recover from incidents affecting your applications hosted on AWS.
- OpsCenter provides a central location where operations engineers and IT professionals can view, investigate, and resolve operational work items (OpsItems) related to AWS resources.

Topics

- AWS Systems Manager Incident Manager
- AWS Systems Manager Explorer
- AWS Systems Manager OpsCenter
- Using Amazon CloudWatch dashboards hosted by Systems Manager

AWS Systems Manager Incident Manager

Use Incident Manager, a tool in 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

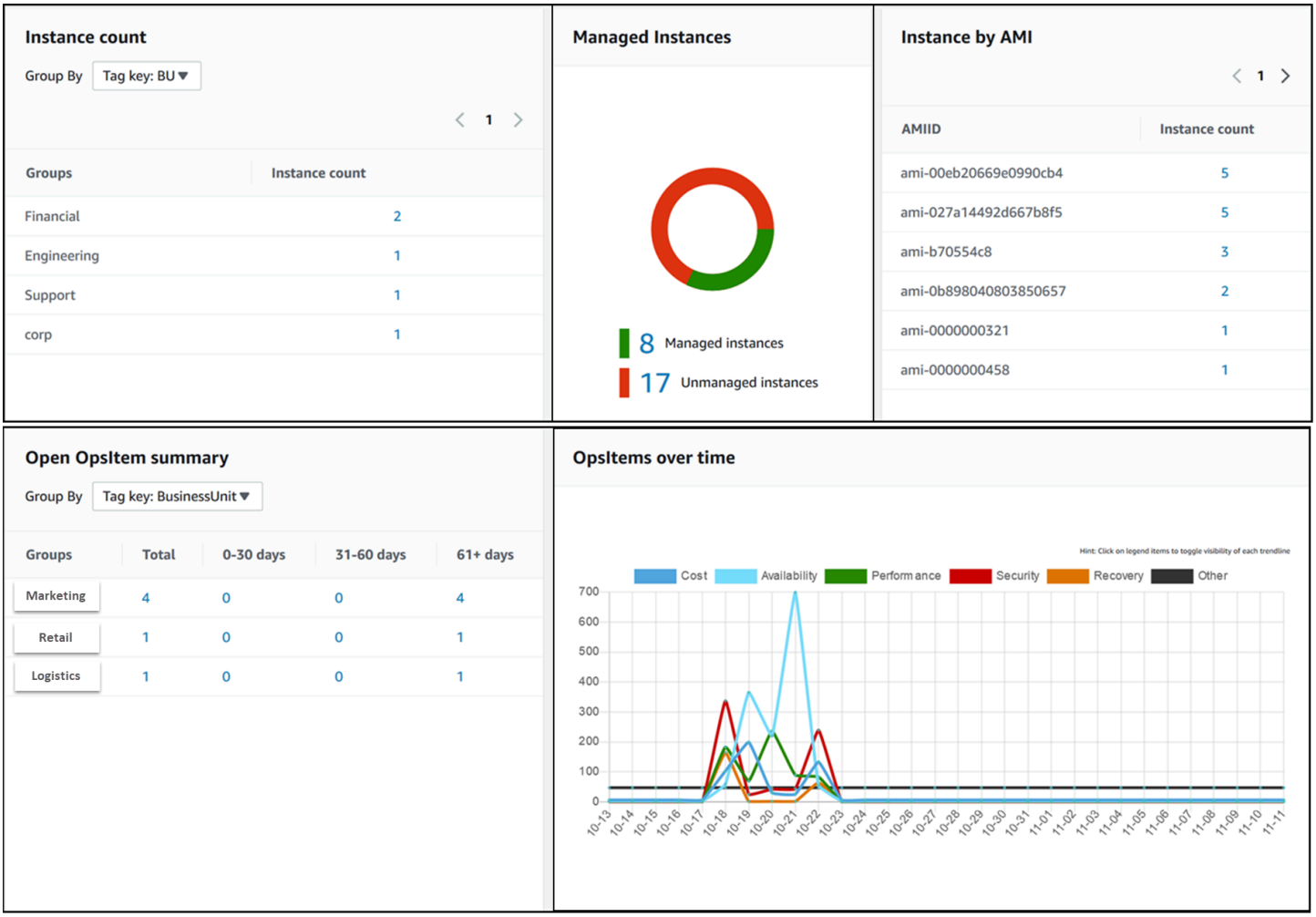
your applications to normal. To learn more about Incident Manager, see the Incident Manager User Guide.

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 the managed nodes in your hybrid and multicloud environment. OpsData also includes information provided by other Systems Manager tools, including Patch Manager patch compliance and State Manager association compliance details. To further simplify how you access OpsData, Explorer displays information from supporting AWS services like AWS Config, AWS Trusted Advisor, AWS Compute Optimizer, and AWS Support (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 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**, **Total noncompliant nodes** (patching), **Noncompliant 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 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 tool. For more information, see Getting started with Systems Manager Explorer and OpsCenter.

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 noncompliant and critical noncompliant nodes.

- **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.

- **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 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 for Explorer.

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
- Using Explorer
- Exporting OpsData from Systems Manager Explorer
- Troubleshooting Systems Manager Explorer

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 tools.

> **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: 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 user, group, or role permissions for OpsCenter, as described in this section.
- Allows default rules for OpsItem creation: 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.
- Activates OpsData sources: Integrated Setup activates the following data sources that populate Explorer widgets.
  - 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

> **Note**
>
> You can change setup configurations at any time on the **Settings** page.

After you complete Integrated Setup, we recommend that you Set up Explorer to display data from multiple Regions and accounts. 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.

Setting up related services for Explorer

AWS Systems Manager Explorer and AWS Systems Manager OpsCenter collect information from, or interact with, other AWS services and Systems Manager tools. We recommend that you set up and configure these other services or tools 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 tools.

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 tool, you must have permission to run Systems Manager

Task	Information
	Automation runbooks. For more information, see Setting up Automation.
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 AWS Systems Manager Patch Manager.
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 AWS Systems Manager State Manager.
Turn on AWS Config Configuration Recorder	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 Managing the Configuration Recorder. **Note** 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.

Task	Information
Turn on AWS Trusted Advisor	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 Support.
Turn on AWS Compute Optimizer	Explorer uses data provided by Compute Optimizer to display details a count of **Under provisioned** and **Over provisioned** EC2 instances, optimization findings, on-demand pricing details, and recommendations for instance type and price. To view this data in Explorer, turn on Compute Optimizer. For more information, see Getting started with AWS Compute Optimizer.
Turn on AWS Security Hub	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 What is AWS Security Hub.

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.

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 Using roles to collect AWS account information for OpsCenter and Explorer.
- `AmazonSSMExplorerExport`: Allows Explorer to export OpsData to a comma-separated value (CSV) file.

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 the `AWSServiceRoleForAmazonSSM_AccountDiscovery` service-linked role. Systems Manager uses this role to get information about your AWS accounts in AWS Organizations. The role uses the following permissions policy.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "organizations:DescribeAccount",
            "organizations:DescribeOrganization",
            "organizations:ListAccounts",
            "organizations:ListAWSServiceAccessForOrganization",
            "organizations:ListChildren",
            "organizations:ListParents"
         ],
         "Resource":"*"
      }
   ]
}
```

For more information about the `AWSServiceRoleForAmazonSSM_AccountDiscovery` role, see Using roles to collect AWS account information for OpsCenter and Explorer.

Configuring permissions for Systems Manager OpsCenter

After you complete Integrated Setup, you must configure user, group, or role permissions so that users can perform actions in OpsCenter.

Before you begin

You can configure OpsCenter to create and manage OpsItems for a single account or across multiple accounts. If you configure OpsCenter to create and manage OpsItems across multiple accounts, you can use either the Systems Manager delegated administrator account or the AWS Organizations management account to manually create, view, or edit OpsItems in other accounts. For more information about the Systems Manager delegated administrator account, see Configuring a delegated administrator for Explorer.

If you configure OpsCenter for a single account, you can only view or edit OpsItems in the account where OpsItems 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 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 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

If you prefer, you can specify read-only permission by adding the following inline policy to your account, group, or role.

JSON

```
{
  "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.

Create a permission using the following and add it to your users, groups, or roles:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetOpsItem",
        "ssm:UpdateOpsItem",
        "ssm:DescribeOpsItems",
        "ssm:CreateOpsItem",
```

```
        "ssm:CreateResourceDataSync",
        "ssm:DeleteResourceDataSync",
        "ssm:ListResourceDataSync",
        "ssm:UpdateResourceDataSync"

      ],
      "Resource": "*"
    }
  ]
}
```

Depending on the identity application that you are using in your organization, you can select any of the following options to configure user access.

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.
- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetOpsItem"
             ],
      "Resource": "*"
      ,
      "Condition": { "StringEquals": { "ssm:resourceTag/Department":
 "Finance" } }
    }
  ]
}
```

Here is an example that specifies API operations for viewing and updating OpsItems. This policy also specifies two sets of tag key-value pairs: Department-Finance and Project-Unity.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ssm:GetOpsItem",
            "ssm:UpdateOpsItem"
         ],
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "ssm:resourceTag/Department":"Finance",
               "ssm:resourceTag/Project":"Unity"
            }
         }
      }
```

```
    ]
}
```

For information about adding tags to an OpsItem, see Create OpsItems manually.

Understanding default EventBridge rules created by Integrated Setup

During the integrated setup process for Explorer and OpsCenter, you can choose to enable a number of default rules that are based on events detected by Amazon EventBridge. When these events are detected, the system automatically creates OpsItems in AWS Systems Manager OpsCenter.

For example, the rule `SSMOpsItems-Autoscaling-instance-termination-failure` results in an OpsItem being created when the termination of an EC2 auto scaling instance fails.

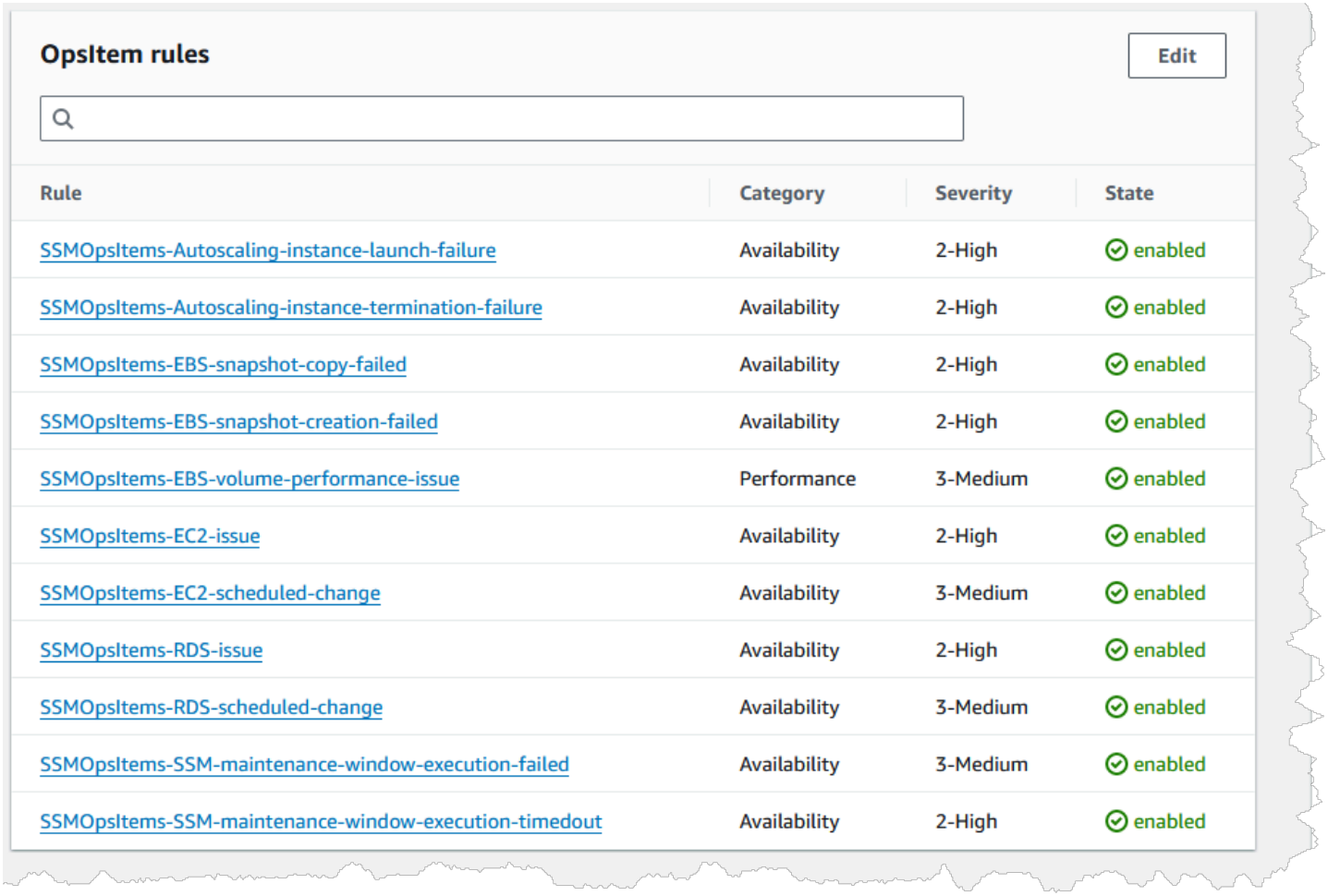
The rule `SSMOpsItems-SSM-maintenance-window-execution-failed` results in an OpsItem being created when a Systems Manager maintenace window fails to complete successfully.

For setup instructions and descriptions of all the EventBridge rules you can enable during the setup process, see Set up OpsCenter.

If you don't want EventBridge to create OpsItems for these events, you can choose not to enable this option in Integrated Setup. If you prefer, you can specify OpsCenter as the target of specific EventBridge events. For more information, see Configure EventBridge rules to create OpsItems.

You can disable a default rule or change its category and severity level in the OpsCenter **Settings** page by choosing **OpsCenter, Settings**, and then choosing **Edit** in the **OpsItem rules** area.

You can also edit the category or severity assigned to an individual OpsItem created from these rules in the Systems Manager console. For information, see Editing an OpsItem.



Configuring a delegated administrator for Explorer

If you aggregate AWS Systems Manager Explorer data from multiple AWS Regions and accounts by using resource data sync with AWS Organizations, then we recommend 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 an individual AWS account.
- You no longer need to be logged into the AWS Organizations management account to administer resource data syncs in Explorer.

A delegated administrator can use the following Explorer resource data sync APIs using the console, SDK, AWS Command Line Interface (AWS CLI), or AWS Tools for Windows PowerShell:

- CreateResourceDataSync

- DeleteResourceDataSync
- ListResourceDataSync
- UpdateResourceDataSync

A delegated administrator can search, filter, and aggregate Explorer data from 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.

A delegated administrator can create a maximum of five resource data syncs for either an entire organization or a subset of organizational units. 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.

> **Note**
>
> You can't use a delegated administrator account to create a resource data sync in opt-in AWS Regions. You must use an AWS Organizations management account.

For more information about resource data sync, see Setting up Systems Manager Explorer to display data from multiple accounts and Regions. For more information about AWS Organizations, see What is AWS Organizations? in the *AWS Organizations User Guide*.

Topics

- Before you begin
- Configure an Explorer delegated administrator
- Deregister an Explorer delegated administrator

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.

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 tools 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.

Topics

- Understanding resource data syncs for Explorer
- Understanding multiple account and Region resource data syncs
- Creating a resource data sync

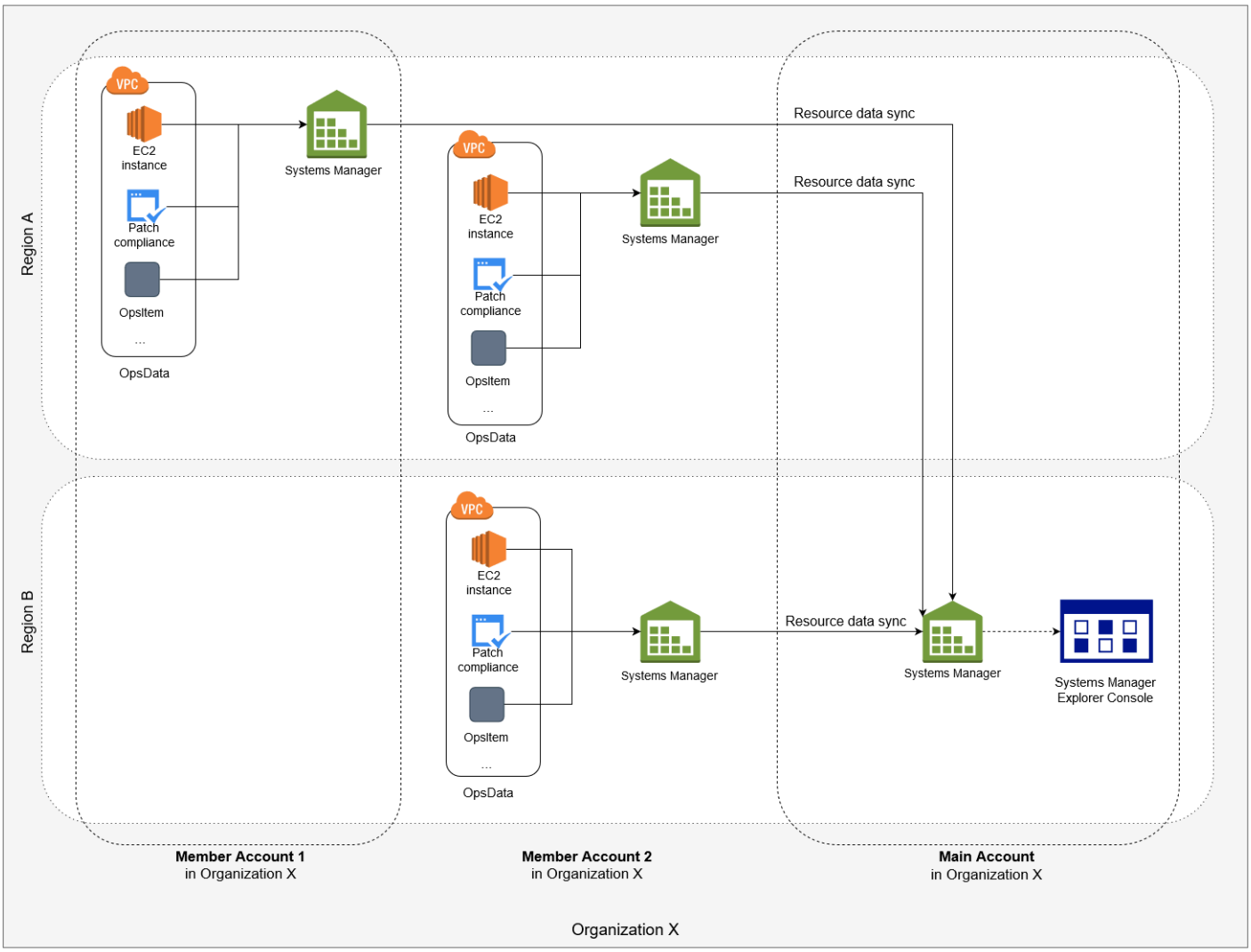
Understanding resource data syncs for Explorer

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.



Understanding 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 in the SSM console, 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. To create a resource data sync without allowing OpsData sources, specify **EnableAllOpsDataSources** as false when creating the data sync. For more information, see the `EnableAllOpsDataSources` parameter details for the ResourceDataSyncSource data type in the *Amazon EC2 Systems Manager API Reference*.

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.
- Most AWS Regions are active by default for your AWS account, but certain Regions are activated only when you manually select them. These Regions are known as opt-in Regions. By default, Explorer cross-account/cross-Region resource data syncs don't

support data aggregation in opt-in Regions. Support was added for the following opt-in Regions on June 30, 2025.

- Europe (Milan)
- Africa (Cape Town)
- Middle East (Bahrain)
- Asia Pacific (Hong Kong)

Note that you can't use a delegated administrator account to create a resource data sync in opt-in Regions. You must use an AWS Organizations management account.

Creating a resource data sync

Before you configure a 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.
- 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 for Explorer.

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.

Using 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

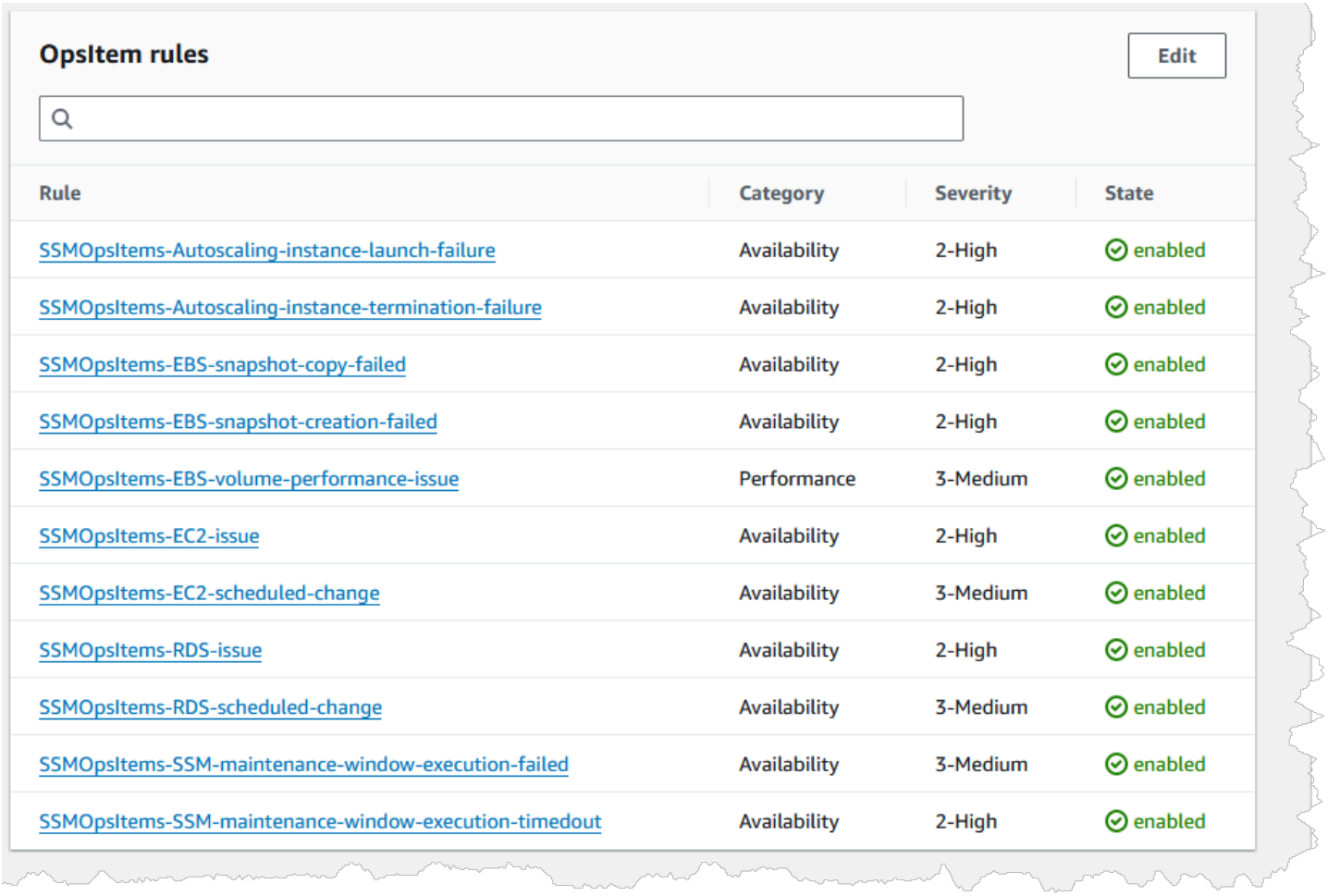
- Editing EventBridge rules created for Explorer
- Editing Systems Manager Explorer data sources
- Customizing the Explorer display
- Receiving findings from AWS Security Hub in Explorer
- Deleting a Systems Manager Explorer resource data sync

Editing EventBridge rules created for Explorer

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.



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

For AWS Regions that are available by default, 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 Inventory
- AWS Systems Manager OpsCenter OpsItems
- 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**
>
> For the Asia Pacific (Osaka) Region, Explorer doesn't display data from AWS Compute Optimizer and AWS Security Hub findings.

For the AWS opt-in Regions, Explorer displays data from the following sources:

- Amazon Elastic Compute Cloud (Amazon EC2)

- AWS Systems Manager Inventory
- AWS Systems Manager OpsCenter OpsItems
- AWS Systems Manager Patch Manager patch compliance
- AWS Systems Manager State Manager association compliance
- AWS Trusted Advisor
- AWS Support Center cases
- AWS Config rule and resource compliance

> **Note**
>
> - To view Support Center cases in Explorer, you must have either an Enterprise or Business account set up with 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 for Explorer.

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**, and then choose the **Configure Dashboard** tab.
4. In the **OpsData sources** section, in the **Status** column, turn on or turn off sources according to the data you want to view.

Customizing the Explorer display

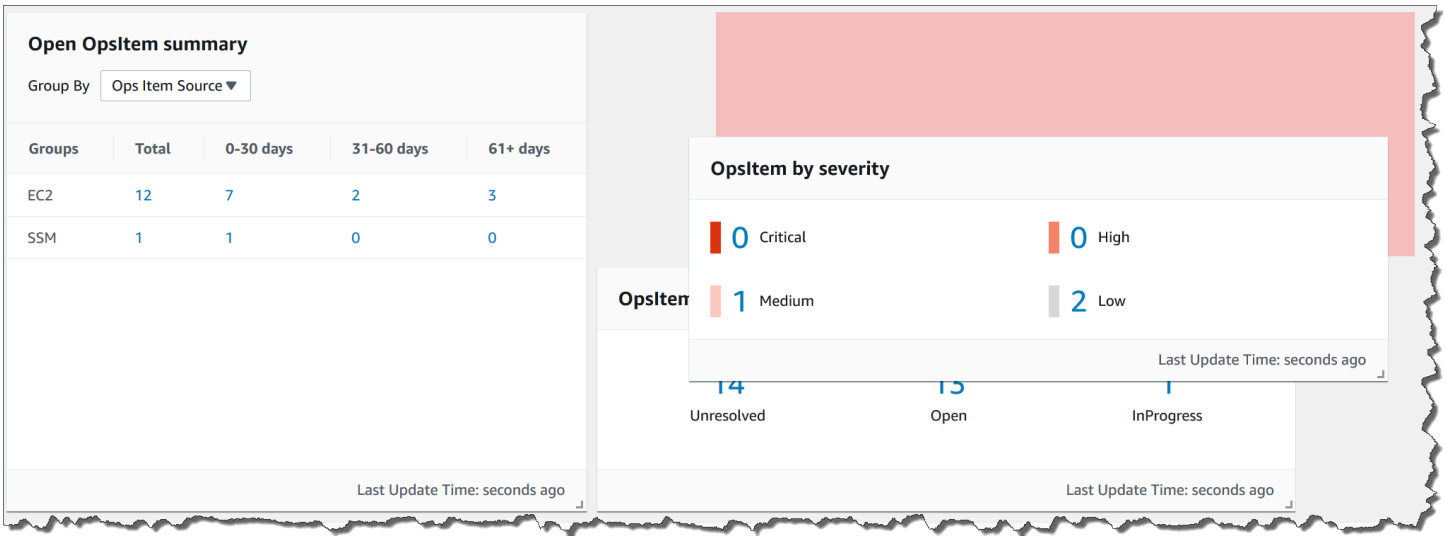
You can customize widget layout in AWS Systems Manager Explorer by using a drag-and-drop capability. You can also customize the OpsData and OpsItems displayed in Explorer by using filters, as described in this topic.

Before you customize widget layout, verify that the widgets you want to view are currently displayed in Explorer. To view some widgets in Explorer (such as the AWS Config compliance widget), you must enable them on the **Configure dashboard** page.

To enable widgets to display in 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 **Dashboard actions**, **Configure dashboard**.
4. Choose the **Configure Dashboard** tab.
5. Either choose **Enable all** or turn on an individual widget or data source.
6. Choose **Explorer** to view your changes.

To customize widget layout in Explorer, choose a widget that you want to move. 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.

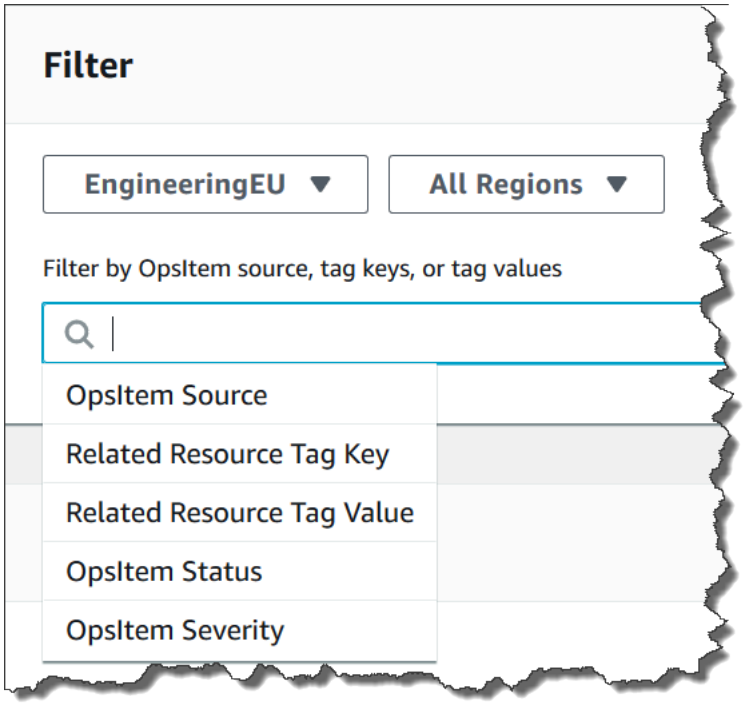
Using filters to the change the data displayed in Explorer

By default, Explorer displays data for the current AWS account and the current Region. If you create one or more resource data syncs, you can use filters to change which sync is active. You can then

choose to display data for a specific Region or all Regions. You can also use the Search bar to filter on different OpsItem and key-tag criteria.

To change the data displayed in Explorer by using filters

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Explorer**.
3. In the **Filter** section, use the **Select a resource data sync** list to choose a sync.
4. Use the **Regions** list to choose either a specific AWS Region or choose **All Regions**.
5. Choose the Search bar, and then choose the criteria on which to filter the data.



6. Press Enter.

Explorer retains the filter options you selected if you close and reopen the page.

Receiving findings from AWS Security Hub in Explorer

AWS Security Hub provides a comprehensive view of your security state in AWS. The service collects security data, called *findings*, from across AWS accounts, services, and supported third-party products. Security Hub findings can help you check your environment against security industry

standards and best practices, analyze your security trends, and identify the highest priority security issues.

Security Hub sends findings to Amazon EventBridge, which uses an event rule to send the findings to Explorer. After you enable integration, as described here, you can view Security Hub findings in an Explorer widget and view finding details in OpsCenter OpsItems. The widget provides a summary of all Security Hub findings based on severity. New findings in Security Hub are usually visible in Explorer within seconds of being created.

Warning

Note the following important information:

- Explorer is integrated with OpsCenter, a tool in Systems Manager. After you enable Explorer integration with Security Hub, OpsCenter automatically creates OpsItems for Security Hub findings. Depending on your AWS environment, enabling integration can result in large numbers of OpsItems, at a cost.

  Before you continue, read about OpsCenter integration with Security Hub. The topic includes specific details about how changes and updates to findings and OpsItems are charged to your account. For more information, see Understanding OpsCenter integration with AWS Security Hub. For OpsCenter pricing information, see AWS Systems Manager Pricing.
- If you create a resource data sync in Explorer while logged into the administrator account, Security Hub integration is automatically enabled for the administrator and all member accounts in the sync. Once enabled, OpsCenter automatically creates OpsItems for Security Hub findings, at a cost. For more information about creating a resource data sync, see Setting up Systems Manager Explorer to display data from multiple accounts and Regions.

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. By default, Explorer creates OpsItems for critical and high severity findings. You can manually configure Explorer to create OpsItems for medium and low severity findings.

Though Explorer doesn't create OpsItems for informational findings, you can view informational operations data (OpsData) in the Security Hub findings summary widget. Explorer creates OpsData for all findings regardless of severity. For more information about Security Hub severity levels, see Severity in the *AWS Security Hub API Reference*.

Enabling integration

This section describes how to enable and configure Explorer to start receiving Security Hub findings.

Before you begin

Complete the following tasks before you configure Explorer to start receiving Security Hub findings.

- Enable 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 severity findings are displayed by default. To display medium and low severity 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**.

> **Important**
>
> If the option to disable Security Hub findings is grayed out in the console, you can disable this setting by running the following command in the AWS CLI. You must run the command while logged into either the AWS Organizations management account or the Systems Manager delegated administrator account. For the `region` parameter, specify the AWS Region where you want to stop receiving Security Hub findings in Explorer.

```
aws ssm update-service-setting --setting-id /ssm/opsdata/SecurityHub --setting-
value Disabled --region AWS Region
```

Here's an example.

```
aws ssm update-service-setting --setting-id /ssm/opsdata/SecurityHub --setting-
value Disabled --region us-east-1
```

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**.

Exporting OpsData from Systems Manager Explorer

You can export 5,000 OpsData items as a comma separated value (.csv) file to an Amazon Simple Storage Service (Amazon S3) bucket from AWS Systems Manager Explorer. Explorer uses the AWS-

ExportOpsDataToS3 automation runbook to export OpsData. When you export OpsData, the system displays the automation runbook page where you can specify details, such as assumeRole, Amazon S3 bucket name, SNS topic ARN, and fields to be exported.

To export OpsData:

- Step 1: Specifying an SNS topic
- Step 2: (Optional) Configuring data export
- Step 3: Exporting OpsData

Step 1: Specifying an SNS topic

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 Creating an Amazon SNS topic.

Step 2: (Optional) Configuring data export

You can configure data export settings from the **Settings** or **Export Ops Data to S3 Bucket** page.

To configure data export 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 Amazon S3 bucket, choose **Select an existing S3 bucket** and choose the bucket from the list.

   To upload the data export file to a new Amazon S3 bucket, choose **Create a new S3 bucket** and enter the name that you want to use for the new bucket.

   > **Note**
   >
   > You can only edit the Amazon S3 bucket name and Amazon SNS topic ARN from the page where you configured those settings for the first time in Explorer. If you set up

the Amazon S3 bucket and the Amazon SNS topic ARN from the **Settings** page, then you can only modify those settings from the **Settings** page.

6. For **Select an Amazon SNS topic ARN**, choose the topic that you want to notify when the export is complete.
7. Choose **Create**.

Step 3: Exporting OpsData

When you export Explorer data, Systems Manager creates an AWS Identity and Access Management (IAM) role named `AmazonSSMExplorerExportRole`. This role uses the following IAM policy.

JSON

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
              "Sid": "OpsSummaryExportAutomationServiceRoleStatement1",
              "Effect": "Allow",
              "Action": [
                  "s3:PutObject"
              ],
              "Resource": [
                 "arn:aws:s3:::amzn-s3-demo-bucket/*"
              ]
          },
          {
              "Sid": "OpsSummaryExportAutomationServiceRoleStatement2",
              "Effect": "Allow",
              "Action": [
                  "s3:GetBucketAcl",
                  "s3:GetBucketLocation"
              ],
              "Resource": [
                 "arn:aws:s3:::amzn-s3-demo-bucket"
              ]
          },
          {
              "Sid": "OpsSummaryExportAutomationServiceRoleStatement3",
              "Effect": "Allow",
```

```
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1:111122223333:SNSTopicName"
            ]
        },
        {
            "Sid": "OpsSummaryExportAutomationServiceRoleStatement4",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:MyLogGroup"
            ]
        },
        {
            "Sid": "OpsSummaryExportAutomationServiceRoleStatement5",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:PutLogEvents",
                "logs:CreateLogStream"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "OpsSummaryExportAutomationServiceRoleStatement6",
            "Effect": "Allow",
            "Action": [
                "ssm:GetOpsSummary"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

The role includes the following trust entity.

JSON

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "OpsSummaryExportAutomationServiceRoleTrustPolicy",
          "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 a data link in an Explorer widget. For example, choose **Unresolved** or **Open issues** in the **OpsItems by status** widget. Or choose the **Critical** or **High** data link in the **OpsItem by severity** widget. Explorer opens the **OpsData** page for the selected data set.
4. Choose **Export Table**.

> **Note**
>
> When you export OpsData for the first time, the system creates an assume role for the export. You can't modify the default assume role.

5. For **S3 Bucket Name**, choose an existing bucket. You can choose **Create** to create an Amazon S3 bucket if needed.

   If you can't change the S3 bucket name, it means that you configured the bucket name from the **Settings** page. You can only change the bucket name from the **Settings** page.

> **Note**
>
> You can only edit the Amazon S3 bucket name and Amazon SNS topic ARN from the page where you configured those settings for the first time in Explorer.

6. For **SNS Topic Arn**, choose an existing Amazon SNS topic ARN to notify when the download completes.

   If you can't change the Amazon SNS topic ARN, it means that you configured the Amazon SNS topic ARN from the **Settings** page. You can only change the topic ARN from the **Settings** page.

7. (Optional) For **SNS Success Message**, specify a success message that you want to display when the export is successfully completed.

8. Choose **Submit**. The system navigates to the previous page and displays the message **Click to view status of export process. View details**.

   You can choose **View details** to view the status of the runbook and progress in Systems Manager Automation.

You can now export OpsData from Explorer to the specified Amazon S3 bucket.

If you can't export data by using this procedure, verify that your user, group, or role includes the `iam:CreatePolicyVersion` and `iam:DeletePolicyVersion` actions. For information about adding these actions to your user, 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.

Resource data sync stopped working

If your resource data sync has stopped working for an *opt-in AWS Region*, verify that the Region was opted into for both the AWS Organizations management account and the member account for which the sync is configured.

Most AWS Regions are active by default for your AWS account, but certain Regions are activated only when you manually select them. These Regions are known as opt-in Regions. By default,

Explorer cross-account/cross-Region resource data syncs don't support data aggregation in opt-in Regions. Support was added for the following opt-in Regions on June 30, 2025.

- Europe (Milan)
- Africa (Cape Town)
- Middle East (Bahrain)
- Asia Pacific (Hong Kong)

Note that you can't use a delegated administrator account to create a resource data sync in opt-in Regions. You must use an AWS Organizations management account.

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 and Configuring a delegated administrator for Explorer.

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.
- When using Explorer to view data from multiple accounts and Regions, verify that you're logged into the AWS Organizations management account or the Explorer delegated administrator 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 or the Explorer delegated administator 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.

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 tool in 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. OpsCenter aggregates and standardizes OpsItems across services while providing contextual investigation data about each OpsItem, related OpsItems, and related resources. OpsCenter also provides Systems Manager Automation 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.

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 Adding related resources to an OpsItem.

- **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 Managing duplicate OpsItems.

- **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 Remediate OpsItem issues.

- **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.
- **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 Viewing OpsCenter logs and reports.
- **Console, CLI, PowerShell, and SDK access to OpsCenter tool**

  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.

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 Managing nodes in hybrid and multicloud environments with Systems Manager.

What are the quotas for OpsCenter?

You can view quotas for all Systems Manager tools in the Systems Manager service quotas in the *Amazon Web Services General Reference*. Unless otherwise noted, each quota is Region-specific.

Set up OpsCenter

AWS Systems Manager uses an integrated setup experience to help you get started with OpsCenter and Explorer, which are tools in Systems Manager. Explorer is a customizable operations dashboard that reports information about your AWS resources. In this documentation, Explorer and OpsCenter setup is called *Integrated Setup*.

You must use Integrated Setup to set up OpsCenter with Explorer. Integrated Setup is only available in the AWS Systems Manager console. You can't set up Explorer and OpsCenter programmatically. For more information, see Getting started with Systems Manager Explorer and OpsCenter.

Before you begin

When you set up OpsCenter, you enable default rules in Amazon EventBridge that automatically create OpsItems. The following table describes the default EventBridge rules that automatically

create OpsItems. You can disable EventBridge rules in the OpsCenter **Settings** page under **OpsItem rules**.

> **Important**
>
> Your account is charged for OpsItems created by default rules. For more information, see AWS Systems Manager Pricing.

Rule name	Description
SSMOpsItems-Autoscaling-instance-launch-failure	This rule creates OpsItems when the launch of an EC2 auto scaling instance failed.
SSMOpsItems-Autoscaling-instance-termination-failure	This rule creates OpsItems when the termination of an EC2 auto scaling instance failed.
SSMOpsItems-EBS-snapshot-copy-failed	This rule creates OpsItems when the system failed to copy an Amazon Elastic Block Store (Amazon EBS) snapshot.
SSMOpsItems-EBS-snapshot-creation-failed	This rule creates OpsItems when the system failed to create an Amazon EBS snapshot.
SSMOpsItems-EBS-volume-performance-issue	This rule corresponds to an AWS Health tracking rule. The rule creates OpsItems whenever there is a performance issue with an Amazon EBS volume (health event = `AWS_EBS_DEGRADED_EBS_VOLUME_PERFORMANCE`).
SSMOpsItems-EC2-issue	This rule corresponds to an AWS Health tracking rule for unexpected events that affect AWS services or resources. The rule creates OpsItems when, for example, a service sends communications about operational issues that are causing service degradation or to raise awarness about localized resource-level issues.

Rule name	Description
	For example, this rule creates an OpsItem for the following event: `AWS_EC2_OPERATIONAL_ISSUE` .
SSMOpsItems-EC2-scheduled-change	This rule corresponds to an AWS Health tracking rule. AWS can schedule events for your instances, such as rebooting, stopping, or starting instances. The rule creates OpsItems for EC2 scheduled events. For more information about scheduled events, see Scheduled events for your instances in the *Amazon EC2 User Guide*.
SSMOpsItems-RDS-issue	This rule corresponds to an AWS Health tracking rule for unexpected events that affect AWS services or resources. The rule creates OpsItems when, for example, a service sends communications about operational issues that are causing service degradation or to raise awarness about localized resource-level issues. For example, this rule creates an OpsItem for the following events: `AWS_RDS_MYSQL_DATABASE_CRASHING_REPEATEDLY` , `AWS_RDS_EXPORT_TASK_FAILED` , and `AWS_RDS_CONNECTIVITY_ISSUE` .

Rule name	Description
SSMOpsItems-RDS-scheduled-change	This rule corresponds to an AWS Health tracking rule. The rule creates OpsItems for Amazon RDS scheduled events. Scheduled events provide information about upcoming changes to your Amazon RDS resources. Some events might recommend that you take action to avoid service disruptions. Other events occur automatically without any action on your part. Your resource might be temporari ly unavailable during the scheduled change activity. For example, this rule creates an OpsItem for the following events: `AWS_RDS_S YSTEM_UPGRADE_SCHEDULED` and `AWS_RDS_MAINTENANCE_SCHEDULED`. For more information about scheduled events, see Event type categories in the *AWS Health User Guide*.
SSMOpsItems-SSM-maintenance-window-e xecution-failed	This rule creates OpsItems when the processin g of the Systems Manager maintenance window failed.
SSMOpsItems-SSM-maintenance-window-e xecution-timedout	This rule creates OpsItems when the launch of the Systems Manager maintenance window timed out.

Use the following procedure to set up OpsCenter.

To set up OpsCenter

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 OpsCenter home page, choose **Get started**.

4. On the OpsCenter setup page, choose **Enable this option to have Explorer configure AWS Config and Amazon CloudWatch events to automatically create OpsItems based on commonly-used rules and events**. If you don't choose this option, OpsCenter remains disabled.

   > **Note**
   >
   > Amazon EventBridge (formerly Amazon CloudWatch Events) provides all functionality of CloudWatch Events and some new features, such as custom event buses, third-party event sources and schema registry.

5. Choose **Enable OpsCenter**.

After you enable OpsCenter, you can do the following from **Settings**:

- Create CloudWatch alarms using the **Open CloudWatch console** button. For more information, see Configure CloudWatch alarms to create OpsItems.
- Enable operational insights. For more information, see Analyzing operational insights to reduce OpsItems.
- Enable AWS Security Hub findings alarms. For more information, see Understanding OpsCenter integration with AWS Security Hub.

Contents

- (Optional) Setting up OpsCenter to centrally manage OpsItems across accounts
- (Optional) Set up Amazon SNS to receive notifications about OpsItems

(Optional) Setting up OpsCenter to centrally manage OpsItems across accounts

You can use Systems Manager OpsCenter to centrally manage OpsItems across multiple AWS accounts in a selected AWS Region. This feature is available after you set up your organization in AWS Organizations. AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage. AWS Organizations includes account management and consolidated billing capabilities that enable you to better meet the budgetary, security, and compliance needs of your business. For more information, see What is AWS Organizations? in the *AWS Organizations User Guide*

Users who belong to the AWS Organizations management account can set up a delegated administrator account for Systems Manager. In the context of OpsCenter, delegated administrators can create, edit, and view OpsItems in member accounts. The delegated administrator can also use Systems Manager Automation runbooks to bulk resolve OpsItems or remediate issues with AWS resources that are generating OpsItems.

> **Note**
>
> You can assign only one account as the delegated administrator for Systems Manager. For more information, see Creating an AWS Organizations delegated administrator for Systems Manager.

Systems Manager offers the following methods for setting up OpsCenter to centrally manage OpsItems across multiple AWS accounts.

- **Quick Setup**: Quick Setup, a tool in Systems Manager, simplifies set up and configuration tasks for Systems Manager tools. For more information, see AWS Systems Manager Quick Setup.

  Quick Setup for OpsCenter helps you complete the following tasks for managing OpsItems across accounts:

  - Registering an account as the delegated administrator (if the delegated administrator hasn't already been designated)
  - Creating required AWS Identity and Access Management (IAM) policies and roles
  - Specifying an AWS Organizations organization or organizational units (OUs) where a delegated administrator can manage OpsItems across accounts

  For more information, see (Optional) Configure OpsCenter to manage OpsItems across accounts by using Quick Setup.

  > **Note**
  >
  > Quick Setup isn't available in all AWS Regions where Systems Manager is currently available. If Quick Setup isn't available in a Region where you want to use it to configure OpsCenter to centrally manage OpsItems across multiple accounts, then you must use the manual method. To view a list of AWS Regions where Quick Setup is available, see Availability of Quick Setup in AWS Regions.

- **Manual set up**: If Quick Setup isn't available in the Region where you want to configure OpsCenter to centrally manage OpsItems across accounts, then you can use the manual procedure to do so. For more information, see (Optional) Manually set up OpsCenter to centrally manage OpsItems across accounts.

(Optional) Configure OpsCenter to manage OpsItems across accounts by using Quick Setup

Quick Setup, a tool in AWS Systems Manager, simplifies setup and configuration tasks for Systems Manager tools. Quick Setup for OpsCenter helps you complete the following tasks for managing OpsItems across accounts:

- Specifying the delegated administrator account
- Creating required AWS Identity and Access Management (IAM) policies and roles
- Specifying an AWS Organizations organization, or a subset of member accounts, where a delegated administrator can manage OpsItems across accounts

When you configure OpsCenter to manage OpsItems across accounts by using Quick Setup, Quick Setup creates the following resources in the specified accounts. These resources give the specified accounts permission to work with OpsItems and use Automation runbooks to fix issues with AWS resources generating OpsItems.

Resources	Accounts
`AWSServiceRoleForAmazonSSM_AccountDiscovery` AWS Identity and Access Management (IAM) service-linked role For more information about this role, see Using roles to collect AWS account information for OpsCenter and Explorer.	AWS Organizations management account and delegated administrator account
`OpsItem-CrossAccountManagementRole` IAM role `AWS-SystemsManager-AutomationAdministrationRole` IAM role	Delegated administrator account

Resources	Accounts
`OpsItem-CrossAccountExecutionRole` IAM role `AWS-SystemsManager-AutomationExecutionRole` IAM role `AWS::SSM::ResourcePolicy` Systems Manager resource policy for the default OpsItem group (`OpsItemGroup`)	All AWS Organizations member accounts

Note

If you previously configured OpsCenter to manage OpsItems across accounts using the manual method, you must delete the AWS CloudFormation stacks or stack sets created during Steps 4 and 5 of that process. If those resources exist in your account when you complete the following procedure, Quick Setup fails to configure cross-account OpsItem management properly.

To configure OpsCenter to manage OpsItems across accounts by using Quick Setup

1. Sign in to the AWS Management Console using the 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 **Quick Setup**.
4. Choose the **Library** tab.
5. Scroll to the bottom and locate the **OpsCenter** configuration tile. Choose **Create**.
6. On the Quick Setup OpsCenter page, in the **Delegated administrator** section, enter an account ID. If you are unable to edit this field, then a delegated administrator account has already been specified for Systems Manager.
7. In the **Targets** section, choose an option. If you choose **Custom**, then select the organizational units (OU) where you want to manage OpsItems across accounts.
8. Choose **Create**.

Quick Setup creates the OpsCenter configuration and deploys the required AWS resources to the designated OUs.

> **Note**
>
> If you don't want to manage OpsItems across multiple accounts, you can delete the configuration from Quick Setup. When you delete the configuration, Quick Setup deletes the following IAM policies and roles created when the configuration was originally deployed:
>
> - `OpsItem-CrossAccountManagementRole` from the delegated administrator account
> - `OpsItem-CrossAccountExecutionRole` and `SSM::ResourcePolicy` from all Organizations member accounts
>
> Quick Setup removes the configuration from all organizational units and AWS Regions where the configuration was originally deployed.

Troubleshooting issues with a Quick Setup configuration for OpsCenter

This section includes information to help you troubleshoot issues when configuring cross-account OpsItem management using Quick Setup.

Topics

- Deployment to these StackSets failed: delegatedAdmin
- Quick Setup configuration status shows Failed

Deployment to these StackSets failed: delegatedAdmin

When creating an OpsCenter configuration, Quick Setup deploys two AWS CloudFormation stack sets in the Organizations management account. The stack sets use the following prefix: `AWS-QuickSetup-SSMOpsCenter`. If Quick Setup displays the following error: `Deployment to these StackSets failed: delegatedAdmin` use the following procedure to fix this issue.

To troubleshoot a StackSets failed:delegatedAdmin error

1. If you received the `Deployment to these StackSets failed: delegatedAdmin` error in a red banner in the Quick Setup console, sign in to the delegated administrator account and the AWS Region designated as the Quick Setup home Region.
2. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
3. Choose the stack created by your Quick Setup configuration. The stack name includes the following: **AWS-QuickSetup-SSMOpsCenter**.

   > **Note**
   >
   > Sometimes CloudFormation deletes failed stack deployments. If the stack isn't available in the **Stacks** table, choose **Deleted** from the filter list.

4. View the **Status** and **Status reason**. For more information about stack statuses, see Stack status codes in the *AWS CloudFormation User Guide*.
5. To understand the exact step that failed, view the **Events** tab and review each event's **Status**. For more information, see Troubleshooting in the *AWS CloudFormation User Guide*.

> **Note**
>
> If you are unable to resolve the deployment failure using the CloudFormation troubleshooting steps, delete the configuration and try again.

Quick Setup configuration status shows Failed

If the **Configuration details** table on the **Configuration details** page shows a configuration status of `Failed`, sign in to the AWS account and Region where it failed.

To troubleshoot a Quick Setup failure to create an OpsCenter configuration

1. Sign in to the AWS account and the AWS Region where the failure occurred.
2. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
3. Choose the stack created by your Quick Setup configuration. The stack name includes the following: **AWS-QuickSetup-SSMOpsCenter**.

> **Note**
>
> Sometimes CloudFormation deletes failed stack deployments. If the stack isn't available in the **Stacks** table, choose **Deleted** from the filter list.

4. View the **Status** and **Status reason**. For more information about stack statuses, see Stack status codes in the *AWS CloudFormation User Guide*.
5. To understand the exact step that failed, view the **Events** tab and review each event's **Status**. For more information, see Troubleshooting in the *AWS CloudFormation User Guide*.

Member account configuration shows ResourcePolicyLimitExceededException

If a stack status shows `ResourcePolicyLimitExceededException`, the account has previously onboarded to OpsCenter cross-account management by using the manual method. To resolve this issue, you must delete the AWS CloudFormation stacks or stack sets created during Steps 4 and 5 of the manual onboarding process. For more information, see Delete a stack set and Deleting a stack on the AWS CloudFormation console in the *AWS CloudFormation User Guide*.

(Optional) Manually set up OpsCenter to centrally manage OpsItems across accounts

This section describes how to manually configure OpsCenter for cross-account OpsItem management. While this process is still supported, it has been replaced by a newer process that uses Systems Manager Quick Setup. For more information, see (Optional) Configure OpsCenter to manage OpsItems across accounts by using Quick Setup.

You can set up a central account to create manual OpsItems for member accounts, and manage and remediate those OpsItems. The central account can be the AWS Organizations management account, or both the AWS Organizations management account and Systems Manager delegated administrator account. We recommend that you use the Systems Manager delegated administrator account as a central account. You can only use this feature after you configure AWS Organizations.

With AWS Organizations, you can consolidate multiple AWS accounts into an organization that you create and manage centrally. The central account user can create OpsItems for all selected member accounts simultaneously, and manage those OpsItems.

Use the process in this section to enable the Systems Manager service principal in Organizations and configure AWS Identity and Access Management (IAM) permissions for working with OpsItems across accounts.

Topics

- Before you begin
- Step 1: Creating a resource data sync
- Step 2: Enabling the Systems Manager service principal in AWS Organizations
- Step 3: Creating the AWSServiceRoleForAmazonSSM_AccountDiscovery service-linked role
- Step 4: Configuring permissions to work with OpsItems across accounts
- Step 5: Configuring permissions to work with related resources across accounts

> **Note**
>
> Only OpsItems of type `/aws/issue` are supported when working in OpsCenter across accounts.

Before you begin

Before you set up OpsCenter to work with OpsItems across accounts, ensure that you have set up the following:

- A Systems Manager delegated administrator account. For more information, see Configuring a delegated administrator for Explorer.
- One organization set up and configured in Organizations. For more information, see Creating and managing an organization in the *AWS Organizations User Guide*.
- You configured Systems Manager Automation to run automation runbooks across multiple AWS Regions and AWS accounts. For more information, see Running automations in multiple AWS Regions and accounts.

Step 1: Creating a resource data sync

After you set up and configure AWS Organizations, you can aggregate OpsItems in OpsCenter for an entire organization by creating a resource data sync. For more information, see Creating a resource data sync. When you create the sync, in the **Add accounts** section, be sure to choose the **Include all accounts from my AWS Organizations configuration** option.

Step 2: Enabling the Systems Manager service principal in AWS Organizations

To enable a user to work with OpsItems across accounts, the Systems Manager service principal must be enabled in AWS Organizations. If you previously configured Systems Manager for multi-account scenarios using other tools, the Systems Manager service principal might already be configured in Organizations. Run the following commands from the AWS Command Line Interface (AWS CLI) to verify. If you *haven't* configured Systems Manager for other multi-account scenarios, skip to the next procedure, *To enable the Systems Manager service principal in AWS Organizations*.

To verify the Systems Manager service principal is enabled in AWS Organizations

1. Download the latest version of the AWS CLI to your local machine.
2. Open the AWS CLI, and run the following command to specify your credentials and an AWS Region.

   ```
   aws configure
   ```

   The system prompts you to specify the following. In the following example, replace each *`user input placeholder`* with your own information.

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

3. Run the following command to verify that the Systems Manager service principal is enabled for AWS Organizations.

   ```
   aws organizations list-aws-service-access-for-organization
   ```

   The command returns information similar to that shown in the following example.

   ```
   {
       "EnabledServicePrincipals": [
           {
               "ServicePrincipal":
    "member.org.stacksets.cloudformation.amazonaws.com",
               "DateEnabled": "2020-12-11T16:32:27.732000-08:00"
           },
           {
   ```

```
            "ServicePrincipal": "opsdatasync.ssm.amazonaws.com",
            "DateEnabled": "2022-01-19T12:30:48.352000-08:00"
        },
        {
            "ServicePrincipal": "ssm.amazonaws.com",
            "DateEnabled": "2020-12-11T16:32:26.599000-08:00"
        }
    ]
}
```

To enable the Systems Manager service principal in AWS Organizations

If you haven't previously configured the Systems Manager service principal for Organizations, use the following procedure to do so. For more information about this command, see enable-aws-service-access in the *AWS CLI Command Reference*.

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already. For information, see Installing CLI and Configuring CLI.
2. Download the latest version of the AWS CLI to your local machine.
3. Open the AWS CLI, and run the following command to specify your credentials and an AWS Region.

   ```
   aws configure
   ```

   The system prompts you to specify the following. In the following example, replace each *user input placeholder* with your own information.

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

4. Run the following command to enable the Systems Manager service principal for AWS Organizations.

   ```
   aws organizations enable-aws-service-access --service-principal "ssm.amazonaws.com"
   ```

Step 3: Creating the AWSServiceRoleForAmazonSSM_AccountDiscovery service-linked role

A service-linked role such as the AWSServiceRoleForAmazonSSM_AccountDiscovery role is a unique type of IAM role that is linked directly to an AWS service, such as Systems Manager. Service-linked roles are predefined by the service and include all the permissions that the service requires to call other AWS services on your behalf. For more information about the AWSServiceRoleForAmazonSSM_AccountDiscovery service-linked role, see Service-linked role permissions for Systems Manager account discovery.

Use the following procedure to create the AWSServiceRoleForAmazonSSM_AccountDiscovery service-linked role by using the AWS CLI. For more information about the command used in this procedure, see create-service-linked-role in the *AWS CLI Command Reference.*

To create the AWSServiceRoleForAmazonSSM_AccountDiscovery service-linked role

1. Sign in to the AWS Organizations management account.
2. While signed in to the Organizations management account, run the following command.

```
aws iam create-service-linked-role \
    --aws-service-name accountdiscovery.ssm.amazonaws.com \
    --description "Systems Manager account discovery for AWS Organizations service-
linked role"
```

Step 4: Configuring permissions to work with OpsItems across accounts

Use AWS CloudFormation stacksets to create an OpsItemGroup resource policy and an IAM execution role that give users permission to work with OpsItems across accounts. To get started, download and unzip the OpsCenterCrossAccountMembers.zip file. This file contains the OpsCenterCrossAccountMembers.yaml AWS CloudFormation template file. When you create a stack set by using this template, CloudFormation automatically creates the OpsItemCrossAccountResourcePolicy resource policy and the OpsItemCrossAccountExecutionRole execution role in the account. For more information about creating a stack set, see Create a stack set in the *AWS CloudFormation User Guide*.

> **Important**
>
> Note the following important information about this task:

- You must deploy the stackset while signed in to the AWS Organizations management account.
- You must repeat this procedure while signed in to *every* account that you want to *target* for working with OpsItems across accounts, including the delegated administrator account.
- If you want to enable cross-account OpsItems administration in different AWS Regions, choose **Add all regions** in the **Specify regions** section of the template. Cross-account OpsItem administration isn't supported for opt-in Regions.

Step 5: Configuring permissions to work with related resources across accounts

An OpsItem can include detailed information about impacted resources such as Amazon Elastic Compute Cloud (Amazon EC2) instances or Amazon Simple Storage Service (Amazon S3) buckets. The `OpsItemCrossAccountExecutionRole` execution role, which you created in the previous Step 4, provides OpsCenter with read-only permissions for member accounts to view related resources. You must also create an IAM role to provide management accounts with permission to view and interact with related resources, which you will complete in this task.

To get started, download and unzip the `OpsCenterCrossAccountManagementRole.zip` file. This file contains the `OpsCenterCrossAccountManagementRole.yaml` AWS CloudFormation template file. When you create a stack by using this template, CloudFormation automatically creates the `OpsCenterCrossAccountManagementRole` IAM role in the account. For more information about creating a stack, see Creating a stack on the AWS CloudFormation console in the *AWS CloudFormation User Guide*.

Important

Note the following important information about this task:

- If you plan to specify an account as a delegated administrator for OpsCenter, be sure to specify that AWS account when you create the stack.
- You must perform this procedure while signed in to the AWS Organizations management account and again while signed in to the delegated administrator account.

(Optional) Set up Amazon SNS to receive notifications about OpsItems

You can configure OpsCenter to send notifications to an Amazon Simple Notification Service (Amazon SNS) topic when the system creates an OpsItem or updates an existing OpsItem.

Complete the following steps to receive notifications for OpsItems.

- Step 1: Creating and subscribing to an Amazon SNS topic
- Step 2: Updating the Amazon SNS access policy
- Step 3: Updating the AWS KMS access policy

> **Note**
>
> If you turn on AWS Key Management Service (AWS KMS) server-side encryption in Step 2, then you must complete Step 3. Otherwise, you can skip Step 3.

- Step 4: Turning on default OpsItems rules to send notifications for new OpsItems

Step 1: Creating and subscribing to an Amazon SNS topic

To receive notifications, you must create and subscribe to an Amazon SNS topic. For more information, see Creating an Amazon SNS topic and Subscribing to an Amazon SNS topic in the *Amazon Simple Notification Service Developer Guide*.

> **Note**
>
> If you're using OpsCenter in multiple AWS Regions or accounts, you must create and subscribe to an Amazon SNS topic in *each* Region or account where you want to receive OpsItem notifications.

Step 2: Updating the Amazon SNS access policy

You have to associate an Amazon SNS topic with OpsItems. Use the following procedure to set up an Amazon SNS access policy so that Systems Manager can publish OpsItems notifications to the Amazon SNS topic that you created in Step 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 that you created in Step 1, and then choose **Edit**.
4. Expand **Access policy**.
5. Add the following `Sid` block to the existing policy. Replace each *`example resource placeholder`* with your own information.

```
{
      "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", // Account ID of the
 SNS topic owner
      "Condition": {
      "StringEquals": {
        "AWS:SourceAccount": "account ID" //  Account ID of the OpsItem owner
      }
   }
}
```

> **Note**
>
> The `aws:SourceAccount` global condition key protects against the confused deputy scenario. To use this condition key, set the value to the account ID of the OpsItem owner. For more information, see Confused Deputy in the *IAM User Guide*.

6. Choose **Save changes**.

The system now sends notifications to the Amazon SNS topic when OpsItems are created or updated.

> **Important**
>
> If you configure the Amazon SNS topic with an AWS Key Management Service (AWS KMS) server-side encryption key in the Step 2, then complete Step 3. Otherwise, you can skip Step 3.

Step 3: Updating the AWS KMS access policy

If you turned on AWS KMS server-side encryption for your Amazon SNS topic, you must also update the access policy of the AWS KMS key that 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 Step 1.

> **Note**
>
> OpsCenter doesn't support publishing OpsItems to an Amazon SNS topic that is 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 that 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. Replace each *`example resource placeholder`* with your own information.

```
{
      "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"
    }
```

In the following example, the new block is entered at line 14.



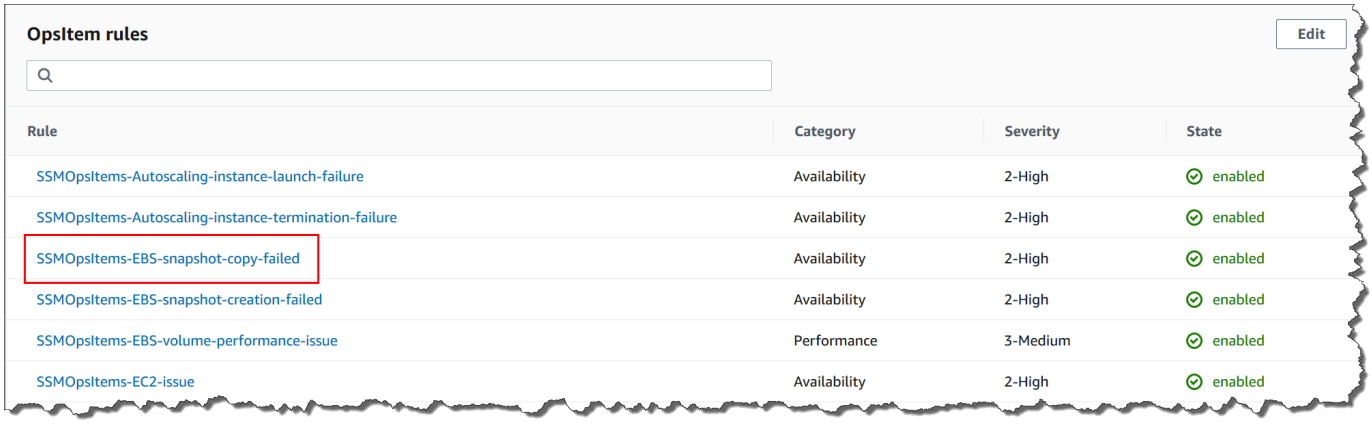
8. Choose **Save changes**.

Step 4: Turning on default OpsItems rules to send notifications for new OpsItems

Default OpsItems rules in Amazon EventBridge aren't configured with an Amazon Resource Name (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 `notifications` block, as shown in the following example.



The rule opens in Amazon EventBridge.

5. On the rule details page, on the **Targets** tab, choose **Edit**.
6. In the **Additional settings** section, choose **Configure input transformer**.
7. In the **Template** box, add 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 in the following example for the US West (Oregon) (us-west-2) Region.

```
{
    "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\" } ]"
        },
        "failure-cause": {
            "value": <failure - cause>
        },
        "source": {
            "value": <source>
        },
        "start-time": {
            "value": <start - time>
        },
        "end-time": {
            "value": <end - time>
        }
    }
}
```

8. Choose **Confirm**.
9. Choose **Next**.
10. Choose **Next**.
11. Choose **Update rule**.

The next time that the system creates an OpsItem for the default rule, it publishes a notification to the Amazon SNS topic.

Integrate OpsCenter with other AWS services

OpsCenter, a tool in AWS Systems Manager, integrates with multiple AWS services to diagnose and remediate issues with AWS resources. You must set up the AWS service before you integrate it with OpsCenter.

By default, the following AWS services are integrated with OpsCenter and can create OpsItems automatically:

- Amazon CloudWatch
- Amazon CloudWatch Application Insights

- Amazon EventBridge
- AWS Config
- AWS Systems Manager Incident Manager

You have to integrate the following services with OpsCenter to create OpsItems automatically:

- Amazon DevOps Guru
- AWS Security Hub

When any of these services create an OpsItem, you can manage and remediate the OpsItem from OpsCenter. For more information, see Manage OpsItems and Remediate OpsItem issues.

For more information about each AWS service and how it integrates with OpsCenter, see the following topics.

Topics

- Understanding OpsCenter integration with Amazon CloudWatch
- Understanding OpsCenter integration with Amazon CloudWatch Application Insights
- Understanding OpsCenter integration with Amazon DevOps Guru
- Understanding OpsCenter integration with Amazon EventBridge
- Understanding OpsCenter integration with AWS Config
- Understanding OpsCenter integration with AWS Security Hub
- Understanding OpsCenter integration with Incident Manager

Understanding OpsCenter integration with Amazon CloudWatch

Amazon CloudWatch monitors your AWS resources and services, and displays metrics on every AWS service that you use. CloudWatch creates an OpsItem when an alarm enters the alarm state. 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.

Some alarms that you can configure in CloudWatch to create OpsItems are shown in the following list:

- 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

You can either create an alarm or edit an existing alarm to create an OpsItem. For more information, see Configure CloudWatch alarms to create OpsItems.

When you enable OpsCenter using Integrated Setup, it integrates CloudWatch with OpsCenter.

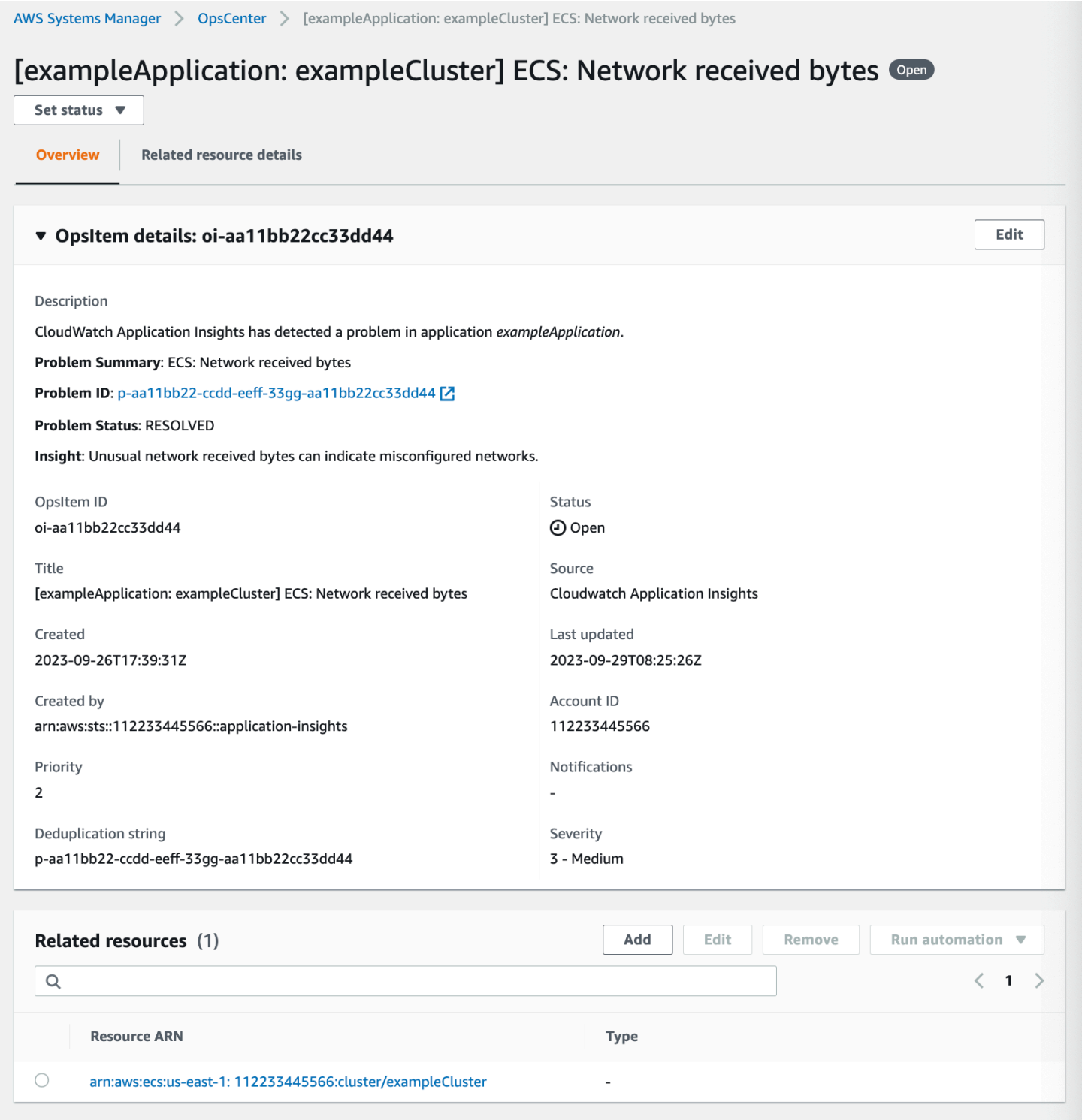
Understanding OpsCenter integration with Amazon CloudWatch Application Insights

Using Amazon CloudWatch Application Insights, you can set up the most appropriate monitors for your application resources to continuously analyze data for signs of problems with your applications. When you configure application resources in CloudWatch Application Insights, you can choose to have the system create OpsItems in OpsCenter. An OpsItem is created on the OpsCenter console for every problem detected with the application. For information, see Set up, configure, and manage your application for monitoring in the *Amazon CloudWatch User Guide*.

> **Note**
>
> Starting October 16, 2023, the title and description for OpsItems created by CloudWatch Application Insights now use the following improved format:

```
OpsItem title: [<APPLICATION NAME>: <RESOURCE ID>] <PROBLEM SUMMARY>

OpsItem description:

CloudWatch Application Insights has detected a problem in application <APPLICATION
 NAME>.
Problem summary: <PROBLEM SUMMARY>
Problem ID: <PROBLEM ID> (hyperlinks to the Application Insights problem summary page)
Problem Status: <PROBLEM STATUS>
Insight: <INSIGHT>
```

Here is an example:



Understanding OpsCenter integration with Amazon DevOps Guru

Amazon DevOps Guru applies machine learning to analyze your operational data, application metrics, and application events to identify behaviors that deviate from normal operating patterns.

If you enable DevOps Guru to generate an OpsItem in OpsCenter, each insight generates a new OpsItem. You can use OpsCenter to manage your OpsItems.

DevOps Guru automatically creates OpsItems. You can enable Amazon DevOps Guru to create OpsItems by using Quick Setup, which is a tool in Systems Manager. The system creates OpsItems by using the AWSServiceRoleForDevOpsGuru AWS Identity and Access Management (IAM) service-linked role.

To integrate OpsCenter with DevOps Guru

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Quick Setup**.
3. On the **Customize DevOps Guru configuration options** page, choose the **Library** tab.
4. In the **DevOps Guru** pane, choose **Create**.
5. For **Configuration options**, select **Enable AWS Systems Manager OpsItems.**
6. Select **Create** after you complete the setup.

Understanding OpsCenter integration with Amazon EventBridge

Amazon EventBridge delivers a stream of events that describe changes in AWS resources. When you enable OpsCenter using Integrated Setup, it integrates EventBridge with OpsCenter, and enables default EventBridge rules. Based on these rules, EventBridge creates OpsItems. Using rules, you can filter and route events to OpsCenter for investigation and remediation.

> **Note**
>
> Amazon EventBridge (formerly Amazon CloudWatch Events) provides all functionality of CloudWatch Events and some new features, such as custom event buses, third-party event sources and schema registry.

Following are some rules that you can configure in EventBridge to create an OpsItem:

- Security Hub: security alert issued
- Amazon DynamoDB a throttling event
- Amazon Elastic Compute Cloud Auto Scaling: failure to launch an instance

- Systems Manager: failure to run an automation
- AWS Health: an alert for scheduled maintenance
- Amazon EC2: instance state changed from running to stop

Based on your requirements, you can either create a rule or edit an existing rule to create an OpsItems. For instructions on how to edit a rule to create an OpsItem, see Configure EventBridge rules to create OpsItems.

Understanding OpsCenter integration with AWS Config

AWS Config provides a detailed view of the configuration of AWS resources in your AWS account.

AWS Config does not integrate *directly* with OpsCenter. Instead, you create an AWS Config rule that sends an event to Amazon EventBridge, such as when AWS Config detects a noncompliant instance. Then EventBridge evaluates that event against an EventBridge rule you've created. If the rule matches, EventBridge transforms the event to an OpsItem and transmits it to OpsCenter as the destination target.

Using this OpsItem, you can track details of the noncompliant resource, record investigative actions, and provide access to consistent remediation actions.

Related info

Configure EventBridge rules to create OpsItems

Using AWS Systems Manager OpsCenter and AWS Config for compliance monitoring

Understanding OpsCenter integration with AWS Security Hub

AWS Security Hub collects security data, called *findings*, from across AWS accounts and services. Using a set of rules to detect and generate findings, Security Hub helps you identify, prioritize, and remediate security issues for the resources you manage. After you configure integration, as described in this topic, Systems Manager creates OpsItems for Security Hub findings in OpsCenter.

> **Note**
>
> OpsCenter has bidirectional integration with Security Hub. This means that if you update the **Status** or **Severity** field for an OpsItem related to a security finding, the system synchronizes the changes with Security Hub. Likewise, any changes to a finding are automatically updated in the corresponding OpsItems in OpsCenter.

When an OpsItem is created from a Security Hub finding, Security Hub metadata is automatically added to the operational data field of the OpsItem. If this metadata is deleted, the bidirectional updates no longer function.

By default, Systems Manager creates OpsItems for critical and high severity findings. You can manually configure OpsCenter to create OpsItems for medium and low severity findings. OpsCenter doesn't create OpsItems for informational findings as they don't require remediation. For more information about Security Hub severity levels, see Severity in the *AWS Security Hub API Reference*.

Before you begin

Before you configure OpsCenter to create OpsItems based on Security Hub findings, verify that you completed the Security Hub set up tasks. For more information, see Setting up Security Hub in the *AWS Security Hub User Guide*.

When you integrate Security Hub with OpsCenter, the system creates OpsItems by using the `AWSServiceRoleForSystemsManagerOpsDataSync` IAM service-linked role. For more information about this role, see Using roles to create OpsData and OpsItems for Explorer.

Warning

Note the following important information about pricing for OpsCenter integration with Security Hub:

- If you are logged into the Security Hub administrator account when you configure OpsCenter and Security Hub integration, the system creates OpsItems for findings in the administrator *and* all member accounts. The OpsItems are all created *in the administrator account*. Depending on a variety of factors, this can lead to an unexpectedly large bill from AWS.

  If you are logged into a member account when you configure integration, the system only creates OpsItems for findings in that individual account. For more information about the Security Hub administrator account, member accounts, and their relation to the EventBridge event feed for findings, see Types of Security Hub integration with EventBridge in the *AWS Security Hub User Guide*.

- For each finding that creates an OpsItem, you are charged the regular price for creating the OpsItem. You are also charged if you edit the OpsItem or if the corresponding finding is updated in Security Hub (which triggers an OpsItem update).
- OpsItems that are created by an integration with AWS Security Hub are *not* currently limited by the maximum quota of 500,000 OpsItems per account in a Region. It is therefore possible for Security Hub alerts to create more than 500,000 chargeable OpsItems in each Region in an account.

  For high-production environments, we therefore recommend limiting the scope of Security Hub findings to high severity issues only.

To configure OpsCenter to create OpsItems for 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. Choose **Settings**.
4. In the **Security Hub findings** section, choose **Edit.**
5. Choose the slider to change **Disabled** to **Enabled**.
6. If you want the system to create OpsItems for medium or low severity findings, toggle these options.
7. Choose **Save** to save your configuration.

Use the following procedure if you no longer want the system to create OpsItems for Security Hub findings.

To stop receiving OpsItems for 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. Choose **Settings**.
4. In the **Security Hub findings** section, choose **Edit.**

5. Choose the slider to change **Enabled** to **Disabled**. If you aren't able to toggle the slider, Security Hub hasn't been enabled for your AWS account.
6. Choose **Save** to save your configuration. OpsCenter no longer creates OpsItems based on Security Hub findings.

> **Important**
>
> A Systems Manager delegated administrator or the AWS Organizations management account can enable Security Hub findings in OpsCenter for multiple accounts and AWS Regions by creating a resource data sync in Explorer. If the **Security Hub** source is enabled in Explorer and a resource data sync exists that targets the member account where you disabled Security Hub integration, then the settings selected by your administrator take precedence. OpsCenter continues to create OpsItems for Security Hub findings. To stop creating OpsItems for Security Hub findings in a member account targeted by a resource data sync, contact your administrator and ask them to remove your account from the resource data sync or turn off the **Security Hub** source in Explorer. For information about changing settings in Explorer, see Editing Systems Manager Explorer data sources.

Understanding OpsCenter integration with Incident Manager

Incident Manager, a tool in AWS Systems Manager, provides an incident management console that helps you mitigate and recover from incidents affecting your AWS hosted applications. An *incident* 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 the system creates an incident in Incident Manager, it also creates an OpsItem in OpsCenter, and displays the incident as a related item. If the OpsItem already exists, Incident Manager doesn't create an OpsItem. The first OpsItem is known as the parent OpsItem. If an incident grows in scale and scope, you can add incidents to an existing OpsItem. If required, you can manually create an incident for an OpsItem. After an incident is closed, you can create an analysis in Incident Manager to review and improve the remediation process for similar issues.

By default, OpsCenter integrates with Incident Manager. If Incident Manager is not set up, the OpsCenter page displays a message to set up Incident Manager. When Incident Manager creates an OpsItem, you can manage and remediate the OpsItem from OpsCenter. For instructions on creating an incident for an OpsItem, see Creating an incident for an OpsItem.

Create OpsItems

After you set up OpsCenter, a tool in AWS Systems Manager, and integrate it with your AWS services, your AWS services automatically create OpsItems based on default rules, events, or alarms.

You can view the statuses and severity levels of default Amazon EventBridge rules. If required, you can create or edit these rules from Amazon EventBridge. You can also view alarms from Amazon CloudWatch, and create or edit alarms. Using rules and alarms, you can configure events for which you want to generate OpsItems automatically.

When the system creates an OpsItem, it's in the **Open** status. You can change the status to **In progress** when you start investigation of the OpsItem and to **Resolved** after you remediate the OpsItem. For more information about how to configure alarms and rules in AWS services to create OpsItems and how to create OpsItems manually, see the following topics.

Topics

- Configure EventBridge rules to create OpsItems
- Configure CloudWatch alarms to create OpsItems
- Create OpsItems manually

Configure EventBridge rules to create OpsItems

When Amazon EventBridge receives an event, it creates a new OpsItem based on default rules. You can create a rule or edit an existing rule to set OpsCenter as the target of an EventBridge event. For information about how to create an event rule, see Creating a rule for an AWS service in the *Amazon EventBridge User Guide*.

To configure an EventBridge rule to create OpsItems in OpsCenter

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, for **Event bus**, choose **default**.
4. For **Rules**, choose a rule by selecting the check box next to its name.
5. Select the name of the rule to open its details page. In **Rule details**, verify that **Status** is set to **Enabled**.

> **Note**
>
> If required, you can update the status using **Edit** in the upper-right corner of the page.

6. Choose the **Targets** tab.
7. On the **Targets** tab, choose **Edit**.
8. For **Target types**, select **AWS service**.
9. For **Select a target**, choose **Systems Manager OpsItem**.
10. For many target types, EventBridge needs permission 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 to give EventBridge permission to create OpsItems in OpsCenter, choose **Use existing role**.
11. In **Additional settings**, for **Configure target input**, choose **Input Transformer**.

    You can use the **Input transformer** option to specify a deduplication string and other important information for OpsItems, such as title and severity.
12. Choose **Configure input transformer**.
13. In **Target input transformer**, for **Input path**, specify the values to parse from the triggering event. For example, to parse the start time, end time, and other details from the event that triggers the rule, use the following JSON.

    ```
    {
        "end-time": "$.detail.EndTime",
        "failure-cause": "$.detail.cause",
        "resources": "$.resources[0]",
        "source": "$.detail.source",
        "start-time": "$.detail.StartTime"
    }
    ```
14. For **Template**, specify the information to send to the target. For example, use the following JSON to pass information to OpsCenter. The information is used to create an OpsItem.

Note

If the input template is in the JSON format, then the object value in the template can't include quotes. For example, the values for resources, failure-cause, source, start time, and end time can't be in quotes.

```
{
    "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",
    "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>
        },
         },
        "resources": {
            "value": <resources>
        }
    }
}
```

For more information about these fields, see Transforming target input in the *Amazon EventBridge User Guide*.

15. Choose **Confirm**.
16. Choose **Next**.
17. Choose **Next**.
18. Choose **Update 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 Manage OpsItems.

Configure CloudWatch alarms to create OpsItems

During the integrated setup of OpsCenter, a tool in AWS Systems Manager, you enable Amazon CloudWatch to automatically create OpsItems based on common alarms. You can create an alarm or edit an existing alarm to create OpsItems in OpsCenter.

CloudWatch 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*.

When a CloudWatch alarm generates an OpsItem, the OpsItem displays **CloudWatch alarm - '*alarm_name*' is in ALARM state**.

To view details about a specific OpsItem, choose the OpsItem and then choose the **Related resource details** tab. You can manually edit OpsItems to change details, such as the severity or category. However, when you edit the severity or the category of an alarm, Systems Manager can't update the severity or category of OpsItems that are already created from the alarm. If an alarm created 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.

For more information about configuring CloudWatch alarms, see the following topics.

Topics

- Configuring a CloudWatch alarm to create OpsItems (console)
- Configuring an existing CloudWatch alarm to create OpsItems (programmatically)

Configuring a CloudWatch alarm to create OpsItems (console)

You can manually create an alarm or update an existing alarm to create OpsItems from Amazon CloudWatch.

To create a CloudWatch alarm and configure Systems Manager as a target of that alarm

1. Complete steps 1–9 as specified in Create a CloudWatch alarm based on a static threshold in the *Amazon CloudWatch User Guide*.
2. In the **Systems Manager action** section, choose **Add Systems Manager OpsCenter action**.
3. Choose **OpsItems**.
4. For **Severity**, choose from 1 to 4.
5. (Optional) For **Category**, choose a category for the OpsItem.
6. Complete steps 11–13 as specified in Create a CloudWatch alarm based on a static threshold in the *Amazon CloudWatch User Guide*.
7. Choose **Next** and complete the wizard.

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.

Configuring an existing CloudWatch alarm to create OpsItems (programmatically)

You can configure Amazon CloudWatch alarms to create OpsItems programmatically by using the AWS Command Line Interface (AWS CLI), AWS CloudFormation templates, or Java code snippets.

Topics

- Before you begin
- Configuring CloudWatch alarms to create OpsItems (AWS CLI)
- Configuring CloudWatch alarms to create or update OpsItems (CloudFormation)
- Configuring CloudWatch alarms to create or update OpsItems (Java)

Before you begin

If you edit an existing alarm programmatically or create an 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 AWS Systems Manager endpoints and quotas.
`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

Parameter	Details
`Category`(optional)	A category for OpsItems created from the alarm. Valid values: `Availability` , `Cost`, `Performance` , `Recovery`, and `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
```

Configuring CloudWatch alarms to create OpsItems (AWS CLI)

This command requires that you specify an ARN for the `alarm-actions` parameter. For information about how to create the ARN, see Before you begin.

To configure a CloudWatch alarm to create OpsItems (AWS CLI)

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

   For information, see Installing or updating the latest version of the AWS CLI.
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. Replace each *example resource placeholder* with your own information.

```
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
```

Configuring CloudWatch alarms to create or update OpsItems (CloudFormation)

This section includes AWS CloudFormation templates that you can use to configure CloudWatch alarms to automatically create or update OpsItems. Each template requires that you specify an ARN for the `AlarmActions` parameter. For information about how to create the ARN, see Before you begin.

Metric alarm – Use the following CloudFormation template to create or update a CloudWatch metric alarm. The alarm specified in this template monitors Amazon Elastic Compute Cloud (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::Instance::Id"
       }
     },
     "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": "GreaterThanThreshold",
           "Threshold": "0",
           "AlarmActions": [ {"Fn::Join" : ["",
["arn: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 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":"CPUusageishigh",
            "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":"Memoryusageishigh",
            "AlarmName":"HighMemoryUsage",
            "ComparisonOperator":"GreaterThanThreshold",
            "EvaluationPeriods":1,
            "MetricName":"MemoryUsage",
            "Namespace":"CustomNamespace",
            "Period":60,
            "Statistic":"Average",
            "Threshold":65,
            "TreatMissingData":"breaching"
         }
      }
   }
```

Configuring CloudWatch alarms to create or update OpsItems (Java)

This section includes Java code snippets that you can use to configure CloudWatch alarms to automatically create or update OpsItems. Each snippet requires that you specify an ARN for the `validSsmActionStr` parameter. For information about how to create the ARN, see Before you begin.

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;
            }
        }
    }
```

Create OpsItems manually

When you find an operational issue, you can manually create an OpsItem from OpsCenter, a tool in AWS Systems Manager, to manage and resolve the issue.

If you set up OpsCenter for cross-account administration, a Systems Manager delegated administrator or AWS Organizations management account can create OpsItems for member accounts. For more information, see (Optional) Manually set up OpsCenter to centrally manage OpsItems across accounts.

You can create OpsItems by using the AWS Systems Manager console, the AWS Command Line Interface (AWS CLI), or AWS Tools for Windows PowerShell.

Topics

- Creating OpsItems manually (console)
- Creating OpsItems manually (AWS CLI)
- Creating OpsItems manually (PowerShell)

Creating OpsItems manually (console)

You can manually create OpsItems using the AWS Systems Manager console. When you create an OpsItem, it's displayed in your OpsCenter account. If you set up OpsCenter for cross-account administration, OpsCenter provides the delegated administrator or management account with the option to create OpsItems for selected member accounts. For more information, see (Optional) Manually set up OpsCenter to centrally manage OpsItems across accounts.

To create an OpsItem 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 **OpsCenter**.
3. Choose **Create OpsItem**. If you don't see this button, choose the **OpsItems** tab, and then choose **Create OpsItem**.
4. (Optional) Choose **Other account**, and then choose the account where you want to create the OpsItem.

   > **Note**
   >
   > This step is required if you're creating OpsItems for a member account.

5. For **Title**, enter a descriptive name to help you understand the purpose of the OpsItem.
6. 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.

7. (Optional) For **Priority**, choose the priority level.
8. (Optional) For **Severity**, choose the severity level.
9. (Optional) For **Category**, choose a category.
10. For **Description**, enter information about this OpsItem including (if applicable) steps for reproducing the issue.

> **Note**
>
> The console supports most markdown formatting in the OpsItem description field. For more information, see Using Markdown in the Console in the *Getting Started with the AWS Management Console Getting Started Guide.*

11. For **Deduplication string**, enter words that the system can use to check for duplicate OpsItems. For more information about deduplication strings, see Managing duplicate OpsItems.
12. (Optional) For **Notifications**, specify the Amazon Resource Name (ARN) of the Amazon SNS topic 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.
13. (Optional) For **Related resources**, choose **Add** to specify the ID or ARN of the impacted resource and any related resources.
14. Choose **Create OpsItem**.

If successful, the page displays the OpsItem. When a delegated administrator or management account creates an OpsItem for selected member accounts, the new OpsItems are displayed in the OpsCenter of the administrator and members accounts. For information about how to configure the options in an OpsItem, see Manage OpsItems.

Creating OpsItems manually (AWS CLI)

The following procedure describes how to create an OpsItem by using the AWS Command Line Interface (AWS CLI).

To create an OpsItem using the AWS CLI

1. Install and configure the AWS Command Line Interface (AWS CLI), if you haven't already.

   For information, see Installing or updating the latest version of the AWS CLI.
2. Open the AWS CLI and run the following command to create an OpsItem. Replace each *example resource placeholder* with your own information.

```
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"
```

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"
```

```
}
```

3. Run the following command to view details about the OpsItem that 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"
            }
        }
    }
}
```

4. 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.

5. Run the following command again to verify that the status changed to `InProgress`.

```
aws ssm get-ops-item --ops-item-id ID
```

Examples of creating an OpsItem

The following code examples show you how to create an OpsItem by using the Linux management portal, macOS, or Windows Server.

Linux management portal or macOS

The following command creates an OpsItem when an Amazon Elastic Compute Cloud (Amazon EC2) instance disk is full.

```
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

The following command creates an OpsItem when an Amazon Relational Database Service (Amazon RDS) instance is not responding.

```
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 Amazon EC2 instance related resource.

```
aws ssm create-ops-item ^
    --title "EC2 instance disk full" ^
    --description "Log clean up may have failed which caused the disk to be full" ^
    --priority 2 ^
    --source ec2 ^
    --operational-data={\"/aws/resources\":{\"Value\":\"[{\\\"""arn\\""":\
\"""arn:aws:ec2:us-east-1:123456789012:instance/i-1234567890abcdef0\\"""}]\",\"Type\":
\"SearchableString\"}}
```

The following command uses the `/aws/automations` key in `OperationalData` to create an OpsItem that specifies the `AWS-RestartEC2Instance` runbook 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\"}}
```

Creating OpsItems manually (PowerShell)

The following procedure describes how to create an OpsItem by using AWS Tools for Windows PowerShell.

To create an OpsItem 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 AWS Region for your PowerShell session.

   ```
   Set-DefaultAWSRegion -Region Region
   ```

3. Run the following command to create a new OpsItem. Replace each *example resource placeholder* with your own information. 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-SSMOpsItem `
       -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 Amazon Resource Name (ARN) of an impaired Amazon Elastic Compute Cloud (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-SSMOpsItem -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
```

Manage OpsItems

OpsCenter, a tool in AWS Systems Manager, tracks OpsItems from their creation to resolution. If you set up OpsCenter for cross-account administration, a delegated administrator or management account can manage OpsItems from their account. For more information, see (Optional) Manually set up OpsCenter to centrally manage OpsItems across accounts.

You can view and manage OpsItems by using the following pages in the Systems Manager console:

- **Summary** – Displays a count of open and in-progress OpsItems, count of OpsItems by source and age, and operational insights. You can filter OpsItems by source and OpsItems status.

- **OpsItems** – Displays a list of OpsItems with multiple fields of information, such as title, ID, priority, description, the source of the OpsItem, and the date and time of last update. Using this page, you can manually create OpsItems, configure sources, change the status of an OpsItem, and filter OpsItems by new incidents. You can choose an OpsItem to display its **OpsItems details** page.

- **OpsItem details** – Provides detailed insights and tools that you can use to manage an OpsItem. The OpsItems details page has the following tabs:
  - **Overview** – Displays related resources, runbooks that ran in the last 30 days, and a list of available runbooks that you can run. You can also view similar OpsItems, add operational data, and add related OpsItems.
  - **Related resource details** – 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.

For more information about how to manage OpsItems, see the following topics.

Topics

- Viewing details of an OpsItem
- Editing an OpsItem
- Adding related resources to an OpsItem
- Adding related OpsItems to an OpsItem
- Adding operational data to an OpsItem
- Creating an incident for an OpsItem
- Managing duplicate OpsItems
- Analyzing operational insights to reduce OpsItems
- Viewing OpsCenter logs and reports

Viewing details of an OpsItem

To get a comprehensive view of an OpsItem, use the **OpsItem details** page in the OpsCenter console. The **Overview** page displays the following information:

- **OpsItems details**– Displays general information for the selected OpsItem.
- **Related Resources** – A related resource is the impacted resource or the resource that initiated the event that created the OpsItem.
- **Automation executions in the last 30 days** – A list of runbooks that ran in last 30 days.
- **Runbooks** – You can choose a runbook from a list of available runbooks.
- **Similar OpsItems** – This 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.
- **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.
- **Related OpsItems** – You can specify the IDs of OpsItems that are in some way related to the current OpsItem.

- **Related Resource Details** – Displays data providers, including Amazon CloudWatch metrics and alarms, AWS CloudTrail logs, and details from AWS Config.

To view details of 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 to view its details.

Editing an OpsItem

The **OpsItem details** section includes information about an OpsItem, including the description, title, source, OpsItem ID, and the status. You can edit a single OpsItem or you can select multiple OpsItems and edit the following fields: **Status**, **Priority**, **Severity**, **Category**.

When Amazon EventBridge creates an OpsItem, it populates the **Title**, **Source**, and **Description** fields. You can edit the **Title** and **Description** fields, but you can't edit the **Source** field.

> **Note**
>
> The console supports most markdown formatting in the OpsItem description field. For more information, see Using Markdown in the Console in the *Getting Started with the AWS Management Console Getting Started Guide.*

Generally, you can edit the following configurable data for an OpsItem:

- **Title** – Name of the OpsItem. The source creates the title of the OpsItem.
- **Description** – Information about this OpsItem including (if applicable) steps for reproducing the issue.
- **Status** – Status of an OpsItem. For a list of valid status values, see OpsItem Status in the *AWS Systems Manager API Reference*.

- **Priority** – Priority of an OpsItem can be between 1 and 5. We recommend that your organization determine what each priority level means and a corresponding service level agreement for each level.
- **Severity** – Severity of an OpsItem can be between 1 to 4, where 1 is critical, 2 is high, 3 is medium, and 4 is low.
- **Category** – Category of an OpsItem can be availability, cost, performance, recovery, or security.
- **Notifications** – When you edit an OpsItem, you can specify the Amazon Resource Name (ARN) of an Amazon Simple Notification Service 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. 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 the topic and the OpsItem are in different Regions, the system returns an error.

OpsCenter has bidirectional integration with AWS Security Hub. When you update an OpsItem status and severity related to a security finding, those changes are automatically sent to Security Hub to ensure you always see the latest and correct information.

When an OpsItem is created from a Security Hub finding, Security Hub metadata is automatically added to the operational data field of the OpsItem. If this metadata is deleted, the bidirectional updates no longer function.

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**.

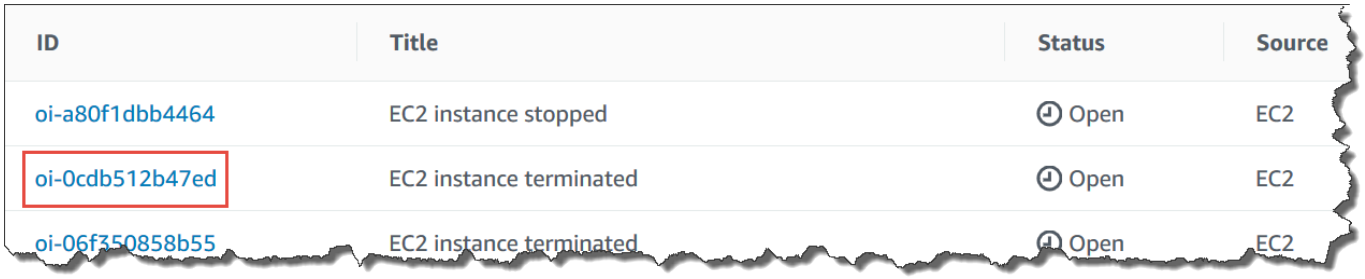
Adding related resources to an OpsItem

Each OpsItem includes a **Related resources** section that lists the Amazon Resource Name (ARN) of the related resource. A *related resource* is the impacted AWS resource that needs to be investigated.

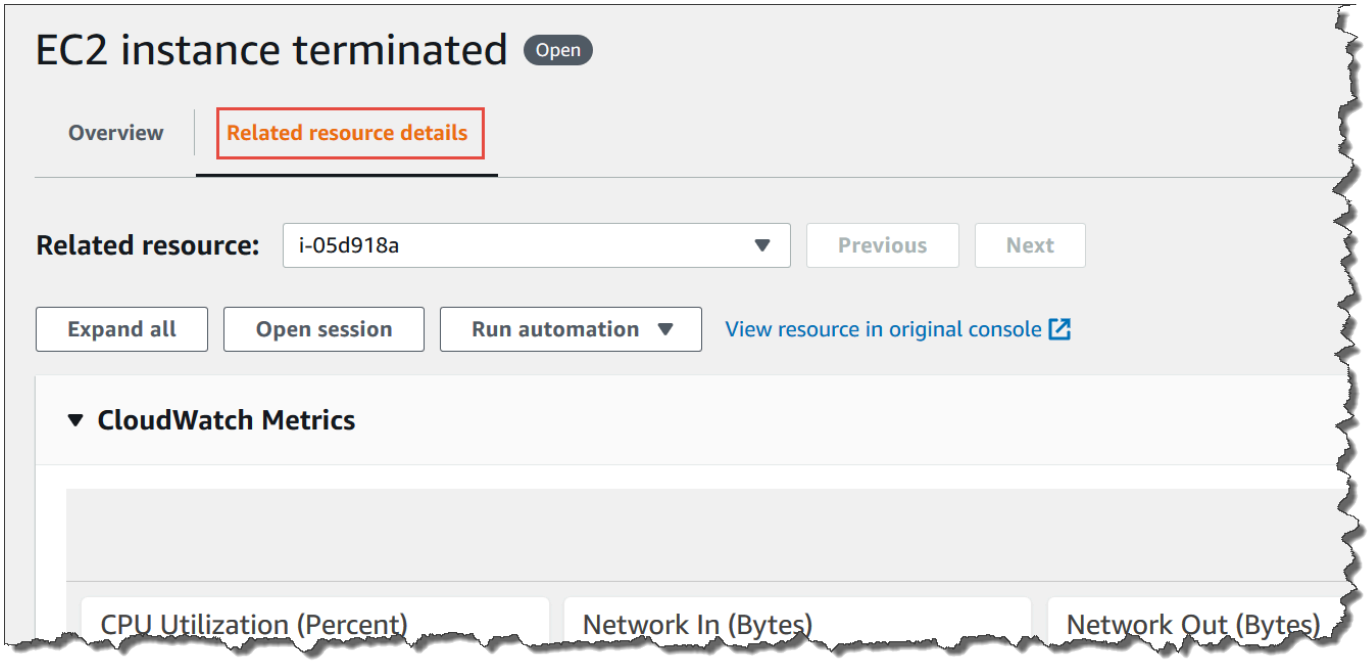
If Amazon EventBridge creates the OpsItem, the system automatically populates the OpsItem with the ARN of the resource. You can manually specify ARNs of related resources. For certain ARN types, OpsCenter automatically creates a deep link that displays details about the resource directly in the OpsCenter console. For example, if you specify the ARN of an Amazon Elastic Compute Cloud (Amazon EC2) instance as a related resource, then OpsCenter pulls in details about that EC2 instance. This allows you to view detailed information about your impacted AWS resources without having to leave OpsCenter.

To view and add related resources 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 the **OpsItems** tab.
4. Choose an OpsItem ID.



5. To view information about the impacted resource, choose the **Related resources details** tab.



This tab displays information about the resource from several AWS services. Expand the **Resource details** section to view information about this resource as provided by the AWS service that hosts it. You can also toggle through other related resources associated with this OpsItem by using the **Related resources** list.

6. To add additional related resources, choose the **Overview** tab.
7. In the **Related resources** section, choose **Add**.
8. For **Resource 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 that you chose in the previous step.

> **Note**
>
> You can manually add the ARNs of additional related resources. Each OpsItem can list a maximum of 100 related resource ARNs.

The following table lists the resource types that automatically create deep links to related resources.

Supported resource types

Resource name	ARN format
AWS Certificate Manager certificate	`arn:aws:acm: region:account-id :certificate/ certificate-id`
Amazon EC2 Auto Scaling group	`arn:aws:autoscaling: region:account-id :autoScalingGroup: groupid:autoScalingGroupName/ groupfriendlyname`
Amazon CloudFront distribution	`arn:aws:cloudfront:: account-id :*`
AWS CloudFormation stack	`arn:aws:cloudformation: region:account-id :stack/stackname /additionalidentifier`
Amazon CloudWatch alarm	`arn:aws:cloudwatch: region:account-id :alarm:alarm-name`
AWS CloudTrail trail	`arn:aws:cloudtrail: region:account-id :trail/trailname`
AWS CodeBuild project	`arn:aws:codebuild: region:account-id :resourcetype /resource`
AWS CodePipeline	`arn:aws:codepipeline: region:account-id :resource-specifier`
Amazon DevOps Guru insight	`arn:aws:devops-guru: region:account-id :insight/ proactive or reactive/resource-id`

Resource name	ARN format
Amazon DynamoDB table	`arn:aws:dynamodb: region:account-id :table/tablename`
Amazon Elastic Compute Cloud (Amazon EC2) customer gateway	`arn:aws:ec2: region:account-id :customer-gateway/ cgw-id`
Amazon EC2 elastic IP	`arn:aws:ec2: region:account-id :eip/eipalloc-id`
Amazon EC2 Dedicated Host	`arn:aws:ec2: region:account-id :dedicated-host/ host-id`
Amazon EC2 instance	`arn:aws:ec2: region:account-id :instance/ instance-id`
Amazon EC2 internet gateway	`arn:aws:ec2: region:account-id :internet-gateway/ igw-id`
Amazon EC2 network access control list (network ACL)	`arn:aws:ec2: region:account-id :network-acl/ nacl-id`
Amazon EC2 network interface	`arn:aws:ec2: region:account-id :network-interface/ eni-id`
Amazon EC2 route table	`arn:aws:ec2: region:account-id :route-table/ route-table-id`
Amazon EC2 security group	`arn:aws:ec2: region:account-id :security-group/ security-group-id`

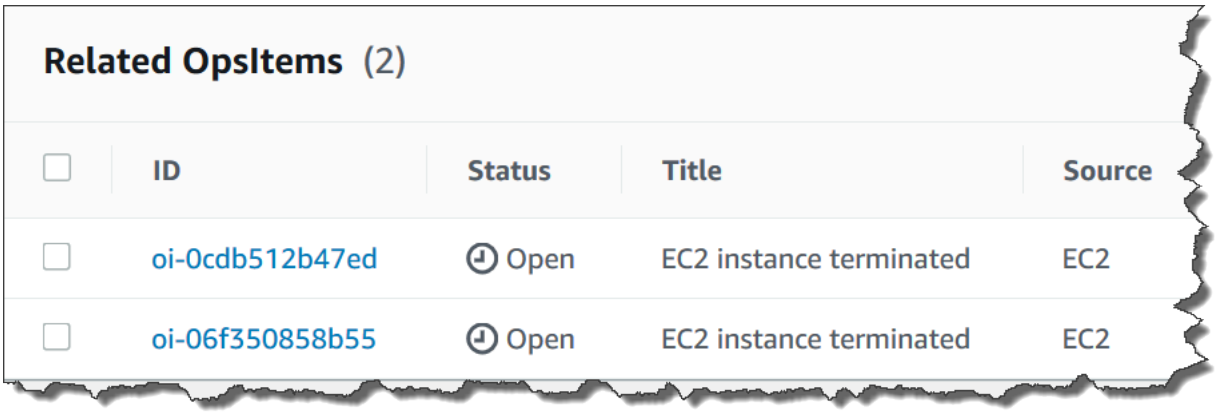
Resource name	ARN format
Amazon EC2 subnet	`arn:aws:ec2: region:account-id :subnet/subnet-id`
Amazon EC2 volume	`arn:aws:ec2: region:account-id :volume/volume-id`
Amazon EC2 VPC	`arn:aws:ec2: region:account-id :vpc/vpc-id`
Amazon EC2 VPN connection	`arn:aws:ec2: region:account-id :vpn-connection/ vpn-id`
Amazon EC2 VPN gateway	`arn:aws:ec2: region:account-id :vpn-gateway/ vgw-id`
AWS Elastic Beanstalk application	`arn:aws:elasticbeanstalk: region:account-id :application/ applicationname`
Elastic Load Balancing (Classic Load Balancer)	`arn:aws:elasticloadbalancing: region:account-id :loadbalancer/ name`
Elastic Load Balancing (Application Load Balancer)	`arn:aws:elasticloadbalancing: region:account-id :loadbalancer/app/ load-balancer-name /load-balancer-id`
Elastic Load Balancing (Network Load Balancer)	`arn:aws:elasticloadbalancing: region:account-id :loadbalancer/net/ load-balancer-name /load-balancer-id`

Resource name	ARN format
AWS Identity and Access Management (IAM) group	`arn:aws:iam:: account-id :group/group-name`
IAM policy	`arn:aws:iam:: account-id :policy/policy-name`
IAM role	`arn:aws:iam:: account-id :role/role-name`
IAM user	`arn:aws:iam:: account-id :user/user-name`
AWS Lambda function	`arn:aws:lambda: region:account-id :function: function-name`
Amazon Relational Database Service (Amazon RDS) cluster	`arn:aws:rds: region:account-id :cluster: db-cluster-name`
Amazon RDS database instance	`arn:aws:rds: region:account-id :db:db-instance-name`
Amazon RDS subscription	`arn:aws:rds: region:account-id :es:subscription-name`
Amazon RDS security group	`arn:aws:rds: region:account-id :secgrp:security-group-name`
Amazon RDS cluster snapshot	`arn:aws:rds: region:account-id :cluster-snapshot:  cluster-snapshot-name`

Resource name	ARN format
Amazon RDS subnet group	`arn:aws:rds:region:account-id:subgrp:subnet-group-name`
Amazon Redshift cluster	`arn:aws:redshift:region:account-id:cluster:cluster-name`
Amazon Redshift parameter group	`arn:aws:redshift:region:account-id:parametergroup:parameter-group-name`
Amazon Redshift security group	`arn:aws:redshift:region:account-id:securitygroup:security-group-name`
Amazon Redshift cluster snapshot	`arn:aws:redshift:region:account-id:snapshot:cluster-name/snapshot-name`
Amazon Redshift subnet group	`arn:aws:redshift:region:account-id:subnetgroup:subnet-group-name`
Amazon Simple Storage Service (Amazon S3) bucket	`arn:aws:s3:::bucket_name`
AWS Config recording of AWS Systems Manager managed node inventory	`arn:aws:ssm:region:account-id:managed-instance-inventory/node_id`
Systems Manager State Manager association	`arn:aws:ssm:region:account-id:association/association_ID`

Adding related OpsItems to an OpsItem

By using **Related OpsItems** of the **OpsItems Details** page, you can investigate operations issues and provide context for an issue. OpsItems can be related in different ways, including a parent-child relationship between OpsItems, a root cause, or a duplicate. You can associate one OpsItem with another to display it in the **Related OpsItem** section.You can specify a maximum of 10 IDs for other OpsItems that are related to the current OpsItem.

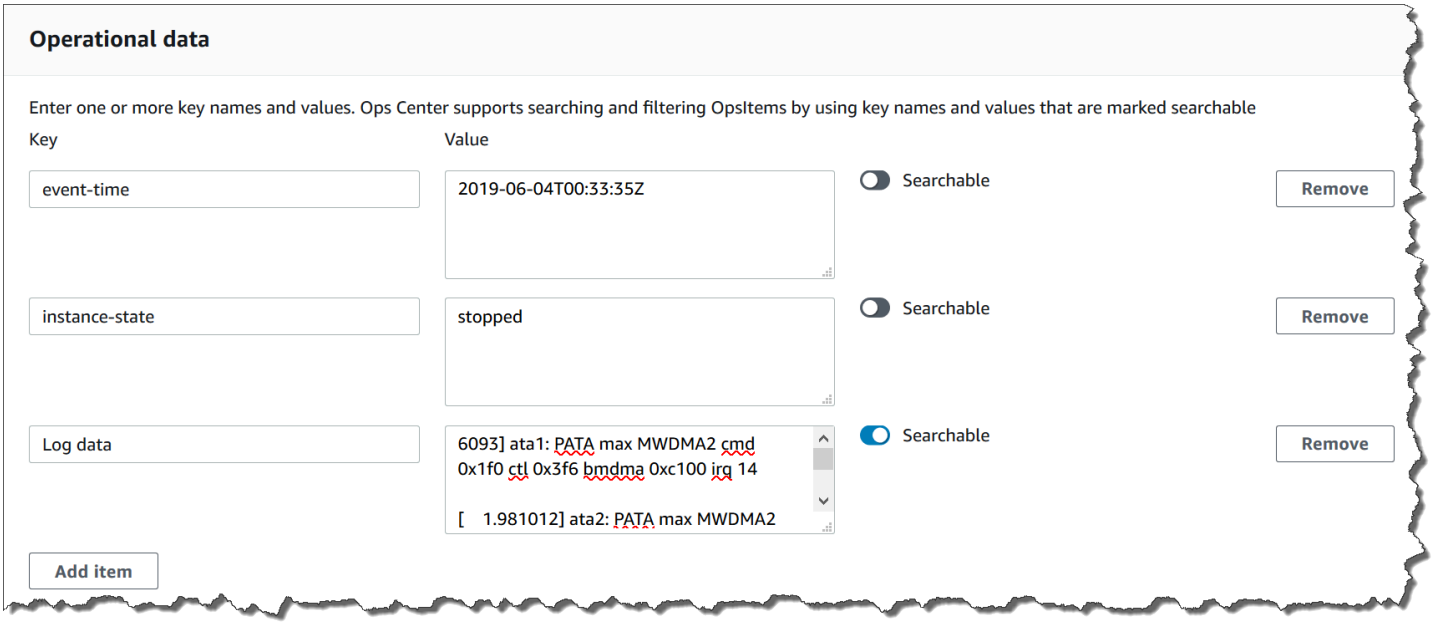


To add a related OpsItem

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**.

Adding operational data to an OpsItem

Operational data is custom data that provides useful reference details about an OpsItem. You can enter multiple key-value pairs of operational data. For example, you can specify log files, error strings, license keys, troubleshooting tips, or other relevant data. The maximum length of the key can be 128 characters, and the maximum size of the value can be 20 KB.



You can 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 its details page.
4. Expand **Operational data**.
5. If no operational data exists for the OpsItem, choose **Add**. If operational data already exists for the OpsItem, choose **Manage**.

   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**.
6. For **Key**, specify a word or words to help users understand the purpose of the data.

> **Important**
>
> Operational data keys *can't* begin with the following: `amazon`, `aws`, `amzn`, `ssm`, `/amazon`, `/aws`, `/amzn`, `/ssm`.

7. For **Value**, specify the data.
8. Choose **Save**.

> **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"}`

Creating an incident for an OpsItem

Use the following procedure to manually create an incident for an OpsItem to track and manage it in AWS Systems Manager Incident Manager, which is a tool in AWS Systems Manager. An *incident* is any unplanned interruption or reduction in quality of services. For more information about Incident Manager, see the section called “Integrate OpsCenter with other AWS services”.

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 an OpsItem, open it.
5. Choose **Start Incident**.
6. For **Response plan**, choose the Incident Manager response plan that 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**.

Managing duplicate OpsItems

OpsCenter can receive multiple duplicate OpsItems for a single source from multiple AWS services. OpsCenter uses a combination of built-in logic and configurable deduplication strings to avoid creating duplicate OpsItems. AWS Systems Manager applies deduplication built-in logic when the CreateOpsItem API operation is called.

AWS Systems Manager uses the following deduplication logic:

1. When creating the OpsItem, Systems Manager creates and stores a hash based on the deduplication string and the resource that initiated the OpsItem.
2. When another request is made to create an OpsItem, the system checks the deduplication string of the new request.
3. If a matching hash exists for this deduplication string, Systems Manager checks the status of the existing OpsItem. If the status of an existing OpsItem is open or in progress, the OpsItem is not created. If the existing OpsItem is resolved, Systems Manager creates a new OpsItem.

After you create an OpsItem, you *can't* edit or change the deduplication strings in that OpsItem.

To manage duplicate OpsItems, you can do the following:

- Edit the deduplication string for an Amazon EventBridge rule that targets OpsCenter. For more information, see Editing a deduplication string in a default EventBridge rule.
- Specify a deduplication string when you manually create an OpsItem. For more information, see Specifying a deduplication string using AWS CLI.
- Review and resolve duplicate OpsItems using operational insights. You can use runbooks to resolve duplicate OpsItems.

To help you resolve duplicate OpsItems and reduce the number of OpsItems created by a source, Systems Manager provides automation runbooks. For information, see Resolving duplicate OpsItems based on insights.

Editing a deduplication string in a default EventBridge rule

Use the following procedure to specify a deduplication string for an EventBridge rule that targets OpsCenter.

To edit a deduplication string for an 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. Go to the **Select target(s)** page.
5. In the **Additional settings** section, choose **Configure input transformer**.
6. In the **Template** box, 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\"}"}}
   ```
7. Edit the deduplication strings, and then choose **Confirm**.
8. Choose **Next**.
9. Choose **Next**.
10. Choose **Update rule**.

Specifying a deduplication string using AWS CLI

You can specify a deduplication string when you manually create a new OpsItem by using either the AWS Systems Manager console or the AWS CLI. For information about entering deduplication strings when you manually create an OpsItem in the console, see Create OpsItems manually.
If you're using the AWS CLI, you can enter the deduplication string for the `OperationalData` parameter. The parameter syntax uses JSON, as shown in the following example.

```
--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"
```

Analyzing operational insights to reduce OpsItems

OpsCenter *operational insights* display information about duplicate OpsItems. OpsCenter automatically analyzes OpsItems in your account and generates three types of *insights*. You can view this information in the **Operational insights** section of the OpsCenter **Summary** tab.

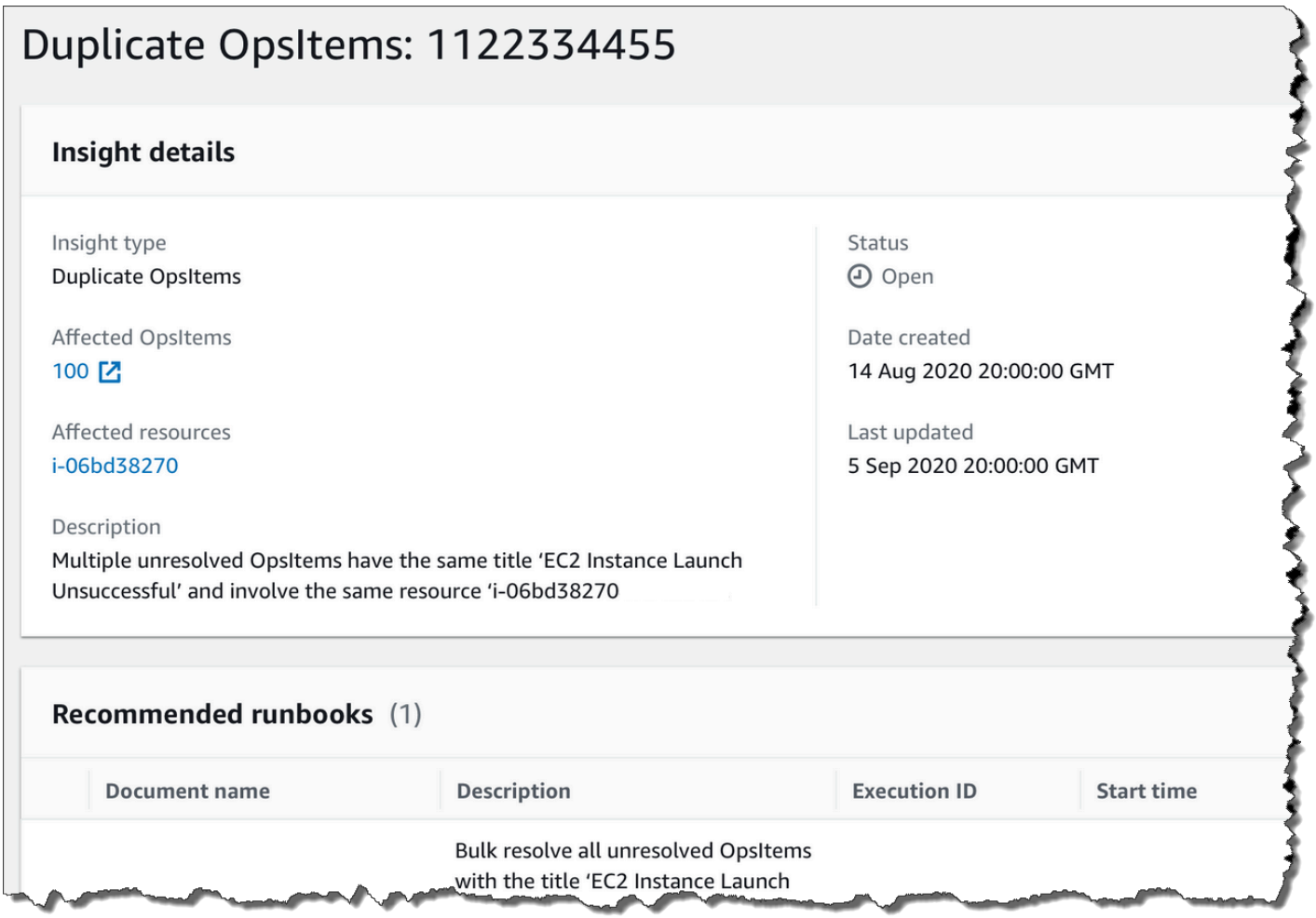
- **Duplicate OpsItems** – An insight is generated when eight or more OpsItems have the same title for the same resource.
- **Most common titles** – An insight is generated when more than 50 OpsItems have the same title.
- **Resources generating the most OpsItems** – An insight is generated when an AWS resource has more than 10 open OpsItems. These insights and their corresponding resources are displayed in the **Resources generating the most OpsItems** table on the OpsCenter **Summary** tab. Resources are listed in decreasing order of OpsItem count.

> **Note**
>
> OpsCenter creates **Resources generating the most OpsItems** insights for the following resource types:
>
> - Amazon Elastic Compute Cloud (Amazon EC2) instances
> - Amazon EC2 security groups
> - Amazon EC2 Auto Scaling group
> - Amazon Relational Database Service (Amazon RDS) database
> - Amazon RDS cluster
> - AWS Lambda function
> - Amazon DynamoDB table
> - Elastic Load Balancing load balancer
> - Amazon Redshift cluster
> - AWS Certificate Manager certificate
> - Amazon Elastic Block Store volume

OpsCenter enforces a limit of 15 insights per type. If a type reaches this limit, OpsCenter stops displaying more insights for that type. To view additional insights, you must resolve all OpsItems associated with an OpsInsight of that type. If a pending insight is prevented from being displayed

in the console because of the 15-insight limit, that insight becomes visible after another insight is closed.

When 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.



Operational insights are turned off by default. For more information about working with operational insights, see the following topics.

Topics

- Enabling operational insights
- Resolving duplicate OpsItems based on insights
- Disabling operational insights

Enabling operational insights

You can enable operational insights on the **OpsCenter** page in the Systems Manager console. When 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 that the service requires to call other AWS services on your behalf. For more information about the `AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role, see Using roles to create operational insight OpsItems in Systems Manager OpsCenter.

> **Note**
>
> Note the following important information:
>
> - Your AWS account is charged for operational insights. For more information, see AWS Systems Manager Pricing.
> - OpsCenter periodically refreshes insights using a batch process. This means the list of insights displayed in OpsCenter might be out of sync.

Use the following procedure to enable and view operational insights in OpsCenter.

To enable and 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. In the **Operational insight is available** message box, choose **Enable**. If you don't see this message, scroll down to the **Operational insights** section and choose **Enable**.
4. After you enable this feature, on the **Summary** tab, scroll down to the **Operational insights** section.
5. To view a filtered list of insights, choose the link beside **Duplicate OpsItems**, **Most common titles**, or **Resources generating the 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

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.

To help you resolve duplicate OpsItems and reduce the number of OpsItems created by a source, Systems Manager provides the following automation runbooks:

- The `AWS-BulkResolveOpsItems` runbook resolves OpsItems that match a specified filter.
- The `AWS-AddOpsItemDedupStringToEventBridgeRule` runbook adds a deduplication string for all OpsItem targets that are associated with a specific Amazon EventBridge rule. This runbook doesn't add a deduplication string if a rule already has one.
- The `AWS-DisableEventBridgeRule` turns off a rule in EventBridge if the rule is generating dozens or hundreds of OpsItems.

To resolve an operational insight

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 **View all operational insights**.
5. Choose an insight ID to view more information.
6. Choose a runbook and choose **Execute**.

Disabling operational insights

When you turn off operational insights, the system stops creating new insights and stops displaying insights in the console. Any active insights remain unchanged in the system, although you won't see them displayed in the console. If you enable this feature again, the system displays previously unresolved insights and starts creating new insights. Use the following procedure to turn off operational insights.

To turn off 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 **Settings**.
4. In the **Operational insights** section, choose **Edit** and then toggle the **Disable** option.
5. Choose **Save**.

Viewing OpsCenter logs and reports

AWS CloudTrail logs AWS Systems Manager OpsCenter API calls to 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. Amazon S3 uses one bucket to store 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.

AWS Systems Manager OpsCenter provides you with the following information about OpsItems:

- **OpsItem status summary** – Provides a summary of OpsItems by status (Open and In progress, Open, or In Progress).
- **Sources with most open OpsItems** – Provides a breakdown of the top AWS services with open OpsItems.
- **OpsItems by source and age** – Provides a count of OpsItems, grouped by source and days since creation.

To view the OpsCenter summary report

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**.

Delete OpsItems

You can delete an individual OpsItem by calling the DeleteOpsItem API operation using the AWS Command Line Interface or the AWS SDK. You can't delete an OpsItem in the AWS Management Console. To delete an OpsItem, your AWS Identity and Access Management (IAM) user, group, or role must have either administrator permission or you must have been granted permission to call the `DeleteOpsItem` API operation.

Important

Note the following important information about this operation.

- Deleting an OpsItem is irreversible. You can't restore a deleted OpsItem.
- This operation uses an *eventual consistency model*, which means the system can take a few minutes to complete this operation. If you delete an OpsItem and immediately call, for example, GetOpsItem, the deleted OpsItem might still appear in the response.
- This operation is idempotent. The system doesn't throw an exception if you repeatedly call this operation for the same OpsItem. If the first call is successful, all additional calls return the same successful response as the first call.
- This operation doesn't support cross-account calls. A delegated administrator or management account can't delete OpsItems in other accounts, even if OpsCenter has been set up for cross-account administration. For more information about cross-account administration, see (Optional) Setting up OpsCenter to centrally manage OpsItems across accounts.
- If you receive the `OpsItemLimitExceededException`, you can delete one or more OpsItems to reduce your total number of OpsItems below the quota limits. For more information about this exception, see Troubleshooting issues with OpsCenter.

Deleting an OpsItem

Use the following procedure to delete an OpsItem.

To delete an OpsItem

1. Install and configure the AWS CLI, if you haven't already. For more information, see Installing or updating the latest version of the AWS CLI.
2. Run the following command. Replace *ID* with the ID of the OpsItem you want to delete.

```
aws ssm delete-ops-item --ops-item-id ID
```

If successful, the command returns no data.

Remediate OpsItem issues

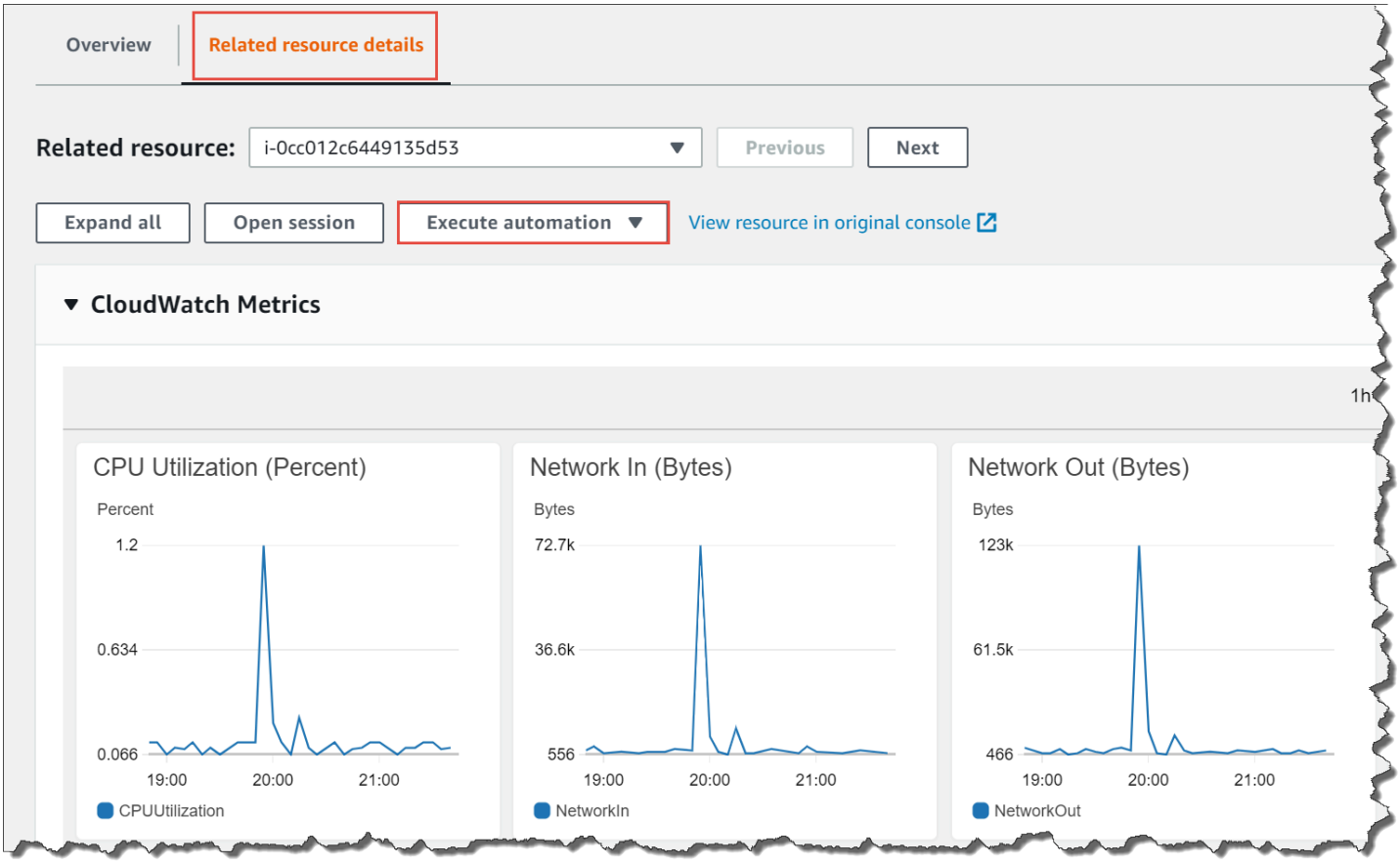
Using AWS Systems Manager Automation runbooks, you can remediate issues with AWS resources that are identified in an OpsItem. Automation uses predefined runbooks to remediate common issues with AWS resources.

Each OpsItem includes the **Runbooks** section that provides a list of runbooks that you can use for remediation. When you choose an Automation runbook from the list, OpsCenter automatically displays some of the fields required to run the document. When you run an Automation runbook, the system associates the runbook with the related resource of the OpsItem. If Amazon EventBridge creates an OpsItem, it associates a runbook with the OpsItem. OpsCenter keeps a 30-day record of Automation runbooks for an OpsItem.

You can choose a status to view important details about the runbook, 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 **Related resource details** page for a selected OpsItem includes the **Run automation** list. You can choose recent or resource-specific Automation runbooks and run them to remediate issues. This page also includes data providers, including Amazon CloudWatch metrics and alarms, AWS CloudTrail logs, and details from AWS Config.



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.

Remediating an OpsItem using a runbook

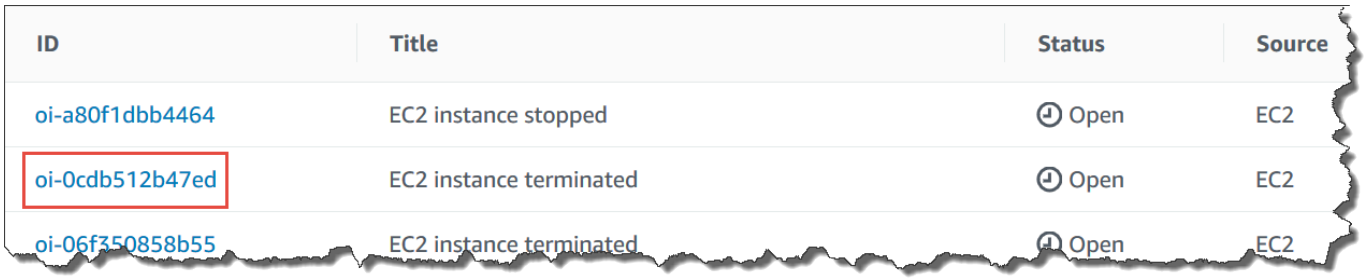
Before you use an Automation runbook to remediate an OpsItem issue, do the following:

- Verify that you have permission to run Systems Manager Automation runbooks. For more information, see Setting up Automation.
- 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 EC2 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.



4. Scroll to the **Runbooks** section.
5. Use the 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 **Submit**.

   Once you start the runbook, the system returns to the previous screen and displays the status.
8. In the **Automation executions in the last 30 days** section, choose the **Execution ID** link to view steps and the status of the execution.

Remediating an OpsItem using an associated runbook

After you run an Automation runbook from an OpsItem, OpsCenter associates the runbook with the OpsItem. An *associated* runbook is ranked higher than other runbooks 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 Manage OpsItems.

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**.

   Once you start the runbook, the system returns to the previous screen and displays the status.

7. In the **Automation executions in the last 30 days** section, choose the **Execution ID** link to view 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, such as `Open` and `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 number of days since creation.

To view OpsCenter summary reports

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **OpsCenter**, and then choose the **Summary** tab.
3. In the **OpsItems by source and age** section, do the following:

   1. (Optional) In the filter field, choose **Source**, select `Equal`, `Begin With`, or `Not Equal`, and then enter a search parameter.
   2. In the adjacent list, select one of the following status values:
      - `Open`
      - `In progress`
      - `Resolved`
      - `Open and in progress`
      - `All`

Troubleshooting issues with OpsCenter

This topic includes information to help you troubleshoot common errors and issues with OpsCenter.

You receive the OpsItemLimitExceededException

If your AWS account has reached the maximum number of OpsItems allowed when you call the CreateOpsItem API operation, you receive an `OpsItemLimitExceededException`. OpsCenter returns the exception if your call would exceed the maximum number of OpsItems for either of the following quotas:

- Total number of OpsItems per AWS account per Region (including `Open` and `Resolved` OpsItems): 500,000
- Maximum number of OpsItems per AWS account per month: 10,000

These quotas apply to OpsItems created from any source except the following:

- OpsItems created by AWS Security Hub findings
- OpsItems that are auto-generated when an Incident Manager incident is opened

OpsItems created from these sources don't count against your OpsItem quotas, but you are charged for each OpsItem.

If you receive an `OpsItemLimitExceededException`, you can manually delete OpsItems until you are below the quota preventing you from creating a new OpsItem. Again, deleting OpsItems created for Security Hub findings or Incident Manager incidents won't reduce your total number of OpsItems enforced by the quotas. You must delete OpsItems from other sources. For information about how to delete an OpsItem, see Delete OpsItems.

You receive a large bill from AWS for large numbers of auto-generated OpsItems

If you configured integration with AWS Security Hub, OpsCenter creates OpsItems for Security Hub findings. Depending on the number of finding Security Hub generates and the account you were logged into when you configured integration, OpsCenter can generate large numbers of OpsItems, at a cost. Here are more specific details related to OpsItems generated by Security Hub findings:

- If you are logged into the Security Hub administrator account when you configure OpsCenter and Security Hub integration, the system creates OpsItems for findings in the administrator *and*

all member accounts. The OpsItems are all created *in the administrator account*. Depending on a variety of factors, this can lead to an unexpectedly large bill from AWS.

If you are logged into a member account when you configure integration, the system only creates OpsItems for findings in that individual account. For more information about the Security Hub administrator account, member accounts, and their relation to the EventBridge event feed for findings, see Types of Security Hub integration with EventBridge in the *AWS Security Hub User Guide*.

- For each finding that creates an OpsItem, you are charged the regular price for creating the OpsItem. You are also charged if you edit the OpsItem or if the corresponding finding is updated in Security Hub (which triggers an OpsItem update).
- OpsItems that are created by an integration with AWS Security Hub are *not* currently limited by the maximum quota of 500,000 OpsItems per account in a Region. It is therefore possible for Security Hub alerts to create more than 500,000 chargeable OpsItems in each Region in an account.

  For high-production environments, we therefore recommend limiting the scope of Security Hub findings to high severity issues only.

> **Important**
>
> If you believe a large number of OpsItems were created in error and your AWS bill is unwarranted, contact Support.

Use the following procedure if you no longer want the system to create OpsItems for Security Hub findings.

To stop receiving OpsItems for 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. Choose **Settings**.
4. In the **Security Hub findings** section, choose **Edit.**

5. Choose the slider to change **Enabled** to **Disabled**. If you aren't able to toggle the slider, Security Hub hasn't been enabled for your AWS account.
6. Choose **Save** to save your configuration. OpsCenter no longer creates OpsItems based on Security Hub findings.

> **Important**
>
> If OpsCenter toggles the setting back to **Enabled** and continues to create OpsItems for findings, log into the Systems Manager delegated administrator account or the AWS Organizations management account and repeat this procedure. If you don't have permission to log into either of those accounts, contact your administrator and ask them to repeat this procedure to disable integration for your account.

Using 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*.

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, 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 Gateway Service (`ssmmessages`). (In AWS Regions launched before 2024, status and execution information might also be sent back by the Amazon Message Delivery Service (service prefix: `ec2messages`).)

If you monitor traffic, you will see that your managed nodes communicate with `ssmmessages.*` endpoints and possibly `ec2messages.*` endpoints. For more information, see Reference: ec2messages, ssmmessages, and other API operations. For information about porting SSM Agent logs to Amazon CloudWatch Logs, see Logging and monitoring in AWS Systems Manager.

Contents

- Learn technical details about the SSM Agent
- Find AMIs with the SSM Agent preinstalled
- Working with SSM Agent on EC2 instances for Linux
- Working with SSM Agent on EC2 instances for macOS
- Working with SSM Agent on EC2 instances for Windows Server
- Checking SSM Agent status and starting the agent
- Checking the SSM Agent version number
- Viewing SSM Agent logs
- Restricting access to root-level commands through SSM Agent
- Automating updates to SSM Agent
- Subscribing to SSM Agent notifications
- Troubleshooting SSM Agent

Learn technical details about the SSM Agent

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 version 3.2.x.x credential behavior
- SSM Agent credentials precedence
- Configuring SSM Agent for use with the Federal Information Processing Standard (FIPS)
- About the local ssm-user account
- SSM Agent and the Instance Metadata Service (IMDS)
- Keeping SSM Agent up-to-date
- Ensuring that the SSM Agent installation directory is not modified, moved, or deleted
- SSM Agent rolling updates by AWS Regions
- SSM Agent communications with AWS managed S3 buckets
- SSM Agent on GitHub
- Understanding SSM Agent hibernation

SSM Agent version 3.2.x.x credential behavior

SSM Agent stores a set of temporary credentials at `/var/lib/amazon/ssm/credentials` (for Linux and macOS) or `%PROGRAMFILES%\Amazon\SSM\credentials` (for Windows Server) when an instance is onboarded using the Default Host Management Configuration in Quick Setup. The temporary credentials have the permissions you specify for the IAM role you chose for Default Host Management Configuration. On Linux, only the root account can access these credentials. On Windows Server, only the SYSTEM account and local Administrators can access these credentials.

SSM Agent credentials precedence

This topic describes important information about how SSM Agent is granted permission to perform actions on your resources.

> **Note**
>
> Support for edge devices differs slightly. You must configure your edge 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 Managing edge devices with Systems Manager.

When SSM Agent is installed on a machine, 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 non-EC2 machine, 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. If a hybrid-activated node is deregistered, the agent may enter hibernation mode. For more information, see Understanding SSM Agent hibernation.

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

For example, say that you have configured an EC2 instance to be managed by Systems Manager. That configuration includes attaching an instance profile. 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 users 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.
5. The IAM role chosen for Default Host Management Configuration.

For related information, see the following topics:

- Instance profiles for EC2 instances – Configure instance permissions required for Systems Manager
- Hybrid activations – Create a hybrid activation to register nodes with Systems Manager
- 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.

Configuring SSM Agent for use with the Federal Information Processing Standard (FIPS)

If you need to use Systems Manager with Federal Information Processing Standard (FIPS) 140-3 validated cryptographic modules, you can configure AWS Systems Manager Agent (SSM Agent) to use the FIPS endpoints in supported Regions.

To configure SSM Agent to connect to FIPS 140-3 endpoints

1. Connect to your managed node.
2. Navigate to the directory that contains the `amazon-ssm-agent.json` file:
   - Linux: `/etc/amazon/ssm/`
   - macOS: `/opt/aws/ssm/`

- Windows Server: `C:\Program Files\Amazon\SSM\`

3. Open the file named `amazon-ssm-agent.json` for editing.

> **Tip**
>
> If no `amazon-ssm-agent.json` file exists yet, 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.

4. Add the following content to the file. Replace the *`region`* placeholder values with the appropriate Region code for your partition:

```
{
    ---Existing file content, if any---
    "Mds": {
        "Endpoint": "ec2messages-fips.region.amazonaws.com",
    },
    "Ssm": {
        "Endpoint": "ssm-fips.region.amazonaws.com",
    },
    "Mgs": {
        "Endpoint": "ssmmessages-fips.region.amazonaws.com",
        "Region": "region"
    },
    "S3": {
        "Endpoint": "s3-fips.dualstack.region.amazonaws.com",
        "Region": region"
    },
    "Kms": {
        "Endpoint": "kms-fips.region.amazonaws.com"
    }
}
```

Supported Regions include the following:

- `us-east-1` for the US East (N. Virginia) Region
- `us-east-2` for the US East (Ohio) Region
- `us-west-1` for the US West (N. California) Region

- `us-west-2` for the US West (Oregon) Region
- `ca-west-1` for the Canada West (Calgary) Region

5. Save the file and 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 in the `amazon-ssm-agent` repository in GitHub.

For more information about AWS support for FIPS, see Federal Information Processing Standard (FIPS) 140-3.

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 tool in 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 Step 2: Verify or add instance permissions for Session Manager.

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*.

Keeping SSM Agent up-to-date

An updated version of SSM Agent is released whenever new tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. 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 tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. 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.

Ensuring that the SSM Agent installation directory is not modified, moved, or deleted

SSM Agent is installed at `/var/lib/amazon/ssm/` (Linux and macOS) and `%PROGRAMFILES%\Amazon\SSM\` (Windows Server). These installation directories contain critical files and folders used by SSM Agent, such as a credentials file, resources for inter-process communication (IPC),

and orchestration folders. Nothing within the installation directory should be modified, moved, or deleted. Otherwise, SSM Agent might cease to function properly.

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.

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 non-EC2 machines in a hybrid and multicloud 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 non-EC2 managed nodes).

For information about using a virtual private cloud (VPC) endpoint in your Systems Manager operations, see Improve the security of EC2 instances by using VPC endpoints for Systems Manager.

> **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:

- Configure instance permissions required for Systems Manager
- Create the IAM service role required for Systems Manager in hybrid and multicloud environments
- Reference: Amazon S3 buckets for patching operations

Contents

- Required bucket permissions
- Example
- Validating hybrid-activated machines using a hardware fingerprint

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
arn:aws:s3:::aws-windows-downloads- *region*/*	Required for some SSM documents that support only Windows Server operating systems, plus some for cross-platform support, such as AWSEC2-ConfigureSTIG .
arn:aws:s3:::amazon-ssm- *region*/*	Required for updating SSM Agent installations. These buckets contain the SSM Agent installation packages, and the installation manifests that are referenced by the AWS-UpdateSSMAgent document and plugin. If these permissions aren't provided, the SSM Agent makes an HTTP call to download the update.
arn:aws:s3:::aws-ssm- *region*/*	Provides access to the S3 bucket containing modules required for use with non-patching Systems Manager documents (SSM Command documents). For example: arn:aws:s3:::aws-ssm-us-east-2/* . The following are some commonly used SSM documents stored in these buckets. • AWS-ConfigureWindowsUpdate • AWS-FindWindowsUpdates • AWS-UpdateSSMAgent • AWS-UpdateEC2Config
arn:aws:s3:::patch-baseline-snapshot- *region*/* -or- arn:aws:s3:::patch-baseline-snapshot- *region-unique-suffix* /*	Provides access to the S3 bucket containing patch baseline snapshots. This is required if you use any of the following SSM Command documents: • AWS-RunPatchBaseline

S3 bucket ARN	Description
	• `AWS-RunPatchBaselineAssociation` • `AWS-RunPatchBaselineWithHooks` • `AWS-ApplyPatchBaseline` (a legacy SSM document) The buckets for most supported AWS Regions use the following format: `arn:aws:s3:::patch-baseline-snapshot-` *region* For some Regions, an additional unique suffix is included in the bucket name. For example, the bucket name for the Middle East (Bahrain) Region (me-south-1) is as follows: • `patch-baseline-snapshot-me-south-1-uduvl7q8` For a complete list of patch baseline snapshot bucket names, see Buckets containing AWS managed patch baseline snapshots. **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.

S3 bucket ARN	Description
For Linux and Windows Server managed nodes: `arn:aws:s3:::aws-patch-manager-` *region*`-`*unique-suffix*` /*` For Amazon EC2 instances for macOS: `arn:aws:s3:::aws-patchmanager-macos-` *region*`-`*unique-suffix*` /*`	Provides access to the S3 bucket containing SSM Command documents for patching operations in Patch Manager. Each bucket name includes a unique suffix, such as `552881074` for buckets in the US East (Ohio) (us-east-2) Region: • `arn:aws:s3:::aws-patch-manager-us-east-2-552881074/*` • `arn:aws:s3::: aws-patchmanager-macos-us-east-2-552881074/*` **SSM documents** The following are some commonly used SSM documents stored in these buckets. • `AWS-RunPatchBaseline` • `AWS-RunPatchBaselineAssociation` • `AWS-RunPatchBaselineWithHooks` • `AWS-InstanceRebootWithHooks` • `AWS-PatchAsgInstance` • `AWS-PatchInstanceWithRollback` For complete lists of AWS managed S3 buckets for patching operations, see the following topics: • Buckets containing SSM Command documents for patching operations (Linux and Windows Server)

S3 bucket ARN	Description
	• Buckets containing SSM Command documents for patching operations (macOS)

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.

JSON

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": "s3:GetObject",
                "Resource": [
                    "arn:aws:s3:::aws-windows-downloads-us-east-2/*",
                    "arn:aws:s3:::amazon-ssm-us-east-2/*",
                    "arn:aws:s3:::aws-ssm-us-east-2/*",
                    "arn:aws:s3:::patch-baseline-snapshot-us-east-2/*",
                    "arn:aws:s3:::aws-patch-manager-us-east-2-552881074/*",
                    "arn:aws:s3:::aws-patchmanager-macos-us-east-2-552881074/*"
                ]
            }
        ]
    }
```

Validating hybrid-activated machines using a hardware fingerprint

When non-EC2 machines in a hybrid and multicloud 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 hybrid-activated 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 machine 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 in 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 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 machines:

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

On Windows Server machines 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**. For more information about hibernation, see Understanding SSM Agent hibernation.

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.

Understanding SSM Agent hibernation

AWS Systems Manager Agent (SSM Agent) hibernation is an operational mode that occurs when the agent can't maintain proper communication with the Systems Manager service. During hibernation, the agent reduces its communication frequency and enters a standby state.

When SSM Agent hibernation occurs

SSM Agent hibernation can occur in the following scenarios:

Deregistered hybrid nodes

When you deregister a hybrid-activated node from Systems Manager, the SSM Agent on that node can't refresh its authorization token. This causes the agent to enter hibernation mode because it can't authenticate with the service.

Hardware fingerprint changes

SSM Agent uses a hardware fingerprint to validate hybrid-activated machines. If one of the components used to calculate this fingerprint changes significantly, the agent might hibernate as a security measure. This is by design to prevent multiple managed nodes from communicating with Systems Manager as the same node. For more information, see Validating hybrid-activated machines using a hardware fingerprint.

SSM Agent hibernation on Amazon EC2 instances

Hibernation can also occur on Amazon EC2 instances under certain conditions, such as when there are connectivity issues or authentication problems with the Systems Manager service.

Hibernation communication behavior

When SSM Agent enters hibernation mode, its communication pattern with the Systems Manager service changes:

- **Normal operation**: The agent regularly communicates with Systems Manager (typically every few minutes) to check for new commands and report status.
- **Hibernation mode**: The ping frequency starts at 5 minutes and gradually increases to once per hour by default (configurable up to 24 hours). This reduced communication frequency helps minimize unnecessary network traffic while still allowing the agent to potentially recover if conditions change.

During hibernation, the agent continues to retry authentication and connection attempts at the reduced frequency, but it can't process new commands or report detailed status information until hibernation is resolved.

Configuration options to prevent hibernation in hybrid instances

The primary configuration option to help prevent hibernation caused by hardware fingerprint changes is adjusting the similarity threshold:

On Linux machines:

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

On Windows Server machines using PowerShell:

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

The similarity threshold determines how strictly the agent compares current machine attributes with the stored hardware hash:

- Higher values require more matching attributes
- Lower values (such as **1**) are more lenient and can help avoid hibernation caused by minor hardware changes

Hibernation logging and monitoring

When SSM Agent enters hibernation mode, it creates log entries that can help you identify and troubleshoot the hibernation state:

- **Agent log files**: Hibernation events are logged in the standard SSM Agent log files. For more information about log file locations, see Troubleshoot issues using SSM Agent log files.
- **Amazon EC2 console logging**: For EC2 instances, hibernation messages are now logged to the Amazon EC2 console system logs, providing additional visibility into agent status. To access the logs, select the instance in the EC2 console, and then choose **Actions**, **Monitor and troubleshoot**, **Get system log**.
- **Specific log files**: When hibernation starts, particular log files are created that contain detailed information about the hibernation trigger and status.

Monitor these log sources to detect hibernation events early and take corrective action to restore normal agent operation.

Recovering from hibernation

To recover from hibernation, address the underlying cause:

- **For deregistered hybrid nodes**: Reregister the node with Systems Manager using a new activation code and ID, as described in the section called “Deregister and reregister a managed node (Linux)” and the section called “Deregister and reregister a managed node (Windows Server)”.

- **For hardware fingerprint issues**: Adjust the similarity threshold as described above under **Configuration options to prevent hibernation in hybrid instances**, or re-register the node if hardware changes are significant.
- **For connectivity issues**: Verify network connectivity and make sure that the required endpoints are accessible. For more information, see Troubleshooting managed node availability using `ssm-cli`.

After you resolve the underlying issue, the agent should automatically exit hibernation mode and resume normal operation at the next communication attempt.

Find AMIs with the SSM Agent preinstalled

AWS Systems Manager Agent (SSM Agent) is preinstalled on some Amazon Machine Images (AMIs) provided by AWS and trusted third-parties.

For example, when you launch an Amazon Elastic Compute Cloud (Amazon EC2) instance created from an AMI with one of the following operating systems, you'll likely find that the SSM Agent is already installed:

- AlmaLinux
- Amazon Linux 2
- Amazon Linux 2 ECS-Optimized Base AMIs
- Amazon Linux 2023 (AL2023)
- Amazon EKS-Optimized Amazon Linux AMIs
- macOS 13.x (Ventura), 14.x (Sonoma), and 15.x (Sequoia)
- Ubuntu Server 16.04 LTS 64-bit (Snap), 18.04, 20.04, 22.04 LTS, 24.04 LTS, 24.0, and 25.04
- Windows Server 2012 R2 AMIs published in November 2016 or later
- Windows Server 2016, 2019, 2022 (excluding Nano versions), and 2025

> **Note**
>
> The version of SSM Agent preinstalled on an AMI might not be the latest available version. As a best practice, we recommend always using the latest available version of SSM Agent on your managed nodes. For more information about automating SSM Agent updates, see Automating updates to SSM Agent.

SSM Agent might be preinstalled on AWS managed AMIs that aren't on this list. This typically indicates that the operating system (OS) is not fully supported by all Systems Manager tools.
SSM Agent might also be preinstalled on AMIs found in AWS Marketplace or in the Community AMIs repository, but AWS doesn't support these AMIs.

Verify the status of SSM Agent

Depending on when it was initialized, an instance created from an AMI on the preceding list might not have SSM Agent preinstalled. It's also possible that an instance has the agent preinstalled, but the agent isn't running. Therefore, we recommend that you check the status of SSM Agent before you try to use Systems Manager on an instance for the first time.

Use the following procedure to verify that SSM Agent is installed and running on an instance. If you find that the agent is not installed, you can manually install it on Linux, macOS, and Windows Server instances.

To verify installation of SSM Agent on an instance

1. After launching a new instance, wait a few minutes for it to initialize.
2. Connect to the instance using your preferred method. For example, you can use SSH to connect to Linux instances or use Remote Desktop to connect to Windows Server instances.
3. Check the status of SSM Agent by running the command for your instance's operating system type.

Operating system	Command
Amazon Linux 2 and Amazon Linux 2023	`sudo systemctl status amazon-ssm-agent`
macOS	There is no command to check SSM Agent status on macOS. You can check the status by locating and evaluating the agent log file `/var/log/amazon/ssm/amazon-ssm-agent.log` .

Operating system	Command
Ubuntu Server (64-bit - Deb)	`sudo systemctl status amazon-ssm-agent`
Ubuntu Server (64-bit - Snap)	`sudo systemctl status snap.amazon-ssm-agent.amazon-ssm-agent.service`
Windows Server	`Get-Service AmazonSSMAgent`

> **Tip**
>
> To view the commands for checking SSM Agent status on all operating system types supported by Systems Manager, see Checking SSM Agent status and starting the agent.

4. Evaluate the command output to learn the status of the SSM Agent.

Status: *Installed and running*

In most cases, the command output indicates that the agent is installed and running.

The following example shows that SSM Agent is installed and running on an Amazon Linux 2 instance.

```
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--
```

The following example shows that SSM Agent is installed and running on a Windows Server instance.

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

Status: *Installed but not running*

In some cases, the command output indicates that the agent is installed but not running.

The following example shows that SSM Agent is installed but not running on an Amazon Linux 2 instance.

```
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--
```

The following example shows that SSM Agent is installed but not running on a Windows Server instance.

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

If the agent is installed but not running, you can activate it manually using the commands for your instance's operating system type.

Operating system	Command

Amazon Linux 2 and Amazon Linux 2023	`sudo systemctl enable amazon-ssm-agent`
	
	`sudo systemctl start amazon-ssm-agent`

Operating system	Command

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

Ubuntu Server (64-bit - Deb)	`sudo systemctl enable amazon-ssm-agent`
	
	`sudo systemctl start amazon-ssm-agent`

Ubuntu Server (64-bit - Snap)	`sudo snap start amazon-ssm-agent`
Windows Server	Run the following command in PowerShell. `Start-Service AmazonSSMAgent`

Status: *Not installed*

In some cases, the command output indicates that the agent is not installed.

The following example shows that SSM Agent is not installed on an Amazon Linux 2 instance.

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

The following example shows that SSM Agent is not installed on a Windows Server instance.

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

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

- Manually installing and uninstalling SSM Agent on EC2 instances for Linux
- Manually installing and uninstalling SSM Agent on EC2 instances for macOS

- Manually installing and uninstalling SSM Agent on EC2 instances for Windows Server

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

- Verifying the signature of SSM Agent
- Manually installing and uninstalling SSM Agent on EC2 instances for Linux
- Configuring SSM Agent to use a proxy on Linux nodes

Verifying the signature of 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 the files are damaged or have been altered, 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	AlmaLinux, Amazon Linux 2, Amazon Linux 2023, RHEL, Oracle Linux, Rocky Linux	`https://s3.`*`region`*`.amazonaws.com/amazon-ssm-`*`region`*`/latest/l`	`amazon-ssm-agent.rpm`

Architecture	Operating system	Signature file URL	Agent download file name
		inux_amd64/amazon-ssm-agent.rpm.sig https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm.sig	
x86_64	Debian Server, Ubuntu Server	https://s3. *region*.amazonaws.com/amazon-ssm-*region*/latest/debian_amd64/amazon-ssm-agent.deb.sig https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_amd64/amazon-ssm-agent.deb.sig	amazon-ssm-agent.deb

Architecture	Operating system	Signature file URL	Agent download file name

ARM64	Amazon Linux 2, Amazon Linux 2023, RHEL	`https://s3.region.amazonaws.com/amazon-ssm-region/latest/linux_arm64/amazon-ssm-agent.rpm.sig`
		
		`https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_arm64/amazon-ssm-agent.rpm.sig`
			`amazon-ssm-agent.rpm`

Verifying the SSM Agent package on a Linux server (v3.3.1802.0 and later)

Before you begin

The procedures for **GPG** and **RPM** in this section apply to SSM Agent version 3.3.1802.0 and later. Before verifying the signature of SSM Agent using the following procedure, ensure that you have downloaded the latest agent package for your operating system. For example, `https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_arm64/amazon-ssm-agent.rpm`. For more information about downloading SSM Agent packages, see Manually installing and uninstalling SSM Agent on EC2 instances for Linux.

If you have a reason to continue using agent version 3.3.1611.0 or earlier, follow the instructions in Verifying the SSM Agent package on a Linux server (v3.3.1611.0 and earlier) instead.

GPG

To verify the SSM Agent package on a Linux server (v3.3.1802.0 and later)

1. Copy one the following public key and save it to a file named `amazon-ssm-agent.gpg`.

> **Important**
>
> The following public key expires on 2026-07-15 (July 15, 2026). Systems Manager will publish a new public key in this topic before the old one expires. We encourage you to subscribe to the RSS feed for this topic to get a notification when the new key is available.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGeRNq4BEACrlf5h6Pz+k+M+QCJJ2LfK7d2Tn9J8iJ9qBK2Vwvuxco1rpSO+
KEI3nTeysPuheximps8WOCADX4VlbsKxMZQLjQM4mA26m1Tiw9nAI4kod4bKjiuM
BMUTCD1wfnjH3zQi4kDUdbpfAEMiPgNLVLH85Wf+lhK+Zm+V38DYzLyVj03kX4wK
iG6RMoxzOBZa5gNsVq+j+oCUITGz/URxH713Rgo8WeoEegI0+7iCBLKg+PM0b7GV
2nzkwWJz796HdkqSg8BwXsYaLTrHxa2P1IpwPCisAkyO7gZaMd6Uj69dtMFO+V8a
Qee6b57qGuFKZw7h1Vvc85PbF1Gy/wNIpary57kUHBFUg1vYep/roJuEbJCq97r5
I2liLl4NAyrWb9r/TAVxlXvqM4iZUhxm8GAp0FywMdBr9ZECClKa5HxuVmlm0Wgl
TXoYTOZKeDg6ZoCvyhNxWneCNip74fohXymeFF5L/budhBwy5wuwSniOgTGLo/4C
VgZHWCcN+d0Q3bx/sl2QNqPg5/xzsxEtymXLdVdwLIsLdEQUnIvy8KTs5jol3Dwi
nnEEyhly6wdaw+qDOhkSOT/VnErrSMkYF8VJfa5GjhCBWKw9JVSkaP2CI/VHOgHM
MKROnulq0hRQBR7RmLYt98xu38BHJWMmF8Ga/HJuIxzD1VmkZOPvDDESUwARAQAB
tCdTU00gQWdlbnQgPHNzbS1hZ2VudC1zaWduZXJAYW1hem9uLmNvbT6JAj8EEwEC
ACkFAmeRNq4CGy8FCQLGmIAHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIXgAAKCRBR
qOBQ0AUuXTdND/9qldQ1E3dYjBVXOnbhiUQL594bkS5VoEX7D4fZ5UMVZa5pGiz+
husnoRUS9rH1cSeq7aHJu9hSCMuMdvRpuoo0CwLB+7HtzJvAO2M01hcEkUYa6Qdj
njTzP0ZjnoenJmqF9SYmVqAI/VPa9mNQ1OJ+HQ3qh5i6w+FoWlVqEdXjZGrWijub
TqyN33i1Y26t7Os/x8I9fUeNx37y/7Kama8LTdtv9GhWiMVBg2IuVf27HCMYofrQ
m2uCGe61IhtsnhsYaYupmljl+6qgdiuCiS9BAsoIGtqTnu8lnKcGyGz6YnRszN+U
1bNE4w+UFpXWJF8ogpYcghJ06aW/LhjZnQSx3VliLdW8eOJzou41yWmiuL3ZY8eW
KAlD+7eYKS6N6fEJCeNO2VX2lcKtDfaOX+lqGIVyexKayMfpi+0frNzt/92YCpF5
3jkeS77vMMVqKIUiIp1OCGv3XsFpIr6Bt2c2throYPDoQL3zvq6vvG40BKeRQ4tT
Y+5vTc8MeNn3LdzTl9pusxTcKifrJq7f5FIsL2CpAX8uQ+Qz+XWsYQQ5PvyUDtOz
nU/MRZaP6HnqY42bzI9ZlKgXi9IE3MXIwoET9YyzFjkIDvat7SlB4uJCpeIqp/KM
OIrTMb7paGLYmBU6YqxNBkDWItNG7NeZzyhh/R/Qqb4vJaf4S+ZqD1RZXokCHAQQ
AQIABgUCZ5E2rwAKCRB90Jej2tf1/CdnD/46It+RNoE00TesZK5n2bijH5Eljw0E
```

```
4/UpMi1SV6t2zY7lIm7TcKNn18tynJNFqB6YXXOwSbBG/fbN2E9RaoUCZw23TmAv
amuHwrfsDqsHb7zzPF0bISYjqEDLQJj/gtEugUc6XY1dEpFSlWJIOvgryG04cFXI
uD2KY87ya4s1R+sEVAJ14K4RlUCiMmzJdR0NJNYJOwBi1gkLEp6jG86ttiG2U7fY
pE2ibV+c0GeIFq8PIzqqENsn9KBuRH5EcbdBwfnsj2XfM4aR3ZtRIdWXkKkdP9Rs
yU5dTF/Y7XPId5h8/gp00+DMlXFBinQ1jE7A7eDYviEFd1ba8P7dIom3Q3gzKiWu
KTGpnykShs5NvpQmvGUF6JqDHI4RK9s3kLqsNyZkhenJfRBrJ/45fQAuP4CRedkF
7PSfX0Xp7kDnKuyK6wEUEfXXrqmuLGDmigTXblO5qgdyMwkOLjiY9znBZbHoKs76
VplOoNgGnN19i3nuMcPf2npFICJv7kTIyn5Fh7pjWDCahl3U/PwoLjrrlEzpyStU
oXSZrK3kiAADEdSODXJl8KYU0Pb27JbRr1ZbWnxb+O39TOhtssstulkR0v+IDGDQ
rQE1b12sKgcNFSzInzWrNGu4S06WN8DYzlrTZ9aSHj+37ZqpXAevi8WOFXKPV3PA
E6+O8RI2451Dcg==
=aDkv
-----END PGP PUBLIC KEY BLOCK-----
```

2. Import the public key into your keyring, and note the returned key value.

```
gpg --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   4096R/D0052E5D 2025-01-22 [expires: 2026-07-15]
      Key fingerprint = 4855 A9E6 8332 16D6 A77D  8FE4 51A8 E050 D005 2E5D
uid                  SSM Agent <ssm-agent-signer@amazon.com>
```

The fingerprint should match the following.

```
4855 A9E6 8332 16D6 A77D 8FE4 51A8 E050 D005 2E5D
```

If the fingerprint doesn't match, don't install the agent. Contact AWS Support.

4. Download the signature file according to your instance's architecture and operating system if you haven't already done so.
5. 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, as listed in the table earlier in this topic.

```
gpg --verify signature-filename agent-download-filename
```

For example, for the x86_64 architecture on Amazon Linux 2:

```
gpg --verify amazon-ssm-agent.rpm.sig amazon-ssm-agent.rpm
```

This command returns output similar to the following:

```
gpg: Signature made Sat 08 Feb 2025 12:05:08 AM UTC using RSA key ID D0052E5D
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: 4855 A9E6 8332 16D6 A77D  8FE4 51A8 E050 D005 2E5D
```

If the output includes the phrase `BAD signature`, check whether you performed the procedure correctly. If you continue to get this response, contact 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 (v3.3.1802.0 and later)

1. Copy the following public key and save it to a file named `amazon-ssm-agent.gpg`.

> **Important**
>
> The following public key expires on 2026-07-15 (July 15, 2026). Systems Manager will publish a new public key in this topic before the old one expires. We encourage you to subscribe to the RSS feed for this topic to get a notification when the new key is available.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)
```

```
mQINBGeRNq4BEACrlf5h6Pz+k+M+QCJJ2LfK7d2Tn9J8iJ9qBK2Vwvuxco1rpSO+
KEI3nTeysPuheximps8WOCADX4VlbsKxMZQLjQM4mA26m1Tiw9nAI4kod4bKjiuM
BMUTCD1wfnjH3zQi4kDUdbpfAEMiPgNLVLH85Wf+lhK+Zm+V38DYzLyVj03kX4wK
iG6RMoxzOBZa5gNsVq+j+oCUITGz/URxH713Rgo8WeoEegI0+7iCBLKg+PM0b7GV
2nzkwWJz796HdkqSg8BwXsYaLTrHxa2P1IpwPCisAkyO7gZaMd6Uj69dtMFO+V8a
Qee6b57qGuFKZw7h1Vvc85PbF1Gy/wNIpary57kUHBFUg1vYep/roJuEbJCq97r5
I2liLl4NAyrWb9r/TAVxlXvqM4iZUhxm8GAp0FywMdBr9ZECClKa5HxuVmlm0Wgl
TXoYTOZKeDg6ZoCvyhNxWneCNip74fohXymeFF5L/budhBwy5wuwSniOgTGLo/4C
VgZHWCcN+d0Q3bx/sl2QNqPg5/xzsxEtymXLdVdwLIsLdEQUnIvy8KTs5jol3Dwi
nnEEyhly6wdaw+qDOhkSOT/VnErrSMkYF8VJfa5GjhCBWKw9JVSkaP2CI/VHOgHM
MKROnulq0hRQBR7RmLYt98xu38BHJWMmF8Ga/HJuIxzD1VmkZOPvDDESUwARAQAB
tCdTU00gQWdlbnQgPHNzbS1hZ2VudC1zaWduZXJAYW1hem9uLmNvbT6JAj8EEwEC
ACkFAmeRNq4CGy8FCQLGmIAHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIXgAAKCRBR
qOBQ0AUuXTdND/9qldQ1E3dYjBVXOnbhiUQL594bkS5VoEX7D4fZ5UMVZa5pGiz+
husnoRUS9rH1cSeq7aHJu9hSCMuMdvRpuoo0CwLB+7HtzJvAO2M01hcEkUYa6Qdj
njTzP0ZjnoenJmqF9SYmVqAI/VPa9mNQ1OJ+HQ3qh5i6w+FoWlVqEdXjZGrWijub
TqyN33i1Y26t7Os/x8I9fUeNx37y/7Kama8LTdtv9GhWiMVBg2IuVf27HCMYofrQ
m2uCGe61IhtsnhsYaYupmljl+6qgdiuCiS9BAsoIGtqTnu8lnKcGyGz6YnRszN+U
1bNE4w+UFpXWJF8ogpYcghJ06aW/LhjZnQSx3VliLdW8eOJzou41yWmiuL3ZY8eW
KAlD+7eYKS6N6fEJCeNO2VX2lcKtDfaOX+lqGIVyexKayMfpi+0frNzt/92YCpF5
3jkeS77vMMVqKIUiIp1OCGv3XsFpIr6Bt2c2throYPDoQL3zvq6vvG40BKeRQ4tT
Y+5vTc8MeNn3LdzTl9pusxTcKifrJq7f5FIsL2CpAX8uQ+Qz+XWsYQQ5PvyUDtOz
nU/MRZaP6HnqY42bzI9ZlKgXi9IE3MXIwoET9YyzFjkIDvat7SlB4uJCpeIqp/KM
OIrTMb7paGLYmBU6YqxNBkDWItNG7NeZzyhh/R/Qqb4vJaf4S+ZqD1RZXokCHAQQ
AQIABgUCZ5E2rwAKCRB90Jej2tf1/CdnD/46It+RNoE00TesZK5n2bijH5Eljw0E
4/UpMi1SV6t2zY7lIm7TcKNn18tynJNFqB6YXXOwSbBG/fbN2E9RaoUCZw23TmAv
amuHwrfsDqsHb7zzPF0bISYjqEDLQJj/gtEugUc6XY1dEpFSlWJIOvgryG04cFXI
uD2KY87ya4s1R+sEVAJ14K4RlUCiMmzJdR0NJNYJOwBi1gkLEp6jG86ttiG2U7fY
pE2ibV+c0GeIFq8PIzqqENsn9KBuRH5EcbdBwfnsj2XfM4aR3ZtRIdWXkKkdP9Rs
yU5dTF/Y7XPId5h8/gp00+DMlXFBinQ1jE7A7eDYviEFd1ba8P7dIom3Q3gzKiWu
KTGpnykShs5NvpQmvGUF6JqDHI4RK9s3kLqsNyZkhenJfRBrJ/45fQAuP4CRedkF
7PSfX0Xp7kDnKuyK6wEUEfXXrqmuLGDmigTXblO5qgdyMwkOLjiY9znBZbHoKs76
VplOoNgGnN19i3nuMcPf2npFICJv7kTIyn5Fh7pjWDCahl3U/PwoLjrrlEzpyStU
oXSZrK3kiAADEdSODXJl8KYU0Pb27JbRr1ZbWnxb+O39TOhtssstulkR0v+IDGDQ
rQE1b12sKgcNFSzInzWrNGu4S06WN8DYzlrTZ9aSHj+37ZqpXAevi8WOFXKPV3PA
E6+O8RI2451Dcg==
=aDkv
-----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. We recommend that you use GPG to verify the fingerprint even if you use RPM to verify the installer package.

```
rpm -qa gpg-pubkey --qf '%{Description}' | gpg --with-fingerprint | grep -A 1
 "ssm-agent-signer@amazon.com"
```

This command returns output similar to the following:

```
pub  4096R/D0052E5D 2025-01-22 SSM Agent <ssm-agent-signer@amazon.com>
      Key fingerprint = 4855 A9E6 8332 16D6 A77D  8FE4 51A8 E050 D005 2E5D
```

The fingerprint should match the following.

```
4855 A9E6 8332 16D6 A77D 8FE4 51A8 E050 D005 2E5D
```

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 *agent-download-filename* with the values you specified when downloading the agent, as listed in the table earlier in this topic.

```
rpm --checksig agent-download-filename
```

For example, for the x86_64 architecture on Amazon Linux 2:

```
rpm --checksig amazon-ssm-agent.rpm
```

This command returns output similar to the following.

```
amazon-ssm-agent.rpm: rsa sha1 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 Support and don't install the agent.

Verifying the SSM Agent package on a Linux server (v3.3.1611.0 and earlier)

Before you begin

The procedures for **GPG** and **RPM** in this section apply to SSM Agent version 3.3.1611.0 and earlier versions. We recommend always using the latest version of the agent. For information, see Verifying the SSM Agent package on a Linux server (v3.3.1802.0 and later). However, if you have a specific reason to continue using agent version 3.3.1611.0 or earlier, follow the instructions in one of the following procedures.

GPG

To verify the SSM Agent package on a Linux server (v3.3.1611.0 and earlier)

1. Copy the following public keys and save it to a file named `amazon-ssm-agent.gpg`.

> **Important**
>
> The public key shown below expired on 2025-02-17 (February 17, 2025) and works for Version 3.3.1611.0 and earlier versions up to 3.2.1542.0, and only if it was used previously to verify the agent's signature. Systems Manager will publish a new public key in this topic before the old one expires. We encourage you to subscribe to the RSS feed for this topic to get a notification when the new key is available.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQENBGTtIoIBCAD2M1aoGIE0FXynAHM/jtuvdAVVaX3Q4ZejTqrX+Jq8ElAMhxyO
GzHu2CDtCYxtVxXK3unptLVt2kGgJwNbhYC393jDeZx5dCda4Nk2YXX1UK3P461i
axuuXRzMYvfM4RZn+7bJTu635tA07q9Xm6MGD4TCTvsjBfViOxbrxOg5ozWbJdSw
fSR8MwUrRfmFpAefRlYfCEuZ8FHywa9U6jLeWt2O/kqrZliJOAGjGzXtB7EZkqKb
faCCxikjjvhF1awdEqSK4DQorC/OvQc4I5kP5y2CJbtXvXO73QH2yE75JMDIIx9x
rOsIRUoSfK3UrWaOVuAnEEn5ueKzZNqGG1J1ABEBAAG0J1NTTSBBZ2VudCA8c3Nt
LWFnZW50LXNpZ25lckBhbWF6b24uY29tPokBPwQTAQIAKQUCZO0iggIbLwUJAsaY
gAcLCQgHAwIBBhUIAgkKCwQWAgMBAh4BAheAAAoJELwfSVyX3QTt+icH/A//tJsW
I+7Ay8FGJh8dJPNy++HIBjVSFdGNJFWNbw1Z8uZcazHEcUCH3FhW4CLQLTZ3OVPz
qvFwzDtRDVIN/Y9EGDhLMFvimrE+/z4olWsJ5DANf6BnX8I5UNIcRt5d8SWH1BEJ
2FWIBZFgKyTDI6XzRC5x4ahtgpOVAGeeKDehs+wh6Ga4W0/K4GsviP1Kyr+Ic2br
NAIq0q0IHyN1q9zam3Y0+jKwEuNmTj+Bjyzshyv/X8S0JWWoXJhkexkOvWeBYNNt
5wI4QcSteyfIzp6KlQF8q11Hzz9D9WaPfcBEYyhq7vLEARobkbQMBzpkmaZua241
```

```
0RaWG50HRvrgm4aJAhwEEAECAAYFAmTtIoMACgkQfdCXo9rX9fwwqBAAzkTgYJ38
sWgxpn7Ux/81F2BWR1sVkmP79i++fXyJlKI8xtcJFQZhzeUos69KBUCy7mgx5bYU
P7NA5o9DUbwz/QS0i1Cqm4+jtFlX0MXe4FikXcqfDPnnzN8mVB2H+fa43iHR1PuH
GgUWuNdxzSoIYRmLZXWmeN5YXPcmixlhLzcE2TOQn1mOKcu2fKdLtBQ8KiEkmjiu
naoLxnUcyk1zMhaha+LzEkQdOyasix0ggylN2ViWVnlmfy0niuXDxW0qZWPdLStF
OODiX3iqGmkH3rDfy6nvxxBR4GIs+MGD72fpWzzrINDgkGI2i2t1+0AX/mps3aTy
+ftlgrim8stYWB58XXDAb0vad06sNye5/zDzfr0I9HupJrTzFhaYJQjWPaSlINto
LDJnBXohiUIPRYRcy/k012oFHDWZHT3H6CyjK9UD5UlxA9H7dsJurANs6FOVRe+7
34uJyxDZ/W7zLG4AVG0zxibrUSoaJxwcOjVPVsQAlrwG/GTs7tcAccsJqbJ1Py/w
9AgJl8VU2qc8POsHNXk348gjP7C8PDnGMpZFzr9f5INctRushpiv7onX+aWJVX7T
n2uX/TP3LCyH/MsrNJrJOQnMYFRLQitciP0E+F+eA3v9CY6mDuyb8JSx5HuGGUsG
S4bKBOcA8vimEpwPoT8CE7fdsZ3Qkwdu+pw=
=zr5w
-----END PGP PUBLIC KEY BLOCK-----
```

2. Import the public key into your keyring, and note the returned key value.

```
gpg --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/97DD04ED 2023-08-28 [expired: 2025-02-17]
      Key fingerprint = DE92 C7DA 3E56 E923 31D6 2A36 BC1F 495C 97DD 04ED
uid                  SSM Agent <ssm-agent-signer@amazon.com>
```

The fingerprint should match the following.

```
DE92 C7DA 3E56 E923 31D6 2A36 BC1F 495C 97DD 04ED
```

If the fingerprint doesn't match, don't install the agent. Contact AWS Support.

4. Download the signature file according to your instance's architecture and operating system if you haven't already done so.

5. 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, as listed in the table earlier in this topic.

```
gpg --verify signature-filename agent-download-filename
```

For example, for the x86_64 architecture on Amazon Linux 2:

```
gpg --verify amazon-ssm-agent.rpm.sig amazon-ssm-agent.rpm
```

This command returns output similar to the following:

```
gpg: Signature made Fri 10 Jan 2025 01:54:18 AM UTC using RSA key ID 97DD04ED
gpg: Good signature from "SSM Agent <ssm-agent-signer@amazon.com>"
gpg: Note: This key has expired!
Primary key fingerprint: DE92 C7DA 3E56 E923 31D6  2A36 BC1F 495C 97DD 04ED
```

If the output includes the phrase `BAD signature`, check whether you performed the procedure correctly. If you continue to get this response, contact 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 (v3.3.1611.0 and earlier)

1. Copy the following public key and save it to a file named `amazon-ssm-agent.gpg`.

> **Important**
>
> The public key shown below expired on 2025-02-17 (February 17, 2025) and works for Version 3.3.1611.0 and earlier versions up to 3.2.1542.0, and only if it was used previously to verify the agent's signature. Systems Manager will publish a new public key in this topic before the old one expires. We encourage you to subscribe to the RSS feed for this topic to get a notification when the new key is available.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQENBGTtIoIBCAD2M1aoGIE0FXynAHM/jtuvdAVVaX3Q4ZejTqrX+Jq8ElAMhxyO
GzHu2CDtCYxtVxXK3unptLVt2kGgJwNbhYC393jDeZx5dCda4Nk2YXX1UK3P461i
axuuXRzMYvfM4RZn+7bJTu635tA07q9Xm6MGD4TCTvsjBfViOxbrxOg5ozWbJdSw
fSR8MwUrRfmFpAefRlYfCEuZ8FHywa9U6jLeWt2O/kqrZliJOAGjGzXtB7EZkqKb
faCCxikjjvhF1awdEqSK4DQorC/OvQc4I5kP5y2CJbtXvXO73QH2yE75JMDIIx9x
rOsIRUoSfK3UrWaOVuAnEEn5ueKzZNqGG1J1ABEBAAG0J1NTTSBBZ2VudCA8c3Nt
LWFnZW50LXNpZ25lckBhbWF6b24uY29tPokBPwQTAQIAKQUCZO0iggIbLwUJAsaY
gAcLCQgHAwIBBhUIAgkKCwQWAgMBAh4BAheAAAoJELwfSVyX3QTt+icH/A//tJsW
I+7Ay8FGJh8dJPNy++HIBjVSFdGNJFWNbw1Z8uZcazHEcUCH3FhW4CLQLTZ3OVPz
qvFwzDtRDVIN/Y9EGDhLMFvimrE+/z4olWsJ5DANf6BnX8I5UNIcRt5d8SWH1BEJ
2FWIBZFgKyTDI6XzRC5x4ahtgpOVAGeeKDehs+wh6Ga4W0/K4GsviP1Kyr+Ic2br
NAIq0q0IHyN1q9zam3Y0+jKwEuNmTj+Bjyzshyv/X8S0JWWoXJhkexkOvWeBYNNt
5wI4QcSteyfIzp6KlQF8q11Hzz9D9WaPfcBEYyhq7vLEARobkbQMBzpkmaZua241
0RaWG50HRvrgm4aJAhwEEAECAAYFAmTtIoMACgkQfdCXo9rX9fwwqBAAzkTgYJ38
sWgxpn7Ux/81F2BWR1sVkmP79i++fXyJlKI8xtcJFQZhzeUos69KBUCy7mgx5bYU
P7NA5o9DUbwz/QS0i1Cqm4+jtFlX0MXe4FikXcqfDPnnzN8mVB2H+fa43iHR1PuH
GgUWuNdxzSoIYRmLZXWmeN5YXPcmixlhLzcE2TOQn1mOKcu2fKdLtBQ8KiEkmjiu
naoLxnUcyk1zMhaha+LzEkQdOyasix0ggylN2ViWVnlmfy0niuXDxW0qZWPdLStF
OODiX3iqGmkH3rDfy6nvxxBR4GIs+MGD72fpWzzrINDgkGI2i2t1+0AX/mps3aTy
+ftlgrim8stYWB58XXDAb0vad06sNye5/zDzfr0I9HupJrTzFhaYJQjWPaSlINto
LDJnBXohiUIPRYRcy/k012oFHDWZHT3H6CyjK9UD5UlxA9H7dsJurANs6FOVRe+7
34uJyxDZ/W7zLG4AVG0zxibrUSoaJxwcOjVPVsQAlrwG/GTs7tcAccsJqbJ1Py/w
9AgJl8VU2qc8POsHNXk348gjP7C8PDnGMpZFzr9f5INctRushpiv7onX+aWJVX7T
n2uX/TP3LCyH/MsrNJrJOQnMYFRLQitciP0E+F+eA3v9CY6mDuyb8JSx5HuGGUsG
S4bKBOcA8vimEpwPoT8CE7fdsZ3Qkwdu+pw=
=zr5w
-----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. We recommend that you use GPG to verify the fingerprint even if you use RPM to verify the installer package.

```
rpm -qa gpg-pubkey --qf '%{Description}' | gpg --with-fingerprint | grep -A 1
 "ssm-agent-signer@amazon.com"
```

This command returns output similar to the following:

```
pub  2048R/97DD04ED 2023-08-28 SSM Agent <ssm-agent-signer@amazon.com>
      Key fingerprint = DE92 C7DA 3E56 E923 31D6 2A36 BC1F 495C 97DD 04ED
```

The fingerprint should match the following.

```
DE92 C7DA 3E56 E923 31D6 2A36 BC1F 495C 97DD 04ED
```

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 *agent-download-filename* with the values you specified when downloading the agent, as listed in the table earlier in this topic.

```
rpm --checksig agent-download-filename
```

For example, for the x86_64 architecture on Amazon Linux 2:

```
rpm --checksig amazon-ssm-agent.rpm
```

This command returns output similar to the following.

```
amazon-ssm-agent.rpm: rsa sha1 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 Support and don't install the agent.

Manually installing and uninstalling 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.

Installation on other machine types

The procedures in this section are designed specifically for Amazon EC2 instances. For on-premises servers, virtual machines, or other non-EC2 environments, use the `ssm-setup-cli` tool as described in How to install the SSM Agent on hybrid Linux nodes.
Using EC2 installation procedures on non-EC2 systems can potentially result in security vulnerabilities. The `ssm-setup-cli` tool provides additional security protections for non-EC2 machines.

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. The commands provided in these procedures can also be passed to Amazon EC2 instances as scripts through user data.

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 Find AMIs with the SSM Agent preinstalled.

Keeping the agent up to date

An updated version of SSM Agent is released whenever new tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. 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 for Systems Manager.

- AlmaLinux
- Amazon Linux 2 and Amazon Linux 2023
- Debian Server
- Oracle Linux
- Red Hat Enterprise Linux
- Rocky Linux
- Ubuntu Server

Uninstalling SSM Agent from Linux instances

Use the package manager for your operating system to uninstall SSM Agent from Linux instances. Depending on the operating system, the uninstall command will be similar to the following example command:

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

Manually install SSM Agent on AlmaLinux instances

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

Before you begin

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

- Ensure that Python 3 is installed on your AlmaLinux instance. This is required in order for SSM Agent to work properly.

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see Manually installing and uninstalling SSM Agent on EC2 instances for Linux.

Topics

- Quick installation commands for SSM Agent on AlmaLinux
- Create custom agent installation commands for AlmaLinux in your Region

Quick installation commands for SSM Agent on AlmaLinux

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 AlmaLinux instance, note the following:

- Ensure that Python 3 is installed on your AlmaLinux instance. This is required in order for SSM Agent to work properly.

To install SSM Agent on AlmaLinux

1. Connect to your AlmaLinux 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 AlmaLinux.

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; vendo>
   Active: active (running) since Tue 2025-04-19 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; vendo>
   Active: inactive (dead) since Tue 2025-04-19 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 AlmaLinux 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 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 AlmaLinux 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 Amazon Linux 2 and Amazon Linux 2023 instances

In most cases, the Amazon Machine Images (AMIs) for Amazon Linux 2 and Amazon Linux 2023 that are provided by AWS come with AWS Systems Manager Agent (SSM Agent) preinstalled by default. For more information, see Find AMIs with the SSM Agent preinstalled.

In the event that SSM Agent isn't preinstalled on a new Amazon Linux 2 or Amazon Linux 2023 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 or Amazon Linux 2023 instance, note the following:

- For important information that applies to installation of SSM Agent on all Linux-based operating systems, see Manually installing and uninstalling SSM Agent on EC2 instances for Linux.
- 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 or Amazon Linux 2023
- Create custom agent installation commands for Amazon Linux 2 or Amazon Linux 2023 in your Region

Quick installation commands for SSM Agent on Amazon Linux 2 or Amazon Linux 2023

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 or Amazon Linux 2023 using quick copy and paste commands

1. Connect to your Amazon Linux 2 or Amazon Linux 2023 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 and Amazon Linux 2023.

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 or Amazon Linux 2023 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 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 2 or Amazon Linux 2023 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 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 Find AMIs with the SSM Agent preinstalled.

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 and uninstalling SSM Agent on EC2 instances for Linux.

Topics

- Quick installation commands for SSM Agent on Debian Server
- Create custom agent installation commands for Debian Server in your Region

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

1. Connect to your Debian 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. Run the following command to change to the temporary directory.

```
cd /tmp/ssm
```

4. 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 2025-04-19 16:25:03 UTC; 4s ago
 Main PID: 628 (amazon-ssm-agen)
   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 2025-04-19 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 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 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/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

```
wget 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.

```
wget 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 Find AMIs with the SSM Agent preinstalled.

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 and uninstalling SSM Agent on EC2 instances for Linux.
- 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
- Create custom agent installation commands for Oracle Linux in your Region

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 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 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 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 Find AMIs with the SSM Agent preinstalled.

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 and uninstalling SSM Agent on EC2 instances for Linux.
- 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, 9.x, and 10.x
- Install SSM Agent on RHEL 7.x

Install SSM Agent on RHEL 8.x, 9.x, and 10.x

The Amazon Machine Images (AMIs) for RHEL 8 and 9 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 and 9 instances.

Before you begin

Before you install SSM Agent on a RHEL 8, 9, or 10 instance, note the following:

- Ensure that either Python 2 or Python 3 is installed on your RHEL 8, 9, or 10 instance. This is required in order for SSM Agent to work properly.

Topics

- Quick installation commands for SSM Agent on RHEL 8, 9, and 10
- Create custom agent installation commands for RHEL 8, 9, and 10 in your Region

Quick installation commands for SSM Agent on RHEL 8, 9, and 10

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, 9.x, or 10.x

1. Connect to your RHEL 8, 9, or 10 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, 9, and 10.

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; vendo>
   Active: active (running) since Tue 2025-04-19 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; vendo>
   Active: inactive (dead) since Tue 2025-04-19 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, 9, and 10 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 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, 9, and 10 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 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
- Create custom agent installation commands for RHEL 7 in your Region

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 2025-04-19 16:47:36 UTC; 22s ago
 Main PID: 1342 (amazon-ssm-agen)
   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 2025-04-19 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 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 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 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 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 Find AMIs with the SSM Agent preinstalled.

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 and uninstalling SSM Agent on EC2 instances for Linux.

Topics

- Quick installation commands for SSM Agent on Rocky Linux
- Create custom agent installation commands for Rocky Linux in your Region

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; vendo>
   Active: active (running) since Tue 2025-04-19 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; vendo>
   Active: inactive (dead) since Tue 2025-04-19 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 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 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 Rocky Linux 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 Ubuntu Server instances

Important

Before you install SSM Agent on a 64-bit version of Ubuntu Server, ensure 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.

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 Find AMIs with the SSM Agent preinstalled.

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 and uninstalling SSM Agent on EC2 instances for Linux.

Topics

- Install SSM Agent on Ubuntu Server 16.04 LTS 64-bit (Snap), 18.04, 20.04, 22.04 LTS, 23.10, 24.04 LTS, 24.0, and 25.04
- Install SSM Agent on Ubuntu Server 16.04 64-bit (deb)
- Install SSM Agent on Ubuntu Server 16.04 32-bit
- Determining the correct SSM Agent version to install on 64-bit Ubuntu Server 16.04 instances

Install SSM Agent on Ubuntu Server 16.04 LTS 64-bit (Snap), 18.04, 20.04, 22.04 LTS, 23.10, 24.04 LTS, 24.0, and 25.04

Before you begin

Before you install SSM Agent on an Ubuntu Server 16.04 LTS 64-bit (Snap), 18.04, 20.04, 22.04 LTS, 23.10, 24.04 LTS, 24.0, and 25.04, 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 16.04 LTS 64-bit (Snap), 18.04, 20.04, 22.04 LTS, 23.10, 24.04 LTS, 24.0, and 25.04, SSM Agent installer files, including agent binaries and config files, are stored in the following directory: `/snap/amazon-ssm-agent/current/`. If you make changes to any configuration files in this directory, then you must copy these files from the `/snap` 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 20.04, 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 18.04, 20.04, 22.04 LTS, 23.10, 24.04 LTS, 24.0, and 25.04, 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 using the instructions for installing SSM Agent on Ubuntu Server 16.04 (deb). Before installing, 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 16.04 LTS 64-bit (Snap), 18.04, 20.04, 22.04 LTS, 23.10, 24.04 LTS, 24.0, and 25.04 (with Snap package)

1. SSM Agent is installed, by default, on Ubuntu Server 16.04 LTS 64-bit (Snap), 18.04, 20.04, 22.04 LTS, 23.10, 24.04 LTS, 24.0, and 25.04 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 64-bit (deb)

> **Important**
>
> Before you install SSM Agent on a 64-bit version of Ubuntu Server, ensure 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.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 Find AMIs with the SSM Agent preinstalled.

In the event that SSM Agent isn't preinstalled on a new Ubuntu Server 16.04 instance prior to version 20180627 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 (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 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 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 the following command to determine if SSM Agent is running.

   Ubuntu Server 16.04

   ```
   sudo systemctl 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 the following command 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
   ```

Create custom installation commands for SSM Agent on Ubuntu Server 16.04 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 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 (deb) 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 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 Find AMIs with the SSM Agent preinstalled.

In the event that SSM Agent isn't preinstalled on a new Ubuntu Server 16.04 instance 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 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 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 URL in the following command include an `ec2-downloads-windows` directory, this is the correct global installation file for Ubuntu Server 16.04 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 the following command to determine if SSM Agent is running.

   Ubuntu Server 16.04

   ```
   sudo systemctl 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 the following command 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
```

Create custom installation commands for SSM Agent on Ubuntu Server 16.04 32-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 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 32-bit (deb) 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 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
>
> 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 **Details** tab, check for a YYYYMMDD identifier in the value under **AMI name** 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"
        }
    ]
}
```

Configuring SSM Agent to use a proxy on Linux nodes

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. If you intend to use Session Manager, note that HTTPS proxy servers aren't supported.

Topics

- Configure SSM Agent to use a proxy (upstart)
- Configure SSM Agent to use a proxy (systemd)

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=http://hostname:port
env no_proxy=IP address for instance metadata services (IMDS)
```

For an HTTPS proxy server:

```
env http_proxy=http://hostname:port
env https_proxy=https://hostname:port
env no_proxy=IP address for instance metadata services (IMDS)
```

> **Important**
>
> Add the `no_proxy` setting to the file and specify the IP address. The IP address for `no_proxy` is the instance metadata services (IMDS) endpoint for Systems Manager. If you don't specify `no_proxy`, calls to Systems Manager take on the identity from the proxy service (if IMDSv1 fallback is enabled) or calls to Systems Manager fail (if IMDSv2 is enforced).
>
> - For IPv4, specify `no_proxy=169.254.169.254`.
> - For IPv6, specify `no_proxy=[fd00:ec2::254]`. The IPv6 address of the instance metadata service is compatible with IMDSv2 commands. The IPv6 address is only accessible on instances built on the AWS Nitro System. For more information, see How Instance Metadata Service Version 2 works in the *Amazon EC2 User Guide*.

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 service stop amazon-ssm-agent
sudo service 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.

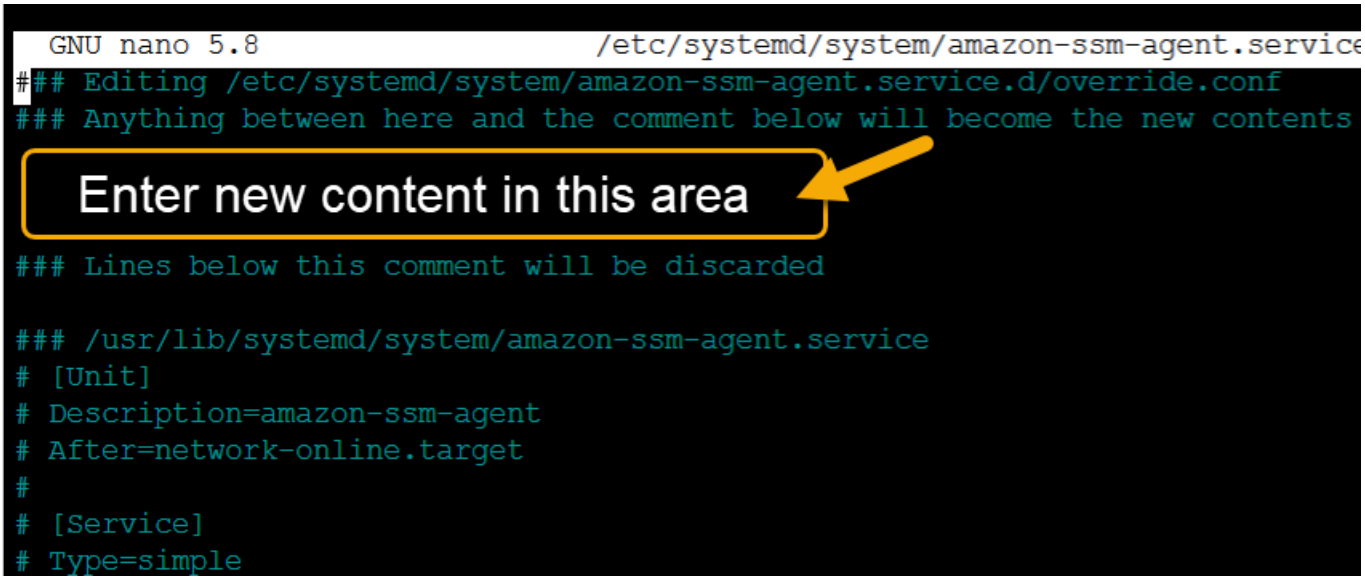
> **Note**
>
> Some of the steps in this procedure contain explicit instructions for Ubuntu Server instances where SSM Agent was installed using Snap.

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=http://hostname:port"
Environment="no_proxy=IP address for instance metadata services (IMDS)"
```

For an HTTPS proxy server:

```
[Service]
Environment="http_proxy=http://hostname:port"
Environment="https_proxy=https://hostname:port"
Environment="no_proxy=IP address for instance metadata services (IMDS)"
```

Important

Add the `no_proxy` setting to the file and specify the IP address. The IP address for `no_proxy` is the instance metadata services (IMDS) endpoint for Systems Manager. If you don't specify `no_proxy`, calls to Systems Manager take on the identity from the proxy service (if IMDSv1 fallback is enabled) or calls to Systems Manager fail (if IMDSv2 is enforced).

- For IPv4, specify `no_proxy=169.254.169.254`.
- For IPv6, specify `no_proxy=[fd00:ec2::254]`. The IPv6 address of the instance metadata service is compatible with IMDSv2 commands. The IPv6 address is only

accessible on instances built on the AWS Nitro System. For more information, see How Instance Metadata Service Version 2 works in the *Amazon EC2 User Guide*.

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, Amazon Linux 2023, and RHEL 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*.

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 EC2 instances and Amazon Machine Images supported by Systems Manager. For more information, see the list of supported operating systems for macOS.

  SSM Agent doesn't need to be manually installed on macOS EC2 instances unless it has been uninstalled.
- EC2 instances for macOS are not supported in all AWS Regions. For lists of Regions where x86-based and M1 EC2 instances for macOS are supported, see macOS workloads in the Amazon EC2 FAQs.
- An updated version of SSM Agent is released whenever new tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. Subscribe to the SSM Agent Release Notes page on GitHub to get notifications about SSM Agent updates.

Topics

- Manually installing and uninstalling SSM Agent on EC2 instances for macOS

Manually installing and uninstalling 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. The commands provided in this procedure can also be passed to Amazon EC2 instances as scripts through user data.

Before you begin

Install `wget` using Homebrew.

To install SSM Agent on macOS

1. Download the agent installer file for x86_64 instances using the following command.

   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*.

   ```
   sudo wget https://s3.region.amazonaws.com/amazon-ssm-region/latest/darwin_amd64/
   amazon-ssm-agent.pkg
   ```

   For Apple silicon instances use the following command.

   ```
   sudo wget https://s3.region.amazonaws.com/amazon-ssm-region/latest/darwin_arm64/
   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 tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. Subscribe to the SSM Agent Release Notes page on GitHub to get notifications about SSM Agent updates.

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 Amazon Machine Images (AMIs) for Windows Server that are provided by AWS. Support is provided for the following operating system (OS) versions.

- Windows Server 2012 R2 AMIs published in November 2016 or later
- Windows Server 2016, 2019, 2022 (excluding Nano versions), and 2025

Support notes for previous versions

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 non-EC2 machines that are configured for Systems Manager in a hybrid and multicloud environment.

Keeping SSM Agent up to date

An updated version of SSM Agent is released whenever new tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. 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 and uninstalling SSM Agent on EC2 instances for Windows Server
- Configure SSM Agent to use a proxy for Windows Server instances

Manually installing and uninstalling 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 provided by Amazon:

- Windows Server 2012 R2 AMIs published in November 2016 or later
- Windows Server 2016, 2019, 2022 (excluding Nano versions), and 2025

Install SSM Agent on EC2 instances for Windows Server

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. The commands provided in this procedure can also be passed to Amazon EC2 instances as scripts through user data.

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.

> ⚠ **Installation on other machine types**
>
> This procedure in this topic applies specifically to installing or reinstalling SSM Agent on an EC2 instance for Windows Server. For on-premises servers, virtual machines, or other non-EC2 environments, use the `ssm-setup-cli` tool as described in the section called "Install SSM Agent on hybrid Windows Server nodes".
>
> Using EC2 installation procedures on non-EC2 systems can potentially result in security vulnerabilities. The `ssm-setup-cli` tool provides additional security protections for non-EC2 machines.

To manually install the latest version of SSM Agent on EC2 instances for Windows Server

1. Connect to your instance by using Remote Desktop or Windows PowerShell. For more information, see Connect to your instance in the *Amazon EC2 User Guide*.

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

```
[System.Net.ServicePointManager]::SecurityProtocol = 'TLS12'
$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

```
[System.Net.ServicePointManager]::SecurityProtocol = 'TLS12'
$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" `
    -Wait
```

```
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

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
```

> **Note**
>
> To uninstall the SSM Agent from a Windows Server instance, open **Control Panel**, **Programs**. Choose the **Uninstall a program** option. Open the context (right-click) menu for **Amazon SSM Agent** and choose **Uninstall**.

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 you intend to use Session Manager, note that HTTPS proxy servers aren't supported.

Before you begin

Before you configure SSM Agent to use a proxy, note the following important information.

In the following procedure, you run a command to configure SSM Agent to use a proxy. The command includes a `no_proxy` setting with an IP address. The IP address is the instance metadata

services (IMDS) endpoint for Systems Manager. If you don't specify `no_proxy`, calls to Systems Manager take on the identity from the proxy service (if IMDSv1 fallback is enabled) or calls to Systems Manager fail (if IMDSv2 is enforced).

- For IPv4, specify `no_proxy=169.254.169.254`.
- For IPv6, specify `no_proxy=[fd00:ec2::254]`. The IPv6 address of the instance metadata service is compatible with IMDSv2 commands. The IPv6 address is only accessible on instances built on the AWS Nitro System. For more information, see How Instance Metadata Service Version 2 works in the *Amazon EC2 User Guide*.

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=IP address for instance metadata services (IMDS)")

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.

```
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: IP address for instance metadata services (IMDS)
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 intended settings. The agent uses the first proxy settings it finds.

> **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 (`HKLM:\SYSTEM\CurrentControlSet\Services\AmazonSSMAgent`)
2. `System` environment variables (`http_proxy`, `https_proxy`, `no_proxy`)

3. `LocalSystem` user account environment variables `http_proxy`, `https_proxy`, `no_proxy`)
4. Browser 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 tools, 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 AWS Systems Manager Run Command.

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.

Operating system	Command to check SSM Agent status	Command to start SSM Agent

Amazon Linux 2and Amazon Linux 2023	`sudo systemctl status amazon-ssm-agent`	`sudo systemctl enable amazon-ssm-agent`
		
		`sudo systemctl start amazon-ssm-agent`

Debian Server11 and 12	`sudo systemctl status amazon-ssm-agent`	`sudo systemctl enable amazon-ssm-agent`
		
		`sudo systemctl start amazon-ssm-agent`

macOS	Check the agent log file at `/var/log/amazon/ssm/amazon-ssm-agent.log`	`sudo launchctl load -w /Library/LaunchDaemons/com.amazon.aws.ssm.plist`
		
		`sudo launchctl start com.amazon.aws.ssm`

Oracle Linux	`sudo systemctl status amazon-ssm-agent`	`sudo systemctl enable amazon-ssm-agent`
		
		`sudo systemctl start amazon-ssm-agent`

Red Hat Enterprise Linux (RHEL) 7.x, 8.x, 9.x, and 10.x	`sudo systemctl status amazon-ssm-agent`	`sudo systemctl enable amazon-ssm-agent`

Operating system	Command to check SSM Agent status	Command to start SSM Agent
		`sudo systemctl start amazon-ssm-agent`
Ubuntu Server 16.04 (32-bit)	`sudo status amazon-ssm-agent`	`sudo start amazon-ssm-agent`

Ubuntu Server 16.04 64-bit instances (deb package installation)	`sudo systemctl status amazon-ssm-agent`	`sudo systemctl enable amazon-ssm-agent`
		
		`sudo systemctl start amazon-ssm-agent`

Ubuntu Server 16.04 LTS, 18.04, 20.04, 22.04 LTS, 23.10, 24.04 LTS, 24.0, and 25.04	`sudo systemctl status snap.amazon-ssm-agent.amazon-ssm-agent.service`	`sudo snap start amazon-ssm-agent`
Windows Server	*Run in PowerShell:* `Get-Service AmazonSSMAgent`	*Run in PowerShell Administrator mode:* `Start-Service AmazonSSMAgent`

More info

- Working with SSM Agent on EC2 instances for Linux
- Working with SSM Agent on EC2 instances for Windows Server
- Checking the SSM Agent version number

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.

> **Note**
>
> To view details about earlier versions, see the SSM Agent release notes on GitHub.

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**.
3. In the **SSM Agent version** column, note the **Agent version** number.

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 2 and Amazon Linux 2023

> **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
```

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/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 7, 8, 9, and 10.

   ```
   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
```

Run the following command for the DNF package utility.

```
dnf info amazon-ssm-agent
```

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.

1. Log in to your managed node.
2. Run the following command for Ubuntu Server 16.04 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/now 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 16.04 LTS 64-bit (Snap), 18.04, 20.04, 22.04 LTS, 23.10, 24.04 LTS, 24.0, and 25.04.

   ```
   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.

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 Logging and monitoring in AWS Systems Manager.

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.

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 tool in 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.
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 20.04, 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 20.04, 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.
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\{{EXECUTABLENAME}}.log"`

   Change this entry to use the following path.

   `filename="`**`C:\ProgramData`**`\Amazon\SSM\Logs\{{EXECUTABLENAME}}.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 and other machine types in hybrid and multicloud environments 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, 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 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 servers, 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
- Restrict session access based on instance tags

Automating updates to SSM Agent

AWS releases a new version of AWS Systems Manager Agent (SSM Agent) when we add or update Systems Manager tools. If your managed nodes use an older version of the agent, then you can't use the new tools or benefit from the updated tools. For these reasons, we recommend that you automate the process of updating SSM Agent on your managed nodes using any of the following methods.

Agent updates on the Bottlerocket operating system

SSM Agent on the Bottlerocket operating system can't be updated using the Systems Manager Command document `AWS-UpdateSSMAgent`. Updates are managed within the Bottlerocket control container. For more information, see Bottlerocket Control Container and Bottlerocket update operator on GitHub.

macOS version requirement

If an instance is running macOS version 11.0 (Big Sur) or later, the instance must have the SSM Agent version 3.1.941.0 or higher to run the AWS-UpdateSSMAgent document. If the instance is running a version of SSM Agent released before 3.1.941.0, update your SSM Agent to run the AWS-UpdateSSMAgent by running `brew update` and `brew upgrade amazon-ssm-agent` commands.

Method	Details
One-click automated update on all managed nodes (Recommended)	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 **Auto update SSM Agent** on the **Settings** tab in Fleet Manager, as described later in this topic.
Global or selective update	You can use State Manager, a tool in 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 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 Walkthrough: Automatically update SSM Agent with the AWS CLI. For information about creating a maintenance window, see Tutorial: Create a maintenance window for patching using the console.
Global or selective update for new environments	If you're getting started with Systems Manager, we recommend that you use the **Update Systems Manager (SSM) Agent every two weeks** option in Quick Setup, a tool in

Method	Details
	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 Set up Amazon EC2 host management using Quick Setup.

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. 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 Updating the SSM Agent using Run Command.

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. 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**.

3. Choose the **Settings** tab.
4. In the **Agent auto update** area, choose **Auto update SSM Agent**.

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**, depending on whether you chose `Email` or `SMS` in the previous step, enter an email address or an area code and number to receive notifications.

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 **Delete**. When prompted for confirmation, choose **Delete**.

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. If your agent appears to be unresponsive or has reduced communication frequency, see Understanding SSM Agent hibernation.

Topics

- SSM Agent is out of date
- Troubleshoot issues using SSM Agent log files
- Agent log files don't rotate (Windows)
- Unable to connect to SSM endpoints
- Verify your VPC configuration
- Verify your VPC DNS-related attributes
- Verify ingress rules on endpoint security groups
- Use ssm-cli to troubleshoot managed node availability

SSM Agent is out of date

An updated version of SSM Agent is released whenever new tools are added to Systems Manager or updates are made to existing tools. Failing to use the latest version of the agent can prevent your managed node from using various Systems Manager tools 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. Subscribe to the SSM Agent Release Notes page on GitHub to get notifications about SSM Agent updates.

Troubleshoot issues using 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.

> **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`.

For additional information about troubleshooting using agent logs, see How do I use SSM Agent logs to troubleshoot issues with SSM Agent in my managed instance? in the *AWS re:Post Knowledge Center*.

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" />
      <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 allow HTTPS (port 443) outbound traffic to the following endpoints:

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

`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*.

Note

Prior to 2024, `ec2messages.region.amazonaws.com` was also required. For AWS Regions launched before 2024, allowing traffic to `ssmmessages.region.amazonaws.com` is still required but optional to `ec2messages.region.amazonaws.com`.
For Regions launched in 2024 and later, allowing traffic to `ssmmessages.region.amazonaws.com` is required, but `ec2messages.region.amazonaws.com` endpoints are not supported for these Regions.

SSM Agent won't work if it can't communicate with the preceding endpoints, as described, even if you use AWS provided Amazon Machine Images (AMIs) such as Amazon Linux 2 or Amazon Linux 2023. 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 Improve the security of EC2 instances by using VPC endpoints for Systems Manager.

Verify your VPC configuration

If you are using a virtual private cloud (VPC), in order to manage EC2 instances with Systems Manager, your VPC endpoints must be configured properly for `ssm.region.amazonaws.com`, `ssmmessages.region.amazonaws.com`, and in some cases explained earlier in this topic in Unable to connect to SSM endpoints, `ec2messages.region.amazonaws.com`.

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.

To troubleshoot issues with your VPC endpoints, do the following:

- Ensure that VPC endpoints are included at the VPC level. If the VPC endpoint with a specific service name is not found on the VPC, first verify that DNS support is enabled at the VPC level. Next, create a new VPC endpoint and associate it with one subnet in each Availability Zone.
- Ensure that a private DNS name is enabled at the VPC endpoint level. Private DNS names are enabled by default but might have been manually disabled at some point.
- Ensure that existing VPC endpoints are associated with the proper subnet. In addition, ensure that the VPCE is already associated with a subnet in that Availability Zone.

For more information, see the following topics:

- Access an AWS service using an interface VPC endpoint in the *AWS PrivateLink Guide*
- Associate a private DNS name in the *AWS PrivateLink Guide*
- Improve the security of EC2 instances by using VPC endpoints for Systems Manager

Verify your VPC DNS-related attributes

If you are using a virtual private cloud (VPC), as part of verifying your VPC configuration, ensure that the attributes `enableDnsSupport` and `enableDnsHostnames` are enabled.

You can enable these attributes using the Amazon EC2 ModifyVPCAttribute API action or the AWS CLI command modify-vpc-attribute.

For information about enabling these attributes in the Amazon VPC Console, see View and update DNS attributes for your VPC in the *Amazon VPC User Guide*.

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.

Verify ingress rules on endpoint security groups

Ensure that any VPC endpoints you have configured (`ssm`, `ssmmessages`, and `ec2messages`) include an ingress rule on their security groups to allow traffic in on port 443. If necessary, you can create a new security group in the VPC with an ingress rule to allow traffic on port 443 for the Classless Inter-Domain Routing (CIDR) block for the VPC. After you create the security group, attach it to each VPC endpoint.

For more information, see the following topics:

- How do I create VPC endpoints so that I can use Systems Manager to manage private EC2 instances without internet access? on AWS re:Post
- VPC CIDR blocks in the *Amazon VPC User Guide*

Use `ssm-cli` to troubleshoot managed node availability

Starting with SSM Agent version 3.1.501.0, you can use `ssm-cli` to determine whether a managed node meets the primary requirements to be managed by Systems Manager, and to appear in lists of managed nodes in Fleet Manager. 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 or non-EC2 machine that you have confirmed is running isn't included in your lists of managed nodes in Systems Manager. These commands are run when you specify the `get-diagnostics` option.

For more information, see Troubleshooting managed node availability using `ssm-cli`.

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
- Data perimeters in AWS Systems Manager
- Identity and access management for AWS Systems Manager
- Using service-linked roles for Systems Manager
- Logging and monitoring in AWS Systems Manager
- Compliance validation for AWS Systems Manager
- Resilience in AWS Systems Manager
- Infrastructure security in AWS Systems Manager
- Configuration and vulnerability analysis in AWS Systems Manager
- Security best practices for Systems Manager

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. You are also responsible for 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 users with AWS IAM Identity Center or 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 require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- 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 sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 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) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text 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 text 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 tool in AWS Systems Manager, include `String`, `StringList`, and `SecureString`.

All parameters, regardless of their type, are encrypted both in transit and at rest. In transit, parameters are encrypted using transport layer security (TLS) to create a secure HTTPS connection for API requests. At rest, they are encrypted with an AWS owned key in AWS Key Management Service (AWS KMS). For more information about AWS owned key encryption, see AWS owned keys in the *AWS Key Management Service Developer Guide* .

The `SecureString` type offers additional encryption options and is recommended for all sensitive data. You can choose from the following types of AWS KMS keys to encrypt and decrypt the value of a `SecureString` parameter:

- The AWS managed key for your account
- A customer managed key (CMK) that you have created in your account
- A CMK in another AWS account that has been shared with you

For more information about AWS KMS encryption, see the AWS Key Management Service Developer Guide.

Content in 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 tool in AWS Systems Manager
- Packages in Distributor, a tool in AWS Systems Manager
- Patching operation logs in Patch Manager, a tool in AWS Systems Manager

- Patch Manager patch override lists
- Scripts or Ansible Playbooks to run in a runbook workflow in Automation, a tool in AWS Systems Manager
- Chef InSpec profiles for use with scans in Compliance, a tool in AWS Systems Manager
- AWS CloudTrail logs
- Session history logs in Session Manager, a tool in AWS Systems Manager
- Reports from Explorer, a tool in AWS Systems Manager
- OpsData from OpsCenter, a tool in 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 tool in 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 1.3 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 1.3, 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 Improve the security of EC2 instances by using VPC endpoints for Systems Manager.

For more information about Amazon Virtual Private Cloud security, see Internetwork traffic privacy in Amazon VPC in the *Amazon VPC User Guide*.

Data perimeters in AWS Systems Manager

A data perimeter is a set of preventive guardrails in your AWS environment that help ensure your data can only be accessed by trusted identities from expected networks and resources. When you implement data perimeter controls, you might need to include exceptions for AWS service-owned resources that Systems Manager accesses on your behalf.

For more information about data perimeters, see Data perimeters on AWS.

AWS service-owned resources accessed by Systems Manager

Systems Manager accesses the AWS service-owned resources listed below to provide functionality.

SSM document categories S3 bucket

Systems Manager accesses an AWS managed S3 bucket to retrieve document category information for AWS Systems Manager Documents. This bucket contains metadata about document categories that help organize and classify SSM Documents in the console.

Resource ARN pattern

`arn:aws:s3:::ssm-document-categories-`*`region`*

Regional examples:

- `arn:aws:s3:::ssm-document-categories-us-east-1`
- `arn:aws:s3:::ssm-document-categories-us-west-2`
- `arn:aws:s3:::ssm-document-categories-eu-west-1`
- `arn:aws:s3:::ssm-document-categories-ap-northeast-1`

When accessed

This resource is accessed when you view SSM Documents in the Systems Manager console or when using APIs that retrieve document metadata and categories.

Data stored

The bucket contains JSON files with document category definitions and metadata. This data is read-only and does not contain customer-specific information.

Identity used

Systems Manager accesses this resource using AWS service credentials on behalf of your requests.

Required permissions

`s3:GetObject` on the bucket contents.

Data perimeter policy considerations

When implementing data perimeter controls using Service Control Policies (SCPs) or VPC endpoint policies with conditions like `aws:ResourceOrgID`, you need to create exceptions for the AWS service-owned resources that Systems Manager requires.

For example, if you're using an SCP with `aws:ResourceOrgID` to restrict access to resources outside your organization, you would need to add an exception for the SSM Document categories bucket:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RestrictToOrgResources",
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceOrgID": "o-example1234567"
        },
        "ForAllValues:StringNotLike": {
          "aws:ResourceArn": [
            "arn:aws:s3:::ssm-document-categories*"
          ]
        }
      }
    }
  ]
}
```

This policy denies access to resources outside your organization, but includes an exception for any S3 bucket that matches the `ssm-document-categories*` pattern, allowing Systems Manager to continue functioning properly.

Similarly, if you're using VPC endpoint policies to restrict S3 access, you would need to ensure that the SSM document categories buckets are accessible through your VPC endpoints.

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
- Authenticating with identities
- Managing access using policies
- How AWS Systems Manager works with IAM
- AWS Systems Manager identity-based policy examples
- AWS managed policies for AWS Systems Manager
- Troubleshooting AWS Systems Manager identity and access

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.

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 service users 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.

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.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might 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 Multi-factor authentication in the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one 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 don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

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 Use cases for IAM users 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. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). 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 Methods to assume a role in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **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 Cross account resource access in IAM 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.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **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 Create 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 AWS 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 Use an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

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**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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 Define custom IAM permissions with customer managed 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 Choose between managed policies and inline policies in the *IAM User Guide*.

For information about AWS managed policies for Systems Manager, see AWS Systems Manager managed policies.

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.

Policy condition keys

The actions that users and roles can perform and the resources on which they can take those actions can be further restricted by specific *conditions*.

In JSON policy documents, 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 `StringEquals` or `StringNotLike`, 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. For more information, see AWS global condition context keys in the *IAM User Guide*.

> **Important**
>
> If you use Systems Manager Automation, we recommend you don't use the aws:SourceIp condition key in your policies. The behavior of this condition key is dependent on multiple factors, including whether an IAM role for Automation runbook execution is supplied and the Automation actions used in the runbook. As a result, the condition key can produce unexpected behavior. For this reason, we recommend you don't use it.

Systems Manager supports a number of its own condition keys. For more information, see Condition Keys for AWS Systems Manager in the *Service Authorization Reference*. The actions and resources you can use a Systems Manager-specific condition key with are listed in Resource types defined by AWS Systems Manager in the *Service Authorization Reference*.

If your policy must depend on a service principal name owned by the Systems Manager service, we recommend you check for its existence or non-existence using the `aws:PrincipalServiceNamesList` multivalued condition key, rather than the `aws:PrincipalServiceName` condition key. The `aws:PrincipalServiceName` condition key contains only one entry from the list of service principal names and it may not always be the service principal name you expect. The following `Condition` block demonstrates checking for the existence of `ssm.amazonaws.com`.

```
{
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:PrincipalServiceNamesList": "ssm.amazonaws.com"
        }
    }
}
```

To view examples of Systems Manager identity-based policies, see AWS Systems Manager identity-based policy examples.

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 an 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 Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) 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
- Systems Manager resource-based policies
- Authorization based on Systems Manager tags
- Systems Manager IAM roles

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

The following tools in AWS Systems Manager use different prefixes before actions.

- AWS AppConfig uses the prefix `appconfig:` before actions.
- Incident Manager uses the prefix `ssm-incidents:` or `ssm-contacts:` before actions.
- Systems Manager GUI Connect uses the prefix `ssm-guiconnect:` before actions.
- Quick Setup uses the prefix `ssm-quicksetup:` 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 table below describes the ARN formats for the resource types supported by Systems Manager.

Note

Note the following exceptions to ARN formats.

- The following tools in AWS Systems Manager use different prefixes before actions.
  - AWS AppConfig uses the prefix `appconfig:` before actions.
  - Incident Manager uses the prefix `ssm-incidents:` or `ssm-contacts:` before actions.
  - Systems Manager GUI Connect uses the prefix `ssm-guiconnect` before actions.
- Documents and automation definition resources that are owned by Amazon, as well as public parameters that are provided by both Amazon and third-party sources, do not include account IDs in their ARN formats. For example:
  - The SSM document `AWS-RunPatchBaseline`:

    `arn:aws:ssm:us-east-2::document/AWS-RunPatchBaseline`
  - The automation runbook `AWS-ConfigureMaintenanceWindows`:

`arn:aws:ssm:us-east-2::automation-definition/AWS-ConfigureMaintenanceWindows`

- The public parameter `/aws/service/bottlerocket/aws-ecs-1-nvidia/x86_64/1.13.4/image_version`:

  `arn:aws:ssm:us-east-2::parameter/aws/service/bottlerocket/aws-ecs-1-nvidia/x86_64/1.13.4/image_version`

For more information about these three resource types, see the following topics:

- Working with documents
- Run an automated operation powered by Systems Manager Automation
- Working with public parameters in Parameter Store

- Quick Setup uses the prefix `ssm-quicksetup:` before actions.

Resource type	ARN format
Application (AWS AppConfig)	arn:aws:appconfig:*region*:*account-id* :application/*application-id*
Association	arn:aws:ssm:*region*:*account-id* :association/*association-id*
Automation execution	arn:aws:ssm:*region*:*account-id* :automation-execution/*automation-execution-id*
Automation definition (with version subresource)	arn:aws:ssm:*region*:*account-id* :automation-definition/*automation-definition-id* :*version-id* [1]
Configuration profile (AWS AppConfig)	arn:aws:appconfig:*region*:*account-id* :application/*application-id* /configurationprofile/*configurationprofile-id*
Contact (Incident Manager)	arn:aws:ssm-contacts:*region*:*account-id* :contact/*contact-alias*
Deployment strategy (AWS AppConfig)	arn:aws:appconfig:*region*:*account-id* :deploymentstrategy/*deploymentstrategy-id*

Resource type	ARN format
Document	arn:aws:ssm:*region*:*account-id* :document/*document-name*
Environment (AWS AppConfig)	arn:aws:appconfig:*region*:*account-id* :application/*application-id* /environment/*environment-id*
Incident	arn:aws:ssm-incidents:*region*:*account-id* :incident-record/*response-plan-name* /*incident-id*
Maintenance window	arn:aws:ssm:*region*:*account-id* :maintenancewindow/*window-id*
Managed node	arn:aws:ssm:*region*:*account-id* :managed-instance/*managed-node-id*
Managed node inventory	arn:aws:ssm:*region*:*account-id* :managed-instance-inventory /*managed-node-id*
OpsItem	arn:aws:ssm:*region*:*account-id* :opsitem/*OpsItem-id*
Parameter	A one-level parameter: • arn:aws:ssm:*region*:*account-id* :parameter/*parameter-name*/ A parameter named with a hierarchical construction: • arn:aws:ssm:*region*:*account-id* :parameter/*parameter-name-root* /*level-2*/*level-3*/*level-4*/*level-5* [2]
Patch baseline	arn:aws:ssm:*region*:*account-id* :patchbaseline/*patch-baseline-id*
Response plan	arn:aws:ssm-incidents:*region*:*account-id* :response-plan/*response-plan-name*
Session	arn:aws:ssm:*region*:*account-id* :session/*session-id* [3]

Resource type	ARN format
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:*region*:*account-id* :*

[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 Parameter Store parameters in Systems Manager.

[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) 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 for Systems Manager

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
- Restricting Run Command access based on tags
- Restrict session access based on instance tags

For information about using the `ssm:Recursive`, `ssm:Policies`, and `ssm:Overwrite` condition keys, see Preventing access to Parameter Store API operations.

Examples

To view examples of Systems Manager identity-based policies, see AWS Systems Manager identity-based policy examples.

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

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.

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 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.

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 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, you must configure instance permissions. For information, see Configure instance permissions required for Systems Manager.

To access non-EC2 nodes in a hybrid and multicloud, the role your AWS account needs is an IAM service role. For information, see Create the IAM service role required for Systems Manager in hybrid and multicloud environments.

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.

AWS Systems Manager managed policies

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 about managed policies for Systems Manager, see AWS managed policies for AWS Systems Manager

For general information about managed policies, see AWS managed policies in the *IAM User Guide*.

AWS Systems Manager identity-based policy examples

By default, AWS Identity and Access Management (IAM) entities (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 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 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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "ssm:DeleteDocument"
      ],
      "Resource" : [
        "arn:aws:ssm:us-east-1:111122223333: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
- Example: Permission to using the Systems Manager console
- Example: Permission to allow users to view their own permissions
- Example: Permission to read and describe individual parameters
- Cross-service confused deputy prevention
- Customer managed policy examples
- Viewing Systems Manager documents based on tags

Policy best practices

Identity-based policies 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 with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and

functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Example: Permission to 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.

If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for IAM entities (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.

To ensure that users and roles can still use the Systems Manager console, also attach the AmazonSSMFullAccess or AmazonSSMReadOnlyAccess AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Example: Permission to 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": "*"
        }
    ]
}
```

Example: Permission to read and describe individual parameters

Example Read and describe one parameter

You can grant access to a parameter by attaching the following policy to an identity.

JSON

```
{
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ssm:GetParameter",
      "ssm:DescribeParameters"
```

```
      ],
    "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/parameter-name"
  }
]
}
```

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 Resource Name (ARN) for an S3 bucket, 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 tools.

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).

JSON

```
{
   "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-1:123456789012:*"
            }
         }
      }
   ]
}
```

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.

JSON

```
{
    "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-1:123456789012: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 entities (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 Example: Permission to using the Systems Manager console.

> **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
- Example 2: Allow a user to list documents for a single Region

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).

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ssm:*"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333:*"
            ]
        }
    ]
}
```

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).

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:ListDocuments"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1:111122223333: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:ec2:us-east-2:aws-account-ID:instance/i-02573cafcfEXAMPLE",
"arn:aws:ec2:us-east-2:aws-account-ID:instance/i-0471e04240EXAMPLE",
"arn:aws:ec2:us-east-2:aws-account-ID:instance/i-07782c72faEXAMPLE"
```

- View details about a command after it has been sent.
- Start and stop workflows in Automation, a tool in 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, 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:ec2:us-east-2:*:instance/*"
```

JSON

```
{
    "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:ec2:us-east-1:111122223333:instance/
i-02573cafcfEXAMPLE",
                "arn:aws:ec2:us-east-1:111122223333:instance/
i-0471e04240EXAMPLE",
                "arn:aws:ec2:us-east-1:111122223333:instance/
i-07782c72faEXAMPLE",

                "arn:aws:ssm:us-east-1:111122223333: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-1:111122223333: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.

JSON

```
{
    "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 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

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

Topics

- AWS managed policy: AmazonSSMServiceRolePolicy
- AWS managed policy: AmazonSSMAutomationRole
- AWS managed policy: AmazonSSMReadOnlyAccess

- AWS managed policy: AWSSystemsManagerOpsDataSyncServiceRolePolicy
- AWS managed policy: AmazonSSMManagedEC2InstanceDefaultPolicy
- AWS managed policy: SSMQuickSetupRolePolicy
- AWS managed policy: AWSQuickSetupDeploymentRolePolicy
- AWS managed policy: AWSQuickSetupPatchPolicyDeploymentRolePolicy
- AWS managed policy: AWSQuickSetupPatchPolicyBaselineAccess
- AWS managed policy: AWSSystemsManagerEnableExplorerExecutionPolicy
- AWS managed policy: AWSSystemsManagerEnableConfigRecordingExecutionPolicy
- AWS managed policy: AWSQuickSetupDevOpsGuruPermissionsBoundary
- AWS managed policy: AWSQuickSetupDistributorPermissionsBoundary
- AWS managed policy: AWSQuickSetupSSMHostMgmtPermissionsBoundary
- AWS managed policy: AWSQuickSetupPatchPolicyPermissionsBoundary
- AWS managed policy: AWSQuickSetupSchedulerPermissionsBoundary
- AWS managed policy: AWSQuickSetupCFGCPacksPermissionsBoundary
- AWS managed policy: AWSQuickSetupStartStopInstancesExecutionPolicy
- AWS managed policy: AWSQuickSetupStartSSMAssociationsExecutionPolicy
- AWS managed policy: AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy
- AWS managed policy: AWS-SSM-DiagnosisAutomation-ExecutionRolePolicy
- AWS managed policy: AWS-SSM-RemediationAutomation-AdministrationRolePolicy
- AWS managed policy: AWS-SSM-RemediationAutomation-ExecutionRolePolicy
- AWS managed policy: AWSQuickSetupSSMManageResourcesExecutionPolicy
- AWS managed policy: AWSQuickSetupSSMLifecycleManagementExecutionPolicy
- AWS managed policy: AWSQuickSetupSSMDeploymentRolePolicy
- AWS managed policy: AWSQuickSetupSSMDeploymentS3BucketRolePolicy
- AWS managed policy: AWSQuickSetupEnableDHMCExecutionPolicy
- AWS managed policy: AWSQuickSetupEnableAREXExecutionPolicy
- AWS managed policy: AWSQuickSetupManagedInstanceProfileExecutionPolicy
- AWS managed policy: AWSQuickSetupReadOnlyAccess
- AWS managed policy: AWSQuickSetupManageJITNAResourcesExecutionPolicy
- AWS managed policy: AWSQuickSetupJITNADeploymentRolePolicy

- AWS managed policy: AWSSystemsManagerJustInTimeAccessServicePolicy
- AWS managed policy: AWSSystemsManagerJustInTimeAccessTokenPolicy
- AWS managed policy: AWSSystemsManagerJustInTimeAccessTokenSessionPolicy
- AWS managed policy: AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy
- AWS managed policy: AWSSystemsManagerNotificationsServicePolicy
- AWS managed policy: AWS-SSM-Automation-DiagnosisBucketPolicy
- AWS managed policy: AWS-SSM-RemediationAutomation-OperationalAccountAdministrationRolePolicy
- AWS managed policy: AWS-SSM-DiagnosisAutomation-OperationalAccountAdministrationRolePolicy
- Systems Manager updates to AWS managed policies
- Additional managed policies for Systems Manager

AWS managed policy: AmazonSSMServiceRolePolicy

This policy provides access to a number of AWS resources that are managed by AWS Systems Manager or used in Systems Manager operations.

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 and view OpsData.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to start and step executions for both Run Command and Automation; and to retrieve information about Run Command and Automation operations; to retrieve information about Parameter Store parameters Change Calendar calendars; to update and retrieve information about Systems Manager service settings for OpsCenterresources; and to read information about tags that have have applied to resources.
- `cloudformation` – Allows principals to retrieve information about stackset operations and stackset instances, and to delete stacksets on the resource `arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-SSM*:*`. Allows principals to delete stack instances that are associated with the following resources:

```
arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-SSM*:*
arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-SSM*:*
arn:aws:cloudformation:*:*:type/resource/*
```

- `cloudwatch` – Allows principals to retrieve information about Amazon CloudWatch alarms.
- `compute-optimizer` – Allows principals to retrieve the enrollment (opt in) status of an account to the AWS Compute Optimizer service, and to retrieve recommendations for Amazon EC2 instances that meet a specific set of stated requirements.
- `config` – Allows principals to retrieve information remediation configurations and configuration recorders in AWS Config, and to determine whether specified AWS Config rules and AWS resources are compliant.
- `events` – Allows principals retrieve information about EventBridge rules; to create EventBridge rules and targets exclusively for the the Systems Manager service (`ssm.amazonaws.com`); and to delete rules and targets for the resource `arn:aws:events:*:*:rule/SSMExplorerManagedRule`.
- `ec2` – Allows principals to retrieve information about Amazon EC2 instances..
- `iam` – Allows principals to pass roles permissions for the Systems Manager service (`ssm.amazonaws.com`).
- `lambda` – Allows principals to invoke Lambda functions that are configured specifically for use by Systems Manager.
- `resource-explorer-2` – Allows principals to retrieve data about EC2 instances to determine whether or not each instance is currently managed by Systems Manager.

  The action `resource-explorer-2:CreateManagedView` is allowed for the `arn:aws:resource-explorer-2:*:*:managed-view/AWSManagedViewForSSM*` resource.
- `resource-groups` – Allows principals to retrieve list resource groups and their members from AWS Resource Groups of resources that belong to a resource group.
- `securityhub` – Allows principals to retrieve information about AWS Security Hub hub resources in the current account.
- `states` – Allows principals to start and retrieve information for AWS Step Functions that are configured specifically for use by Systems Manager.
- `support` – Allows principals to retrieve information about checks and cases in AWS Trusted Advisor.

- `tag` – Allows principals to retrieve information about all the tagged or previously tagged resources that are located in a specified AWS Region for an account.

To view more details about the policy, including the latest version of the JSON policy document, see AmazonSSMServiceRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AmazonSSMAutomationRole

You can attach the `AmazonSSMAutomationRole` policy to your IAM identities. This policy provides permissions for the AWS Systems Manager Automation service to run activities defined within Automation runbooks.

Permissions details

This policy includes the following permissions.

- `lambda` – Allows principals to invoke Lambda functions with names that begin with "Automation". This is required for Automation runbooks to execute Lambda functions as part of their workflow.
- `ec2` – Allows principals to perform various Amazon EC2 operations including creating, copying, and deregistering images; managing snapshots; starting, running, stopping, and terminating instances; managing instance status; and creating, deleting, and describing tags. These permissions enable Automation runbooks to manage Amazon EC2 resources during execution.
- `cloudformation` – Allows principals to create, describe, update, and delete AWS CloudFormation stacks. This enables Automation runbooks to manage infrastructure as code through CloudFormation.
- `ssm` – Allows principals to use all Systems Manager actions. This comprehensive access is required for Automation runbooks to interact with all Systems Manager capabilities.
- `sns` – Allows principals to publish messages to Amazon SNS topics with names that begin with "Automation". This enables Automation runbooks to send notifications during execution.

To view more details about the policy, including the latest version of the JSON policy document, see AmazonSSMAutomationRole in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AmazonSSMReadOnlyAccess

You can attach the `AmazonSSMReadOnlyAccess` policy to your IAM identities. This policy grants read-only access to AWS Systems Manager API operations including `Describe*`, `Get*`, and `List*`.

To view more details about the policy, including the latest version of the JSON policy document, see AmazonSSMReadOnlyAccess in the *AWS Managed Policy Reference Guide*.

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 Explorer.

`AWSSystemsManagerOpsDataSyncServiceRolePolicy` allows the `AWSServiceRoleForSystemsManagerOpsDataSync` service-linked role to create and update OpsItems and OpsData from AWS Security Hub findings.

The 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.

```
{
   "Effect": "Deny",
   "Action": "securityhub:BatchUpdateFindings",
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "securityhub:ASFFSyntaxPath/Workflow.Status": "SUPPRESSED"
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": "securityhub:BatchUpdateFindings",
   "Resource": "*",
   "Condition": {
    "Null": {
     "securityhub:ASFFSyntaxPath/Confidence": false
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": "securityhub:BatchUpdateFindings",
   "Resource": "*",
   "Condition": {
    "Null": {
     "securityhub:ASFFSyntaxPath/Criticality": false
    }
   }
  },
  {
```

```
    "Effect": "Deny",
    "Action": "securityhub:BatchUpdateFindings",
    "Resource": "*",
    "Condition": {
     "Null": {
      "securityhub:ASFFSyntaxPath/Note.Text": false
     }
    }
   },
   {
    "Effect": "Deny",
    "Action": "securityhub:BatchUpdateFindings",
    "Resource": "*",
    "Condition": {
     "Null": {
      "securityhub:ASFFSyntaxPath/Note.UpdatedBy": false
     }
    }
   },
   {
    "Effect": "Deny",
    "Action": "securityhub:BatchUpdateFindings",
    "Resource": "*",
    "Condition": {
     "Null": {
      "securityhub:ASFFSyntaxPath/RelatedFindings": false
     }
    }
   },
   {
    "Effect": "Deny",
    "Action": "securityhub:BatchUpdateFindings",
    "Resource": "*",
    "Condition": {
     "Null": {
      "securityhub:ASFFSyntaxPath/Types": false
     }
    }
   },
   {
    "Effect": "Deny",
    "Action": "securityhub:BatchUpdateFindings",
    "Resource": "*",
    "Condition": {
```

```
    "Null": {
     "securityhub:ASFFSyntaxPath/UserDefinedFields.key": false
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": "securityhub:BatchUpdateFindings",
   "Resource": "*",
   "Condition": {
    "Null": {
     "securityhub:ASFFSyntaxPath/UserDefinedFields.value": false
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": "securityhub:BatchUpdateFindings",
   "Resource": "*",
   "Condition": {
    "Null": {
     "securityhub:ASFFSyntaxPath/VerificationState": false
    }
   }
```

To view more details about the policy, including the latest version of the JSON policy document, see AWSSystemsManagerOpsDataSyncServiceRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AmazonSSMManagedEC2InstanceDefaultPolicy

You should only attach `AmazonSSMManagedEC2InstanceDefaultPolicy` to IAM roles for Amazon EC2 instances that you want to have permission to use Systems Manager functionality. You shouldn't attached this role to other IAM entities, such as IAM users and IAM groups, or to IAM roles that serve other purposes. For more information, see Managing EC2 instances automatically with Default Host Management Configuration.

This policy grants permissions that allow SSM Agent on your Amazon EC2 instance to communicate with the Systems Manager service in the cloud in order to perform a variety of tasks. It also grants permissions for the two services that provide authorization tokens to ensure that operations are performed on the correct instance.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to retrieve Documents, execute commands using Run Command, establish sessions using Session Manager, collect an inventory of the instance, and scan for patches and patch compliance using Patch Manager.
- `ssmmessages` – Allows principals to access, for each instance, a personalized authorization token that was created by the *Amazon Message Gateway Service*. Systems Manager validates the personalized authorization token against the Amazon Resource Name (ARN) of the instance that was provided in the API operation. This access is necessary to ensure that SSM Agent performs the API operations on the correct instance.
- `ec2messages` – Allows principals to access, for each instance, a personalized authorization token that was created by the *Amazon Message Delivery Service*. Systems Manager validates the personalized authorization token against the Amazon Resource Name (ARN) of the instance that was provided in the API operation. This access is necessary to ensure that SSM Agent performs the API operations on the correct instance.

For related information about the `ssmmessages` and `ec2messages` endpoints, including the differences between the two, see Agent-related API operations (`ssmmessages` and `ec2messages` endpoints).

To view more details about the policy, including the latest version of the JSON policy document, see AmazonSSMManagedEC2InstanceDefaultPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: SSMQuickSetupRolePolicy

You can't attach SSMQuickSetupRolePolicy 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 maintain Quick Setup-provisioned resource health and consistency.

This policy grants read-only permissions that allow Systems Manager to check configuration health, ensure consistent use of parameters and provisioned resources, and remediate resources when drift is detected. It also grants administrative permissions for creating a service-linked role.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to read information Resource Data Syncs and SSM Documents in Systems Manager, including in delegated administrator accounts. This is required so Quick Setup can determine the state that configured resources are intended to be in.
- `organizations` – Allows principals to read information about the member accounts that belong to an organization as configured in AWS Organizations. This is required so Quick Setup can identify all accounts in an organization where resource health checks are to be performed.
- `cloudformation` – Allows principals to read information from AWS CloudFormation. This is required so Quick Setup can gather data about the AWS CloudFormation stacks used to manage the state of resources and CloudFormation stackset operations.

To view more details about the policy, including the latest version of the JSON policy document, see SSMQuickSetupRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupDeploymentRolePolicy

The managed policy `AWSQuickSetupDeploymentRolePolicy` supports multiple Quick Setup configuration types. These configuration types create IAM roles and automations that configure frequently used Amazon Web Services services and features with recommended best practices.

You can attach `AWSQuickSetupDeploymentRolePolicy` to your IAM entities.

This policy grants administrative permissions needed to create resources associated with the following Quick Setup configurations:

- Set up Amazon EC2 host management using Quick Setup
- Create an AWS Config configuration recorder using Quick Setup
- Deploy AWS Config conformance pack using Quick Setup
- Set up DevOps Guru using Quick Setup
- Deploy Distributor packages using Quick Setup
- Stop and start EC2 instances automatically on a schedule using Quick Setup

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to read, create, update, and delete SSM documents with names beginning with "AWSQuickSetup-" or "AWSOperationsPack-" when called via AWS

CloudFormation; to read specific AWS owned documents including "AWSQuickSetupType-ManageInstanceProfile"; to create, update, and delete associations for Quick Setup documents and AWS owned documents when called via AWS CloudFormation; and to clean up legacy resources tagged with `QuickSetupID`. This enables Quick Setup to deploy and manage automation workflows and associations.

- `cloudformation` – Allows principals to read information about AWS CloudFormation stacks and stack sets; and to create, update, and delete AWS CloudFormation stacks and change sets for resources with names beginning with "StackSet-AWS-QuickSetup-". This enables Quick Setup to manage infrastructure deployments across accounts and regions.
- `config` – Allows principals to read information about AWS Config conformance packs and their status; and to create and delete conformance packs with names beginning with "AWS-QuickSetup-" when called via AWS CloudFormation. This enables Quick Setup to deploy compliance monitoring configurations.
- `events` – Allows principals to manage EventBridge rules and targets for resources with names containing "QuickSetup-". This enables Quick Setup to create scheduled automation workflows.
- `iam` – Allows principals to create service-linked roles for AWS Config and Systems Manager; to create, manage, and delete IAM roles with names beginning with "AWS-QuickSetup-" or "AWSOperationsPack-" when called via AWS CloudFormation; to pass these roles to Systems Manager and EventBridge services; to attach specific AWS managed policies to these roles; and to set permissions boundaries using specific Quick Setup managed policies. This enables Quick Setup to create the necessary service roles for its operations.
- `resource-groups` – Allows principals to retrieve resource group queries. This enables Quick Setup to target specific sets of resources for configuration management.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupDeploymentRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupPatchPolicyDeploymentRolePolicy

The managed policy `AWSQuickSetupPatchPolicyDeploymentRolePolicy` supports the Configure patching for instances in an organization using a Quick Setup patch policy Quick Setup type. This configuration type helps automate patching of applications and nodes in a single account or across your organization.

You can attach `AWSQuickSetupPatchPolicyDeploymentRolePolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

This policy grants administrative permissions that allow Quick Setup to create resources associated with a patch policy configuration.

Permissions details

This policy includes the following permissions.

- `iam` – Allows principals to manage and delete IAM roles required for Automation configuration tasks; and to manage Automation role policies.
- `cloudformation` – Allows principals to read AWS CloudFormation stack information; and to control AWS CloudFormation stacks that were created by Quick Setup using AWS CloudFormation stack sets.
- `ssm` – Allows principals to create, update, read, and delete Automation runbooks required for configuration tasks; and to create, update, and delete State Manager associations.
- `resource-groups` – Allows principals to retrieve resource queries that are associated with resource groups targeted by Quick Setup configurations.
- `s3` – Allows principals to list Amazon S3 buckets; and to manage the buckets for storing patch policy access logs.
- `lambda` – Allows principals to manage AWS Lambda remediation functions that maintain configurations in the correct state.
- `logs` – Allows principals to describe and manage log groups for Lambda configuration resources.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupPatchPolicyDeploymentRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupPatchPolicyBaselineAccess

The managed policy `AWSQuickSetupPatchPolicyBaselineAccess` supports the Configure patching for instances in an organization using a Quick Setup patch policy Quick Setup type. This configuration type helps automate patching of applications and nodes in a single account or across your organization.

You can attach `AWSQuickSetupPatchPolicyBaselineAccess` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

This policy provides read-only permissions to access patch baselines that have been configured by an administrator in the current AWS account or organization using Quick Setup. The patch baselines are stored in an Amazon S3 bucket and can be used for patching instances in a single account or across an entire organization.

Permissions details

This policy includes the following permission.

- `s3` – Allows principals to read patch baseline overrides stored in Amazon S3 buckets.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupPatchPolicyBaselineAccess in the *AWS Managed Policy Reference Guide*.

AWS managed policy: `AWSSystemsManagerEnableExplorerExecutionPolicy`

The managed policy `AWSSystemsManagerEnableExplorerExecutionPolicy` supports enabling Explorer, a tool in AWS Systems Manager.

You can attach `AWSSystemsManagerEnableExplorerExecutionPolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

This policy grants administrative permissions for enabling Explorer. This includes permissions to update related Systems Manager service settings, and to create a service-linked role for Systems Manager.

Permissions details

This policy includes the following permissions.

- `config` – Allows principals to help enable Explorer by providing read-only access to configuration recorder details.
- `iam` – Allows principals to help enable Explorer.
- `ssm` – Allows principals to start an Automation workflow that enables Explorer.

To view more details about the policy, including the latest version of the JSON policy document, see AWSSystemsManagerEnableExplorerExecutionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSSystemsManagerEnableConfigRecordingExecutionPolicy

The managed policy `AWSSystemsManagerEnableConfigRecordingExecutionPolicy` supports the Create an AWS Config configuration recorder using Quick Setup Quick Setup configuration type. This configuration type enables Quick Setup to track and record changes to the AWS resource types you choose for AWS Config. It also enables Quick Setup to configure delivery and notifications options for the recorded data.

You can attach `AWSSystemsManagerEnableConfigRecordingExecutionPolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

This policy grants administrative permissions that allow Quick Setup to enable and configure AWS Config configuration recording.

Permissions details

This policy includes the following permissions.

- `s3` – Allows principals to create and configure Amazon S3 buckets for delivery of configuration recordings.
- `sns` – Allows principals to list and create Amazon SNS topics.
- `config` – Allows principals to configure and start the configuration recorder; and to help enable Explorer.
- `iam` – Allows principals to create, get, and pass a service-linked role for AWS Config; and to create a service-linked role for Systems Manager; and to help enable Explorer.
- `ssm` – Allows principals to start an Automation workflow that enables Explorer.
- `compute-optimizer` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.
- `support` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.

To view more details about the policy, including the latest version of the JSON policy document, see AWSSystemsManagerEnableConfigRecordingExecutionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupDevOpsGuruPermissionsBoundary

> **Note**
>
> This policy is a *permissions boundary*. A permissions boundary sets the maximum permissions that an identity-based policy can grant to an IAM entity. You should not use and attach Quick Setup permissions boundary policies on your own. Quick Setup permissions boundary policies should only be attached to Quick Setup managed roles. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.

The managed policy `AWSQuickSetupDevOpsGuruPermissionsBoundary` supports the Set up DevOps Guru using Quick Setup type. The configuration type enables the machine learning-powered Amazon DevOps Guru. The DevOps Guru service can help improve an application's operational performance and availability.

When you create an `AWSQuickSetupDevOpsGuruPermissionsBoundary` configuration using Quick Setup, the system applies this permissions boundary to the IAM roles that are created when the configuration is deployed. The permissions boundary limits the scope of the roles that Quick Setup creates.

This policy grants administrative permissions that allow Quick Setup to enable and configure Amazon DevOps Guru.

Permissions details

This policy includes the following permissions.

- `iam` – Allows principals to create service-linked roles for DevOps Guru and Systems Manager; and to list roles that help enable Explorer.
- `cloudformation` – Allows principals to list and describe AWS CloudFormation stacks.
- `sns` – Allows principals to list and create Amazon SNS topics.
- `devops-guru` – Allows principals to configure DevOps Guru; and to add a notification channel.

- `config` – – Allows principals to help enable Explorer by providing read-only access to configuration recorder details.
- `ssm` – Allows principals to start an Automation workflow that enables Explorer; and to read and update Explorer service settings.
- `compute-optimizer` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.
- `support` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupDevOpsGuruPermissionsBoundary in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupDistributorPermissionsBoundary

> **Note**
>
> This policy is a *permissions boundary*. A permissions boundary sets the maximum permissions that an identity-based policy can grant to an IAM entity. You should not use and attach Quick Setup permissions boundary policies on your own. Quick Setup permissions boundary policies should only be attached to Quick Setup managed roles. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.

The managed policy `AWSQuickSetupDistributorPermissionsBoundary` supports the Deploy Distributor packages using Quick Setup Quick Setup configuration type. The configuration type helps enable the distribution of software packages, such as agents, to your Amazon Elastic Compute Cloud (Amazon EC2) instances, using Distributor, a tool in AWS Systems Manager.

When you create an `AWSQuickSetupDistributorPermissionsBoundary` configuration using Quick Setup, the system applies this permissions boundary to the IAM roles that are created when the configuration is deployed. The permissions boundary limits the scope of the roles that Quick Setup creates.

This policy grants administrative permissions that allow Quick Setup to enable the distribution of software packages, such as agents, to your Amazon EC2 instances using Distributor.

Permissions details

This policy includes the following permissions.

- `iam` – Allows principals to get and pass the Distributor automation role; to create, read, update, and delete the default instance role; to pass the default instance role to Amazon EC2 and Systems Manager; to attach instance management policies to instance roles; to create a service-linked role for Systems Manager; to add the default instance role to instance profiles; to read information about IAM roles and instance profiles; and to create the default instance profile.
- `ec2` – Allows principals to associate the default instance profile with EC2 instances; and to help enable Explorer.
- `ssm` – Allows principals to start automation workflows that which configure instances and install packages; and to help start the automation workflow that enables Explorer; and to read and update Explorer service settings.
- `config` – Allows principals to help enable Explorer by providing read-only access to configuration recorder details.
- `compute-optimizer` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.
- `support` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupDistributorPermissionsBoundary in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupSSMHostMgmtPermissionsBoundary

> **Note**
>
> This policy is a *permissions boundary*. A permissions boundary sets the maximum permissions that an identity-based policy can grant to an IAM entity. You should not use and attach Quick Setup permissions boundary policies on your own. Quick Setup permissions boundary policies should only be attached to Quick Setup managed roles. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.

The managed policy `AWSQuickSetupSSMHostMgmtPermissionsBoundary` supports the Set up Amazon EC2 host management using Quick Setup Quick Setup configuration type. This

configuration type configures IAM roles and enables commonly used Systems Manager tools to securely manage your Amazon EC2 instances.

When you create an `AWSQuickSetupSSMHostMgmtPermissionsBoundary` configuration using Quick Setup, the system applies this permissions boundary to the IAM roles that are created when the configuration is deployed. The permissions boundary limits the scope of the roles that Quick Setup creates.

This policy grants administrative permissions that allow Quick Setup to enable and configure Systems Manager tools needed for securely managing EC2 instances.

Permissions details

This policy includes the following permissions.

- `iam` – Allows principals to get and pass the service role to Automation. Allows principals to create, read, update, and delete the default instance role; to pass the default instance role to Amazon EC2 and Systems Manager; to attach instance management policies to instance roles; to create a service-linked role for Systems Manager; to add the default instance role to instance profiles; to read information about IAM roles and instance profiles; and to create the default instance profile.
- `ec2` – Allows principals to associate and disassociate the default instance profile with EC2 instances.
- `ssm` – Allows principals to start Automation workflows that enable Explorer; to read and update Explorer service settings; to configure instances; and to enable Systems Manager tools on instances.
- `compute-optimizer` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.
- `support` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupSSMHostMgmtPermissionsBoundary in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupPatchPolicyPermissionsBoundary

> **Note**
>
> This policy is a *permissions boundary*. A permissions boundary sets the maximum permissions that an identity-based policy can grant to an IAM entity. You should not use and attach Quick Setup permissions boundary policies on your own. Quick Setup permissions boundary policies should only be attached to Quick Setup managed roles. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.

The managed policy `AWSQuickSetupPatchPolicyPermissionsBoundary` supports the Configure patching for instances in an organization using a Quick Setup patch policy Quick Setup type. This configuration type helps automate patching of applications and nodes in a single account or across your organization.

When you create an `AWSQuickSetupPatchPolicyPermissionsBoundary` configuration using Quick Setup, the system applies this permissions boundary to the IAM roles that are created when the configuration is deployed. The permissions boundary limits the scope of the roles that Quick Setup creates.

This policy grants administrative permissions that allow Quick Setup to enable and configure patch policies in Patch Manager, a tool in AWS Systems Manager.

Permissions details

This policy includes the following permissions.

- `iam` – Allows principals to get the Patch Manager Automation role; to pass Automation roles to Patch Manager patching operations; to create the default instance role, `AmazonSSMRoleForInstancesQuickSetup`; to pass the default instance role to Amazon EC2 and Systems Manager; to attach selected AWS managed policies to the instance role; to create a service-linked role for Systems Manager; to add the default instance role to instance profiles; to read information about instance profiles and roles; to create a default instance profile; and to tag roles that have permissions to read patch baseline overrides.
- `ssm` – Allows principals to update the instance role this is managed by Systems Manager; to manage associations created by Patch Manager patch policies created in Quick Setup; to tag instances targeted by a patch policy configuration; to read information about instances and

patching status; to start Automation workflows that configure, enable and remediate instance patching; to start automation workflows that enable Explorer; to help enable Explorer; and to read and update Explorer service settings.

- `ec2` – Allows principals to associate and disassociate the default instance profile with EC2 instances; to tag instances targeted by a patch policy configuration; to tag instances targeted by a patch policy configuration; and to help enable Explorer.
- `s3` – Allows principals to create and configure S3 buckets to store patch baseline overrides.
- `lambda` – Allows principals to invoke AWS Lambda functions that configure patching and to perform clean-up operations after a Quick Setup patch policy configuration is deleted.
- `logs` – Allows principals to configure logging for Patch Manager Quick Setup AWS Lambda functions.
- `config` – Allows principals to help enable Explorer by providing read-only access to configuration recorder details.
- `compute-optimizer` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.
- `support` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupPatchPolicyPermissionsBoundary in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupSchedulerPermissionsBoundary

> **Note**
>
> This policy is a *permissions boundary*. A permissions boundary sets the maximum permissions that an identity-based policy can grant to an IAM entity. You should not use and attach Quick Setup permissions boundary policies on your own. Quick Setup permissions boundary policies should only be attached to Quick Setup managed roles. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.

The managed policy `AWSQuickSetupSchedulerPermissionsBoundary` supports the Stop and start EC2 instances automatically on a schedule using Quick Setup Quick Setup configuration type.

This configuration type lets you stop and start your EC2 instances and other resources at the times you specify.

When you create an `AWSQuickSetupSchedulerPermissionsBoundary` configuration using Quick Setup, the system applies this permissions boundary to the IAM roles that are created when the configuration is deployed. The permissions boundary limits the scope of the roles that Quick Setup creates.

This policy grants administrative permissions that allow Quick Setup to enable and configure scheduled operations on EC2 instances and other resources.

Permissions details

This policy includes the following permissions.

- `iam` – Allows principals to retrieve and pass roles for instance management automation actions; to manage, pass, and attach default instance roles for EC2 instance management; to create default instance profiles; to add default instance roles to instance profiles; to create a service-linked role for Systems Manager; to read information about IAM roles and instance profiles; to associate a default instance profile with EC2 instances; and to start Automation workflows to configure instances and enable Systems Manager tools on them.
- `ssm` – Allows principals to start Automation workflows that enable Explorer; and to read and update Explorer service settings.
- ec2 – Allows principals to locate targeted instances and to start and stop them on a schedule.
- `config` – Allows principals to help enable Explorer by providing read-only access to configuration recorder details.
- `compute-optimizer` – Allows principals to help enable Explorer by providing read-only access to determine whether a resource is enrolled with AWS Compute Optimizer.
- `support` – Allows principals to help enable Explorer by providing read-only access to AWS Trusted Advisor checks for an account.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupSchedulerPermissionsBoundary in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupCFGCPacksPermissionsBoundary

> **Note**
>
> This policy is a *permissions boundary*. A permissions boundary sets the maximum permissions that an identity-based policy can grant to an IAM entity. You should not use and attach Quick Setup permissions boundary policies on your own. Quick Setup permissions boundary policies should only be attached to Quick Setup managed roles. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.

The managed policy `AWSQuickSetupCFGCPacksPermissionsBoundary`supports the Deploy AWS Config conformance pack using Quick Setup Quick Setup configuration type. This configuration type deploys AWS Config conformance packs. Conformance packs are collections of AWS Config rules and remediation actions that can be deployed as a single entity.

When you create an `AWSQuickSetupCFGCPacksPermissionsBoundary` configuration using Quick Setup, the system applies this permissions boundary to the IAM roles that are created when the configuration is deployed. The permissions boundary limits the scope of the roles that Quick Setup creates.

This policy grants administrative permissions that allow Quick Setup to deploy AWS Config conformance packs.

Permissions details

This policy includes the following permissions.

- `iam` – Allows principals to create, get, and pass a service-linked role for AWS Config.
- `sns` – Allows principals to list platform applications in Amazon SNS.
- `config` – Allows principals to deploy AWS Config conformance packs; to get the status of conformance packs; and to get information about configuration recorders.
- `ssm` – Allows principals to get information about SSM documents and Automation workflows; to get information about resource tags; and to get information about and update service settings.
- `compute-optimizer` – Allows principals to get the opt-in status of an account.
- `support` – Allows principals to get information about AWS Trusted Advisor checks.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupCFGCPacksPermissionsBoundary in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupStartStopInstancesExecutionPolicy

You can attach `AWSQuickSetupStartStopInstancesExecutionPolicy` to your IAM entities. This policy provides permissions for Quick Setup to manage the starting and stopping of Amazon EC2 instances using Systems Manager automation.

Permissions details

This policy includes the following permissions.

- `ec2` – Allows principals to view detailed information about Amazon EC2 instances, their status, Regions, and tags. Also allows starting and stopping Amazon EC2 instances.
- `tag` – Allows principals to retrieve tag information for Amazon EC2 instances to identify instances that should be managed by the automation.
- `ssm` – Allows principals to check calendar states for Quick Setup change calendars, and to execute Systems Manager automation and associations for instance management.
- `iam` – Allows principals to pass Quick Setup IAM roles to Systems Manager for automation execution, with specific conditions that restrict the service and resources that can use these roles.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupStartStopInstancesExecutionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupStartSSMAssociationsExecutionPolicy

This policy grants permissions that allow Quick Setup to run the `AWSQuickSetupType-StartSSMAssociations` Automation runbook. This runbook is used to start State Manager associations that are created by Quick Setup configurations.

You can attach `AWSQuickSetupStartSSMAssociationsExecutionPolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to start automation executions specifically for the `AWSQuickSetupType-StartSSMAssociations` document. This is required for Quick Setup to run associations that configure managed instances.
- `iam` – Allows principals to pass roles with names that begin with "AWS-QuickSetup-" to the Systems Manager service. This permission is restricted to use with specific SSM documents related to starting associations. This is required for Quick Setup to pass the appropriate execution role to the automation process.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupStartSSMAssociationsExecutionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy

The policy `AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy` provides permissions for diagnosing issues with nodes that interact with Systems Manager services by starting Automation workflows in accounts and Regions where nodes are managed.

You can attach `AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform diagnosis actions on your behalf.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to run specific Automation runbooks that diagnose node issues, access the execution status for workflows, and retrieve automation execution details. The policy grants permissions to describe automation executions, describe automation step executions, get automation execution details, and start automation executions for diagnosis-related documents.
- `kms` – Allows principals to use customer-specified AWS Key Management Service keys for decryption and data key generation when accessing encrypted objects in Amazon S3 buckets used for diagnosis operations. These permissions are restricted to keys tagged with `SystemsManagerManaged` and used via Amazon S3 service with specific encryption context requirements.
- `sts` – Allows principals to assume diagnosis execution roles to run Automation runbooks in the same account. This permission is restricted to roles with the `AWS-SSM-`

`DiagnosisExecutionRole` naming pattern and includes a condition to ensure the resource account matches the principal account.

- `iam` – Allows principals to pass the diagnosis administration role to Systems Manager to run Automation runbooks. This permission is restricted to roles with the `AWS-SSM-DiagnosisAdminRole` naming pattern and can only be passed to the Systems Manager service.
- `s3` – Allows principals to access, read, write, and delete objects in Amazon S3 buckets used for diagnosis operations. These permissions are restricted to buckets with the `do-not-delete-ssm-diagnosis-` naming pattern and include conditions to ensure operations are performed within the same account.

To view more details about the policy, including the latest version of the JSON policy document, see AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWS-SSM-DiagnosisAutomation-ExecutionRolePolicy

The managed policy `AWS-SSM-DiagnosisAutomation-ExecutionRolePolicy` provides administrative permission for running Automation runbooks in a targeted AWS account and Region to diagnose issues with managed nodes that interact with Systems Manager services.

You can attach `AWS-SSM-DiagnosisAutomation-ExecutionRolePolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

Permissions details

This policy includes the following permissions.

- `ec2` – Allows principals to describe Amazon EC2 and Amazon VPC resources and their configurations to diagnose issues with Systems Manager services. This includes permissions to describe VPCs, VPC attributes, VPC endpoints, subnets, security groups, instances, and internet gateways.
- `ssm` – Allows principals to run diagnosis-specific Automation runbooks and access the automation workflow status and execution metadata. This includes permissions to describe automation step executions, describe instance information, describe automation executions, get automation execution details, and start automation executions for specific AWS unmanaged EC2 diagnosis documents.

- kms – Allows principals to use customer-specified AWS Key Management Service keys for decryption and data key generation when accessing encrypted objects in Amazon S3 buckets used for diagnosis operations. These permissions are restricted to keys tagged with SystemsManagerManaged and used via Amazon S3 service with specific encryption context requirements for diagnosis buckets.
- iam – Allows principals to pass the diagnosis execution role to Systems Manager to run Automation documents. This permission is restricted to roles with the AWS-SSM-DiagnosisExecutionRole naming pattern and can only be passed to the Systems Manager service.

To view more details about the policy, including the latest version of the JSON policy document, see AWS-SSM-DiagnosisAutomation-ExecutionRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWS-SSM-RemediationAutomation-AdministrationRolePolicy

The policy AWS-SSM-RemediationAutomation-AdministrationRolePolicy provides permissions for remediating issues with Systems Manager services by executing activities defined within Automation documents, primarily used for running the Automation documents. This policy enables starting Automation workflows in accounts and Regions where nodes are managed to address connectivity and configuration issues.

You can attach AWS-SSM-RemediationAutomation-AdministrationRolePolicy to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform remediation actions on your behalf.

Permissions details

This policy includes the following permissions.

- ssm – Allows principals to run specific Automation runbooks that remediate node issues, access the execution status for workflows, and retrieve automation execution details. The policy grants permissions to describe automation executions, describe automation step executions, get automation execution details, and start automation executions for remediation-related documents.
- kms – Allows principals to use customer-specified AWS Key Management Service keys for decryption and data key generation when accessing encrypted objects in Amazon S3 buckets

used for remediation operations. These permissions are restricted to keys tagged with `SystemsManagerManaged` and used via Amazon S3 service with specific encryption context requirements.

- `sts` – Allows principals to assume remediation execution roles to run Automation runbooks in the same account. This permission is restricted to roles with the `AWS-SSM-RemediationExecutionRole` naming pattern and includes a condition to ensure the resource account matches the principal account.
- `iam` – Allows principals to pass the remediation administration role to Systems Manager to run Automation runbooks. This permission is restricted to roles with the `AWS-SSM-RemediationAdminRole` naming pattern and can only be passed to the Systems Manager service.
- `s3` – Allows principals to access, read, write, and delete objects in Amazon S3 buckets used for remediation operations. These permissions are restricted to buckets with the `do-not-delete-ssm-diagnosis-` naming pattern and include conditions to ensure operations are performed within the same account.

To view more details about the policy, including the latest version of the JSON policy document, see AWS-SSM-RemediationAutomation-AdministrationRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWS-SSM-RemediationAutomation-ExecutionRolePolicy

The managed policy `AWS-SSM-RemediationAutomation-ExecutionRolePolicy` provides permissions for running Automation runbooks in a specific target account and Region to remediate networking and connectivity issues with managed nodes that interact with Systems Manager services. This policy enables remediation activities defined within Automation documents, primarily used for running the Automation documents to address connectivity and configuration issues.

You can attach the policy to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform remediation actions on your behalf.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to retrieve information about Automation executions and their step executions, and to start specific remediation Automation runbooks including `AWS-`

`OrchestrateUnmanagedEC2Actions` and `AWS-RemediateSSMAgent` documents. The policy grants permissions to describe automation executions, describe automation step executions, get automation execution details, and start automation executions for remediation-related documents.

- `ec2` – Allows principals to describe and modify Amazon VPC networking resources to remediate connectivity issues. This includes:
  - Describing Amazon VPC attributes, subnets, Amazon VPC endpoints, and security groups.
  - Creating Amazon VPC endpoints for Systems Manager services (`ssm`, `ssmmessages`, and `ec2messages`) with required tags.
  - Modifying Amazon VPC attributes to enable DNS support and hostnames.
  - Creating and managing security groups with specific tags for Amazon VPC endpoint access.
  - Authorizing and revoking security group rules for HTTPS access with appropriate tags.
  - Creating tags on Amazon VPC endpoints, security groups, and security group rules during resource creation.
- `kms` – Allows principals to use customer-specified AWS Key Management Service keys for decryption and data key generation when accessing encrypted objects in Amazon S3 buckets used for remediation operations. These permissions are restricted to keys tagged with `SystemsManagerManaged` and used via Amazon S3 service with specific encryption context requirements.
- `iam` – Allows principals to pass the remediation execution role to Systems Manager to run Automation runbooks. This permission is restricted to roles with the `AWS-SSM-RemediationExecutionRole` naming pattern and can only be passed to the Systems Manager service.

To view more details about the policy, including the latest version of the JSON policy document, see AWS-SSM-RemediationAutomation-ExecutionRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupSSMManageResourcesExecutionPolicy

This policy grants permissions that allow Quick Setup to run the `AWSQuickSetupType-SSM-SetupResources` Automation runbook. This runbook creates IAM roles for Quick Setup associations, which in turn are created by a `AWSQuickSetupType-SSM` deployment. It also grants permissions to clean up an associated Amazon S3 bucket on during a Quick Setup delete operation.

You can attach the policy to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

Permissions details

This policy includes the following permissions.

- `iam` – Allows principals to list and manage IAM roles for use with Quick Setup Systems Manager Explorer operations; to view, attach, and detach IAM policies for use with Quick Setup and Systems Manager Explorer These permissions are required so Quick Setup can create the roles needed for some of its configuration operations.
- `s3` – Allows principals to retrieve information about objects in, and to delete objects from Amazon S3 buckets, in the principal account, that are used specifically in Quick Setup configuration operations. This is required so that S3 objects that are no longer needed after configuration can be removed.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupSSMManageResourcesExecutionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupSSMLifecycleManagementExecutionPolicy

The `AWSQuickSetupSSMLifecycleManagementExecutionPolicy` policy grants administrative permissions that allow Quick Setup to run the a AWS CloudFormation custom resource on lifecycle events during Quick Setup deployment in Systems Manager.

You can attach this policy to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to get information about automation executions and start automation executions for setting up certain Quick Setup operations.
- `iam` – Allows principals to pass roles from IAM for setting up certain Quick Setup resources.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupSSMLifecycleManagementExecutionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupSSMDeploymentRolePolicy

The managed policy `AWSQuickSetupSSMDeploymentRolePolicy` grants administrative permissions that allow Quick Setup to create resources that are used during the Systems Manager onboarding process.

Though you can manually attach this policy to your IAM entities, this is not recommended. Quick Setup creates entities that attach this policy to a service role that allows Systems Manager to perform actions on your behalf.

This policy is not related to the `SSMQuickSetupRolePolicy` policy, which is used to provide permissions for the `AWSServiceRoleForSSMQuickSetup` service-linked role.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to manage associations for certain resources that are created using AWS CloudFormation templates and a specific set of SSM documents; to manage roles and role policies using for diagnosing and remediating managed nodes through AWS CloudFormation templates; and to attach and delete policies for Quick Setup lifecycle events
- `iam` – Allows principals to tag roles and pass roles permissions for the Systems Manager service and Lambda service, and to pass role permissions for diagnosis operations.
- `lambda` – Allows principals to tag and manage functions for the Quick Setup lifecycle in the principal account using AWS CloudFormation templates.
- `cloudformation` – Allows principals to read information from AWS CloudFormation. This is required so Quick Setup can gather data about the AWS CloudFormation stacks used to manage the state of resources and CloudFormation stackset operations.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupSSMDeploymentRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupSSMDeploymentS3BucketRolePolicy

The `AWSQuickSetupSSMDeploymentS3BucketRolePolicy` policy grants permissions for listing all S3 buckets in an account; and for managing and retrieving information about specific buckets in the principal account that are managed through AWS CloudFormation templates.

You can attach `AWSQuickSetupSSMDeploymentS3BucketRolePolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

Permissions details

This policy includes the following permissions.

- `s3` – Allows principals list all S3 buckets in an account; and to manage and retrieve information about specific buckets in the principal account that are managed through AWS CloudFormation templates.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupSSMDeploymentS3BucketRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupEnableDHMCExecutionPolicy

This policy grants administrative permissions that allow principals to run the `AWSQuickSetupType-EnableDHMC` Automation runbook, which enables Default Host Management Configuration. The Default Host Management Configuration setting allows Systems Manager to automatically manage Amazon EC2 instances as *managed instances*. A managed instance is an EC2 instance that is configured for use with Systems Manager. This policy also grants permissions for creating IAM roles that are specified in Systems Manager service settings as the default roles for SSM Agent.

You can attach `AWSQuickSetupEnableDHMCExecutionPolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to update and get information about Systems Manager service settings.
- `iam` – Allows principals to create and retrieve information about IAM roles for Quick Setup operations.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupEnableDHMCExecutionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupEnableAREXExecutionPolicy

This policy grants administrative permissions that allow Systems Manager to run the `AWSQuickSetupType-EnableAREX` Automation runbook, which enables AWS Resource Explorer for use with Systems Manager. Resource Explorer makes it possible to view resources in your account with a search experience similar to an Internet search engine. The policy also grants permissions for managing Resource Explorer indexes and views.

You can attach `AWSQuickSetupEnableAREXExecutionPolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

Permissions details

This policy includes the following permissions.

- `iam` – Allows principals to to create a service-linked role in the AWS Identity and Access Management (IAM) service.
- `resource-explorer-2` – Allows principals to retrieve information about Resource Explorer views and indexes; to create Resource Explorer views and indexes; to change the index type for indexes displayed in Quick Setup.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupEnableAREXExecutionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupManagedInstanceProfileExecutionPolicy

This policy grants administrative permissions that allow Systems Manager to create a default IAM instance profile for the Quick Setup tool, and to attach it to Amazon EC2 instances that don't already have an instance profile attached. The policy also grants Systems Manager the ability to attach permissions to existing instance profiles. This is done to ensure that the permissions required for Systems Manager to communicate with SSM Agent on EC2 instances are in place.

You can attach `AWSQuickSetupManagedInstanceProfileExecutionPolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to start automation workflows associated with Quick Setup processes.
- `ec2` – Allows principals to attach IAM instance profiles to EC2 instances that are managed by Quick Setup.
- `iam` – Allows principals to create, update, and retrieve information about roles from IAM that are used in Quick Setup processes; to create IAM instance profiles; to attach the `AmazonSSMManagedInstanceCore` managed policy to IAM instance profiles.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupManagedInstanceProfileExecutionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupReadOnlyAccess

This policy grants read-only permissions that allow principals to view AWS Systems Manager Quick Setup data and reports, including information from other AWS service resources that are required for Quick Setup operations.

You can attach the `AWSQuickSetupReadOnlyAccess` policy to your IAM identities.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to read SSM Command documents and Automation runbooks; and to retrieve the status of State Manager association executions.
- `cloudformation` – Allows principals to initiate operations that are required for retrieving the status of AWS CloudFormation deployments.
- `organizations` – Allows principals to read the status of accounts in an AWS Organizations organization.
- `ssm-quicksetup` – Allows principals to perform read-only actions in Quick Setup.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupReadOnlyAccess in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupManageJITNAResourcesExecutionPolicy

The managed policy `AWSQuickSetupManageJITNAResourcesExecutionPolicy` enables Quick Setup, a tool in Systems Manager, to set up just-in-time node access.

You can attach `AWSQuickSetupManageJITNAResourcesExecutionPolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

This policy grants administrative permissions that allow Systems Manager to create resources associated with just-in-time node access.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to get and update the service setting that specifies the identity provider for just-in-time node access.
- `iam` – Allows principals to create, tag, and get roles, attach role policies for just-in-time node access managed policies, and create service-linked roles for just-in-time node access and notifications.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupManageJITNAResourcesExecutionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSQuickSetupJITNADeploymentRolePolicy

The managed policy `AWSQuickSetupJITNADeploymentRolePolicy` allows Quick Setup to deploy the configuration type required to set up just-in-time node access.

You can attach `AWSQuickSetupJITNADeploymentRolePolicy` to your IAM entities. Systems Manager also attaches this policy to a service role that allows Systems Manager to perform actions on your behalf.

This policy grants administrative permissions that allow Systems Manager to create resources associated with just-in-time node access.

Permissions details

This policy includes the following permissions.

- `cloudformation` – Allows principals to create, update, delete, and read AWS CloudFormation stacks.
- `ssm` – Allows principals to create, delete, update, and read State Manager associations that are called by AWS CloudFormation.
- `iam` – Allows principals create, delete, read and tag IAM roles that are called by AWS CloudFormation.

To view more details about the policy, including the latest version of the JSON policy document, see AWSQuickSetupJITNADeploymentRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSSystemsManagerJustInTimeAccessServicePolicy

The managed policy `AWSSystemsManagerJustInTimeAccessServicePolicy` provides access to AWS resources managed or used by the AWS Systems Manager just-in-time access framework. This policy update adds automation execution tagging permissions to enable customers to scope down operator permissions to specific tags.

You can't attach `AWSSystemsManagerJustInTimeAccessServicePolicy` 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 enable just-in-time node access.

This policy grants administrative permissions that allows access to resources associated with just-in-time node access.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to create and manage OpsItems, add tags to OpsItems and automation executions, get and update OpsItems, retrieve and describe documents, describe OpsItems and sessions, list documents and tags for managed instances.
- `ssm-guiconnect` – Allows principals to list connections.
- `identitystore` – Allows principals to get user and group IDs, describe users, and list group membership.

- `sso-directory` – Allows principals to describe users and determine if a user is a member of a group.
- `sso` – Allows principals to describe registered Regions and list instances and directory associations.
- `cloudwatch` – Allows principals to put metric data for the `AWS/SSM/JustInTimeAccess` namespace.
- `ec2` – Allows principals to describe tags.

To view more details about the policy, including the latest version of the JSON policy document, see AWSSystemsManagerJustInTimeAccessServicePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSSystemsManagerJustInTimeAccessTokenPolicy

The managed policy `AWSSystemsManagerJustInTimeAccessTokenPolicy` allows Systems Manager to generate access tokens used for just-in-time node access.

You can attach `AWSSystemsManagerJustInTimeAccessTokenPolicy` to your IAM entities.

This policy grants administrative permissions that allow Systems Manager to start sessions and generate access tokens for just-in-time node access.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to send commands, list command invocations, and start, resume, and end Session Manager sessions.
- `ssm-guiconnect` – Allows principals to start, describe, and end Systems Manager GUI Connect RDP connections.
- `kms` – Allows principals to create grants and generate key data.
- `sso` – Allows principals to list directory associations.
- `identitystore` – Allows principals to describe a user.

To view more details about the policy, including the latest version of the JSON policy document, see AWSSystemsManagerJustInTimeAccessTokenPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSSystemsManagerJustInTimeAccessTokenSessionPolicy

The managed policy `AWSSystemsManagerJustInTimeAccessTokenSessionPolicy` allows Systems Manager to apply scoped down permissions to a just-in-time node access token.

You can attach `AWSSystemsManagerJustInTimeAccessTokenSessionPolicy` to your IAM entities.

This policy grants administrative permissions that allow Systems Manager to scope down permissions for just-in-time node access tokens.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to start Session Manager sessions using the `SSM-SessionManagerRunShell` document. Also when called first via `ssm-guiconnect`, start sessions using the `AWS-StartPortForwardingSession` document, list command invocations, and send commands using the `AWSSSO-CreateSSOUser` document.
- `ssm-guiconnect` – Allows principals to cancel, get, and start connections on all resources.
- `kms` – Allows principals to create grants and generate data keys for keys tagged with `SystemsManagerJustInTimeNodeAccessManaged` when called via `ssm-guiconnect` through an AWS service.
- `sso` – Allows principals to list directory associations when called via `ssm-guiconnect`.
- `identitystore` – Allows principals to describe a user when called via `ssm-guiconnect`.

To view more details about the policy, including the latest version of the JSON policy document, see AWSSystemsManagerJustInTimeAccessTokenSessionPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy

The managed policy `AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy` allows Systems Manager to share deny-access policies from the delegated administrator account to member accounts, and replicate the policies across multiple AWS Regions.

You can attach `AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy` to your IAM entities.

This policy provides the administrative permissions necessary for Systems Manager to share and create deny-access policies. This ensures that deny-access policies are applied to all accounts in an AWS Organizations organization and Regions configured for just-in-time node access.

Permissions details

This policy includes the following permissions.

- `ssm` – Allows principals to manage SSM documents and resource policies.
- `ssm-quicksetup` – Allows principals to read Quick Setup configuration managers.
- `organizations` – Allows principals to list details about an AWS Organizations organization and delegated administrators.
- `ram` – Allows principals to create, tag, and describe resource shares.
- `iam` – Allows principals to describe a service role.

To view more details about the policy, including the latest version of the JSON policy document, see AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSSystemsManagerNotificationsServicePolicy

The managed policy `AWSSystemsManagerNotificationsServicePolicy` allows Systems Manager to send email notifications for just-in-time node access requests to access request approvers.

You can't attach `AWSSystemsManagerJustInTimeAccessServicePolicy` 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 send just-in-time node access request notifications.

This policy grants administrative permissions that allow Systems Manager to send email notifications for just-in-time node access requests to access request approvers.

Permissions details

This policy includes the following permissions.

- `identitystore` – Allows principals to list and describe users and group membership.

- `sso` – Allows principals to list instances, directories, and describe registered Regions.
- `sso-directory` – Allows principals to describe users and list members in a group.
- `iam` – Allows principals to get information about roles.

To view more details about the policy, including the latest version of the JSON policy document, see AWSSystemsManagerNotificationsServicePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWS-SSM-Automation-DiagnosisBucketPolicy

The managed policy `AWS-SSM-Automation-DiagnosisBucketPolicy` provides permissions for diagnosing issues with nodes that interact with AWS Systems Manager services, by allowing access to S3 buckets that are used for diagnosis and remediation of issues.

You can attach the `AWS-SSM-Automation-DiagnosisBucketPolicy` policy to your IAM identities. Systems Manager also attaches this policy to an IAM role that allows Systems Manager to perform diagnosis actions on your behalf.

Permissions details

This policy includes the following permissions.

- `s3` – Allows principals to access and write objects to an Amazon S3 bucket.

To view more details about the policy, including the latest version of the JSON policy document, see AWS-SSM-Automation-DiagnosisBucketPolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWS-SSM-RemediationAutomation-OperationalAccountAdministrationRolePolicy

The managed policy `AWS-SSM-RemediationAutomation-OperationalAccountAdministrationRolePolicy` provides permissions for an operational account to diagnose issues with nodes by providing organization-specific permissions.

You can attach `AWS-SSM-RemediationAutomation-OperationalAccountAdministrationRolePolicy` to your IAM identities. Systems Manager also attaches this policy to an IAM role that allows Systems Manager to perform diagnosis actions on your behalf.

Permissions details

This policy includes the following permissions.

- `organizations` – Allows principals to list a root of the organization, and get member accounts to determine target accounts.
- `sts` – Allows principals to assume remediation execution roles to run SSM Automation documents across accounts and Regions, within the same organization.

To view more details about the policy, including the latest version of the JSON policy document, see AWS-SSM-RemediationAutomation-OperationalAccountAdministrationRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWS-SSM-DiagnosisAutomation-OperationalAccountAdministrationRolePolicy

The managed policy `AWS-SSM-DiagnosisAutomation-OperationalAccountAdministrationRolePolicy` provides permissions for an operational account to diagnose issues with nodes by providing organization-specific permissions.

You can attach the `AWS-SSM-DiagnosisAutomation-OperationalAccountAdministrationRolePolicy` policy to your IAM identities. Systems Manager also attaches this policy to an IAM role that allows Systems Manager to perform diagnosis actions on your behalf.

Permissions details

This policy includes the following permissions.

- `organizations` – Allows principals to list a root of the organization, and get member accounts to determine target accounts.
- `sts` – Allows principals to assume diagnosis execution roles to run SSM Automation documents across accounts and Regions, within the same organization.

To view more details about the policy, including the latest version of the JSON policy document, see AWS-SSM-DiagnosisAutomation-OperationalAccountAdministrationRolePolicy in the *AWS Managed Policy Reference Guide*.

Systems Manager updates to AWS managed policies

In the following table, view details about updates to AWS managed policies for Systems Manager since this service began tracking these changes on March 12, 2021. For information about other managed policies for the Systems Manager service, see Additional managed policies for Systems Manager later in this topic. For automatic alerts about changes to this page, subscribe to the RSS feed on the Systems Manager Document history page.

Change	Description	Date
AWSSystemsManagerJustInTimeAccessServicePolicy – Updated managed policy	Systems Manager updated the managed policy to add automation execution tagging permissions. The service needs to tag automation executions with `SystemsManagerJustInTimeNodeAccessManaged=true` tag to enable customers to scope down operator permissions to specific tags.	August 25, 2025
AWSQuickSetupStartSSMAssociationsExecutionPolicy – New policy	Systems Manager added a new policy to allow Quick Setup to run the `AWSQuickSetupType-StartSSMAssociations` Automation runbook. This runbook is used to start State Manager associations that are created by Quick Setup configurations.	August 12, 2025
AWSQuickSetupStartStopInstancesExecutionPolicy – New policy	Systems Manager added a new policy to allow Quick Setup to start and stop	August 12, 2025

Change	Description	Date
	Amazon EC2 instances on a schedule. This policy provides the necessary permissions for the Quick Setup scheduler configuration type to manage instance state based on defined schedules.	
AWSQuickSetupDeploymentRolePolicy – Update to documentation	Systems Manager has updated the `AWSQuickSetupDeploymentRolePolicy` managed policy to grant permissions for additional resources. In addition, the documentation for `AWSQuickSetupDeploymentRolePolicy` has been updated with more detailed descriptions of the permissions granted by this policy for Quick Setup configuration management operations.	August 12, 2025

Change	Description	Date
AWS-SSM-RemediationAutomation-ExecutionRolePolicy – Update to an existing policy	Systems Manager updated the managed policy to improve the security posture of the ssm:StartAutomationExecution API by requiring permissions for both "document" and "automation-execution" resource types. The updated policy provides more comprehensive and detailed permissions for remediation automation execution, including enhanced descriptions for networking remediation capabilities, more specific Amazon VPC endpoint creation permissions, detailed security group management permissions, and improved resource tagging controls for remediation operations.	July 16th, 2025

Change	Description	Date
AWS-SSM-RemediationAutomation-AdministrationRolePolicy – Update to an existing policy	Systems Manager updated the managed policy to support API authorization improvements for remediation automation operations. The updated policy enhances permissions for executing activities defined within Automation documents, with improved security controls and resource access patterns for remediation workflows.	July 16th, 2025
AWS-SSM-DiagnosisAutomation-ExecutionRolePolicy – Update to an existing policy	Systems Manager updated the managed policy to provide more detailed and accurate permissions for diagnosis automation execution. The updated policy includes enhanced descriptions for Amazon EC2 and Amazon VPC resource access, more specific SSM automation permissions, and improved AWS KMS and IAM permission descriptions with proper resource restrictions.	July 16th, 2025

Change	Description	Date
AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy – Update to an existing policy	Systems Manager updated the managed policy to provide more specific permissions and security conditions for diagnosis automation operations. The updated policy provides enhanced security controls for AWS KMS key usage, Amazon S3 bucket access, and role assumptions, with stricter resource-based conditions and account-level restrictions.	July 16th, 2025
AWSQuickSetupDeploymentRolePolicy – Update to a policy	Systems Manager added permissions to the managed policy `AWSQuickSetupDeploymentRolePolicy` for accessing the Amazon owned runbook AWSQuickSetupType-ManageInstanceProfile. This permission makes it possible for Quick Setup to create associations using the managed policy instead of inline policies.	July 14th, 2025

Change	Description	Date
AmazonSSMAutomationRole – Update to documentation	Systems Manager added comprehensive documentation for the existing `AmazonSSMAutomationRole` policy, which provides permissions for the Systems Manager Automation service to run activities defined within Automation runbooks.	July 15, 2025
AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy – Update to an policy	Systems Manager added permissions to allow Systems Manager to tag a resource shared by AWS Resource Access Manager for just-in-time node access.	April 30th, 2025
AWSQuickSetupManageJITNAResourcesExecutionPolicy – Update to a policy	Systems Manager added permissions to allow Systems Manager to tag IAM roles created for just-in-time node access.	April 30th, 2025
AWSSystemsManagerJustInTimeAccessTokenSessionPolicy – New policy	Systems Manager added a new policy to allow Systems Manager to apply scoped down permissions to a just-in-time node access token.	April 30th, 2025

Change	Description	Date
AWSSystemsManagerNotificationsServicePolicy – New policy	Systems Manager added a new policy to allow Systems Manager to send email notifications for just-in-time node access requests to access request approvers.	April 30th, 2025
AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy – New policy	Systems Manager added a new policy to allow Systems Manager to replicate approval policies to different Regions.	April 30th, 2025
AWSSystemsManagerJustInTimeAccessTokenPolicy – New policy	Systems Manager added a new policy to allow Systems Manager to generate access tokens used for just-in-time node access.	April 30th, 2025
AWSSystemsManagerJustInTimeAccessServicePolicy – New policy	Systems Manager added a new policy to provide permissions to AWS resources managed or used by the Systems Manager just-in-time node access feature.	April 30th, 2025
AWSQuickSetupManageJITNAResourcesExecutionPolicy – New policy	Systems Manager added a new policy to allow Quick Setup, a tool in Systems Manager, to create the IAM roles necessary for just-in-time node access.	April 30th, 2025

Change	Description	Date
AWSQuickSetupJITNADeploymentRolePolicy – New policy	Systems Manager added a new policy that provides permissions that allow Quick Setup to deploy the configuration type required to set up just-in-time node access.	April 30th, 2025
AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy – Update to an policy	Systems Manager added permissions to allow Systems Manager to tag a resource shared by AWS Resource Access Manager for just-in-time node access.	April 30th, 2025
AWSQuickSetupManageJITNAResourcesExecutionPolicy – Update to an policy	Systems Manager added permissions to allow Systems Manager to tag IAM roles created for just-in-time node access.	April 30th, 2025
AWSSystemsManagerJustInTimeAccessTokenSessionPolicy – New policy	Systems Manager added a new policy to allow Systems Manager to apply scoped down permissions to a just-in-time node access token.	April 30th, 2025
AWSSystemsManagerNotificationsServicePolicy – New policy	Systems Manager added a new policy to allow Systems Manager to send email notifications for just-in-time node access requests to access request approvers.	April 30th, 2025

Change	Description	Date
AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy – New policy	Systems Manager added a new policy to allow Systems Manager to replicate approval policies to different Regions.	April 30th, 2025
AWSSystemsManagerJustInTimeAccessTokenPolicy – New policy	Systems Manager added a new policy to allow Systems Manager to generate access tokens used for just-in-time node access.	April 30th, 2025
AWSSystemsManagerJustInTimeAccessServicePolicy – New policy	Systems Manager added a new policy to provide permissions to AWS resources managed or used by the Systems Manager just-in-time node access feature.	April 30th, 2025
AWSQuickSetupManageJITNAResourcesExecutionPolicy – New policy	Systems Manager added a new policy to allow Quick Setup, a tool in Systems Manager, to create the IAM roles necessary for just-in-time node access.	April 30th, 2025
AWSQuickSetupJITNADeploymentRolePolicy – New policy	Systems Manager added a new policy that provides permissions that allow Quick Setup to deploy the configuration type required to set up just-in-time node access.	April 30th, 2025

Change	Description	Date
AWS-SSM-DiagnosisAutomation-OperationalAccountAdministrationRolePolicy – New policy	Systems Manager added a new policy that provides permissions for an operational account to diagnose issues with nodes by providing organization-specific permissions.	November 21, 2024
AWS-SSM-RemediationAutomation-OperationalAccountAdministrationRolePolicy – New policy	Systems Manager added a new policy that provides permissions for an operational account to diagnose issues with nodes by providing organization-specific permissions.	November 21, 2024
AWS-SSM-Automation-DiagnosisBucketPolicy – New policy	Systems Manager added a new policy to support starting Automation workflows that diagnose issues with managed nodes in targeted accounts and Regions.	November 21, 2024
AmazonSSMServiceRolePolicy – Update to an existing policy	Systems Manager added new permissions to allow AWS Resource Explorer to gather details about Amazon EC2 instances and display the results in widgets in the new Systems Manager Dashboard.	November 21, 2024

Change	Description	Date
SSMQuickSetupRolePolicy – Update to an existing policy	Systems Manager has updated the managed policy SSMQuickSetupRolePolicy . This updates allows the associated service-linked role AWSServiceRoleForSSMQuickSetup to manage resource data syncs.	November 21, 2024
AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy – New policy	Systems Manager added a new policy to support starting Automation workflows that diagnose issues with managed nodes in targeted account and Regions.	November 21, 2024
AWS-SSM-DiagnosisAutomation-ExecutionRolePolicy – New policy	Systems Manager added a new policy to support starting Automation workflows that diagnose issues with managed nodes in a targeted account and Region.	November 21, 2024
AWS-SSM-RemediationAutomation-AdministrationRolePolicy – New policy	Systems Manager added a new policy to support starting Automation workflows that remediate issues in managed nodes in targeted accounts and Regions.	November 21, 2024

Change	Description	Date
AWS-SSM-RemediationAutomation-ExecutionRolePolicy – New policy	Systems Manager added a new policy to support starting Automation workflows that remediate issues in managed nodes in a targeted account and Region.	November 21, 2024
AWSQuickSetupSSMDeploymentRolePolicy – Update to an policy	Systems Manager added permissions to allow Systems Manager to tag IAM roles and Lambda created for the unified console.	May 7th, 2025
AWSQuickSetupSSMManageResourcesExecutionPolicy – New policy	Systems Manager added a new policy to support running an operation in Quick Setup that creates IAM roles for Quick Setup associations, which in turn are created by a AWSQuickSetupType-SSM deployment.	November 21, 2024
AWSQuickSetupSSMLifecycleManagementExecutionPolicy – New policy	Systems Manager added a new policy to support Quick Setup running a AWS CloudFormation custom resource on lifecycle events during a Quick Setup deployment.	November 21, 2024

Change	Description	Date
AWSQuickSetupSSMDeploymentRolePolicy – New policy	Systems Manager added a new policy to support granting administrative permissions that allow Quick Setup to create resources that are using during the Systems Manager onboarding process.	November 21, 2024
AWSQuickSetupSSMDeploymentS3BucketRolePolicy – New policy	Systems Manager added a new policy to support managing and retrieving information about specific buckets in the principal account that are managed through AWS CloudFormation templates	November 21, 2024
AWSQuickSetupEnableDHMCExecutionPolicy – New policy	Systems Manager is introducing a new policy to allow Quick Setup to create an IAM role that itself uses the existing AmazonSSMManagedEC2InstanceDefaultPolicy. This policy contains all the permissions required for SSM Agent to communicate with Systems Manager service. The new policy also allows modifications to the Systems Manager service settings.	November 21, 2024

Change	Description	Date
AWSQuickSetupEnableAREXExecutionPolicy – New policy	Systems Manager added a new policy to allow Quick Setup to create a service-linked role for AWS Resource Explorer, for accessing Resource Explorer views and aggregator indexes.	November 21, 2024
AWSQuickSetupManagedInstanceProfileExecutionPolicy – New policy	Systems Manager added a new policy to allow Quick Setup to create a default Quick Setup instance profile and to attach it to any Amazon EC2 instances that lack an associated instance profile. This new policy also allows Quick Setup to attach permissions to existing profiles to ensure that all required Systems Manager permissions have been granted.	November 21, 2024
AWSQuickSetupReadOnlyAccess – New policy	Systems Manager added a new policy to grant read-only permissions that allow principals to view AWS Systems Manager Quick Setup data and reports, including information from other AWS service resources that are required for Quick Setup operations.	November 21, 2024

Change	Description	Date
SSMQuickSetupRolePolicy – Update to an existing policy	Systems Manager added new permissions to allow Quick Setup to check the health of additional AWS CloudFormation stack sets that it has created.	August 13, 2024
AmazonSSMManagedEC2InstanceDefaultPolicy – Update to an existing policy	Systems Manager has added statement IDs (Sids) to the JSON policy for AmazonSSMManagedEC2InstanceDefaultPolicy. These Sids provide inline descriptions of the purpose of each policy statement.	July 18, 2024
SSMQuickSetupRolePolicy – New policy	Systems Manager added a new policy to allow Quick Setup to check the health of deployed resources and remediate instances that have drifted from the original configuration.	July 3, 2024
AWSQuickSetupDeploymentRolePolicy – New policy	Systems Manager added a new policy to support multiple Quick Setup configuration types that create IAM roles and automations, which in turn configure frequently used Amazon Web Services services and features with recommended best practices.	July 3, 2024

Change	Description	Date
AWSQuickSetupPatchPolicyDeploymentRolePolicy – New policy	Systems Manager added a new policy to allow Quick Setup to create resources associated with Patch Manager patch policy Quick Setup configurations.	July 3, 2024
AWSQuickSetupPatchPolicyBaselineAccess – New policy	Systems Manager added a new policy to allow Quick Setup to access patch baselines in Patch Manager with read-only permissions.	July 3, 2024
AWSSystemsManagerEnableExplorerExecutionPolicy – New policy	Systems Manager added a new policy to allow Quick Setup to grant administrative permissions for enabling Explorer.	July 3, 2024
AWSSystemsManagerEnableConfigRecordingExecutionPolicy – New policy	Systems Manager added a new policy to allow Quick Setup to enable and configure AWS Config configuration recording.	July 3, 2024
AWSQuickSetupDevOpsGuruPermissionsBoundary – New policy	Systems Manager added a new policy to allow Quick Setup to enable and configure Amazon DevOps Guru.	July 3, 2024
AWSQuickSetupDistributorPermissionsBoundary – New policy	Systems Manager added a new policy to allow Quick Setup to enable and configure Distributor, a tool in AWS Systems Manager.	July 3, 2024

Change	Description	Date
AWSQuickSetupSSMHostMgmtPermissionsBoundary – New policy	Systems Manager added a new policy to allow Quick Setup to enable and configure Systems Manager tools for securely managing Amazon EC2 instances.	July 3, 2024
AWSQuickSetupPatchPolicyPermissionsBoundary – New policy	Systems Manager added a new policy to allow Quick Setup to enable and configure patch policies in Patch Manager, a tool in AWS Systems Manager.	July 3, 2024
AWSQuickSetupSchedulerPermissionsBoundary – New policy	Systems Manager added a new policy to allow Quick Setup to enable and configure scheduled operations on Amazon EC2 instances and other resources.	July 3, 2024
AWSQuickSetupCFGCPacksPermissionsBoundary – New policy	Systems Manager added a new policy to allow Quick Setup to deploy AWS Config conformance packs.	July 3, 2024
AWSSystemsManagerOpsDataSyncServiceRolePolicy – Update to an existing policy	OpsCenter updated the policy to improve the security of the service code within the service-linked role for Explorer to manage OpsData-related operations.	July 3, 2023

Change	Description	Date
AmazonSSMManagedEC2InstanceDefaultPolicy – New policy	Systems Manager added a new policy to allow Systems Manager functionality on Amazon EC2 instances without the use of an IAM instance profile.	August 18, 2022
AmazonSSMServiceRolePolicy – 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 that config and the compute-optimizer meet the necessary requirements before allowing OpsData.	April 27, 2021
AWSSystemsManagerOpsDataSyncServiceRolePolicy – 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
AmazonSSMServiceRolePolicy – 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

Change	Description	Date
Systems Manager started tracking changes	Systems Manager started tracking changes for its AWS managed policies.	March 12, 2021

Additional managed policies for Systems Manager

In addition to the managed policies described earlier in this topic, the following policies are also supported by Systems Manager.

- `AmazonSSMAutomationApproverAccess` – AWS managed policy that allows access to view automation executions and send approval decisions to automation that is waiting for approval.
- `AmazonSSMDirectoryServiceAccess` – AWS managed policy that that allows SSM Agent to access AWS Directory Service on behalf of the user for requests to join the domain by the managed node.
- `AmazonSSMFullAccess` – AWS managed policy that grants full access to the Systems Manager API and documents.
- `AmazonSSMMaintenanceWindowRole` – AWS managed policy that provides maintenance windows with permissions to the Systems Manager API.
- `AmazonSSMManagedInstanceCore` – AWS managed policy that allows a node to use Systems Manager service core functionality.
- `AmazonSSMPatchAssociation` – AWS managed policy that provides access to child instances for patch association operations.
- `AmazonSSMReadOnlyAccess` – AWS managed policy that grants access to Systems Manager read-only API operations, such as `Get*` and `List*`.
- `AWSSSMOpsInsightsServiceRolePolicy` – AWS managed policy that provides permissions for creating and updating operational insight *OpsItems* in Systems Manager. Used to provide permissions through the service-linked role `AWSServiceRoleForAmazonSSM_OpsInsights`.
- `AWSSystemsManagerAccountDiscoveryServicePolicy` – AWS managed policy that grants Systems Manager permission to discover AWS account information.
- `AmazonEC2RoleforSSM` – This policy is no longer supported and should not be used. In its place, use the `AmazonSSMManagedInstanceCore` policy to allow Systems Manager service core

functionality on EC2 instances. For information, see Configure instance permissions required for Systems Manager.

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
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Systems Manager resources

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 sign-in credentials.

The following example error occurs when the `mateojackson` 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, your policies must be updated 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 that are 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's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

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.
- 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 Cross account resource access in IAM 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.

> **Note**
>
> A *service role* role differs from a service-linked 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.

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.

> **Note**
>
> For non-EC2 nodes in a hybrid and multicloud environment , you need an additional IAM role that allows those machines 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 the IAM service role required for Systems Manager in hybrid and multicloud environments and AssumeRole in the *AWS Security Token Service API Reference*.

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.

Topics

- Using roles to collect inventory and view OpsData
- Using roles to collect AWS account information for OpsCenter and Explorer
- Using roles to create OpsData and OpsItems for Explorer
- Using roles to create operational insight OpsItems in Systems Manager OpsCenter
- Using roles to maintain Quick Setup-provisioned resource health and consistency
- Using roles to export Explorer OpsData
- Using roles to enable just-in-time node access
- Using roles to send just-in-time node access request notifications

Using roles to collect inventory and view OpsData

Systems Manager uses the service-linked role named **`AWSServiceRoleForAmazonSSM`**. AWS Systems Manager uses this IAM service role to manage AWS resources on your behalf.

Service-linked role permissions for inventory, OpsData, and OpsItems

The `AWSServiceRoleForAmazonSSM` service-linked role trusts only `ssm.amazonaws.com` to assume this role.

You can use the Systems Manager service-linked role `AWSServiceRoleForAmazonSSM` for the following:

- The Systems Manager Inventory tool uses the service-linked role `AWSServiceRoleForAmazonSSM` to collect inventory metadata from tags and resource groups.
- The Explorer tool uses the service-linked role `AWSServiceRoleForAmazonSSM` 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.

> **Important**
>
> Previously, the Systems Manager console provided you with the ability to choose the AWS managed IAM service-linked role `AWSServiceRoleForAmazonSSM` to use as the maintenance role for your tasks. Using this role and its associated policy, `AmazonSSMServiceRolePolicy`, for maintenance window tasks is no longer

recommended. If you're using this role for maintenance window tasks now, we encourage you to stop using it. Instead, create your own IAM role that enables communication between Systems Manager and other AWS services when your maintenance window tasks run.
For more information, see Setting up Maintenance Windows.

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.

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*.

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 tools, ensure that none are using the role before attempting to delete it.

- **Inventory:** If you delete the service-linked role used by the Inventory tool, 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.
- **Explorer:** If you delete the service-linked role used by the Explorer tool, 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 or resource groups, 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.

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 OpsCenter and Explorer

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.

Service-linked role permissions for Systems Manager account discovery

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:DescribeOrganizationalUnit`
- `organizations:DescribeOrganization`
- `organizations:ListAccounts`
- `organizations:ListAWSServiceAccessForOrganization`
- `organizations:ListChildren`
- `organizations:ListParents`
- `organizations:ListDelegatedServicesForAccount`
- `organizations:ListDelegatedAdministrators`
- `organizations:ListRoots`

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 want to use Explorer and OpsCenter, tools in Systems Manager, across multiple AWS accounts. For OpsCenter, you must manually create the service-linked role. For more information, see (Optional) Manually set up OpsCenter to centrally manage OpsItems across accounts.

For Explorer, 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.

> **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 Manager AWSServiceRoleForAmazonSSM_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 Systems Manager endpoints and quotas.

Updates to the AWSServiceRoleForAmazonSSM_AccountDiscovery service-linked role

View details about updates to the AWSServiceRoleForAmazonSSM_AccountDiscovery service-linked role 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 page.

Change	Description	Date
New permissions added	This service-linked role now includes organizations:DescribeOrganizationalUnit and organizations:ListRoots permissions. These permissions enable an AWS Organizations management account or a Systems Manager delegated administrator account to work with OpsItems across accounts. For more information, see (Optional) Manually set up OpsCenter to centrally manage OpsItems across accounts.	October 17, 2022

Using roles to create OpsData and OpsItems for Explorer

Systems Manager uses the service-linked role named **`AWSServiceRoleForSystemsManagerOpsDataSync`**. AWS Systems Manager uses this IAM service role for Explorer to create OpsData and OpsItems.

Service-linked role permissions for Systems Manager OpsData sync

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.

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 that allows Explorer to create OpsData and OpsItems** use case. In the AWS CLI or the AWS API, create a service-linked role with the `opsdatasync.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.
- To stop OpsCenter from creating new OpsItems for Security Hub findings, see

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

AWS Region name	Region identity	Support in Systems Manager
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
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

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.

`AWSServiceRoleForAmazonSSM_OpsInsights` service-linked role permissions for Systems Manager operational insight OpsItems

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:

JSON

```
{
 "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 deactivate operational insights in Systems Manager OpsCenter. For more information, see Analyzing operational insights to reduce OpsItems.

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

Region name	Region identity	Support in Systems Manager
Europe (Milan)	eu-south-1	Yes
South America (São Paulo)	sa-east-1	Yes
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

Using roles to maintain Quick Setup-provisioned resource health and consistency

Systems Manager uses the service-linked role named **`AWSServiceRoleForSSMQuickSetup`**.

`AWSServiceRoleForSSMQuickSetup` service-linked role permissions for Systems Manager

The `AWSServiceRoleForSSMQuickSetup` service-linked role trusts the following services to assume the role:

- `ssm-quicksetup.amazonaws.com`

AWS Systems Manager uses this IAM service role to check configuration health, ensure consistent use of parameters and provisioned resources, and remediate resources when drift is detected.

The role permissions policy allows Systems Manager to complete the following actions on the specified resources:

- `ssm` (Systems Manager) – Reads information about the state that configured resources are intended to be in, including in delegated administrator accounts.
- `iam` (AWS Identity and Access Management) – This is required for resource data syncs to be accessible across entire organizations in AWS Organizations.

- `organizations` (AWS Organizations) – Reads information about the member accounts that belong to an organization as configured in Organizations.
- `cloudformation` (AWS CloudFormation) – Reads information about CloudFormation stacks used to manage the state of resources and CloudFormation stackset operations.

The managed policy that is used to provide permissions for the `AWSServiceRoleForSSMQuickSetup` role is `SSMQuickSetupRolePolicy`. For details about the permissions it grants, see AWS managed policy: SSMQuickSetupRolePolicy.

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 `AWSServiceRoleForSSMQuickSetup` service-linked role for Systems Manager

You don't need to manually create the AWSServiceRoleForSSMQuickSetup service-linked role. When you create a Quick Setup configuration in the AWS Management Console, Systems Manager creates the service-linked role for you.

Editing the `AWSServiceRoleForSSMQuickSetup` service-linked role for Systems Manager

Systems Manager does not allow you to edit the `AWSServiceRoleForSSMQuickSetup` 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 `AWSServiceRoleForSSMQuickSetup` 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 AWSServiceRoleForSSMQuickSetup service-linked role

Before you can use IAM to delete the AWSServiceRoleForSSMQuickSetup service-linked role, you must first delete the Quick Setup configurations that are using the role. For more information, see Editing and deleting your configuration.

Manually delete the AWSServiceRoleForSSMQuickSetup service-linked role

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForSSMQuickSetup service-linked role. For more information, see the following topics:

- Deleting a service-linked role in the *IAM User Guide*
- delete-configuration-manager in the Quick Setup section of the *AWS CLI Reference*
- DeleteConfigurationManager in the *Quick Setup API Reference*

Supported Regions for the Systems Manager AWSServiceRoleForSSMQuickSetup service-linked role

Systems Manager does not support using service-linked roles in every Region where the service is available. You can use the AWSServiceRoleForSSMQuickSetup role in the following Regions.

- 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 (Stockholm)
- Europe (Ireland)

- Europe (London)
- Europe (Paris)
- South America (São Paulo)

Using roles to export Explorer OpsData

AWS Systems Manager Explorer uses the **AmazonSSMExplorerExportRole** service role to export operations data (OpsData) using the `AWS-ExportOpsDataToS3` automation runbook.

Service-linked role permissions for Explorer

The `AmazonSSMExplorerExportRole` service-linked role trusts only `ssm.amazonaws.com` to assume this role.

You can use the `AmazonSSMExplorerExportRole` service-linked role to export operations data (OpsData) using the `AWS-ExportOpsDataToS3` automation runbook. You can export 5,000 OpsData items from Explorer as a comma separated value (.csv) file to an Amazon Simple Storage Service (Amazon S3) bucket.

The role permissions policy allows Systems Manager to complete the following actions on the specified resources:

- `s3:PutObject`
- `s3:GetBucketAcl`
- `s3:GetBucketLocation`
- `sns:Publish`
- `logs:DescribeLogGroups`
- `logs:DescribeLogStreams`
- `logs:CreateLogGroup`
- `logs:PutLogEvents`
- `logs:CreateLogStream`
- `ssm:GetOpsSummary`

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 AmazonSSMExplorerExportRole service-linked role for Systems Manager

Systems Manager creates the AmazonSSMExplorerExportRole service-linked role when you export OpsData using Explorer in the Systems Manager console. For more information, see Exporting OpsData from Systems Manager Explorer.

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 AmazonSSMExplorerExportRole service-linked role for Systems Manager

Systems Manager doesn't allow you to edit the AmazonSSMExplorerExportRole 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 AmazonSSMExplorerExportRole 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.

> **Note**
>
> If the Systems Manager service is using the role when you try to delete tags or resource groups, 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 AmazonSSMExplorerExportRole

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.

To manually delete the `AmazonSSMExplorerExportRole` service-linked role using IAM

Use the IAM console, the AWS CLI, or the IAM API to delete the `AmazonSSMExplorerExportRole` service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for the Systems Manager `AmazonSSMExplorerExportRole` service-linked role

Systems Manager supports using the `AmazonSSMExplorerExportRole` 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 enable just-in-time node access

Systems Manager uses the service-linked role named **`AWSServiceRoleForSystemsManagerJustInTimeAccess`**. AWS Systems Manager uses this IAM service role to enable just-in-time node access.

Service-linked role permissions for Systems Manager just-in-time node access

The `AWSServiceRoleForSystemsManagerJustInTimeAccess` service-linked role trusts the following services to assume the role:

- `ssm.amazonaws.com`

The role permissions policy allows Systems Manager to complete the following actions on the specified resources:

- `ssm:CreateOpsItem`
- `ssm:GetOpsItem`
- `ssm:UpdateOpsItem`
- `ssm:DescribeOpsItems`
- `ssm:DescribeSessions`
- `ssm:ListTagsForResource`
- `ssm-guiconnect:ListConnections`
- `identitystore:ListGroupMembershipsForMember`
- `identitystore:DescribeUser`

- identitystore:GetGroupId
- identitystore:GetUserId
- sso-directory:DescribeUsers
- sso-directory:IsMemberInGroup
- sso:ListInstances
- sso:DescribeRegisteredRegions
- sso:ListDirectoryAssociations
- ec2:DescribeTags

The managed policy that is used to provide permissions for the AWSServiceRoleForSystemsManagerJustInTimeAccess role is AWSSystemsManagerEnableJustInTimeAccessPolicy. For details about the permissions it grants, see AWS managed policy: AWSSystemsManagerJustInTimeAccessServicePolicy.

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 AWSServiceRoleForSystemsManagerJustInTimeAccess service-linked role for Systems Manager

You don't need to manually create a service-linked role. When you enable just-in-time node access 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 November 19, 2024, when it began supporting service-linked roles, then Systems Manager created the AWSServiceRoleForSystemsManagerJustInTimeAccess 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 just-in-time node access 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 that allows Systems Manager to enable just-in-time node access.** 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 `AWSServiceRoleForSystemsManagerJustInTimeAccess` service-linked role for Systems Manager

Systems Manager doesn't allow you to edit the `AWSServiceRoleForSystemsManagerJustInTimeAccess` 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 `AWSServiceRoleForSystemsManagerJustInTimeAccess` 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.

To manually delete the `AWSServiceRoleForSystemsManagerJustInTimeAccess` service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the `AWSServiceRoleForSystemsManagerJustInTimeAccess` service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for the Systems Manager `AWSServiceRoleForSystemsManagerJustInTimeAccess` service-linked role

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
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 send just-in-time node access request notifications

Systems Manager uses the service-linked role named **`AWSServiceRoleForSystemsManagerNotifications`**. AWS Systems Manager uses this IAM service role to send notifications to access request approvers.

Service-linked role permissions for Systems Manager just-in-time node access notifications

The `AWSServiceRoleForSystemsManagerNotifications` service-linked role trusts the following services to assume the role:

- `ssm.amazonaws.com`

The role permissions policy allows Systems Manager to complete the following actions on the specified resources:

- `identitystore:ListGroupMembershipsForMember`
- `identitystore:ListGroupMemberships`
- `identitystore:DescribeUser`
- `sso:ListInstances`
- `sso:DescribeRegisteredRegions`
- `sso:ListDirectoryAssociations`
- `sso-directory:DescribeUser`
- `sso-directory:ListMembersInGroup`
- `iam:GetRole`

The managed policy that is used to provide permissions for the `AWSServiceRoleForSystemsManagerNotifications` role is `AWSSystemsManagerNotificationsServicePolicy`. For details about the permissions it grants, see AWS managed policy: AWSSystemsManagerNotificationsServicePolicy.

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 `AWSServiceRoleForSystemsManagerNotifications` service-linked role for Systems Manager

You don't need to manually create a service-linked role. When you enable just-in-time node access 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 November 19, 2024, when it began supporting service-linked roles, then Systems Manager created the `AWSServiceRoleForSystemsManagerNotifications` 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 just-in-time node access 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 that allows Systems Manager to send notifications to access request approvers.** 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 `AWSServiceRoleForSystemsManagerNotifications` service-linked role for Systems Manager

Systems Manager doesn't allow you to edit the `AWSServiceRoleForSystemsManagerNotifications` 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 `AWSServiceRoleForSystemsManagerNotifications` 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.

To manually delete the `AWSServiceRoleForSystemsManagerNotifications` service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the `AWSServiceRoleForSystemsManagerNotifications` service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for the Systems Manager `AWSServiceRoleForSystemsManagerNotifications` service-linked role

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

AWS Region name	Region identity	Support in Systems Manager
Asia Pacific (Seoul)	ap-northeast-2	Yes
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
Europe (Stockholm)	eu-north-1	Yes
South America (São Paulo)	sa-east-1	Yes
AWS GovCloud (US)	us-gov-west-1	No

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.

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 Using Amazon CloudWatch dashboards hosted by Systems Manager.

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 tools. For more information, see Monitoring Systems Manager events with Amazon EventBridge.

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 unified CloudWatch Logs (CloudWatch agent) and Viewing SSM Agent logs.

AWS Systems Manager Compliance

You can use Compliance, a tool in 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 tool in AWS Systems Manager, and associations in State Manager, a tool in AWS Systems Manager. For more information, see AWS Systems Manager Compliance.

AWS Systems Manager Explorer

Explorer, a tool in 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.

AWS Systems Manager OpsCenter

OpsCenter, a tool in 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 tool in 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.

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, tools in 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.

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*.

More info

- Logging and monitoring in AWS Systems Manager

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.

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 AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Systems Manager through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). 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 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
- 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
- SSM Agent installation best practices
- Systems Manager monitoring and auditing best practices

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 policies and Permissions boundaries for IAM entities
- Service control policies

Use recommended settings for SSM Agent when configured to use a proxy

If you configure SSM Agent to use a proxy, use the no_proxy variable with the IP address of the Systems Manager instance metadata service to ensure that calls to Systems Manager don't take on the identity of the proxy service.

For more information, see Configuring SSM Agent to use a proxy on Linux nodes and Configure SSM Agent to use a proxy for Windows Server instances.

Use SecureString parameters to encrypt and protect secret data

In Parameter Store, a tool in 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.

It's more difficult to establish access control policies for these operations when using an AWS managed key. For example, you if you use an AWS managed key to encrypt `SecureString` parameters and don't want users to work with `SecureString` parameters, the user's IAM policies must explicitly deny access to the default key.

For more information, see Restricting access to Parameter Store parameters using IAM policies 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 Data elements and parameters.

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 Improve the security of EC2 instances by using VPC endpoints for Systems Manager and Internetwork traffic privacy in Amazon VPC in the *Amazon VPC User Guide*.

Restrict Session Manager users to sessions using interactive commands and specific SSM session documents

Session Manager, a tool in AWS Systems Manager, provides several methods for starting sessions 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).

For added security, you can limit Session Manager access to specific Amazon EC2 instances and specific Session Manager session documents. You grant or revoke Session Manager access in this way by using AWS Identity and Access Management (IAM) policies. For more information, see Step 3: Control session access to managed nodes.

Provide temporary node permissions for Automation workflows

During a workflow in Automation, a tool in 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. We also recommend verifying the signature of SSM Agent as part of your update process. For information, see Verifying the signature of SSM Agent.
- 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 Install the Session Manager plugin for the AWS CLI.
- 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.

SSM Agent installation best practices

When installing SSM Agent, use the appropriate installation method for your machine type. In particular, use the `ssm-setup-cli` tool for all non-EC2 installations in a hybrid and multicloud environment. This tool provides additional security protections for non-EC2 machines.

To install the agent on on-premises servers and virtual machines, use the `ssm-setup-cli` tool as described in the following topics:

- the section called “Install SSM Agent on hybrid Linux nodes”
- the section called “Install SSM Agent on hybrid Windows Server nodes”

To install the agent on EC2 instances, use the appropriate installation procedure for your operating system type:

- the section called “Manually installing and uninstalling SSM Agent on EC2 instances for Linux”
- the section called “Manually installing and uninstalling SSM Agent on EC2 instances for macOS”
- the section called “Manually installing and uninstalling SSM Agent on EC2 instances for Windows Server”

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 unified CloudWatch Logs (CloudWatch agent) and Monitoring Systems Manager events with Amazon EventBridge.

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.

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 required 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.

More info

- 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

Code examples for Systems Manager using AWS SDKs

The following code examples show how to use Systems Manager with an AWS software development kit (SDK).

Basics are code examples that show you how to perform the essential operations within a service.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Get started

Hello Systems Manager

The following code examples show how to get started using Systems Manager.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <awsAccount>

                Where:
                    awsAccount - Your AWS Account number.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        listDocuments(ssmClient, awsAccount);
    }

    /*
    This code automatically fetches the next set of results using the `nextToken`
and
    stops once the desired maxResults (20 in this case) have been reached.
    */
    public static void listDocuments(SsmClient ssmClient, String awsAccount) {
        String nextToken = null;
        int totalDocumentsReturned = 0;
        int maxResults = 20;
        do {
            ListDocumentsRequest request = ListDocumentsRequest.builder()
                .documentFilterList(
                    DocumentFilter.builder()
                        .key("Owner")
                        .value(awsAccount)
                        .build()
                    )
                .maxResults(maxResults)
                .nextToken(nextToken)
                .build();

            ListDocumentsResponse response = ssmClient.listDocuments(request);
```

```
            response.documentIdentifiers().forEach(identifier ->
 System.out.println("Document Name: " + identifier.name()));
            nextToken = response.nextToken();
            totalDocumentsReturned += response.documentIdentifiers().size();
        } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- For API details, see ListDocuments in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { paginateListDocuments, SSMClient } from "@aws-sdk/client-ssm";

// Call ListDocuments and display the result.
export const main = async () => {
  const client = new SSMClient();
  const listDocumentsPaginated = [];
  console.log(
    "Hello, AWS Systems Manager! Let's list some of your documents:\n",
  );
  try {
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListDocuments({ client }, { MaxResults: 5 });
    for await (const page of paginator) {
      listDocumentsPaginated.push(...page.DocumentIdentifiers);
    }
  } catch (caught) {
    console.error(`There was a problem saying hello: ${caught.message}`);
    throw caught;
  }

  for (const { Name, DocumentFormat, CreatedDate } of listDocumentsPaginated) {
```

```
    console.log(`${Name} - ${DocumentFormat} - ${CreatedDate}`);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- For API details, see ListDocuments in *AWS SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import boto3
from botocore.exceptions import ClientError


def hello_systems_manager(ssm_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Systems Manager
    client and list the first 5 documents in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ssm_client: A Boto3 AWS Systems Manager Client object. This object
 wraps
                           the low-level AWS Systems Manager service API.
    """
    print("Hello, AWS Systems Manager! Let's list some of your documents:\n")

    paginator = ssm_client.get_paginator("list_documents")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 5})
```

```
    for page in page_iterator:
        for document in page["DocumentIdentifiers"]:
            print(f"  {document['Name']}")


if __name__ == "__main__":
    try:
        hello_systems_manager(boto3.client("ssm"))
    except ClientError as err:
        print("Hello systems manager had an error.")
        print(err.response["Error"]["Code"])
        print(err.response["Error"]["Message"])
```

- For API details, see ListDocuments in *AWS SDK for Python (Boto3) API Reference*.

Code examples

- Basic examples for Systems Manager using AWS SDKs
  - Hello Systems Manager
  - Learn the basics of Systems Manager with an AWS SDK
  - Actions for Systems Manager using AWS SDKs
    - Use AddTagsToResource with a CLI
    - Use CancelCommand with a CLI
    - Use CreateActivation with a CLI
    - Use CreateAssociation with a CLI
    - Use CreateAssociationBatch with a CLI
    - Use CreateDocument with an AWS SDK or CLI
    - Use CreateMaintenanceWindow with an AWS SDK or CLI
    - Use CreateOpsItem with an AWS SDK or CLI
    - Use CreatePatchBaseline with a CLI
    - Use DeleteActivation with a CLI
    - Use DeleteAssociation with a CLI
    - Use DeleteDocument with an AWS SDK or CLI
    - Use DeleteMaintenanceWindow with an AWS SDK or CLI
    - Use DeleteOpsItem with an AWS SDK

- Use DeleteParameter with a CLI
- Use DeletePatchBaseline with a CLI
- Use DeregisterManagedInstance with a CLI
- Use DeregisterPatchBaselineForPatchGroup with a CLI
- Use DeregisterTargetFromMaintenanceWindow with a CLI
- Use DeregisterTaskFromMaintenanceWindow with a CLI
- Use DescribeActivations with a CLI
- Use DescribeAssociation with a CLI
- Use DescribeAssociationExecutionTargets with a CLI
- Use DescribeAssociationExecutions with a CLI
- Use DescribeAutomationExecutions with a CLI
- Use DescribeAutomationStepExecutions with a CLI
- Use DescribeAvailablePatches with a CLI
- Use DescribeDocument with a CLI
- Use DescribeDocumentPermission with a CLI
- Use DescribeEffectiveInstanceAssociations with a CLI
- Use DescribeEffectivePatchesForPatchBaseline with a CLI
- Use DescribeInstanceAssociationsStatus with a CLI
- Use DescribeInstanceInformation with a CLI
- Use DescribeInstancePatchStates with a CLI
- Use DescribeInstancePatchStatesForPatchGroup with a CLI
- Use DescribeInstancePatches with a CLI
- Use DescribeMaintenanceWindowExecutionTaskInvocations with a CLI
- Use DescribeMaintenanceWindowExecutionTasks with a CLI
- Use DescribeMaintenanceWindowExecutions with a CLI
- Use DescribeMaintenanceWindowTargets with a CLI
- Use DescribeMaintenanceWindowTasks with a CLI
- Use DescribeMaintenanceWindows with a CLI
- Use DescribeOpsItems with an AWS SDK or CLI
- Use DescribeParameters with an AWS SDK or CLI

- Use DescribePatchBaselines with a CLI
- Use DescribePatchGroupState with a CLI
- Use DescribePatchGroups with a CLI
- Use GetAutomationExecution with a CLI
- Use GetCommandInvocation with a CLI
- Use GetConnectionStatus with a CLI
- Use GetDefaultPatchBaseline with a CLI
- Use GetDeployablePatchSnapshotForInstance with a CLI
- Use GetDocument with a CLI
- Use GetInventory with a CLI
- Use GetInventorySchema with a CLI
- Use GetMaintenanceWindow with a CLI
- Use GetMaintenanceWindowExecution with a CLI
- Use GetMaintenanceWindowExecutionTask with a CLI
- Use GetParameter with an AWS SDK or CLI
- Use GetParameterHistory with a CLI
- Use GetParameters with a CLI
- Use GetPatchBaseline with a CLI
- Use GetPatchBaselineForPatchGroup with a CLI
- Use ListAssociationVersions with a CLI
- Use ListAssociations with a CLI
- Use ListCommandInvocations with an AWS SDK or CLI
- Use ListCommands with a CLI
- Use ListComplianceItems with a CLI
- Use ListComplianceSummaries with a CLI
- Use ListDocumentVersions with a CLI
- Use ListDocuments with a CLI
- Use ListInventoryEntries with a CLI
- Use ListResourceComplianceSummaries with a CLI
- Use ListTagsForResource with a CLI

- Use ModifyDocumentPermission with a CLI
- Use PutComplianceItems with a CLI
- Use PutInventory with a CLI
- Use PutParameter with an AWS SDK or CLI
- Use RegisterDefaultPatchBaseline with a CLI
- Use RegisterPatchBaselineForPatchGroup with a CLI
- Use RegisterTargetWithMaintenanceWindow with a CLI
- Use RegisterTaskWithMaintenanceWindow with a CLI
- Use RemoveTagsFromResource with a CLI
- Use SendCommand with an AWS SDK or CLI
- Use StartAutomationExecution with a CLI
- Use StartSession with a CLI
- Use StopAutomationExecution with a CLI
- Use UpdateAssociation with a CLI
- Use UpdateAssociationStatus with a CLI
- Use UpdateDocument with a CLI
- Use UpdateDocumentDefaultVersion with a CLI
- Use UpdateMaintenanceWindow with an AWS SDK or CLI
- Use UpdateManagedInstanceRole with a CLI
- Use UpdateOpsItem with an AWS SDK or CLI
- Use UpdatePatchBaseline with a CLI

Basic examples for Systems Manager using AWS SDKs

The following code examples show how to use the basics of AWS Systems Manager with AWS SDKs.

Examples

- Hello Systems Manager
- Learn the basics of Systems Manager with an AWS SDK
- Actions for Systems Manager using AWS SDKs

- Use AddTagsToResource with a CLI

- Use CancelCommand with a CLI
- Use CreateActivation with a CLI
- Use CreateAssociation with a CLI
- Use CreateAssociationBatch with a CLI
- Use CreateDocument with an AWS SDK or CLI
- Use CreateMaintenanceWindow with an AWS SDK or CLI
- Use CreateOpsItem with an AWS SDK or CLI
- Use CreatePatchBaseline with a CLI
- Use DeleteActivation with a CLI
- Use DeleteAssociation with a CLI
- Use DeleteDocument with an AWS SDK or CLI
- Use DeleteMaintenanceWindow with an AWS SDK or CLI
- Use DeleteOpsItem with an AWS SDK
- Use DeleteParameter with a CLI
- Use DeletePatchBaseline with a CLI
- Use DeregisterManagedInstance with a CLI
- Use DeregisterPatchBaselineForPatchGroup with a CLI
- Use DeregisterTargetFromMaintenanceWindow with a CLI
- Use DeregisterTaskFromMaintenanceWindow with a CLI
- Use DescribeActivations with a CLI
- Use DescribeAssociation with a CLI
- Use DescribeAssociationExecutionTargets with a CLI
- Use DescribeAssociationExecutions with a CLI
- Use DescribeAutomationExecutions with a CLI
- Use DescribeAutomationStepExecutions with a CLI
- Use DescribeAvailablePatches with a CLI
- Use DescribeDocument with a CLI
- Use DescribeDocumentPermission with a CLI
- Use DescribeEffectiveInstanceAssociations with a CLI
- Use DescribeEffectivePatchesForPatchBaseline with a CLI

- Use DescribeInstanceAssociationsStatus with a CLI
- Use DescribeInstanceInformation with a CLI
- Use DescribeInstancePatchStates with a CLI
- Use DescribeInstancePatchStatesForPatchGroup with a CLI
- Use DescribeInstancePatches with a CLI
- Use DescribeMaintenanceWindowExecutionTaskInvocations with a CLI
- Use DescribeMaintenanceWindowExecutionTasks with a CLI
- Use DescribeMaintenanceWindowExecutions with a CLI
- Use DescribeMaintenanceWindowTargets with a CLI
- Use DescribeMaintenanceWindowTasks with a CLI
- Use DescribeMaintenanceWindows with a CLI
- Use DescribeOpsItems with an AWS SDK or CLI
- Use DescribeParameters with an AWS SDK or CLI
- Use DescribePatchBaselines with a CLI
- Use DescribePatchGroupState with a CLI
- Use DescribePatchGroups with a CLI
- Use GetAutomationExecution with a CLI
- Use GetCommandInvocation with a CLI
- Use GetConnectionStatus with a CLI
- Use GetDefaultPatchBaseline with a CLI
- Use GetDeployablePatchSnapshotForInstance with a CLI
- Use GetDocument with a CLI
- Use GetInventory with a CLI
- Use GetInventorySchema with a CLI
- Use GetMaintenanceWindow with a CLI
- Use GetMaintenanceWindowExecution with a CLI
- Use GetMaintenanceWindowExecutionTask with a CLI
- Use GetParameter with an AWS SDK or CLI
- Use GetParameterHistory with a CLI
- Use GetParameters with a CLI

- Use GetPatchBaseline with a CLI
- Use GetPatchBaselineForPatchGroup with a CLI
- Use ListAssociationVersions with a CLI
- Use ListAssociations with a CLI
- Use ListCommandInvocations with an AWS SDK or CLI
- Use ListCommands with a CLI
- Use ListComplianceItems with a CLI
- Use ListComplianceSummaries with a CLI
- Use ListDocumentVersions with a CLI
- Use ListDocuments with a CLI
- Use ListInventoryEntries with a CLI
- Use ListResourceComplianceSummaries with a CLI
- Use ListTagsForResource with a CLI
- Use ModifyDocumentPermission with a CLI
- Use PutComplianceItems with a CLI
- Use PutInventory with a CLI
- Use PutParameter with an AWS SDK or CLI
- Use RegisterDefaultPatchBaseline with a CLI
- Use RegisterPatchBaselineForPatchGroup with a CLI
- Use RegisterTargetWithMaintenanceWindow with a CLI
- Use RegisterTaskWithMaintenanceWindow with a CLI
- Use RemoveTagsFromResource with a CLI
- Use SendCommand with an AWS SDK or CLI
- Use StartAutomationExecution with a CLI
- Use StartSession with a CLI
- Use StopAutomationExecution with a CLI
- Use UpdateAssociation with a CLI
- Use UpdateAssociationStatus with a CLI
- Use UpdateDocument with a CLI

- Use UpdateDocumentDefaultVersion with a CLI

- Use UpdateMaintenanceWindow with an AWS SDK or CLI
- Use UpdateManagedInstanceRole with a CLI
- Use UpdateOpsItem with an AWS SDK or CLI
- Use UpdatePatchBaseline with a CLI

Hello Systems Manager

The following code examples show how to get started using Systems Manager.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <awsAccount>

                Where:
                    awsAccount - Your AWS Account number.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        listDocuments(ssmClient, awsAccount);
    }

    /*
    This code automatically fetches the next set of results using the `nextToken`
 and
    stops once the desired maxResults (20 in this case) have been reached.
    */
    public static void listDocuments(SsmClient ssmClient, String awsAccount) {
        String nextToken = null;
        int totalDocumentsReturned = 0;
        int maxResults = 20;
        do {
            ListDocumentsRequest request = ListDocumentsRequest.builder()
                .documentFilterList(
                    DocumentFilter.builder()
                        .key("Owner")
                        .value(awsAccount)
                        .build()
                    )
                .maxResults(maxResults)
                .nextToken(nextToken)
                .build();

            ListDocumentsResponse response = ssmClient.listDocuments(request);
            response.documentIdentifiers().forEach(identifier ->
 System.out.println("Document Name: " + identifier.name()));
            nextToken = response.nextToken();
            totalDocumentsReturned += response.documentIdentifiers().size();
        } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- For API details, see ListDocuments in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { paginateListDocuments, SSMClient } from "@aws-sdk/client-ssm";

// Call ListDocuments and display the result.
export const main = async () => {
  const client = new SSMClient();
  const listDocumentsPaginated = [];
  console.log(
    "Hello, AWS Systems Manager! Let's list some of your documents:\n",
  );
  try {
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListDocuments({ client }, { MaxResults: 5 });
    for await (const page of paginator) {
      listDocumentsPaginated.push(...page.DocumentIdentifiers);
    }
  } catch (caught) {
    console.error(`There was a problem saying hello: ${caught.message}`);
    throw caught;
  }

  for (const { Name, DocumentFormat, CreatedDate } of listDocumentsPaginated) {
    console.log(`${Name} - ${DocumentFormat} - ${CreatedDate}`);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- For API details, see ListDocuments in *AWS SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import boto3
from botocore.exceptions import ClientError


def hello_systems_manager(ssm_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Systems Manager
    client and list the first 5 documents in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ssm_client: A Boto3 AWS Systems Manager Client object. This object
 wraps
                           the low-level AWS Systems Manager service API.
    """
    print("Hello, AWS Systems Manager! Let's list some of your documents:\n")

    paginator = ssm_client.get_paginator("list_documents")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 5})
    for page in page_iterator:
        for document in page["DocumentIdentifiers"]:
            print(f"  {document['Name']}")


if __name__ == "__main__":
    try:
        hello_systems_manager(boto3.client("ssm"))
    except ClientError as err:
        print("Hello systems manager had an error.")
        print(err.response["Error"]["Code"])
```

```
        print(err.response["Error"]["Message"])
```

- For API details, see ListDocuments in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Learn the basics of Systems Manager with an AWS SDK

The following code examples show how to:

- Create a maintenance window.
- Modify the maintenance window schedule.
- Create a document.
- Send a command to a specified EC2 instance.
- Create an OpsItem.
- Update and resolve the OpsItem.
- Delete the maintenance window, OpsItem, and document.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.SsmException;

import java.util.Scanner;
public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
```

```
   public static void main(String[] args) {
       String usage = """
           Usage:
             <instanceId> <title> <source> <category> <severity>

           Where:
               instanceId - The Amazon EC2 Linux/UNIX instance Id that AWS
Systems Manager uses (ie, i-0149338494ed95f06).
               title - The title of the parameter (default is Disk Space Alert).
               source - The source of the parameter (default is EC2).
               category - The category of the parameter. Valid values are
'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
Performance).
               severity - The severity of the parameter. Severity should be a
number from 1 to 4 (default is 2).
       """;

       if (args.length != 1) {
           System.out.println(usage);
           System.exit(1);
       }

       Scanner scanner = new Scanner(System.in);
       SSMActions actions = new SSMActions();
       String documentName;
       String windowName;
       String instanceId = args[0];
       String title = "Disk Space Alert" ;
       String source = "EC2" ;
       String category = "Availability" ;
       String severity = "2" ;

       System.out.println(DASHES);
       System.out.println("""
               Welcome to the AWS Systems Manager SDK Basics scenario.
               This Java program demonstrates how to interact with AWS Systems
Manager using the AWS SDK for Java (v2).
               AWS Systems Manager is the operations hub for your AWS
applications and resources and a secure end-to-end management solution.
               The program's primary functionalities include creating a
maintenance window, creating a document, sending a command to a document,
               listing documents, listing commands, creating an OpsItem,
modifying an OpsItem, and deleting AWS SSM resources.
```

```
                Upon completion of the program, all AWS resources are cleaned up.
                Let's get started...

                """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("1. Create an SSM maintenance window.");
        System.out.println("Please enter the maintenance window name (default is
ssm-maintenance-window):");
        String win = scanner.nextLine();
        windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
        String winId = null;
        try {
            winId = actions.createMaintenanceWindow(windowName);
            waitForInputToContinue(scanner);
            System.out.println("The maintenance window ID is: " + winId);
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM maintenance window already exists.
Retrieving existing window ID...");
            String existingWinId = actions.createMaintenanceWindow(windowName);
            System.out.println("Existing window ID: " + existingWinId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("2. Modify the maintenance window by changing the
schedule");
        waitForInputToContinue(scanner);
        try {
            actions.updateSSMMaintenanceWindow(winId, windowName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM maintenance window was successfully
updated");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
```

```
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("3. Create an SSM document that defines the actions
that Systems Manager performs on your managed nodes.");
        System.out.println("Please enter the document name (default is
ssmdocument):");
        String doc = scanner.nextLine();
        documentName = doc.isEmpty() ? "ssmdocument" : doc;
        try {
            actions.createSSMDoc(documentName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM document was successfully created");
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM document already exists. Moving on");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("4. Now we are going to run a command on an EC2
instance");
        waitForInputToContinue(scanner);
        String commandId="";
        try {
            commandId = actions.sendSSMCommand(documentName, instanceId);
            waitForInputToContinue(scanner);
            System.out.println("The command was successfully sent. Command ID: "
+ commandId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
        } catch (InterruptedException e) {
            System.err.println("Thread was interrupted: " + e.getMessage());
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println("5. Lets get the time when the specific command was
sent to the specific managed node");
        waitForInputToContinue(scanner);
        try {
            actions.displayCommands(commandId);
            System.out.println("The command invocations were successfully
displayed.");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
             6. Now we will create an SSM OpsItem.
             A SSM OpsItem is a feature provided by Amazon's Systems Manager
(SSM) service.
             It is a type of operational data item that allows you to manage and
track various operational issues,
             events, or tasks within your AWS environment.

             You can create OpsItems to track and manage operational issues as
they arise.
             For example, you could create an OpsItem whenever your application
detects a critical error
             or an anomaly in your infrastructure.
            """);

        waitForInputToContinue(scanner);
        String opsItemId;
        try {
            opsItemId = actions.createSSMOpsItem(title, source, category,
severity);
            System.out.println(opsItemId + " was created");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
```

```
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Now we will update the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        String description = "An update to "+opsItemId ;
        try {
            actions.updateOpsItem(opsItemId, title, description);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("8. Now we will get the status of the SSM OpsItem
"+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.describeOpsItems(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("9. Now we will resolve the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.resolveOpsItem(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
```

```
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("10. Would you like to delete the AWS Systems Manager
resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the resources.");
            waitForInputToContinue(scanner);
            try {
                actions.deleteMaintenanceWindow(winId);
                actions.deleteDoc(documentName);
            } catch (SsmException e) {
                System.err.println("SSM error: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("Unexpected error: " + e.getMessage());
                return;
            }
        } else {
            System.out.println("The AWS Systems Manager resources will not be
deleted");
        }
        System.out.println(DASHES);

        System.out.println("This concludes the AWS Systems Manager SDK Basics
scenario.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
```

```
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

A wrapper class for Systems Manager SDK methods.

```
public class SSMActions {

    private static SsmAsyncClient ssmAsyncClient;

    private static SsmAsyncClient getAsyncClient() {
        if (ssmAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryPolicy(RetryPolicy.builder()
                            .numRetries(3)
                            .build())
                    .build();

            ssmAsyncClient = SsmAsyncClient.builder()
                    .region(Region.US_EAST_1)
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return ssmAsyncClient;
    }

    /**
     * Deletes an AWS SSM document asynchronously.
     *
```

```
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Deletes an AWS SSM Maintenance Window asynchronously.
     *
     * @param winId The ID of the Maintenance Window to delete.
     * <p>
```

```
    * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
    * If an exception occurs, it handles the error appropriately.
    */
   public void deleteMaintenanceWindow(String winId) {
       DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
               .windowId(winId)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().deleteMaintenanceWindow(windowRequest)
                   .thenAccept(response -> {
                       System.out.println("The maintenance window was
successfully deleted.");
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
           } else {
               throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
           }
       });

       try {
           future.join();
       } catch (CompletionException ex) {
           throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
       }
   }

   /**
    * Resolves an AWS SSM OpsItem asynchronously.
    *
    * @param opsID The ID of the OpsItem to resolve.
    * <p>
```

```
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
```

```
    * If the request is successful, it prints the title and status of each
OpsItem.
    * If an exception occurs, it handles the error appropriately.
    */
   public void describeOpsItems(String key) {
       OpsItemFilter filter = OpsItemFilter.builder()
               .key(OpsItemFilterKey.OPS_ITEM_ID)
               .values(key)
               .operator(OpsItemFilterOperator.EQUAL)
               .build();

       DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
               .maxResults(10)
               .opsItemFilters(filter)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().describeOpsItems(itemsRequest)
                   .thenAccept(itemsResponse -> {
                       List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                       for (OpsItemSummary item : items) {
                           System.out.println("The item title is " +
item.title() + " and the status is " + item.status().toString());
                       }
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
           } else {
               throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
           }
       });

       try {
           future.join();
       } catch (CompletionException ex) {
```

```
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Updates the AWS SSM OpsItem asynchronously.
     *
     * @param opsItemId The ID of the OpsItem to update.
     * @param title The new title of the OpsItem.
     * @param description The new description of the OpsItem.
     * <p>
     * This method initiates an asynchronous request to update an SSM OpsItem.
     * If the request is successful, it completes without returning a value.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateOpsItem(String opsItemId, String title, String description)
{
        Map<String, OpsItemDataValue> operationalData = new HashMap<>();
        operationalData.put("key1",
OpsItemDataValue.builder().value("value1").build());
        operationalData.put("key2",
OpsItemDataValue.builder().value("value2").build());

        CompletableFuture<Void> future =
getOpsItem(opsItemId).thenCompose(opsItem -> {
            UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
                    .opsItemId(opsItemId)
                    .title(title)
                    .operationalData(operationalData)
                    .status(opsItem.statusAsString())
                    .description(description)
                    .build();

            return getAsyncClient().updateOpsItem(request).thenAccept(response ->
{
                System.out.println(opsItemId + " updated successfully.");
            }).exceptionally(ex -> {
                throw new CompletionException(ex);
            });
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
```

```
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }


    private static CompletableFuture<OpsItem> getOpsItem(String opsItemId) {
        GetOpsItemRequest request =
GetOpsItemRequest.builder().opsItemId(opsItemId).build();
        return
getAsyncClient().getOpsItem(request).thenApply(GetOpsItemResponse::opsItem);
    }

    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
```

```
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }

    /**
     * Displays the date and time when the specific command was invoked.
     *
     * @param commandId The ID of the command to describe.
     * <p>
     * This method initiates an asynchronous request to list command invocations
and prints the date and time of each command invocation.
     * If an exception occurs, it handles the error appropriately.
     */
    public void displayCommands(String commandId) {
        ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
                .commandId(commandId)
                .build();

        CompletableFuture<ListCommandInvocationsResponse> future =
getAsyncClient().listCommandInvocations(commandInvocationsRequest);
        future.thenAccept(response -> {
            List<CommandInvocation> commandList = response.commandInvocations();
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
            for (CommandInvocation invocation : commandList) {
                System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
            }
```

```
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }

    /**
     * Sends a SSM command to a managed node asynchronously.
     *
     * @param documentName The name of the document to use.
     * @param instanceId The ID of the instance to send the command to.
     * @return The command ID.
     * <p>
     * This method initiates asynchronous requests to send a SSM command to a
managed node.
     * It waits until the document is active, sends the command, and checks the
command execution status.
     */
    public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
        // Before we use Document to send a command - make sure it is active.
        CompletableFuture<Void> documentActiveFuture =
CompletableFuture.runAsync(() -> {
            boolean isDocumentActive = false;
            DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                    .name(documentName)
                    .build();

            while (!isDocumentActive) {
                CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
                String documentStatus =
response.join().document().statusAsString();
                if (documentStatus.equals("Active")) {
                    System.out.println("The SSM document is active and ready to
use.");
                    isDocumentActive = true;
                } else {
```

```
                    System.out.println("The SSM document is not active. Status: "
+ documentStatus);
                    try {
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        });

        documentActiveFuture.join();

        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
                .documentName(documentName)
                .instanceIds(instanceId)
                .build();

        // Send the command.
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse>
invocationFuture = getAsyncClient().getCommandInvocation(invocationRequest);
```

```
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution
successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }

    /**
     * Creates an AWS SSM document asynchronously.
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
```

```
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

```
   /**
    * Updates an SSM maintenance window asynchronously.
    *
    * @param id The ID of the maintenance window to update.
    * @param name The new name for the maintenance window.
    * <p>
    * This method initiates an asynchronous request to update an SSM maintenance
window.
    * If the request is successful, it prints a success message.
    * If an exception occurs, it handles the error appropriately.
    */
   public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
       UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
               .windowId(id)
               .allowUnassociatedTargets(true)
               .duration(24)
               .enabled(true)
               .name(name)
               .schedule("cron(0 0 ? * MON *)")
               .build();

       CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
       future.whenComplete((response, ex) -> {
           if (response != null) {
               System.out.println("The SSM maintenance window was successfully
updated");
           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof SsmException) {
                   throw new CompletionException(cause);
               } else {
                   throw new RuntimeException(cause);
               }
           }
       }).join();
   }

   /**
    * Creates an SSM maintenance window asynchronously.
    *
```

```
    * @param winName The name of the maintenance window.
    * @return The ID of the created or existing maintenance window.
    * <p>
    * This method initiates an asynchronous request to create an SSM maintenance
window.
    * If the request is successful, it prints the maintenance window ID.
    * If an exception occurs, it handles the error appropriately.
    */
   public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
       CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
               .name(winName)
               .description("This is my maintenance window")
               .allowUnassociatedTargets(true)
               .duration(2)
               .cutoff(1)
               .schedule("cron(0 10 ? * MON-FRI *)")
               .build();

       CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
       final String[] windowId = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String maintenanceWindowId = response.windowId();
               System.out.println("The maintenance window id is " +
maintenanceWindowId);
               windowId[0] = maintenanceWindowId;
           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof DocumentAlreadyExistsException) {
                   throw new CompletionException(cause);
               } else if (cause instanceof SsmException) {
                   throw new CompletionException(cause);
               } else {
                   throw new RuntimeException(cause);
               }
           }
       }).join();

       if (windowId[0] == null) {
           MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
```

```
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
 DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture
 = getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
 describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
 CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
 windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow

- ListCommandInvocations
- SendCommand
- UpdateOpsItem

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { fileURLToPath } from "node:url";
import {
  CreateDocumentCommand,
  CreateMaintenanceWindowCommand,
  CreateOpsItemCommand,
  DeleteDocumentCommand,
  DeleteMaintenanceWindowCommand,
  DeleteOpsItemCommand,
  DescribeOpsItemsCommand,
  DocumentAlreadyExists,
  OpsItemStatus,
  waitUntilCommandExecuted,
  CancelCommandCommand,
  paginateListCommandInvocations,
  SendCommandCommand,
  UpdateMaintenanceWindowCommand,
  UpdateOpsItemCommand,
  SSMClient,
} from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";
```

```
/**
 * @typedef {{
 *   ssmClient: import('@aws-sdk/client-ssm').SSMClient,
 *   documentName?: string
 *   maintenanceWindow?: string
 *   winId?: int
 *   ec2InstanceId?: string
 *   requestedDateTime?: Date
 *   opsItemId?: string
 *   askToDeleteResources?: boolean
 * }} State
 */

const defaultMaintenanceWindow = "ssm-maintenance-window";
const defaultDocumentName = "ssmdocument";
// The timeout duration is highly dependent on the specific setup and environment
 necessary. This example handles only the most common error cases, and uses a
 much shorter duration than most productions systems would use.
const COMMAND_TIMEOUT_DURATION_SECONDS = 30; // 30 seconds

const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});

const greet = new ScenarioOutput(
  "greet",
  `Welcome to the AWS Systems Manager SDK Getting Started scenario.
    This program demonstrates how to interact with Systems Manager using the AWS
 SDK for JavaScript V3.
    Systems Manager is the operations hub for your AWS applications and resources
 and a secure end-to-end management solution.
    The program's primary functions include creating a maintenance window,
 creating a document, sending a command to a document,
    listing documents, listing commands, creating an OpsItem, modifying an
 OpsItem, and deleting Systems Manager resources.
    Upon completion of the program, all AWS resources are cleaned up.
    Let's get started...`,
  { header: true },
);

const createMaintenanceWindow = new ScenarioOutput(
  "createMaintenanceWindow",
  "Step 1: Create a Systems Manager maintenance window.",
);
```

```
const getMaintenanceWindow = new ScenarioInput(
  "maintenanceWindow",
  "Please enter the maintenance window name:",
  { type: "input", default: defaultMaintenanceWindow },
);

export const sdkCreateMaintenanceWindow = new ScenarioAction(
  "sdkCreateMaintenanceWindow",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new CreateMaintenanceWindowCommand({
          Name: state.maintenanceWindow,
          Schedule: "cron(0 10 ? * MON-FRI *)", //The schedule of the maintenance
 window in the form of a cron or rate expression.
          Duration: 2, //The duration of the maintenance window in hours.
          Cutoff: 1, //The number of hours before the end of the maintenance
 window that Amazon Web Services Systems Manager stops scheduling new tasks for
 execution.
          AllowUnassociatedTargets: true, //Allow the maintenance window to run
 on managed nodes, even if you haven't registered those nodes as targets.
        }),
      );
      state.winId = response.WindowId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while creating the maintenance window. Please fix the
 error and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const modifyMaintenanceWindow = new ScenarioOutput(
  "modifyMaintenanceWindow",
  "Modify the maintenance window by changing the schedule.",
);

const sdkModifyMaintenanceWindow = new ScenarioAction(
  "sdkModifyMaintenanceWindow",
  async (/** @type {State} */ state) => {
```

```
    try {
      await state.ssmClient.send(
        new UpdateMaintenanceWindowCommand({
          WindowId: state.winId,
          Schedule: "cron(0 0 ? * MON *)",
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while modifying the maintenance window. Please fix the
 error and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const createSystemsManagerActions = new ScenarioOutput(
  "createSystemsManagerActions",
  "Create a document that defines the actions that Systems Manager performs on
 your EC2 instance.",
);

const getDocumentName = new ScenarioInput(
  "documentName",
  "Please enter the document: ",
  { type: "input", default: defaultDocumentName },
);

const sdkCreateSSMDoc = new ScenarioAction(
  "sdkCreateSSMDoc",
  async (/** @type {State} */ state) => {
    const contentData = `{
                "schemaVersion": "2.2",
                "description": "Run a simple shell command",
                "mainSteps": [
                    {
                        "action": "aws:runShellScript",
                        "name": "runEchoCommand",
                        "inputs": {
                          "runCommand": [
                            "echo 'Hello, world!'"
                          ]
```

```
                        }
                    }
                ]
            }`;
    try {
      await state.ssmClient.send(
        new CreateDocumentCommand({
          Content: contentData,
          Name: state.documentName,
          DocumentType: "Command",
        }),
      );
    } catch (caught) {
      console.log(`Exception type: (${typeof caught})`);
      if (caught instanceof DocumentAlreadyExists) {
        console.log("Document already exists. Continuing...\n");
      } else {
        console.error(caught.message);
        console.log(
          `An error occurred while creating the document. Please fix the error
 and try again. Error message: ${caught.message}`,
        );
        throw caught;
      }
    }
  },
);

const ec2HelloWorld = new ScenarioOutput(
  "ec2HelloWorld",
  `Now you have the option of running a command on an EC2 instance that echoes
 'Hello, world!'. In order to run this command, you must provide the instance ID
 of a Linux EC2 instance. If you do not already have a running Linux EC2 instance
 in your account, you can create one using the AWS console. For information
 about creating an EC2 instance, see https://docs.aws.amazon.com/AWSEC2/latest/
UserGuide/ec2-launch-instance-wizard.html.`,
);

const enterIdOrSkipEC2HelloWorld = new ScenarioInput(
  "enterIdOrSkipEC2HelloWorld",
  "Enter your EC2 InstanceId or press enter to skip this step: ",
  { type: "input", default: "" },
);
```

```
const sdkEC2HelloWorld = new ScenarioAction(
  "sdkEC2HelloWorld",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new SendCommandCommand({
          DocumentName: state.documentName,
          InstanceIds: [state.ec2InstanceId],
          TimeoutSeconds: COMMAND_TIMEOUT_DURATION_SECONDS,
        }),
      );
      state.CommandId = response.Command.CommandId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while sending the command. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.enterIdOrSkipEC2HelloWorld === "",
  },
);

const sdkGetCommandTime = new ScenarioAction(
  "sdkGetCommandTime",
  async (/** @type {State} */ state) => {
    const listInvocationsPaginated = [];
    console.log(
      "Let's get the time when the specific command was sent to the specific
 managed node.",
    );

    console.log(
      `First, we'll wait for the command to finish executing. This may take up to
 ${COMMAND_TIMEOUT_DURATION_SECONDS} seconds.`,
    );
    const commandExecutedResult = waitUntilCommandExecuted(
      { client: state.ssmClient },
      {
        CommandId: state.CommandId,
```

```
        InstanceId: state.ec2InstanceId,
      },
    );
    // This is necessary because the TimeoutSeconds of SendCommandCommand is only
for the delivery, not execution.
    try {
      await new Promise((_, reject) =>
        setTimeout(
          reject,
          COMMAND_TIMEOUT_DURATION_SECONDS * 1000,
          new Error("Command Timed Out"),
        ),
      );
    } catch (caught) {
      if (caught.message === "Command Timed Out") {
        commandExecutedResult.state = "TIMED_OUT";
      } else {
        throw caught;
      }
    }

    if (commandExecutedResult.state !== "SUCCESS") {
      console.log(
        `The command with id: ${state.CommandId} did not execute in the allotted
time. Canceling command.`,
      );
      state.ssmClient.send(
        new CancelCommandCommand({
          CommandId: state.CommandId,
        }),
      );
      state.enterIdOrSkipEC2HelloWorld === "";
      return;
    }

    for await (const page of paginateListCommandInvocations(
      { client: state.ssmClient },
      { CommandId: state.CommandId },
    )) {
      listInvocationsPaginated.push(...page.CommandInvocations);
    }
    /**
     * @type {import('@aws-sdk/client-ssm').CommandInvocation}
     */
```

```
    const commandInvocation = listInvocationsPaginated.shift(); // Because the
 call was made with CommandId, there's only one result, so shift it off.
    state.requestedDateTime = commandInvocation.RequestedDateTime;

    console.log(
      `The command invocation happened at: ${state.requestedDateTime}.`,
    );
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.enterIdOrSkipEC2HelloWorld === "",
  },
);

const createSSMOpsItem = new ScenarioOutput(
  "createSSMOpsItem",
  `Now we will create a Systems Manager OpsItem. An OpsItem is a feature provided
 by the Systems Manager service. It is a type of operational data item that
 allows you to manage and track various operational issues, events, or tasks
 within your AWS environment.
You can create OpsItems to track and manage operational issues as they arise.
 For example, you could create an OpsItem whenever your application detects a
 critical error or an anomaly in your infrastructure.`,
);

const sdkCreateSSMOpsItem = new ScenarioAction(
  "sdkCreateSSMOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new CreateOpsItemCommand({
          Description: "Created by the System Manager Javascript API",
          Title: "Disk Space Alert",
          Source: "EC2",
          Category: "Performance",
          Severity: "2",
        }),
      );
      state.opsItemId = response.OpsItemId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while creating the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
```

```
      );
      throw caught;
    }
  },
);

const updateOpsItem = new ScenarioOutput(
  "updateOpsItem",
  (/** @type {State} */ state) =>
    `Now we will update the OpsItem: ${state.opsItemId}`,
);

const sdkUpdateOpsItem = new ScenarioAction(
  "sdkUpdateOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const _response = await state.ssmClient.send(
        new UpdateOpsItemCommand({
          OpsItemId: state.opsItemId,
          Description: `An update to ${state.opsItemId}`,
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while updating the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const getOpsItemStatus = new ScenarioOutput(
  "getOpsItemStatus",
  (/** @type {State} */ state) =>
    `Now we will get the status of the OpsItem: ${state.opsItemId}`,
);

const sdkOpsItemStatus = new ScenarioAction(
  "sdkGetOpsItemStatus",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
```

```
        new DescribeOpsItemsCommand({
          OpsItemId: state.opsItemId,
        }),
      );
      state.opsItemStatus = response.OpsItemStatus;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while describing the ops item. Please fix the error
 and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const resolveOpsItem = new ScenarioOutput(
  "resolveOpsItem",
  (/** @type {State} */ state) =>
    `Now we will resolve the OpsItem: ${state.opsItemId}`,
);

const sdkResolveOpsItem = new ScenarioAction(
  "sdkResolveOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const _response = await state.ssmClient.send(
        new UpdateOpsItemCommand({
          OpsItemId: state.opsItemId,
          Status: OpsItemStatus.RESOLVED,
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while updating the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const askToDeleteResources = new ScenarioInput(
```

```
  "askToDeleteResources",
  "Would you like to delete the Systems Manager resources created during this
 example run?",
  { type: "confirm" },
);

const confirmDeleteChoice = new ScenarioOutput(
  "confirmDeleteChoice",
  (/** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You chose to delete the resources.";
    }
    return "The Systems Manager resources will not be deleted. Please delete them
 manually to avoid charges.";
  },
);

export const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    try {
      await state.ssmClient.send(
        new DeleteOpsItemCommand({
          OpsItemId: state.opsItemId,
        }),
      );
      console.log(`The ops item: ${state.opsItemId} was successfully deleted.`);
    } catch (caught) {
      console.log(
        `There was a problem deleting the ops item: ${state.opsItemId}. Please
 delete it manually. Error: ${caught.message}`,
      );
    }

    try {
      await state.ssmClient.send(
        new DeleteMaintenanceWindowCommand({
          Name: state.maintenanceWindow,
          WindowId: state.winId,
        }),
      );
      console.log(
        `The maintenance window: ${state.maintenanceWindow} was successfully
 deleted.`,
```

```
      );
    } catch (caught) {
      console.log(
        `There was a problem deleting the maintenance window: ${state.opsItemId}.
 Please delete it manually. Error: ${caught.message}`,
      );
    }

    try {
      await state.ssmClient.send(
        new DeleteDocumentCommand({
          Name: state.documentName,
        }),
      );
      console.log(
        `The document: ${state.documentName} was successfully deleted.`,
      );
    } catch (caught) {
      console.log(
        `There was a problem deleting the document: ${state.documentName}. Please
 delete it manually. Error: ${caught.message}`,
      );
    }
  },
  { skipWhen: (/** @type {{}} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the Systems Manager Basics scenario for the AWS Javascript SDK
 v3. Thank you!",
);

const myScenario = new Scenario(
  "SSM Basics",
  [
    greet,
    pressEnter,
    createMaintenanceWindow,
    getMaintenanceWindow,
    sdkCreateMaintenanceWindow,
    modifyMaintenanceWindow,
    pressEnter,
    sdkModifyMaintenanceWindow,
```

```
    createSystemsManagerActions,
    getDocumentName,
    sdkCreateSSMDoc,
    ec2HelloWorld,
    enterIdOrSkipEC2HelloWorld,
    sdkEC2HelloWorld,
    sdkGetCommandTime,
    pressEnter,
    createSSMOpsItem,
    pressEnter,
    sdkCreateSSMOpsItem,
    updateOpsItem,
    pressEnter,
    sdkUpdateOpsItem,
    getOpsItemStatus,
    pressEnter,
    sdkOpsItemStatus,
    resolveOpsItem,
    pressEnter,
    sdkResolveOpsItem,
    askToDeleteResources,
    confirmDeleteChoice,
    sdkDeleteResources,
    goodbye,
  ],
  { ssmClient: new SSMClient({}) },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
```

```
}
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run an interactive scenario at a command prompt.

```
class SystemsManagerScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
 Systems Manager."""

    def __init__(self, document_wrapper, maintenance_window_wrapper,
 ops_item_wrapper):
        """
        :param document_wrapper: An object that wraps Systems Manager document
 functions.
        :param maintenance_window_wrapper: An object that wraps Systems Manager
 maintenance window functions.
        :param ops_item_wrapper: An object that wraps Systems Manager OpsItem
 functions.
        """
        self.document_wrapper = document_wrapper
```

```
        self.maintenance_window_wrapper = maintenance_window_wrapper
        self.ops_item_wrapper = ops_item_wrapper

    def run(self):
        """Demonstrates how to use the AWS SDK for Python (Boto3) to get started
 with Systems Manager."""
        try:
            print("-" * 88)
            print(
                """
Welcome to the AWS Systems Manager SDK Getting Started scenario.
This program demonstrates how to interact with Systems Manager using the AWS SDK
 for Python (Boto3).
Systems Manager is the operations hub for your AWS applications and resources and
 a secure end-to-end management
solution. The program's primary functions include creating a maintenance window,
 creating a document, sending a
command to a document, listing documents, listing commands, creating an OpsItem,
 modifying an OpsItem, and deleting
Systems Manager resources. Upon completion of the program, all AWS resources are
 cleaned up.
Let's get started..."""
            )
            q.ask("Please hit Enter")

            print("-" * 88)
            print("Create a Systems Manager maintenance window.")
            maintenance_window_name = q.ask(
                "Please enter the maintenance window name (default is ssm-
maintenance-window):",
            )
            if not maintenance_window_name:
                maintenance_window_name = "ssm-maintenance-window"

            self.maintenance_window_wrapper.create(
                name=maintenance_window_name,
                schedule="cron(0 10 ? * MON-FRI *)",
                duration=2,
                cutoff=1,
                allow_unassociated_targets=True,
            )

            print("-" * 88)
            print("Modify the maintenance window by changing the schedule")
```

```
            q.ask("Please hit Enter")

            self.maintenance_window_wrapper.update(
                name=maintenance_window_name,
                schedule="cron(0 0 ? * MON *)",
                duration=24,
                cutoff=1,
                allow_unassociated_targets=True,
                enabled=True,
            )

            print("-" * 88)
            print(
                "Create a document that defines the actions that Systems Manager
 performs on your EC2 instance."
            )
            document_name = q.ask(
                "Please enter the document name (default is ssmdocument):"
            )

            if not document_name:
                document_name = "ssmdocument"

            self.document_wrapper.create(
                name=document_name,
                content="""
{
    "schemaVersion": "2.2",
    "description": "Run a simple shell command",
    "mainSteps": [
        {
            "action": "aws:runShellScript",
            "name": "runEchoCommand",
            "inputs": {
              "runCommand": [
                "echo 'Hello, world!'"
              ]
            }
        }
    ]
}
            """,
            )
```

```
            self.document_wrapper.wait_until_active()

            print(
                """
Now you have the option of running a command on an EC2 instance that echoes
 'Hello, world!'.
In order to run this command, you must provide the instance ID of a Linux EC2
 instance. If you do
not already have a running Linux EC2 instance in your account, you can create one
 using the AWS console.
For information about creating an EC2 instance, see
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-instance-
wizard.html.
            """
            )

            if q.ask(
                "Would you like to run a command on an EC2 instance? (y/n)",
                q.is_yesno,
            ):
                instance_id = q.ask(
                    "Please enter the instance ID of the EC2 instance:",
 q.non_empty
                )
                command_id = self.document_wrapper.send_command(
                    instance_ids=[instance_id]
                )

                self.document_wrapper.wait_command_executed(
                    command_id=command_id, instance_id=instance_id
                )

                print("-" * 88)
                print(
                    "Lets get the time when the specific command was sent to the
 specific managed node"
                )
                q.ask("Please hit Enter")


 self.document_wrapper.list_command_invocations(instance_id=instance_id)

            print("-" * 88)
            print("-" * 88)
```

```
            print(
                """
Now we will create a  Systems Manager OpsItem.
An OpsItem is a feature provided by the Systems Manager service.
It is a type of operational data item that allows you to manage and track various
 operational issues,
events, or tasks within your AWS environment.

You can create OpsItems to track and manage operational issues as they arise.
For example, you could create an OpsItem whenever your application detects a
 critical error
or an anomaly in your infrastructure.
            """
            )
            q.ask("Please hit Enter")

            self.ops_item_wrapper.create(
                title="Disk Space Alert",
                description="Created by the Systems Manager Python (Boto3) API",
                source="EC2",
                category="Performance",
                severity="2",
            )

            print("-" * 88)
            print("-" * 88)
            print(f"Now we will update  the OpsItem {self.ops_item_wrapper.id}")
            q.ask("Please hit Enter")

            self.ops_item_wrapper.update(
                title="Disk Space Alert",
                description=f"An update to {self.ops_item_wrapper.id}",
            )

            print(
                f"Now we will get the status of the OpsItem
 {self.ops_item_wrapper.id}"
            )
            q.ask("Please hit Enter")

            # It may take a second for the ops item to be available
            counter = 0
            while not self.ops_item_wrapper.describe() and counter < 5:
                counter += 1
```

```
                time.sleep(1)

            print(f"Now we will resolve the OpsItem {self.ops_item_wrapper.id}")
            q.ask("Please hit Enter")

            self.ops_item_wrapper.update(status="Resolved")

            print("-" * 88)
            print("-" * 88)
            if q.ask(
                "Would you like to delete the Systems Manager resources? (y/n)",
                q.is_yesno,
            ):
                print("You selected to delete the resources.")
                self.cleanup()
            else:
                print("The Systems Manager resources will not be deleted")

            print("-" * 88)
            print("This concludes the Systems Manager SDK Getting Started
 scenario.")
            print("-" * 88)

        except Exception:
            self.cleanup()
            raise

    def cleanup(self):
        self.maintenance_window_wrapper.delete()
        self.ops_item_wrapper.delete()
        self.document_wrapper.delete()


if __name__ == "__main__":
    try:
        scenario = SystemsManagerScenario(
            DocumentWrapper.from_client(),
            MaintenanceWindowWrapper.from_client(),
            OpsItemWrapper.from_client(),
        )
        scenario.run()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Define a class that wraps document and command actions.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, content, name):
        """
        Creates a document.

        :param content: The content of the document.
        :param name: The name of the document.
        """
        try:
            self.ssm_client.create_document(
                Name=name, Content=content, DocumentType="Command"
            )
            self.name = name
        except self.ssm_client.exceptions.DocumentAlreadyExists:
            print(f"Document {name} already exists.")
            self.name = name
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def delete(self):
        """
        Deletes an AWS Systems Manager document.
        """
        if self.name is None:
            return

        try:
            self.ssm_client.delete_document(Name=self.name)
            print(f"Deleted document {self.name}.")
            self.name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def send_command(self, instance_ids):
        """
        Sends a command to one or more instances.

        :param instance_ids: The IDs of the instances to send the command to.
        :return: The ID of the command.
        """
        try:
            response = self.ssm_client.send_command(
                InstanceIds=instance_ids, DocumentName=self.name,
TimeoutSeconds=3600
            )
            return response["Command"]["CommandId"]
        except ClientError as err:
            logger.error(
                "Couldn't send command to %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def describe(self):
        """
        Describes the document.

        :return: Document status.
        """
        try:
            response = self.ssm_client.describe_document(Name=self.name)
            return response["Document"]["Status"]
        except ClientError as err:
            logger.error(
                "Couldn't get %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def wait_until_active(self, max_attempts=20, delay=5):
        """
        Waits until the document is active.

        :param max_attempts: The maximum number of attempts for checking the
status.
        :param delay: The delay in seconds between each check.
        """
        attempt = 0
        status = ""
        while attempt <= max_attempts:
            status = self.describe()
            if status == "Active":
                break
            attempt += 1
            time.sleep(delay)

        if status != "Active":
            logger.error("Document is not active.")
        else:
            logger.info("Document is active.")

    def wait_command_executed(self, command_id, instance_id):
```

```
        """
        Waits until the command is executed on the instance.

        :param command_id: The ID of the command.
        :param instance_id: The ID of the instance.
        """

        waiter = self.ssm_client.get_waiter("command_executed")
        waiter.wait(CommandId=command_id, InstanceId=instance_id)

    def list_command_invocations(self, instance_id):
        """
        Lists the commands for an instance.

        :param instance_id: The ID of the instance.
        :return: The list of commands.
        """
        try:
            paginator = self.ssm_client.get_paginator("list_command_invocations")
            command_invocations = []
            for page in paginator.paginate(InstanceId=instance_id):
                command_invocations.extend(page["CommandInvocations"])
            num_of_commands = len(command_invocations)
            print(
                f"{num_of_commands} command invocation(s) found for instance
{instance_id}."
            )

            if num_of_commands > 10:
                print("Displaying the first 10 commands:")
                num_of_commands = 10
            date_format = "%A, %d %B %Y %I:%M%p"
            for command in command_invocations[:num_of_commands]:
                print(
                    f"   The time of command invocation is
{command['RequestedDateTime'].strftime(date_format)}"
                )
        except ClientError as err:
            logger.error(
                "Couldn't list commands for %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

Define a class that wraps ops item actions.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, title, source, category, severity, description):
        """
        Create an OpsItem

        :param title: The OpsItem title.
        :param source: The OpsItem source.
        :param category: The OpsItem category.
        :param severity: The OpsItem severity.
        :param description: The OpsItem description.

        """
        try:
            response = self.ssm_client.create_ops_item(
                Title=title,
                Source=source,
                Category=category,
```

```
                Severity=severity,
                Description=description,
            )
            self.id = response["OpsItemId"]
        except self.ssm_client.exceptions.OpsItemLimitExceededException as err:
            logger.error(
                "Couldn't create ops item because you have exceeded your open
OpsItem limit. "
                "Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create ops item %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete(self):
        """
        Delete the OpsItem.
        """
        if self.id is None:
            return
        try:
            self.ssm_client.delete_ops_item(OpsItemId=self.id)
            print(f"Deleted ops item with id {self.id}")
            self.id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe(self):
        """
```

```
        Describe an OpsItem.
        """
        try:
            paginator = self.ssm_client.get_paginator("describe_ops_items")
            ops_items = []
            for page in paginator.paginate(
                OpsItemFilters=[
                    {"Key": "OpsItemId", "Values": [self.id], "Operator":
"Equal"}
                ]
            ):
                ops_items.extend(page["OpsItemSummaries"])

            for item in ops_items:
                print(
                    f"The item title is {item['Title']} and the status is
{item['Status']}"
                )
            return len(ops_items) > 0
        except ClientError as err:
            logger.error(
                "Couldn't describe ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update(self, title=None, description=None, status=None):
        """
        Update an OpsItem.

        :param title: The new OpsItem title.
        :param description: The new OpsItem description.
        :param status: The new OpsItem status.
        :return:
        """
        args = dict(OpsItemId=self.id)
        if title is not None:
            args["Title"] = title
        if description is not None:
            args["Description"] = description
        if status is not None:
```

```
            args["Status"] = status
        try:
            self.ssm_client.update_ops_item(**args)
        except ClientError as err:
            logger.error(
                "Couldn't update ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Define a class that wraps maintenance window actions.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, name, schedule, duration, cutoff,
 allow_unassociated_targets):
        """
        Create an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
```

```
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes
as targets.
        """
        try:
            response = self.ssm_client.create_maintenance_window(
                Name=name,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.window_id = response["WindowId"]
            self.name = name
            logger.info("Created maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to create maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create maintenance window %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self):
        """
        Delete the associated AWS Systems Manager maintenance window.
        """
        if self.window_id is None:
            return

        try:
            self.ssm_client.delete_maintenance_window(WindowId=self.window_id)
```

```
            logger.info("Deleted maintenance window %s.", self.window_id)
            print(f"Deleted maintenance window {self.name}")
            self.window_id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete maintenance window %s. Here's why: %s: %s",
                self.window_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update(
        self, name, enabled, schedule, duration, cutoff,
allow_unassociated_targets
    ):
        """
        Update an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param enabled: Whether the maintenance window is enabled to run on
managed nodes.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes
as targets.
        """
        try:
            self.ssm_client.update_maintenance_window(
                WindowId=self.window_id,
                Name=name,
                Enabled=enabled,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.name = name
            logger.info("Updated maintenance window %s.", self.window_id)
        except ParamValidationError as error:
```

```
            logger.error(
                "Parameter validation error when trying to update maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't update maintenance window %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Actions for Systems Manager using AWS SDKs

The following code examples demonstrate how to perform individual Systems Manager actions with AWS SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

The following examples include only the most commonly used actions. For a complete list, see the AWS Systems Manager API Reference.

Examples

- Use AddTagsToResource with a CLI
- Use CancelCommand with a CLI
- Use CreateActivation with a CLI
- Use CreateAssociation with a CLI
- Use CreateAssociationBatch with a CLI
- Use CreateDocument with an AWS SDK or CLI
- Use CreateMaintenanceWindow with an AWS SDK or CLI
- Use CreateOpsItem with an AWS SDK or CLI
- Use CreatePatchBaseline with a CLI
- Use DeleteActivation with a CLI
- Use DeleteAssociation with a CLI
- Use DeleteDocument with an AWS SDK or CLI
- Use DeleteMaintenanceWindow with an AWS SDK or CLI
- Use DeleteOpsItem with an AWS SDK
- Use DeleteParameter with a CLI
- Use DeletePatchBaseline with a CLI
- Use DeregisterManagedInstance with a CLI
- Use DeregisterPatchBaselineForPatchGroup with a CLI
- Use DeregisterTargetFromMaintenanceWindow with a CLI
- Use DeregisterTaskFromMaintenanceWindow with a CLI
- Use DescribeActivations with a CLI
- Use DescribeAssociation with a CLI
- Use DescribeAssociationExecutionTargets with a CLI
- Use DescribeAssociationExecutions with a CLI
- Use DescribeAutomationExecutions with a CLI
- Use DescribeAutomationStepExecutions with a CLI
- Use DescribeAvailablePatches with a CLI
- Use DescribeDocument with a CLI
- Use DescribeDocumentPermission with a CLI

- Use DescribeEffectiveInstanceAssociations with a CLI
- Use DescribeEffectivePatchesForPatchBaseline with a CLI
- Use DescribeInstanceAssociationsStatus with a CLI
- Use DescribeInstanceInformation with a CLI
- Use DescribeInstancePatchStates with a CLI
- Use DescribeInstancePatchStatesForPatchGroup with a CLI
- Use DescribeInstancePatches with a CLI
- Use DescribeMaintenanceWindowExecutionTaskInvocations with a CLI
- Use DescribeMaintenanceWindowExecutionTasks with a CLI
- Use DescribeMaintenanceWindowExecutions with a CLI
- Use DescribeMaintenanceWindowTargets with a CLI
- Use DescribeMaintenanceWindowTasks with a CLI
- Use DescribeMaintenanceWindows with a CLI
- Use DescribeOpsItems with an AWS SDK or CLI
- Use DescribeParameters with an AWS SDK or CLI
- Use DescribePatchBaselines with a CLI
- Use DescribePatchGroupState with a CLI
- Use DescribePatchGroups with a CLI
- Use GetAutomationExecution with a CLI
- Use GetCommandInvocation with a CLI
- Use GetConnectionStatus with a CLI
- Use GetDefaultPatchBaseline with a CLI
- Use GetDeployablePatchSnapshotForInstance with a CLI
- Use GetDocument with a CLI
- Use GetInventory with a CLI
- Use GetInventorySchema with a CLI
- Use GetMaintenanceWindow with a CLI
- Use GetMaintenanceWindowExecution with a CLI
- Use GetMaintenanceWindowExecutionTask with a CLI

- Use GetParameter with an AWS SDK or CLI
- Use GetParameterHistory with a CLI
- Use GetParameters with a CLI
- Use GetPatchBaseline with a CLI
- Use GetPatchBaselineForPatchGroup with a CLI
- Use ListAssociationVersions with a CLI
- Use ListAssociations with a CLI
- Use ListCommandInvocations with an AWS SDK or CLI
- Use ListCommands with a CLI
- Use ListComplianceItems with a CLI
- Use ListComplianceSummaries with a CLI
- Use ListDocumentVersions with a CLI
- Use ListDocuments with a CLI
- Use ListInventoryEntries with a CLI
- Use ListResourceComplianceSummaries with a CLI
- Use ListTagsForResource with a CLI
- Use ModifyDocumentPermission with a CLI
- Use PutComplianceItems with a CLI
- Use PutInventory with a CLI
- Use PutParameter with an AWS SDK or CLI
- Use RegisterDefaultPatchBaseline with a CLI
- Use RegisterPatchBaselineForPatchGroup with a CLI
- Use RegisterTargetWithMaintenanceWindow with a CLI
- Use RegisterTaskWithMaintenanceWindow with a CLI
- Use RemoveTagsFromResource with a CLI
- Use SendCommand with an AWS SDK or CLI
- Use StartAutomationExecution with a CLI
- Use StartSession with a CLI
- Use StopAutomationExecution with a CLI
- Use UpdateAssociation with a CLI

- Use UpdateAssociationStatus with a CLI
- Use UpdateDocument with a CLI
- Use UpdateDocumentDefaultVersion with a CLI
- Use UpdateMaintenanceWindow with an AWS SDK or CLI
- Use UpdateManagedInstanceRole with a CLI
- Use UpdateOpsItem with an AWS SDK or CLI
- Use UpdatePatchBaseline with a CLI

Use `AddTagsToResource` with a CLI

The following code examples show how to use `AddTagsToResource`.

CLI

AWS CLI

Example 1: To add tags to a maintenance window

The following `add-tags-to-resource` example adds a tag to the specified maintenance window.

```
aws ssm add-tags-to-resource \
    --resource-type "MaintenanceWindow" \
    --resource-id "mw-03eb9db428EXAMPLE" \
    --tags "Key=Stack,Value=Production"
```

This command produces no output.

Example 2: To add tags to a parameter

The following `add-tags-to-resource` example adds two tags to to the specified parameter.

```
aws ssm add-tags-to-resource \
    --resource-type "Parameter" \
    --resource-id "My-Parameter" \
    --tags '[{"Key":"Region","Value":"East"},{"Key":"Environment",
 "Value":"Production"}]'
```

This command produces no output.

Example 3: To add tags to an SSM document

The following `add-tags-to-resource` example adds a tag to to the specified document.

```
aws ssm add-tags-to-resource \
    --resource-type "Document" \
    --resource-id "My-Document" \
    --tags "Key=Quarter,Value=Q322"
```

This command produces no output.

For more information, see Tagging Systems Manager resources in the *AWS Systems Manager User Guide*.

- For API details, see AddTagsToResource in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example updates a maintenance window with new tags. There is no output if the command succeeds. The syntax used by this example requires PowerShell version 3 or later.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Example 2: With PowerShell version 2, you must use New-Object to create each tag. There is no output if the command succeeds.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- For API details, see AddTagsToResource in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example updates a maintenance window with new tags. There is no output if the command succeeds. The syntax used by this example requires PowerShell version 3 or later.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Example 2: With PowerShell version 2, you must use New-Object to create each tag. There is no output if the command succeeds.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- For API details, see AddTagsToResource in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CancelCommand` with a CLI

The following code examples show how to use `CancelCommand`.

CLI

AWS CLI

Example 1: To cancel a command for all instances

The following `cancel-command` example attempts to cancel the specified command that is already running for all instances.

```
aws ssm cancel-command \
    --command-id "662add3d-5831-4a10-b64a-f2ff3EXAMPLE"
```

This command produces no output.

Example 2: To cancel a command for specific instances

The following `cancel-command` example attempts to cancel a command for the specified instance only.

```
aws ssm cancel-command \
    --command-id "662add3d-5831-4a10-b64a-f2ff3EXAMPLE"
    --instance-ids "i-02573cafcfEXAMPLE"
```

This command produces no output.

For more information, see Tagging Systems Manager Parameters in the *AWS Systems Manager User Guide*.

- For API details, see CancelCommand in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example attempts to cancel a command. There is no output if the operation succeeds.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- For API details, see CancelCommand in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example attempts to cancel a command. There is no output if the operation succeeds.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- For API details, see CancelCommand in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateActivation` with a CLI

The following code examples show how to use `CreateActivation`.

CLI

AWS CLI

To create a managed instance activation

The following `create-activation` example creates a managed instance activation.

```
aws ssm create-activation \
    --default-instance-name "HybridWebServers" \
    --iam-role "HybridWebServersRole" \
    --registration-limit 5
```

Output:

```
{
    "ActivationId": "5743558d-563b-4457-8682-d16c3EXAMPLE",
    "ActivationCode": "dRmgnYaFv567vEXAMPLE"
}
```

For more information, see Step 4: Create a Managed-Instance Activation for a Hybrid Environment in the *AWS Systems Manager User Guide*.

- For API details, see CreateActivation in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example creates a managed instance.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole
 "SSMAutomationRole" -RegistrationLimit 10
```

Output:

```
ActivationCode       ActivationId
--------------       ------------
```

```
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- For API details, see CreateActivation in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example creates a managed instance.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole
 "SSMAutomationRole" -RegistrationLimit 10
```

Output:

```
ActivationCode       ActivationId
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- For API details, see CreateActivation in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateAssociation` with a CLI

The following code examples show how to use `CreateAssociation`.

CLI

AWS CLI

Example 1: To associate a document using instance IDs

This example associates a configuration document with an instance, using instance IDs.

```
aws ssm create-association \
    --instance-id "i-0cb2b964d3e14fd9f" \
    --name "AWS-UpdateSSMAgent"
```

Output:

```
{
```

```
    "AssociationDescription": {
        "Status": {
            "Date": 1487875500.33,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-0cb2b964d3e14fd9f",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487875500.33,
        "Date": 1487875500.33,
        "Targets": [
            {
                "Values": [
                    "i-0cb2b964d3e14fd9f"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

For more information, see CreateAssociation in the *AWS Systems Manager API Reference*.

Example 2: To associate a document using targets

This example associates a configuration document with an instance, using targets.

```
aws ssm create-association \
    --name "AWS-UpdateSSMAgent" \
    --targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f"
```

Output:

```
{
    "AssociationDescription": {
        "Status": {
            "Date": 1487875500.33,
```

```
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-0cb2b964d3e14fd9f",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487875500.33,
        "Date": 1487875500.33,
        "Targets": [
            {
                "Values": [
                    "i-0cb2b964d3e14fd9f"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

For more information, see CreateAssociation in the *AWS Systems Manager API Reference*.

Example 3: To create an association that runs only once

This example creates a new association that only runs once on the specified date and time. Associations created with a date in the past or present (by the time it is processed the date is in the past) run immediately.

```
aws ssm create-association \
    --name "AWS-UpdateSSMAgent" \
    --targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f" \
    --schedule-expression "at(2020-05-14T15:55:00)"  \
    --apply-only-at-cron-interval
```

Output:

```
{
    "AssociationDescription": {
```

```
        "Status": {
            "Date": 1487875500.33,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-0cb2b964d3e14fd9f",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487875500.33,
        "Date": 1487875500.33,
        "Targets": [
            {
                "Values": [
                    "i-0cb2b964d3e14fd9f"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

For more information, see CreateAssociation in the *AWS Systems Manager API Reference* or Reference: Cron and rate expressions for Systems Manager in the *AWS Systems Manager User Guide*.

- For API details, see CreateAssociation in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example associates a configuration document with an instance, using instance IDs.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Output:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Associated
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : Associated with AWS-UpdateSSMAgent
Status.AdditionalInfo :
```

Example 2: This example associates a configuration document with an instance, using targets.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Output:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Example 3: This example associates a configuration document with an instance, using targets and parameters.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
$params = @{
  "action"="configure"
  "mode"="ec2"
  "optionalConfigurationSource"="ssm"
  "optionalConfigurationLocation"=""
  "optionalRestart"="yes"
}
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName
 "CWConfiguration" -Target $target -Parameter $params
```

Output:

```
Name                  : Configure-CloudWatch
```

```
InstanceId            :
Date                  : 5/17/2018 3:17:44 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Example 4: This example creates an association with all instances in the region, with `AWS-GatherSoftwareInventory`. It also provides custom files and registry locations in the parameters to collect

```
$params =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$params["windowsRegistry"] ='[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
 -Target @{Key="instanceids";Values="*"} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

Output:

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- For API details, see CreateAssociation in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example associates a configuration document with an instance, using instance IDs.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Output:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Associated
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : Associated with AWS-UpdateSSMAgent
Status.AdditionalInfo :
```

Example 2: This example associates a configuration document with an instance, using targets.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Output:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Example 3: This example associates a configuration document with an instance, using targets and parameters.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
$params = @{
  "action"="configure"
  "mode"="ec2"
  "optionalConfigurationSource"="ssm"
  "optionalConfigurationLocation"=""
  "optionalRestart"="yes"
}
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName
 "CWConfiguration" -Target $target -Parameter $params
```

Output:

```
Name                  : Configure-CloudWatch
```

```
InstanceId            :
Date                  : 5/17/2018 3:17:44 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Example 4: This example creates an association with all instances in the region, with `AWS-GatherSoftwareInventory`. It also provides custom files and registry locations in the parameters to collect

```
$params =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$params["windowsRegistry"] ='[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
 -Target @{Key="instanceids";Values="*"} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

Output:

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- For API details, see CreateAssociation in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateAssociationBatch` with a CLI

The following code examples show how to use `CreateAssociationBatch`.

CLI

AWS CLI

To create multiple associations

This example associates a configuration document with multiple instances. The output returns a list of successful and failed operations, if applicable.

Command:

```
aws ssm create-association-batch --entries "Name=AWS-
UpdateSSMAgent,InstanceId=i-1234567890abcdef0" "Name=AWS-
UpdateSSMAgent,InstanceId=i-9876543210abcdef0"
```

Output:

```
{
  "Successful": [
      {
          "Name": "AWS-UpdateSSMAgent",
          "InstanceId": "i-1234567890abcdef0",
          "AssociationVersion": "1",
          "Date": 1550504725.007,
          "LastUpdateAssociationDate": 1550504725.007,
          "Status": {
              "Date": 1550504725.007,
              "Name": "Associated",
              "Message": "Associated with AWS-UpdateSSMAgent"
          },
          "Overview": {
              "Status": "Pending",
              "DetailedStatus": "Creating"
          },
          "DocumentVersion": "$DEFAULT",
          "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
          "Targets": [
              {
                  "Key": "InstanceIds",
                  "Values": [
                      "i-1234567890abcdef0"
                  ]
              }
```

```
            ]
        },
        {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-9876543210abcdef0",
            "AssociationVersion": "1",
            "Date": 1550504725.057,
            "LastUpdateAssociationDate": 1550504725.057,
            "Status": {
                "Date": 1550504725.057,
                "Name": "Associated",
                "Message": "Associated with AWS-UpdateSSMAgent"
            },
            "Overview": {
                "Status": "Pending",
                "DetailedStatus": "Creating"
            },
            "DocumentVersion": "$DEFAULT",
            "AssociationId": "9c9f7f20-5154-4fed-a83e-0123456789ab",
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-9876543210abcdef0"
                    ]
                }
            ]
        }
    ],
    "Failed": []
}
```

- For API details, see CreateAssociationBatch in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example associates a configuration document with multiple instances. The output returns a list of successful and failed operations, if applicable.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
```

```
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Output:

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Example 2: This example will show the full details of a successful operation.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- For API details, see CreateAssociationBatch in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example associates a configuration document with multiple instances. The output returns a list of successful and failed operations, if applicable.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Output:

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Example 2: This example will show the full details of a successful operation.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- For API details, see CreateAssociationBatch in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateDocument` with an AWS SDK or CLI

The following code examples show how to use `CreateDocument`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

CLI

AWS CLI

To create a document

The following `create-document` example creates a Systems Manager document.

```
aws ssm create-document \
    --content file://exampleDocument.yml \
    --name "Example" \
    --document-type "Automation" \
    --document-format YAML
```

Output:

```
{
    "DocumentDescription": {
        "Hash":
 "fc2410281f40779e694a8b95975d0f9f316da8a153daa94e3d9921102EXAMPLE",
        "HashType": "Sha256",
        "Name": "Example",
        "Owner": "29884EXAMPLE",
        "CreatedDate": 1583256349.452,
```

```
        "Status": "Creating",
        "DocumentVersion": "1",
        "Description": "Document Example",
        "Parameters": [
            {
                "Name": "AutomationAssumeRole",
                "Type": "String",
                "Description": "(Required) The ARN of the role that allows
 Automation to perform the actions on your behalf. If no role is specified,
 Systems Manager Automation uses your IAM permissions to execute this document.",
                "DefaultValue": ""
            },
            {
                "Name": "InstanceId",
                "Type": "String",
                "Description": "(Required) The ID of the Amazon EC2 instance.",
                "DefaultValue": ""
            }
        ],
        "PlatformTypes": [
            "Windows",
            "Linux"
        ],
        "DocumentType": "Automation",
        "SchemaVersion": "0.3",
        "LatestVersion": "1",
        "DefaultVersion": "1",
        "DocumentFormat": "YAML",
        "Tags": []
    }
}
```

For more information, see Creating Systems Manager Documents in the *AWS Systems Manager User Guide*.

- For API details, see CreateDocument in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Creates an AWS SSM document asynchronously.
 *
 * @param docName The name of the document to create.
 * <p>
 * This method initiates an asynchronous request to create an SSM document.
 * If the request is successful, it prints the document status.
 * If an exception occurs, it handles the error appropriately.
 */
public void createSSMDoc(String docName) throws SsmException {
    String jsonData = """
    {
    "schemaVersion": "2.2",
    "description": "Run a simple shell command",
    "mainSteps": [
        {
            "action": "aws:runShellScript",
            "name": "runEchoCommand",
            "inputs": {
              "runCommand": [
                "echo 'Hello, world!'"
              ]
            }
          }
        ]
    }
    """;

    CreateDocumentRequest request = CreateDocumentRequest.builder()
            .content(jsonData)
            .name(docName)
            .documentType(DocumentType.COMMAND)
```

```
                .build();

        CompletableFuture<CreateDocumentResponse> future =
 getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
 response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

- For API details, see CreateDocument in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { CreateDocumentCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ content, name, documentType }) => {
  const client = new SSMClient({});
```

```
  try {
    const { documentDescription } = await client.send(
      new CreateDocumentCommand({
        Content: content, // The content for the new SSM document. The content
 must not exceed 64KB.
        Name: name,
        DocumentType: documentType, // Document format type can be JSON, YAML, or
 TEXT. The default format is JSON.
      }),
    );
    console.log("Document created successfully.");
    return { DocumentDescription: documentDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DocumentAlreadyExists") {
      console.warn(`${caught.message}. Did you provide a new document name?`);
    } else {
      throw caught;
    }
  }
};
```

- For API details, see CreateDocument in *AWS SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example creates a document in your account. The document must be in JSON format. For more information about writing a configuration document, see Configuration Document in the SSM API Reference.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Output:

```
CreatedDate     : 3/1/2017 1:21:33 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
```

```
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Creating
```

- For API details, see CreateDocument in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example creates a document in your account. The document must be in JSON format. For more information about writing a configuration document, see Configuration Document in the SSM API Reference.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Output:

```
CreatedDate     : 3/1/2017 1:21:33 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Creating
```

- For API details, see CreateDocument in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, content, name):
        """
        Creates a document.

        :param content: The content of the document.
        :param name: The name of the document.
        """
        try:
            self.ssm_client.create_document(
                Name=name, Content=content, DocumentType="Command"
            )
            self.name = name
        except self.ssm_client.exceptions.DocumentAlreadyExists:
            print(f"Document {name} already exists.")
            self.name = name
        except ClientError as err:
```

```
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see CreateDocument in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateMaintenanceWindow` with an AWS SDK or CLI

The following code examples show how to use `CreateMaintenanceWindow`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

CLI

AWS CLI

Example 1: To create a maintenance window

The following `create-maintenance-window` example creates a new maintenance window that every five minutes for up to two hours (as needed), prevents new tasks from starting within one hour of the end of the maintenance window execution, allows unassociated targets (instances that you haven't registered with the maintenance window), and indicates through the use of custom tags that its creator intends to use it in a tutorial.

```
aws ssm create-maintenance-window \
    --name "My-Tutorial-Maintenance-Window" \
    --schedule "rate(5 minutes)" \
    --duration 2 --cutoff 1 \
```

```
    --allow-unassociated-targets \
    --tags "Key=Purpose,Value=Tutorial"
```

Output:

```
{
    "WindowId": "mw-0c50858d01EXAMPLE"
}
```

Example 2: To create a maintenance window that runs only once

The following `create-maintenance-window` example creates a new maintenance window that only runs one time on the specified date and time.

```
aws ssm create-maintenance-window \
    --name My-One-Time-Maintenance-Window \
    --schedule "at(2020-05-14T15:55:00)" \
    --duration 5 \
    --cutoff 2 \
    --allow-unassociated-targets \
    --tags "Key=Environment,Value=Production"
```

Output:

```
{
    "WindowId": "mw-01234567890abcdef"
}
```

For more information, see Maintenance Windows in the *AWS Systems Manager User Guide*.

- For API details, see CreateMaintenanceWindow in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /**
    * Creates an SSM maintenance window asynchronously.
    *
    * @param winName The name of the maintenance window.
    * @return The ID of the created or existing maintenance window.
    * <p>
    * This method initiates an asynchronous request to create an SSM maintenance
window.
    * If the request is successful, it prints the maintenance window ID.
    * If an exception occurs, it handles the error appropriately.
    */
   public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
       CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
               .name(winName)
               .description("This is my maintenance window")
               .allowUnassociatedTargets(true)
               .duration(2)
               .cutoff(1)
               .schedule("cron(0 10 ? * MON-FRI *)")
               .build();

       CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
       final String[] windowId = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String maintenanceWindowId = response.windowId();
               System.out.println("The maintenance window id is " +
maintenanceWindowId);
               windowId[0] = maintenanceWindowId;
           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof DocumentAlreadyExistsException) {
                   throw new CompletionException(cause);
               } else if (cause instanceof SsmException) {
                   throw new CompletionException(cause);
               } else {
                   throw new RuntimeException(cause);
               }
           }
```

```
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture
= getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
```

- For API details, see CreateMaintenanceWindow in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { CreateMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM maintenance window.
 * @param {{ name: string, allowUnassociatedTargets: boolean, duration: number,
 cutoff: number, schedule: string, description?: string }}
 */
export const main = async ({
  name,
  allowUnassociatedTargets, // Allow the maintenance window to run on managed
 nodes, even if you haven't registered those nodes as targets.
  duration, // The duration of the maintenance window in hours.
  cutoff, // The number of hours before the end of the maintenance window that
 Amazon Web Services Systems Manager stops scheduling new tasks for execution.
  schedule, // The schedule of the maintenance window in the form of a cron or
 rate expression.
  description = undefined,
}) => {
  const client = new SSMClient({});

  try {
    const { windowId } = await client.send(
      new CreateMaintenanceWindowCommand({
        Name: name,
        Description: description,
        AllowUnassociatedTargets: allowUnassociatedTargets, // Allow the
 maintenance window to run on managed nodes, even if you haven't registered those
 nodes as targets.
        Duration: duration, // The duration of the maintenance window in hours.
```

```
        Cutoff: cutoff, // The number of hours before the end of the maintenance
 window that Amazon Web Services Systems Manager stops scheduling new tasks for
 execution.
        Schedule: schedule, // The schedule of the maintenance window in the form
 of a cron or rate expression.
      }),
    );
    console.log(`Maintenance window created with Id: ${windowId}`);
    return { WindowId: windowId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
```

- For API details, see CreateMaintenanceWindow in *AWS SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example creates a new maintenance window with the specified name that runs at 4 PM on every Tuesday for 4 hours, with a 1 hour cutoff, and that allows unassociated targets.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Output:

```
mw-03eb53e1ea7383998
```

- For API details, see CreateMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example creates a new maintenance window with the specified name that runs at 4 PM on every Tuesday for 4 hours, with a 1 hour cutoff, and that allows unassociated targets.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Output:

```
mw-03eb53e1ea7383998
```

- For API details, see CreateMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
```

```
        return cls(ssm_client)


    def create(self, name, schedule, duration, cutoff,
allow_unassociated_targets):
        """
        Create an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes
as targets.
        """
        try:
            response = self.ssm_client.create_maintenance_window(
                Name=name,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.window_id = response["WindowId"]
            self.name = name
            logger.info("Created maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to create maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create maintenance window %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see CreateMaintenanceWindow in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateOpsItem` with an AWS SDK or CLI

The following code examples show how to use `CreateOpsItem`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

CLI

AWS CLI

To create an OpsItems

The following `create-ops-item` example 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"
```

Output:

```
{
    "OpsItemId": "oi-1a2b3c4d5e6f"
}
```

For more information, see Creating OpsItems in the *AWS Systems Manager User Guide*.

- For API details, see CreateOpsItem in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();
```

```
        CompletableFuture<CreateOpsItemResponse> future =
 getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }
```

- For API details, see CreateOpsItem in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { CreateOpsItemCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM OpsItem.
 * @param {{ title: string, source: string, category?: string, severity?:
 string }}
 */
export const main = async ({
  title,
  source,
  category = undefined,
  severity = undefined,
```

```
}) => {
  const client = new SSMClient({});
  try {
    const { opsItemArn, opsItemId } = await client.send(
      new CreateOpsItemCommand({
        Title: title,
        Source: source, // The origin of the OpsItem, such as Amazon EC2 or
 Systems Manager.
        Category: category,
        Severity: severity,
      }),
    );
    console.log(`Ops item created with id: ${opsItemId}`);
    return { OpsItemArn: opsItemArn, OpsItemId: opsItemId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
```

- For API details, see CreateOpsItem in *AWS SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
```

```
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, title, source, category, severity, description):
        """
        Create an OpsItem

        :param title: The OpsItem title.
        :param source: The OpsItem source.
        :param category: The OpsItem category.
        :param severity: The OpsItem severity.
        :param description: The OpsItem description.

        """
        try:
            response = self.ssm_client.create_ops_item(
                Title=title,
                Source=source,
                Category=category,
                Severity=severity,
                Description=description,
            )
            self.id = response["OpsItemId"]
        except self.ssm_client.exceptions.OpsItemLimitExceededException as err:
            logger.error(
                "Couldn't create ops item because you have exceeded your open
OpsItem limit. "
                "Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        except ClientError as err:
            logger.error(
```

```
                "Couldn't create ops item %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see CreateOpsItem in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreatePatchBaseline` with a CLI

The following code examples show how to use `CreatePatchBaseline`.

CLI

AWS CLI

Example 1: To create a patch baseline with auto-approval

The following `create-patch-baseline` example creates a patch baseline for Windows Server that approves patches for a production environment seven days after they are released by Microsoft.

```
aws ssm create-patch-baseline \
    --name "Windows-Production-Baseline-AutoApproval" \
    --operating-system "WINDOWS" \
    --approval-
rules "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,1
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]}]},App
 \
    --description "Baseline containing all updates approved for Windows Server
 production systems"
```

Output:

```
{
    "BaselineId": "pb-045f10b4f3EXAMPLE"
```

```
}
```

Example 2: To create a patch baseline with an approval cutoff date

The following `create-patch-baseline` example creates a patch baseline for Windows Server that approves all patches for a production environment that are released on or before July 7, 2020.

```
aws ssm create-patch-baseline \
    --name "Windows-Production-Baseline-AutoApproval" \
    --operating-system "WINDOWS" \
    --approval-
rules "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,1
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]}]},App
 \
    --description "Baseline containing all updates approved for Windows Server
 production systems"
```

Output:

```
{
    "BaselineId": "pb-045f10b4f3EXAMPLE"
}
```

Example 3: To create a patch baseline with approval rules stored in a JSON file

The following `create-patch-baseline` example creates a patch baseline for Amazon Linux 2017.09 that approves patches for a production environment seven days after they are released, specifies approval rules for the patch baseline, and specifies a custom repository for patches.

```
aws ssm create-patch-baseline \
    --cli-input-json file://my-amazon-linux-approval-rules-and-repo.json
```

Contents of `my-amazon-linux-approval-rules-and-repo.json`:

```
{
    "Name": "Amazon-Linux-2017.09-Production-Baseline",
    "Description": "My approval rules patch baseline for Amazon Linux 2017.09
 instances",
    "OperatingSystem": "AMAZON_LINUX",
```

```
    "Tags": [
        {
            "Key": "Environment",
            "Value": "Production"
        }
    ],
    "ApprovalRules": {
        "PatchRules": [
            {
                "ApproveAfterDays": 7,
                "EnableNonSecurity": true,
                "PatchFilterGroup": {
                    "PatchFilters": [
                        {
                            "Key": "SEVERITY",
                            "Values": [
                                "Important",
                                "Critical"
                            ]
                        },
                        {
                            "Key": "CLASSIFICATION",
                            "Values": [
                                "Security",
                                "Bugfix"
                            ]
                        },
                        {
                            "Key": "PRODUCT",
                            "Values": [
                                "AmazonLinux2017.09"
                            ]
                        }
                    ]
                }
            }
        ]
    },
    "Sources": [
        {
            "Name": "My-AL2017.09",
            "Products": [
                "AmazonLinux2017.09"
            ],
```

```
            "Configuration": "[amzn-main] \nname=amzn-main-Base
\nmirrorlist=http://repo./$awsregion./$awsdomain//$releasever/main/
mirror.list //nmirrorlist_expire=300//nmetadata_expire=300 \npriority=10
 \nfailovermethod=priority \nfastestmirror_enabled=0 \ngpgcheck=1
 \ngpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-ga \nenabled=1 \nretries=3
 \ntimeout=5\nreport_instanceid=yes"
        }
    ]
}
```

Example 4: To create a patch baseline that specifies approved and rejected patches

The following `create-patch-baseline` example explicitly specifies patches to approve and reject as exception to the default approval rules.

```
aws ssm create-patch-baseline \
    --name "Amazon-Linux-2017.09-Alpha-Baseline" \
    --description "My custom approve/reject patch baseline for Amazon Linux
 2017.09 instances" \
    --operating-system "AMAZON_LINUX" \
    --approved-patches "CVE-2018-1234567,example-pkg-EE-2018*.amzn1.noarch" \
    --approved-patches-compliance-level "HIGH" \
    --approved-patches-enable-non-security \
    --tags "Key=Environment,Value=Alpha"
```

For more information, see Create a Custom Patch Baseline in the *AWS Systems Manager User Guide*.

- For API details, see CreatePatchBaseline in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example creates a patch baseline that approves patches, seven days after they are released by Microsoft, for managed instances running Windows Server 2019 in a production environment.

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule
$rule.ApproveAfterDays = 7
```

```
$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup

$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
$rule.PatchFilterGroup = $ruleFilters

New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description
 "Baseline containing all updates approved for production systems" -
ApprovalRules_PatchRule $rule
```

Output:

```
pb-0z4z6221c4296b23z
```

- For API details, see CreatePatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example creates a patch baseline that approves patches, seven days after they are released by Microsoft, for managed instances running Windows Server 2019 in a production environment.

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule
```

```
$rule.ApproveAfterDays = 7

$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup

$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
$rule.PatchFilterGroup = $ruleFilters

New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description
 "Baseline containing all updates approved for production systems" -
ApprovalRules_PatchRule $rule
```

Output:

```
pb-0z4z6221c4296b23z
```

- For API details, see CreatePatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteActivation` with a CLI

The following code examples show how to use `DeleteActivation`.

CLI

AWS CLI

To delete a managed instance activation

The following `delete-activation` example deletes a managed instance activation.

```
aws ssm delete-activation \
    --activation-id "aa673477-d926-42c1-8757-1358cEXAMPLE"
```

This command produces no output.

For more information, see Setting Up AWS Systems Manager for Hybrid Environments in the *AWS Systems Manager User Guide*.

- For API details, see DeleteActivation in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example deletes an activation. There is no output if the command succeeds.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- For API details, see DeleteActivation in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example deletes an activation. There is no output if the command succeeds.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- For API details, see DeleteActivation in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteAssociation` with a CLI

The following code examples show how to use `DeleteAssociation`.

CLI

AWS CLI

Example 1: To delete an association using the association ID

The following `delete-association` example deletes the association for the specified association ID. There is no output if the command succeeds.

```
aws ssm delete-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

This command produces no output.

For more information, see Editing and creating a new version of an association in the *AWS Systems Manager User Guide*.

Example 2: To delete an association

The following `delete-association` example deletes the association between an instance and a document. There is no output if the command succeeds.

```
aws ssm delete-association \
    --instance-id "i-1234567890abcdef0" \
    --name "AWS-UpdateSSMAgent"
```

This command produces no output.

For more information, see Working with associations in Systems Manager in the *AWS Systems Manager User Guide*.

- For API details, see DeleteAssociation in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example deletes the association between an instance and a document. There is no output if the command succeeds.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-
UpdateSSMAgent"
```

- For API details, see DeleteAssociation in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example deletes the association between an instance and a document. There is no output if the command succeeds.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-
UpdateSSMAgent"
```

- For API details, see DeleteAssociation in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteDocument` with an AWS SDK or CLI

The following code examples show how to use `DeleteDocument`.

CLI

AWS CLI

To delete a document

The following `delete-document` example deletes a Systems Manager document.

```
aws ssm delete-document \
    --name "Example"
```

This command produces no output.

For more information, see Creating Systems Manager Documents in the *AWS Systems Manager User Guide*.

- For API details, see DeleteDocument in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
```

```
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
 cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }
```

- For API details, see DeleteDocument in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { DeleteDocumentCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Delete an SSM document.
 * @param {{ documentName: string }}
 */
export const main = async ({ documentName }) => {
  const client = new SSMClient({});
  try {
    await client.send(new DeleteDocumentCommand({ Name: documentName }));
    console.log(`Document '${documentName}' deleted.`);
```

```
    return { Deleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    } else {
      throw caught;
    }
  }
};
```

- For API details, see DeleteDocument in *AWS SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example deletes a document. There is no output if the command succeeds.

```
Remove-SSMDocument -Name "RunShellScript"
```

- For API details, see DeleteDocument in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example deletes a document. There is no output if the command succeeds.

```
Remove-SSMDocument -Name "RunShellScript"
```

- For API details, see DeleteDocument in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Deletes an AWS Systems Manager document.
        """
        if self.name is None:
            return

        try:
            self.ssm_client.delete_document(Name=self.name)
            print(f"Deleted document {self.name}.")
            self.name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see DeleteDocument in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteMaintenanceWindow` with an AWS SDK or CLI

The following code examples show how to use `DeleteMaintenanceWindow`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

CLI

AWS CLI

To delete a maintenance window

This `delete-maintenance-window` example removes the specified maintenance window.

```
aws ssm delete-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9"
```

Output:

```
{
    "WindowId":"mw-1a2b3c4d5e6f7g8h9"
}
```

For more information, see Delete a Maintenance Window (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see DeleteMaintenanceWindow in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /**
    * Deletes an AWS SSM Maintenance Window asynchronously.
    *
    * @param winId The ID of the Maintenance Window to delete.
    * <p>
    * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
    * If an exception occurs, it handles the error appropriately.
    */
   public void deleteMaintenanceWindow(String winId) {
       DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
               .windowId(winId)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().deleteMaintenanceWindow(windowRequest)
                   .thenAccept(response -> {
                       System.out.println("The maintenance window was
successfully deleted.");
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
```

```
            } else {
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }
```

- For API details, see DeleteMaintenanceWindow in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { DeleteMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Delete an SSM maintenance window.
 * @param {{ windowId: string }}
 */
export const main = async ({ windowId }) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new DeleteMaintenanceWindowCommand({ WindowId: windowId }),
    );
    console.log(`Maintenance window '${windowId}' deleted.`);
    return { Deleted: true };
```

```
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    } else {
      throw caught;
    }
  }
};
```

- For API details, see DeleteMaintenanceWindow in *AWS SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example removes a maintenance window.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Output:

```
mw-06d59c1a07c022145
```

- For API details, see DeleteMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example removes a maintenance window.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Output:

```
mw-06d59c1a07c022145
```

- For API details, see DeleteMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Delete the associated AWS Systems Manager maintenance window.
        """
        if self.window_id is None:
            return

        try:
            self.ssm_client.delete_maintenance_window(WindowId=self.window_id)
            logger.info("Deleted maintenance window %s.", self.window_id)
            print(f"Deleted maintenance window {self.name}")
            self.window_id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete maintenance window %s. Here's why: %s: %s",
```

```
                self.window_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see DeleteMaintenanceWindow in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteOpsItem` with an AWS SDK

The following code example shows how to use `DeleteOpsItem`.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
```

```
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Delete the OpsItem.
        """
        if self.id is None:
            return
        try:
            self.ssm_client.delete_ops_item(OpsItemId=self.id)
            print(f"Deleted ops item with id {self.id}")
            self.id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see DeleteOpsItem in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteParameter` with a CLI

The following code examples show how to use `DeleteParameter`.

CLI

AWS CLI

To delete a parameter

The following `delete-parameter` example deletes the specified single parameter.

```
aws ssm delete-parameter \
    --name "MyParameter"
```

This command produces no output.

For more information, see Working with Parameter Store in the *AWS Systems Manager User Guide*.

- For API details, see DeleteParameter in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example deletes a parameter. There is no output if the command succeeds.

```
Remove-SSMParameter -Name "helloWorld"
```

- For API details, see DeleteParameter in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example deletes a parameter. There is no output if the command succeeds.

```
Remove-SSMParameter -Name "helloWorld"
```

- For API details, see DeleteParameter in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeletePatchBaseline` with a CLI

The following code examples show how to use `DeletePatchBaseline`.

CLI

AWS CLI

To delete a patch baseline

The following `delete-patch-baseline` example deletes the specified patch baseline.

```
aws ssm delete-patch-baseline \
    --baseline-id "pb-045f10b4f382baeda"
```

Output:

```
{
    "BaselineId": "pb-045f10b4f382baeda"
}
```

For more information, see Update or Delete a Patch Baseline (Console) in the *AWS Systems Manager User Guide*.

- For API details, see DeletePatchBaseline in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example deletes a patch baseline.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Output:

```
pb-045f10b4f382baeda
```

- For API details, see DeletePatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example deletes a patch baseline.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Output:

```
pb-045f10b4f382baeda
```

- For API details, see DeletePatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeregisterManagedInstance` with a CLI

The following code examples show how to use `DeregisterManagedInstance`.

CLI

AWS CLI

To deregister a managed instance

The following `deregister-managed-instance` example deregisters the specified managed instance.

```
aws ssm deregister-managed-instance \
    --instance-id 'mi-08ab247cdfEXAMPLE'
```

This command produces no output.

For more information, see Deregistering managed nodes in a hybrid and multicloud environment in the *AWS Systems Manager User Guide*.

- For API details, see DeregisterManagedInstance in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example deregisters a managed instance. There is no output if the command succeeds.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- For API details, see DeregisterManagedInstance in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example deregisters a managed instance. There is no output if the command succeeds.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- For API details, see DeregisterManagedInstance in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeregisterPatchBaselineForPatchGroup` with a CLI

The following code examples show how to use `DeregisterPatchBaselineForPatchGroup`.

CLI

AWS CLI

To deregister a patch group from a patch baseline

The following `deregister-patch-baseline-for-patch-group` example deregisters the specified patch group from the specified patch baseline.

```
aws ssm deregister-patch-baseline-for-patch-group \
    --patch-group "Production" \
    --baseline-id "pb-0ca44a362fEXAMPLE"
```

Output:

```
{
  "PatchGroup":"Production",
  "BaselineId":"pb-0ca44a362fEXAMPLE"
```

```
}
```

For more information, see Add a Patch Group to a Patch Baseline in the *AWS Systems Manager User Guide*.

- For API details, see DeregisterPatchBaselineForPatchGroup in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example deregisters a patch group from a patch baseline.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Output:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- For API details, see DeregisterPatchBaselineForPatchGroup in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example deregisters a patch group from a patch baseline.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Output:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- For API details, see DeregisterPatchBaselineForPatchGroup in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeregisterTargetFromMaintenanceWindow` with a CLI

The following code examples show how to use `DeregisterTargetFromMaintenanceWindow`.

CLI

AWS CLI

To remove a target from a maintenance window

The following `deregister-target-from-maintenance-window` example removes the specified target from the specified maintenance window.

```
aws ssm deregister-target-from-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --window-target-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
```

Output:

```
{
    "WindowId":"mw-ab12cd34ef56gh78",
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

For more information, see Update a Maintenance Window (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see DeregisterTargetFromMaintenanceWindow in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example removes a target from a maintenance window.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId
 "6ab5c208-9fc4-4697-84b7-b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Output:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- For API details, see DeregisterTargetFromMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example removes a target from a maintenance window.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId
 "6ab5c208-9fc4-4697-84b7-b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Output:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- For API details, see DeregisterTargetFromMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeregisterTaskFromMaintenanceWindow` with a CLI

The following code examples show how to use `DeregisterTaskFromMaintenanceWindow`.

CLI

AWS CLI

To remove a task from a maintenance window

The following `deregister-task-from-maintenance-window` example removes the specified task from the specified maintenance window.

```
aws ssm deregister-task-from-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --window-task-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c"
```

Output:

```
{
    "WindowTaskId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c",
    "WindowId":"mw-ab12cd34ef56gh78"
}
```

For more information, see Systems Manager Maintenance Windows Tutorials (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see DeregisterTaskFromMaintenanceWindow in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example removes a task from a maintenance window.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Output:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- For API details, see DeregisterTaskFromMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example removes a task from a maintenance window.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Output:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- For API details, see DeregisterTaskFromMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeActivations` with a CLI

The following code examples show how to use `DescribeActivations`.

CLI

AWS CLI

To describe activations

The following `describe-activations` example lists details about the activations in your AWS account.

```
aws ssm describe-activations
```

Output:

```
{
    "ActivationList": [
        {
            "ActivationId": "5743558d-563b-4457-8682-d16c3EXAMPLE",
            "Description": "Example1",
            "IamRole": "HybridWebServersRole,
            "RegistrationLimit": 5,
```

```
            "RegistrationsCount": 5,
            "ExpirationDate": 1584316800.0,
            "Expired": false,
            "CreatedDate": 1581954699.792
        },
        {
            "ActivationId": "3ee0322b-f62d-40eb-b672-13ebfEXAMPLE",
            "Description": "Example2",
            "IamRole": "HybridDatabaseServersRole",
            "RegistrationLimit": 5,
            "RegistrationsCount": 5,
            "ExpirationDate": 1580515200.0,
            "Expired": true,
            "CreatedDate": 1578064132.002
        },
    ]
}
```

For more information, see Step 4: Create a Managed-Instance Activation for a Hybrid Environment in the *AWS Systems Manager User Guide*.

- For API details, see DescribeActivations in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example provides details about the activations on your account.

```
Get-SSMActivation
```

Output:

```
ActivationId        : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate         : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description         :
ExpirationDate      : 3/2/2017 12:01:51 AM
Expired             : False
IamRole             : AutomationRole
RegistrationLimit   : 10
RegistrationsCount  : 0
```

- For API details, see DescribeActivations in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example provides details about the activations on your account.

```
Get-SSMActivation
```

Output:

```
ActivationId        : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate         : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description         :
ExpirationDate      : 3/2/2017 12:01:51 AM
Expired             : False
IamRole             : AutomationRole
RegistrationLimit   : 10
RegistrationsCount  : 0
```

- For API details, see DescribeActivations in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeAssociation` with a CLI

The following code examples show how to use `DescribeAssociation`.

CLI

AWS CLI

Example 1: To get details of an association

The following `describe-association` example describes the association for the specified association ID.

```
aws ssm describe-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Output:


```
{
    "AssociationDescription": {
        "Name": "AWS-GatherSoftwareInventory",
        "AssociationVersion": "1",
        "Date": 1534864780.995,
        "LastUpdateAssociationDate": 1543235759.81,
        "Overview": {
            "Status": "Success",
            "AssociationStatusAggregatedCount": {
                "Success": 2
            }
        },
        "DocumentVersion": "$DEFAULT",
        "Parameters": {
            "applications": [
                "Enabled"
            ],
            "awsComponents": [
                "Enabled"
            ],
            "customInventory": [
                "Enabled"
            ],
            "files": [
                ""
            ],
            "instanceDetailedInformation": [
                "Enabled"
            ],
            "networkConfig": [
                "Enabled"
            ],
            "services": [
                "Enabled"
            ],
            "windowsRegistry": [
                ""
            ],
            "windowsRoles": [
                "Enabled"
            ],
            "windowsUpdates": [
```

```
                "Enabled"
            ]
        },
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "InstanceIds",
                "Values": [
                    "*"
                ]
            }
        ],
        "ScheduleExpression": "rate(24 hours)",
        "LastExecutionDate": 1550501886.0,
        "LastSuccessfulExecutionDate": 1550501886.0,
        "AssociationName": "Inventory-Association"
    }
}
```

For more information, see Editing and creating a new version of an association in the *AWS Systems Manager User Guide*.

Example 2: To get details of an association for a specific instance and document

The following `describe-association` example describes the association between an instance and a document.

```
aws ssm describe-association \
    --instance-id "i-1234567890abcdef0" \
    --name "AWS-UpdateSSMAgent"
```

Output:

```
{
    "AssociationDescription": {
        "Status": {
            "Date": 1487876122.564,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-1234567890abcdef0",
```

```
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Associated",
            "AssociationStatusAggregatedCount": {
                "Pending": 1
            }
        },
        "AssociationId": "d8617c07-2079-4c18-9847-1234567890ab",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487876122.564,
        "Date": 1487876122.564,
        "Targets": [
            {
                "Values": [
                    "i-1234567890abcdef0"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

For more information, see Editing and creating a new version of an association in the *AWS Systems Manager User Guide*.

- For API details, see DescribeAssociation in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example describes the association between an instance and a document.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Output:

```
Name                 : AWS-UpdateSSMAgent
InstanceId           : i-0000293ffd8c57862
Date                 : 2/23/2017 6:55:22 PM
Status.Name          : Pending
Status.Date          : 2/20/2015 8:31:11 AM
```

```
Status.Message        : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- For API details, see DescribeAssociation in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example describes the association between an instance and a document.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Output:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- For API details, see DescribeAssociation in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeAssociationExecutionTargets` with a CLI

The following code examples show how to use `DescribeAssociationExecutionTargets`.

CLI

AWS CLI

To get details of an association execution

The following `describe-association-execution-targets` example describes the specified association execution.

```
aws ssm describe-association-execution-targets \
```

```
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --execution-id "7abb6378-a4a5-4f10-8312-0123456789ab"
```

Output:

```
{
    "AssociationExecutionTargets": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
            "ResourceId": "i-1234567890abcdef0",
            "ResourceType": "ManagedInstance",
            "Status": "Success",
            "DetailedStatus": "Success",
            "LastExecutionDate": 1550505538.497,
            "OutputSource": {
                "OutputSourceId": "97fff367-fc5a-4299-aed8-0123456789ab",
                "OutputSourceType": "RunCommand"
            }
        }
    ]
}
```

For more information, see Viewing association histories in the *AWS Systems Manager User Guide*.

- For API details, see DescribeAssociationExecutionTargets in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example displays the resource ID and its execution status that are part of the the association execution targets

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Output:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Example 2: This command checks the particular execution of a particular automation since yesterday, where a command document is associated. It further checkes if the association execution failed, and if so, it will display the command invocation details for the execution along with the instance id

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
 00).AddDays(-1)}

foreach ($execution in $AssociationExecution) {
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
        Get-SSMCommandInvocation -CommandId
 $execution.OutputSource.OutputSourceId -Detail:$true | Select-Object -
ExpandProperty CommandPlugins
    }
}
```

Output:

```
There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0
 on i-0a1caf234f56d7dc8


Name                   : aws:runPowerShellScript
Output                 :
                         ----------ERROR-------
                         failed to run commands: exit status 1
OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
```

```
ResponseCode           : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime  : 5/29/2019 11:04:49 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Failed
StatusDetails          : Failed
```

- For API details, see DescribeAssociationExecutionTargets in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example displays the resource ID and its execution status that are part of the the association execution targets

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Output:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Example 2: This command checks the particular execution of a particular automation since yesterday, where a command document is associated. It further checkes if the association execution failed, and if so, it will display the command invocation details for the execution along with the instance id

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
 00).AddDays(-1)}
```

```
foreach ($execution in $AssociationExecution) {
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
        Get-SSMCommandInvocation -CommandId
 $execution.OutputSource.OutputSourceId -Detail:$true | Select-Object -
ExpandProperty CommandPlugins
    }
}
```

Output:

```
There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0
 on i-0a1caf234f56d7dc8


Name                   : aws:runPowerShellScript
Output                 :
                         ----------ERROR-------
                         failed to run commands: exit status 1
OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime  : 5/29/2019 11:04:49 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Failed
StatusDetails          : Failed
```

- For API details, see DescribeAssociationExecutionTargets in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeAssociationExecutions` with a CLI

The following code examples show how to use `DescribeAssociationExecutions`.

CLI

AWS CLI

Example 1: To get details of all executions for an association

The following `describe-association-executions` example describes all executions of the specified association.

```
aws ssm describe-association-executions \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Output:

```
{
    "AssociationExecutions": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "474925ef-1249-45a2-b93d-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505827.119,
            "ResourceCountByStatus": "{Success=1}"
        },
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505536.843,
            "ResourceCountByStatus": "{Success=1}"
        },
        ...
    ]
}
```

For more information, see Viewing association histories in the *AWS Systems Manager User Guide*.

Example 2: To get details of all executions for an association after a specific date and time

The following `describe-association-executions` example describes all executions of an association after the specified date and time.

```
aws ssm describe-association-executions \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --filters "Key=CreatedTime,Value=2019-02-18T16:00:00Z,Type=GREATER_THAN"
```

Output:

```
{
    "AssociationExecutions": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "474925ef-1249-45a2-b93d-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505827.119,
            "ResourceCountByStatus": "{Success=1}"
        },
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505536.843,
            "ResourceCountByStatus": "{Success=1}"
        },
        ...
    ]
}
```

For more information, see Viewing association histories in the *AWS Systems Manager User Guide*.

- For API details, see DescribeAssociationExecutions in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example returns the executions for the association ID provided

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Output:

```
AssociationId         : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status                : Success
```

- For API details, see DescribeAssociationExecutions in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example returns the executions for the association ID provided

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Output:

```
AssociationId         : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status                : Success
```

- For API details, see DescribeAssociationExecutions in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeAutomationExecutions` with a CLI

The following code examples show how to use `DescribeAutomationExecutions`.

CLI

AWS CLI

To describe an automation execution

The following `describe-automation-executions` example displays details about an Automation execution.

```
aws ssm describe-automation-executions \
    --filters Key=ExecutionId,Values=73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

Output:

```
{
    "AutomationExecutionMetadataList": [
        {
            "AutomationExecutionId": "73c8eef8-f4ee-4a05-820c-e354fEXAMPLE",
            "DocumentName": "AWS-StartEC2Instance",
            "DocumentVersion": "1",
            "AutomationExecutionStatus": "Success",
            "ExecutionStartTime": 1583737233.748,
            "ExecutionEndTime": 1583737234.719,
            "ExecutedBy": "arn:aws:sts::29884EXAMPLE:assumed-role/
mw_service_role/OrchestrationService",
            "LogFile": "",
            "Outputs": {},
            "Mode": "Auto",
            "Targets": [],
            "ResolvedTargets": {
                "ParameterValues": [],
                "Truncated": false
            },
            "AutomationType": "Local"
```

```
        }
    ]
}
```

For more information, see Running a Simple Automation Workflow in the *AWS Systems Manager User Guide*.

- For API details, see DescribeAutomationExecutions in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example describes all active and terminated Automation Executions associated with your account.

```
Get-SSMAutomationExecutionList
```

Output:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutedBy                : admin
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
LogFile                   :
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

Example 2: This example displays ExecutionID, document, execution start/end timestamp for executions with AutomationExecutionStatus other than 'Success'

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Output:

```
AutomationExecutionId                DocumentName
 AutomationExecutionStatus ExecutionStartTime   ExecutionEndTime
--------------------                 ------------
 ------------------------ ------------------   ----------------
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
 Cancelled                 4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
 Cancelled                 4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi
 Failed                    4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- For API details, see DescribeAutomationExecutions in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example describes all active and terminated Automation Executions associated with your account.

```
Get-SSMAutomationExecutionList
```

Output:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutedBy                : admin
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
LogFile                   :
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

Example 2: This example displays ExecutionID, document, execution start/end timestamp for executions with AutomationExecutionStatus other than 'Success'

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Output:

```
AutomationExecutionId                DocumentName
 AutomationExecutionStatus ExecutionStartTime   ExecutionEndTime
---------------------                ------------
 ------------------------ ------------------   ----------------
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
 Cancelled                 4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
 Cancelled                 4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi
 Failed                    4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- For API details, see DescribeAutomationExecutions in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeAutomationStepExecutions` with a CLI

The following code examples show how to use `DescribeAutomationStepExecutions`.

CLI

AWS CLI

Example 1: To describe all steps for an automation execution

The following `describe-automation-step-executions` example displays details about the steps of an Automation execution.

```
aws ssm describe-automation-step-executions \
    --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

Output:

```
{
    "StepExecutions": [
        {
            "StepName": "startInstances",
```

```
            "Action": "aws:changeInstanceState",
            "ExecutionStartTime": 1583737234.134,
            "ExecutionEndTime": 1583737234.672,
            "StepStatus": "Success",
            "Inputs": {
                "DesiredState": "\"running\"",
                "InstanceIds": "[\"i-0cb99161f6EXAMPLE\"]"
            },
            "Outputs": {
                "InstanceStates": [
                    "running"
                ]
            },
            "StepExecutionId": "95e70479-cf20-4d80-8018-7e4e2EXAMPLE",
            "OverriddenParameters": {}
        }
    ]
}
```

Example 2: To describe a specific step for an automation execution

The following `describe-automation-step-executions` example displays details about a specific step of an Automation execution.

```
aws ssm describe-automation-step-executions \
    --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE \
    --filters Key=StepExecutionId,Values=95e70479-cf20-4d80-8018-7e4e2EXAMPLE
```

For more information, see Running an Automation Workflow Step by Step (Command Line) in the *AWS Systems Manager User Guide*.

- For API details, see DescribeAutomationStepExecutions in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This exmaple displays information about all active and terminated step executions in an Automation workflow.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Output:

```
StepName                  Action                    StepStatus
--------                  ------                    ----------
LaunchInstance            aws:runInstances          Success
OSCompatibilityCheck      aws:runCommand            Success
RunPreUpdateScript        aws:runCommand            Success
UpdateEC2Config           aws:runCommand            Cancelled
UpdateSSMAgent            aws:runCommand            Pending
UpdateAWSPVDriver         aws:runCommand            Pending
UpdateAWSEnaNetworkDriver aws:runCommand            Pending
UpdateAWSNVMe             aws:runCommand            Pending
InstallWindowsUpdates     aws:runCommand            Pending
RunPostUpdateScript       aws:runCommand            Pending
RunSysprepGeneralize      aws:runCommand            Pending
StopInstance              aws:changeInstanceState   Pending
CreateImage               aws:createImage           Pending
TerminateInstance         aws:changeInstanceState   Pending
```

- For API details, see DescribeAutomationStepExecutions in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This exmaple displays information about all active and terminated step executions in an Automation workflow.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Output:

```
StepName                  Action                    StepStatus
--------                  ------                    ----------
LaunchInstance            aws:runInstances          Success
OSCompatibilityCheck      aws:runCommand            Success
RunPreUpdateScript        aws:runCommand            Success
UpdateEC2Config           aws:runCommand            Cancelled
UpdateSSMAgent            aws:runCommand            Pending
UpdateAWSPVDriver         aws:runCommand            Pending
UpdateAWSEnaNetworkDriver aws:runCommand            Pending
UpdateAWSNVMe             aws:runCommand            Pending
InstallWindowsUpdates     aws:runCommand            Pending
```

```
RunPostUpdateScript       aws:runCommand          Pending
RunSysprepGeneralize      aws:runCommand          Pending
StopInstance              aws:changeInstanceState Pending
CreateImage               aws:createImage         Pending
TerminateInstance         aws:changeInstanceState Pending
```

- For API details, see DescribeAutomationStepExecutions in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeAvailablePatches` with a CLI

The following code examples show how to use `DescribeAvailablePatches`.

CLI

AWS CLI

To get available patches

The following `describe-available-patches` example retrieves details about all available patches for Windows Server 2019 that have a MSRC severity of Critical.

```
aws ssm describe-available-patches \
    --
filters "Key=PRODUCT,Values=WindowsServer2019" "Key=MSRC_SEVERITY,Values=Critical"
```

Output:

```
{
    "Patches": [
        {
            "Id": "fe6bd8c2-3752-4c8b-ab3e-1a7ed08767ba",
            "ReleaseDate": 1544047205.0,
            "Title": "2018-11 Update for Windows Server 2019 for x64-based
 Systems (KB4470788)",
            "Description": "Install this update to resolve issues in Windows.
 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.",
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4470788",
            "Vendor": "Microsoft",
            "ProductFamily": "Windows",
            "Product": "WindowsServer2019",
            "Classification": "SecurityUpdates",
            "MsrcSeverity": "Critical",
            "KbNumber": "KB4470788",
            "MsrcNumber": "",
            "Language": "All"
        },
        {
            "Id": "c96115e1-5587-4115-b851-22baa46a3f11",
            "ReleaseDate": 1549994410.0,
            "Title": "2019-02 Security Update for Adobe Flash Player for Windows
 Server 2019 for x64-based Systems (KB4487038)",
            "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.",
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4487038",
            "Vendor": "Microsoft",
            "ProductFamily": "Windows",
            "Product": "WindowsServer2019",
            "Classification": "SecurityUpdates",
            "MsrcSeverity": "Critical",
            "KbNumber": "KB4487038",
            "MsrcNumber": "",
            "Language": "All"
        },
        ...
    ]
}
```

To get details of a specific patch

The following `describe-available-patches` example retrieves details about the specified patch.

```
aws ssm describe-available-patches \
    --filters "Key=PATCH_ID,Values=KB4480979"
```

Output:

```
{
    "Patches": [
        {
            "Id": "680861e3-fb75-432e-818e-d72e5f2be719",
            "ReleaseDate": 1546970408.0,
            "Title": "2019-01 Security Update for Adobe Flash Player for Windows
 Server 2016 for x64-based Systems (KB4480979)",
            "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.",
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4480979",
            "Vendor": "Microsoft",
            "ProductFamily": "Windows",
            "Product": "WindowsServer2016",
            "Classification": "SecurityUpdates",
            "MsrcSeverity": "Critical",
            "KbNumber": "KB4480979",
            "MsrcNumber": "",
            "Language": "All"
        }
    ]
}
```

For more information, see How Patch Manager Operations Work in the *AWS Systems Manager User Guide*.

- For API details, see DescribeAvailablePatches in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example gets all available patches for Windows Server 2012 that have a MSRC severity of Critical. The syntax used by this example requires PowerShell version 3 or later.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}
```

```
Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Output:

```
Classification : SecurityUpdates
ContentUrl     : https://support.microsoft.com/en-us/kb/2727528
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.
Id             : 1eb507be-2040-4eeb-803d-abc55700b715
KbNumber       : KB2727528
Language       : All
MsrcNumber     : MS12-072
MsrcSeverity   : Critical
Product        : WindowsServer2012
ProductFamily  : Windows
ReleaseDate    : 11/13/2012 6:00:00 PM
Title          : Security Update for Windows Server 2012 (KB2727528)
Vendor         : Microsoft
...
```

Example 2: With PowerShell version 2, you must use New-Object to create each filter.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Example 3: This example fetches all the updates which are released in last 20 days and applicable to products matching WindowsServer2019

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20)
 | Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate,
 Product, Title
```

Output:

```
ReleaseDate         Product           Title
-----------         -------           -----
4/9/2019 5:00:12 PM WindowsServer2019 2019-04 Security Update for Adobe Flash
 Player for Windows Server 2019 for x64-based Systems (KB4493478)
4/9/2019 5:00:06 PM WindowsServer2019 2019-04 Cumulative Update for Windows
 Server 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM WindowsServer2019 2019-03 Servicing Stack Update for Windows
 Server 2019 for x64-based Systems (KB4493510)
```

- For API details, see DescribeAvailablePatches in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example gets all available patches for Windows Server 2012 that have a MSRC severity of Critical. The syntax used by this example requires PowerShell version 3 or later.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Output:

```
Classification : SecurityUpdates
ContentUrl     : https://support.microsoft.com/en-us/kb/2727528
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.
Id             : 1eb507be-2040-4eeb-803d-abc55700b715
KbNumber       : KB2727528
Language       : All
MsrcNumber     : MS12-072
```

```
MsrcSeverity   : Critical
Product        : WindowsServer2012
ProductFamily  : Windows
ReleaseDate    : 11/13/2012 6:00:00 PM
Title          : Security Update for Windows Server 2012 (KB2727528)
Vendor         : Microsoft
...
```

Example 2: With PowerShell version 2, you must use New-Object to create each filter.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Example 3: This example fetches all the updates which are released in last 20 days and applicable to products matching WindowsServer2019

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20)
 | Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate,
 Product, Title
```

Output:

```
ReleaseDate         Product           Title
-----------         -------           -----
4/9/2019 5:00:12 PM WindowsServer2019 2019-04 Security Update for Adobe Flash
 Player for Windows Server 2019 for x64-based Systems (KB4493478)
4/9/2019 5:00:06 PM WindowsServer2019 2019-04 Cumulative Update for Windows
 Server 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM WindowsServer2019 2019-03 Servicing Stack Update for Windows
 Server 2019 for x64-based Systems (KB4493510)
```

- For API details, see DescribeAvailablePatches in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeDocument` with a CLI

The following code examples show how to use `DescribeDocument`.

CLI

AWS CLI

To display details of a document

The following `describe-document` example displays details about a Systems Manager document in your AWS account.

```
aws ssm describe-document \
    --name "Example"
```

Output:

```
{
    "Document": {
        "Hash":
 "fc2410281f40779e694a8b95975d0f9f316da8a153daa94e3d9921102EXAMPLE",
        "HashType": "Sha256",
        "Name": "Example",
        "Owner": "29884EXAMPLE",
        "CreatedDate": 1583257938.266,
        "Status": "Active",
        "DocumentVersion": "1",
        "Description": "Document Example",
        "Parameters": [
            {
                "Name": "AutomationAssumeRole",
                "Type": "String",
                "Description": "(Required) The ARN of the role that allows
 Automation to perform the actions on your behalf. If no role is specified,
 Systems Manager Automation uses your IAM permissions to execute this document.",
                "DefaultValue": ""
            },
```

```
            {
                "Name": "InstanceId",
                "Type": "String",
                "Description": "(Required) The ID of the Amazon EC2 instance.",
                "DefaultValue": ""
            }
        ],
        "PlatformTypes": [
            "Windows",
            "Linux"
        ],
        "DocumentType": "Automation",
        "SchemaVersion": "0.3",
        "LatestVersion": "1",
        "DefaultVersion": "1",
        "DocumentFormat": "YAML",
        "Tags": []
    }
}
```

For more information, see Creating Systems Manager Documents in the *AWS Systems Manager User Guide*.

- For API details, see DescribeDocument in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example returns information about a document.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Output:

```
CreatedDate     : 2/24/2017 5:25:13 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            :
 f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
```

```
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Active
```

- For API details, see DescribeDocument in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example returns information about a document.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Output:

```
CreatedDate     : 2/24/2017 5:25:13 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            :
 f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Active
```

- For API details, see DescribeDocument in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeDocumentPermission` with a CLI

The following code examples show how to use `DescribeDocumentPermission`.

CLI

AWS CLI

To describe document permissions

The following `describe-document-permission` example displays permission details about a Systems Manager document that is shared publicly.

```
aws ssm describe-document-permission \
    --name "Example" \
    --permission-type "Share"
```

Output:

```
{
    "AccountIds": [
        "all"
    ],
    "AccountSharingInfoList": [
        {
            "AccountId": "all",
            "SharedDocumentVersion": "$DEFAULT"
        }
    ]
}
```

For more information, see Share a Systems Manager Document in the *AWS Systems Manager User Guide*.

- For API details, see DescribeDocumentPermission in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all the versions for a document.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Output:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- For API details, see DescribeDocumentPermission in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all the versions for a document.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Output:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- For API details, see DescribeDocumentPermission in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeEffectiveInstanceAssociations` with a CLI

The following code examples show how to use `DescribeEffectiveInstanceAssociations`.

CLI

AWS CLI

To get details of the effective associations for an instance

The following `describe-effective-instance-associations` example retrieves details about the effective associations for an instance.

Command:

```
aws ssm describe-effective-instance-associations --instance-
id "i-1234567890abcdef0"
```

Output:

```
{
    "Associations": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "InstanceId": "i-1234567890abcdef0",
            "Content": "{\n    \"schemaVersion\": \"1.2\",\n    \"description\":
 \"Update the Amazon SSM Agent to the latest version or specified version.\",\n
  \"parameters\": {\n        \"version\": {\n            \"default\": \"\",\n
        \"description\": \"(Optional) A specific version of the Amazon SSM Agent
 to install. If not specified, the agent will be updated to the latest version.
\",\n            \"type\": \"String\"\n        },\n        \"allowDowngrade
\": {\n            \"default\": \"false\",\n            \"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.\",\n            \"type
\": \"String\",\n            \"allowedValues\": [\n                \"true\",\n
              \"false\"\n            ]\n        }\n    },\n    \"runtimeConfig
\": {\n        \"aws:updateSsmAgent\": {\n            \"properties\": [\n
         {\n                \"agentName\": \"amazon-ssm-agent\",\n
  \"source\": \"https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/ssm-agent-
manifest.json\",\n                \"allowDowngrade\": \"{{ allowDowngrade }}\",\n
                \"targetVersion\": \"{{ version }}\"\n                }\n
    ]\n        }\n    }\n}\n",
            "AssociationVersion": "1"
        }
    ]
}
```

- For API details, see DescribeEffectiveInstanceAssociations in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example describes the effective associations for an instance.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5
```

Output:

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Example 2: This example displays the contents of the effective associations for an instance.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Output:

```
{
    "schemaVersion": "1.2",
    "description": "Update the Amazon SSM Agent to the latest version or
 specified version.",
    "parameters": {
        "version": {
            "default": "",
            "description": "(Optional) A specific version of the Amazon SSM Agent
 to install. If not specified, the agen
t will be updated to the latest version.",
            "type": "String"
        },
        "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"
            ]
        }
    },
    "runtimeConfig": {
        "aws:updateSsmAgent": {
            "properties": [
                {
                "agentName": "amazon-ssm-agent",
                "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
                "allowDowngrade": "{{ allowDowngrade }}",
                "targetVersion": "{{ version }}"
                }
            ]
        }
    }
}
```

- For API details, see DescribeEffectiveInstanceAssociations in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example describes the effective associations for an instance.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5
```

Output:

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Example 2: This example displays the contents of the effective associations for an instance.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Output:

```
{
    "schemaVersion": "1.2",
    "description": "Update the Amazon SSM Agent to the latest version or
 specified version.",
    "parameters": {
        "version": {
            "default": "",
            "description": "(Optional) A specific version of the Amazon SSM Agent
 to install. If not specified, the agen
t will be updated to the latest version.",
            "type": "String"
        },
        "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"
            ]
        }
    },
    "runtimeConfig": {
        "aws:updateSsmAgent": {
            "properties": [
                {
                "agentName": "amazon-ssm-agent",
                "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
                "allowDowngrade": "{{ allowDowngrade }}",
                "targetVersion": "{{ version }}"
                }
            ]
        }
    }
}
```

- For API details, see DescribeEffectiveInstanceAssociations in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeEffectivePatchesForPatchBaseline` with a CLI

The following code examples show how to use `DescribeEffectivePatchesForPatchBaseline`.

CLI

AWS CLI

Example 1: To get all patches defined by a custom patch baseline

The following `describe-effective-patches-for-patch-baseline` example returns the patches defined by a custom patch baseline in the current AWS account. Note that for a custom baseline, only the ID is required for `--baseline-id`.

```
aws ssm describe-effective-patches-for-patch-baseline \
    --baseline-id "pb-08b654cf9b9681f04"
```

Output:

```
{
    "EffectivePatches": [
        {
            "Patch": {
                "Id": "fe6bd8c2-3752-4c8b-ab3e-1a7ed08767ba",
                "ReleaseDate": 1544047205.0,
                "Title": "2018-11 Update for Windows Server 2019 for x64-based
 Systems (KB4470788)",
                "Description": "Install this update to resolve issues in Windows.
 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.",
                "ContentUrl": "https://support.microsoft.com/en-us/kb/4470788",
                "Vendor": "Microsoft",
```

```
                "ProductFamily": "Windows",
                "Product": "WindowsServer2019",
                "Classification": "SecurityUpdates",
                "MsrcSeverity": "Critical",
                "KbNumber": "KB4470788",
                "MsrcNumber": "",
                "Language": "All"
            },
            "PatchStatus": {
                "DeploymentStatus": "APPROVED",
                "ComplianceLevel": "CRITICAL",
                "ApprovalDate": 1544047205.0
            }
        },
        {
            "Patch": {
                "Id": "915a6b1a-f556-4d83-8f50-b2e75a9a7e58",
                "ReleaseDate": 1549994400.0,
                "Title": "2019-02 Cumulative Update for .NET Framework 3.5 and
4.7.2 for Windows Server 2019 for x64 (KB4483452)",
                "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.",
                "ContentUrl": "https://support.microsoft.com/en-us/kb/4483452",
                "Vendor": "Microsoft",
                "ProductFamily": "Windows",
                "Product": "WindowsServer2019",
                "Classification": "SecurityUpdates",
                "MsrcSeverity": "Important",
                "KbNumber": "KB4483452",
                "MsrcNumber": "",
                "Language": "All"
            },
            "PatchStatus": {
                "DeploymentStatus": "APPROVED",
                "ComplianceLevel": "CRITICAL",
                "ApprovalDate": 1549994400.0
            }
        },
        ...
    ],
```

```
    "NextToken": "--token string truncated--"
}
```

Example 2: To get all patches defined by an AWS managed patch baseline

The following `describe-effective-patches-for-patch-baseline` example returns the patches defined by an AWS managed patch baseline. Note that for an AWS managed baseline, the complete baseline ARN is required for `--baseline-id`

```
aws ssm describe-effective-patches-for-patch-baseline \
    --baseline-id "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-020d361a05defe4ed"
```

See example 1 for sample output.

For more information, see How Security Patches Are Selected in the *AWS Systems Manager User Guide*.

- For API details, see DescribeEffectivePatchesForPatchBaseline in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all patch baselines, with a maximum result list of 1.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Output:

```
Patch                                       PatchStatus
-----                                       -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Example 2: This example displays the patch status for all patch baselines, with a maximum result list of 1.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Output:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- For API details, see DescribeEffectivePatchesForPatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all patch baselines, with a maximum result list of 1.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Output:

```
Patch                                       PatchStatus
-----                                       -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Example 2: This example displays the patch status for all patch baselines, with a maximum result list of 1.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Output:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- For API details, see DescribeEffectivePatchesForPatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeInstanceAssociationsStatus` with a CLI

The following code examples show how to use `DescribeInstanceAssociationsStatus`.

CLI

AWS CLI

To describe the status of an instance's associations

This example shows details of the associations for an instance.

Command:

```
aws ssm describe-instance-associations-status --instance-id "i-1234567890abcdef0"
```

Output:

```
{
  "InstanceAssociationStatusInfos": [
      {
          "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
          "Name": "AWS-GatherSoftwareInventory",
          "DocumentVersion": "1",
          "AssociationVersion": "1",
          "InstanceId": "i-1234567890abcdef0",
          "ExecutionDate": 1550501886.0,
          "Status": "Success",
          "ExecutionSummary": "1 out of 1 plugin processed, 1 success, 0 failed,
 0 timedout, 0 skipped. ",
          "AssociationName": "Inventory-Association"
      },
      {
          "AssociationId": "5c5a31f6-6dae-46f9-944c-0123456789ab",
          "Name": "AWS-UpdateSSMAgent",
          "DocumentVersion": "1",
          "AssociationVersion": "1",
          "InstanceId": "i-1234567890abcdef0",
          "ExecutionDate": 1550505828.548,
```

```
            "Status": "Success",
            "DetailedStatus": "Success",
            "AssociationName": "UpdateSSMAgent"
        }
    ]
}
```

- For API details, see DescribeInstanceAssociationsStatus in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example shows details of the associations for an instance.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Output:

```
AssociationId    : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus   : Pending
DocumentVersion  : 1
ErrorCode        :
ExecutionDate    : 2/20/2015 8:31:11 AM
ExecutionSummary : temp_status_change
InstanceId       : i-0000293ffd8c57862
Name             : AWS-UpdateSSMAgent
OutputUrl        :
Status           : Pending
```

Example 2: This example checks the instance association status for the given instance id and further, displays the execution status of those associations

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-
Object {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Output:

```
AssociationId         : 512a34a5-c678-1234-1234-12345678db9e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
```

```
DetailedStatus        :
ExecutionId           : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=9}
Status                : Success
```

- For API details, see DescribeInstanceAssociationsStatus in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example shows details of the associations for an instance.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Output:

```
AssociationId    : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus   : Pending
DocumentVersion  : 1
ErrorCode        :
ExecutionDate    : 2/20/2015 8:31:11 AM
ExecutionSummary : temp_status_change
InstanceId       : i-0000293ffd8c57862
Name             : AWS-UpdateSSMAgent
OutputUrl        :
Status           : Pending
```

Example 2: This example checks the instance association status for the given instance id and further, displays the execution status of those associations

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-
Object {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Output:

```
AssociationId         : 512a34a5-c678-1234-1234-12345678db9e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate     : 1/1/0001 12:00:00 AM
```

```
ResourceCountByStatus : {Success=9}
Status                : Success
```

- For API details, see DescribeInstanceAssociationsStatus in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeInstanceInformation` with a CLI

The following code examples show how to use `DescribeInstanceInformation`.

CLI

AWS CLI

Example 1: To describe managed instance information

The following `describe-instance-information` example retrieves details of each of your managed instances.

```
aws ssm describe-instance-information
```

Example 2: To describe information about a specific managed instance

The following `describe-instance-information` example shows details of the managed instance `i-028ea792daEXAMPLE`.

```
aws ssm describe-instance-information \
    --filters "Key=InstanceIds,Values=i-028ea792daEXAMPLE"
```

Example 3: To describe information about managed instances with a specific tag key

The following `describe-instance-information` example shows details for managed instances that have the tag key `DEV`.

```
aws ssm describe-instance-information \
    --filters "Key=tag-key,Values=DEV"
```

Output:

```
{
    "InstanceInformationList": [
        {
            "InstanceId": "i-028ea792daEXAMPLE",
            "PingStatus": "Online",
            "LastPingDateTime": 1582221233.421,
            "AgentVersion": "2.3.842.0",
            "IsLatestVersion": true,
            "PlatformType": "Linux",
            "PlatformName": "SLES",
            "PlatformVersion": "15.1",
            "ResourceType": "EC2Instance",
            "IPAddress": "192.0.2.0",
            "ComputerName": "ip-198.51.100.0.us-east-2.compute.internal",
            "AssociationStatus": "Success",
            "LastAssociationExecutionDate": 1582220806.0,
            "LastSuccessfulAssociationExecutionDate": 1582220806.0,
            "AssociationOverview": {
                "DetailedStatus": "Success",
                "InstanceAssociationStatusAggregatedCount": {
                    "Success": 2
                }
            }
        }
    ]
}
```

For more information, see Managed Instances in the *AWS Systems Manager User Guide*.

- For API details, see DescribeInstanceInformation in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example shows details of each of your instances.

```
Get-SSMInstanceInformation
```

Output:

```
ActivationId                          :
AgentVersion                          : 2.0.672.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : ip-172-31-44-222.us-
west-2.compute.internal
IamRole                               :
InstanceId                            : i-0cb2b964d3e14fd9f
IPAddress                             : 172.31.44.222
IsLatestVersion                       : True
LastAssociationExecutionDate          : 2/24/2017 3:18:09 AM
LastPingDateTime                      : 2/24/2017 3:35:03 AM
LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
Name                                  :
PingStatus                            : ConnectionLost
PlatformName                          : Amazon Linux AMI
PlatformType                          : Linux
PlatformVersion                       : 2016.09
RegistrationDate                      : 1/1/0001 12:00:00 AM
ResourceType                          : EC2Instance
```

Example 2: This example shows how to use the -Filter parameter to filter results to only those AWS Systems Manager instances in region `us-east-1` with an `AgentVersion` of `2.2.800.0`. You can find a list of valid -Filter key values in the InstanceInformation API reference topic (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager-Type-InstanceInformation-ActivationId).

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

Output:

```
ActivationId                          :
AgentVersion                          : 2.2.800.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
```

```
AssociationStatus                     : Success
ComputerName                          : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                               :
InstanceId                            : i-EXAMPLEb0792d98ce
IPAddress                             : 10.0.0.01
IsLatestVersion                       : False
LastAssociationExecutionDate          : 8/16/2018 12:02:50 AM
LastPingDateTime                      : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                  :
PingStatus                            : Online
PlatformName                          : Microsoft Windows Server 2016 Datacenter
PlatformType                          : Windows
PlatformVersion                       : 10.0.14393
RegistrationDate                      : 1/1/0001 12:00:00 AM
ResourceType                          : EC2Instance

ActivationId                          :
AgentVersion                          : 2.2.800.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                               :
InstanceId                            : i-EXAMPLEac7501d023
IPAddress                             : 10.0.0.02
IsLatestVersion                       : False
LastAssociationExecutionDate          : 8/16/2018 12:00:20 AM
LastPingDateTime                      : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                  :
PingStatus                            : Online
PlatformName                          : Microsoft Windows Server 2016 Datacenter
PlatformType                          : Windows
PlatformVersion                       : 10.0.14393
RegistrationDate                      : 1/1/0001 12:00:00 AM
ResourceType                          : EC2Instance
```

Example 3: This example shows how to use the -InstanceInformationFilterList parameter to filter results to only those AWS Systems Manager instances in region `us-east-1` with `PlatformTypes` of `Windows` or `Linux`. You can find a list of valid -InstanceInformationFilterList key values in the InstanceInformationFilter API

reference topic (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html).

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList
 $Filters
```

Output:

```
ActivationId                          :
AgentVersion                          : 2.2.800.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                               :
InstanceId                            : i-EXAMPLEb0792d98ce
IPAddress                             : 10.0.0.27
IsLatestVersion                       : False
LastAssociationExecutionDate          : 8/16/2018 12:02:50 AM
LastPingDateTime                      : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                  :
PingStatus                            : Online
PlatformName                          : Ubuntu Server 18.04 LTS
PlatformType                          : Linux
PlatformVersion                       : 18.04
RegistrationDate                      : 1/1/0001 12:00:00 AM
ResourceType                          : EC2Instance

ActivationId                          :
AgentVersion                          : 2.2.800.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                               :
InstanceId                            : i-EXAMPLEac7501d023
IPAddress                             : 10.0.0.100
IsLatestVersion                       : False
```

```
LastAssociationExecutionDate           : 8/16/2018 12:00:20 AM
LastPingDateTime                       : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance
```

Example 4: This example lists ssm managed instances and exports InstanceId, PingStatus, LastPingDateTime and PlatformName to a csv file.

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus,
 LastPingDateTime, PlatformName | Export-Csv Instance-details.csv -
NoTypeInformation
```

- For API details, see DescribeInstanceInformation in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example shows details of each of your instances.

```
Get-SSMInstanceInformation
```

Output:

```
ActivationId                           :
AgentVersion                           : 2.0.672.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : ip-172-31-44-222.us-
west-2.compute.internal
IamRole                                :
InstanceId                             : i-0cb2b964d3e14fd9f
IPAddress                              : 172.31.44.222
IsLatestVersion                        : True
LastAssociationExecutionDate           : 2/24/2017 3:18:09 AM
LastPingDateTime                       : 2/24/2017 3:35:03 AM
LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
```

```
Name                                    :
PingStatus                              : ConnectionLost
PlatformName                            : Amazon Linux AMI
PlatformType                            : Linux
PlatformVersion                         : 2016.09
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Example 2: This example shows how to use the -Filter parameter to filter results to only those AWS Systems Manager instances in region `us-east-1` with an `AgentVersion` of `2.2.800.0`. You can find a list of valid -Filter key values in the InstanceInformation API reference topic (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager-Type-InstanceInformation-ActivationId).

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

Output:

```
ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEb0792d98ce
IPAddress                               : 10.0.0.01
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:02:50 AM
LastPingDateTime                        : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
```

```
ResourceType                            : EC2Instance

ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEac7501d023
IPAddress                               : 10.0.0.02
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:00:20 AM
LastPingDateTime                        : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Example 3: This example shows how to use the -InstanceInformationFilterList parameter to filter results to only those AWS Systems Manager instances in region `us-east-1` with `PlatformTypes` of `Windows` or `Linux`. You can find a list of valid -InstanceInformationFilterList key values in the InstanceInformationFilter API reference topic (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html).

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList
 $Filters
```

Output:

```
ActivationId                            :
AgentVersion                            : 2.2.800.0
```

```
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEb0792d98ce
IPAddress                               : 10.0.0.27
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:02:50 AM
LastPingDateTime                        : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Ubuntu Server 18.04 LTS
PlatformType                            : Linux
PlatformVersion                         : 18.04
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance

ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEac7501d023
IPAddress                               : 10.0.0.100
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:00:20 AM
LastPingDateTime                        : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Example 4: This example lists ssm managed instances and exports InstanceId, PingStatus, LastPingDateTime and PlatformName to a csv file.

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus,
 LastPingDateTime, PlatformName | Export-Csv Instance-details.csv -
NoTypeInformation
```

- For API details, see DescribeInstanceInformation in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeInstancePatchStates` with a CLI

The following code examples show how to use `DescribeInstancePatchStates`.

CLI

AWS CLI

To get the patch summary states for instances

This `describe-instance-patch-states` example gets the patch summary states for an instance.

```
aws ssm describe-instance-patch-states \
    --instance-ids "i-1234567890abcdef0"
```

Output:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "PatchGroup": "my-patch-group",
            "BaselineId": "pb-0713accee01234567",
            "SnapshotId": "521c3536-930c-4aa9-950e-01234567abcd",
            "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"
        }
    ]
}
```

For more information, see About Patch Compliance in the *AWS Systems Manager User Guide*.

- For API details, see DescribeInstancePatchStates in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example gets the patch summary states for an instance.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Example 2: This example gets the patch summary states for two instances.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- For API details, see DescribeInstancePatchStates in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example gets the patch summary states for an instance.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Example 2: This example gets the patch summary states for two instances.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- For API details, see DescribeInstancePatchStates in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeInstancePatchStatesForPatchGroup` with a CLI

The following code examples show how to use `DescribeInstancePatchStatesForPatchGroup`.

CLI

AWS CLI

Example 1: To get the instance states for a patch group

The following `describe-instance-patch-states-for-patch-group` example retrieves details about the patch summary states per-instance for the specified patch group.

```
aws ssm describe-instance-patch-states-for-patch-group \
    --patch-group "Production"
```

Output:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-02573cafcfEXAMPLE",
            "PatchGroup": "Production",
            "BaselineId": "pb-0c10e65780EXAMPLE",
            "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
            "OwnerInformation": "",
            "InstalledCount": 32,
            "InstalledOtherCount": 1,
            "InstalledPendingRebootCount": 0,
            "InstalledRejectedCount": 0,
```

```
            "MissingCount": 2,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 2671,
            "NotApplicableCount": 400,
            "OperationStartTime": "2021-08-04T11:03:50.590000-07:00",
            "OperationEndTime": "2021-08-04T11:04:21.555000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 0
        },
        {
            "InstanceId": "i-0471e04240EXAMPLE",
            "PatchGroup": "Production",
            "BaselineId": "pb-09ca3fb51fEXAMPLE",
            "SnapshotId": "05d8ffb0-1bbe-4812-ba2d-d9b7bEXAMPLE",
            "OwnerInformation": "",
            "InstalledCount": 32,
            "InstalledOtherCount": 1,
            "InstalledPendingRebootCount": 0,
            "InstalledRejectedCount": 0,
            "MissingCount": 2,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 2671,
            "NotApplicableCount": 400,
            "OperationStartTime": "2021-08-04T22:06:20.340000-07:00",
            "OperationEndTime": "2021-08-04T22:07:11.220000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 0
        }
    ]
}
```


Example 2: To get the instance states for a patch group with more than five missing patches

The following `describe-instance-patch-states-for-patch-group` example retrieves details about the patch summary states for the specified patch group for instances with more than five missing patches.

```
aws ssm describe-instance-patch-states-for-patch-group \
    --filters Key=MissingCount,Type=GreaterThan,Values=5 \
    --patch-group "Production"
```

Output:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-02573cafcfEXAMPLE",
            "PatchGroup": "Production",
            "BaselineId": "pb-0c10e65780EXAMPLE",
            "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
            "OwnerInformation": "",
            "InstalledCount": 46,
            "InstalledOtherCount": 4,
            "InstalledPendingRebootCount": 1,
            "InstalledRejectedCount": 1,
            "MissingCount": 7,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 232,
            "NotApplicableCount": 654,
            "OperationStartTime": "2021-08-04T11:03:50.590000-07:00",
            "OperationEndTime": "2021-08-04T11:04:21.555000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 1
        }
    ]
}
```

Example 3: To get the instance states for a patch group with fewer than ten instances that require a reboot

The following `describe-instance-patch-states-for-patch-group` example retrieves details about the patch summary states for the specified patch group for instances with fewer than ten instances requiring a reboot.

```
aws ssm describe-instance-patch-states-for-patch-group \
    --filters Key=InstalledPendingRebootCount,Type=LessThan,Values=10 \
```

```
    --patch-group "Production"
```

Output:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-02573cafcfEXAMPLE",
            "BaselineId": "pb-0c10e65780EXAMPLE",
            "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
            "PatchGroup": "Production",
            "OwnerInformation": "",
            "InstalledCount": 32,
            "InstalledOtherCount": 1,
            "InstalledPendingRebootCount": 4,
            "InstalledRejectedCount": 0,
            "MissingCount": 2,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 846,
            "NotApplicableCount": 212,
            "OperationStartTime": "2021-08-046T11:03:50.590000-07:00",
            "OperationEndTime": "2021-08-06T11:04:21.555000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 0
        }
    ]
}
```

For more information, see Understanding patch compliance state values in the *AWS Systems Manager User Guide*.

- For API details, see DescribeInstancePatchStatesForPatchGroup in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example gets the patch summary states per-instance for a patch group.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- For API details, see DescribeInstancePatchStatesForPatchGroup in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example gets the patch summary states per-instance for a patch group.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- For API details, see DescribeInstancePatchStatesForPatchGroup in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeInstancePatches` with a CLI

The following code examples show how to use `DescribeInstancePatches`.

CLI

AWS CLI

Example 1: To get the patch state details for an instance

The following `describe-instance-patches` example retrieves details about the patches for the specified instance.

```
aws ssm describe-instance-patches \
    --instance-id "i-1234567890abcdef0"
```

Output:

```
{
    "Patches": [
        {
```

```
            "Title": "2019-01 Security Update for Adobe Flash Player for Windows
 Server 2016 for x64-based Systems (KB4480979)",
            "KBId": "KB4480979",
            "Classification": "SecurityUpdates",
            "Severity": "Critical",
            "State": "Installed",
            "InstalledTime": "2019-01-09T00:00:00+00:00"
        },
        {
            "Title": "",
            "KBId": "KB4481031",
            "Classification": "",
            "Severity": "",
            "State": "InstalledOther",
            "InstalledTime": "2019-02-08T00:00:00+00:00"
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

Example 2: To get a list of patches in the Missing state for an instance

The following `describe-instance-patches` example retrieves information about patches that are in the Missing state for the specified instance.

```
aws ssm describe-instance-patches \
    --instance-id "i-1234567890abcdef0" \
    --filters Key=State,Values=Missing
```

Output:

```
{
    "Patches": [
        {
            "Title": "Windows Malicious Software Removal Tool x64 - February 2019
 (KB890830)",
            "KBId": "KB890830",
            "Classification": "UpdateRollups",
            "Severity": "Unspecified",
            "State": "Missing",
            "InstalledTime": "1970-01-01T00:00:00+00:00"
```

```
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

For more information, see About Patch Compliance States in the *AWS Systems Manager User Guide*.

Example 3: To get a list of patches installed since a specified InstalledTime for an instance

The following `describe-instance-patches` example retrieves information about patches installed since a specified time for the specified instance by combining the use of `--filters` and `--query`.

```
aws ssm describe-instance-patches \
    --instance-id "i-1234567890abcdef0" \
    --filters Key=State,Values=Installed \
    --query "Patches[?InstalledTime >= `2023-01-01T16:00:00`]"
```

Output:

```
{
    "Patches": [
        {
            "Title": "2023-03 Cumulative Update for Windows Server 2019 (1809)
 for x64-based Systems (KB5023702)",
            "KBId": "KB5023702",
            "Classification": "SecurityUpdates",
            "Severity": "Critical",
            "State": "Installed",
            "InstalledTime": "2023-03-16T11:00:00+00:00"
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

- For API details, see DescribeInstancePatches in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example gets the patch compliance details for an instance.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- For API details, see DescribeInstancePatches in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example gets the patch compliance details for an instance.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- For API details, see DescribeInstancePatches in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeMaintenanceWindowExecutionTaskInvocations` with a CLI

The following code examples show how to use `DescribeMaintenanceWindowExecutionTaskInvocations`.

CLI

AWS CLI

To get the specific task invocations performed for a maintenance window task execution

The following `describe-maintenance-window-execution-task-invocations` example lists the invocations for the specified task executed as part of the specified maintenance window execution.

```
aws ssm describe-maintenance-window-execution-task-invocations \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2a638355" \
```

```
    --task-id "ac0c6ae1-daa3-4a89-832e-d384503b6586"
```

Output:

```
{
    "WindowExecutionTaskInvocationIdentities": [
        {
            "Status": "SUCCESS",
            "Parameters": "{\"documentName\":\"AWS-RunShellScript\",
\"instanceIds\":[\"i-0000293ffd8c57862\"],\"parameters\":{\"commands\":[\"df\"]},
\"maxConcurrency\":\"1\",\"maxErrors\":\"1\"}",
            "InvocationId": "e274b6e1-fe56-4e32-bd2a-8073c6381d8b",
            "StartTime": 1487692834.723,
            "EndTime": 1487692834.871,
            "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2a638355",
            "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d384503b6586"
        }
    ]
}
```

For more information, see View Information About Tasks and Task Executions (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see DescribeMaintenanceWindowExecutionTaskInvocations in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists the invocations for a task executed as part of a maintenance window execution.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Output:

```
EndTime           : 2/21/2017 4:00:34 PM
ExecutionId       :
```

```
InvocationId      : e274b6e1-fe56-4e32-bd2a-8073c6381d8b
OwnerInformation  :
Parameters        : {"documentName":"AWS-RunShellScript","instanceIds":
["i-0000293ffd8c57862"],"parameters":{"commands":["df"]},"maxConcurrency":"1",
                    "maxErrors":"1"}
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The instance IDs list contains an invalid entry.
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
WindowTargetId    :
```

- For API details, see DescribeMaintenanceWindowExecutionTaskInvocations in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists the invocations for a task executed as part of a maintenance window execution.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Output:

```
EndTime           : 2/21/2017 4:00:34 PM
ExecutionId       :
InvocationId      : e274b6e1-fe56-4e32-bd2a-8073c6381d8b
OwnerInformation  :
Parameters        : {"documentName":"AWS-RunShellScript","instanceIds":
["i-0000293ffd8c57862"],"parameters":{"commands":["df"]},"maxConcurrency":"1",
                    "maxErrors":"1"}
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The instance IDs list contains an invalid entry.
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
WindowTargetId    :
```

- For API details, see DescribeMaintenanceWindowExecutionTaskInvocations in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeMaintenanceWindowExecutionTasks` with a CLI

The following code examples show how to use `DescribeMaintenanceWindowExecutionTasks`.

CLI

AWS CLI

To list all tasks associated with a maintenance window execution

The following `ssm describe-maintenance-window-execution-tasks` example lists the tasks associated with the specified maintenance window execution.

```
aws ssm describe-maintenance-window-execution-tasks \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE"
```

Output:

```
{
    "WindowExecutionTaskIdentities": [
        {
            "Status": "SUCCESS",
            "TaskArn": "AWS-RunShellScript",
            "StartTime": 1487692834.684,
            "TaskType": "RUN_COMMAND",
            "EndTime": 1487692835.005,
            "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",
            "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"
        }
    ]
}
```

For more information, see View Information About Tasks and Task Executions (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see DescribeMaintenanceWindowExecutionTasks in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists the tasks associated with a maintenance window execution.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Output:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : SUCCESS
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskType          : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- For API details, see DescribeMaintenanceWindowExecutionTasks in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists the tasks associated with a maintenance window execution.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Output:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : SUCCESS
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskType          : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- For API details, see DescribeMaintenanceWindowExecutionTasks in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeMaintenanceWindowExecutions` with a CLI

The following code examples show how to use `DescribeMaintenanceWindowExecutions`.

CLI

AWS CLI

Example 1: To list all executions for a maintenance window

The following `describe-maintenance-window-executions` example lists all of the executions for the specified maintenance window.

```
aws ssm describe-maintenance-window-executions \
    --window-id "mw-ab12cd34eEXAMPLE"
```

Output:

```
{
    "WindowExecutions": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowExecutionId": "6027b513-64fe-4cf0-be7d-1191aEXAMPLE",
            "Status": "IN_PROGRESS",
            "StartTime": "2021-08-04T11:00:00.000000-07:00"

        },
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowExecutionId": "ff75b750-4834-4377-8f61-b3cadEXAMPLE",
            "Status": "SUCCESS",
            "StartTime": "2021-08-03T11:00:00.000000-07:00",
            "EndTime": "2021-08-03T11:37:21.450000-07:00"
        },
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowExecutionId": "9fac7dd9-ff21-42a5-96ad-bbc4bEXAMPLE",
            "Status": "FAILED",
```

```
            "StatusDetails": "One or more tasks in the orchestration failed.",
            "StartTime": "2021-08-02T11:00:00.000000-07:00",
            "EndTime": "2021-08-02T11:22:36.190000-07:00"
        }
    ]
}
```

Example 2: To list all executions for a maintenance window before a specified date

The following `describe-maintenance-window-executions` example lists all of the executions for the specified maintenance window before the specified date.

```
aws ssm describe-maintenance-window-executions \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=ExecutedBefore,Values=2021-08-03T00:00:00Z"
```

Output:

```
{
    "WindowExecutions": [
        {
        "WindowId": "mw-ab12cd34eEXAMPLE",
        "WindowExecutionId": "9fac7dd9-ff21-42a5-96ad-bbc4bEXAMPLE",
        "Status": "FAILED",
        "StatusDetails": "One or more tasks in the orchestration failed.",
        "StartTime": "2021-08-02T11:00:00.000000-07:00",
        "EndTime": "2021-08-02T11:22:36.190000-07:00"
    }
    ]
}
```

Example 3: To list all executions for a maintenance window after a specified date

The following `describe-maintenance-window-executions` example lists all of the executions for the specified maintenance window after the specified date.

```
aws ssm describe-maintenance-window-executions \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=ExecutedAfter,Values=2021-08-04T00:00:00Z"
```

Output:

```
{
    "WindowExecutions": [
        {
        "WindowId": "mw-ab12cd34eEXAMPLE",
        "WindowExecutionId": "6027b513-64fe-4cf0-be7d-1191aEXAMPLE",
        "Status": "IN_PROGRESS",
        "StartTime": "2021-08-04T11:00:00.000000-07:00"
        }
    ]
}
```

For more information, see View information about tasks and task executions (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see DescribeMaintenanceWindowExecutions in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all of the executions for a maintenance window.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Output:

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime         : 2/20/2017 6:30:16 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
WindowExecutionId : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId          : mw-03eb9db42890fb82d
```

Example 2: This example lists all of the executions for a maintenance window before a specified date.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Example 3: This example lists all of the executions for a maintenance window after a specified date.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- For API details, see DescribeMaintenanceWindowExecutions in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all of the executions for a maintenance window.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Output:

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime         : 2/20/2017 6:30:16 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
WindowExecutionId : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId          : mw-03eb9db42890fb82d
```

Example 2: This example lists all of the executions for a maintenance window before a specified date.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Example 3: This example lists all of the executions for a maintenance window after a specified date.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- For API details, see DescribeMaintenanceWindowExecutions in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeMaintenanceWindowTargets` with a CLI

The following code examples show how to use `DescribeMaintenanceWindowTargets`.

CLI

AWS CLI

Example 1: To list all targets for a Maintenance Window

The following `describe-maintenance-window-targets` example lists all of the targets for a maintenance window.

```
aws ssm describe-maintenance-window-targets \
    --window-id "mw-06cf17cbefEXAMPLE"
```

Output:

```
{
    "Targets": [
        {
            "ResourceType": "INSTANCE",
            "OwnerInformation": "Single instance",
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "Targets": [
                {
                    "Values": [
                        "i-0000293ffdEXAMPLE"
                    ],
                    "Key": "InstanceIds"
                }
            ],
            "WindowTargetId": "350d44e6-28cc-44e2-951f-4b2c9EXAMPLE"
        },
        {
            "ResourceType": "INSTANCE",
            "OwnerInformation": "Two instances in a list",
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "Targets": [
```

```
                {
                    "Values": [
                        "i-0000293ffdEXAMPLE",
                        "i-0cb2b964d3EXAMPLE"
                    ],
                    "Key": "InstanceIds"
                }
            ],
            "WindowTargetId": "e078a987-2866-47be-bedd-d9cf4EXAMPLE"
        }
    ]
}
```

Example 2: To list all targets for a maintenance window matching a specific owner information value

This `describe-maintenance-window-targets` example lists all of the targets for a maintenance window with a specific value.

```
aws ssm describe-maintenance-window-targets \
    --window-id "mw-0ecb1226ddEXAMPLE" \
    --filters "Key=OwnerInformation,Values=CostCenter1"
```

Output:

```
{
    "Targets": [
        {
            "WindowId": "mw-0ecb1226ddEXAMPLE",
            "WindowTargetId": "da89dcc3-7f9c-481d-ba2b-edcb7d0057f9",
            "ResourceType": "INSTANCE",
            "Targets": [
                {
                    "Key": "tag:Environment",
                    "Values": [
                        "Prod"
                    ]
                }
            ],
            "OwnerInformation": "CostCenter1",
            "Name": "ProdTarget1"
        }
    ]
```

```
}
```

For more information, see View Information About Maintenance Windows (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see DescribeMaintenanceWindowTargets in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all of the targets for a maintenance window.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Output:

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : e078a987-2866-47be-bedd-d9cf49177d3a
```

- For API details, see DescribeMaintenanceWindowTargets in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all of the targets for a maintenance window.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Output:

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
```

```
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : e078a987-2866-47be-bedd-d9cf49177d3a
```

- For API details, see DescribeMaintenanceWindowTargets in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeMaintenanceWindowTasks` with a CLI

The following code examples show how to use `DescribeMaintenanceWindowTasks`.

CLI

AWS CLI

Example 1: To list all tasks for a maintenance window

The following `describe-maintenance-window-tasks` example lists all of the tasks for the specified maintenance window.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-06cf17cbefEXAMPLE"
```

Output:

```
{
    "Tasks": [
        {
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "WindowTaskId": "018b31c3-2d77-4b9e-bd48-c91edEXAMPLE",
            "TaskArn": "AWS-RestartEC2Instance",
            "TaskParameters": {},
```

```
            "Type": "AUTOMATION",
            "Description": "Restarting EC2 Instance for maintenance",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "My-Automation-Example-Task",
            "Priority": 0,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ]
        },
        {
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "WindowTaskId": "1943dee0-0a17-4978-9bf4-3cc2fEXAMPLE",
            "TaskArn": "AWS-DisableS3BucketPublicReadWrite",
            "TaskParameters": {},
            "Type": "AUTOMATION",
            "Description": "Automation task to disable read/write access on
 public S3 buckets",
            "MaxConcurrency": "10",
            "MaxErrors": "5",
            "Name": "My-Disable-S3-Public-Read-Write-Access-Automation-Task",
            "Priority": 0,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ]
        }
    ]
}
```

Example 2: To list all tasks for a maintenance window that invokes the AWS-RunPowerShellScript command document

The following `describe-maintenance-window-tasks` example lists all of the tasks for the specified maintenance window that invokes the `AWS-RunPowerShellScript` command document.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=TaskArn,Values=AWS-RunPowerShellScript"
```

Output:

```
{
    "Tasks": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
            "TaskArn": "AWS-RunPowerShellScript",
            "Type": "RUN_COMMAND",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ],
            "TaskParameters": {},
            "Priority": 1,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "MyTask"
        }
    ]
}
```

Example 3: To list all tasks for a maintenance window that have a Priority of 3

The following `describe-maintenance-window-tasks` example lists all of the tasks for the specified maintenance window that have a `Priority` of 3.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=Priority,Values=3"
```

Output:

```
{
    "Tasks": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
            "TaskArn": "AWS-RunPowerShellScript",
            "Type": "RUN_COMMAND",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ],
            "TaskParameters": {},
            "Priority": 3,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "MyRunCommandTask"
        },
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "ee45feff-ad65-4a6c-b478-5cab8EXAMPLE",
            "TaskArn": "AWS-RestartEC2Instance",
            "Type": "AUTOMATION",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
```

```
                }
            ],
            "TaskParameters": {},
            "Priority": 3,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "10",
            "MaxErrors": "5",
            "Name": "My-Automation-Task",
            "Description": "A description for my Automation task"
        }
    ]
}
```

Example 4: To list all tasks for a maintenance window that have a Priority of 1 and use Run Command

This `describe-maintenance-window-tasks` example lists all of the tasks for the specified maintenance window that have a `Priority` of 1 and use `Run Command`.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=Priority,Values=1" "Key=TaskType,Values=RUN_COMMAND"
```

Output:

```
{
    "Tasks": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
            "TaskArn": "AWS-RunPowerShellScript",
            "Type": "RUN_COMMAND",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ],
            "TaskParameters": {},
            "Priority": 1,
```

```
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "MyRunCommandTask"
        }
    ]
}
```

For more information, see View information about maintenance windows (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see DescribeMaintenanceWindowTasks in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all of the tasks for a maintenance window.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Output:

```
LoggingInfo    :
MaxConcurrency : 1
MaxErrors      : 1
Priority       : 10
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets        : {InstanceIds}
TaskArn        : AWS-RunShellScript
TaskParameters : {[commands,
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
Type           : RUN_COMMAND
WindowId       : mw-06cf17cbefcb4bf4f
WindowTaskId   : a23e338d-ff30-4398-8aa3-09cd052ebf17
```

- For API details, see DescribeMaintenanceWindowTasks in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all of the tasks for a maintenance window.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Output:

```
LoggingInfo    :
MaxConcurrency : 1
MaxErrors      : 1
Priority       : 10
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets        : {InstanceIds}
TaskArn        : AWS-RunShellScript
TaskParameters : {[commands,
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
Type           : RUN_COMMAND
WindowId       : mw-06cf17cbefcb4bf4f
WindowTaskId   : a23e338d-ff30-4398-8aa3-09cd052ebf17
```

- For API details, see DescribeMaintenanceWindowTasks in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeMaintenanceWindows` with a CLI

The following code examples show how to use `DescribeMaintenanceWindows`.

CLI

AWS CLI

Example 1: To list all maintenance windows

The following `describe-maintenance-windows` example lists all maintenance windows in your AWS account in the current Region.

```
aws ssm describe-maintenance-windows
```

Output:

```
{
    "WindowIdentities": [
        {
            "WindowId": "mw-0ecb1226ddEXAMPLE",
            "Name": "MyMaintenanceWindow-1",
            "Enabled": true,
            "Duration": 2,
            "Cutoff": 1,
            "Schedule": "rate(180 minutes)",
            "NextExecutionTime": "2020-02-12T23:19:20.596Z"
        },
        {
            "WindowId": "mw-03eb9db428EXAMPLE",
            "Name": "MyMaintenanceWindow-2",
            "Enabled": true,
            "Duration": 3,
            "Cutoff": 1,
            "Schedule": "rate(7 days)",
            "NextExecutionTime": "2020-02-17T23:22:00.956Z"
        },
    ]
}
```

Example 2: To list all enabled maintenance windows

The following `describe-maintenance-windows` example lists all enabled maintenance windows.

```
aws ssm describe-maintenance-windows \
    --filters "Key=Enabled,Values=true"
```

Example 3: To list maintenance windows matching a specific name

This `describe-maintenance-windows` example lists all maintenance windows with the specified name.

```
aws ssm describe-maintenance-windows \
    --filters "Key=Name,Values=MyMaintenanceWindow"
```

For more information, see View Information About Maintenance Windows (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see DescribeMaintenanceWindows in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all maintenance windows on your account.

```
Get-SSMMaintenanceWindowList
```

Output:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- For API details, see DescribeMaintenanceWindows in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all maintenance windows on your account.

```
Get-SSMMaintenanceWindowList
```

Output:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- For API details, see DescribeMaintenanceWindows in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeOpsItems` with an AWS SDK or CLI

The following code examples show how to use `DescribeOpsItems`.

CLI

AWS CLI

To list a set of OpsItems

The following `describe-ops-items` example displays a list of all open OpsItems in your AWS account.

```
aws ssm describe-ops-items \
    --ops-item-filters "Key=Status,Values=Open,Operator=Equal"
```

Output:

```
{
    "OpsItemSummaries": [
        {
            "CreatedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-CWE-
Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "CreatedTime": "2020-03-14T17:02:46.375000-07:00",
            "LastModifiedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-
CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "LastModifiedTime": "2020-03-14T17:02:46.375000-07:00",
            "Source": "SSM",
            "Status": "Open",
            "OpsItemId": "oi-7cfc5EXAMPLE",
            "Title": "SSM Maintenance Window execution failed",
            "OperationalData": {
                "/aws/dedup": {
                    "Value": "{\"dedupString\":\"SSMOpsItems-SSM-maintenance-
window-execution-failed\"}",
                    "Type": "SearchableString"
                },
                "/aws/resources": {
```

```
                    "Value": "[{\"arn\":\"arn:aws:ssm:us-
east-2:111222333444:maintenancewindow/mw-034093d322EXAMPLE\"}]",
                    "Type": "SearchableString"
                }
            },
            "Category": "Availability",
            "Severity": "3"
        },
        {
            "CreatedBy": "arn:aws:sts::1112223233444:assumed-role/OpsItem-CWE-
Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "CreatedTime": "2020-02-26T11:43:15.426000-08:00",
            "LastModifiedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-
CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "LastModifiedTime": "2020-02-26T11:43:15.426000-08:00",
            "Source": "EC2",
            "Status": "Open",
            "OpsItemId": "oi-6f966EXAMPLE",
            "Title": "EC2 instance stopped",
            "OperationalData": {
                "/aws/automations": {
                    "Value": "[ { \"automationType\": \"AWS:SSM:Automation\",
 \"automationId\": \"AWS-RestartEC2Instance\" } ]",
                    "Type": "SearchableString"
                },
                "/aws/dedup": {
                    "Value": "{\"dedupString\":\"SSMOpsItems-EC2-instance-stopped
\"}",
                    "Type": "SearchableString"
                },
                "/aws/resources": {
                    "Value": "[{\"arn\":\"arn:aws:ec2:us-
east-2:111222333444:instance/i-0beccfbc02EXAMPLE\"}]",
                    "Type": "SearchableString"
                }
            },
            "Category": "Availability",
            "Severity": "3"
        }
    ]
}
```

For more information, see Working with OpsItems in the *AWS Systems Manager User Guide*.

- For API details, see DescribeOpsItems in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /**
    * Describes AWS SSM OpsItems asynchronously.
    *
    * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
    *
    * This method initiates an asynchronous request to describe SSM OpsItems.
    * If the request is successful, it prints the title and status of each
OpsItem.
    * If an exception occurs, it handles the error appropriately.
    */
   public void describeOpsItems(String key) {
       OpsItemFilter filter = OpsItemFilter.builder()
               .key(OpsItemFilterKey.OPS_ITEM_ID)
               .values(key)
               .operator(OpsItemFilterOperator.EQUAL)
               .build();

       DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
               .maxResults(10)
               .opsItemFilters(filter)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().describeOpsItems(itemsRequest)
                   .thenAccept(itemsResponse -> {
                       List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                       for (OpsItemSummary item : items) {
                           System.out.println("The item title is " +
item.title() + " and the status is " + item.status().toString());
```

```
                        }
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- For API details, see DescribeOpsItems in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import {
  OpsItemFilterOperator,
  OpsItemFilterKey,
  paginateDescribeOpsItems,
```

```
  SSMClient,
} from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Describe SSM OpsItems.
 * @param {{ opsItemId: string }}
 */
export const main = async ({ opsItemId }) => {
  const client = new SSMClient({});
  try {
    const describeOpsItemsPaginated = [];
    for await (const page of paginateDescribeOpsItems(
      { client },
      {
        OpsItemFilters: {
          Key: OpsItemFilterKey.OPSITEM_ID,
          Operator: OpsItemFilterOperator.EQUAL,
          Values: opsItemId,
        },
      },
    )) {
      describeOpsItemsPaginated.push(...page.OpsItemSummaries);
    }
    console.log("Here are the ops items:");
    console.log(describeOpsItemsPaginated);
    return { OpsItemSummaries: describeOpsItemsPaginated };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    }
    throw caught;
  }
};
```

- For API details, see DescribeOpsItems in *AWS SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def describe(self):
        """
        Describe an OpsItem.
        """
        try:
            paginator = self.ssm_client.get_paginator("describe_ops_items")
            ops_items = []
            for page in paginator.paginate(
                OpsItemFilters=[
                    {"Key": "OpsItemId", "Values": [self.id], "Operator":
 "Equal"}
                ]
            ):
```

```
                ops_items.extend(page["OpsItemSummaries"])

            for item in ops_items:
                print(
                    f"The item title is {item['Title']} and the status is
{item['Status']}"
                )
            return len(ops_items) > 0
        except ClientError as err:
            logger.error(
                "Couldn't describe ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see DescribeOpsItems in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeParameters` with an AWS SDK or CLI

The following code examples show how to use `DescribeParameters`.

CLI

AWS CLI

Example 1: To list all parameters

The following `describe-parameters` example lists all parameters in the current AWS account and Region.

```
aws ssm describe-parameters
```

Output:

```
{
    "Parameters": [
        {
            "Name": "MySecureStringParameter",
            "Type": "SecureString",
            "KeyId": "alias/aws/ssm",
            "LastModifiedDate": 1582155479.205,
            "LastModifiedUser": "arn:aws:sts::111222333444:assumed-role/Admin/
Richard-Roe-Managed",
            "Description": "This is a SecureString parameter",
            "Version": 2,
            "Tier": "Advanced",
            "Policies": [
                {
                    "PolicyText": "{\"Type\":\"Expiration\",\"Version\":\"1.0\",
\"Attributes\":{\"Timestamp\":\"2020-07-07T22:30:00Z\"}}",
                    "PolicyType": "Expiration",
                    "PolicyStatus": "Pending"
                },
                {
                    "PolicyText": "{\"Type\":\"ExpirationNotification\",\"Version
\":\"1.0\",\"Attributes\":{\"Before\":\"12\",\"Unit\":\"Hours\"}}",
                    "PolicyType": "ExpirationNotification",
                    "PolicyStatus": "Pending"
                }
            ]
        },
        {
            "Name": "MyStringListParameter",
            "Type": "StringList",
            "LastModifiedDate": 1582154764.222,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is a StringList parameter",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "LastModifiedDate": 1582154711.976,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Alejandro-
Rosalez",
```

```
            "Description": "This is a String parameter",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "latestAmi",
            "Type": "String",
            "LastModifiedDate": 1580862415.521,
            "LastModifiedUser": "arn:aws:sts::111222333444:assumed-role/lambda-
ssm-role/Automation-UpdateSSM-Param",
            "Version": 3,
            "Tier": "Standard",
            "Policies": []
        }
    ]
}
```

Example 2: To list all parameters matching specific metadata

This `describe-parameters` example lists all parameters matching a filter.

aws ssm describe-parameters --filters "Key=Type,Values=StringList"

Output:

```
{
    "Parameters": [
        {
            "Name": "MyStringListParameter",
            "Type": "StringList",
            "LastModifiedDate": 1582154764.222,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is a StringList parameter",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        }
    ]
}
```

For more information, see Searching for Systems Manager Parameters in the *AWS Systems Manager User Guide*.

- For API details, see DescribeParameters in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetParameter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String paraName = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    public static void getParaValue(SsmClient ssmClient, String paraName) {
        try {
            GetParameterRequest parameterRequest = GetParameterRequest.builder()
                    .name(paraName)
                    .build();

            GetParameterResponse parameterResponse =
 ssmClient.getParameter(parameterRequest);
            System.out.println("The parameter value is " +
 parameterResponse.parameter().value());

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see DescribeParameters in *AWS SDK for Java 2.x API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all parameters.

```
Get-SSMParameterList
```

Output:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- For API details, see DescribeParameters in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all parameters.

```
Get-SSMParameterList
```

Output:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- For API details, see DescribeParameters in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Rust

SDK for Rust

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
async fn show_parameters(client: &Client) -> Result<(), Error> {
    let resp = client.describe_parameters().send().await?;
```

```
    for param in resp.parameters() {
        println!("  {}", param.name().unwrap_or_default());
    }

    Ok(())
}
```

- For API details, see DescribeParameters in *AWS SDK for Rust API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribePatchBaselines` with a CLI

The following code examples show how to use `DescribePatchBaselines`.

CLI

AWS CLI

Example 1: To list all patch baselines

The following `describe-patch-baselines` example retreives details for all patch baselines in your account in the current Region.

```
aws ssm describe-patch-baselines
```

Output:

```
{
    "BaselineIdentities": [
        {
            "BaselineName": "AWS-SuseDefaultPatchBaseline",
            "DefaultBaseline": true,
            "BaselineDescription": "Default Patch Baseline for Suse Provided by
 AWS.",
            "BaselineId": "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-0123fdb36e334a3b2",
            "OperatingSystem": "SUSE"
```

```
        },
        {
            "BaselineName": "AWS-DefaultPatchBaseline",
            "DefaultBaseline": false,
            "BaselineDescription": "Default Patch Baseline Provided by AWS.",
            "BaselineId": "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-020d361a05defe4ed",
            "OperatingSystem": "WINDOWS"
        },
        ...
        {
            "BaselineName": "MyWindowsPatchBaseline",
            "DefaultBaseline": true,
            "BaselineDescription": "My patch baseline for EC2 instances for
 Windows Server",
            "BaselineId": "pb-0ad00e0dd7EXAMPLE",
            "OperatingSystem": "WINDOWS"
        }
    ]
}
```

Example 2: To list all patch baselines provided by AWS

The following `describe-patch-baselines` example lists all patch baselines provided by AWS.

```
aws ssm describe-patch-baselines \
    --filters "Key=OWNER,Values=[AWS]"
```

Example 3: To list all patch baselines that you own

The following `describe-patch-baselines` example lists all custom patch baselines created in your account in the current Region.

```
aws ssm describe-patch-baselines \
    --filters "Key=OWNER,Values=[Self]"
```

For more information, see About Predefined and Custom Patch Baselines in the *AWS Systems Manager User Guide*.

- For API details, see DescribePatchBaselines in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all patch baselines.

```
Get-SSMPatchBaseline
```

Output:

```
BaselineDescription                                                BaselineId
                                                        BaselineName
-------------------                                                ----------
                                                        ------------
Default Patch Baseline Provided by AWS.                            arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems
 pb-045f10b4f382baeda
 Production-B...
Baseline containing all updates approved for production systems
 pb-0a2f1059b670ebd31
 Production-B...
```

Example 2: This example lists all patch baselines provided by AWS. The syntax used by this example requires PowerShell version 3 or later.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Output:

```
Get-SSMPatchBaseline -Filter $filter1
```

Example 3: This example lists all patch baselines with you as the owner. The syntax used by this example requires PowerShell version 3 or later.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Output:

```
Get-SSMPatchBaseline -Filter $filter1
```

Example 4: With PowerShell version 2, you must use New-Object to create each tag.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Output:

```
BaselineDescription                     BaselineId
                            BaselineName             DefaultBaselin

                                                     e
-------------------                     ----------
                            ------------             --------------
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline
 True
```

- For API details, see DescribePatchBaselines in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all patch baselines.

```
Get-SSMPatchBaseline
```

Output:

```
BaselineDescription                                          BaselineId
                                                  BaselineName
-------------------                                          ----------
                                                  ------------
Default Patch Baseline Provided by AWS.                      arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems
 pb-045f10b4f382baeda
 Production-B...
```

```
Baseline containing all updates approved for production systems
 pb-0a2f1059b670ebd31
 Production-B...
```

Example 2: This example lists all patch baselines provided by AWS. The syntax used by this example requires PowerShell version 3 or later.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Output:

```
Get-SSMPatchBaseline -Filter $filter1
```

Example 3: This example lists all patch baselines with you as the owner. The syntax used by this example requires PowerShell version 3 or later.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Output:

```
Get-SSMPatchBaseline -Filter $filter1
```

Example 4: With PowerShell version 2, you must use New-Object to create each tag.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Output:

```
BaselineDescription                     BaselineId
                            BaselineName             DefaultBaselin

                                                     e
-------------------                     ----------
                            ------------             --------------
```

```
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline
 True
```

- For API details, see DescribePatchBaselines in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribePatchGroupState` with a CLI

The following code examples show how to use `DescribePatchGroupState`.

CLI

AWS CLI

To get the state of a patch group

The following `describe-patch-group-state` example retrieves the high-level patch compliance summary for a patch group.

```
aws ssm describe-patch-group-state \
    --patch-group "Production"
```

Output:

```
{
    "Instances": 21,
    "InstancesWithCriticalNonCompliantPatches": 1,
    "InstancesWithFailedPatches": 2,
    "InstancesWithInstalledOtherPatches": 3,
    "InstancesWithInstalledPatches": 21,
    "InstancesWithInstalledPendingRebootPatches": 2,
    "InstancesWithInstalledRejectedPatches": 1,
    "InstancesWithMissingPatches": 3,
    "InstancesWithNotApplicablePatches": 4,
    "InstancesWithOtherNonCompliantPatches": 1,
    "InstancesWithSecurityNonCompliantPatches": 1,
```

```
    "InstancesWithUnreportedNotApplicablePatches": 2
}
```

For more information, see About patch groups <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-patchgroups.html>__ and Understanding patch compliance state values in the *AWS Systems Manager User Guide*.

- For API details, see DescribePatchGroupState in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example gets the high-level patch compliance summary for a patch group.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Output:

```
Instances                          : 4
InstancesWithFailedPatches         : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches      : 3
InstancesWithMissingPatches        : 0
InstancesWithNotApplicablePatches  : 0
```

- For API details, see DescribePatchGroupState in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example gets the high-level patch compliance summary for a patch group.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Output:

```
Instances                          : 4
InstancesWithFailedPatches         : 1
```

```
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches      : 3
InstancesWithMissingPatches        : 0
InstancesWithNotApplicablePatches  : 0
```

- For API details, see DescribePatchGroupState in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribePatchGroups` with a CLI

The following code examples show how to use `DescribePatchGroups`.

CLI

AWS CLI

To display patch group registrations

The following `describe-patch-groups` example lists the patch group registrations.

```
aws ssm describe-patch-groups
```

Output:

```
{
    "Mappings": [
        {
            "PatchGroup": "Production",
            "BaselineIdentity": {
                "BaselineId": "pb-0123456789abcdef0",
                "BaselineName": "ProdPatching",
                "OperatingSystem": "WINDOWS",
                "BaselineDescription": "Patches for Production",
                "DefaultBaseline": false
            }
        },
        {
```

```
            "PatchGroup": "Development",
            "BaselineIdentity": {
                "BaselineId": "pb-0713accee01234567",
                "BaselineName": "DevPatching",
                "OperatingSystem": "WINDOWS",
                "BaselineDescription": "Patches for Development",
                "DefaultBaseline": true
            }
        },
        ...
    ]
}
```

For more information, see Create a Patch Group <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html>__ and Add a Patch Group to a Patch Baseline in the *AWS Systems Manager User Guide*.

- For API details, see DescribePatchGroups in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists the patch group registrations.

```
Get-SSMPatchGroup
```

Output:

```
BaselineIdentity                                                PatchGroup
----------------                                                ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- For API details, see DescribePatchGroups in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists the patch group registrations.

```
Get-SSMPatchGroup
```

Output:

```
BaselineIdentity                                                PatchGroup
----------------                                                ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- For API details, see DescribePatchGroups in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetAutomationExecution` with a CLI

The following code examples show how to use `GetAutomationExecution`.

CLI

AWS CLI

To display details about an automation execution

The following `get-automation-execution` example displays detailed information about an Automation execution.

```
aws ssm get-automation-execution \
    --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

Output:

```
{
    "AutomationExecution": {
        "AutomationExecutionId": "73c8eef8-f4ee-4a05-820c-e354fEXAMPLE",
        "DocumentName": "AWS-StartEC2Instance",
        "DocumentVersion": "1",
        "ExecutionStartTime": 1583737233.748,
        "ExecutionEndTime": 1583737234.719,
        "AutomationExecutionStatus": "Success",
        "StepExecutions": [
```

```
            {
                "StepName": "startInstances",
                "Action": "aws:changeInstanceState",
                "ExecutionStartTime": 1583737234.134,
                "ExecutionEndTime": 1583737234.672,
                "StepStatus": "Success",
                "Inputs": {
                    "DesiredState": "\"running\"",
                    "InstanceIds": "[\"i-0cb99161f6EXAMPLE\"]"
                },
                "Outputs": {
                    "InstanceStates": [
                        "running"
                    ]
                },
                "StepExecutionId": "95e70479-cf20-4d80-8018-7e4e2EXAMPLE",
                "OverriddenParameters": {}
            }
        ],
        "StepExecutionsTruncated": false,
        "Parameters": {
            "AutomationAssumeRole": [
                ""
            ],
            "InstanceId": [
                "i-0cb99161f6EXAMPLE"
            ]
        },
        "Outputs": {},
        "Mode": "Auto",
        "ExecutedBy": "arn:aws:sts::29884EXAMPLE:assumed-role/mw_service_role/
OrchestrationService",
        "Targets": [],
        "ResolvedTargets": {
            "ParameterValues": [],
            "Truncated": false
        }
    }
}
```

For more information, see Walkthrough: Patch a Linux AMI (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see GetAutomationExecution in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example displays the details of an Automation Execution.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Output:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
 are not authorized to perform this operation. Encoded
                            authorization failure message:
 B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
                            Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                            pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU

 gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-q0CK0ezS3xfh_y0ilaUGOAZG-
xjQFuvU_JZedWpla3xi-MZsmblAifBI
                            (Service: AmazonEC2; Status Code: 403; Error Code:
 UnauthorizedOperation; Request ID:
                            6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                : {[AutomationAssumeRole,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

```
StepExecutions            : {launchInstance, updateOSSoftware, stopInstance,
 createImage...}
```

Example 2: This example lists step details for the given automation execution id

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Output:

```
StepName                  Action                  StepStatus ValidNextSteps
--------                  ------                  ---------- --------------
LaunchInstance            aws:runInstances        Success
 {OSCompatibilityCheck}
OSCompatibilityCheck      aws:runCommand          Success    {RunPreUpdateScript}
RunPreUpdateScript        aws:runCommand          Success    {UpdateEC2Config}
UpdateEC2Config           aws:runCommand          Cancelled  {}
UpdateSSMAgent            aws:runCommand          Pending    {}
UpdateAWSPVDriver         aws:runCommand          Pending    {}
UpdateAWSEnaNetworkDriver aws:runCommand          Pending    {}
UpdateAWSNVMe             aws:runCommand          Pending    {}
InstallWindowsUpdates     aws:runCommand          Pending    {}
RunPostUpdateScript       aws:runCommand          Pending    {}
RunSysprepGeneralize      aws:runCommand          Pending    {}
StopInstance              aws:changeInstanceState Pending    {}
CreateImage               aws:createImage         Pending    {}
TerminateInstance         aws:changeInstanceState Pending    {}
```

- For API details, see GetAutomationExecution in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example displays the details of an Automation Execution.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Output:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
```

```
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
 are not authorized to perform this operation. Encoded
                            authorization failure message:
 B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
                            Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                            pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU

 gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-q0CK0ezS3xfh_y0ilaUGOAZG-
xjQFuvU_JZedWpla3xi-MZsmblAifBI
                            (Service: AmazonEC2; Status Code: 403; Error Code:
 UnauthorizedOperation; Request ID:
                            6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                : {[AutomationAssumeRole,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions            : {launchInstance, updateOSSoftware, stopInstance,
 createImage...}
```

Example 2: This example lists step details for the given automation execution id

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Output:

```
StepName                  Action                  StepStatus ValidNextSteps
--------                  ------                  ---------- --------------
```

```
LaunchInstance            aws:runInstances        Success
 {OSCompatibilityCheck}
OSCompatibilityCheck      aws:runCommand          Success    {RunPreUpdateScript}
RunPreUpdateScript        aws:runCommand          Success    {UpdateEC2Config}
UpdateEC2Config           aws:runCommand          Cancelled  {}
UpdateSSMAgent            aws:runCommand          Pending    {}
UpdateAWSPVDriver         aws:runCommand          Pending    {}
UpdateAWSEnaNetworkDriver aws:runCommand          Pending    {}
UpdateAWSNVMe             aws:runCommand          Pending    {}
InstallWindowsUpdates     aws:runCommand          Pending    {}
RunPostUpdateScript       aws:runCommand          Pending    {}
RunSysprepGeneralize      aws:runCommand          Pending    {}
StopInstance              aws:changeInstanceState Pending    {}
CreateImage               aws:createImage         Pending    {}
TerminateInstance         aws:changeInstanceState Pending    {}
```

- For API details, see GetAutomationExecution in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetCommandInvocation` with a CLI

The following code examples show how to use `GetCommandInvocation`.

CLI

AWS CLI

To display the details of a command invocation

The following `get-command-invocation` example lists all the invocations of the specified command on the specified instance.

```
aws ssm get-command-invocation \
    --command-id "ef7fdfd8-9b57-4151-a15c-db9a12345678" \
    --instance-id "i-1234567890abcdef0"
```

Output:

```
{
    "CommandId": "ef7fdfd8-9b57-4151-a15c-db9a12345678",
    "InstanceId": "i-1234567890abcdef0",
    "Comment": "b48291dd-ba76-43e0-b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-
d6ce8EXAMPLE",
    "DocumentName": "AWS-UpdateSSMAgent",
    "DocumentVersion": "",
    "PluginName": "aws:updateSsmAgent",
    "ResponseCode": 0,
    "ExecutionStartDateTime": "2020-02-19T18:18:03.419Z",
    "ExecutionElapsedTime": "PT0.091S",
    "ExecutionEndDateTime": "2020-02-19T18:18:03.419Z",
    "Status": "Success",
    "StatusDetails": "Success",
    "StandardOutputContent": "Updating amazon-ssm-agent from 2.3.842.0 to latest
\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-ssm-us-
east-2/ssm-agent-manifest.json\namazon-ssm-agent 2.3.842.0 has already been
 installed, update skipped\n",
    "StandardOutputUrl": "",
    "StandardErrorContent": "",
    "StandardErrorUrl": "",
    "CloudWatchOutputConfig": {
        "CloudWatchLogGroupName": "",
        "CloudWatchOutputEnabled": false
    }
}
```

For more information, see Understanding Command Statuses in the *AWS Systems Manager User Guide*.

- For API details, see GetCommandInvocation in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example displays the details of a command executed on an instance.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Output:

```
CommandId              : b8eac879-0541-439d-94ec-47a80d554f44
Comment                : IP config
DocumentName           : AWS-RunShellScript
ExecutionElapsedTime   : PT0.004S
ExecutionEndDateTime   : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId             : i-0cb2b964d3e14fd9f
PluginName             : aws:runShellScript
ResponseCode           : 0
StandardErrorContent   :
StandardErrorUrl       :
StandardOutputContent  :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- For API details, see GetCommandInvocation in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example displays the details of a command executed on an instance.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Output:

```
CommandId              : b8eac879-0541-439d-94ec-47a80d554f44
Comment                : IP config
DocumentName           : AWS-RunShellScript
ExecutionElapsedTime   : PT0.004S
ExecutionEndDateTime   : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId             : i-0cb2b964d3e14fd9f
PluginName             : aws:runShellScript
ResponseCode           : 0
StandardErrorContent   :
StandardErrorUrl       :
StandardOutputContent  :
StandardOutputUrl      :
Status                 : Success
```

```
StatusDetails          : Success
```

- For API details, see GetCommandInvocation in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetConnectionStatus` with a CLI

The following code examples show how to use `GetConnectionStatus`.

CLI

AWS CLI

To display the connection status of a managed instance

This `get-connection-status` example returns the connection status of the specified managed instance.

```
aws ssm get-connection-status \
    --target i-1234567890abcdef0
```

Output:

```
{
    "Target": "i-1234567890abcdef0",
    "Status": "connected"
}
```

- For API details, see GetConnectionStatus in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example retrieves the Session Manager connection status for an instance to determine whether it is connected and ready to receive Session Manager connections.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Output:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- For API details, see GetConnectionStatus in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example retrieves the Session Manager connection status for an instance to determine whether it is connected and ready to receive Session Manager connections.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Output:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- For API details, see GetConnectionStatus in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetDefaultPatchBaseline` with a CLI

The following code examples show how to use `GetDefaultPatchBaseline`.

CLI

AWS CLI

Example 1: To display the default Windows patch baseline

The following `get-default-patch-baseline` example retrieves details for the default patch baseline for Windows Server.

```
aws ssm get-default-patch-baseline
```

Output:

```
{
  "BaselineId": "pb-0713accee01612345",
  "OperatingSystem": "WINDOWS"
}
```

Example 2: To display the default patch baseline for Amazon Linux

The following `get-default-patch-baseline` example retrieves details for the default patch baseline for Amazon Linux.

```
aws ssm get-default-patch-baseline \
    --operating-system AMAZON_LINUX
```

Output:

```
{
    "BaselineId": "pb-047c6eb9c8fc12345",
    "OperatingSystem": "AMAZON_LINUX"
}
```

For more information, see About Predefined and Custom Patch Baselines <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-baselines.html>__ and Set an Existing Patch Baseline as the Default in the *AWS Systems Manager User Guide*.

- For API details, see GetDefaultPatchBaseline in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example displays the default patch baseline.

```
Get-SSMDefaultPatchBaseline
```

Output:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- For API details, see GetDefaultPatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example displays the default patch baseline.

```
Get-SSMDefaultPatchBaseline
```

Output:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- For API details, see GetDefaultPatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetDeployablePatchSnapshotForInstance` with a CLI

The following code examples show how to use `GetDeployablePatchSnapshotForInstance`.

CLI

AWS CLI

To retrieve the current snapshot for the patch baseline an instance uses

The following `get-deployable-patch-snapshot-for-instance` example retrieves details for the current snapshot for the specified patch baseline used by an instance. This command must be run from the instance using the instance credentials. To ensure it uses the instance credentials, run `aws configure` and specify only the Region of your instance. Leave the `Access Key` and `Secret Key` fields empty.

Tip: Use uuidgen to generate a snapshot-id.

```
aws ssm get-deployable-patch-snapshot-for-instance \
    --instance-id "i-1234567890abcdef0" \
    --snapshot-id "521c3536-930c-4aa9-950e-01234567abcd"
```

Output:

```
{
    "InstanceId": "i-1234567890abcdef0",
    "SnapshotId": "521c3536-930c-4aa9-950e-01234567abcd",
    "Product": "AmazonLinux2018.03",
    "SnapshotDownloadUrl": "https://patch-baseline-snapshot-us-
east-1.s3.amazonaws.com/
ed85194ef27214f5984f28b4d664d14f7313568fea7d4b6ac6c10ad1f729d7e7-773304212436/
AMAZON_LINUX-521c3536-930c-4aa9-950e-01234567abcd?X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20190215T164031Z&X-Amz-
SignedHeaders=host&X-Amz-Expires=86400&X-Amz-Credential=AKIAJ5C56P35AEBRX2QQ
%2F20190215%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-
Signature=efaaaf6e3878e77f48a6697e015efdbda9c426b09c5822055075c062f6ad2149"
}
```

For more information, see Parameter name: Snapshot ID in the *AWS Systems Manager User Guide*.

- For API details, see GetDeployablePatchSnapshotForInstance in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example displays the current snapshot for the patch baseline used by an Instance. This command must be run from the instance using the instance credentials. To ensure it uses the instance credentials, the example passes an `Amazon.Runtime.InstanceProfileAWSCredentials` object to the Credentials parameter.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Output:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Example 2: This example shows how to get the full SnapshotDownloadUrl. This command must be run from the instance using the instance credentials. To ensure it uses the instance credentials, the example configures the PowerShell session to use an `Amazon.Runtime.InstanceProfileAWSCredentials` object.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Output:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- For API details, see GetDeployablePatchSnapshotForInstance in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example displays the current snapshot for the patch baseline used by an Instance. This command must be run from the instance using the instance credentials. To ensure it uses the instance credentials, the example passes an `Amazon.Runtime.InstanceProfileAWSCredentials` object to the Credentials parameter.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Output:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
```

```
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Example 2: This example shows how to get the full SnapshotDownloadUrl. This command must be run from the instance using the instance credentials. To ensure it uses the instance credentials, the example configures the PowerShell session to use an `Amazon.Runtime.InstanceProfileAWSCredentials` object.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Output:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- For API details, see GetDeployablePatchSnapshotForInstance in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetDocument` with a CLI

The following code examples show how to use `GetDocument`.

CLI

AWS CLI

To get document content

The following `get-document` example displays the content of a Systems Manager document.

```
aws ssm get-document \
    --name "AWS-RunShellScript"
```

Output:

```
{
    "Name": "AWS-RunShellScript",
    "DocumentVersion": "1",
    "Status": "Active",
    "Content": "{\n    \"schemaVersion\":\"1.2\",\n    \"description\":\"Run
 a shell script or specify the commands to run.\",\n    \"parameters\":{\n
       \"commands\":{\n            \"type\":\"StringList\",\n
 \"description\":\"(Required) Specify a shell script or a command to run.\",
\n            \"minItems\":1,\n            \"displayType\":\"textarea\"\n
      },\n        \"workingDirectory\":{\n            \"type\":\"String\",\n
            \"default\":\"\",\n            \"description\":\"(Optional) The
 path to the working directory on your instance.\",\n            \"maxChars
\":4096\n        },\n        \"executionTimeout\":{\n            \"type\":
\"String\",\n            \"default\":\"3600\",\n            \"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).\",\n            \"allowedPattern\":\"([1-9][0-9]{0,4})|(1[0-6][0-9]
{4})|(17[0-1][0-9]{3})|(172[0-7][0-9]{2})|(172800)\"\n        }\n    },\n
 \"runtimeConfig\":{\n        \"aws:runShellScript\":{\n            \"properties
\":[\n                {\n                    \"id\":\"0.aws:runShellScript
\",\n                    \"runCommand\":\"{{ commands }}\",\n
      \"workingDirectory\":\"{{ workingDirectory }}\",\n
 \"timeoutSeconds\":\"{{ executionTimeout }}\"\n                }\n
  ]\n        }\n    }\n}\n",
    "DocumentType": "Command",
    "DocumentFormat": "JSON"
}
```

For more information, see AWS Systems Manager Documents in the *AWS Systems Manager User Guide*.

- For API details, see GetDocument in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example returns the content of a document.

```
Get-SSMDocument -Name "RunShellScript"
```

Output:

```
Content
-------
{...
```

Example 2: This example displays the complete contents of a document.

```
(Get-SSMDocument -Name "RunShellScript").Content
{
   "schemaVersion":"2.0",
   "description":"Run an updated script",
   "parameters":{
      "commands":{
         "type":"StringList",
         "description":"(Required) Specify a shell script or a command to run.",
         "minItems":1,
         "displayType":"textarea"
      }
   },
   "mainSteps":[
      {
         "action":"aws:runShellScript",
         "name":"runShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      },
      {
         "action":"aws:runPowerShellScript",
         "name":"runPowerShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      }
   ]
}
```

- For API details, see GetDocument in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example returns the content of a document.

```
Get-SSMDocument -Name "RunShellScript"
```

Output:

```
Content
-------
{...
```

Example 2: This example displays the complete contents of a document.

```
(Get-SSMDocument -Name "RunShellScript").Content
{
   "schemaVersion":"2.0",
   "description":"Run an updated script",
   "parameters":{
      "commands":{
         "type":"StringList",
         "description":"(Required) Specify a shell script or a command to run.",
         "minItems":1,
         "displayType":"textarea"
      }
   },
   "mainSteps":[
      {
         "action":"aws:runShellScript",
         "name":"runShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      },
      {
         "action":"aws:runPowerShellScript",
         "name":"runPowerShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      }
   ]
}
```

- For API details, see GetDocument in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetInventory` with a CLI

The following code examples show how to use `GetInventory`.

CLI

AWS CLI

To view your inventory

This example gets the custom metadata for your inventory.

Command:

```
aws ssm get-inventory
```

Output:

```
{
  "Entities": [
      {
          "Data": {
              "AWS:InstanceInformation": {
                  "Content": [
                      {
                          "ComputerName": "ip-172-31-44-222.us-
west-2.compute.internal",
                          "InstanceId": "i-0cb2b964d3e14fd9f",
                          "IpAddress": "172.31.44.222",
                          "AgentType": "amazon-ssm-agent",
                          "ResourceType": "EC2Instance",
                          "AgentVersion": "2.0.672.0",
                          "PlatformVersion": "2016.09",
                          "PlatformName": "Amazon Linux AMI",
                          "PlatformType": "Linux"
                      }
                  ],
                  "TypeName": "AWS:InstanceInformation",
                  "SchemaVersion": "1.0",
```

```
                    "CaptureTime": "2017-02-20T18:03:58Z"
                }
            },
            "Id": "i-0cb2b964d3e14fd9f"
        }
    ]
}
```

- For API details, see GetInventory in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example gets the custom metadata for your inventory.

```
Get-SSMInventory
```

Output:

```
Data
     Id
----
     --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- For API details, see GetInventory in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example gets the custom metadata for your inventory.

```
Get-SSMInventory
```

Output:

```
Data
     Id
----
     --
```

```
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- For API details, see GetInventory in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetInventorySchema` with a CLI

The following code examples show how to use `GetInventorySchema`.

CLI

AWS CLI

To view your inventory schema

This example returns a list of inventory type names for the account.

Command:

```
aws ssm get-inventory-schema
```

Output:

```
{
  "Schemas": [
      {
          "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"
                }
            ]
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

To view the inventory schema for a specific inventory type

This example return the inventory schema for a the AWS:AWSComponent inventory type.

Command:

```
aws ssm get-inventory-schema --type-name "AWS:AWSComponent"
```

- For API details, see GetInventorySchema in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example returns a list of inventory type names for the account.

```
Get-SSMInventorySchema
```

- For API details, see GetInventorySchema in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example returns a list of inventory type names for the account.

```
Get-SSMInventorySchema
```

- For API details, see GetInventorySchema in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetMaintenanceWindow` with a CLI

The following code examples show how to use `GetMaintenanceWindow`.

CLI

AWS CLI

To get information about a maintenance window

The following `get-maintenance-window` example retrieves details about the specified maintenance window.

```
aws ssm get-maintenance-window \
    --window-id "mw-03eb9db428EXAMPLE"
```

Output:

```
{
    "AllowUnassociatedTargets": true,
    "CreatedDate": 1515006912.957,
    "Cutoff": 1,
```

```
    "Duration": 6,
    "Enabled": true,
    "ModifiedDate": 2020-01-01T10:04:04.099Z,
    "Name": "My-Maintenance-Window",
    "Schedule": "rate(3 days)",
    "WindowId": "mw-03eb9db428EXAMPLE",
    "NextExecutionTime": "2020-02-25T00:08:15.099Z"
}
```

For more information, see View information about maintenance windows (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see GetMaintenanceWindow in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example gets details about a maintenance window.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Output:

```
AllowUnassociatedTargets : False
CreatedDate              : 2/20/2017 6:14:05 PM
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
ModifiedDate             : 2/20/2017 6:14:05 PM
Name                     : TestMaintWin
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- For API details, see GetMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example gets details about a maintenance window.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Output:

```
AllowUnassociatedTargets : False
CreatedDate              : 2/20/2017 6:14:05 PM
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
ModifiedDate             : 2/20/2017 6:14:05 PM
Name                     : TestMaintWin
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- For API details, see GetMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetMaintenanceWindowExecution` with a CLI

The following code examples show how to use `GetMaintenanceWindowExecution`.

CLI

AWS CLI

To get information about a maintenance window task execution

The following `get-maintenance-window-execution` example lists information about a task executed as part of the specified maintenance window execution.

```
aws ssm get-maintenance-window-execution \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE"
```

Output:

```
{
    "Status": "SUCCESS",
    "TaskIds": [
        "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"
```

```
    ],
    "StartTime": 1487692834.595,
    "EndTime": 1487692835.051,
    "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",
}
```

For more information, see View Information About Tasks and Task Executions (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see GetMaintenanceWindowExecution in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists information about a task executed as part of a maintenance window execution.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Output:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds           : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- For API details, see GetMaintenanceWindowExecution in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists information about a task executed as part of a maintenance window execution.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Output:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds           : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- For API details, see GetMaintenanceWindowExecution in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetMaintenanceWindowExecutionTask` with a CLI

The following code examples show how to use `GetMaintenanceWindowExecutionTask`.

CLI

AWS CLI

To get information about a maintenance window task execution

The following `get-maintenance-window-execution-task` example lists information about a task that is part of the specified maintenance window execution.

```
aws ssm get-maintenance-window-execution-task \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE" \
    --task-id "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"
```

Output:

```
{
    "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",
    "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE",
    "TaskArn": "AWS-RunPatchBaseline",
    "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
    "Type": "RUN_COMMAND",
    "TaskParameters": [
```

```
        {
            "BaselineOverride": {
                "Values": [
                    ""
                ]
            },
            "InstallOverrideList": {
                "Values": [
                    ""
                ]
            },
            "Operation": {
                "Values": [
                    "Scan"
                ]
            },
            "RebootOption": {
                "Values": [
                    "RebootIfNeeded"
                ]
            },
            "SnapshotId": {
                "Values": [
                    "{{ aws:ORCHESTRATION_ID }}"
                ]
            },
            "aws:InstanceId": {
                "Values": [
                    "i-02573cafcfEXAMPLE",
                    "i-0471e04240EXAMPLE",
                    "i-07782c72faEXAMPLE"
                ]
            }
        }
    ],
    "Priority": 1,
    "MaxConcurrency": "1",
    "MaxErrors": "3",
    "Status": "SUCCESS",
    "StartTime": "2021-08-04T11:45:35.088000-07:00",
    "EndTime": "2021-08-04T11:53:09.079000-07:00"
}
```

For more information, see View information about tasks and task executions (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see GetMaintenanceWindowExecutionTask in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists information about a task that was part of a maintenance window execution.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-
d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Output:

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole       : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The maximum error count was exceeded.
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters    :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsM
                   meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- For API details, see GetMaintenanceWindowExecutionTask in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists information about a task that was part of a maintenance window execution.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-
d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Output:

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole       : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The maximum error count was exceeded.
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters    :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsM
                   meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- For API details, see GetMaintenanceWindowExecutionTask in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetParameter` with an AWS SDK or CLI

The following code examples show how to use `GetParameter`.

CLI

AWS CLI

Example 1: To display the value of a parameter

The following `get-parameter` example lists the value for the specified single parameter.

```
aws ssm get-parameter \
```

```
    --name "MyStringParameter"
```

Output:

```
{
    "Parameter": {
        "Name": "MyStringParameter",
        "Type": "String",
        "Value": "Veni",
        "Version": 1,
        "LastModifiedDate": 1530018761.888,
        "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/MyStringParameter"
        "DataType": "text"
    }
}
```

For more information, see Working with Parameter Store in the *AWS Systems Manager User Guide*.

Example 2: To decrypt the value of a SecureString parameter

The following `get-parameter` example decrypts the value of the specified `SecureString` parameter.

```
aws ssm get-parameter \
    --name "MySecureStringParameter" \
    --with-decryption
```

Output:

```
{
    "Parameter": {
        "Name": "MySecureStringParameter",
        "Type": "SecureString",
        "Value": "16679b88-310b-4895-a943-e0764EXAMPLE",
        "Version": 2,
        "LastModifiedDate": 1582155479.205,
        "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/
MySecureStringParameter"
        "DataType": "text"
    }
}
```

For more information, see Working with Parameter Store in the *AWS Systems Manager User Guide*.

Example 3: To display the value of a parameter using labels

The following `get-parameter` example lists the value for the specified single parameter with a specified label.

```
aws ssm get-parameter \
    --name "MyParameter:label"
```

Output:

```
{
    "Parameter": {
        "Name": "MyParameter",
        "Type": "String",
        "Value": "parameter version 2",
        "Version": 2,
        "Selector": ":label",
        "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
        "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
        "DataType": "text"
    }
}
```

For more information, see Working with parameter labels in the *AWS Systems Manager User Guide*.

Example 4: To display the value of a parameter using versions

The following `get-parameter` example lists the value for the specified single parameter version.

```
aws ssm get-parameter \
    --name "MyParameter:2"
```

Output:

```
{
    "Parameter": {
        "Name": "MyParameter",
```

```
        "Type": "String",
        "Value": "parameter version 2",
        "Version": 2,
        "Selector": ":2",
        "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
        "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
        "DataType": "text"
    }
}
```

For more information, see Working with parameter labels in the *AWS Systems Manager User Guide*.

- For API details, see GetParameter in *AWS CLI Command Reference*.

Rust

SDK for Rust

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error>
{
        let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
            self.inner
                .get_parameters_by_path()
                .path(path)
                .into_paginator()
                .send(),
        )
        .flat_map(|item| item.parameters.unwrap_or_default())
        .collect()
        .await;
        // Fail on the first error
        let params = maybe_params
            .into_iter()
            .collect::<Result<Vec<Parameter>, _>>()?;
        Ok(params)
```

```
    }
```

- For API details, see GetParameter in *AWS SDK for Rust API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetParameterHistory` with a CLI

The following code examples show how to use `GetParameterHistory`.

CLI

AWS CLI

To get a value history for a parameter

The following `get-parameter-history` example lists the history of changes for the specified parameter, including its value.

```
aws ssm get-parameter-history \
    --name "MyStringParameter"
```

Output:

```
{
    "Parameters": [
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "LastModifiedDate": 1582154711.976,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is the first version of my String parameter",
            "Value": "Veni",
            "Version": 1,
            "Labels": [],
            "Tier": "Standard",
            "Policies": []
        },
        {
```

```
            "Name": "MyStringParameter",
            "Type": "String",
            "LastModifiedDate": 1582156093.471,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is the second version of my String parameter",
            "Value": "Vidi",
            "Version": 2,
            "Labels": [],
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "LastModifiedDate": 1582156117.545,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is the third version of my String parameter",
            "Value": "Vici",
            "Version": 3,
            "Labels": [],
            "Tier": "Standard",
            "Policies": []
        }
    ]
}
```

For more information, see Working with parameter versions in the *AWS Systems Manager User Guide*.

- For API details, see GetParameterHistory in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists the value history for a parameter.

```
Get-SSMParameterHistory -Name "Welcome"
```

Output:

```
Description      :
```

```
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- For API details, see GetParameterHistory in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists the value history for a parameter.

```
Get-SSMParameterHistory -Name "Welcome"
```

Output:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- For API details, see GetParameterHistory in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetParameters` with a CLI

The following code examples show how to use `GetParameters`.

CLI

AWS CLI

Example 1: To list the values for a parameter

The following `get-parameters` example lists the values for the three specified parameters.

```
aws ssm get-parameters \
    --names "MyStringParameter" "MyStringListParameter" "MyInvalidParameterName"
```

Output:

```
{
    "Parameters": [
        {
            "Name": "MyStringListParameter",
            "Type": "StringList",
            "Value": "alpha,beta,gamma",
            "Version": 1,
            "LastModifiedDate": 1582154764.222,
            "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/
MyStringListParameter"
            "DataType": "text"
        },
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "Value": "Vici",
            "Version": 3,
            "LastModifiedDate": 1582156117.545,
            "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/
MyStringParameter"
            "DataType": "text"
        }
    ],
    "InvalidParameters": [
        "MyInvalidParameterName"
    ]
}
```

For more information, see Working with Parameter Store in the *AWS Systems Manager User Guide*.

Example 2: To list names and values of multiple parameters using the ``--query`` option

The following `get-parameters` example lists the names and values for the specified parameters.

```
aws ssm get-parameters \
    --names MyStringParameter MyStringListParameter \
    --query "Parameters[*].{Name:Name,Value:Value}"
```

Output:

```
[
    {
        "Name": "MyStringListParameter",
        "Value": "alpha,beta,gamma"
    },
    {
        "Name": "MyStringParameter",
        "Value": "Vidi"
    }
]
```

For more information, see Working with Parameter Store in the *AWS Systems Manager User Guide*.

Example 3: To display the value of a parameter using labels

The following `get-parameter` example lists the value for the specified single parameter with a specified label.

```
aws ssm get-parameter \
    --name "MyParameter:label"
```

Output:

```
{
    "Parameters": [
        {
            "Name": "MyLabelParameter",
            "Type": "String",
            "Value": "parameter by label",
            "Version": 1,
```

```
            "Selector": ":label",
            "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
            "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
            "DataType": "text"
        },
        {
            "Name": "MyVersionParameter",
            "Type": "String",
            "Value": "parameter by version",
            "Version": 2,
            "Selector": ":2",
            "LastModifiedDate": "2021-03-24T16:20:28.236000-07:00",
            "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/unlabel-param",
            "DataType": "text"
        }
    ],
    "InvalidParameters": []
}
```

For more information, see Working with parameter labels in the *AWS Systems Manager User Guide*.

- For API details, see GetParameters in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists the values for a parameter.

```
Get-SSMParameterValue -Name "Welcome"
```

Output:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

Example 2: This example lists the details of the value.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Output:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- For API details, see GetParameters in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists the values for a parameter.

```
Get-SSMParameterValue -Name "Welcome"
```

Output:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

Example 2: This example lists the details of the value.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Output:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- For API details, see GetParameters in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetPatchBaseline` with a CLI

The following code examples show how to use `GetPatchBaseline`.

CLI

AWS CLI

To display a patch baseline

The following `get-patch-baseline` example retrieves the details for the specified patch baseline.

```
aws ssm get-patch-baseline \
    --baseline-id "pb-0123456789abcdef0"
```

Output:

```
{
    "BaselineId": "pb-0123456789abcdef0",
    "Name": "WindowsPatching",
    "OperatingSystem": "WINDOWS",
    "GlobalFilters": {
        "PatchFilters": []
    },
    "ApprovalRules": {
        "PatchRules": [
            {
                "PatchFilterGroup": {
                    "PatchFilters": [
                        {
                            "Key": "PRODUCT",
                            "Values": [
                                "WindowsServer2016"
                            ]
                        }
                    ]
                },
                "ComplianceLevel": "CRITICAL",
                "ApproveAfterDays": 0,
                "EnableNonSecurity": false
            }
        ]
    },
    "ApprovedPatches": [],
    "ApprovedPatchesComplianceLevel": "UNSPECIFIED",
    "ApprovedPatchesEnableNonSecurity": false,
```

```
    "RejectedPatches": [],
    "RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
    "PatchGroups": [
        "QA",
        "DEV"
    ],
    "CreatedDate": 1550244180.465,
    "ModifiedDate": 1550244180.465,
    "Description": "Patches for Windows Servers",
    "Sources": []
}
```

For more information, see About Patch Baselines in the *AWS Systems Manager User Guide*.

- For API details, see GetPatchBaseline in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example displays the details for a patch baseline.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Output:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:02:19 PM
Name            : Production-Baseline
PatchGroups     : {}
RejectedPatches : {}
```

- For API details, see GetPatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example displays the details for a patch baseline.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Output:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:02:19 PM
Name            : Production-Baseline
PatchGroups     : {}
RejectedPatches : {}
```

- For API details, see GetPatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetPatchBaselineForPatchGroup` with a CLI

The following code examples show how to use `GetPatchBaselineForPatchGroup`.

CLI

AWS CLI

To display the patch baseline for a patch group

The following `get-patch-baseline-for-patch-group` example retrieves details about the patch baseline for the specified patch group.

```
aws ssm get-patch-baseline-for-patch-group \
    --patch-group "DEV"
```

Output:

```
{
```

```
    "PatchGroup": "DEV",
    "BaselineId": "pb-0123456789abcdef0",
    "OperatingSystem": "WINDOWS"
}
```

For more information, see Create a Patch Group <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html>__ and Add a Patch Group to a Patch Baseline in the *AWS Systems Manager User Guide*.

- For API details, see GetPatchBaselineForPatchGroup in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example displays the patch baseline for a patch group.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Output:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- For API details, see GetPatchBaselineForPatchGroup in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example displays the patch baseline for a patch group.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Output:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- For API details, see GetPatchBaselineForPatchGroup in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListAssociationVersions` with a CLI

The following code examples show how to use `ListAssociationVersions`.

CLI

AWS CLI

To list all versions of an association for a specific association ID

The following `list-association-versions` example lists all versions of the specified associations.

```
aws ssm list-association-versions \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Output:

```
{
"AssociationVersions": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "CreatedDate": 1550505536.726,
            "Name": "AWS-UpdateSSMAgent",
            "Parameters": {
                "allowDowngrade": [
                    "false"
                ],
                "version": [
                    ""
                ]
            },
            "Targets": [
                {
```

```
                    "Key": "InstanceIds",
                    "Values": [
                        "i-1234567890abcdef0"
                    ]
                }
            ],
            "ScheduleExpression": "cron(0 00 12 ? * SUN *)",
            "AssociationName": "UpdateSSMAgent"
        }
    ]
}
```

For more information, see Working with associations in Systems Manager in the *AWS Systems Manager User Guide*.

- For API details, see ListAssociationVersions in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example retrieves all versions of the association provided.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Output:

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    :
AssociationVersion : 2
ComplianceSeverity :
CreatedDate        : 3/12/2019 9:21:01 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression :
Targets            : {InstanceIds}

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
```

```
AssociationName    : test-case-1234567890
AssociationVersion : 1
ComplianceSeverity :
CreatedDate        : 3/2/2019 8:53:29 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression : rate(30minutes)
Targets            : {InstanceIds}
```

- For API details, see ListAssociationVersions in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example retrieves all versions of the association provided.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Output:

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    :
AssociationVersion : 2
ComplianceSeverity :
CreatedDate        : 3/12/2019 9:21:01 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression :
Targets            : {InstanceIds}

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    : test-case-1234567890
AssociationVersion : 1
ComplianceSeverity :
CreatedDate        : 3/2/2019 8:53:29 AM
```

```
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression : rate(30minutes)
Targets            : {InstanceIds}
```

- For API details, see ListAssociationVersions in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListAssociations` with a CLI

The following code examples show how to use `ListAssociations`.

CLI

AWS CLI

Example 1: To list your associations for a specific instance

The following list-associations example lists all associations with the AssociationName, UpdateSSMAgent.

```
aws ssm list-associations /
    --association-filter-list "key=AssociationName,value=UpdateSSMAgent"
```

Output:

```
{
    "Associations": [
        {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-1234567890abcdef0",
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
```

```
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-016648b75dd622dab"
                    ]
                }
            ],
            "Overview": {
                "Status": "Pending",
                "DetailedStatus": "Associated",
                "AssociationStatusAggregatedCount": {
                    "Pending": 1
                }
            },
            "ScheduleExpression": "cron(0 00 12 ? * SUN *)",
            "AssociationName": "UpdateSSMAgent"
        }
    ]
}
```

For more information, see Working with associations in Systems Manager in the *Systems Manager User Guide*.

Example 2: To list your associations for a specific document

The following list-associations example lists all associations for the specified document.

```
aws ssm list-associations /
    --association-filter-list "key=Name,value=AWS-UpdateSSMAgent"
```

Output:

```
{
    "Associations": [
        {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-1234567890abcdef0",
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "Targets": [
                {
                    "Key": "InstanceIds",
```

```
                    "Values": [
                        "i-1234567890abcdef0"
                    ]
                }
            ],
            "LastExecutionDate": 1550505828.548,
            "Overview": {
                "Status": "Success",
                "DetailedStatus": "Success",
                "AssociationStatusAggregatedCount": {
                    "Success": 1
                }
            },
            "ScheduleExpression": "cron(0 00 12 ? * SUN *)",
            "AssociationName": "UpdateSSMAgent"
        },
    {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-9876543210abcdef0",
            "AssociationId": "fbc07ef7-b985-4684-b82b-0123456789ab",
            "AssociationVersion": "1",
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-9876543210abcdef0"
                    ]
                }
            ],
            "LastExecutionDate": 1550507531.0,
            "Overview": {
                "Status": "Success",
                "AssociationStatusAggregatedCount": {
                    "Success": 1
                }
            }
        }
    ]
}
```

For more information, see Working with associations in Systems Manager in the *Systems Manager User Guide*.

- For API details, see ListAssociations in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all the associations for an instance. The syntax used by this example requires PowerShell version 3 or later.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Output:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Example 2: This example lists all associations for a configuration document. The syntax used by this example requires PowerShell version 3 or later.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

Output:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Example 3: With PowerShell version 2, you must use New-Object to create each filter.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
```

```
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Output:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

- For API details, see ListAssociations in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all the associations for an instance. The syntax used by this example requires PowerShell version 3 or later.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Output:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Example 2: This example lists all associations for a configuration document. The syntax used by this example requires PowerShell version 3 or later.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
```

```
Get-SSMAssociationList -AssociationFilterList $filter2
```

Output:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Example 3: With PowerShell version 2, you must use New-Object to create each filter.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Output:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

- For API details, see ListAssociations in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListCommandInvocations` with an AWS SDK or CLI

The following code examples show how to use `ListCommandInvocations`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

CLI

AWS CLI

To list the invocations of a specific command

The following `list-command-invocations` example lists all the invocations of a command.

```
aws ssm list-command-invocations \
    --command-id "ef7fdfd8-9b57-4151-a15c-db9a12345678" \
    --details
```

Output:

```
{
    "CommandInvocations": [
        {
            "CommandId": "ef7fdfd8-9b57-4151-a15c-db9a12345678",
            "InstanceId": "i-02573cafcfEXAMPLE",
            "InstanceName": "",
            "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
            "DocumentName": "AWS-UpdateSSMAgent",
            "DocumentVersion": "",
            "RequestedDateTime": 1582136283.089,
            "Status": "Success",
            "StatusDetails": "Success",
            "StandardOutputUrl": "",
            "StandardErrorUrl": "",
            "CommandPlugins": [
                {
                    "Name": "aws:updateSsmAgent",
                    "Status": "Success",
                    "StatusDetails": "Success",
                    "ResponseCode": 0,
                    "ResponseStartDateTime": 1582136283.419,
```

```
                    "ResponseFinishDateTime": 1582136283.51,
                    "Output": "Updating amazon-ssm-agent from 2.3.842.0 to latest
\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-ssm-us-
east-2/ssm-agent-manifest.json\namazon-ssm-agent 2.3.842.0 has already been
 installed, update skipped\n",
                    "StandardOutputUrl": "",
                    "StandardErrorUrl": "",
                    "OutputS3Region": "us-east-2",
                    "OutputS3BucketName": "",
                    "OutputS3KeyPrefix": ""
                }
            ],
            "ServiceRole": "",
            "NotificationConfig": {
                "NotificationArn": "",
                "NotificationEvents": [],
                "NotificationType": ""
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        },
        {
            "CommandId": "ef7fdfd8-9b57-4151-a15c-db9a12345678",
            "InstanceId": "i-0471e04240EXAMPLE",
            "InstanceName": "",
            "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
            "DocumentName": "AWS-UpdateSSMAgent",
            "DocumentVersion": "",
            "RequestedDateTime": 1582136283.02,
            "Status": "Success",
            "StatusDetails": "Success",
            "StandardOutputUrl": "",
            "StandardErrorUrl": "",
            "CommandPlugins": [
                {
                    "Name": "aws:updateSsmAgent",
                    "Status": "Success",
                    "StatusDetails": "Success",
                    "ResponseCode": 0,
                    "ResponseStartDateTime": 1582136283.812,
                    "ResponseFinishDateTime": 1582136295.031,
```

```
                    "Output": "Updating amazon-ssm-agent from 2.3.672.0 to
 latest\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-
ssm-us-east-2/ssm-agent-manifest.json\nSuccessfully downloaded https://s3.us-
east-2.amazonaws.com/amazon-ssm-us-east-2/amazon-ssm-agent-updater/2.3.842.0/
amazon-ssm-agent-updater-snap-amd64.tar.gz\nSuccessfully downloaded https://
s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/amazon-ssm-agent/2.3.672.0/
amazon-ssm-agent-snap-amd64.tar.gz\nSuccessfully downloaded https://s3.us-
east-2.amazonaws.com/amazon-ssm-us-east-2/amazon-ssm-agent/2.3.842.0/amazon-ssm-
agent-snap-amd64.tar.gz\nInitiating amazon-ssm-agent update to 2.3.842.0\namazon-
ssm-agent updated successfully to 2.3.842.0",
                    "StandardOutputUrl": "",
                    "StandardErrorUrl": "",
                    "OutputS3Region": "us-east-2",
                    "OutputS3BucketName": "",
                    "OutputS3KeyPrefix": "8bee3135-398c-4d31-99b6-e42d2EXAMPLE/
i-0471e04240EXAMPLE/awsupdateSsmAgent"
                }
            ],
            "ServiceRole": "",
            "NotificationConfig": {
                "NotificationArn": "",
                "NotificationEvents": [],
                "NotificationType": ""
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        }
    ]
}
```

For more information, see Understanding Command Statuses in the *AWS Systems Manager User Guide*.

- For API details, see ListCommandInvocations in *AWS CLI Command Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { paginateListCommandInvocations, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * List SSM command invocations on an instance.
 * @param {{ instanceId: string }}
 */
export const main = async ({ instanceId }) => {
  const client = new SSMClient({});
  try {
    const listCommandInvocationsPaginated = [];
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListCommandInvocations(
      { client },
      {
        InstanceId: instanceId,
      },
    );
    for await (const page of paginator) {
      listCommandInvocationsPaginated.push(...page.CommandInvocations);
    }
    console.log("Here is the list of command invocations:");
    console.log(listCommandInvocationsPaginated);
    return { CommandInvocations: listCommandInvocationsPaginated };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Did you provide a valid instance ID?`);
    }
    throw caught;
  }
};
```

- For API details, see ListCommandInvocations in *AWS SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all the invocations of a command.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -
Detail $true
```

Output:

```
CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins     : {aws:runShellScript}
Comment            : IP config
DocumentName       : AWS-RunShellScript
InstanceId         : i-0cb2b964d3e14fd9f
InstanceName       :
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime  : 2/22/2017 8:13:16 PM
ServiceRole        :
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
TraceOutput        :
```

Example 2: This example lists CommandPlugins for invocation of the command id e1eb2e3c-ed4c-5123-45c1-234f5612345f

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Output:

```
Name                  : aws:runPowerShellScript
Output                : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
 remainingdownload: s3://dd-aess-r-ctmer/KUMO.png to ..\..\programdata\KUMO.png
                        kumo available
```

```
OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- For API details, see ListCommandInvocations in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all the invocations of a command.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -
Detail $true
```

Output:

```
CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins     : {aws:runShellScript}
Comment            : IP config
DocumentName       : AWS-RunShellScript
InstanceId         : i-0cb2b964d3e14fd9f
InstanceName       :
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime  : 2/22/2017 8:13:16 PM
ServiceRole        :
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
TraceOutput        :
```

Example 2: This example lists CommandPlugins for invocation of the command id e1eb2e3c-ed4c-5123-45c1-234f5612345f

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Output:

```
Name                  : aws:runPowerShellScript
Output                : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
 remainingdownload: s3://dd-aess-r-ctmer/KUMO.png to ..\..\programdata\KUMO.png
                       kumo available

OutputS3BucketName    :
OutputS3KeyPrefix     :
OutputS3Region        : eu-west-1
ResponseCode          : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl      :
StandardOutputUrl     :
Status                : Success
StatusDetails         : Success
```

- For API details, see ListCommandInvocations in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
```

```
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def list_command_invocations(self, instance_id):
        """
        Lists the commands for an instance.

        :param instance_id: The ID of the instance.
        :return: The list of commands.
        """
        try:
            paginator = self.ssm_client.get_paginator("list_command_invocations")
            command_invocations = []
            for page in paginator.paginate(InstanceId=instance_id):
                command_invocations.extend(page["CommandInvocations"])
            num_of_commands = len(command_invocations)
            print(
                f"{num_of_commands} command invocation(s) found for instance
{instance_id}."
            )

            if num_of_commands > 10:
                print("Displaying the first 10 commands:")
                num_of_commands = 10
            date_format = "%A, %d %B %Y %I:%M%p"
            for command in command_invocations[:num_of_commands]:
                print(
                    f"   The time of command invocation is
{command['RequestedDateTime'].strftime(date_format)}"
                )
        except ClientError as err:
            logger.error(
                "Couldn't list commands for %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- For API details, see ListCommandInvocations in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListCommands` with a CLI

The following code examples show how to use `ListCommands`.

CLI

AWS CLI

Example 1: To get the status of a specific command

The following `list-commands` example retrieves and displays the status of the specified command.

```
aws ssm list-commands \
    --command-id "0831e1a8-a1ac-4257-a1fd-c831bEXAMPLE"
```

Example 2: To get the status of commands requested after a specific date

The following `list-commands` example retrieves the details of commands requested after the specified date.

```
aws ssm list-commands \
    --filter "key=InvokedAfter,value=2020-02-01T00:00:00Z"
```

Example 3: To list all commands requested in an AWS account

The following `list-commands` example lists all commands requested by users in the current AWS account and Region.

```
aws ssm list-commands
```

Output:


```
{
    "Commands": [
        {
            "CommandId": "8bee3135-398c-4d31-99b6-e42d2EXAMPLE",
            "DocumentName": "AWS-UpdateSSMAgent",
            "DocumentVersion": "",
            "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
            "ExpiresAfter": "2020-02-19T11:28:02.500000-08:00",
            "Parameters": {},
            "InstanceIds": [
                "i-028ea792daEXAMPLE",
                "i-02feef8c46EXAMPLE",
                "i-038613f3f0EXAMPLE",
                "i-03a530a2d4EXAMPLE",
                "i-083b678d37EXAMPLE",
                "i-0dee81debaEXAMPLE"
            ],
            "Targets": [],
            "RequestedDateTime": "2020-02-19T10:18:02.500000-08:00",
            "Status": "Success",
            "StatusDetails": "Success",
            "OutputS3BucketName": "",
            "OutputS3KeyPrefix": "",
            "MaxConcurrency": "50",
            "MaxErrors": "100%",
            "TargetCount": 6,
            "CompletedCount": 6,
            "ErrorCount": 0,
            "DeliveryTimedOutCount": 0,
            "ServiceRole": "",
            "NotificationConfig": {
                "NotificationArn": "",
                "NotificationEvents": [],
                "NotificationType": ""
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        }
        {
```

```
            "CommandId": "e9ade581-c03d-476b-9b07-26667EXAMPLE",
            "DocumentName": "AWS-FindWindowsUpdates",
            "DocumentVersion": "1",
            "Comment": "",
            "ExpiresAfter": "2020-01-24T12:37:31.874000-08:00",
            "Parameters": {
                "KbArticleIds": [
                    ""
                ],
                "UpdateLevel": [
                    "All"
                ]
            },
            "InstanceIds": [],
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-00ec29b21eEXAMPLE",
                        "i-09911ddd90EXAMPLE"
                    ]
                }
            ],
            "RequestedDateTime": "2020-01-24T11:27:31.874000-08:00",
            "Status": "Success",
            "StatusDetails": "Success",
            "OutputS3BucketName": "my-us-east-2-bucket",
            "OutputS3KeyPrefix": "my-rc-output",
            "MaxConcurrency": "50",
            "MaxErrors": "0",
            "TargetCount": 2,
            "CompletedCount": 2,
            "ErrorCount": 0,
            "DeliveryTimedOutCount": 0,
            "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "NotificationConfig": {
                "NotificationArn": "arn:aws:sns:us-east-2:111222333444:my-us-
east-2-notification-arn",
                "NotificationEvents": [
                    "All"
                ],
                "NotificationType": "Invocation"
            },
```

```
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        }
        {
            "CommandId": "d539b6c3-70e8-4853-80e5-0ce4fEXAMPLE",
            "DocumentName": "AWS-RunPatchBaseline",
            "DocumentVersion": "1",
            "Comment": "",
            "ExpiresAfter": "2020-01-24T12:21:04.350000-08:00",
            "Parameters": {
                "InstallOverrideList": [
                    ""
                ],
                "Operation": [
                    "Install"
                ],
                "RebootOption": [
                    "RebootIfNeeded"
                ],
                "SnapshotId": [
                    ""
                ]
            },
            "InstanceIds": [],
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-00ec29b21eEXAMPLE",
                        "i-09911ddd90EXAMPLE"
                    ]
                }
            ],
            "RequestedDateTime": "2020-01-24T11:11:04.350000-08:00",
            "Status": "Success",
            "StatusDetails": "Success",
            "OutputS3BucketName": "my-us-east-2-bucket",
            "OutputS3KeyPrefix": "my-rc-output",
            "MaxConcurrency": "50",
            "MaxErrors": "0",
            "TargetCount": 2,
            "CompletedCount": 2,
```

```
            "ErrorCount": 0,
            "DeliveryTimedOutCount": 0,
            "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "NotificationConfig": {
                "NotificationArn": "arn:aws:sns:us-east-2:111222333444:my-us-
east-2-notification-arn",
                "NotificationEvents": [
                    "All"
                ],
                "NotificationType": "Invocation"
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        }
    ]
}
```

For more information, see Running Commands Using Systems Manager Run Command in the *AWS Systems Manager User Guide*.

- For API details, see ListCommands in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all commands requested.

```
Get-SSMCommand
```

Output:

```
CommandId          : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment            : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
CompletedCount     : 1
DocumentName       : AWS-RefreshAssociation
ErrorCount         : 0
ExpiresAfter       : 2/24/2017 3:19:08 AM
InstanceIds        : {i-0cb2b964d3e14fd9f}
```

```
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[associationIds,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 2/24/2017 3:18:08 AM
ServiceRole        :
Status             : Success
StatusDetails      : Success
TargetCount        : 1
Targets            : {}
```

Example 2: This example gets the status of a specific command.

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Example 3: This example retrieves all SSM commands invoked after 2019-04-01T00:00:00Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} |
 Select-Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -
Property RequestedDateTime -Descending
```

Output:

```
CommandId                            DocumentName               Status
 RequestedDateTime
---------                            ------------               ------
 -----------------
edb1b23e-456a-7adb-aef8-90e-012ac34f AWS-RunPowerShellScript    Cancelled
 4/16/2019 5:45:23 AM
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 AWS-ConfigureAWSPackage    Success
 4/6/2019 9:19:42 AM
12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed
 4/2/2019 4:13:07 AM
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks        Failed
 4/1/2019 2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript    Success
 4/1/2019 1:05:55 PM
```

```
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks        Failed
 4/1/2019 11:13:09 AM
```

- For API details, see ListCommands in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all commands requested.

```
Get-SSMCommand
```

Output:

```
CommandId          : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment            : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
CompletedCount     : 1
DocumentName       : AWS-RefreshAssociation
ErrorCount         : 0
ExpiresAfter       : 2/24/2017 3:19:08 AM
InstanceIds        : {i-0cb2b964d3e14fd9f}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[associationIds,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 2/24/2017 3:18:08 AM
ServiceRole        :
Status             : Success
StatusDetails      : Success
TargetCount        : 1
Targets            : {}
```

Example 2: This example gets the status of a specific command.

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Example 3: This example retrieves all SSM commands invoked after 2019-04-01T00:00:00Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} |
 Select-Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -
Property RequestedDateTime -Descending
```

Output:

```
CommandId                            DocumentName               Status
 RequestedDateTime
---------                            ------------               ------
 -----------------
edb1b23e-456a-7adb-aef8-90e-012ac34f AWS-RunPowerShellScript    Cancelled
 4/16/2019 5:45:23 AM
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 AWS-ConfigureAWSPackage    Success
 4/6/2019 9:19:42 AM
12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed
 4/2/2019 4:13:07 AM
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks        Failed
 4/1/2019 2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript    Success
 4/1/2019 1:05:55 PM
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks        Failed
 4/1/2019 11:13:09 AM
```

- For API details, see ListCommands in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListComplianceItems` with a CLI

The following code examples show how to use `ListComplianceItems`.

CLI

AWS CLI

To list compliance items for a specific instance

This example lists all compliance items for the specified instance.

Command:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-
types "ManagedInstance"
```

Output:

```
{
  "ComplianceItems": [
      {
          "ComplianceType": "Association",
          "ResourceType": "ManagedInstance",
          "ResourceId": "i-1234567890abcdef0",
          "Id": "8dfe3659-4309-493a-8755-0123456789ab",
          "Title": "",
          "Status": "COMPLIANT",
          "Severity": "UNSPECIFIED",
          "ExecutionSummary": {
              "ExecutionTime": 1550408470.0
          },
          "Details": {
              "DocumentName": "AWS-GatherSoftwareInventory",
              "DocumentVersion": "1"
          }
      },
      {
          "ComplianceType": "Association",
          "ResourceType": "ManagedInstance",
          "ResourceId": "i-1234567890abcdef0",
          "Id": "e4c2ed6d-516f-41aa-aa2a-0123456789ab",
          "Title": "",
          "Status": "COMPLIANT",
          "Severity": "UNSPECIFIED",
          "ExecutionSummary": {
              "ExecutionTime": 1550508475.0
          },
          "Details": {
              "DocumentName": "AWS-UpdateSSMAgent",
              "DocumentVersion": "1"
          }
      },
              ...
  ],
  "NextToken": "--token string truncated--"
```

```
}
```

To list compliance items for a specific instance and association ID

This example lists all compliance items for the specified instance and association ID.

Command:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-
types "ManagedInstance" --
filters "Key=ComplianceType,Values=Association,Type=EQUAL" "Key=Id,Values=e4c2ed6d-516f-4
aa2a-0123456789ab,Type=EQUAL"
```

To list compliance items for a instance after a specific date and time

This example lists all compliance items for an instance after the specified date and time.

Command:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-
types "ManagedInstance" --
filters "Key=ExecutionTime,Values=2019-02-18T16:00:00Z,Type=GREATER_THAN"
```

- For API details, see ListComplianceItems in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists compliance items list for the given resource id and type, filtering compliance-type being 'Association'

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Output:

```
ComplianceType   : Association
Details          : {[DocumentName, AWS-GatherSoftwareInventory],
 [DocumentVersion, 1]}
ExecutionSummary :
 Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
```

```
Id                : 123a45a1-c234-1234-1245-67891236db4e
ResourceId        : i-1a2caf345f67d0dc2
ResourceType      : ManagedInstance
Severity          : UNSPECIFIED
Status            : COMPLIANT
Title             :
```

- For API details, see ListComplianceItems in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists compliance items list for the given resource id and type, filtering compliance-type being 'Association'

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Output:

```
ComplianceType   : Association
Details          : {[DocumentName, AWS-GatherSoftwareInventory],
 [DocumentVersion, 1]}
ExecutionSummary :
 Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
Id               : 123a45a1-c234-1234-1245-67891236db4e
ResourceId       : i-1a2caf345f67d0dc2
ResourceType     : ManagedInstance
Severity         : UNSPECIFIED
Status           : COMPLIANT
Title            :
```

- For API details, see ListComplianceItems in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListComplianceSummaries` with a CLI

The following code examples show how to use `ListComplianceSummaries`.

CLI

AWS CLI

To list compliance summaries for all compliance types

This example lists compliance summaries for all compliance types in your account.

Command:

```
aws ssm list-compliance-summaries
```

Output:

```
{
  "ComplianceSummaryItems": [
      {
          "ComplianceType": "Association",
          "CompliantSummary": {
              "CompliantCount": 2,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 2
              }
          },
          "NonCompliantSummary": {
              "NonCompliantCount": 0,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 0
              }
          }
      },
      {
          "ComplianceType": "Patch",
```

```
            "CompliantSummary": {
                "CompliantCount": 1,
                "SeveritySummary": {
                    "CriticalCount": 0,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 1
                }
            },
            "NonCompliantSummary": {
                "NonCompliantCount": 1,
                "SeveritySummary": {
                    "CriticalCount": 1,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 0
                }
            }
        },
                ...
    ],
    "NextToken": "eyJOZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}
```

To list compliance summaries for a specific compliance type

This example lists the compliance summary for the Patch compliance type.

Command:

```
aws ssm list-compliance-summaries --
filters "Key=ComplianceType,Values=Patch,Type=EQUAL"
```

- For API details, see ListComplianceSummaries in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example returns a summary count of compliant and non-compliant resources for all compliance types.

```
Get-SSMComplianceSummaryList
```

Output:

```
ComplianceType CompliantSummary
 NonCompliantSummary
-------------- ---------------
 -------------------
FleetTotal     Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Association    Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec  Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch          Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- For API details, see ListComplianceSummaries in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example returns a summary count of compliant and non-compliant resources for all compliance types.

```
Get-SSMComplianceSummaryList
```

Output:

```
ComplianceType CompliantSummary
 NonCompliantSummary
-------------- ---------------
 -------------------
FleetTotal     Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

```
Association    Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec  Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch          Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- For API details, see ListComplianceSummaries in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListDocumentVersions` with a CLI

The following code examples show how to use `ListDocumentVersions`.

CLI

AWS CLI

To list document versions

The following `list-document-versions` example lists all versions for a Systems Manager document.

```
aws ssm list-document-versions \
    --name "Example"
```

Output:

```
{
    "DocumentVersions": [
        {
            "Name": "Example",
            "DocumentVersion": "1",
            "CreatedDate": 1583257938.266,
            "IsDefaultVersion": true,
            "DocumentFormat": "YAML",
            "Status": "Active"
```

```
        }
    ]
}
```

For more information, see Sending Commands that Use the Document Version Parameter in the *AWS Systems Manager User Guide*.

- For API details, see ListDocumentVersions in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all the versions for a document.

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

Output:

```
CreatedDate       : 6/1/2021 5:19:10 PM
DocumentFormat    : JSON
DocumentVersion   : 1
IsDefaultVersion  : True
Name              : AWS-UpdateSSMAgent
Status            : Active
```

- For API details, see ListDocumentVersions in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all the versions for a document.

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

Output:

```
CreatedDate       : 6/1/2021 5:19:10 PM
DocumentFormat    : JSON
DocumentVersion   : 1
IsDefaultVersion  : True
Name              : AWS-UpdateSSMAgent
```

```
Status              : Active
```

- For API details, see ListDocumentVersions in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListDocuments` with a CLI

The following code examples show how to use `ListDocuments`.

CLI

AWS CLI

Example 1: To list documents

The following `list-documents` example lists documents owned by the requesting account tagged with the custom tag.

```
aws ssm list-documents \
    --filters Key=Owner,Values=Self Key=tag:DocUse,Values=Testing
```

Output:

```
{
    "DocumentIdentifiers": [
        {
            "Name": "Example",
            "Owner": "29884EXAMPLE",
            "PlatformTypes": [
                "Windows",
                "Linux"
            ],
            "DocumentVersion": "1",
            "DocumentType": "Automation",
            "SchemaVersion": "0.3",
            "DocumentFormat": "YAML",
            "Tags": [
                {
```

```
                    "Key": "DocUse",
                    "Value": "Testing"
                }
            ]
        }
    ]
}
```

For more information, see AWS Systems Manager Documents in the *AWS Systems Manager User Guide*.

Example 2: To list shared documents

The following `list-documents` example lists shared documents, including private shared documents not owned by AWS.

```
aws ssm list-documents \
    --filters Key=Name,Values=sharedDocNamePrefix  Key=Owner,Values=Private
```

Output:

```
{
    "DocumentIdentifiers": [
        {
            "Name": "Example",
            "Owner": "12345EXAMPLE",
            "PlatformTypes": [
                "Windows",
                "Linux"
            ],
            "DocumentVersion": "1",
            "DocumentType": "Command",
            "SchemaVersion": "0.3",
            "DocumentFormat": "YAML",
            "Tags": []
        }
    ]
}
```

For more information, see AWS Systems Manager Documents in the *AWS Systems Manager User Guide*.

- For API details, see ListDocuments in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Lists all the configuration documents in your account.

```
Get-SSMDocumentList
```

Output:

```
DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ApplyPatchBaseline
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureAWSPackage
Owner           : Amazon
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 2.0

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureCloudWatch
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2
...
```

Example 2: This example retrieves all automation documents with name matchingi 'Platform'

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"}
 | Where-Object Name -Match "Platform"
```

Output:

```
DocumentFormat  : JSON
DocumentType    : Automation
```

```
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- For API details, see ListDocuments in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Lists all the configuration documents in your account.

```
Get-SSMDocumentList
```

Output:

```
DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ApplyPatchBaseline
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureAWSPackage
Owner           : Amazon
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 2.0

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureCloudWatch
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2
...
```

Example 2: This example retrieves all automation documents with name matchingi 'Platform'

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"}
 | Where-Object Name -Match "Platform"
```

Output:

```
DocumentFormat  : JSON
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- For API details, see ListDocuments in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListInventoryEntries` with a CLI

The following code examples show how to use `ListInventoryEntries`.

CLI

AWS CLI

Example 1: To view specific inventory type entries for an instance

This following `list-inventory-entries` example lists the inventory entries for the AWS:Application inventory type on a specific instance.

```
aws ssm list-inventory-entries \
    --instance-id "i-1234567890abcdef0" \
    --type-name "AWS:Application"
```

Output:

```
{
  "TypeName": "AWS:Application",
  "InstanceId": "i-1234567890abcdef0",
  "SchemaVersion": "1.1",
  "CaptureTime": "2019-02-15T12:17:55Z",
  "Entries": [
    {
      "Architecture": "i386",
      "Name": "Amazon SSM Agent",
      "PackageId": "{88a60be2-89a1-4df8-812a-80863c2a2b68}",
      "Publisher": "Amazon Web Services",
      "Version": "2.3.274.0"
    },
    {
      "Architecture": "x86_64",
      "InstalledTime": "2018-05-03T13:42:34Z",
      "Name": "AmazonCloudWatchAgent",
      "Publisher": "",
      "Version": "1.200442.0"
    }
  ]
}
```

Example 2: To view custom inventory entries assigned to an instance

The following `list-inventory-entries` example lists a custom inventory entry assigned to an instance.

```
aws ssm list-inventory-entries \
    --instance-id "i-1234567890abcdef0" \
    --type-name "Custom:RackInfo"
```

Output:

```
{
  "TypeName": "Custom:RackInfo",
  "InstanceId": "i-1234567890abcdef0",
  "SchemaVersion": "1.0",
  "CaptureTime": "2021-05-22T10:01:01Z",
  "Entries": [
    {
      "RackLocation": "Bay B/Row C/Rack D/Shelf E"
```

```
    }
  ]
}
```

- For API details, see ListInventoryEntries in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists all the custom inventory entries for an instance.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Output:

```
CaptureTime   : 2016-08-22T10:01:01Z
Entries       :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
InstanceId    : i-0cb2b964d3e14fd9f
NextToken     :
SchemaVersion : 1.0
TypeName      : Custom:RackInfo
```

Example 2: This example lists the details.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Output:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- For API details, see ListInventoryEntries in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists all the custom inventory entries for an instance.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Output:

```
CaptureTime   : 2016-08-22T10:01:01Z
Entries       :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
InstanceId    : i-0cb2b964d3e14fd9f
NextToken     :
SchemaVersion : 1.0
TypeName      : Custom:RackInfo
```

Example 2: This example lists the details.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Output:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- For API details, see ListInventoryEntries in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListResourceComplianceSummaries` with a CLI

The following code examples show how to use `ListResourceComplianceSummaries`.

CLI

AWS CLI

To list resource-level compliance summary counts

This example lists resource-level compliance summary counts.

Command:

```
aws ssm list-resource-compliance-summaries
```

Output:

```
{
  "ResourceComplianceSummaryItems": [
      {
          "ComplianceType": "Association",
          "ResourceType": "ManagedInstance",
          "ResourceId": "i-1234567890abcdef0",
          "Status": "COMPLIANT",
          "OverallSeverity": "UNSPECIFIED",
          "ExecutionSummary": {
              "ExecutionTime": 1550509273.0
          },
          "CompliantSummary": {
              "CompliantCount": 2,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 2
              }
          },
          "NonCompliantSummary": {
              "NonCompliantCount": 0,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 0
              }
          }
      },
      {
          "ComplianceType": "Patch",
          "ResourceType": "ManagedInstance",
```

```
            "ResourceId": "i-9876543210abcdef0",
            "Status": "COMPLIANT",
            "OverallSeverity": "UNSPECIFIED",
            "ExecutionSummary": {
                "ExecutionTime": 1550248550.0,
                "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
                "ExecutionType": "Command"
            },
            "CompliantSummary": {
                "CompliantCount": 397,
                "SeveritySummary": {
                    "CriticalCount": 0,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 397
                }
            },
            "NonCompliantSummary": {
                "NonCompliantCount": 0,
                "SeveritySummary": {
                    "CriticalCount": 0,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 0
                }
            }
        }
    ],
    "NextToken": "--token string truncated--"
}
```

To list resource-level compliance summaries for a specific compliance type

This example lists resource-level compliance summaries for the Patch compliance type.

Command:

```
aws ssm list-resource-compliance-summaries --
filters "Key=ComplianceType,Values=Patch,Type=EQUAL"
```

- For API details, see ListResourceComplianceSummaries in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example gets a resource-level summary count. The summary includes information about compliant and non-compliant statuses and detailed compliance-item severity counts for products that match "Windows10". Because the MaxResult default is 100 if the parameter is not specified, and this value is not valid, MaxResult parameter is added, and the value is set to 50.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- For API details, see ListResourceComplianceSummaries in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example gets a resource-level summary count. The summary includes information about compliant and non-compliant statuses and detailed compliance-item severity counts for products that match "Windows10". Because the MaxResult default is 100 if the parameter is not specified, and this value is not valid, MaxResult parameter is added, and the value is set to 50.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- For API details, see ListResourceComplianceSummaries in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListTagsForResource` with a CLI

The following code examples show how to use `ListTagsForResource`.

CLI

AWS CLI

To list the tags applied to a patch baseline

The following `list-tags-for-resource` example lists the tags for a patch baseline.

```
aws ssm list-tags-for-resource \
    --resource-type "PatchBaseline" \
    --resource-id "pb-0123456789abcdef0"
```

Output:

```
{
    "TagList": [
        {
            "Key": "Environment",
            "Value": "Production"
        },
        {
            "Key": "Region",
            "Value": "EMEA"
        }
    ]
}
```

For more information, see Tagging AWS Resources in the *AWS General Reference*.

- For API details, see ListTagsForResource in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example lists the tags for a maintenance window.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Output:

```
Key   Value
---   -----
Stack Production
```

- For API details, see ListTagsForResource in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example lists the tags for a maintenance window.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Output:

```
Key   Value
---   -----
Stack Production
```

- For API details, see ListTagsForResource in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ModifyDocumentPermission` with a CLI

The following code examples show how to use `ModifyDocumentPermission`.

CLI

AWS CLI

To modify document permissions

The following `modify-document-permission` example shares a Systems Manager document publicly.

```
aws ssm modify-document-permission \
    --name "Example" \
    --permission-type "Share" \
    --account-ids-to-add "All"
```

This command produces no output.

For more information, see Share a Systems Manager Document in the *AWS Systems Manager User Guide*.

- For API details, see ModifyDocumentPermission in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example adds "share" permissions to all accounts for a document. There is no output if the command succeeds.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Example 2: This example adds "share" permissions to a specific account for a document. There is no output if the command succeeds.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- For API details, see ModifyDocumentPermission in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example adds "share" permissions to all accounts for a document. There is no output if the command succeeds.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Example 2: This example adds "share" permissions to a specific account for a document. There is no output if the command succeeds.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- For API details, see ModifyDocumentPermission in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `PutComplianceItems` with a CLI

The following code examples show how to use `PutComplianceItems`.

CLI

AWS CLI

To register a compliance type and compliance details to a designated instance

This example registers the compliance type `Custom:AVCheck` to the specified managed instance. There is no output if the command succeeds.

Command:

```
aws ssm put-compliance-items --resource-id "i-1234567890abcdef0" --
resource-type "ManagedInstance" --compliance-type "Custom:AVCheck"
 --execution-summary "ExecutionTime=2019-02-18T16:00:00Z" --
items "Id=Version2.0,Title=ScanHost,Severity=CRITICAL,Status=COMPLIANT"
```

- For API details, see PutComplianceItems in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example writes a custom compliance item for the given managed instance

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()
$item.Id = "07Jun2019-3"
$item.Severity="LOW"
$item.Status="COMPLIANT"
$item.Title="Fin-test-1 - custom"
Write-SSMComplianceItem -ResourceId mi-012dcb3ecea45b678 -ComplianceType
 Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- For API details, see PutComplianceItems in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example writes a custom compliance item for the given managed instance

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()
$item.Id = "07Jun2019-3"
$item.Severity="LOW"
$item.Status="COMPLIANT"
$item.Title="Fin-test-1 - custom"
Write-SSMComplianceItem -ResourceId mi-012dcb3ecea45b678 -ComplianceType
 Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- For API details, see PutComplianceItems in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `PutInventory` with a CLI

The following code examples show how to use `PutInventory`.

CLI

AWS CLI

To assign customer metadata to an instance

This example assigns rack location information to an instance. There is no output if the command succeeds.

Command (Linux):

```
aws ssm put-inventory --instance-id "i-016648b75dd622dab" --items
 '[{"TypeName": "Custom:RackInfo","SchemaVersion": "1.0","CaptureTime":
 "2019-01-22T10:01:01Z","Content":[{"RackLocation": "Bay B/Row C/Rack D/Shelf
 E"}]}]'
```

Command (Windows):

```
aws ssm put-inventory --instance-id "i-016648b75dd622dab" --
items "TypeName=Custom:RackInfo,SchemaVersion=1.0,CaptureTime=2019-01-22T10:01:01Z,Conten
 B/Row C/Rack D/Shelf F'}]"
```

- For API details, see PutInventory in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example assigns rack location information to an instance. There is no output if the command succeeds.

```
$data = New-Object
 "System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")

$items = New-Object
 "System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
 System.String]]"
```

```
$items.Add($data)

$customInventoryItem = New-Object
 Amazon.SimpleSystemsManagement.Model.InventoryItem
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)

Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- For API details, see PutInventory in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example assigns rack location information to an instance. There is no output if the command succeeds.

```
$data = New-Object
 "System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")

$items = New-Object
 "System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
 System.String]]"
$items.Add($data)

$customInventoryItem = New-Object
 Amazon.SimpleSystemsManagement.Model.InventoryItem
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)

Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- For API details, see PutInventory in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `PutParameter` with an AWS SDK or CLI

The following code examples show how to use `PutParameter`.

CLI

AWS CLI

Example 1: To change a parameter value

The following `put-parameter` example changes the value of the specified parameter.

```
aws ssm put-parameter \
    --name "MyStringParameter" \
    --type "String" \
    --value "Vici" \
    --overwrite
```

Output:

```
{
    "Version": 2,
    "Tier": "Standard"
}
```

For more information, see Create a Systems Manager parameter (AWS CLI), Managing parameter tiers, and Working with parameter policies in the *AWS Systems Manager User Guide*.

Example 2: To create an advanced parameter

The following `put-parameter` example creates an advanced parameter.

```
aws ssm put-parameter \
    --name "MyAdvancedParameter" \
    --description "This is an advanced parameter" \
    --value "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
 eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
```

```
veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat [truncated]" \
    --type "String" \
    --tier Advanced
```

Output:

```
{
    "Version": 1,
    "Tier": "Advanced"
}
```

For more information, see Create a Systems Manager parameter (AWS CLI), Managing parameter tiers, and Working with parameter policies in the *AWS Systems Manager User Guide*.

Example 3: To convert a standard parameter to an advanced parameter

The following `put-parameter` example converts an existing standard parameter into an advanced parameter.

```
aws ssm put-parameter \
    --name "MyConvertedParameter" \
    --value "abc123" \
    --type "String" \
    --tier Advanced \
    --overwrite
```

Output:

```
{
    "Version": 2,
    "Tier": "Advanced"
}
```

For more information, see Create a Systems Manager parameter (AWS CLI), Managing parameter tiers, and Working with parameter policies in the *AWS Systems Manager User Guide*.

Example 4: To create a parameter with a policy attached

The following `put-parameter` example creates an advanced parameter with a parameter policy attached.

```
aws ssm put-parameter \
    --name "/Finance/Payroll/q2accesskey" \
    --value "P@sSwW)rd" \
    --type "SecureString" \
    --tier Advanced \
    --policies "[{\"Type\":\"Expiration\",\"Version\":\"1.0\",\"Attributes\":
{\"Timestamp\":\"2020-06-30T00:00:00.000Z\"}},{\"Type\":\"ExpirationNotification
\",\"Version\":\"1.0\",\"Attributes\":{\"Before\":\"5\",\"Unit\":\"Days\"}},
{\"Type\":\"NoChangeNotification\",\"Version\":\"1.0\",\"Attributes\":{\"After\":
\"60\",\"Unit\":\"Days\"}}]"
```

Output:

```
{
    "Version": 1,
    "Tier": "Advanced"
}
```

For more information, see Create a Systems Manager parameter (AWS CLI), Managing parameter tiers, and Working with parameter policies in the *AWS Systems Manager User Guide*.

Example 5: To add a policy to an existing parameter

The following `put-parameter` example attaches a policy to an existing advanced parameter.

```
aws ssm put-parameter \
    --name "/Finance/Payroll/q2accesskey" \
    --value "N3wP@sSwW)rd" \
    --type "SecureString" \
    --tier Advanced \
    --policies "[{\"Type\":\"Expiration\",\"Version\":\"1.0\",\"Attributes\":
{\"Timestamp\":\"2020-06-30T00:00:00.000Z\"}},{\"Type\":\"ExpirationNotification
\",\"Version\":\"1.0\",\"Attributes\":{\"Before\":\"5\",\"Unit\":\"Days\"}},
{\"Type\":\"NoChangeNotification\",\"Version\":\"1.0\",\"Attributes\":{\"After\":
\"60\",\"Unit\":\"Days\"}}]"
    --overwrite
```

Output:

```
{
    "Version": 2,
    "Tier": "Advanced"
}
```

For more information, see Create a Systems Manager parameter (AWS CLI), Managing parameter tiers, and Working with parameter policies in the *AWS Systems Manager User Guide*.

- For API details, see PutParameter in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                    paraValue - The value of the parameter.
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
        String paraValue = args[1];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        putParaValue(ssmClient, paraName, paraValue);
        ssmClient.close();
    }

    public static void putParaValue(SsmClient ssmClient, String paraName, String
 value) {
        try {
            PutParameterRequest parameterRequest = PutParameterRequest.builder()
                    .name(paraName)
                    .type(ParameterType.STRING)
                    .value(value)
                    .build();

            ssmClient.putParameter(parameterRequest);
            System.out.println("The parameter was successfully added.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see PutParameter in *AWS SDK for Java 2.x API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example creates a parameter. There is no output if the command succeeds.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Example 2: This example changes a parameter. There is no output if the command succeeds.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- For API details, see PutParameter in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example creates a parameter. There is no output if the command succeeds.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Example 2: This example changes a parameter. There is no output if the command succeeds.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- For API details, see PutParameter in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Rust

SDK for Rust

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
async fn make_parameter(
    client: &Client,
    name: &str,
    value: &str,
    description: &str,
) -> Result<(), Error> {
    let resp = client
        .put_parameter()
        .overwrite(true)
        .r#type(ParameterType::String)
        .name(name)
        .value(value)
        .description(description)
        .send()
        .await?;

    println!("Success! Parameter now has version: {}", resp.version());

    Ok(())
}
```

- For API details, see PutParameter in *AWS SDK for Rust API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `RegisterDefaultPatchBaseline` with a CLI

The following code examples show how to use `RegisterDefaultPatchBaseline`.

CLI

AWS CLI

To set the default patch baseline

The following `register-default-patch-baseline` example registers the specified custom patch baseline as the default patch baseline for the operating system type that it supports.

```
aws ssm register-default-patch-baseline \
    --baseline-id "pb-abc123cf9bEXAMPLE"
```

Output:

```
{
    "BaselineId":"pb-abc123cf9bEXAMPLE"
}
```

The following `register-default-patch-baseline` example registers the default patch baseline provided by AWS for CentOS as the default patch baseline.

```
aws ssm register-default-patch-baseline \
    --baseline-id "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-0574b43a65ea646ed"
```

Output:

```
{
    "BaselineId":"pb-abc123cf9bEXAMPLE"
}
```

For more information, see About Predefined and Custom Patch Baselines in the *AWS Systems Manager User Guide*.

- For API details, see RegisterDefaultPatchBaseline in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example registers a patch baseline as the default patch baseline.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Output:

```
pb-03da896ca3b68b639
```

- For API details, see RegisterDefaultPatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example registers a patch baseline as the default patch baseline.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Output:

```
pb-03da896ca3b68b639
```

- For API details, see RegisterDefaultPatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `RegisterPatchBaselineForPatchGroup` with a CLI

The following code examples show how to use `RegisterPatchBaselineForPatchGroup`.

CLI

AWS CLI

To register a patch baseline for a patch group

The following `register-patch-baseline-for-patch-group` example registers a patch baseline for a patch group.

```
aws ssm register-patch-baseline-for-patch-group \
    --baseline-id "pb-045f10b4f382baeda" \
    --patch-group "Production"
```

Output:

```
{
    "BaselineId": "pb-045f10b4f382baeda",
```

```
    "PatchGroup": "Production"
}
```

For more information, see Create a Patch Group <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html>__ and Add a Patch Group to a Patch Baseline in the *AWS Systems Manager User Guide*.

- For API details, see RegisterPatchBaselineForPatchGroup in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example registers a patch baseline for a patch group.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Output:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- For API details, see RegisterPatchBaselineForPatchGroup in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example registers a patch baseline for a patch group.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Output:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- For API details, see RegisterPatchBaselineForPatchGroup in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `RegisterTargetWithMaintenanceWindow` with a CLI

The following code examples show how to use `RegisterTargetWithMaintenanceWindow`.

CLI

AWS CLI

Example 1: To register a single target with a maintenance window

The following `register-target-with-maintenance-window` example registers an instance with a maintenance window.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --target "Key=InstanceIds,Values=i-0000293ffd8c57862" \
    --owner-information "Single instance" \
    --resource-type "INSTANCE"
```

Output:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Example 2: To register multiple targets with a maintenance window using instance IDs

The following `register-target-with-maintenance-window` example registers two instances with a maintenance window by specifying their instance IDs.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --target "Key=InstanceIds,Values=i-0000293ffd8c57862,i-0cb2b964d3e14fd9f" \
    --owner-information "Two instances in a list" \
    --resource-type "INSTANCE"
```

Output:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Example 3: To register targets with a maintenance window using resource tags

The following `register-target-with-maintenance-window` example registers instances with a maintenance window by specifying resource tags that have been applied to the instances.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-06cf17cbefcb4bf4f" \
    --targets "Key=tag:Environment,Values=Prod" "Key=Role,Values=Web" \
    --owner-information "Production Web Servers" \
    --resource-type "INSTANCE"
```

Output:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Example 4: To register targets using a group of tag keys

The following `register-target-with-maintenance-window` example register instances that all have one or more tag keys assigned to them, regardless of their key values.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-0c50858d01EXAMPLE" \
    --resource-type "INSTANCE" \
    --target "Key=tag-key,Values=Name,Instance-Type,CostCenter"
```

Output:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Example 5: To register targets using a resource group name

The following `register-target-with-maintenance-window` example register a specified resource group, regardless of the type of resources it contains.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-0c50858d01EXAMPLE" \
    --resource-type "RESOURCE_GROUP" \
    --target "Key=resource-groups:Name,Values=MyResourceGroup"
```

Output:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

For more information, see Register a Target Instance with the Maintenance Window (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see RegisterTargetWithMaintenanceWindow in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example registers an instance with a maintenance window.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Output:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Example 2: This example registers multiple instances with a maintenance window.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Output:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Example 3: This example registers an instance with a maintenance window using EC2 tags.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Output:

```
2994977e-aefb-4a71-beac-df620352f184
```

- For API details, see RegisterTargetWithMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example registers an instance with a maintenance window.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Output:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Example 2: This example registers multiple instances with a maintenance window.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Output:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Example 3: This example registers an instance with a maintenance window using EC2 tags.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Output:

```
2994977e-aefb-4a71-beac-df620352f184
```

- For API details, see RegisterTargetWithMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `RegisterTaskWithMaintenanceWindow` with a CLI

The following code examples show how to use `RegisterTaskWithMaintenanceWindow`.

CLI

AWS CLI

Example 1: To register an Automation task with a maintenance window

The following `register-task-with-maintenance-window` example registers an Automation task with a maintenance window that is targeted at an instance.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-082dcd7649EXAMPLE" \
    --targets Key=InstanceIds,Values=i-1234520122EXAMPLE \
    --task-arn AWS-RestartEC2Instance \
    --service-role-arn arn:aws:iam::111222333444:role/SSM --task-type AUTOMATION
 \
    --task-invocation-parameters "{\"Automation\":{\"DocumentVersion\":\"\$LATEST
\",\"Parameters\":{\"InstanceId\":[\"{{RESOURCE_ID}}\"]}}}" \
    --priority 0 \
    --max-concurrency 1 \
```

```
    --max-errors 1 \
    --name "AutomationExample" \
    --description "Restarting EC2 Instance for maintenance"
```

Output:

```
{
    "WindowTaskId":"11144444-5555-6666-7777-88888888"
}
```

For more information, see Register a Task with the Maintenance Window (AWS CLI) in the *AWS Systems Manager User Guide*.

Example 2: To register a Lambda task with a Maintenance Window

The following `register-task-with-maintenance-window` example registers a Lambda task with a Maintenance Window that is targeted at an instance.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-082dcd7649dee04e4" \
    --targets Key=InstanceIds,Values=i-12344d305eEXAMPLE \
    --task-arn arn:aws:lambda:us-east-1:111222333444:function:SSMTestLAMBDA \
    --service-role-arn arn:aws:iam::111222333444:role/SSM \
    --task-type LAMBDA \
    --task-invocation-parameters '{"Lambda":{"Payload":"{\"InstanceId\":
\"{{RESOURCE_ID}}\",\"targetType\":\"{{TARGET_TYPE}}\"}","Qualifier":"$LATEST"}}'
 \
    --priority 0 \
    --max-concurrency 10 \
    --max-errors 5 \
    --name "Lambda_Example" \
    --description "My Lambda Example"
```

Output:

```
{
    "WindowTaskId":"22244444-5555-6666-7777-88888888"
}
```

For more information, see Register a Task with the Maintenance Window (AWS CLI) in the *AWS Systems Manager User Guide*.

Example 3: To register a Run Command task with a maintenance window

The following `register-task-with-maintenance-window` example registers a Run Command task with a maintenance window that is targeted at an instance.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-082dcd7649dee04e4" \
    --targets "Key=InstanceIds,Values=i-12344d305eEXAMPLE" \
    --service-role-arn "arn:aws:iam::111222333444:role/SSM" \
    --task-type "RUN_COMMAND" \
    --name "SSMInstallPowerShellModule" \
    --task-arn "AWS-InstallPowerShellModule" \
    --task-invocation-parameters "{\"RunCommand\":{\"Comment\":\"\",
\"OutputS3BucketName\":\"runcommandlogs\",\"Parameters\":{\"commands\":[\"Get-
Module -ListAvailable\"],\"executionTimeout\":[\"3600\"],\"source\":[\"https:\/
\/gallery.technet.microsoft.com\/EZOut-33ae0fb7\/file\/110351\/1\/EZOut.zip\"],
\"workingDirectory\":[\"\\\\\"]},\"TimeoutSeconds\":600}}" \
    --max-concurrency 1 \
    --max-errors 1 \
    --priority 10
```

Output:

```
{
    "WindowTaskId":"33344444-5555-6666-7777-88888888"
}
```

For more information, see Register a Task with the Maintenance Window (AWS CLI) in the *AWS Systems Manager User Guide*.

Example 4: To register a Step Functions task with a maintenance window

The following `register-task-with-maintenance-window` example registers a Step Functions task with a maintenance window that is targeted at an instance.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-1234d787d6EXAMPLE" \
    --targets Key=WindowTargetIds,Values=12347414-69c3-49f8-95b8-ed2dcEXAMPLE \
    --task-arn arn:aws:states:us-
east-1:111222333444:stateMachine:SSMTestStateMachine \
    --service-role-arn arn:aws:iam::111222333444:role/MaintenanceWindows \
```

```
    --task-type STEP_FUNCTIONS \
    --task-invocation-parameters '{"StepFunctions":{"Input":"{\"InstanceId\":
\"{{RESOURCE_ID}}\"}"}}' \
    --priority 0 \
    --max-concurrency 10 \
    --max-errors 5 \
    --name "Step_Functions_Example" \
    --description "My Step Functions Example"
```

Output:

```
{
    "WindowTaskId":"44444444-5555-6666-7777-88888888"
}
```

For more information, see Register a Task with the Maintenance Window (AWS CLI) in the *AWS Systems Manager User Guide*.

Example 5: To register a task using a maintenance windows target ID

The following `register-task-with-maintenance-window` example registers a task using a maintenance window target ID. The maintenance window target ID was in the output of the `aws ssm register-target-with-maintenance-window` command. You can also retrieve it from the output of the `aws ssm describe-maintenance-window-targets` command.

```
aws ssm register-task-with-maintenance-window \
    --targets "Key=WindowTargetIds,Values=350d44e6-28cc-44e2-951f-4b2c9EXAMPLE" \
    --task-arn "AWS-RunShellScript" \
    --service-role-arn "arn:aws:iam::111222333444:role/MaintenanceWindowsRole" \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --task-type "RUN_COMMAND" \
    --task-parameters  "{\"commands\":{\"Values\":[\"df\"]}}" \
    --max-concurrency 1 \
    --max-errors 1 \
    --priority 10
```

Output:

```
{
    "WindowTaskId":"33344444-5555-6666-7777-88888888"
```

```
}
```

For more information, see Register a Task with the Maintenance Window (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see RegisterTaskWithMaintenanceWindow in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example registers a task with a maintenance window using an instance ID. The output is the Task ID.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
Priority 10 -TaskParameter $parameters
```

Output:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Example 2: This example registers a task with a maintenance window using a target ID. The output is the Task ID.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
```

```
 @{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -
TaskType "RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

Output:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Example 3: This example creates a parameter object for the run command document `AWS-RunPowerShellScript` and creates a task with given maintenance window using target ID. The return output is the task ID.

```
$parameters =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target =
 @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props
```

Output:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Example 4: This example registers an AWS Systems Manager Automation task by using a document named `Create-Snapshots`.

```
$automationParameters = @{}
$automationParameters.Add( "instanceId", @("{{ TARGET_ID }}") )
```

```
$automationParameters.Add( "AutomationAssumeRole",
 @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456`
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role"`
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots"`
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" }`
    -TaskType "AUTOMATION"`
    -Priority 4`
    -Automation_DocumentVersion '$DEFAULT' -Automation_Parameter
 $automationParameters -Name "Create-Snapshots"
```

- For API details, see RegisterTaskWithMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example registers a task with a maintenance window using an instance ID. The output is the Task ID.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
Priority 10 -TaskParameter $parameters
```

Output:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Example 2: This example registers a task with a maintenance window using a target ID. The output is the Task ID.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
```

```
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -
TaskType "RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

Output:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Example 3: This example creates a parameter object for the run command document `AWS-RunPowerShellScript` and creates a task with given maintenance window using target ID. The return output is the task ID.

```
$parameters =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target =
 @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props
```

Output:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Example 4: This example registers an AWS Systems Manager Automation task by using a document named `Create-Snapshots`.

```
$automationParameters = @{}
$automationParameters.Add( "instanceId", @("{{ TARGET_ID }}") )
$automationParameters.Add( "AutomationAssumeRole",
 @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456`
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role"`
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots"`
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" }`
    -TaskType "AUTOMATION"`
    -Priority 4`
    -Automation_DocumentVersion '$DEFAULT' -Automation_Parameter
 $automationParameters -Name "Create-Snapshots"
```

- For API details, see RegisterTaskWithMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `RemoveTagsFromResource` with a CLI

The following code examples show how to use `RemoveTagsFromResource`.

CLI

AWS CLI

To remove a tag from a patch baseline

The following `remove-tags-from-resource` example removes tags from a patch baseline.

```
aws ssm remove-tags-from-resource \
    --resource-type "PatchBaseline" \
    --resource-id "pb-0123456789abcdef0" \
```

```
    --tag-keys "Region"
```

This command produces no output.

For more information, see Tagging AWS Resources in the *AWS General Reference*.

- For API details, see RemoveTagsFromResource in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example removes a tag from a maintenance window. There is no output if the command succeeds.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- For API details, see RemoveTagsFromResource in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example removes a tag from a maintenance window. There is no output if the command succeeds.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- For API details, see RemoveTagsFromResource in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use SendCommand with an AWS SDK or CLI

The following code examples show how to use SendCommand.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

CLI

AWS CLI

Example 1: To run a command on one or more remote instances

The following send-command example runs an echo command on a target instance.

```
aws ssm send-command \
    --document-name "AWS-RunShellScript" \
    --parameters 'commands=["echo HelloWorld"]' \
    --targets "Key=instanceids,Values=i-1234567890abcdef0" \
    --comment "echo HelloWorld"
```

Output:

```
{
    "Command": {
        "CommandId": "92853adf-ba41-4cd6-9a88-142d1EXAMPLE",
        "DocumentName": "AWS-RunShellScript",
        "DocumentVersion": "",
        "Comment": "echo HelloWorld",
        "ExpiresAfter": 1550181014.717,
        "Parameters": {
            "commands": [
                "echo HelloWorld"
            ]
        },
        "InstanceIds": [
            "i-0f00f008a2dcbefe2"
        ],
        "Targets": [],
        "RequestedDateTime": 1550173814.717,
        "Status": "Pending",
        "StatusDetails": "Pending",
        "OutputS3BucketName": "",
        "OutputS3KeyPrefix": "",
        "MaxConcurrency": "50",
        "MaxErrors": "0",
        "TargetCount": 1,
        "CompletedCount": 0,
```

```
        "ErrorCount": 0,
        "DeliveryTimedOutCount": 0,
        "ServiceRole": "",
        "NotificationConfig": {
            "NotificationArn": "",
            "NotificationEvents": [],
            "NotificationType": ""
        },
        "CloudWatchOutputConfig": {
            "CloudWatchLogGroupName": "",
            "CloudWatchOutputEnabled": false
        }
    }
}
```

For more information, see Running Commands Using Systems Manager Run Command in the *AWS Systems Manager User Guide*.

Examle 2: To get IP information about an instance

The following `send-command` example retrieves the IP information about an instance.

```
aws ssm send-command \
    --instance-ids "i-1234567890abcdef0" \
    --document-name "AWS-RunShellScript" \
    --comment "IP config" \
    --parameters "commands=ifconfig"
```

See example 1 for sample output.

For more information, see Running Commands Using Systems Manager Run Command in the *AWS Systems Manager User Guide*.

Example 3: To run a command on instances with specific tags

The following `send-command` example runs a command on instances that have the tag key "ENV" and the value "Dev".

```
aws ssm send-command \
    --targets "Key=tag:ENV,Values=Dev" \
    --document-name "AWS-RunShellScript" \
    --parameters "commands=ifconfig"
```

See example 1 for sample output.

For more information, see Running Commands Using Systems Manager Run Command in the *AWS Systems Manager User Guide*.

Example 4: To run a command that sends SNS notifications

The following send-command example runs a command that sends SNS notifications for all notification events and the Command notification type.

```
aws ssm send-command \
    --instance-ids "i-1234567890abcdef0" \
    --document-name "AWS-RunShellScript" \
    --comment "IP config" \
    --parameters "commands=ifconfig" \
    --service-role-arn "arn:aws:iam::123456789012:role/SNS_Role" \
    --notification-config "NotificationArn=arn:aws:sns:us-
east-1:123456789012:SNSTopicName,NotificationEvents=All,NotificationType=Command"
```

See example 1 for sample output.

For more information, see Running Commands Using Systems Manager Run Command in the *AWS Systems Manager User Guide*.

Example 5: To run a command that outputs to S3 and CloudWatch

The following send-command example runs a command that outputs command details to an S3 bucket and to a CloudWatch Logs log group.

```
aws ssm send-command \
    --instance-ids "i-1234567890abcdef0" \
    --document-name "AWS-RunShellScript" \
    --comment "IP config" \
    --parameters "commands=ifconfig" \
    --output-s3-bucket-name "s3-bucket-name" \
    --output-s3-key-prefix "runcommand" \
    --cloud-watch-output-
config "CloudWatchOutputEnabled=true,CloudWatchLogGroupName=CWLGroupName"
```

See example 1 for sample output.

For more information, see Running Commands Using Systems Manager Run Command in the *AWS Systems Manager User Guide*.

Example 6: To run commands on multiple instances with different tags

The following send-command example runs a command on instances with two different tag keys and values.

```
aws ssm send-command \
    --document-name "AWS-RunPowerShellScript" \
    --parameters commands=["echo helloWorld"] \
    --targets Key=tag:Env,Values=Dev Key=tag:Role,Values=WebServers
```

See example 1 for sample output.

For more information, see Running Commands Using Systems Manager Run Command in the *AWS Systems Manager User Guide*.

Example 7: To target multiple instances with the same tag key

The following send-command example runs a command on instances that have the same tag key but with different values.

```
aws ssm send-command \
    --document-name "AWS-RunPowerShellScript" \
    --parameters commands=["echo helloWorld"] \
    --targets Key=tag:Env,Values=Dev,Test
```

See example 1 for sample output.

For more information, see Running Commands Using Systems Manager Run Command in the *AWS Systems Manager User Guide*.

Example 8: To run a command that uses a shared document

The following send-command example runs a shared document on a target instance.

```
aws ssm send-command \
    --document-name "arn:aws:ssm:us-east-1:123456789012:document/ExampleDocument"
 \
    --targets "Key=instanceids,Values=i-1234567890abcdef0"
```

See example 1 for sample output.

For more information, see Using shared SSM documents in the *AWS Systems Manager User Guide*.

- For API details, see SendCommand in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /**
    * Sends a SSM command to a managed node asynchronously.
    *
    * @param documentName The name of the document to use.
    * @param instanceId The ID of the instance to send the command to.
    * @return The command ID.
    * <p>
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture =
CompletableFuture.runAsync(() -> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus =
response.join().document().statusAsString();
```

```
                if (documentStatus.equals("Active")) {
                    System.out.println("The SSM document is active and ready to
use.");
                    isDocumentActive = true;
                } else {
                    System.out.println("The SSM document is not active. Status: "
+ documentStatus);
                    try {
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        });

        documentActiveFuture.join();

        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
                .documentName(documentName)
                .instanceIds(instanceId)
                .build();

        // Send the command.
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);
```

```
                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse>
invocationFuture = getAsyncClient().getCommandInvocation(invocationRequest);
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution
successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }
```

- For API details, see SendCommand in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { SendCommandCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Send an SSM command to a managed node.
 * @param {{ documentName: string }}
 */
export const main = async ({ documentName }) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new SendCommandCommand({
        DocumentName: documentName,
      }),
    );
    console.log("Command sent successfully.");
    return { Success: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Did you provide a valid document name?`);
    } else {
      throw caught;
    }
  }
};
```

- For API details, see SendCommand in *AWS SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example runs an echo command on a target instance.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Output:

```
CommandId          : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment            :
CompletedCount     : 0
DocumentName       : AWS-RunPowerShellScript
ErrorCount         : 0
ExpiresAfter       : 3/7/2017 10:48:37 PM
InstanceIds        : {}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[commands,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 3/7/2017 9:48:37 PM
ServiceRole        :
Status             : Pending
StatusDetails      : Pending
TargetCount        : 0
Targets            : {instanceids}
```

Example 2: This example shows how to run a command that accepts nested parameters.

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
 @{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32OpenSSH"}'; "commandLine"=".\Install-
Win32OpenSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- For API details, see SendCommand in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example runs an echo command on a target instance.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Output:

```
CommandId          : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment            :
CompletedCount     : 0
DocumentName       : AWS-RunPowerShellScript
ErrorCount         : 0
ExpiresAfter       : 3/7/2017 10:48:37 PM
InstanceIds        : {}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[commands,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 3/7/2017 9:48:37 PM
ServiceRole        :
Status             : Pending
StatusDetails      : Pending
TargetCount        : 0
Targets            : {instanceids}
```

Example 2: This example shows how to run a command that accepts nested parameters.

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
 @{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32OpenSSH"}'; "commandLine"=".\Install-
Win32OpenSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- For API details, see SendCommand in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def send_command(self, instance_ids):
        """
        Sends a command to one or more instances.

        :param instance_ids: The IDs of the instances to send the command to.
        :return: The ID of the command.
        """
        try:
            response = self.ssm_client.send_command(
                InstanceIds=instance_ids, DocumentName=self.name,
 TimeoutSeconds=3600
            )
            return response["Command"]["CommandId"]
        except ClientError as err:
            logger.error(
                "Couldn't send command to %s. Here's why: %s: %s",
```

```
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see SendCommand in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `StartAutomationExecution` with a CLI

The following code examples show how to use `StartAutomationExecution`.

CLI

AWS CLI

Example 1: To execute an automation document

The following `start-automation-execution` example runs an Automation document.

```
aws ssm start-automation-execution \
    --document-name "AWS-UpdateLinuxAmi" \
    --parameters "AutomationAssumeRole=arn:aws:iam::123456789012:role/
SSMAutomationRole,SourceAmiId=ami-EXAMPLE,IamInstanceProfileName=EC2InstanceRole"
```

Output:

```
{
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"
}
```

For more information, see Running an Automation Workflow Manually in the *AWS Systems Manager User Guide*.

Example 2: To run a shared automation document

The following `start-automation-execution` example runs a shared Automation document.

```
aws ssm start-automation-execution \
    --document-name "arn:aws:ssm:us-east-1:123456789012:document/ExampleDocument"
```

Output:

```
{
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"
}
```

For more information, see Using shared SSM documents in the *AWS Systems Manager User Guide*.

- For API details, see StartAutomationExecution in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example runs a document specifying an Automation role, an AMI source ID, and an Amazon EC2 instance role.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-
f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Output:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- For API details, see StartAutomationExecution in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example runs a document specifying an Automation role, an AMI source ID, and an Amazon EC2 instance role.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-
f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Output:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- For API details, see StartAutomationExecution in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `StartSession` with a CLI

The following code examples show how to use `StartSession`.

CLI

AWS CLI

Example 1: To start a Session Manager session

This `start-session` example establishes a connection with an instance for a Session Manager session. Note that this interactive command requires the Session Manager plugin to be installed on the client machine making the call.

```
aws ssm start-session \
    --target "i-1234567890abcdef0"
```

Output:

```
Starting session with SessionId: Jane-Roe-07a16060613c408b5
```

Example 2: To start a Session Manager session using SSH

This `start-session` example establishes a connection with an instance for a Session Manager session using SSH. Note that this interactive command requires the Session Manager plugin to be installed on the client machine making the call, and that the command uses the default user on the instance, such as `ec2-user` for EC2 instances for Linux.

```
ssh -i /path/my-key-pair.pem ec2-user@i-02573cafcfEXAMPLE
```

Output:

```
Starting session with SessionId: ec2-user-07a16060613c408b5
```

For more information, see Start a Session and Install the Session Manager Plugin for the AWS CLI in the *AWS Systems Manager User Guide*.

- For API details, see StartSession in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example initiates a connection to a target for a Session Manager session, enabling port forwarding.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' }
```

Output:

```
SessionId    StreamUrl
----------    ----------
random-id0    wss://ssmmessages.amazonaws.com/v1/data-channel/random-id
```

- For API details, see StartSession in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example initiates a connection to a target for a Session Manager session, enabling port forwarding. Note: You need to add Region parameter if not already set using AWSCredentials.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' } -Region 'us-west-1'
```

Output:

```
Starting session with SessionId: testUser-xi4glew849asyeryde34u4dfsdfy
Port 8080 opened for sessionId testUser-xi4glew849asyeryde34u4dfsdfy.
Waiting for connections...
```

Example 2: This example creates an interactive session with a specified instance for a Session Manager session.

```
Start-SSMSession -Target 'i-1234567890abcdef0' -Region 'us-west-1'
```

Output:

```
Starting session with SessionId : testUser-xi4glew849asyeryde34u4dfsdfy
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.



Install the latest PowerShell for new features and improvements!


PS C:\Windows\system32> whoami
ec2amaz-fnsdrwv\ec2-test-user
PS C:\Windows\system32>
```

Example 3: This example creates a session without connecting to it and returns the SessionId, StreamUrl, and TokenValue properties required to connect to the session.

```
Start-SSMSession -Target 'i-1234567890abcdef0' -Region 'us-west-1' -
DisablePluginInvocation
```

Output:

```
SessionId        : testUser-xi4glew849asyeryde34u4dfsdfy
StreamUrl        : {StreamUrl value redacted}
```

```
TokenValue       : {Token value redacted}
ContentLength    : 1207
HttpStatusCode   : OK
```

- For API details, see StartSession in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `StopAutomationExecution` with a CLI

The following code examples show how to use `StopAutomationExecution`.

CLI

AWS CLI

To stop an automation execution

The following `stop-automation-execution` example stops an Automation document.

```
aws ssm stop-automation-execution
    --automation-execution-id "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"
```

This command produces no output.

For more information, see Running an Automation Workflow Manually in the *AWS Systems Manager User Guide*.

- For API details, see StopAutomationExecution in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example stops an Automation Execution. There is no output if the command succeeds.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- For API details, see StopAutomationExecution in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example stops an Automation Execution. There is no output if the command succeeds.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- For API details, see StopAutomationExecution in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateAssociation` with a CLI

The following code examples show how to use `UpdateAssociation`.

CLI

AWS CLI

Example 1: To update a document association

The following `update-association` example updates an association with a new document version.

```
aws ssm update-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --document-version "\$LATEST"
```

Output:

```
{
    "AssociationDescription": {
        "Name": "AWS-UpdateSSMAgent",
```

```
        "AssociationVersion": "2",
        "Date": 1550508093.293,
        "LastUpdateAssociationDate": 1550508106.596,
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "DocumentVersion": "$LATEST",
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "tag:Name",
                "Values": [
                    "Linux"
                ]
            }
        ],
        "LastExecutionDate": 1550508094.879,
        "LastSuccessfulExecutionDate": 1550508094.879
    }
}
```

For more information, see Editing and creating a new version of an association in the *AWS Systems Manager User Guide*.

Example 2: To update the schedule expression of an association

The following `update-association` example updates the schedule expression for the specified association.

```
aws ssm update-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --schedule-expression "cron(0 0 0/4 1/1 * ? *)"
```

Output:

```
{
    "AssociationDescription": {
        "Name": "AWS-HelloWorld",
        "AssociationVersion": "2",
        "Date": "2021-02-08T13:54:19.203000-08:00",
        "LastUpdateAssociationDate": "2021-06-29T11:51:07.933000-07:00",
```

```
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "DocumentVersion": "$DEFAULT",
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "aws:NoOpAutomationTag",
                "Values": [
                    "AWS-NoOpAutomationTarget-Value"
                ]
            }
        ],
        "ScheduleExpression": "cron(0 0 0/4 1/1 * ? *)",
        "LastExecutionDate": "2021-06-26T19:00:48.110000-07:00",
        "ApplyOnlyAtCronInterval": false
    }
}
```

For more information, see Editing and creating a new version of an association in the *AWS Systems Manager User Guide*.

- For API details, see UpdateAssociation in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example updates an association with a new document version.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Output:

```
Name                 : AWS-UpdateSSMAgent
InstanceId           :
Date                 : 3/1/2017 6:22:21 PM
Status.Name          :
Status.Date          :
Status.Message       :
```

```
Status.AdditionalInfo :
```

- For API details, see UpdateAssociation in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example updates an association with a new document version.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Output:

```
Name                 : AWS-UpdateSSMAgent
InstanceId           :
Date                 : 3/1/2017 6:22:21 PM
Status.Name          :
Status.Date          :
Status.Message       :
Status.AdditionalInfo :
```

- For API details, see UpdateAssociation in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateAssociationStatus` with a CLI

The following code examples show how to use `UpdateAssociationStatus`.

CLI

AWS CLI

To update the association status

The following `update-association-status` example updates the association status of the association between an instance and a document.

```
aws ssm update-association-status \
    --name "AWS-UpdateSSMAgent" \
```

```
    --instance-id "i-1234567890abcdef0" \
    --association-
status "Date=1424421071.939,Name=Pending,Message=temp_status_change,AdditionalInfo=Additi
Config-Needed"
```

Output:

```
{
    "AssociationDescription": {
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-1234567890abcdef0",
        "AssociationVersion": "1",
        "Date": 1550507529.604,
        "LastUpdateAssociationDate": 1550507806.974,
        "Status": {
            "Date": 1424421071.0,
            "Name": "Pending",
            "Message": "temp_status_change",
            "AdditionalInfo": "Additional-Config-Needed"
        },
        "Overview": {
            "Status": "Success",
            "AssociationStatusAggregatedCount": {
                "Success": 1
            }
        },
        "DocumentVersion": "$DEFAULT",
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "InstanceIds",
                "Values": [
                    "i-1234567890abcdef0"
                ]
            }
        ],
        "LastExecutionDate": 1550507808.0,
        "LastSuccessfulExecutionDate": 1550507808.0
    }
}
```

For more information, see Working with associations in Systems Manager in the *AWS Systems Manager User Guide*.

- For API details, see UpdateAssociationStatus in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example updates the association status of the association between an instance and a configuration document.

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
 "i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
 -AssociationStatus_Name "Pending" -AssociationStatus_Message
 "temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

Output:

```
Name                 : AWS-UpdateSSMAgent
InstanceId           : i-0000293ffd8c57862
Date                 : 2/23/2017 6:55:22 PM
Status.Name          : Pending
Status.Date          : 2/20/2015 8:31:11 AM
Status.Message       : temporary_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- For API details, see UpdateAssociationStatus in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example updates the association status of the association between an instance and a configuration document.

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
 "i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
 -AssociationStatus_Name "Pending" -AssociationStatus_Message
 "temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

Output:

```
Name                    : AWS-UpdateSSMAgent
InstanceId              : i-0000293ffd8c57862
Date                    : 2/23/2017 6:55:22 PM
Status.Name             : Pending
Status.Date             : 2/20/2015 8:31:11 AM
Status.Message          : temporary_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- For API details, see UpdateAssociationStatus in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateDocument` with a CLI

The following code examples show how to use `UpdateDocument`.

CLI

AWS CLI

To create a new version of a document

The following `update-document` example creates a new version of a document when run on a Windows computer. The document specified by `--document` must be in JSON format. Note that `file://` must be referenced followed by the path of the content file. Because of the $ at the beginning of the `--document-version` parameter, On Windows you must surround the value with double quotes. On Linux, MacOS, or at a PowerShell prompt, you must surround the value with single quotes.

Windows version:

```
aws ssm update-document \
    --name "RunShellScript" \
    --content "file://RunShellScript.json" \
    --document-version "$LATEST"
```

Linux/Mac version:

```
aws ssm update-document \
    --name "RunShellScript" \
    --content "file://RunShellScript.json" \
    --document-version '$LATEST'
```

Output:

```
{
  "DocumentDescription": {
      "Status": "Updating",
      "Hash": "f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b",
      "Name": "RunShellScript",
      "Parameters": [
          {
              "Type": "StringList",
              "Name": "commands",
              "Description": "(Required) Specify a shell script or a command to
 run."
          }
      ],
      "DocumentType": "Command",
      "PlatformTypes": [
          "Linux"
      ],
      "DocumentVersion": "2",
      "HashType": "Sha256",
      "CreatedDate": 1487899655.152,
      "Owner": "809632081692",
      "SchemaVersion": "2.0",
      "DefaultVersion": "1",
      "LatestVersion": "2",
      "Description": "Run an updated script"
  }
}
```

- For API details, see UpdateDocument in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This creates a new version of a document with the updated contents of the json file you specify. The document must be in JSON format. You can obtain the document version with the "Get-SSMDocumentVersionList" cmdlet.

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-
Content -Raw "c:\temp\RunShellScript.json")
```

Output:

```
CreatedDate     : 3/1/2017 2:59:17 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 2
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 2
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Updating
```

- For API details, see UpdateDocument in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This creates a new version of a document with the updated contents of the json file you specify. The document must be in JSON format. You can obtain the document version with the "Get-SSMDocumentVersionList" cmdlet.

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-
Content -Raw "c:\temp\RunShellScript.json")
```

Output:

```
CreatedDate     : 3/1/2017 2:59:17 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 2
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 2
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Updating
```

- For API details, see UpdateDocument in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateDocumentDefaultVersion` with a CLI

The following code examples show how to use `UpdateDocumentDefaultVersion`.

CLI

AWS CLI

To update the default version of a document

The following `update-document-default-version` example updates the default version of a Systems Manager document.

```
aws ssm update-document-default-version \
    --name "Example" \
    --document-version "2"
```

Output:

```
{
    "Description": {
        "Name": "Example",
        "DefaultVersion": "2"
    }
}
```

For more information, see Writing SSM Document Content in the *AWS Systems Manager User Guide*.

- For API details, see UpdateDocumentDefaultVersion in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This updates the default version of a document. You can obtain the available document versions with the "Get-SSMDocumentVersionList" cmdlet.

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Output:

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- For API details, see UpdateDocumentDefaultVersion in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This updates the default version of a document. You can obtain the available document versions with the "Get-SSMDocumentVersionList" cmdlet.

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Output:

```
DefaultVersion Name
-------------- ----
```

```
2               RunShellScript
```

- For API details, see UpdateDocumentDefaultVersion in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateMaintenanceWindow` with an AWS SDK or CLI

The following code examples show how to use `UpdateMaintenanceWindow`.

CLI

AWS CLI

Example 1: To update a maintenance window

The following `update-maintenance-window` example updates the name of a maintenance window.

```
aws ssm update-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9" \
    --name "My-Renamed-MW"
```

Output:

```
{
    "Cutoff": 1,
    "Name": "My-Renamed-MW",
    "Schedule": "cron(0 16 ? * TUE *)",
    "Enabled": true,
    "AllowUnassociatedTargets": true,
    "WindowId": "mw-1a2b3c4d5e6f7g8h9",
    "Duration": 4
}
```

Example 2: To disable a maintenance window

The following `update-maintenance-window` example disables a maintenance window.

```
aws ssm update-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9" \
    --no-enabled
```

Example 3: To enable a maintenance window

The following `update-maintenance-window` example enables a maintenance window.

```
aws ssm update-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9" \
    --enabled
```

For more information, see Update a Maintenance Window (AWS CLI) in the *AWS Systems Manager User Guide*.

- For API details, see UpdateMaintenanceWindow in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
```

```
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();

        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }
```

- For API details, see UpdateMaintenanceWindow in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { UpdateMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
```

```
import { parseArgs } from "node:util";

/**
 * Update an SSM maintenance window.
 * @param {{ windowId: string, allowUnassociatedTargets?: boolean, duration?:
 number, enabled?: boolean, name?: string, schedule?: string }}
 */
export const main = async ({
  windowId,
  allowUnassociatedTargets = undefined, //Allow the maintenance window to run on
 managed nodes, even if you haven't registered those nodes as targets.
  duration = undefined, //The duration of the maintenance window in hours.
  enabled = undefined,
  name = undefined,
  schedule = undefined, //The schedule of the maintenance window in the form of a
 cron or rate expression.
}) => {
  const client = new SSMClient({});
  try {
    const { opsItemArn, opsItemId } = await client.send(
      new UpdateMaintenanceWindowCommand({
        WindowId: windowId,
        AllowUnassociatedTargets: allowUnassociatedTargets,
        Duration: duration,
        Enabled: enabled,
        Name: name,
        Schedule: schedule,
      }),
    );
    console.log("Maintenance window updated.");
    return { OpsItemArn: opsItemArn, OpsItemId: opsItemId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Are these values correct?`);
    } else {
      throw caught;
    }
  }
};
```

- For API details, see UpdateMaintenanceWindow in *AWS SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example updates the name of a maintenance window.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-
MW"
```

Output:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Example 2: This example enables a maintenance window.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Output:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Example 3: This example disables a maintenance window.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Output:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
```

```
Duration                  : 2
Enabled                   : False
Name                      : My-Renamed-MW
Schedule                  : cron(0 */30 * * * ? *)
WindowId                  : mw-03eb9db42890fb82d
```

- For API details, see UpdateMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example updates the name of a maintenance window.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-
MW"
```

Output:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Example 2: This example enables a maintenance window.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Output:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Example 3: This example disables a maintenance window.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Output:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- For API details, see UpdateMaintenanceWindow in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)
```

```
   def update(
       self, name, enabled, schedule, duration, cutoff,
allow_unassociated_targets
   ):
       """
       Update an AWS Systems Manager maintenance window.

       :param name: The name of the maintenance window.
       :param enabled: Whether the maintenance window is enabled to run on
managed nodes.
       :param schedule: The schedule of the maintenance window.
       :param duration: The duration of the maintenance window.
       :param cutoff: The cutoff time of the maintenance window.
       :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                          if you haven't registered those nodes
as targets.
       """
       try:
           self.ssm_client.update_maintenance_window(
               WindowId=self.window_id,
               Name=name,
               Enabled=enabled,
               Schedule=schedule,
               Duration=duration,
               Cutoff=cutoff,
               AllowUnassociatedTargets=allow_unassociated_targets,
           )
           self.name = name
           logger.info("Updated maintenance window %s.", self.window_id)
       except ParamValidationError as error:
           logger.error(
               "Parameter validation error when trying to update maintenance
window %s. Here's why: %s",
               self.window_id,
               error,
           )
           raise
       except ClientError as err:
           logger.error(
               "Couldn't update maintenance window %s. Here's why: %s: %s",
               self.name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see UpdateMaintenanceWindow in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateManagedInstanceRole` with a CLI

The following code examples show how to use `UpdateManagedInstanceRole`.

CLI

AWS CLI

To update the IAM role of a managed instance

The following `update-managed-instance-role` example updates the IAM instance profile of a managed instance.

```
aws ssm update-managed-instance-role \
    --instance-id "mi-08ab247cdfEXAMPLE" \
    --iam-role "ExampleRole"
```

This command produces no output.

For more information, see Step 4: Create an IAM Instance Profile for Systems Manager in the *AWS Systems Manager User Guide*.

- For API details, see UpdateManagedInstanceRole in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example updates the role of a managed instance. There is no output if the command succeeds.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- For API details, see UpdateManagedInstanceRole in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example updates the role of a managed instance. There is no output if the command succeeds.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- For API details, see UpdateManagedInstanceRole in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateOpsItem` with an AWS SDK or CLI

The following code examples show how to use `UpdateOpsItem`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Learn the basics

CLI

AWS CLI

To update an OpsItem

The following `update-ops-item` example updates the description, priority, and category for an OpsItem. In addition, the command specifies an SNS topic where the notifications are sent when this OpsItem is edited or changed.

```
aws ssm update-ops-item \
    --ops-item-id "oi-287b5EXAMPLE" \
    --description "Primary OpsItem for failover event 2020-01-01-fh398yf" \
    --priority 2 \
    --category "Security" \
    --notifications "Arn=arn:aws:sns:us-east-2:111222333444:my-us-east-2-topic"
```

Output:

```
This command produces no output.
```

For more information, see Working with OpsItems in the *AWS Systems Manager User Guide*.

- For API details, see UpdateOpsItem in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /**
     * Resolves an AWS SSM OpsItem asynchronously.
     *
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
```

```
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- For API details, see UpdateOpsItem in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import { UpdateOpsItemCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Update an SSM OpsItem.
 * @param {{ opsItemId: string, status?: OpsItemStatus }}
 */
export const main = async ({
  opsItemId,
  status = undefined, // The OpsItem status. Status can be Open, In Progress, or
 Resolved
}) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new UpdateOpsItemCommand({
        OpsItemId: opsItemId,
        Status: status,
      }),
    );
    console.log("Ops item updated.");
    return { Success: true };
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "OpsItemLimitExceededException"
    ) {
      console.warn(
        `Couldn't create ops item because you have exceeded your open OpsItem
 limit. ${caught.message}.`,
      );
    } else {
```

```
      throw caught;
    }
  }
};
```

- For API details, see UpdateOpsItem in *AWS SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def update(self, title=None, description=None, status=None):
        """
        Update an OpsItem.

        :param title: The new OpsItem title.
```

```
        :param description: The new OpsItem description.
        :param status: The new OpsItem status.
        :return:
        """
        args = dict(OpsItemId=self.id)
        if title is not None:
            args["Title"] = title
        if description is not None:
            args["Description"] = description
        if status is not None:
            args["Status"] = status
        try:
            self.ssm_client.update_ops_item(**args)
        except ClientError as err:
            logger.error(
                "Couldn't update ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- For API details, see UpdateOpsItem in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdatePatchBaseline` with a CLI

The following code examples show how to use `UpdatePatchBaseline`.

CLI

AWS CLI

Example 1: To update a patch baseline

The following `update-patch-baseline` example adds the specified two patches as rejected and one patch as approved to the specified patch baseline.

```
aws ssm update-patch-baseline \
        --baseline-id "pb-0123456789abcdef0" \
        --rejected-patches "KB2032276" "MS10-048" \
        --approved-patches "KB2124261"
```

Output:

```
{
    "BaselineId": "pb-0123456789abcdef0",
    "Name": "WindowsPatching",
    "OperatingSystem": "WINDOWS",
    "GlobalFilters": {
        "PatchFilters": []
    },
    "ApprovalRules": {
        "PatchRules": [
            {
                "PatchFilterGroup": {
                    "PatchFilters": [
                        {
                            "Key": "PRODUCT",
                            "Values": [
                                "WindowsServer2016"
                            ]
                        }
                    ]
                },
                "ComplianceLevel": "CRITICAL",
                "ApproveAfterDays": 0,
                "EnableNonSecurity": false
            }
        ]
    },
    "ApprovedPatches": [
        "KB2124261"
    ],
    "ApprovedPatchesComplianceLevel": "UNSPECIFIED",
    "ApprovedPatchesEnableNonSecurity": false,
    "RejectedPatches": [
        "KB2032276",
        "MS10-048"
    ],
    "RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
```

```
    "CreatedDate": 1550244180.465,
    "ModifiedDate": 1550244180.465,
    "Description": "Patches for Windows Servers",
    "Sources": []
}
```

Example 2: To rename a patch baseline

The following `update-patch-baseline` example renames the specified patch baseline.

```
aws ssm update-patch-baseline \
    --baseline-id "pb-0713accee01234567" \
    --name "Windows-Server-2012-R2-Important-and-Critical-Security-Updates"
```

For more information, see Update or Delete a Patch Baseline <https://docs.aws.amazon.com/systems-manager/latest/userguide/patch-baseline-update-or-delete.html>`__ in the *AWS Systems Manager User Guide*.

- For API details, see UpdatePatchBaseline in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: This example adds two patches as rejected and one patch as approved to an existing patch baseline.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Output:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {KB2124261}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:22:10 PM
Name            : Production-Baseline
RejectedPatches : {KB2032276, MS10-048}
```

- For API details, see UpdatePatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This example adds two patches as rejected and one patch as approved to an existing patch baseline.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Output:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {KB2124261}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:22:10 PM
Name            : Production-Baseline
RejectedPatches : {KB2032276, MS10-048}
```

- For API details, see UpdatePatchBaseline in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

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 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 tool in 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*.

Topics

- Monitoring tools
- Sending node logs to unified CloudWatch Logs (CloudWatch agent)

- Sending SSM Agent logs to CloudWatch Logs
- Monitoring your change request events
- Monitoring your automations
- Monitoring Run Command metrics using Amazon CloudWatch
- Logging AWS Systems Manager API calls with AWS CloudTrail
- Logging Automation action output with CloudWatch Logs
- Configuring Amazon CloudWatch Logs for Run Command
- Monitoring Systems Manager events with Amazon EventBridge
- Monitoring Systems Manager status changes using Amazon SNS notifications

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.

Sending node logs to unified 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 tool in AWS Systems Manager.

> **Note**
>
> AWS Systems Manager supports migrating from SSM Agent to the unified CloudWatch agent for collecting logs and metrics on 64-bit versions of Windows only. For information about setting up the unified 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. The SSM Agent aws:cloudWatch plugin is not supported. We recommend using only the unified CloudWatch agent for your log collection processes. For more information, see the following topics:

- Sending node logs to unified CloudWatch Logs (CloudWatch agent)
- Migrate Windows Server node log collection to the CloudWatch agent
- Collecting metrics, logs, and traces 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 Logging session activity and Enabling and disabling session logging.

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
- Manually migrating to the CloudWatch agent

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 unified CloudWatch agent for collecting logs and metrics on 64-bit versions of Windows only. For information about setting up the unified 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. If the migration isn't successful or the results aren't as expected, you can try Rolling back to log collection with SSM 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}`.

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.

> **Tip**
>
> If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

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-02573cafcfEXAMPLE*.

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.

   Tip

   If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

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 tool in 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)

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-RunPowerShellScript`.
4. 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`.

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.

> **Tip**
>
> If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

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.

   > **Tip**
   >
   > If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

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. If the migration isn't successful or the results aren't as expected, you can Rolling back to log collection with SSM Agent.

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 tool in 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 Parameter Store parameters in Systems Manager.

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\`
   `\AWS.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.

   > **Tip**
   >
   > If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

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.

   Tip

   If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

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, on-premises servers, and virtual machines (VMs) 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.

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 20.04, 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 20.04, 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 as 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 your change request events

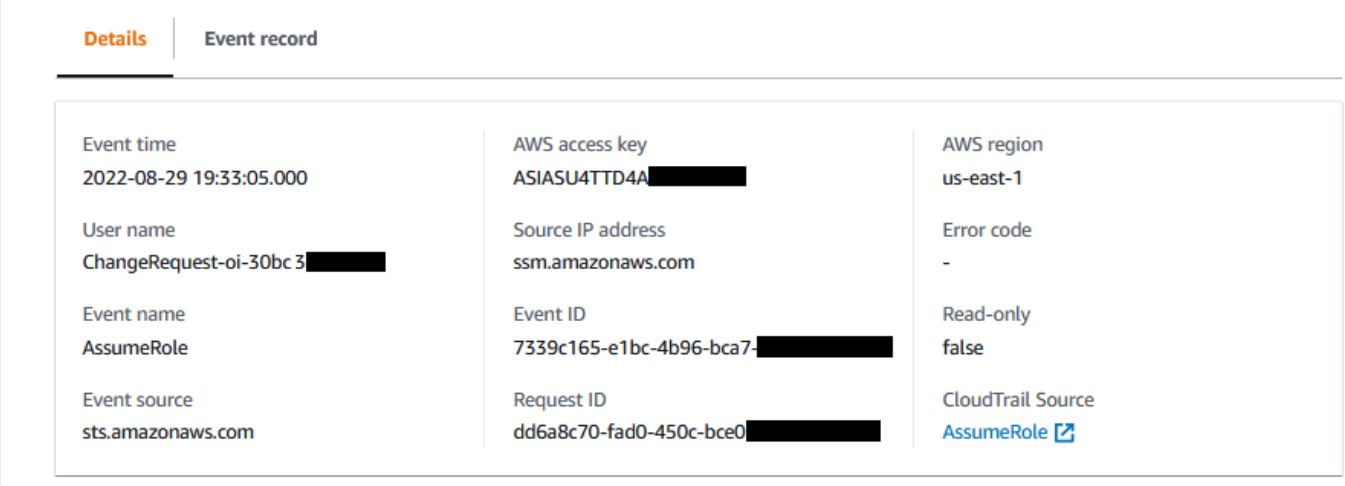
After turning on integration with AWS CloudTrail Lake and creating an event data store, you can view auditable details about the change requests that are run in your account or organization. This includes details such as the following:

- The identity of the user that initiated the change request
- The AWS Regions where the changes were made
- The source IP address for the request
- The AWS access key used for the request
- The API actions run for the change request
- The request parameters included for those actions
- The resources updated during the process

The following are samples of event details you can view for a change request after creating the event data store in AWS CloudTrail Lake.

Details

The following image shows the high-level information about a change request available on the **Details** tab. These details include information such as the time the change request operation began, the ID of the user that initiated the change request, the impacted AWS Region, and the event ID and request ID associated with the request.



Event record

The following image shows the structure of the JSON content provided by CloudTrail Lake for a change request event. This data is provided on the **Event record** tab in a change request.



> **Important**
>
> If you're using Change Manager for an organization, you can complete the following procedure while signed in to either the management account or the delegated administrator account for Change Manager.
> However, to use the delegated administrator account to complete these steps, the same delegated administrator account must be specified for both CloudTrail and Change Manager.
> When you sign in to the management account for Change Manager, you can add or change the delegated administrator account for CloudTrail on the CloudTrail Settings page. This must be done before the delegated administrator account can create an event data store for use by the entire organization.

To turn on CloudTrail Lake event tracking from Change Manager

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

3. Choose the **Requests** tab.
4. Choose any existing change request, and then choose the **Associated events** tab.
5. Choose **Enable CloudTrail Lake**.
6. Follow the steps in Create an event data store for CloudTrail events in the *AWS CloudTrail User Guide*.

   To ensure that event data for your change requests is stored, make the following selections as you complete the procedure:

   - For **Event type**, leave the defaults **AWS events** and **CloudTrail events** selected.
   - If you're using Change Manager with an organization, select **Enable for all accounts in my organization**.
   - For **Management events**, do not clear the **Write** check box.

   Other options you choose when creating your event data store don't affect the storage of event data for your change requests.

Monitoring your automations

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 tool in 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 tool in 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.

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`.

Metric	Description
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. CloudTrail captures all API calls for Systems Manager as events. The calls captured include calls from the Systems Manager console and code calls to the Systems Manager API operations. 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, when it was made, and additional details.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or user credentials.
- Whether the request was made on behalf of an IAM Identity Center user.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

CloudTrail is active in your AWS account when you create the account and you automatically have access to the CloudTrail **Event history**. The CloudTrail **Event history** provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management

events in an AWS Region. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*. There are no CloudTrail charges for viewing the **Event history**.

For an ongoing record of events in your AWS account past 90 days, create a trail or a CloudTrail Lake event data store.

CloudTrail trails

A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. All trails created using the AWS Management Console are multi-Region. You can create a single-Region or a multi-Region trail by using the AWS CLI. Creating a multi-Region trail is recommended because you capture activity in all AWS Regions in your account. If you create a single-Region trail, you can view only the events logged in the trail's AWS Region. For more information about trails, see Creating a trail for your AWS account and Creating a trail for an organization in the *AWS CloudTrail User Guide*.

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail, however, there are Amazon S3 storage charges. For more information about CloudTrail pricing, see AWS CloudTrail Pricing. For information about Amazon S3 pricing, see Amazon S3 Pricing.

CloudTrail Lake event data stores

CloudTrail Lake lets you run SQL-based queries on your events. CloudTrail Lake converts existing events in row-based JSON format to Apache ORC format. ORC is a columnar storage format that is optimized for fast retrieval of data. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying advanced event selectors. The selectors that you apply to an event data store control which events persist and are available for you to query. For more information about CloudTrail Lake, see Working with AWS CloudTrail Lake in the *AWS CloudTrail User Guide*.

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the pricing option you want to use for the event data store. The pricing option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

Systems Manager data events in CloudTrail

Data events provide information about the resource operations performed on or in a resource (for example, creating or opening a control channel). These are also known as data plane operations. Data events are often high-volume activities. By default, CloudTrail doesn't log data events. The CloudTrail **Event history** doesn't record data events.

Additional charges apply for data events. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

You can log data events for the Systems Manager resource types by using the CloudTrail console, AWS CLI, or CloudTrail API operations. For more information about how to log data events, see Logging data events with the AWS Management Console and Logging data events with the AWS Command Line Interface in the *AWS CloudTrail User Guide*.

The following table lists the Systems Manager resource types for which you can log data events. The **Data event type (console)** column shows the value to choose from the **Data event type** list on the CloudTrail console. The **resources.type value** column shows the `resources.type` value, which you would specify when configuring advanced event selectors using the AWS CLI or CloudTrail APIs. The **Data APIs logged to CloudTrail** column shows the API calls logged to CloudTrail for the resource type.

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
Systems Manager	`AWS::SSMMessages::ControlChannel`	• `CreateControlChannel` • `OpenControlChannel` For more information about these operations, see Actions defined by Amazon Message Gateway Service in the *Service Authorization Reference*.
Systems Manager managed node	`AWS::SSM::ManagedNode`	• `RequestManagedInstanceRoleToken` – This

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
		event is generated when the AWS Systems Manager Agent (SSM Agent) running on a node managed by Systems Manager requests credentials from the Systems Manager credentia l service. For more information about the `RequestManagedInst anceRoleToken` operation , see Validating hybrid-ac tivated machines using a hardware fingerprint

You can configure advanced event selectors to filter on the `eventName`, `readOnly`, and `resources.`ARN fields to log only those events that are important to you. For more information about these fields, see AdvancedFieldSelector in the *AWS CloudTrail API Reference*.

Systems Manager management events in CloudTrail

Management events provide information about management operations that are performed on resources in your AWS account. These are also known as control plane operations. By default, CloudTrail logs management events.

Systems Manager logs all control plane operations to CloudTrail as management events. Systems Manager API operations 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).

Systems Manager event examples

An event represents a single request from any source and includes information about the requested API 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 events don't appear in any specific order.

Examples:

- Management event examples
- Data event examples

Management event examples

Example 1: `DeleteDocument`

The following example shows a CloudTrail event 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",
    "eventCategory": "Management"
}
```

Example 2: `StartConnection`

The following example shows a CloudTrail event 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-02573cafcfEXAMPLE"
    },
    "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"
}
```


Data event examples

Example 1: `CreateControlChannel`

The following example shows a CloudTrail event that demonstrates the `CreateControlChannel` operation.

```
{
  "eventVersion":"1.08",
  "userIdentity":{
    "type":"AssumedRole",
    "principalId":"AKIAI44QH8DHBEXAMPLE",
    "arn":"arn:aws:sts::123456789012:assumed-role/exampleRole",
    "accountId":"123456789012",
    "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
    "sessionContext":{
      "sessionIssuer":{
        "type":"Role",
        "principalId":"AKIAI44QH8DHBEXAMPLE",
        "arn":"arn:aws:iam::123456789012:role/exampleRole",
        "accountId":"123456789012",
        "userName":"exampleRole"
      },
      "attributes":{
        "creationDate":"2023-05-04T23:14:50Z",
        "mfaAuthenticated":"false"
      }
    }
  },
  "eventTime":"2023-05-04T23:53:55Z",
  "eventSource":"ssm.amazonaws.com",
  "eventName":"CreateControlChannel",
  "awsRegion":"us-east-1",
  "sourceIPAddress":"192.0.2.0",
  "userAgent":"example-agent",
  "requestParameters":{
    "channelId":"44295c1f-49d2-48b6-b218-96823EXAMPLE",
    "messageSchemaVersion":"1.0",
    "requestId":"54993150-0e8f-4142-aa54-3438EXAMPLE",
    "userAgent":"example-agent"
  },
  "responseElements":{
    "messageSchemaVersion":"1.0",
    "tokenValue":"Value hidden due to security reasons.",
    "url":"example-url"
  },
  "requestID":"54993150-0e8f-4142-aa54-3438EXAMPLE",
  "eventID":"a48a28de-7996-4ca1-a3a0-a51fEXAMPLE",
  "readOnly":false,
  "resources":[
```

```
    {
      "accountId":"123456789012",
      "type":"AWS::SSMMessages::ControlChannel",
      "ARN":"arn:aws:ssmmessages:us-east-1:123456789012:control-
channel/44295c1f-49d2-48b6-b218-96823EXAMPLE"
    }
  ],
  "eventType":"AwsApiCall",
  "managementEvent":false,
  "recipientAccountId":"123456789012",
  "eventCategory":"Data"
}
```

Example 2: `RequestManagedInstanceRoleToken`

The following example shows a CloudTrail event that demonstrates the `RequestManagedInstanceRoleToken` operation.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "123456789012:aws:ec2-instance:i-02854e4bEXAMPLE",
        "arn": "arn:aws:sts::123456789012:assumed-role/aws:ec2-instance/
i-02854e4bEXAMPLE",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "123456789012:aws:ec2-instance",
                "arn": "arn:aws:iam::123456789012:role/aws:ec2-instance",
                "accountId": "123456789012",
                "userName": "aws:ec2-instance"
            },
            "attributes": {
                "creationDate": "2023-08-27T03:34:46Z",
                "mfaAuthenticated": "false"
            },
            "ec2RoleDelivery": "2.0"
        }
    },
    "eventTime": "2023-08-27T03:37:15Z",
```

```
    "eventSource": "ssm.amazonaws.com",
    "eventName": "RequestManagedInstanceRoleToken",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Apache-HttpClient/UNAVAILABLE (Java/1.8.0_362)",
    "requestParameters": {
        "fingerprint": "i-02854e4bf85EXAMPLE"
    },
    "responseElements": null,
    "requestID": "2582cced-455b-4189-9b82-7b48EXAMPLE",
    "eventID": "7f200508-e547-4c27-982d-4da0EXAMLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::SSM::ManagedNode",
            "ARN": "arn:aws:ec2:us-east-1:123456789012:instance/i-02854e4bEXAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "123456789012",
    "eventCategory": "Data"
}
```


For information about CloudTrail record contents, see CloudTrail record contents in the *AWS CloudTrail User Guide*.

Logging Automation action output with CloudWatch Logs

Automation, a tool in 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 for Amazon owned runbooks, the user or role that runs the automation must have permissions for the following operations:

- `logs:CreateLogGroup`
- `logs:CreateLogStream`
- `logs:DescribeLogGroups`
- `logs:DescribeLogStreams`
- `logs:PutLogEvents`

For runbooks that you own, the same permissions must be added to the IAM service role (or AssumeRole) you use to run the runbook.

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.

3. 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 tool in AWS Systems Manager, you can specify where you want to send the command output. By default, Systems Manager returns only the first 24,000 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 managed node to use the AWS Identity and Access Management (IAM) managed policies `AmazonSSMManagedInstanceCore` and `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 Configure instance permissions required for Systems Manager. 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 the IAM service role required for Systems Manager in hybrid and multicloud environments.

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 \
    --instance-ids "instance ID" \
    --document-name "AWS-RunShellScript" \
    --parameters "commands=echo helloWorld" \
    --cloud-watch-output-config
 "CloudWatchOutputEnabled=true,CloudWatchLogGroupName=log group name"
```

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 just-in-time node access request status update for manual approvals.
- A failed just-in-time node access request.
- A maintenance window being turned off.
- An Automation workflow completing successfully. Automation is a tool in 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 tools:

- Just-in-time node access (Events are emitted on a best effort basis.)
- 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 and Amazon EventBridge event examples for Systems Manager.

Supported Systems Manager target types

EventBridge supports the following three Systems Manager tools 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 Sample scenarios: Systems Manager targets in Amazon EventBridge rules.

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
- Amazon EventBridge event examples for Systems Manager
- Sample scenarios: Systems Manager targets in Amazon EventBridge rules

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 **More info** 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**.
3. Choose **Create rule**.
4. 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.
5. For **Event bus**, 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 **default**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
6. For **Rule type**, choose **Rule with an event pattern**.
7. Choose **Next**.
8. For **Event source**, choose **AWS events or EventBridge partner events**.
9. In the **Event pattern** section, choose **Event pattern form**.
10. For **Event source**, choose **AWS services**.
11. For **AWS service**, choose **Systems Manager**.
12. 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 tools:
      - 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).

13. (Optional) To make the rule more specific, add filter values. For example, if you chose **State Manager** and want to limit the rule to the state of a single managed instance that is targeted by an Association, for **Specific type(s)**, choose **EC2 State Manager Instance Association State Change**.

    For complete details about supported detail types, see Reference: Amazon EventBridge event patterns and types for Systems Manager.

    Some detail types have other supported options such as status. The available options depend on the tool you selected.
14. Choose **Next**.
15. For **Target types**, choose **AWS service**.
16. For **Select a target**, choose a target such as an Amazon SNS topic or AWS Lambda function. The target is triggered when an event is received that matches the event pattern defined in the rule.
17. 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**.
18. (Optional) Choose **Add another target** to add another target for this rule.
19. Choose **Next**.

20. (Optional) Enter one or more tags for the rule. For more information, see Amazon EventBridge tags in the *Amazon EventBridge User Guide*.
21. Choose **Next**.
22. Review the details of the rule and choose **Create rule**.

More info

- Creating an EventBridge event that uses a runbook (console)
- Passing data to Automation using input transformers
- Remediating compliance issues using EventBridge
- Viewing inventory delete actions in EventBridge
- Configure EventBridge rules to create OpsItems
- Configuring EventBridge rules for parameters and parameter policies

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
- AWS Systems Manager Change Calendar Events
- AWS Systems Manager Change Manager Events
- AWS Systems Manager Compliance Events
- AWS Systems Manager Maintenance Windows Events
- AWS Systems Manager Parameter Store Events
- AWS Systems Manager OpsCenter Events
- AWS Systems Manager Run Command Events
- AWS Systems Manager State Manager Events

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": "2024-11-29T19:43:35Z",
  "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, 2024 7:43:25 PM",
    "StartTime": "Nov 29, 2024 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": "2024-11-29T19:43:35Z",
  "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, 2024 7:43:20 PM",
```

```
    "EndTime": "Nov 29, 2024 7:43:26 PM",
    "Time": 5753.0,
    "ExecutedBy": "arn:aws:iam::123456789012:user/userName"
  }
}
```

AWS Systems Manager Change Calendar Events

Use the information in this topic to plan and understand the behavior of EventBridge events for AWS Systems Manager Change Calendar.

> **Note**
>
> State changes for calendars shared from other AWS accounts are not currently supported.

Change Calendar integration with Amazon EventBridge

AWS Systems Manager Change Calendar integrates with Amazon EventBridge to notify you of calendar state changes. Be aware of the following behaviors related to the underlying scheduling architecture:

Event timing and reliability

- EventBridge delivers notifications on a best-effort basis with up to a 15-minute scheduling tolerance.
- State change events reflect overall calendar status transitions, not individual calendar events.
- When multiple calendar events occur simultaneously, EventBridge generates only one event per actual calendar state change.
- EventBridge triggers events only when the calendar's overall state transitions (for example, from CLOSED to OPEN), not for individual calendar events that don't result in a state change.
- Advisory events that don't modify the calendar state don't trigger EventBridge notifications.

Event modifications and timing considerations

- If you modify calendar events within 15 minutes of their scheduled start or end time, EventBridge might generate duplicate notifications or miss notifications.
- This behavior occurs because the scheduling system might not have sufficient time to properly update or cancel previously scheduled notifications.

- For recurring events, this behavior typically affects only the first occurrence after modification.

Adjacent and overlapping events

- When calendar events are scheduled within 5 minutes of each other, state transition events might or might not occur, depending on the actual state change.
- Creating overlapping events in certain orders might generate additional EventBridge events even when no actual state change occurs.
- To ensure predictable behavior, avoid creating or modifying calendar events close to their execution times.

Best practices

- Design your EventBridge rules and downstream automation to handle potential duplicate events.
- Implement idempotency in your automation workflows to prevent issues from duplicate notifications.
- Allow sufficient lead time (at least 15 minutes) when you create or modify calendar events.
- Test your EventBridge integrations thoroughly with your specific calendar event patterns.

Calendar OPEN

```
{
    "version": "0",
    "id": "47a3f03a-f30d-1011-ac9a-du3bdEXAMPLE",
    "detail-type": "Calendar State Change",
    "source": "aws.ssm",
    "account": "123456789012",
    "time": "2024-09-19T18:00:07Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:ssm:us-east-2:123456789012:document/MyCalendar"
    ],
    "detail": {
        "state": "OPEN",
        "atTime": "2024-09-19T18:00:07Z",
        "nextTransitionTime": "2024-10-11T18:00:07Z"
    }
}
```

Calendar CLOSED

```
{
    "version": "0",
    "id": "f30df03a-1011-ac9a-47a3-f761eEXAMPLE",
    "detail-type": "Calendar State Change",
    "source": "aws.ssm",
    "account": "123456789012",
    "time": "2024-09-17T21:40:02Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:ssm:us-east-2:123456789012:document/MyCalendar"
    ],
    "detail": {
        "state": "CLOSED",
        "atTime": "2024-08-17T21:40:00Z",
        "nextTransitionTime": "2024-09-19T18:00:07Z"
    }
}
```

AWS Systems Manager Change Manager Events

Change request status update notification - example 1

```
{
  "version": "0",
  "id": "feab80c1-a8ff-c721-b8b1-96ce70939696",
  "detail-type": "Change Request Status Update",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2024-10-24T10:51:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-west-2:123456789012:opsitem/oi-12345abcdef",
    "arn:aws:ssm:us-west-2:123456789012:document/MyRunbook1"
  ],
  "detail": {
    "change-request-id": "d0585556-80f6-4522-8dad-dada6d45b67d",
    "change-request-title": "A change request title",
    "ops-item-id": "oi-12345abcdef",
    "ops-item-created-by": "arn:aws:iam::123456789012:user/JohnDoe",
    "ops-item-created-time": "2024-10-24T10:50:33.180334Z",
    "ops-item-modified-by": "arn:aws:iam::123456789012:user/JohnDoe",
```

```
    "ops-item-modified-time": "2024-10-24T10:50:33.180340Z",
    "ops-item-status": "InProgress",
    "change-template-document-name": "MyChangeTemplate",
    "runbook-document-arn": "arn:aws:ssm:us-west-2:123456789012:document/MyRunbook1",
    "runbook-document-version": "1",
    "auto-approve": true,
    "approvers": [
      "arn:aws:iam::123456789012:user/JaneDoe"
    ]
  }
}
```

Change request status update notification - example 2

```
{
  "version": "0",
  "id": "25ce6b03-2e4e-1a2b-2a8f-6c9de8d278d2",
  "detail-type": "Change Request Status Update",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2024-10-24T10:51:52Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-west-2:123456789012:opsitem/oi-abcdef12345",
    "arn:aws:ssm:us-west-2:123456789012:document/MyRunbook1"
  ],
  "detail": {
    "change-request-id": "d0585556-80f6-4522-8dad-dada6d45b67d",
    "change-request-title": "A change request title",
    "ops-item-id": "oi-abcdef12345",
    "ops-item-created-by": "arn:aws:iam::123456789012:user/JohnDoe",
    "ops-item-created-time": "2024-10-24T10:50:33.180334Z",
    "ops-item-modified-by": "arn:aws:iam::123456789012:user/JohnDoe",
    "ops-item-modified-time": "2024-10-24T10:50:33.997163Z",
    "ops-item-status": "Rejected",
    "change-template-document-name": "MyChangeTemplate",
    "runbook-document-arn": "arn:aws:ssm:us-west-2:123456789012:document/MyRunbook1",
    "runbook-document-version": "1",
    "auto-approve": true,
    "approvers": [
      "arn:aws:iam::123456789012:user/JaneDoe"
    ]
  }
```

```
}
```

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": "2024-07-17T19:03:26Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:ssm:us-east-2:123456789012:managed-instance/i-01234567890abcdef"
  ],
  "detail": {
    "last-runtime": "2024-01-01T10:10:10Z",
    "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": "2024-07-17T19:02:31Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:ssm:us-east-2:123456789012:managed-instance/i-01234567890abcdef"
  ],
  "detail": {
    "last-runtime": "2024-01-01T10:10:10Z",
```

```
    "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": "2024-07-17T19:03:26Z",
  "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": "2024-07-17T19:02:31Z",
  "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

Valid status values include REGISTERED and DEREGISTERED.

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-0123456789ab",
   "detail-type":"Maintenance Window Target Registration Notification",
   "source":"aws.ssm",
   "account":"123456789012",
   "time":"2024-11-16T00:58:37Z",
   "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

Valid status values include the following:

- CANCELLED

- CANCELLING
- FAILED
- IN_PROGRESS
- PENDING
- SKIPPED_OVERLAPPING
- SUCCESS TIMED_OUT

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-0123456789ab",
   "detail-type":"Maintenance Window Execution State-change Notification",
   "source":"aws.ssm",
   "account":"123456789012",
   "time":"2025-06-02T14:52:18Z",
   "region":"us-east-2",
   "resources":[
      "arn:aws:ssm:us-east-2:123456789012:maintenancewindow/mw-0c50858d01EXAMPLE"
   ],
   "detail":{
      "start-time":"2025-06-02T14:48:28.039273Z",
      "end-time":"2025-06-02T14:52:18.083773Z",
      "window-id":"mw-0ed7251d3fcf6e0c2",
      "window-execution-id":"14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",
      "status":"SUCCESS"
   }
}
```

Task Execution Type

Valid status values include 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":"2025-06-02T14:52:18Z",
   "region":"us-east-2",
```

```
   "resources":[
      "arn:aws:ssm:us-east-2:123456789012:maintenancewindow/mw-0c50858d01EXAMPLE"
   ],
   "detail":{
      "start-time":"2025-06-02T14:48:28.039273Z",
      "task-execution-id":"6417e808-7f35-4d1a-843f-123456789012",
      "end-time":"2025-06-02T14:52:18.083773Z",
      "window-id":"mw-0ed7251d3fcf6e0c2",
      "window-execution-id":"14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",
      "status":"SUCCESS"
   }
}
```

Task Target Processed

Valid status values include `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":"2025-06-02T14:52:18Z",
   "region":"us-east-2",
   "resources":[
      "arn:aws:ssm:us-east-2:123456789012:maintenancewindow/mw-123456789012345678"
   ],
   "detail":{
      "start-time":"2025-06-02T14:48:28.039273Z",
      "end-time":"2025-06-02T14:52:18.083773Z",
      "window-id":"mw-0ed7251d3fcf6e0c2",
      "window-execution-id":"791b72e0-f0da-4021-8b35-f95dfEXAMPLE",
      "task-execution-id":"c9b05aba-197f-4d8d-be34-e73fbEXAMPLE",
      "window-target-id":"e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE",
      "status":"SUCCESS",
      "owner-information":"Owner"
   }
}
```

Window State Change

Valid status values include `ENABLED` and `DISABLED`.

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-0123456789ab",
   "detail-type":"Maintenance Window State-change Notification",
   "source":"aws.ssm",
   "account":"123456789012",
   "time":"2024-11-16T00:58:37Z",
   "region":"us-east-2",
   "resources":[
      "arn:aws:ssm:us-east-2:123456789012:maintenancewindow/mw-0c50858d01EXAMPLE"
   ],
   "detail":{
      "window-id":"mw-0c50858d01EXAMPLE",
      "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": "2024-05-22T16:43:48Z",
  "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": "2024-05-22T16:44:48Z",
  "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": "2024-05-22T16:45:48Z",
  "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 OpsCenter Events

OpsCenter OpsItem create notification


```
{
  "version": "0",
  "id": "aae66adc-7aac-f0c0-7854-7691e8c079b8",
  "detail-type": "OpsItem Create",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2024-10-19T02:48:11Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-west-2:123456789012:opsitem/oi-123456abcdef"
  ],
  "detail": {
    "created-by": "arn:aws:iam::123456789012:user/JohnDoe",
    "created-time": "2024-10-19T02:46:53.629361Z",
    "source": "aws.ssm",
    "status": "Open",
    "ops-item-id": "oi-123456abcdef",
    "title": "An issue title",
    "ops-item-type": "/aws/issue",
    "description": "A long description may appear here"
  }
}
```


OpsCenter OpsItem update notification


```
{
  "version": "0",
  "id": "2fb5b168-b725-41dd-a890-29311200089c",
  "detail-type": "OpsItem Update",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2024-10-19T02:48:11Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-west-2:123456789012:opsitem/oi-123456abcdef"
  ],
  "detail": {
    "created-by": "arn:aws:iam::123456789012:user/JohnDoe",
    "created-time": "2024-10-19T02:46:54.049271Z",
```

```
    "modified-by": "arn:aws:iam::123456789012:user/JohnDoe",
    "modified-time": "2024-10-19T02:46:54.337354Z",
    "source": "aws.ssm",
    "status": "Open",
    "ops-item-id": "oi-123456abcdef",
    "title": "An issue title",
    "ops-item-type": "/aws/issue",
    "description": "A long description may appear here"
  }
}
```

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": "2024-07-10T21:51:32Z",
    "region": "us-east-2",
    "resources": ["arn:aws:ec2:us-east-2:123456789012:instance/i-02573cafcfEXAMPLE"],
    "detail": {
        "command-id": "e8d3c0e4-71f7-4491-898f-c9b35bee5f3b",
        "document-name": "AWS-RunPowerShellScript",
        "expire-after": "2024-07-14T22:01:30.049Z",
        "parameters": {
            "executionTimeout": ["3600"],
            "commands": ["date"]
        },
        "requested-date-time": "2024-07-10T21:51:30.049Z",
        "status": "Success"
    }
}
```

Run Command Invocation Status-change Notification

```
{
    "version": "0",
    "id": "4780e1b8-f56b-4de5-95f2-95dbEXAMPLE",
    "detail-type": "EC2 Command Invocation Status-change Notification",
```

```
    "source": "aws.ssm",
    "account": "123456789012",
    "time": "2024-07-10T21:51:32Z",
    "region": "us-east-2",
    "resources": ["arn:aws:ec2:us-east-2:123456789012:instance/i-02573cafcfEXAMPLE"],
    "detail": {
        "command-id": "e8d3c0e4-71f7-4491-898f-c9b35bee5f3b",
        "document-name": "AWS-RunPowerShellScript",
        "instance-id": "i-02573cafcfEXAMPLE",
        "requested-date-time": "2024-07-10T21:51:30.049Z",
        "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":"2024-05-16T23:01:10Z",
   "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":"2024-02-13T17:22:54.458Z",
      "last-successful-execution-date":"2024-05-16T23:00:01Z",
      "last-execution-date":"2024-05-16T23:00:01Z",
      "last-updated-date":"2024-02-13T17:22:54.458Z",
      "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":"2024-02-23T15:23:48Z",
   "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":"34fcb7e0-9a14-4984-9989-0e04e3f60bd8",
      "instance-id":"i-02573cafcfEXAMPLE",
      "document-name":"my-custom-document",
      "document-version":"1",
      "targets":"[{\"key\":\"instanceids\",\"values\":[\"i-02573cafcfEXAMPLE\"]}]",
      "creation-date":"2024-02-23T15:23:48Z",
      "last-successful-execution-date":"2024-02-23T16:23:48Z",
      "last-execution-date":"2024-02-23T16:23:48Z",
      "status":"Success",
      "detailed-status":"",
      "error-code":"testErrorCode",
      "execution-summary":"testExecutionSummary",
      "output-url":"sampleurl",
      "instance-association-cwe-version":"1"
   }
}
```

Sample scenarios: Systems Manager targets in Amazon EventBridge rules

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 tools in AWS Systems Manager, as target actions when an EventBridge event occurs.

The following are several examples of ways you can use these tools 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 tools in 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.

Important

Note the following important information.

- Amazon Simple Notification Service FIFO topics aren't supported.
- Amazon Q Developer in chat applications isn't supported for monitoring Systems Manager with Amazon SNS. If you wish to use Amazon Q Developer in chat applications to monitor Systems Manager, you must use it with Amazon EventBridge. For information about monitoring Systems Manager using EventBridge, see Monitoring Systems Manager events with Amazon EventBridge. For information about Amazon EventBridge and Amazon Q Developer in chat applications, see Tutorial: Creating an EventBridge rule that sends notifications to Amazon Q Developer in chat applications in the *Amazon Q Developer in chat applications Administrator Guide*.

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
- Cancelled

For information about the conditions that cause a command to enter one of these statuses, see Understanding command statuses.

> **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

Field	Type	Description
documentName	String	The name of the SSM document used to run this command.
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.

Field	Type	Description
instanceIds	StringList	The nodes that were targeted by the command. **Note** 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.
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
- Task 2: Create an IAM policy for Amazon SNS notifications
- Task 3: Create an IAM role for Amazon SNS notifications
- Task 4: Configure user access
- Task 5: Attach the iam:PassRole policy to your maintenance window role

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 inititating 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 one of the following, depending on whether the Amazon SNS topic uses AWS KMS encryption:

SNS topic not encrypted

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:111122223333: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.

SNS topic encrypted

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:111122223333:sns-topic-
name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/kms-key-
id"
        }
    ]
}
```

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.

kms-key-id represents the ID of the symmetric encryption KMS key in AWS KMS to use for encrypting and decrypting the topic, in the format 1234abcd-12ab-34cd-56ef-12345EXAMPLE.

> **Note**
>
> There is a charge for using AWS KMS encryption. For more information, see Managing Amazon SNS encryption keys and costs in the *AWS Key Management Service Developer Guide*.

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. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
3. Choose the **AWS service** role type, and then choose Systems Manager.
4. Choose the Systems Manager use case. Then, choose **Next**.
5. On the **Attach permissions policies** page, select the box to the left of the name of the custom policy you created in Task 2. For example: **`my-sns-publish-permissions`**.
6. (Optional) Set a permissions boundary. This is an advanced feature that is available for service roles, but not service-linked roles.

   Expand the **Permissions boundary** section and choose **Use a permissions boundary to control the maximum role permissions**. IAM includes a list of the AWS managed and customer managed policies in your account. Select the policy to use for the permissions boundary or choose **Create policy** to open a new browser tab and create a new policy from scratch. For more information, see Creating IAM policies in the *IAM User Guide*. After you create the policy, close that tab and return to your original tab to select the policy to use for the permissions boundary.
7. Choose **Next**.

8. If possible, enter a role name or role name suffix to help you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both **PRODROLE** and **prodrole**. Because various entities might reference the role, you cannot edit the name of the role after it has been created.
9. (Optional) For **Description**, enter a description for the new role.
10. Choose **Edit** in the **Step 1: Select trusted entities** or **Step 2: Select permissions** sections to edit the use cases and permissions for the role.
11. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see Tagging IAM resources in the *IAM User Guide*.
12. Review the role and then choose **Create role**.
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 an IAM entity (user, role, or group) is assigned administrator permissions, then the user or role has access to Run Command and Maintenance Windows, tools in AWS Systems Manager.

For entities without administrator permissions, an administrator must grant the following permissions to the IAM entity:

- The `AmazonSSMFullAccess` managed policy, or a policy that provides comparable permissions.
- `iam:PassRole` permissions for the role created in Task 3: Create an IAM role for Amazon SNS notifications. For example:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/sns-role-name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ssm.amazonaws.com"
                }
            }
        }
    ]
}
```

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.
- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) 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 Setting up Maintenance Windows. If you need to create a custom service role for maintenance window tasks, see Setting up Maintenance Windows.

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 **Add permissions, Attach policies** to attach it.
6. Choose **Add permissions, Create 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 tools in 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.

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": "amzn-s3-demo-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 tool in 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**.
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.

   > **Tip**
   >
   > If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.
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-02573cafcfEXAMPLE, i-0471e04240EXAMPLE"
 --document-name "AWS-RunPowerShellScript" --parameters '{"commands":
["Get-Process"]}' --service-role "arn:aws:iam::111122223333:role/
SNS_Role" --notification-config '{"NotificationArn":"arn:aws:sns:us-
east-1:111122223333:SNSTopic","NotificationEvents":
["All"],"NotificationType":"Command"}'
```

Alternative commands

Specify parameters in the following command to target managed instances using tags.

```
aws ssm send-command --targets "Key=tag:TagName,Values=TagKey" --document-name
 "Name" --parameters '{"commands":["input"]}' --service-role "SNSRoleARN" --
notification-config '{"NotificationArn":"SNSTopicName","NotificationEvents":
["All"],"NotificationType":"Command"}'
```

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 tool in 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 Create and manage maintenance windows using the console 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.

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**.
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.

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 Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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 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": "j2l8d5b5c-mw66-tk4d-r3g9-1d4d1EXAMPLE"
}
```

6. After the next execution of your maintenance window, check your email for a message from Amazon SNS and open the email 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.

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
- Integration with other products and services

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.

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 • Working with managed nodes • AWS Systems Manager State Manager • AWS Systems Manager Run Command • AWS Systems Manager Patch Manager • AWS Systems Manager Session Manager • AWS Systems Manager Distributor • AWS Systems Manager Compliance • AWS Systems Manager Inventory
Amazon EC2 Auto Scaling	Auto Scaling 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. 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. **Learn more** Updating AMIs for Auto Scaling groups

Amazon Elastic Container Service (Amazon ECS)	Amazon ECS is a highly scalable, fast, container management service that allows you to run, stop, and manage Docker containers on a cluster. 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 tool in Systems Manager, and then referencing them in your container definition. **Learn more** • Manage container instances remotely using AWS Systems Manager • Specifying sensitive data using Systems Manager Parameter Store

AWS Lambda	Lambda 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. Systems Manager allows you to use Lambda functions within Automation runbook content by using the `aws:invokeLambdaFunction` action. To use parameters from Parameter Store in AWS Lambda functions, you can use the AWS Parameters and Secrets Lambda Extension to retrieve parameter values and cache them for future use. **Learn more** Update a golden AMI using Automation, AWS Lambda, and Parameter Store Using Parameter Store parameters in AWS Lambda functions

Internet of Things (IoT)

AWS IoT Greengrass core devices	AWS IoT Greengrass is an open source IoT edge runtime and cloud service that helps you build, deploy and manage IoT applications on your devices. Systems Manager offers native support for AWS IoT Greengrass core devices. **Learn more** Managing edge devices with Systems Manager

AWS IoT core devices	AWS IoT 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 *managed nodes* in a hybrid and multicloud environment. **Learn more** Managing nodes in hybrid and multicloud environments with Systems Manager

Storage

Amazon Simple Storage Service (Amazon S3)	Amazon S3 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. Systems Manager allows you to run remote scripts and SSM documents that are stored in Amazon S3. Distributor, a tool in AWS Systems Manager, uses Amazon S3 to store packages. You can also send output to Amazon S3 for Run Command and Session Manager, tools in AWS Systems Manager. **Learn more** • Running scripts from Amazon S3

	• Running documents from remote locations • AWS Systems Manager Distributor • Logging session data using Amazon S3 (console)

Developer Tools

AWS CodeBuild	CodeBuild 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. Parameter Store allows you to store sensitive information for your build specifications and projects. **Learn more** • Build specification reference for CodeBuild • Create a build project in AWS CodeBuild
AWS CDK	The AWS Cloud Development Kit (AWS CDK) is a framework for defining cloud infrastructure as code, with programming languages, and deploying it through AWS CloudFormation. Application Manager allows you to view your CDK constructs grouped as applications, view the application structure including the underlying resources, view alerts, investigate and remediate operational issues, and track costs in the Application Manager console.

	Learn more • Viewing overview information about an application • Viewing application resources

Security, Identity, and Compliance

AWS Identity and Access Management (IAM)	IAM 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. Systems Manager allows you to control access to services using IAM. **Learn more** • How AWS Systems Manager works with IAM • Actions, resources, and condition keys for AWS Systems Manager • Configure instance permissions required for Systems Manager
AWS Secrets Manager	Secrets Manager provides easier managemen t of secrets. Secrets can be database credentia ls, passwords, third-party API keys, and even arbitrary text. Parameter Store allows you to retrieve Secrets Manager secrets when using other AWS services that already support references to Parameter Store parameters. **Learn more**

	Referencing AWS Secrets Manager secrets from Parameter Store parameters

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 tool in 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, tools in 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

	• Understanding OpsCenter integration with AWS Security Hub • Integrating Patch Manager with AWS Security Hub

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** • AWS KMS encryption for AWS Systems Manager Parameter Store SecureString parameters • Turn on KMS key encryption of session data (console)

Management and Governance

AWS CloudFormation	AWS CloudFormation 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. 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. **Learn more** Using dynamic references to specify template values
AWS CloudTrail	CloudTrail 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. Systems Manager integrates with CloudTrai l which captures most Systems Manager API calls as events. These include API calls initiated from the Systems Manager console and calls made to the Systems Manager APIs. **Learn more** Logging AWS Systems Manager API calls with AWS CloudTrail

Amazon CloudWatch Logs	Amazon CloudWatch Logs 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. Systems Manager supports sending logs for the SSM Agent, Run Command, and Session Manager to CloudWatch Logs. **Learn more** • Sending node logs to unified CloudWatch Logs (CloudWatch agent) • Configuring Amazon CloudWatch Logs for Run Command • Logging session data using Amazon CloudWatch Logs (console)

Amazon EventBridge	EventBridge 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. Systems Manager has multiple events that are supported by EventBridge allowing you to take actions based on the content of those events. **Learn more** Monitoring Systems Manager events with Amazon EventBridge **Note** Amazon EventBridge is the preferred way to manage your events. CloudWatch Events and EventBrid ge 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 EventBrid ge User Guide*.

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
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 • 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 consolida ted billing capabilities that allow you to better meet the budgetary, security, and compliance needs of your business. Integration between Change Manager, a tool in AWS Systems Manager, with Organizat ions 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, a tool in AWS Systems Manager, and Explorer allows you to aggregate inventory and operations data (OpsData) from multiple AWS Regions and AWS accounts. Integration between Quick Setup , a tool in 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** Improve the security of EC2 instances by using VPC endpoints for Systems Manager

Analytics

Amazon Athena	Athena 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. 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 **Detailed View** page in the Systems Manager Inventory console.

	Learn more • Querying inventory data from multiple Regions and accounts • Walkthrough: Using resource data sync to aggregate inventory data
AWS Glue	AWS Glue 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. Systems Manager uses AWS Glue to crawl the Inventory data in your S3 bucket. **Learn more** Querying inventory data from multiple Regions and accounts
Amazon QuickSight	QuickSight 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. Systems Manager resource data sync sends inventory data collected from all of your managed nodes to a single S3 bucket. You can use QuickSight to query and analyze the aggregated data. **Learn more** • Creating a resource data sync for Inventory • Walkthrough: Using resource data sync to aggregate inventory data

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 • Setting up notifications or triggering actions based on Parameter Store events

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

- Referencing AWS Secrets Manager secrets from Parameter Store parameters
- AWS KMS encryption for AWS Systems Manager Parameter Store SecureString parameters
- Use Parameter Store parameters in Amazon Elastic Kubernetes Service
- Using Parameter Store parameters in AWS Lambda functions

Running scripts from Amazon S3

This section describes how to download and run scripts from Amazon Simple Storage Service (Amazon S3). The following topic includes information and terminology relating to Amazon S3. To learn more about Amazon S3, see What is 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 an 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, tools in AWS Systems Manager, and then downloads and runs the script.
- Verify that your user, role, or group has been granted the AWS Identity and Access Management (IAM) permissions needed 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**.
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.

     > **Note**
     >
     > Replace https://s3.*`aws-api-domain`* with the URL for your bucket. You can copy your bucket URL in Amazon S3 on the **Objects** tab.

     ```
     {"path":"https://s3.aws-api-domain/path to script"}
     ```

     The following is an example.

     ```
     {"path":"https://amzn-s3-demo-bucket.s3.us-east-2.amazonaws.com/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.

   > **Tip**
   >
   > If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more

information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

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 Installing or updating the latest version of the AWS CLI.

2. Run the following command. Replace each *example resource placeholder* with your own information.

   Note

   Replace https://s3.*aws-api-domain*/*script path* with the URL for your bucket. You can copy your bucket URL in Amazon S3 on the **Objects** tab.

   Linux & macOS

```
aws ssm send-command \
    --document-name "AWS-RunRemoteScript" \
    --output-s3-bucket-name "amzn-s3-demo-bucket" \
    --output-s3-key-prefix "key-prefix" \
    --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" ^
    --output-s3-bucket-name "amzn-s3-demo-bucket" ^
    --output-s3-key-prefix "key-prefix" ^
    --targets "Key=InstanceIds,Values=instance-id" ^
    --parameters "sourceType"="S3",sourceInfo='{\"path\":\"https://s3.aws-api-domain/script path\"}',"commandLine"="script name and arguments"
```

PowerShell

```
Send-SSMCommand `
    -DocumentName "AWS-RunRemoteScript" `
    -OutputS3BucketName "amzn-s3-demo-bucket" `
    -OutputS3KeyPrefix "key-prefix" `
    -Target @{Key="InstanceIds";Values=@("instance-id")} `
    -Parameter @{
        sourceType = "S3";
        sourceInfo = '{"path": "s3://bucket-name/path/to/script"}';
        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 tool in 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 tools. 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.
- Secrets Manager secrets in Parameter Store must have a prefix of `/aws/reference/secretsmanager`. The following are examples:

  ```
  /aws/reference/secretsmanager/CFCreds1
  ```

  ```
  /aws/reference/secretsmanager/myapp/db/password
  ```

- 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 Systems 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 DynamoDB client with different credentials
    final AmazonDynamoDB client = AmazonDynamoDBClient.builder()
            .withCredentials(new AWSStaticCredentialsProvider(differentAWSCreds))
            .withRegion(getParameter("region")) //Getting configuration from
 Parameter Store
            .build();
    return client;
}

/**
* Helper method to retrieve parameter value
* @param parameterName identifier of the 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!al875",
        "Version": 0,
        "SourceResult":
              "{
                   \"CreatedDate\": 1526334434.743,
                   \"Name\": \"s1-secret\",
                   \"VersionId\": \"aaabbbccc-1111-222-333-123456789\",
                   \"SecretString\": \"Fl*MEishm!al875\",
                   \"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!al875",
        "Version": 0,
        "SourceResult":
              "{
                   \"CreatedDate\": 1526334434.743,
                   \"Name\": \"s1-secret\",
                   \"VersionId\": \"11111-aaa-bbb-ccc-123456789\",
                   \"SecretString\": \"Fl*MEishm!al875\",
                   \"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!al875",
        "Version": 0,
        "SourceResult":
              "{
                   \"CreatedDate\": 1526334434.743,
                   \"Name\": \"s1-secret\",
                   \"VersionId\": \"11111-aaa-bbb-ccc-123456789\",
                   \"SecretString\": \"Fl*MEishm!al875\",
                   \"VersionStages\": [\"AWSCURRENT\"],
                   \"ARN\": \"arn:aws:secretsmanager:us-
east-2:123456789012:secret:s1-secret-E18LRP\"
               }"
        "Selector": ":AWSCURRENT"
      }
      "LastModifiedDate": 2018-05-14T21:47:14.743Z,
      "ARN": "arn:aws:secretsmanager:us-east-2:123456789012:secret:s1-secret-
                E18LRP",
}
```

AWS KMS encryption for AWS Systems Manager Parameter Store SecureString parameters

With AWS Systems Manager Parameter Store, you can create SecureString parameters, which are parameters that have a plaintext parameter name and an encrypted parameter value. Parameter Store uses AWS KMS to encrypt and decrypt the parameter values of `SecureString` parameters.

With Parameter Store, you can create, store, and manage data as parameters with values. You can create a parameter in Parameter Store and use it in multiple applications and services subject to policies and permissions that you design. When you need to change a parameter value, you change one instance, rather than managing error-prone changes to numerous sources. Parameter Store supports a hierarchical structure for parameter names, so you can qualify a parameter for specific uses.

To manage sensitive data, you can create `SecureString` parameters. Parameter Store uses AWS KMS keys to encrypt the parameter values of `SecureString` parameters when you create or change them. It also uses KMS keys to decrypt the parameter values when you access them. You can use the AWS managed key that Parameter Store creates for your account or specify your own customer managed key.

> **Important**
>
> Parameter Store supports only symmetric KMS keys. You cannot use an asymmetric KMS key to encrypt your parameters. For help determining whether a KMS key is symmetric or asymmetric, see Identify different key types in the *AWS Key Management Service Developer Guide*.

Parameter Store supports two tiers of `SecureString` parameters: *standard* and *advanced*. Standard parameters, which cannot exceed 4096 bytes, are encrypted and decrypted directly under the KMS key that you specify. To encrypt and decrypt advanced `SecureString` parameters, Parameter Store uses envelope encryption with the AWS Encryption SDK. You can convert a standard `SecureString` parameter to an advanced parameter, but you cannot convert an advanced parameter to a standard one. For more information about the difference between standard and advanced `SecureString` parameters, see Managing parameter tiers.

Topics

- Protecting standard SecureString parameters
- Protecting advanced SecureString parameters
- Setting permissions to encrypt and decrypt parameter values
- Parameter Store encryption context
- Troubleshooting KMS key issues in Parameter Store

Protecting standard SecureString parameters

Parameter Store does not perform any cryptographic operations. Instead, it relies on AWS KMS to encrypt and decrypt `SecureString` parameter values. When you create or change a standard `SecureString` parameter value, Parameter Store calls the AWS KMS Encrypt operation. This operation uses a symmetric encryption KMS key directly to encrypt the parameter value instead of using the KMS key to generate a data key.

You can select the KMS key that Parameter Store uses to encrypt the parameter value. If you do not specify a KMS key, Parameter Store uses the AWS managed key that Systems Manager automatically creates in your account. This KMS key has the `aws/ssm` alias.

To view the default `aws/ssm` KMS key for your account, use the DescribeKey operation in the AWS KMS API. The following example uses the `describe-key` command in the AWS Command Line Interface (AWS CLI) with the `aws/ssm` alias name.

```
aws kms describe-key \
    --key-id alias/aws/ssm
```

To create a standard `SecureString` parameter, use the PutParameter operation in the Systems Manager API. Omit the `Tier` parameter or specify a value of `Standard`, which is the default. Include a `Type` parameter with a value of `SecureString`. To specify a KMS key, use the `KeyId` parameter. The default is the AWS managed key for your account, `aws/ssm`.

Parameter Store then calls the AWS KMS `Encrypt` operation with the KMS key and the plaintext parameter value. AWS KMS returns the encrypted parameter value, which Parameter Store stores with the parameter name.

The following example uses the Systems Manager put-parameter command and its `--type` parameter in the AWS CLI to create a `SecureString` parameter. Because the command omits the optional `--tier` and `--key-id` parameters, Parameter Store creates a standard `SecureString` parameter and encrypts it under the AWS managed key.

```
aws ssm put-parameter \
    --name MyParameter \
    --value "secret_value" \
    --type SecureString
```

The following similar example uses the `--key-id` parameter to specify a customer managed key. The example uses a KMS key ID to identify the KMS key, but you can use any valid KMS key

identifier. Because the command omits the `Tier` parameter (`--tier`), Parameter Store creates a standard `SecureString` parameter, not an advanced one.

```
aws ssm put-parameter \
    --name param1 \
    --value "secret" \
    --type SecureString \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

When you get a `SecureString` parameter from Parameter Store, its value is encrypted. To get a parameter, use the GetParameter operation in the Systems Manager API.

The following example uses the Systems Manager get-parameter command in the AWS CLI to get the `MyParameter` parameter from Parameter Store without decrypting its value.

```
aws ssm get-parameter --name MyParameter
```

```
{
    "Parameter": {
        "Type": "SecureString",
        "Name": "MyParameter",
        "Value": "AQECAHgnOkMROh5LaLXkA4j0+vYi6tmM17Lg"
    }
}
```

To decrypt the parameter value before returning it, set the `WithDecryption` parameter of `GetParameter` to `true`. When you use `WithDecryption`, Parameter Store calls the AWS KMS Decrypt operation on your behalf to decrypt the parameter value. As a result, the `GetParameter` request returns the parameter with a plaintext parameter value, as shown in the following example.

```
aws ssm get-parameter \
    --name MyParameter \
    --with-decryption
```

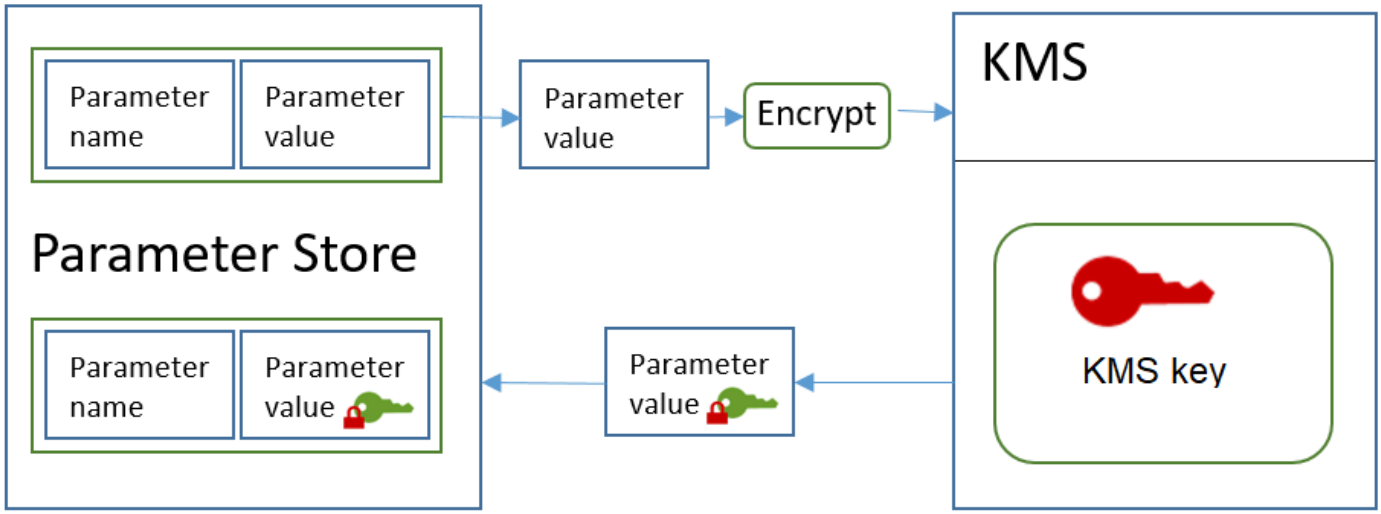
```
{
    "Parameter": {
        "Type": "SecureString",
        "Name": "MyParameter",
```

```
        "Value": "secret_value"
    }
}
```

The following workflow shows how Parameter Store uses a KMS key to encrypt and decrypt a standard `SecureString` parameter.

Encrypt a standard parameter

1. When you use `PutParameter` to create a `SecureString` parameter, Parameter Store sends an `Encrypt` request to AWS KMS. That request includes the plaintext parameter value, the KMS key that you chose, and the Parameter Store encryption context. During transmission to AWS KMS, the plaintext value in the `SecureString` parameter is protected by Transport Layer Security (TLS).
2. AWS KMS encrypts the parameter value with the specified KMS key and encryption context. It returns the ciphertext to Parameter Store, which stores the parameter name and its encrypted value.



Decrypt a standard parameter

1. When you include the `WithDecryption` parameter in a `GetParameter` request, Parameter Store sends a `Decrypt` request to AWS KMS with the encrypted `SecureString` parameter value and the Parameter Store encryption context.
2. AWS KMS uses the same KMS key and the supplied encryption context to decrypt the encrypted value. It returns the plaintext (decrypted) parameter value to Parameter Store. During transmission, the plaintext data is protected by TLS.

3. Parameter Store returns the plaintext parameter value to you in the `GetParameter` response.

Protecting advanced SecureString parameters

When you use `PutParameter` to create an advanced `SecureString` parameter, Parameter Store uses envelope encryption with the AWS Encryption SDK and a symmetric encryption AWS KMS key to protect the parameter value. Each advanced parameter value is encrypted under a unique data key, and the data key is encrypted under a KMS key. You can use the AWS managed key for the account (`aws/ssm`) or any customer managed key.

The AWS Encryption SDK is an open-source, client-side library that helps you to encrypt and decrypt data using industry standards and best practices. It's supported on multiple platforms and in multiple programming languages, including a command-line interface. You can view the source code and contribute to its development in GitHub.

For each `SecureString` parameter value, Parameter Store calls the AWS Encryption SDK to encrypt the parameter value using a unique data key that AWS KMS generates (GenerateDataKey). The AWS Encryption SDK returns to Parameter Store an encrypted message that includes the encrypted parameter value and an encrypted copy of the unique data key. Parameter Store stores the entire encrypted message in the `SecureString` parameter value. Then, when you get an advanced `SecureString` parameter value, Parameter Store uses the AWS Encryption SDK to decrypt the parameter value. This requires a call to AWS KMS to decrypt the encrypted data key.

To create an advanced `SecureString` parameter, use the PutParameter operation in the Systems Manager API. Set the value of `Tier` parameter to `Advanced`. Include a `Type` parameter with a value of `SecureString`. To specify a KMS key, use the `KeyId` parameter. The default is the AWS managed key for your account, `aws/ssm`.

```
aws ssm put-parameter \
    --name MyParameter \
    --value "secret_value" \
    --type SecureString \
    --tier Advanced
```

The following similar example uses the `--key-id` parameter to specify a customer managed key. The example uses the Amazon Resource Name (ARN) of the KMS key, but you can use any valid KMS key identifier.

```
aws ssm put-parameter \
```

```
    --name MyParameter \
    --value "secret_value" \
    --type SecureString \
    --tier Advanced \
    --key-id arn:aws:kms:us-
east-2:987654321098:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

When you get a `SecureString` parameter from Parameter Store, its value is the encrypted message that the AWS Encryption SDK returned. To get a parameter, use the GetParameter operation in the Systems Manager API.

The following example uses the Systems Manager `GetParameter` operation to get the `MyParameter` parameter from Parameter Store without decrypting its value.

```
aws ssm get-parameter --name MyParameter
```

```
{
    "Parameter": {
        "Type": "SecureString",
        "Name": "MyParameter",
        "Value": "AQECAHgnOkMROh5LaLXkA4j0+vYi6tmM17Lg"
    }
}
```

To decrypt the parameter value before returning it, set the `WithDecryption` parameter of `GetParameter` to `true`. When you use `WithDecryption`, Parameter Store calls the AWS KMS Decrypt operation on your behalf to decrypt the parameter value. As a result, the `GetParameter` request returns the parameter with a plaintext parameter value, as shown in the following example.

```
aws ssm get-parameter \
    --name MyParameter \
    --with-decryption
```

```
{
    "Parameter": {
        "Type": "SecureString",
        "Name": "MyParameter",
        "Value": "secret_value"
    }
```

```
}
```

You cannot convert an advanced `SecureString` parameter to a standard one, but you can convert a standard `SecureString` to an advanced one. To convert a standard `SecureString` parameter to an advanced `SecureString`, use the `PutParameter` operation with the `Overwrite` parameter. The `Type` must be `SecureString` and the `Tier` value must be `Advanced`. The `KeyId` parameter, which identifies a customer managed key, is optional. If you omit it, Parameter Store uses the AWS managed key for the account. You can specify any KMS key that the principal has permission to use, even if you used a different KMS key to encrypt the standard parameter.

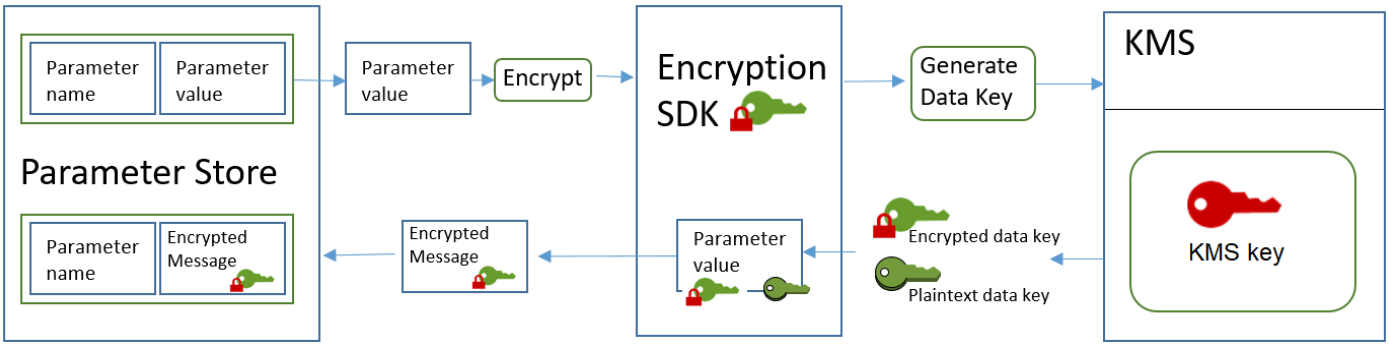
When you use the `Overwrite` parameter, Parameter Store uses the AWS Encryption SDK to encrypt the parameter value. Then it stores the newly encrypted message in Parameter Store.

```
aws ssm put-parameter \
    --name myStdParameter \
    --value "secret_value"  \
    --type SecureString \
    --tier Advanced \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --overwrite
```

The following workflow shows how Parameter Store uses a KMS key to encrypt and decrypt an advanced `SecureString` parameter.

Encrypt an advanced parameter

1. When you use `PutParameter` to create an advanced `SecureString` parameter, Parameter Store uses the AWS Encryption SDK and AWS KMS to encrypt the parameter value. Parameter Store calls the AWS Encryption SDK with the parameter value, the KMS key that you specified, and the Parameter Store encryption context.
2. The AWS Encryption SDK sends a GenerateDataKey request to AWS KMS with the identifier of the KMS key that you specified and the Parameter Store encryption context. AWS KMS returns two copies of the unique data key: one in plaintext and one encrypted under the KMS key. (The encryption context is used when encrypting the data key.)
3. The AWS Encryption SDK uses the plaintext data key to encrypt the parameter value. It returns an encrypted message that includes the encrypted parameter value, the encrypted data key, and other data, including the Parameter Store encryption context.
4. Parameter Store stores the encrypted message as the parameter value.



Decrypt an advanced parameter

1. You can include the `WithDecryption` parameter in a `GetParameter` request to get an advanced `SecureString` parameter. When you do, Parameter Store passes the encrypted message from the parameter value to a decryption method of the AWS Encryption SDK.
2. The AWS Encryption SDK calls the AWS KMS Decrypt operation. It passes in the encrypted data key and the Parameter Store encryption context from the encrypted message.
3. AWS KMS uses the KMS key and the Parameter Store encryption context to decrypt the encrypted data key. Then it returns the plaintext (decrypted) data key to the AWS Encryption SDK.
4. The AWS Encryption SDK uses the plaintext data key to decrypt the parameter value. It returns the plaintext parameter value to Parameter Store.
5. Parameter Store verifies the encryption context and returns the plaintext parameter value to you in the `GetParameter` response.

Setting permissions to encrypt and decrypt parameter values

To encrypt a standard `SecureString` parameter value, the user needs `kms:Encrypt` permission. To encrypt an advanced `SecureString` parameter value, the user needs `kms:GenerateDataKey` permission. To decrypt either type of `SecureString` parameter value, the user needs `kms:Decrypt` permission.

You can use AWS Identity and Access Management (IAM) policies to allow or deny permission for a user to call the Systems Manager `PutParameter` and `GetParameter` operations.

If you are using customer managed keys to encrypt your `SecureString` parameter values, you can use IAM policies and key policies to manage encrypt and decrypt permissions. However, you cannot

establish access control policies for the default `aws/ssm` KMS key. For detailed information about controlling access to customer managed keys, see KMS key access and permissions in the *AWS Key Management Service Developer Guide*.

The following example shows an IAM policy designed for standard `SecureString` parameters. It allows the user to call the Systems Manager `PutParameter` operation on all parameters in the `FinancialParameters` path. The policy also allows the user to call the AWS KMS `Encrypt` operation on an example customer managed key.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:PutParameter"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/
FinancialParameters/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt"
            ],
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

The next example shows an IAM policy that is designed for advanced `SecureString` parameters. It allows the user to call the Systems Manager `PutParameter` operation on all parameters in the `ReservedParameters` path. The policy also allows the user to call the AWS KMS `GenerateDataKey` operation on an example customer managed key.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:PutParameter"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/
ReservedParameters/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-id"
        }
    ]
}
```

The final example also shows an IAM policy that can be used for standard or advanced `SecureString` parameters. It allows the user to call the Systems Manager `GetParameter` operations (and related operations) on all parameters in the `ITParameters` path. The policy also allows the user to call the AWS KMS `Decrypt` operation on an example customer managed key.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter*"
            ],
            "Resource": "arn:aws:ssm:us-east-1:111122223333:parameter/
ITParameters/*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

Parameter Store encryption context

An *encryption context* is a set of key–value pairs that contain arbitrary nonsecret data. When you include an encryption context in a request to encrypt data, AWS KMS cryptographically binds the encryption context to the encrypted data. To decrypt the data, you must pass in the same encryption context.

You can also use the encryption context to identify a cryptographic operation in audit records and logs. The encryption context appears in plaintext in logs, such as AWS CloudTrail logs.

The AWS Encryption SDK also takes an encryption context, although it handles it differently. Parameter Store supplies the encryption context to the encryption method. The AWS Encryption SDK cryptographically binds the encryption context to the encrypted data. It also includes the encryption context in plain text in the header of the encrypted message that it returns. However, unlike AWS KMS, the AWS Encryption SDK decryption methods do not take an encryption context as input. Instead, when it decrypts data, the AWS Encryption SDK gets the encryption context from the encrypted message. Parameter Store verifies that the encryption context includes the value that it expects before returning the plaintext parameter value to you.

Parameter Store uses the following encryption context in its cryptographic operations:

- Key: PARAMETER_ARN
- Value: The Amazon Resource Name (ARN) of the parameter that is being encrypted.

The format of the encryption context is as follows:

```
"PARAMETER_ARN":"arn:aws:ssm:region-id:account-id:parameter/parameter-name"
```

For example, Parameter Store includes this encryption context in calls to encrypt and decrypt the `MyParameter` parameter in an example AWS account and region.

```
"PARAMETER_ARN":"arn:aws:ssm:us-east-2:111122223333:parameter/MyParameter"
```

If the parameter is in a Parameter Store hierarchical path, the path and name are included in the encryption context. For example, this encryption context is used when encrypting and decrypting the `MyParameter` parameter in the `/ReadableParameters` path in an example AWS account and region.

```
"PARAMETER_ARN":"arn:aws:ssm:us-east-2:111122223333:parameter/ReadableParameters/
MyParameter"
```

You can decrypt an encrypted `SecureString` parameter value by calling the AWS KMS `Decrypt` operation with the correct encryption context and the encrypted parameter value that the Systems Manager `GetParameter` operation returns. However, we encourage you to decrypt Parameter Store parameter values by using the `GetParameter` operation with the `WithDecryption` parameter.

You can also include the encryption context in an IAM policy. For example, you can permit a user to decrypt only one particular parameter value or set of parameter values.

The following example IAM policy statement allows the user to the get value of the `MyParameter` parameter and to decrypt its value using the specified KMS key. However the permissions apply only when the encryption context matches specified string. These permissions do not apply to any other parameter or KMS key, and the call to `GetParameter` fails if the encryption context does not match the string.

Before using a policy statement like this one, replace the *`example ARNs`* with valid values.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter*"
```

```
            ],
            "Resource": "arn:aws:ssm:us-
east-1:111122223333:parameter/MyParameter"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-id",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:PARAMETER_ARN":"arn:aws:ssm:us-
east-1:111122223333:parameter/MyParameter"
                }
            }
        }
    ]
}
```

Troubleshooting KMS key issues in Parameter Store

To perform any operation on a `SecureString` parameter, Parameter Store must be able to use the AWS KMS KMS key that you specify for your intended operation. Most of the Parameter Store failures related to KMS keys are caused by the following problems:

- The credentials that an application is using do not have permission to perform the specified action on the KMS key.

  To fix this error, run the application with different credentials or revise the IAM or key policy that is preventing the operation. For help with AWS KMS IAM and key policies, see KMS key access and permissions in the *AWS Key Management Service Developer Guide*.
- The KMS key is not found.

  This typically happens when you use an incorrect identifier for the KMS key. Find the correct identifiers for the KMS key and try the command again.
- The KMS key is not enabled. When this occurs, Parameter Store returns an InvalidKeyId exception with a detailed error message from AWS KMS. If the KMS key state is `Disabled`, enable it. If it is `Pending Import`, complete the import procedure. If the key state is `Pending Deletion`, cancel the key deletion or use a different KMS key.

To find the key state of a KMS key, use the DescribeKey operation.

Use Parameter Store parameters in Amazon Elastic Kubernetes Service

To show parameters from Parameter Store, a tool of AWS Systems Manager, as files mounted in Amazon EKS Pods, you can use the AWS Secrets and Configuration Provider for the Kubernetes Secrets Store CSI Driver. The ASCP works with Amazon Elastic Kubernetes Service 1.17+ running an Amazon EC2 node group. AWS Fargate node groups are not supported.

With the ASCP, you can store and manage your parameter in Parameter Store and then retrieve them through your workloads running on Amazon EKS. If your parameter contains multiple key-value pairs in JSON format, you can choose which ones to mount in Amazon EKS. The ASCP uses JMESPath syntax to query the key-value pairs in your secret. The ASCP also works with AWS Secrets Manager secrets.

The ASCP offers two methods of authentication with Amazon EKS The first approach uses IAM Roles for Service Accounts (IRSA). The second approach uses Pod Identities. Each approach has its benefits and use cases.

ASCP with IAM Roles for Service Accounts (IRSA)

The ASCP with IAM Roles for Service Accounts (IRSA) allows you to mount parameters from Parameter Store as files in your Amazon EKS Pods. This approach is suitable when:

- You need to mount parameters as files in your Pods.
- You're using Amazon EKS version 1.17 or later with Amazon EC2 node groups.
- You want to retrieve specific key-value pairs from JSON-formatted parameters.

For more information, see the section called “Integrate ASCP with IRSA for Amazon EKS”.

ASCP with Pod Identity

ASCP with EKS Pod Identity

The ASCP with Pod Identity method enhances security and simplifies configuration for accessing parameters in Parameter Store. This approach is beneficial when:

- You need more granular permission management at the Pod level.

- You're using Amazon EKS version 1.24 or later.
- You want improved performance and scalability.

For more information, see the section called “Integrate ASCP with Pod Identity for Amazon EKS”.

Choosing the right approach

Consider the following factors when deciding between ASCP with IRSA and ASCP with Pod Identity:

- Amazon EKSversion: Pod Identity requires Amazon EKS 1.24+, while CSI driver works with Amazon EKS 1.17+.
- Security requirements: Pod Identity offers more granular control at the Pod level.
- Performance: Pod Identity generally performs better in high-scale environments.
- Complexity: Pod Identity simplifies setup by eliminating the need for separate service accounts.

Choose the method that best aligns with your specific requirements and Amazon EKS environment.

Install ASCP for Amazon EKS

This section explains how to install the AWS Secrets and Configuration Provider for Amazon EKS. With ASCP, you can mount parameters from Parameter Store and secrets from AWS Secrets Manager as files in Amazon EKS Pods.

Prerequisites

- An Amazon EKS cluster
  - Version 1.24 or later for Pod Identity
  - Version 1.17 or later for IRSA
- The AWS CLI installed and configured
- kubectl installed and configured for your Amazon EKS cluster
- Helm (version 3.0 or later)

Install and configure the ASCP

The ASCP is available on GitHub in the secrets-store-csi-provider-aws repository. The repo also contains example YAML files for creating and mounting a secret by changing the `objectType` value from `secretsmanager` to `ssmparameter`.

During installation, you can configure the ASCP to use a FIPS endpoint. For a list of Systems Manager endpoints, see Systems Manager service endpoints in the *Amazon Web Services General Reference*.

To install the ASCP by using Helm

1. To make sure the repo is pointing to the latest charts, use `helm repo update`.
2. Add the Secrets Store CSI Driver chart.

```
helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
```

3. Install the chart. To configure throttling, add the following flag: `--set-json 'k8sThrottlingParams={"qps": "number of queries per second", "burst": "number of queries per second"}'`

```
helm install -n kube-system csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver
```

4. Add the ASCP chart.

```
helm repo add aws-secrets-manager https://aws.github.io/secrets-store-csi-driver-provider-aws
```

5. Install the chart. To use a FIPS endpoint, add the following flag: `--set useFipsEndpoint=true`

```
helm install -n kube-system secrets-provider-aws aws-secrets-manager/secrets-store-csi-driver-provider-aws
```

To install by using the YAML in the repo

- Use the following commands.

```
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
```

```
kubectl apply -f https://raw.githubusercontent.com/aws/secrets-store-csi-driver-
provider-aws/main/deployment/aws-provider-installer.yaml
```

Verify the installations

To verify the installations of your EKS cluster, Secrets Store CSI driver, and ASCP plugin, follow these steps:

1. Verify the EKS cluster:

   ```
   eksctl get cluster --name clusterName
   ```

   This command should return information about your cluster.

2. Verify the Secrets Store CSI driver installation:

   ```
   kubectl get pods -n kube-system -l app=secrets-store-csi-driver
   ```

   You should see Pods running with names like `csi-secrets-store-secrets-store-csi-driver-xxx`.

3. Verify the ASCP plugin installation:

   YAML installation

   ```
   $ kubectl get pods -n kube-system -l app=csi-secrets-store-provider-aws
   ```

   Example output:

   ```
   NAME                                     READY   STATUS    RESTARTS   AGE
   csi-secrets-store-provider-aws-12345      1/1     Running   0          2m
   ```

   Helm installation

   ```
   $  kubectl get pods -n kube-system -l app=secrets-store-csi-driver-provider-aws
   ```

   Example output:

```
NAME                                                     READY   STATUS    RESTARTS
 AGE
secrets-provider-aws-secrets-store-csi-driver-provider-67890       1/1
 Running   0          2m
```

You should see Pods in the Running state.

After running these commands, if everything is set up correctly, you should see all components running without any errors. If you encounter any issues, you may need to troubleshoot by checking the logs of the specific Pods that are having problems.

Troubleshooting

1. To check the logs of the ASCP provider, run:

   ```
   kubectl logs -n kube-system -l app=csi-secrets-store-provider-aws
   ```

2. Check the status of all pods in the kube-system namespace.

   Replace the *default placeholder text* with your own pod ID:

   ```
   kubectl -n kube-system get pods
   ```

   ```
   kubectl -n kube-system logs pod/pod-id
   ```

   All Pods related to the CSI driver and ASCP should be in the 'Running' state.

3. Check the CSI driver version:

   ```
   kubectl get csidriver secrets-store.csi.k8s.io -o yaml
   ```

   This command should return information about the installed CSI driver.

Additional resources

For more information about using ASCP with Amazon EKS, see the following resources:

- Using Pod Identity with Amazon EKS

- AWS Secrets Store CSI Driver on GitHub

Use AWS Secrets and Configuration Provider CSI with Pod Identity for Amazon EKS

The AWS Secrets and Configuration Provider integration with the Pod Identity Agent for Amazon Elastic Kubernetes Service provides enhanced security, simplified configuration, and improved performance for applications running on Amazon EKS. Pod Identity simplifies AWS Identity and Access Management (IAM) authentication for Amazon EKS when retrieving parameters from AWS Systems Manager Parameter Store or secrets from Secrets Manager.

Amazon EKS Pod Identity streamlines the process of configuring IAM permissions for Kubernetes applications by allowing permissions to be set up directly through Amazon EKS interfaces, reducing the number of steps and eliminating the need to switch between Amazon EKS and IAM services. Pod Identity enables the use of a single IAM role across multiple clusters without updating trust policies and supports role session tags for more granular access control. This approach not only simplifies policy management by allowing reuse of permission policies across roles but also enhances security by enabling access to AWS resources based on matching tags.

How it works

1. Pod Identity assigns an IAM role to the Pod.
2. ASCP uses this role to authenticate with AWS services.
3. If authorized, ASCP retrieves the requested parameters and makes them available to the Pod.

For more information, see Understand how Amazon EKS Pod Identity works in the *Amazon EKS User Guide*.

Prerequisites

> **Important**
>
> Pod Identity is supported only for Amazon EKS in the cloud. It is not supported for Amazon EKS Anywhere, Red Hat OpenShift Service on AWS, or self-managed Kubernetes clusters on Amazon EC2 instances.

- Amazon EKS cluster (version 1.24 or later)

- Access to AWS CLI and Amazon EKS cluster via `kubectl`
- (Optional) Access to two AWS accounts for cross-account access

Install the Amazon EKS Pod Identity Agent

To use Pod Identity with your cluster, you must install the Amazon EKS Pod Identity Agent add-on.

To install the Pod Identity Agent

- Install the Pod Identity Agent add-on on your cluster.

  Replace the *default placeholder text* with your own values:

```
eksctl create addon \
  --name eks-pod-identity-agent \
  --cluster clusterName \
  --region region
```

Set up ASCP with Pod Identity

1. Create a permissions policy that grants `ssm:GetParameters` and `ssm:DescribeParameters` permission to the parameters that the Pod needs to access.
2. Create an IAM role that can be assumed by the Amazon EKS service principal for Pod Identity:

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "pods.eks.amazonaws.com"
        },
        "Action": [
          "sts:AssumeRole",
          "sts:TagSession"
        ]
      }
```

```
    ]
  }
```

Attach the IAM policy to the role.

Replace the *default placeholder text* with your own values:

```
aws iam attach-role-policy \
  --role-name MY_ROLE \
  --policy-arn POLICY_ARN
```

3. Create a Pod Identity association. For an example, see Create a Pod Identity association in the *Amazon EKS User Guide*
4. Create the `SecretProviderClass` that specifies which parameters or secrets to mount in the Pod:

```
kubectl apply -f kubectl apply -f https://raw.githubusercontent.com/aws/
secrets-store-csi-driver-provider-aws/main/examples/ExampleSecretProviderClass-
PodIdentity.yaml
```

The key difference in `SecretProviderClass` between IRSA and Pod Identity is the optional parameter `usePodIdentity`. It is an optional field that determines the authentication approach. When not specified, it defaults to using IAM Roles for Service Accounts (IRSA).

- To use EKS Pod Identity, use any of these values: `"true", "True", "TRUE", "t", "T"`.
- To explicitly use IRSA, set to any of these values: `"false", "False", "FALSE", "f", or "F"`.

5. Deploy the Pod that mounts the parameters or secrets under `/mnt/secrets-store`:

```
kubectl apply -f kubectl apply -f https://raw.githubusercontent.com/aws/secrets-
store-csi-driver-provider-aws/main/examples/ExampleDeployment-PodIdentity.yaml
```

6. If you use a private Amazon EKS cluster, make sure that the VPC that the cluster is in has an AWS STS endpoint. For information about creating an endpoint, see Interface VPC endpoints in the *AWS Identity and Access Management User Guide*.

Verify the secret mount

To verify that the parameter or secret is mounted properly, run the following command.

Replace the *default placeholder text* with your own values:

```
kubectl exec -it $(kubectl get pods | awk '/pod-identity-deployment/{print $1}' | head
 -1) -- cat /mnt/secrets-store/MyParameter
```

To set up Amazon EKS Pod Identity to access to parameters in Parameter Store

1. Create a permissions policy that grants `ssm:GetParameters` and `ssm:DescribeParameters` permission to the parameters that the Pod needs to access.
2. Create a parameter in Parameter Store, if you do not already have one. For information, see Creating Parameter Store parameters in Systems Manager.

Troubleshoot

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
  ```

Use AWS Secrets and Configuration Provider CSI with IAM Roles for Service Accounts (IRSA)

Topics

- Prerequisites
- Set up access control
- Identify which parameters to mount
- Troubleshoot

Prerequisites

- Amazon EKS cluster (version 1.17 or later)
- Access to AWS CLI and Amazon EKS cluster via `kubectl`

Set up access control

The ASCP retrieves the Amazon EKS Pod Identity and exchanges it for an IAM role. You set permissions in an IAM policy for that IAM role. When the ASCP assumes the IAM role, it gets access to the parameters you authorized. Other containers can't access the parameters unless you also associate them with the IAM role.

To grant your Amazon EKS Pod access to parameters in Parameter Store

1. Create a permissions policy that grants `ssm:GetParameters` and `ssm:DescribeParameters` permission to the parameters that the Pod needs to access.
2. 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 in the *Amazon EKS User Guide*.
3. Create an IAM role for service account and attach the policy to it. For more information, see Create an IAM role for a service account in the *Amazon EKS User Guide*.
4. If you use a private Amazon EKS cluster, make sure that the VPC that the cluster is in has an AWS STS endpoint. For information about creating an endpoint, see Interface VPC endpoints in the *AWS Identity and Access Management User Guide*.

Identify which parameters to mount

To determine which parameters the ASCP mounts in Amazon EKS as files on the filesystem, you create a the section called "SecretProviderClass" YAML file. The `SecretProviderClass` lists the parameters to mount and the file name to mount them as. The `SecretProviderClass` must be in the same namespace as the Amazon EKS Pod it references.

Mount the parameters as files

The following instructions show how to mount parameters as files using example YAML files ExampleSecretProviderClass.yaml and ExampleDeployment.yaml.

To mount parameters in Amazon EKS

1. Apply the `SecretProviderClass` to the Pod:

```
kubectl apply -f ExampleSecretProviderClass.yaml
```

2. Deploy your Pod:

```
kubectl apply -f ExampleDeployment.yaml
```

3. The ASCP mounts the files.

Troubleshoot

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` *`name-space`*.

```
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 *pod-id* 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
```

- **Verify that the `SecretProviderClass` CRD is installed:**

```
kubectl get crd secretproviderclasses.secrets-store.csi.x-k8s.io
```

  This command should return information about the `SecretProviderClass` custom resource definition.

- **Verify that the SecretProviderClass object was created.**

```
kubectl get secretproviderclass SecretProviderClassName -o yaml
```

AWS Secrets and Configuration Provider code examples

ASCP authentication and access control examples

Example: IAM policy allowing Amazon EKS Pod Identity service (pods.eks.amazonaws.com) to assume the role and tag the session:

JSON

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

```
        "sts:TagSession"
      ]
    }
  ]
}
```

SecretProviderClass

You use YAML to describe which parameters to mount in Amazon EKS using the ASCP. For examples, see the section called “SecretProviderClass usage”.

SecretProviderClass YAML structure

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
   name: name
spec:
  provider: aws
  parameters:
    region:
    failoverRegion:
    pathTranslation:
    usePodIdentity:
    preferredAddressType:
    objects:
```

The parameters field contains the details of the mount request:

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.

If you also specify `failoverRegion`, the ASCP tries to retrieve the parameter from both Regions. If either Region returns a `4xx` error, for example for an authentication issue, the ASCP does not mount either parameter. If the parameter is retrieved successfully from `region`, then the ASCP mounts that parameter value. If the parameter is not retrieved successfully from `region`, but it is retrieved successfully from `failoverRegion`, then the ASCP mounts that parameter value.

`failoverRegion`

(Optional) If you include this field, the ASCP tries to retrieve the parameter from the Regions defined in `region` and this field. If either Region returns a `4xx` error, for example for an authentication issue, the ASCP does not mount either parameter. If the parameter is retrieved successfully from `region`, then the ASCP mounts that parameter value. If the parameter is not retrieved successfully from `region`, but it is retrieved successfully from `failoverRegion`, then the ASCP mounts that parameter value. For an example of how to use this field, see Multi-Region parameter failover.

pathTranslation

(Optional) A single substitution character to use if the file name in Amazon EKS will contain the path separator character, such as slash (/) on Linux. The ASCP can't create a mounted file that contains a path separator character. Instead, the ASCP replaces the path separator character with a different character. If you don't use this field, the replacement character is underscore (_), so for example, `My/Path/Parameter` mounts as `My_Path_Parameter`.

To prevent character substitution, enter the string `False`.

`usePodIdentity`

(Optional) Determines the authentication approach. When not specified, it defaults to IAM Roles for Service Accounts (IRSA) (IRSA).

- To use EKS Pod Identity, use any of these values: `"true""`, `"True"`, `"TRUE"`, `"t"`, or `"T"`.
- To explicitly use IRSA, set to any of these values: `"false"`, `"False"`, `"FALSE"`, `"f"`, or `"F""=`.

`preferredAddressType`

(Optional) Specifies the preferred IP address type for Pod Identity Agent endpoint communication. The field is only applicable when using EKS Pod Identity feature and will be ignored when using IAM Roles for Service Accounts.Values are case-insensitive. Valid values are:

- `"ipv4"`, `"IPv4""`, or `"IPV4"` – Force the use of Pod Identity Agent IPv4 endpoint
- `"ipv6"`, `"IPv6"`, or `"IPV6"` – Force the use of Pod Identity Agent IPv6 endpoint
- not specified – Use auto endpoint selection, trying IPv4 endpoint first and falling back to IPv6 endpoint if IPv4 fails

objects

A string containing a YAML declaration of the secrets to be mounted. We recommend using a YAML multi-line string or pipe (|) character.

objectName

Required. Specifies the name of the parameter or secret to be fetched. For Parameter Store, this is the `Name` of the parameter and can be either the name or full ARN of the parameter. For Secrets Manager this is the `SecretId` parameter and can be either the friendly name or full ARN of the secret.

objectType

Required if you don't use a Secrets Manager ARN for `objectName`. For Parameter Store, use `ssmparameter`. For Secrets Manager, use `secretsmanager`.

objectAlias

(Optional) The file name of the secret in the Amazon EKS Pod. If you don't specify this field, the `objectName` appears as the file name.

objectVersion

(Optional) The version ID of the parameter. Not recommended because you must update the version ID every time you update the parameter. By default the most recent version is used. If you include a `failoverRegion`, this field represents the primary `objectVersion`.

objectVersionLabel

(Optional) The alias for the version. The default is the most recent version `AWSCURRENT`. If you include a `failoverRegion`, this field represents the primary `objectVersionLabel`.

jmesPath

(Optional) A map of the keys in the parameter to the files to be mounted in Amazon EKS. To use this field, your parameter value must be in JSON format.

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`.

If you use this field, you must include the subfields `path` and `objectAlias`.

path

A key from a key-value pair in the JSON of the parameter value. If the field contains a hyphen, use single quotes to escape it, for example: `path: '"hyphenated-path"'`

objectAlias

The file name to be mounted in the Amazon EKS Pod. If the field contains a hyphen, use single quotes to escape it, for example: `objectAlias: '"hyphenated-alias"'`

`failoverObject`

(Optional) If you specify this field, the ASCP tries to retrieve both the parameter specified in the primary `objectName` and the parameter specified in the `failoverObject` `objectName` sub-field. If either returns a `4xx` error, for example for an authentication issue, the ASCP does not mount either parameter. If the parameter is retrieved successfully from the primary `objectName`, then the ASCP mounts that parameter value. If the parameter is not retrieved successfully from the primary `objectName`, but it is retrieved successfully from the failover `objectName`, then the ASCP mounts that parameter value. If you include this field, you must include the field `objectAlias`. For an example of how to use this field, see Failover to a different parameter.

You typically use this field when the failover parameter isn't a replica. For an example of how to specify a replica, see Multi-Region parameter failover.

objectName

The name or full ARN of the failover parameter. If you use an ARN, the Region in the ARN must match the field `failoverRegion`.

objectVersion

(Optional) The version ID of the parameter. Must match the primary `objectVersion`. Not recommended because you must update the version ID every time you update the parameter. By default the most recent version is used.

objectVersionLabel

(Optional) The alias for the version. The default is the most recent version `AWSCURRENT`.

Create a basic SecretProviderClass configuration to mount parameters in your Amazon EKS Pods.

Pod Identity

SecretProviderClass to use a parameter in the same Amazon EKS cluster:

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-parameter-store
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "MyParameter"
        objectType: "ssmparameter"
    usePodIdentity: "true"
```

IRSA

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: deployment-aws-parameter
spec:
  provider: aws
  parameters:
    objects: |
        - objectName: "MyParameter"
          objectType: "ssmparameter"
```

SecretProviderClass usage

Use these examples to create `SecretProviderClass` configurations for different scenarios.

Example: Mount parameters by name or ARN

This example shows how to mount three different types of parameters:

- A parameter specified by full ARN

- A parameter specified by name
- A parameter version of a secret

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-parameters
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "arn:aws:ssm:us-east-2:777788889999:parameter:MyParameter2-d4e5f6"
      - objectName: "MyParameter3"
        objectType: "ssmparameter"
      - objectName: "MyParameter4"
        objectType: "ssmparameter"
        objectVersionLabel: "AWSCURRENT"
```

Example: Mount key-value pairs from a parameter

This example shows how to mount specific key-value pairs from a JSON-formatted parameter:

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-parameters
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "arn:aws:ssm:us-east-2:777788889999:parameter:MyParameter-a1b2c3"
        jmesPath:
            - path: username
              objectAlias: dbusername
            - path: password
              objectAlias: dbpassword
```

Example: Failover configuration examples

These examples show how to configure failover for parameters.

Multi-Region parameter failover

This example shows how to configure automatic failover for a parameter replicated across multiple Regions:

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-parameters
spec:
  provider: aws
  parameters:
    region: us-east-1
    failoverRegion: us-east-2
    objects: |
      - objectName: "MyParameter"
```

Failover to a different parameter

This example shows how to configure failover to a different parameter (not a replica):

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-parameters
spec:
  provider: aws
  parameters:
    region: us-east-1
    failoverRegion: us-east-2
    objects: |
      - objectName: "arn:aws:ssm:us-east-1:777788889999:parameter:MyParameter-a1b2c3"
        objectAlias: "MyMountedParameter"
        failoverObject:
          - objectName: "arn:aws:ssm:us-
east-2:777788889999:parameter:MyFailoverParameter-d4e5f6"
```

Additional resources

For more information about using ASCP with Amazon EKS, see the following resources:

- Using Pod Identity with Amazon EKS

- AWS Secrets Store CSI Driver on GitHub

Using Parameter Store parameters in AWS Lambda functions

Parameter Store, a tool in 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.

To use parameters from Parameter Store in AWS Lambda functions without using an SDK, you can use the AWS Parameters and Secrets Lambda Extension. This extension retrieves parameter values and caches them for future use. Using the Lambda extension can reduce your costs by reducing the number of API calls to Parameter Store. Using the extension can also improve latency because retrieving a cached parameter is faster than retrieving it from Parameter Store.

A Lambda extension is a companion process that augments the capabilities of a Lambda function. An extension is like a client that runs in parallel to a Lambda invocation. This parallel client can interface with your function at any point during its lifecycle. For more information about Lambda extensions, see Lambda Extensions API in the *AWS Lambda Developer Guide*.

The AWS Parameters and Secrets Lambda Extension works for both Parameter Store and AWS Secrets Manager. To learn how to use the Lambda extension with secrets from Secrets Manager, see Use AWS Secrets Manager secrets in AWS Lambda functions in the *AWS Secrets Manager User Guide*.

Related info

Using the AWS Parameter and Secrets Lambda extension to cache parameters and secrets (AWS Compute Blog)

How the extension works

To use parameters in a Lambda function *without* the Lambda extension, you must configure your Lambda function to receive configuration updates by integrating with the `GetParameter` API action for Parameter Store.

When you use the AWS Parameters and Secrets Lambda Extension, the extension retrieves the parameter value from Parameter Store and stores it in the local cache. Then, the cached value is used for further invocations until it expires. Cached values expire after they pass their time-to-live (TTL). You can configure the TTL value using the `SSM_PARAMETER_STORE_TTL` environment variable, as explained later in this topic.

If the configured cache TTL has not expired, the cached parameter value is used. If the time has expired, the cached value is invalidated and the parameter value is retrieved from Parameter Store.

Also, the system detects parameter values that are used frequently and maintains them in the cache while clearing those that are expired or unused.

> **Important**
>
> The extension can be invoked only in the `INVOKE` phase of the Lambda operation and not during the `INIT` phase.

Implementation details

Use the following details to help you configure the AWS Parameters and Secrets Lambda Extension.

Authentication

To authorize and authenticate Parameter Store requests, the extension uses the same credentials as those used to run the Lambda function itself. Therefore, the AWS Identity and Access Management (IAM) role used to run the function must have the following permissions to interact with Parameter Store:

- `ssm:GetParameter` – Required to retrieve parameters from Parameter Store
- `kms:Decrypt` – Required if you are retrieving `SecureString` parameters from Parameter Store

For more information, see AWS Lambda execution role in the *AWS Lambda Developer Guide*.

Instantiation

Lambda instantiates separate instances corresponding to the concurrency level that your function requires. Each instance is isolated and maintains its own local cache of your configuration data. For more information about Lambda instances and concurrency, see Configuring reserved concurrency in the *AWS Lambda Developer Guide*.

No SDK dependence

The AWS Parameters and Secrets Lambda Extension works independently of any AWS SDK language library. An AWS SDK is not required to make GET requests to Parameter Store.

Localhost port

Use `localhost` in your GET requests. The extension makes requests to localhost port 2773. You do not need to specify an external or internal endpoint to use the extension. You can configure the port by setting the environment variable `PARAMETERS_SECRETS_EXTENSION_HTTP_PORT`.

For example, in Python, your GET URL might look something like the following example.

```
parameter_url = ('http://localhost:' + port + '/systemsmanager/parameters/get/?
name=' + ssm_parameter_path)
```

Changes to a parameter value before TTL expires

The extension doesn't detect changes to the parameter value and doesn't perform an auto-refresh before the TTL expires. If you change a parameter value, operations that use the cached parameter value might fail until the cache is next refreshed. If you expect frequent changes to a parameter value, we recommend setting a shorter TTL value.

Header requirement

To retrieve parameters from the extension cache, the header of your GET request must include an `X-Aws-Parameters-Secrets-Token` reference. Set the token to `AWS_SESSION_TOKEN`, which is provided by Lambda for all running functions. Using this header indicates that the caller is within the Lambda environment.

Example

The following example in Python demonstrates a basic request to retrieve the value of a cached parameter.

```
import urllib.request
import os
import json

aws_session_token = os.environ.get('AWS_SESSION_TOKEN')

def lambda_handler(event, context):
    # Retrieve /my/parameter from Parameter Store using extension cache
    req = urllib.request.Request('http://localhost:2773/systemsmanager/parameters/
get?name=%2Fmy%2Fparameter')
    req.add_header('X-Aws-Parameters-Secrets-Token', aws_session_token)
```

```
    config = urllib.request.urlopen(req).read()

    return json.loads(config)
```

ARM support

The extension supports the ARM architecture in most AWS Regions where the x86_64 and x86 architectures are supported. If you are using the ARM architecture, we suggest you verify your architecture is supported. For complete lists of extension ARNs, see AWS Parameters and Secrets Lambda Extension ARNs.

Logging

Lambda logs execution information about the extension along with the function by using Amazon CloudWatch Logs. By default, the extension logs a minimal amount of information to CloudWatch. To log more details, set the environment variable `PARAMETERS_SECRETS_EXTENSION_LOG_LEVEL` to `DEBUG`.

Adding the extension to a Lambda function

To use the AWS Parameters and Secrets Lambda Extension, you add the extension to your Lambda function as a layer.

Use one of the following methods to add the extension to your function.

AWS Management Console (Add layer option)

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose your function. In the **Layers** area, choose **Add a layer**.
3. In the **Choose a layer** area, choose the **AWS layers** option.
4. For **AWS layers**, choose **AWS-Parameters-and-Secrets-Lambda-Extension**, choose a version, and then choose **Add**.

AWS Management Console (Specify ARN option)

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose your function. In the **Layers** area, choose **Add a layer**.
3. In the **Choose a layer** area, choose the **Specify an ARN** option.
4. For **Specify an ARN**, enter the extension ARN for your AWS Region and architecture, and then choose **Add**.

AWS Command Line Interface

Run the following command in the AWS CLI. Replace each *example resource placeholder* with your own information.

```
aws lambda update-function-configuration \
    --function-name function-name \
    --layers layer-ARN
```

Related information

Using layers with your Lambda function

Configuring extensions (.zip file archive)

AWS Parameters and Secrets Lambda Extension environment variables

You can configure the extension by changing the following environment variables. To see the current settings, set PARAMETERS_SECRETS_EXTENSION_LOG_LEVEL to DEBUG. For more information, see Using AWS Lambda environment variables in the *AWS Lambda Developer Guide*.

> **Note**
>
> AWS Lambda records operation details about the Lambda extension and Lambda function in Amazon CloudWatch Logs.

Environment variable	Details	Required	Valid values	Default value
SSM_PARAMETER_STORE_TIMEOUT_MILLIS	Timeout, in milliseconds, for requests to Parameter Store. A value of 0 (zero) indicates no timeout.	No	All whole numbers	0 (zero)

Environment variable	Details	Required	Valid values	Default value
SECRETS_MANAGER_TIMEOUT_MILLIS	Timeout, in milliseconds, for requests to Secrets Manager. A value of 0 (zero) indicates no timeout.	No	All whole numbers	0 (zero)
SSM_PARAMETER_STORE_TTL	Maximum valid lifetime, in seconds, of a parameter in the cache before it is invalidated. A value of 0 (zero) indicates that the cache should be bypassed. This variable is ignored if the value for PARAMETERS_SECRETS_EXTENSION_CACHE_SIZE is 0 (zero).	No	0 (zero) to 300 s (Five minutes)	300 s (Five minutes)

Environment variable	Details	Required	Valid values	Default value
SECRETS_MANAGER_TTL	Maximum valid lifetime, in seconds, of a secret in the cache before it is invalidated. A value of 0 (zero) indicates that the cache is bypassed. This variable is ignored if the value for PARAMETERS_SECRETS_EXTENSION_CACHE_SIZE is 0 (zero).	No	0 (zero) to 300 s (Five minutes)	300 s (5 minutes)
PARAMETERS_SECRETS_EXTENSION_CACHE_ENABLED	Determines whether the cache for the extension is enabled. Value values: TRUE | FALSE	No	TRUE | FALSE	TRUE

Environment variable	Details	Required	Valid values	Default value
PARAMETERS_SECRETS_EXTENSION_CACHE_SIZE	The maximum size of the cache in terms of number of items. A value of 0 (zero) indicates that the cache is bypassed. This variable is ignored if both cache TTL values are 0 (zero).	No	0 (zero) to 1000	1000
PARAMETERS_SECRETS_EXTENSION_HTTP_PORT	The port for the local HTTP server.	No	1 - 65535	2773

Environment variable	Details	Required	Valid values	Default value
PARAMETERS_SECRETS_EXTENSION_MAX_CONNECTIONS	Maximum number of connections for the HTTP clients that the extension uses to make requests to Parameter Store or Secrets Manager. This is a per-client configuration for the number of connections that both the Secrets Manager client and Parameter Store client make to the backend services.	No	Minimum of 1; No maximum limit.	3

Environment variable	Details	Required	Valid values	Default value
PARAMETERS_SECRETS_EXTENSION_LOG_LEVEL	The level of detail reported in logs for the extension. We recommend using DEBUG for the most detail about your cache configuration as you set up and test the extension. Logs for Lambda operations are automatically pushed to an associated CloudWatch Logs log group.	No	DEBUG | WARN | ERROR | NONE | INFO	INFO

Sample commands for using the AWS Systems Manager Parameter Store and AWS Secrets Manager Extension

The examples in this section demonstrate API actions for use with the AWS Systems Manager Parameter Store and AWS Secrets Manager extension.

Sample commands for Parameter Store

The Lambda extension uses read-only access to the **GetParameter** API action.

To call this action, make an HTTP GET call similar to the following. This command format provides access to parameters in the standard parameter tier.

```
GET http://localhost:port/systemsmanager/parameters/get?name=parameter-
name&version=version&label=label&withDecryption={true|false}
```

In this example, *parameter-name* represents the full parameter name, such as `MyParameter`, for a parameter not in a hierarchy, or `%2FDev%2FProduction%2FEast%2FProject-ABC %2FMyParameter` for a parameter named `/Dev/Production/East/Project-ABC/ MyParameter` that is part of a hierarchy.

> **Note**
>
> When using GET calls, parameter values must be encoded for HTTP to preserve special characters. For example, instead of formatting a hierarchical path like `/a/b/c`, encode characters that could be interpreted as part of the URL, such as `%2Fa%2Fb%2Fc`.

version and *label* are the selectors available for use with the `GetParameter` action.

```
GET http://localhost:port/systemsmanager/parameters/get/?name=MyParameter&version=5
```

To call a parameter in a hierarchy, make an HTTP GET call similar to the following.

```
GET http://localhost:port/systemsmanager/parameters/get?name=%2Fa%2Fb%2F&label=release
```

To call a public (global) parameter, make an HTTP GET call similar to the following.

```
GET http://localhost:port/systemsmanager/parameters/get/?name=%2Faws%2Fservice%20list
%2F...
```

To make an HTTP GET call to a Secrets Manager secret by using Parameter Store references, make an HTTP GET call similar to the following.

```
GET http://localhost:port/systemsmanager/parameters/get?name=%2Faws%2Freference
%2Fsecretsmanager%2F...
```

To make a call using the Amazon Resource Name (ARN) for a parameter, make an HTTP GET call similar to the following.

```
GET http://localhost:port/systemsmanager/parameters/get?name=arn:aws:ssm:us-
east-1:123456789012:parameter/MyParameter
```

To make a call that accesses a `SecureString` parameter with decryption, make an HTTP GET call similar to the following.

```
GET http://localhost:port/systemsmanager/parameters/get?
name=MyParameter&withDecryption=true
```

You can specify that parameters aren't decrypted by omitting `withDecryption` or explicitly setting it to `false`. You can also specify either a version or a label, but not both. If you do, only the first of these that is placed after question mark (?) in the URL is used.

AWS Parameters and Secrets Lambda Extension ARNs

The following tables provide extension ARNs for supported architectures and Regions.

Topics

- Extension ARNs for the x86_64 and x86 architectures
- Extension ARNs for ARM64 and Mac with Apple silicon architectures

Extension ARNs for the x86_64 and x86 architectures

Last updated: September 03, 2025

Region	ARN
US East (Ohio)	`arn:aws:lambda:us-east-2:590474943231:layer:AWS-Parameters-and-Secrets-Lambda-Extension:23`
US East (N. Virginia)	`arn:aws:lambda:us-east-1:177933569100:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19`

Region	ARN
US West (N. California)	arn:aws:lambda:us-west-1:997803712105:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
US West (Oregon)	arn:aws:lambda:us-west-2:345057560386:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Africa (Cape Town)	arn:aws:lambda:af-south-1:317013901791:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Asia Pacific (Hong Kong)	arn:aws:lambda:ap-east-1:768336418462:layer:AWS-Parameters-and-Secrets-Lambda-Extension:21
Asia Pacific (Taipei)	arn:aws:lambda:ap-east-2:890742577149:layer:AWS-Parameters-and-Secrets-Lambda-Extension:7
Asia Pacific (Hyderabad) Region	arn:aws:lambda:ap-south-2:070087711984:layer:AWS-Parameters-and-Secrets-Lambda-Extension:16
Asia Pacific (Jakarta)	arn:aws:lambda:ap-southeast-3:490737872127:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19

Region	ARN
Asia Pacific (Melbourne)	arn:aws:lambda:ap-southeast-4:090732460067:layer:AWS-Parameters-and-Secrets-Lambda-Extension:9
Asia Pacific (Malaysia)	arn:aws:lambda:ap-southeast-5:381492012281:layer:AWS-Parameters-and-Secrets-Lambda-Extension:8
Asia Pacific (Mumbai)	arn:aws:lambda:ap-south-1:176022468876:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Asia Pacific (Osaka)	arn:aws:lambda:ap-northeast-3:576959938190:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Asia Pacific (Seoul)	arn:aws:lambda:ap-northeast-2:738900069198:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Asia Pacific (Singapore)	arn:aws:lambda:ap-southeast-1:044395824272:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Asia Pacific (Sydney)	arn:aws:lambda:ap-southeast-2:665172237481:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19

Region	ARN
Asia Pacific (Thailand)	arn:aws:lambda:ap-southeast-7:941377119484:layer:AWS-Parameters-and-Secrets-Lambda-Extension:7
Asia Pacific (Tokyo)	arn:aws:lambda:ap-northeast-1:133490724326:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Canada (Central)	arn:aws:lambda:ca-central-1:200266452380:layer:AWS-Parameters-and-Secrets-Lambda-Extension:21
Canada West (Calgary)	arn:aws:lambda:ca-west-1:243964427225:layer:AWS-Parameters-and-Secrets-Lambda-Extension:9
China (Beijing)	arn:aws-cn:lambda:cn-north-1:287114880934:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
China (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:287310001119:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Europe (Frankfurt)	arn:aws:lambda:eu-central-1:187925254637:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19

Region	ARN
Europe (Ireland)	arn:aws:lambda:eu-west-1:015030872274:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Europe (London)	arn:aws:lambda:eu-west-2:133256977650:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Europe (Milan)	arn:aws:lambda:eu-south-1:325218067255:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Europe (Paris)	arn:aws:lambda:eu-west-3:780235371811:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Europe (Spain) Region	arn:aws:lambda:eu-south-2:524103009944:layer:AWS-Parameters-and-Secrets-Lambda-Extension:18
Europe (Stockholm)	arn:aws:lambda:eu-north-1:427196147048:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Israel (Tel Aviv)	arn:aws:lambda:il-central-1:148806536434:layer:AWS-Parameters-and-Secrets-Lambda-Extension:9

Region	ARN
Europe (Zurich) Region	arn:aws:lambda:eu-central-2:772501565639:layer:AWS-Parameters-and-Secrets-Lambda-Extension:16
Mexico (Central) Region	arn:aws:lambda:mx-central-1:241533131596:layer:AWS-Parameters-and-Secrets-Lambda-Extension:6
Middle East (Bahrain)	arn:aws:lambda:me-south-1:832021897121:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
Middle East (UAE)	arn:aws:lambda:me-central-1:858974508948:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
South America (São Paulo)	arn:aws:lambda:sa-east-1:933737806257:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
AWS GovCloud (US-East)	arn:aws-us-gov:lambda:us-gov-east-1:129776340158:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19
AWS GovCloud (US-West)	arn:aws-us-gov:lambda:us-gov-west-1:127562683043:layer:AWS-Parameters-and-Secrets-Lambda-Extension:19

Extension ARNs for ARM64 and Mac with Apple silicon architectures

Last updated: September 03, 2025

Region	ARN
US East (Ohio)	arn:aws:lambda:us-east-2:590474943231:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:23
US East (N. Virginia)	arn:aws:lambda:us-east-1:177933569100:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:19
US West (N. California) Region	arn:aws:lambda:us-west-1:997803712105:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:16
US West (Oregon)	arn:aws:lambda:us-west-2:345057560386:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:19
Africa (Cape Town) Region	arn:aws:lambda:af-south-1:317013901791:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:16
Asia Pacific (Hong Kong) Region	arn:aws:lambda:ap-east-1:768336418462:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:18
Asia Pacific (Taipei)	arn:aws:lambda:ap-east-2:890742577149:layer:AWS-Parame

Region	ARN
	ters-and-Secrets-Lambda-Ext ension-Arm64:3
Asia Pacific (Hyderabad) Region	arn:aws:lambda:ap-south-2:0 70087711984:layer:AWS-Param eters-and-Secrets-Lambda-Ex tension-Arm64:8
Asia Pacific (Jakarta) Region	arn:aws:lambda:ap-southeast -3:490737872127:layer:AWS-P arameters-and-Secrets-Lambda- Extension-Arm64:16
Asia Pacific (Melbourne)	arn:aws:lambda:ap-southeast -4:090732460067:layer:AWS-P arameters-and-Secrets-Lambda- Extension-Arm64:8
Asia Pacific (Malaysia)	arn:aws:lambda:ap-southeast -5:381492012281:layer:AWS-P arameters-and-Secrets-Lambda- Extension-Arm64:8
Asia Pacific (Mumbai)	arn:aws:lambda:ap-south-1:1 76022468876:layer:AWS-Param eters-and-Secrets-Lambda-Ex tension-Arm64:19
Asia Pacific (Osaka)	arn:aws:lambda:ap-northeast -3:576959938190:layer:AWS-P arameters-and-Secrets-Lambda- Extension-Arm64:16

Region	ARN
Asia Pacific (Seoul) Region	arn:aws:lambda:ap-northeast-2:738900069198:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:16
Asia Pacific (Singapore)	arn:aws:lambda:ap-southeast-1:044395824272:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:19
Asia Pacific (Sydney)	arn:aws:lambda:ap-southeast-2:665172237481:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:19
Asia Pacific (Thailand)	arn:aws:lambda:ap-southeast-7:941377119484:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:6
Asia Pacific (Tokyo)	arn:aws:lambda:ap-northeast-1:133490724326:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:19
Canada (Central) Region	arn:aws:lambda:ca-central-1:200266452380:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:18
Canada West (Calgary)	arn:aws:lambda:ca-west-1:243964427225:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:8

Region	ARN
China (Beijing)	arn:aws-cn:lambda:cn-north-1:287114880934:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:8
China (Ningxia)	arn:aws-cn:lambda:cn-northwest-1:287310001119:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:8
Europe (Frankfurt)	arn:aws:lambda:eu-central-1:187925254637:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:19
Europe (Ireland)	arn:aws:lambda:eu-west-1:015030872274:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:19
Europe (London)	arn:aws:lambda:eu-west-2:133256977650:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:19
Europe (Milan) Region	arn:aws:lambda:eu-south-1:325218067255:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:16
Europe (Paris) Region	arn:aws:lambda:eu-west-3:780235371811:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:16

Region	ARN
Europe (Spain) Region	arn:aws:lambda:eu-south-2:524103009944:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:10
Europe (Stockholm) Region	arn:aws:lambda:eu-north-1:427196147048:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:16
Israel (Tel Aviv)	arn:aws:lambda:il-central-1:148806536434:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:8
Europe (Zurich) Region	arn:aws:lambda:eu-central-2:772501565639:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:8
Mexico (Central) Region	arn:aws:lambda:mx-central-1:241533131596:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:5
Middle East (Bahrain) Region	arn:aws:lambda:me-south-1:832021897121:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:16
Middle East (UAE)	arn:aws:lambda:me-central-1:858974508948:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:8

Region	ARN
South America (São Paulo) Region	`arn:aws:lambda:sa-east-1:933737806257:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:16`
AWS GovCloud (US-East)	`arn:aws-us-gov:lambda:us-gov-east-1:129776340158:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:8`
AWS GovCloud (US-West)	`arn:aws-us-gov:lambda:us-gov-west-1:127562683043:layer:AWS-Parameters-and-Secrets-Lambda-Extension-Arm64:8`

Integration with other products and services

AWS Systems Manager has built-in integration for the products and services shown in the following table.

Ansible	Ansible is an IT automation platform that makes your applications and systems easier to deploy. Systems Manager provides the Systems Manager document (SSM document) `AWS-ApplyAnsiblePlaybooks` which allows you to create State Manager associations that run Ansible playbooks. **Learn more** Creating associations that run Ansible playbooks

Chef	Chef is an IT automation tool that makes your applications and systems easier to deploy. Systems Manager provides the `AWS-ApplyChefRecipes` SSM document, which allows you to create associations in State Manager, a tool in AWS Systems Manager, that run Chef recipes. **Learn more** Creating associations that run Chef recipes Systems Manager also integrates with Chef InSpec profiles, allowing you to run compliance scans and view compliant and noncompliant nodes. **Learn more** Using Chef InSpec profiles with Systems Manager Compliance
GitHub	GitHub provides hosting for software development version control and collaboration. Systems Manager provides the SSM document `AWS-RunDocument`, which allows you to run other SSM documents stored in GitHub, and the SSM document `AWS-RunRemoteScript`, which allows you to run scripts stored in GitHub. **Learn more** • Running documents from remote locations • Running scripts from GitHub

Jenkins	Jenkins is an open-source automation server that allows developers to reliably build, test, and deploy their software. Automation, a tool in Systems Manager, can be used as a post-build step to pre-install application releases into Amazon Machine Images (AMIs). **Learn more** Updating AMIs using Automation and Jenkins
ServiceNow	ServiceNow is an enterprise service management system that allows you to manage your IT services and operations. Automation, Change Manager, Incident Manager, and OpsCenter, all tools in Systems Manager, integrate with ServiceNow by using the AWS Service Management Connector. With this integration, you can view, create, update, add correspondence, and resolve AWS Support cases from ServiceNow. **Learn more** Integrating with ServiceNow

Terraform	HashiCorp Terraform is an open-source *infrastructure as code* (IaC) software tool that provides a command line interface (CLI) workflow to manage various cloud services. For Systems Manager, you can use Terraform to manage or provision the following: **Resources** • aws_ssm_activation • aws_ssm_association • aws_ssm_default_patch_baseline • aws_ssm_document • aws_ssm_maintenance_window • aws_ssm_maintenance_window_target • aws_ssm_maintenance_window_task • aws_ssm_parameter • aws_ssm_patch_baseline • aws_ssm_patch_group • aws_ssm_resource_data_sync • aws_ssm_service_setting **Data sources** • aws_ssm_document • aws_ssm_instances • ssm_maintenance_windows • aws_ssm_parameter • aws_ssm_parameters_by_path • aws_ssm_patch_baseline

Topics

- Running scripts from GitHub

- Using Chef InSpec profiles with Systems Manager Compliance
- Integrating with ServiceNow

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.

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,tools in 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.
- GitHub Enterprise repositories are not supported.

Topics

- Run Ansible Playbooks from GitHub
- Run Python scripts from GitHub

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 Parameter Store parameters in Systems Manager.

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**.
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:bbc1ddb94...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.

> **Tip**
>
> If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

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 Installing or updating the latest version of the AWS CLI.

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-02573cafcfEXAMPLE" \
    --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**.
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.

> **Tip**
>
> If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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.

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 Installing or updating the latest version of the AWS CLI.

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-02573cafcfEXAMPLE" --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. Chef 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 tool in 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 *only* for Amazon Elastic Compute Cloud (Amazon EC2) instances that you manage with Systems Manager. On-premises servers or virtual machines (VMs) are not supported. 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 ChefChef 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 tool in AWS Systems Manager, for on-demand scans, or you can schedule regular Compliance scans by using State Manager, a tool in 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**.
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.
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.

   > **Tip**
   >
   > If a managed node you expect to see isn't listed, see Troubleshooting managed node availability 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 (hybrid-activated machines) assigned to the instance, not those of the IAM user performing this task. For more information, see Configure instance permissions required for Systems Manager or Create an IAM service role for a hybrid environment. 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 Installing or updating the latest version of the AWS CLI.

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.aws-api-domain/amzn-s3-demo-
   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.aws-api-
domain/amzn-s3-demo-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
```

Integrating with ServiceNow

ServiceNow provides a cloud-based service management system to create and manage organization-level workflows, such as for IT services, ticketing systems, and support. The AWS Service Management Connector integrates ServiceNow with Systems Manager to provision, manage, and operate AWS resources from ServiceNow. You can use the AWS Service Management Connector to integrate ServiceNow with Automation, Change Manager, Incident Manager, and OpsCenter, all tools in AWS Systems Manager.

You can perform the following tasks using ServiceNow:

- Run automation playbooks from Systems Manager.
- View, update, and resolve incidents from Systems Manager OpsItems.
- View and manage operational items, such as incidents, through Systems Manager OpsCenter.
- View and run Systems Manager change requests from a curated list of pre-approved change templates.
- Manage and resolve incidents involving AWS hosted applications by integrating with Incident Manager.

Note

For information about how to integrate with ServiceNow, see Configuring AWS service integrations in the *AWS Service Management Connector Administrator Guide*.

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

- Using this service with an AWS SDK
- Reference: Amazon S3 buckets for patching operations
- Reference: Amazon EventBridge event patterns and types for Systems Manager
- Reference: Cron and rate expressions for Systems Manager
- Reference: ec2messages, ssmmessages, and other API operations
- Reference: Date and time string formats for Systems Manager

Using this service with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples
AWS CLI	AWS CLI code examples
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS Tools for PowerShell	AWS Tools for PowerShell code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples
AWS SDK for Rust	AWS SDK for Rust code examples

SDK documentation	Code examples
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP code examples
AWS SDK for Swift	AWS SDK for Swift code examples

Example availability

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

Reference: Amazon S3 buckets for patching operations

In the course of performing various Patch Manager patching operations, AWS Systems Manager Agent (SSM Agent) accesses certain Amazon Simple Storage Service (Amazon S3) buckets that are owned and maintained by Amazon Web Services (AWS). 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 non-EC2 machines in a hybrid and multicloud environment. Otherwise, your resources can't access these public buckets.

This reference topic lists the patching-related buckets for each supported AWS Region.

For information about specifying these S3 buckets in EC2 instance profiles, see SSM Agent communications with AWS managed S3 buckets.

For information about using VPC endpoints with Systems Manager, see Improve the security of EC2 instances by using VPC endpoints for Systems Manager.

Topics

- Buckets containing SSM Command documents for patching operations (Linux and Windows Server)
- Buckets containing SSM Command documents for patching operations (macOS)
- Buckets containing AWS managed patch baseline snapshots

Buckets containing SSM Command documents for patching operations (Linux and Windows Server)

Buckets with the format `aws-patch-manager-`*`region`*`-`*`unique-suffix` contain the following documents used by Patch Manager patching operations on the Linux and Windows Server operating systems:

- `AWS-RunPatchBaseline`
- `AWS-RunPatchBaselineAssociation`
- `AWS-RunPatchBaselineWithHooks`
- `AWS-InstanceRebootWithHooks`
- `AWS-PatchAsgInstance`
- `AWS-PatchInstanceWithRollback`

Region name	Region code	**`aws-patch-manager-`*`region`*`-`*`suffix`* bucket**
US East (Ohio)	us-east-2	aws-patch-manager-us-east-2-552881074
US East (N. Virginia)	us-east-1	aws-patch-manager-us-east-1-1970c647d
US West (N. California)	us-west-1	aws-patch-manager-us-west-1-8badb4304
US West (Oregon)	us-west-2	aws-patch-manager-us-west-2-34d7f99f8
Africa (Cape Town)	af-south-1	aws-patch-manager-af-south-1-bdd5f65a9
Asia Pacific (Hong Kong)	ap-east-1	aws-patch-manager-ap-east-1-632356271

Region name	Region code	aws-patch-manager-*region*-*suffix* bucket
Asia Pacific (Hyderabad)	ap-south-2	aws-patch-manager-ap-south-2-32f4b4128
Asia Pacific (Jakarta)	ap-southeast-3	aws-patch-manager-ap-southeast-3-aa48fc462
Asia Pacific (Melbourne)	ap-southeast-4	aws-patch-manager-ap-southeast-4-01e2c40d3
Asia Pacific (Mumbai)	ap-south-1	aws-patch-manager-ap-south-1-cb7c62ff9
Asia Pacific (Osaka)	ap-northeast-3	aws-patch-manager-ap-northeast-3-67373598a
Asia Pacific (Seoul)	ap-northeast-2	aws-patch-manager-ap-northeast-2-10467995c
Asia Pacific (Singapore)	ap-southeast-1	aws-patch-manager-ap-southeast-1-7fdfd9ef7
Asia Pacific (Sydney)	ap-southeast-2	aws-patch-manager-ap-southeast-2-17283a275
Asia Pacific (Tokyo)	ap-northeast-1	aws-patch-manager-ap-northeast-1-4849fa78f
Canada (Central)	ca-central-1	aws-patch-manager-ca-central-1-3148e69e3
Canada West (Calgary)	ca-west-1	aws-patch-manager-ca-west-1-9e3a4b2f9
Europe (Frankfurt)	eu-central-1	aws-patch-manager-eu-central-1-9163fdaaf

Region name	Region code	aws-patch-manager-*region*-*suffix* bucket
Europe (Ireland)	eu-west-1	aws-patch-manager-eu-west-1-5522fb710
Europe (London)	eu-west-2	aws-patch-manager-eu-west-2-902a2bc74
Europe (Milan)	eu-south-1	aws-patch-manager-eu-south-1-c52f3f594
Europe (Paris)	eu-west-3	aws-patch-manager-eu-west-3-29bf85721
Europe (Spain)	eu-south-2	aws-patch-manager-eu-south-2-a4cf248b1
Europe (Stockholm)	eu-north-1	aws-patch-manager-eu-north-1-795879e9b
Europe (Zurich)	eu-central-2	aws-patch-manager-eu-central-2-184ce43c8
Israel (Tel Aviv)	il-central-1	aws-patch-manager-il-central-1-e221cb57b
Middle East (Bahrain)	me-south-1	aws-patch-manager-me-south-1-a53fc9dce
Middle East (UAE)	me-central-1	aws-patch-manager-me-central-1-2932f2f80
South America (São Paulo)	sa-east-1	aws-patch-manager-sa-east-1-ddf4b6a09

Buckets containing SSM Command documents for patching operations (macOS)

Buckets with the format `aws-patchmanager-macos-`*`region`*`-`*`unique-suffix` contain the following documents used by Patch Manager patching operations on the macOS operating system:

- `AWS-RunPatchBaseline`
- `AWS-RunPatchBaselineAssociation`
- `AWS-RunPatchBaselineWithHooks`
- `AWS-InstanceRebootWithHooks`
- `AWS-PatchAsgInstance`
- `AWS-PatchInstanceWithRollback`

Region name	Region code	**`aws-patchmanager-macos-`*`region`*`-`*`suffix`** **bucket**
US East (Ohio)	us-east-2	aws-patchmanager-macos-us-east-2-552881074
US East (N. Virginia)	us-east-1	aws-patchmanager-macos-us-east-1-1970c647d
US West (N. California)	us-west-1	aws-patchmanager-macos-us-west-1-8badb4304
US West (Oregon)	us-west-2	aws-patchmanager-macos-us-west-2-34d7f99f8
Africa (Cape Town)	af-south-1	aws-patchmanager-macos-af-south-1-bdd5f65a9
Asia Pacific (Hong Kong)	ap-east-1	aws-patchmanager-macos-ap-east-1-632356271

Region name	Region code	aws-patchmanager-macos- *region-suffix* bucket
Asia Pacific (Hyderabad)	ap-south-2	aws-patchmanager-macos-ap-south-2-32f4b4128
Asia Pacific (Jakarta)	ap-southeast-3	aws-patchmanager-macos-ap-southeast-3-aa48fc462
Asia Pacific (Melbourne)	ap-southeast-4	aws-patchmanager-macos-ap-southeast-4-01e2c40d3
Asia Pacific (Mumbai)	ap-south-1	aws-patchmanager-macos-ap-south-1-cb7c62ff9
Asia Pacific (Osaka)	ap-northeast-3	aws-patchmanager-macos-ap-northeast-3-67373598a
Asia Pacific (Seoul)	ap-northeast-2	aws-patchmanager-macos-ap-northeast-2-10467995c
Asia Pacific (Singapore)	ap-southeast-1	aws-patchmanager-macos-ap-southeast-1-7fdfd9ef7
Asia Pacific (Sydney)	ap-southeast-2	aws-patchmanager-macos-ap-southeast-2-17283a275
Asia Pacific (Tokyo)	ap-northeast-1	aws-patchmanager-macos-ap-northeast-1-4849fa78f
Canada (Central)	ca-central-1	aws-patchmanager-macos-ca-central-1-3148e69e3
Canada West (Calgary)	ca-west-1	aws-patchmanager-macos-ca-west-1-9e3a4b2f9
Europe (Frankfurt)	eu-central-1	aws-patchmanager-macos-eu-central-1-9163fdaaf

Region name	Region code	aws-patchmanager-macos- *region-suffix* bucket
Europe (Ireland)	eu-west-1	aws-patchmanager-macos-eu-west-1-5522fb710
Europe (London)	eu-west-2	aws-patchmanager-macos-eu-west-2-902a2bc74
Europe (Milan)	eu-south-1	aws-patchmanager-macos-eu-south-1-c52f3f594
Europe (Paris)	eu-west-3	aws-patchmanager-macos-eu-west-3-29bf85721
Europe (Spain)	eu-south-2	aws-patchmanager-macos-eu-south-2-a4cf248b1
Europe (Stockholm)	eu-north-1	aws-patchmanager-macos-eu-north-1-795879e9b
Europe (Zurich)	eu-central-2	aws-patchmanager-macos-eu-central-2-184ce43c8
Israel (Tel Aviv)	il-central-1	aws-patchmanager-macos-il-central-1-e221cb57b
Middle East (Bahrain)	me-south-1	aws-patchmanager-macos-me-south-1-a53fc9dce
Middle East (UAE)	me-central-1	aws-patchmanager-macos-me-central-1-2932f2f80
South America (São Paulo)	sa-east-1	aws-patchmanager-macos-sa-east-1-ddf4b6a09

Buckets containing AWS managed patch baseline snapshots

Buckets with the format `patch-baseline-snapshot-`*`region`* or `patch-baseline-snapshot-`*`region`*`-`*`unique-suffix`* contain AWS managed patch baseline snapshots. Access to this S3 bucket is required if you use any of the following SSM documents:

- `AWS-RunPatchBaseline`
- `AWS-RunPatchBaselineAssociation`
- `AWS-RunPatchBaselineWithHooks`
- `AWS-ApplyPatchBaseline` (a legacy SSM Document)

Region name	Region code	`patch-baseline-snapshot-*` bucket
US East (Ohio)	us-east-2	patch-baseline-snapshot-us-east-2
US East (N. Virginia)	us-east-1	patch-baseline-snapshot-us-east-1
US West (N. California)	us-west-1	patch-baseline-snapshot-us-west-1
US West (Oregon)	us-west-2	patch-baseline-snapshot-us-west-2
Africa (Cape Town)	af-south-1	patch-baseline-snapshot-af-south-1-tbxdb5b9
Asia Pacific (Hong Kong)	ap-east-1	patch-baseline-snapshot-ap-east-1
Asia Pacific (Hyderabad)	ap-south-2	patch-baseline-snapshot-ap-south-2-50209442
Asia Pacific (Jakarta)	ap-southeast-3	patch-baseline-snapshot-ap-southeast-3-be0a3174

Region name	Region code	`patch-baseline-snapshot-*` bucket
Asia Pacific (Melbourne)	ap-southeast-4	patch-baseline-snapshot-ap-southeast-4-dc6f76ce
Asia Pacific (Mumbai)	ap-south-1	patch-baseline-snapshot-ap-south-1
Asia Pacific (Osaka)	ap-northeast-3	patch-baseline-snapshot-ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2	patch-baseline-snapshot-ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1	patch-baseline-snapshot-ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2	patch-baseline-snapshot-ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1	patch-baseline-snapshot-ap-northeast-1
Canada (Central)	ca-central-1	patch-baseline-snapshot-ca-central-1
Canada West (Calgary)	ca-west-1	patch-baseline-snapshot-ca-west-1
Europe (Frankfurt)	eu-central-1	patch-baseline-snapshot-eu-central-1
Europe (Ireland)	eu-west-1	patch-baseline-snapshot-eu-west-1
Europe (London)	eu-west-2	patch-baseline-snapshot-eu-west-2

Region name	Region code	patch-baseline-snapshot-* bucket
Europe (Milan)	eu-south-1	patch-baseline-snapshot-eu-south-1
Europe (Paris)	eu-west-3	patch-baseline-snapshot-eu-west-3
Europe (Spain)	eu-south-2	patch-baseline-snapshot-eu-south-2-df2c9d70
Europe (Stockholm)	eu-north-1	patch-baseline-snapshot-eu-north-1
Europe (Zurich)	eu-central-2	patch-baseline-snapshot-eu-central-2
Israel (Tel Aviv)	il-central-1	patch-baseline-snapshot-il-central-1
Middle East (Bahrain)	me-south-1	patch-baseline-snapshot-me-south-1-uduvl7q8
Middle East (UAE)	me-central-1	patch-baseline-snapshot-me-central-1
South America (São Paulo)	sa-east-1	patch-baseline-snapshot-sa-east-1

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
- Amazon EventBridge event examples for Systems Manager
- 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: • Approved • Canceled • Failed • PendingApproval

Event type name	Description of events you can add to a rule
	• PendingChangeCalendarOverride • Rejected • Scheduled • Success • TimedOut
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: • Canceled • Failed • Success • TimedOut

Event type: Change Calendar

Event type name	Description of events you can add to a rule
Calendar State Change	The state of a Change Calendar changes. You can add one or both of the following state changes to an event rule: • OPEN • CLOSED State changes for calendars shared from other AWS accounts aren't supported.

Event type: Change Manager

Event type name	Description of events you can add to a rule
Change Request Status Update	The state of a Change Manager change request. You can use the following states in an event rule: • Approved • Rejected • InProgress

Event type: Configuration Compliance

Event type name	Description of events you can add to a rule
Configuration Compliance State Change	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: • compliant • non_compliant

Event type: Inventory

Event type name	Description of events you can add to a rule
Inventory Resource State Change	The deletion of custom inventory and a PutInventory call that uses an old schema version. You can add one or more of the following state changes to an event rule:

Event type name	Description of events you can add to a rule
	• Custom inventory type deleted event on a specific node. EventBridge sends one event per node per custom InventoryType. • Custom inventory type deleted event for all nodes. • 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. For more information, see Using EventBridge to monitor Inventory events.

Event type: Maintenance Window

Event type name	Description of events you can add to a rule
Maintenance Window Status-change Notification	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: • DISABLED • ENABLED
Maintenance Window Target Registration Notification	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: • DEREGISTERED • REGISTERED • UPDATED

Event type name	Description of events you can add to a rule
Maintenance Window Execution State-change Notification	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: • CANCELLED • CANCELLING • FAILED • IN_PROGRESS • PENDING • SKIPPED_OVERLAPPING • SUCCESS • TIMED_OUT
Maintenance Window Task Execution State-change Notification	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: • CANCELLED • CANCELLING • FAILED • IN_PROGRESS • SUCCESS • TIMED_OUT

Event type name	Description of events you can add to a rule
Maintenance Window Task Target Invocation State-change Notification	The state of a maintenance window task on a specific target changes. 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: • CANCELLED • CANCELLING • FAILED • IN_PROGRESS • SUCCESS • TIMED_OUT 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.
Maintenance Window Task Registration Notification	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: • DEREGISTERED • REGISTERED • UPDATED

Event type: OpsCenter

Event type name	Description of events you can add to a rule
OpsItem Create	Occurs when an OpsItem is created. You can add rules for one of the following OpsItem types: • /aws/issue • /aws/task • /aws/insight • /aws/actionitem
OpsItem Update	Occurs when an OpsItem is updated. You can add rules for one of the following OpsItem types: • /aws/issue • /aws/task • /aws/insight • /aws/actionitem

Event type: Parameter Store

Event type name	Description of events you can add to a rule
Parameter Store Change	The state of a parameter changes. You can add one or more of the following state changes to an event rule: • Create • Update • Delete • LabelParameterVersion

Event type name	Description of events you can add to a rule
	For more information, see Configuring EventBridge rules for parameters and parameter policies.
Parameter Store Policy Action	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: • Expiration • ExpirationNotification • NoChangeNotification For more information, see Configuring EventBridge rules for parameters and parameter policies.

Event type: Run Command

Event type name	Description of events you can add to a rule
EC2 Command Invocation Status-change Notification	The status of a command sent to an individua l managed instance changes. You can add one or more of the following status changes to an event rule: • Success • InProgress • TimedOut • Canceled • Failed
EC2 Command Status-change Notification	The overall status of a command changes. You can add one or more of the following status changes to an event rule:

Event type name	Description of events you can add to a rule
	• Success • InProgress • TimedOut • Canceled • Failed

Event type: State Manager

Event type name	Description of events you can add to a rule
EC2 State Manager Association State Change	The *overall* 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: • Failed • Pending • Success
EC2 State Manager Instance Association State Change	The state of a *single* managed instance that is targeted by an Association changes. You can add one or more of the following state changes to an event rule: • Failed • Pending • Success

Reference: Cron and rate expressions for Systems Manager

When you create a State Manager association or a maintenance window in AWS Systems Manager, you specify a schedule for when the window or the association should run. You can specify a schedule as either a time-based entry, called a *cron expression*, or a frequency-based entry, called a *rate expression*.

General information about cron and rate expressions

The following information applies to cron and rate expressions for both maintenance windows and associations.

Single-run schedules

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. For example: `"at(2020-07-07T15:55:00)"`

Schedule offsets

Associations and maintenance windows 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.

Maintenance window example

In the following command, the cron expression schedules a maintenance window to run the third Tuesday of every month at 11:30 PM. However, because the schedule offset is 2, the maintenance window won't run until 11:30 PM two days later.

```
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
```

Association example

In the following command, the cron expression schedules an association to run the second Thursday of each month. However, because the schedule offset is 3, the assocation won't run until the next Sunday, three days later.

```
aws ssm create-association \
    --name "AWS-UpdateSSMAgent" \
    --targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f" \
    --schedule-expression "cron(0 0 ? * THU#2 *)"  \
    --schedule-offset 3
```

```
--apply-only-at-cron-interval
```

> **Note**
>
> To use an offset with an assocation, you must specify the `--apply-only-at-cron-interval` option. This option tells the system not to run an association immediately after you create it.

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.

Required fields

Cron expressions for maintenance windows have six required fields. Cron expressions for associations have five. (State Manager doesn't currently support specifying months in cron expressions for associations.) An additional 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

Minutes	Hours	Day of month	Month	Day of week	Year	Meaning
						(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.

Supported values for cron expressions

Field	Values	Wildcards
Minutes	0-59	, - * /
Hours	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
Month (maintenance windows only)	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 for cron expressions

The following table shows the wildcard values that cron expressions support.

> **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.

Supported wildcards for cron expressions

Wildcard	Description
,	The , (comma) wildcard includes additiona l 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

Wildcard	Description
	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 Thursday of the month.

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

Unit	`minute`
	
	`minutes`
	
	`hour`

Field	Values

	`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.

Topics

- Cron and rate expressions for associations
- Cron and rate expressions for maintenance windows

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, or every specified day and time of the week; a specific day in a specific week of the month, 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 * * * ? *)`
- For an association that collects metadata for Inventory, a tool in AWS Systems Manager, we recommend using a rate expression.
- State Manager doesn't currently support specifying months in cron expressions for associations.

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 use `--apply-only-at-cron-interval` parameter from the command line. This option tells State Manager not to run an association immediately after you create it.

The following table presents cron examples for associations.

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

Example	Details
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

AWS CLI examples for associations

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.

> **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.

```
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
```

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

Example	Details
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

AWS CLI examples for maintenance windows

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-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
```

More info

CRON expression at the *Wikipedia website*

Reference: ec2messages, ssmmessages, and other API operations

If you monitor API operations, you might see calls to the following operations:

- ec2messages:AcknowledgeMessage
- ec2messages:DeleteMessage
- ec2messages:FailMessage
- ec2messages:GetEndpoint
- ec2messages:GetMessages
- ec2messages:SendReply
- ssmmessages:CreateControlChannel
- ssmmessages:CreateDataChannel
- ssmmessages:OpenControlChannel
- ssmmessages:OpenDataChannel

- ssm:DescribeDocumentParameters
- ssm:DescribeInstanceProperties
- ssm:GetCalendar
- ssm:GetManifest
- ssm:ListInstanceAssociations
- ssm:PutCalendar
- ssm:PutConfigurePackageResult
- ssm:RegisterManagedInstance
- ssm:RequestManagedInstanceRoleToken
- ssm:UpdateInstanceAssociationStatus
- ssm:UpdateInstanceInformation
- ssm:UpdateManagedInstancePublicKey

These are special operations used by AWS Systems Manager, as described in the rest of this topic.

Agent-related API operations (ssmmessages and ec2messages endpoints)

ssmmessages API operations

Systems Manager uses the ssmmessages endpoint for the following types of API operations:

- Operations from Systems Manager Agent (SSM Agent) to the Systems Manager service in the cloud.
- Operations from SSM Agent to Session Manager, a tool in AWS Systems Manager, in the cloud. This endpoint is required to create and delete session channels with the Session Manager service in the cloud. Additionally, if connectivity is allowed, SSM Agent receives Command documents through this Amazon Message Gateway Service. If connectivity is not allowed, SSM Agent receives Command documents through the Amazon Message Delivery Service. For more information, see Actions, resources, and condition keys for Amazon Message Gateway Service.
- Operations from Run Command.

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.

> **Important**
>
> `ec2messages:*` API operations are supported only in AWS Regions that launched before 2024. In Regions launched in 2024 and later, only `ssmmessages:*` API operations are supported.

Endpoint connection precedence

Beginning with version 3.3.40.0 of SSM Agent, Systems Manager began using the `ssmmessages:*` endpoint (Amazon Message Gateway Service) whenever available instead of the `ec2messages:*` endpoint (Amazon Message Delivery Service).

If you provide access to `ssmmessages:*` in your AWS Identity and Access Management (IAM) permission policies, SSM Agent connects to the `ssmmessages:*` endpoint, even if your IAM instance profile is configured to allow both endpoints. This includes policies for IAM instance profiles and IAM service roles you have created yourself, and for IAM instance profiles created by the Quick Setup Host management configuration and Default Host Management Configuration.

If you have provided permissions for both endpoints and monitor API operations using, for example, CloudWatch Metrics, you will see no calls to `ec2messages:*`.

For AWS Regions launched before 2024: You can safely remove `ec2messages:*` permissions from your policies at this time.

Endpoint connection failover

For AWS Regions launched before 2024 only: If your IAM instance profile does not provide permissions for `ssmmessages:*` at the time the agent starts, but only `ec2messages:*`, SSM Agent connects to the `ec2messages:*` endpoint. If you have both `ssmmessages:*` and `ec2messages:*` at the time SSM Agent starts, but remove `ssmmessages:*` after the agent starts, SSM Agent soon switches the connection to the `ec2messages:*` endpoint. For Regions launched in 2024 and later, only the `ssmmessages:*` endpoint is supported.

For more information about the `ssmmessages` and `ec2messages:*` endpoints, see the following topics in the *AWS Service Authorization Reference*.

- Actions, resources, and condition keys for Amazon Message Gateway Service (`ssmmessages`).
- Actions, resources, and condition keys for Amazon Message Delivery Service (`ec2messages:*`)

`ssm:*` namespace instance-related API operations

`DescribeDocumentParameters`

Systems Manager runs this API operation to render specific nodes in the Amazon EC2 console. Results of the `DescribeDocumentParameters` operation are displayed in the Documents node.

`DescribeInstanceProperties`

Systems Manager runs this API operations to render specific nodes in the Amazon EC2 console. Results of the `DescribeInstanceProperties` operation are displayed in the Fleet Manager node.

`GetCalendar`

Systems Manager runs this API operation to render Change Calendar type documents in the Change Calendar console.

`GetManifest`

SSM Agent runs this API operation to determine system requirements for installing or updating a specified version of an AWS Systems Manager Distributor package. This is a legacy API operation and not available in AWS Regions launched after 2017.

`ListInstanceAssociations`

SSM Agent runs this API operation to see if a new State Manager association is available. This API operation is required for State Manager to function.

`PutCalendar`

Systems Manager runs this API operation to update Change Calendar type documents in the Change Calendar console.

`PutConfigurePackageResult`

SSM Agent runs this API operation to publish installation error and latency metrics for public Distributor packages to the package owner's account.

`RegisterManagedInstance`

SSM Agent runs this API operation for the following scenarios:

- To register an on-premises server or virtual machine (VM) with Systems Manager as a managed instance using an activation code and ID.
- To register AWS IoT Greengrass Version 2 credentials.

This operation is also called by Amazon EC2 instances running SSM Agent version 3.1.x or later.

`RequestManagedInstanceRoleToken`

SSM Agent runs this API operation to retrieve temporary credentials to access the managed node.

`UpdateInstanceAssociationStatus`

SSM Agent runs this API operation to update an association. This API operation is required for State Manager, a tool in AWS Systems Manager, to function.

`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.

`UpdateManagedInstancePublicKey`

SSM Agent runs this API operation to provide the public key after rotating the key pair on the managed node. The public key is used to authenticate the requests, signed with the private key, to get temporary credentials from Systems Manager.

ssm:* namespace other API operations

`ExecuteApi`

Systems Manager delegated administrators who manage OpsItems in OpsCenter require access to this API action so they can view related resource details about OpsItems across multiple AWS accounts. Specifically, this API gives a delegated administrator permission to view the following OpsItem details in the AWS Management Console: the OpsItem description, tags, AWS CloudFormation template, AWS Config changes, CloudWatch Logs alarms, and AWS CloudTrail events. For more information about working with OpsItems across accounts, see (Optional) Manually set up OpsCenter to centrally manage OpsItems across accounts. For more

information about related resource details for OpsItems, see Adding related resources to an OpsItem.

Reference: Date and time string formats 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.

```
2024-05-08T15:16:43Z
```

This represents May 8, 2024 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 tools. 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

- Create self-service Automation runbooks for infrastructure.
- Use Automation, a tool in 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 using the `AWS-UpdateLinuxAmi` and `AWS-UpdateWindowsAmi` Automation runbooks, or using custom Automation runbooks that you create.

Compliance

- As a security best practice, we recommend that you update the AWS Identity and Access Management (IAM) role used by your managed nodes to restrict the node's ability to use the PutComplianceItems API action. This API action registers a compliance type and other compliance details on a designated resource, such as an Amazon EC2 instance or a managed node. For more information, see Configuring permissions for Compliance.

Inventory

- Use Inventory, a tool in AWS Systems Manager, with AWS Config to audit your application configurations over time.

Maintenance Windows

- 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, tools of AWS Systems Manager, see Choosing between State Manager and Maintenance Windows.

Parameter Store

- Use Parameter Store, a tool in 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.

Patch Manager

- Use Patch Manager, a tool in AWS Systems Manager, to roll out patches at scale and increase fleet compliance visibility across your nodes.
- Integrate Patch Manager with AWS Security Hub 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.
- Use only one method at a time for scanning managed nodes for patch compliance to avoid unintentionally overwriting compliance data.

Run Command

- Manage Instances at Scale without SSH Access Using EC2 Run Command.
- Audit all API calls made by or on behalf of Run Command, a tool in 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 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 Parameter Store parameters using IAM policies.

> **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.
- Use fine-grained access permissions for Run Command (and all Systems Manager tools) by using AWS Identity and Access Management (IAM) policies.

Session Manager

- Log session activity in your AWS account using AWS CloudTrail.
- Log session data in your AWS account using Amazon CloudWatch Logs or Amazon S3.
- Control user session access to instances.
- Restrict access to commands in a session.
- Turn off or turn on ssm-user account administrative permissions.

State Manager

- Update SSM Agent at least once a month using the pre-configured `AWS-UpdateSSMAgent` document.
- (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.
- For information about the differences between State Manager and Maintenance Windows, see Choosing between State Manager and Maintenance Windows.

Managed nodes

- 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 Set the time for your Amazon EC2 instance.
- On Linux managed nodes, verify the signature of SSM Agent.

More info

- Security best practices for Systems Manager

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 tool 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 tags, Resource Groups, or AWS CloudFormation stacks.
Automation	If you create AWS resources by using Systems Manager Automation, you must manually

Capability or artifact	Details
	delete those resources by using the correspon ding AWS Management Console. If you created a custom runbook, you can delete the underlying SSM document. For more informati on, see Deleting custom SSM documents.
Change Calendar	You can delete a change calendar and a change calendar event. For more information, see Deleting a change calendar and Deleting a Change Calendar event.
Change Manager	You can delete a change template. For more information, see Deleting change templates.
Compliance	Systems Manager Compliance automatic ally displays compliance data about Patch Manager patching and State Manager associati ons. You can't delete this data. If you configure d a resource data sync to centralize complianc e data in an S3 bucket, you can delete the sync. For more information, see Deleting a resource data sync for Compliance.
Distributor	You can delete packages in Distributor. For more information, see Delete a Distributor package.

Capability or artifact	Details
Explorer	You can disconnect from the sources from which Explorer collects OpsData. For more information, see Editing Systems Manager Explorer data sources. You can also delete a resource data sync used by Explorer to aggregate OpsData and OpsItems from multiple AWS Regions and accounts to a single Amazon Simple Storage Service (Amazon S3) bucket. For more information, see Deleting a Systems Manager Explorer resource data sync. For information about deleting an S3 bucket, see Deleting a bucket in the *Amazon Simple Email Service Developer Guide*.
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 Terminate your instance (Linux) and Terminate your instance (Windows) .

Capability or artifact	Details
Inventory	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 informati on, see 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, then we also recommend deleting resource data syncs used for inventory data collection. For more informati on, see Deleting an Inventory resource data sync.
Maintenance Windows	You can delete a maintenance window, a maintenance window target, and a maintenan ce window task. For more information, see Update or delete maintenance window resources using the console.
OpsCenter	You can delete an individual OpsItem by calling the DeleteOpsItem API operation using the AWS Command Line Interface or the AWS SDK. You can't delete an OpsItem in the AWS Management Console. For more information, see Delete OpsItems.
Parameter Store	You can delete a parameter that you have created. For more information, see Deleting parameters from Parameter Store.
Patch Manager	You can delete a custom patch baseline. For more information, see Updating or deleting a custom patch baseline.

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 Deleting associations.
Run Command	After a command finishes processin g, information about it is stored in the **Command history** tab. You can't delete information from the **Command history** tab.
Automation	After an automation finishes processing, information about it is stored in the **Execution s** tab. You can't delete information from the **Executions** tab.
Service-linked role	Systems Manager automatically creates service-linked roles for some tools. You can delete these roles. For more information, see Deleting the `AWSServiceRoleForA mazonSSM` service-linked role for Systems Manager.
Session Manager	Session Manager doesn't retain data about your resources after you terminate a session. To terminate a session, see End a session.
SSM Agent	You can manually uninstall SSM Agent from your nodes. For more information, see the following topics. • Linux: Manually installing and uninstalling SSM Agent on EC2 instances for Linux • macOS: Manually installing and uninstalling SSM Agent on EC2 instances for macOS • Windows Server: Open **Control panel** and then choose **Add/remove programs**.

Capability or artifact	Details
State Manager	You can delete an association. For more information, see Deleting associations.
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 Deleting custom SSM documents.

Choosing between State Manager and Maintenance Windows

State Manager and Maintenance Windows, both tools in 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 tool in AWS Systems Manager, sets and maintains the targeted 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 tool 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 required 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 tool in 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 in a single Region that are 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.

  > **Note**
  >
  > Using a maintenance window, your patching operation is limited to a single Region in a single account. Using a patch policy created in Quick Setup, a tool in Systems Manager, you can instead configure patching for some or all accounts and Regions in an organization created in AWS Organizations. For more information, see Patch policy configurations in Quick Setup.

- **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
AWS CloudFormation integration	AWS CloudFormation templates support State Manager associations.	AWS CloudFormation templates support maintenan

Feature	State Manager	Maintenance Windows
		ce windows, window targets, and window tasks.
Compliance	Every State Manager associati on reports compliance with respect to the required state of the targeted resource. You can use the Compliance Dashboard to aggregate and view the reported complianc e.	Not applicable.
Configuration Management integration	State Manager supports external targeted state solutions such as Microsoft PowerShell Desired State Configuration (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.	Not applicable.

Feature	State Manager	Maintenance Windows
Documents	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 (required state for managed nodes).
Monitoring	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.

Feature	State Manager	Maintenance Windows
Priorities within tasks	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.
Safety controls	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.

Feature	State Manager	Maintenance Windows
Scheduling	You can run State Manager associations on demand, at a particular cron interval, at a given rate, or after they're created. This is useful if you want to maintain the required state of your resources in a consistent and timely manner. **Important** Cron expressions for State Manager associations do not support the months field, such as `03` or `MAR` for the month of March. If you require monthly or quarterly configuration updates, a maintenance window can best meet your needs. For more information, see Reference: Cron and rate expressions for Systems Manager.	Maintenance windows support several scheduling options including `at` expressions (for example, `"at(2021-07-07T13:15:30)"`), 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.

Feature	State Manager	Maintenance Windows
Targeting	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.
Tasks within maintenance windows	Not applicable.	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. Overall, maintenance windows support four task types: • AWS Systems Manager Run Command commands • AWS Systems Manager Automation workflows • AWS Lambda functions • AWS Step Functions tasks

Related information

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

Pricing

Some Systems Manager tools charge a fee. For more information, see AWS Systems Manager pricing.

AWS Systems Manager documentation library

AWS Systems Manager Documentation – Access all user documentation for Systems Manager, including AWS AppConfig, Incident Manager, and AWS Systems Manager for SAP.

AWS re:Post

AWS re:Post – AWS managed question and answer (Q & A) service offering crowd-sourced, expert-reviewed answers to your technical questions.

AWS Blog & Podcast

Read blog posts about Systems Manager in the AWS Management Tools Category, and other posts tagged with #Systems Manager.

Service quotas

Review Systems Manager service quotas in the *Amazon Web Services General Reference*. Unless otherwise noted, each quota applies to a single Region in an AWS account.

Service Authorization Reference for Systems Manager

In the *AWS Service Authorization Reference*, view information about the actions, resources, and condition context keys you can use in AWS Identity and Access Management (IAM) policies for Systems Manager.

AWS Systems Manager Service Level Agreement

The AWS 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.

General AWS resources

The following general resources can help you as you work with AWS.

- 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 Center – Explore tutorials, download tools, and learn about AWS developer events.
- AWS Developer Tools – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- Getting Started Resource Center – Learn how to set up your AWS account, join the AWS community, and launch your first application.
- Hands-On Tutorials – Follow step-by-step tutorials to launch your first application on AWS.
- 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.
- Support – The primary webpage for information about 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.

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.

Change	Description	Date
New topic: Understanding SSM Agent hibernation	This new topic explains that hibernation is an operational mode that occurs when the agent can't maintain proper communication with the Systems Manager service. During hibernation, the agent reduces its communication frequency and enters a standby state. For details about agent hibernation, see Understanding SSM Agent hibernation.	September 2, 2025
Updated managed policy: AWSSystemsManagerJustInTimeAccessServicePolicy	Systems Manager has updated the managed policy `AWSSystemsManagerJustInTimeAccessServicePolicy` to add automation execution tagging permissions, making it possible for customers to scope down operator permissions to specific tags. For information, see Systems Manager updates to AWS managed policies.	August 25, 2025

Updated descriptions for node detail tabs (unified console)	We've updated the descriptions of content on the tabs on the node details page in the unified console (Tabs: **Tags**, **Inventory**, **Association**, **Patches**, **Configuration compliance**). For information, see What is the unified console?.	August 20, 2025
New security best practice topic for installing SSM Agent	The new topic SSM Agent installation best practices explains the importance of using the correct method for installing the agent depending on the machine type, EC2 instance or hybrid-activated machine.	August 18, 2025

New console-based Automation features	AWS Systems Manager Automation has introduced enhanced execution capabilities with automatic retry for throttled operations and the ability to rerun automation executions. The automatic throttling retry feature helps ensure reliable automation execution at scale by automatically retrying API calls that encounter throttling limits, with configurable time limits up to 3600 seconds. (All Automation actions except aws:executeScript automatic throttling retry. Additionally, you can now easily rerun completed automation executions with identical or modified parameters, streamlining repeated tasks and reducing operational overhead without manually recreating configurations. For more information, see the following topics: • Configuring automatic retry for throttled operations • Rerunning automation executions • Running automations in multiple AWS Regions and accounts	August 14, 2025

Updates for the AWSQuickSetupDeploymentRolePolicy managed policy	Systems Manager has updated the `AWSQuickSetupDeploymentRolePolicy` managed policy to grant association-related permissions to additional `AWSQuickSetupType-*` Automation runbooks and to allow Quick Setup to attach and detach additional IAM policies. In addition, we have updated the documentation for `AWSQuickSetupDeploymentRolePolicy` to provide more detailed descriptions of the permissions granted for Quick Setup configuration management operations. For information, see Systems Manager updates to AWS managed policies.	August 12, 2025

New topic: Data perimeters in AWS Systems Manager	You can configure data perimeter policies for AWS service-owned resources that Systems Manager accesses. Data perimeters in AWS Systems Manager describes the purpose of the Systems Manager managed `ssm-document-categories-` *region* S3 bucket. This bucket contains metadata about document categories that help organize and classify SSM Documents in the console.	August 12, 2025
New managed policy for Quick Setup scheduler : AWSQuickSetupStartStopInstancesExecutionPolicy	Systems Manager has released a new managed policy, `AWSQuickSetupStartStopInstancesExecutionPolicy` , that provides permissions for Quick Setup to start and stop Amazon EC2 instances on a schedule. For information, see Systems Manager updates to AWS managed policies.	August 12, 2025

New AWS managed policy for Quick Setup: AWSQuickSetupStartSSMAssociationsExecutionPolicy	Systems Manager has released a new managed policy `AWSQuickSetupStartSSMAssociationsExecutionPolicy` that allows Quick Setup to run the `AWSQuickSetupType-StartSSMAssociations` Automation runbook. For information, see Systems Manager updates to AWS managed policies.	August 12, 2025
New troubleshooting guidance for zypper package lock dependency failures on SLES managed nodes	Added comprehensive troubleshooting guidance for resolving zypper package lock dependency failures that occur during Patch Manager `Install` operations on SUSE Linux Enterprise Server managed nodes. The new section Issue: Zypper package lock dependency failures on SLES managed nodes provides detailed information about identifying, diagnosing, and resolving package lock conflicts that prevent successful patch installation.	July 28, 2025

Enhanced troubleshooting guidance for skipped inventory associations	Updated the Skipped troubleshooting section to provide comprehensive guidance on inventory association priority handling. The enhanced documentation now includes detailed information about Quick Setup inventory associations, explicit targeting scenarios, and step-by-step resolution procedures for managing conflicting inventory associations. Key improvements include: • Explanation of inventory association priority order (Quick Setup, explicit, and global associations) • Common scenarios with specific examples and expected outcomes • Detailed resolution steps for removing Quick Setup configurations when using custom inventory associations • Important notes about data collection behavior and association precedence	July 24, 2025

Updated managed policy: AWS-SSM-RemediationAutomation-ExecutionRolePolicy	Systems Manager has updated the managed policy `AWS-SSM-RemediationAutomation-ExecutionRolePolicy` to improve the security posture of the ssm:StartAutomationExecution API by requiring permissions for both "document" and "automation-execution" resource types. This update is being made to improve the security posture of the ssm:StartAutomationExecution API by requiring permissions for both "document" and "automation-execution" resource types, providing more comprehensive and detailed permissions for remediation automation execution. For information, see Systems Manager updates to AWS managed policies.	July 16, 2025

Updated managed policy: AWS-SSM-RemediationAutomation-AdministrationRolePolicy	Systems Manager has updated the managed policy `AWS-SSM-RemediationAutomation-AdministrationRolePolicy` to support API authorization improvements for remediation automation operations. The updated policy enhances permissions for executing activities defined within Automation documents, with improved security controls and resource access patterns for remediation workflows. For information, see Systems Manager updates to AWS managed policies.	July 16, 2025

Updated managed policy: AWS-SSM-DiagnosisAutomation-ExecutionRolePolicy	Systems Manager has updated the managed policy `AWS-SSM-DiagnosisAutomation-ExecutionRolePolicy` to provide more detailed and accurate permissions for diagnosis automation execution. The updated policy includes enhanced descriptions for Amazon EC2 and Amazon VPC resource access, more specific SSM automation permissions, and improved AWS KMS and IAM permission descriptions with proper resource restrictions. For information, see Systems Manager updates to AWS managed policies.	July 16, 2025

Updated managed policy: AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy	Systems Manager has updated the managed policy `AWS-SSM-DiagnosisAutomation-AdministrationRolePolicy` to include more specific permissions and security conditions for diagnosis automation operations. The updated policy provides enhanced security controls for AWS KMS key usage, Amazon S3 bucket access, and role assumptions, with stricter resource-based conditions and account-level restrictions. For information, see Systems Manager updates to AWS managed policies.	July 16, 2025
Added documentation for AmazonSSMAutomationRole managed policy	Systems Manager has updated the documentation for the existing managed policy `AmazonSSMAutomationRole` . For information, see Systems Manager updates to AWS managed policies.	July 15, 2025

Enhanced security for SSM documents with environment variable interpolation	AWS Systems Manager Command documents now support environment variable interpolation when processin g parameters. This feature, available in schema version 2.2 and with SSM Agent version 3.3.2746.0 or higher, helps prevent command injection attacks by treating parameter values as literal strings rather than potential ly executable commands. When set to `ENV_VAR`, the agent creates an environment variable named in the format `SSM_`*`parameter-name`* that contains the parameter's value. For more information, see the following topics: • Document components • Secure parameter handling examples • Data elements and parameters • Command document plugin reference	July 14, 2025

Explorer resource data sync available in four opt-in AWS Regions	By default, Explorer cross-account/cross-Region resource data syncs don't support data aggregation in opt-in Regions. Support was added for the following opt-in Regions on June 30, 2025. • Europe (Milan) • Africa (Cape Town) • Middle East (Bahrain) • Asia Pacific (Hong Kong) Note that you can't use a delegated administrator account to create a resource data sync in opt-in Regions. You must use an AWS Organizations management account. For more information, see Understanding multiple account and Region resource data syncs.	June 30, 2025
New versions of the AWS Parameters and Secrets Lambda Extension	New versions of the AWS Parameters and Secrets Lambda Extension are now available for all supported Regions and architectures. In addition, support has been added for the Asia Pacific (Taipei) Region (ap-east-2).	June 26, 2025

New topic: Change Calendar integration with Amazon EventBridge	The topic Change Calendar integration with Amazon EventBridge provides information about this integration in the following categories: • Event timing and reliability • Event modifications and timing considerations • Adjacent and overlapping events • Best practices	June 25, 2025
Runtime support update for Automation action 'aws:executeScript'	The `Runtime` parameter in the Automation action aws:executeScript now supports PowerShell 7.4 (`dotnet8`) scripts.	June 24, 2025
SSM Agent and Patch Manager now support RHEL 10 and Ubuntu Server 25.04	You can now manage Red Hat Enterprise Linux 10 and Ubuntu Server 25.04 machines with Systems Manager, including patching managed nodes using Patch Manager. For complete lists of supported OSs and versions, see the following topics: • Supported operating systems for Systems Manager • Supported operating systems for Patch Manager	June 23, 2025

Updates to just-in-time node access topics	We've made the following updates to content in the Just-in-time node access using Systems Manager chapter: • Clarified authentication support for just-in-time node access. For information, see Setting up just-in-time access with Systems Manager. • Clarified the requirement to tag the AWS KMS keys used for Session Manager encryption and RDP recording in just-in-time node access with the tag key `SystemsManagerJustInTimeNodeAccessManaged` and tag value `true`. For information, see Update just-in-time node access session preferences and Configuring S3 bucket encryption for RDP recordings. • Updated Configure notifications for just-in-time access requests with additional configuration details.	June 17, 2025

Enhanced RDP recording for just-in-time node access	AWS Systems Manager just-in-time node access now provides additional informati on about RDP recording configurations. When using AWS KMS encryption on S3 buckets where recordings are stored, you must provide access to the KMS key used for bucket encryption to the `ssm-guiconnect` service principal. This release also introduces a new recording status, `ProcessingError` , which indicates intermedi ate failures during video processing that could be due to service dependency failures or missing permissio ns.For more information, see Recording RDP connections.	June 13, 2025
Updated information about patch selection and installat ion in Patch Manager	Added information to explain how Patch Manager handles obsolete packages during patch selection and installat ion. For more information, see How security patches are selected and How patches are installed.	June 12, 2025

Updated managed policy for Quick Setup: AWSQuickSetupDeploymentRolePolicy	Systems Manager has updated the managed policy `AWSQuickSetupDeploymentRolePolicy` so that Quick Setup can create associations using this managed policy instead of inline policies. For information, see Systems Manager updates to AWS managed policies.	June 10, 2025
Expanded introduction to Patch Manager	To better introduce Patch Manager to potential customers and users, we have added the following three sections to the Patch Manager chapter: • How can Patch Manager benefit my organization? • Who should use Patch Manager? • What are the main features of Patch Manager?	May 21, 2025

Updated managed policy for Quick Setup: AWSQuickSetupSSMDeploymentRolePolicy	Systems Manager has updated the managed policy `AWSQuickSetupSSMDeploymentRolePolicy` to provide permissions to tag IAM roles and Lambda resources created for the unified console. For information, see Systems Manager updates to AWS managed policies.	May 7, 2025
Feature configuration options for the unified console	Systems Manager now provides feature configuration options when setting up the unified console. These feature configurations allow you to choose whether to set up Default Host Management Configuration (DHMC), inventory metadata collection, and automatic SSM Agent updates. For more information, see Setting up Systems Manager unified console for an organization and Setting up Systems Manager unified console for a single account and Region.	May 7, 2025

Updated managed policy for Quick Setup: AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy	Systems Manager has updated the managed policy `AWSSystemsManagerJustInTimeNodeAccessRolePropagationPolicy` to provide permissions to tag a resource shared by AWS Resource Access Manager for just-in-time node access. For information, see Systems Manager updates to AWS managed policies.	April 29, 2025
Updated managed policy for Quick Setup: AWSQuickSetupManageJITNAResourcesExecutionPolicy	Systems Manager has updated the managed policy `AWSQuickSetupManageJITNAResourcesExecutionPolicy` to provide permissions to tag IAM roles created for just-in-time node access. For information, see Systems Manager updates to AWS managed policies.	April 29, 2025

Systems Manager introduces just-in-time node access	AWS Systems Manager now supports a just-in-time access method of connecting to managed nodes. This new method of connecting to managed nodes helps improve the security of your nodes by providing temporary, time-bound access to nodes only when needed, eliminating the need for long-standing permissions. Additionally, Systems Manager provides session recording for RDP sessions to Windows Server nodes to help you meet compliance requirements, perform root cause analysis, and more. For more information, see Just-in-time node access using Systems Manager.	April 29, 2025
New managed policies for just-in-time node access	Systems Manager has released several new managed policies to support just-in-time node access. For information, see Systems Manager updates to AWS managed policies.	April 29, 2025
New versions of the AWS Parameters and Secrets Lambda Extension	New versions of the AWS Parameters and Secrets Lambda Extension are now available for all supported Regions and architectures.	April 23, 2025

New topic: "What is compliance in Patch Manager?"	Patch compliance for the managed nodes in your fleet is not defined by AWS or other third parties. The new topic What is compliance in Patch Manager? clarifies that patch compliance for a node managed by Systems Manager means that all patches have been installed on the node that meet the approval criteria specified in the associated patch baseline. Compliance is defined by you, in the rules you configure in a patch baseline, or in a predefined patch baseline provided by AWS that you choose for your patching operations.	April 11, 2025
New versions of the AWS Parameters and Secrets Lambda Extension	New versions of the AWS Parameters and Secrets Lambda Extension are now available. In addition, extension support has been added for the following Regions: • Asia Pacific (Thailand) (ap-southeast-7) • Mexico (Central) (mx-central-1)	April 3, 2025

New topic: Configuring permissions for Compliance	As a security best practice, we recommend that you update the AWS Identity and Access Management (IAM) role used by your managed nodes to restrict the node's ability to use the PutComplianceItems API action. This API action registers a compliance type and other compliance details on a designated resource, such as an Amazon EC2 instance or a managed node. For more information, see Configuring permissions for Compliance.	March 11, 2025
Updated topic: Reviewing node insights	In the unified console for Systems Manager, the **Review node insights** page reports the number of managed and unmanaged EC2 instances in your organization or account. We have clarified that Systems Manager includes an EC2 instance in the "Unmanaged" count when it is in a `Stopped` state. For more information, see What is an unmanaged instance?.	February 25, 2025

Updated topic: Verify the signature of SSM Agent	The AWS Systems Manager Agent (SSM Agent) RPM and Debian 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 the files are damaged or have been altered, the verification fails. You can verify the signature of the installer package using either RPM or GPG. The topic includes a new public key for SSM Agent. For more information, see Verify the signature of SSM Agent.	February 14, 2025
New topic: Verify the signature of the Session Manager plugin	The Session Manager plugin RPM and Debian installer packages for Linux instances are cryptographically signed. You can use a public key to verify that the plugin binary and package is original and unmodified. If the file is altered or damaged, the verification fails. You can verify the signature of the installer package using the GNU Privacy Guard (GPG) tool. For more information, see Verify the signature of the Session Manager plugin.	February 13, 2025

Session Manager plugin updated	The Session Manager plugin was recently updated with the following enhancements: Upgraded the Go version to 1.23 in the Dockerfile. Updated the version configuration step in the README. For more information about the Session Manager plugin, see Install the Session Manager plugin for the AWS CLI.	February 6, 2025
TPS quotas published for Automation API actions	We have published the transaction per second (TPS) quotas, formerly known as limited, for several Systems Manager Automation-related API actions. For information, see Systems Manager service quotas in the *Amazon Web Services General Reference*.	January 21, 2025
Terminology change: Systems Manager capabilities renamed as "tools"	Until now, the functional units that make up Systems Manager have been called *capabilities*, such as the Automation capability, the Parameter Store capability, and so on. These functional units are now referred to as *tools*, such as the Automation tool and the Parameter Store tool. This user guide has been updated to reflect the change.	January 10, 2025

Patch Manager support for additional macOS versions (Amazon EC2 AMIs)	Patch Manager now supports macOS versions 12.7, 13.6 and 13.7, and 15.*x*. For complete lists of OSs and versions supported by Patch Manager, see Supported operating systems for Patch Manager.	January 8, 2025
SSM Agent and Patch Manager support for additional versions: CentOS Stream, Ubuntu Server, and Windows Server	Patch Manager now supports CentOS Stream 9, Ubuntu Server 24.10, and Windows Server 2025. SSM Agent now supports Ubuntu Server 24.10 and Windows Server 2025. (Agent support for CentOS Stream 9 was previousl y released.) For complete lists of supported OSs and versions, see the following topics: • Supported operating systems for Systems Manager • Supported operating systems for Patch Manager	November 22, 2024

New topic: AWS KMS encryption for Parameter Store SecureString parameters	Learn how AWS Systems Manager Parameter Store uses AWS Key Management Service to encrypt the values of `SecureString` parameters in Parameter Store in the following topic: • AWS KMS encryption for SecureString parameters in Parameter Store	November 22, 2024
New and updated managed policies for Systems Manager	To support new features for Systems Manager, we are releasing multiple new managed policies to support new Systems Manager configurations and operations, and updating other managed policies. For more information, see Systems Manager updates to AWS managed policies.	November 21, 2024

A new, simplified node management experience for Systems Manager	AWS Systems Manager has released a new unified console experience for managing nodes at scale across accounts and Regions. You can now see all managed and unmanaged nodes across your organizations' AWS accounts and Regions from a single place. You can also identify, diagnose, and remediate unmanaged nodes. Systems Manager is also now integrated with Amazon Q Developer (Amazon Q), which extends your ability to see and control your nodes from anywhere in the AWS Management Console by entering natural language prompts. With this release, you can also now use AWS Organizations to allow a delegated administrator account to manage nodes across the organization from a central viewpoint. For more information, see the following topics: • What is AWS Systems Manager? • Setting up AWS Systems Manager • Complete node tasks with AWS Systems Manager	November 21, 2024

Session Manager plugin bug fix	The Session Manager plugin was recently updated with the following bug fix: Rolled back change that added credentials to OpenDataC hannel requests.	November 20, 2024
Session Manager plugin enhancements	**This version was deprecated on 11/20/2024.** The Session Manager plugin was recently updated with the following enhancements. • Added credentials to OpenDataChannel requests. • Upgraded the `testify` and `objx` dependent packages.	November 6, 2024
Additional OS version support for macOS	Systems Manager now supports version 15.*x* (Sequoia) of the macOS operating system (EC2 instances only). For a list of all supported OSs and versions, see Supported operating systems for Systems Manager.	November 6, 2024

Patch Manager: Updates to supportd package name formats for Approved and Rejected lists	For several operating systems, we have updated and expanded the lists of formats for package names that you can specify in **Approved patches** and **Rejected patches** lists in your patch baselines. For information, see Package name formats for Amazon Linux 1, Amazon Linux 2, Amazon Linux 2022, Amazon Linux 2023, CentOS, Oracle Linux, and Red Hat Enterprise Linux (RHEL).	November 1, 2024
Additional operating version support for Patch Manager	Patch Manager now supports additional versions of Oracle Linux (8.10 and 9.4) and Ubuntu Server (23.10 and 24.04). For lists of all supported operating systems and versions, see the following topics: • Supported operating systems for Systems Manager • Supported operating systems for Patch Manager	November 1, 2024

SSM Agent support for additional versions: CentOS Stream, Oracle Linux, and Ubuntu Server	SSM Agent now supports CentOS Stream 9, Oracle Linux 8.10 and 9.4, and Ubuntu Server 24.04 LTS, in addition to earlier supported versions. For complete lists of supported OSs and versions for Systems Manager, see Supported operating systems for Systems Manager.	October 25, 2024
Session Manager plugin enhancement	Added support for passing the plugin version with OpenDataChannel requests.	October 10, 2024
New: View details about RDP connections made using Fleet Manager	You can now view informati on about Remote Desktop Protocol connections that have been made by users in your AWS account. For information, see Viewing information about current and completed connections.	October 10, 2024
Patch Manager now supports SLES version 15.6	Patching support for SUSE Linux Enterprise Server (SLES) 15.6 has been released. You can now patch SLES 15.6 machines using Patch Manager. For a full list of operating systems and versions supportd by Patch Manager, see Supported operating systems for Patch Manager.	September 29, 2024

New versions of the AWS Parameters and Secrets Lambda Extension	New versions of the AWS Parameters and Secrets Lambda Extension are now available. Support for all architectures has been introduced for the Asia Pacific (Malaysia) Region (ap-southeast-5). In addition, ARM64 and Mac with Apple silicon architecture extension support has been added for the following Regions: • Asia Pacific (Hyderabad) (ap-south-2) • Asia Pacific (Melbourne) (ap-southeast-4) • Canada West (Calgary) (ca-west-1) • Europe (Zurich) (eu-central-2) • Europe (Spain) (eu-south-2) • Middle East (UAE) (me-central-) • China (Beijing) (cn-north-1) • China (Ningxia) (cn-northwest-1) • Israel (Tel Aviv) (il-central-1) • AWS GovCloud (US-East) (us-gov-east-1) • AWS GovCloud (US-West) (us-gov-west-1)	September 19, 2024

New topic: Configuring permissions for maintenance windows using the AWS CLI	The topic Configuring permissions for maintenance windows using the AWS CLI provides instructions for creating a custom service role (and its policies) for running maintenance window tasks on a user's behalf.	August 19, 2024
SSM Agent and Patch Manager support for additional versions: AlmaLinux, Oracle Linux, and Rocky Linux	SSM Agent and Patch Manager now support versions 8.10, 9.3, and 9.4 of AlmaLinux and Rocky Linux, and version 9.3 of Oracle Linux, in addition to earlier supported versions. For complete lists of supported OSs and versions, see the following topics: • Supported operating systems for Systems Manager • Supported operating systems for Patch Manager	August 14, 2024

New IAM policy condition for Parameter Store support: ssm:Policies	Using `ssm:Policies` , a newly supported condition for IAM policies, you can prevent Entities from creating or updating parameter that include a parameter policy. For more information, see the following topics: • ssm:Policies: Prevent creation or updates to parameters that use parameter policies • Condition keys for AWS Systems Manager in the *Service Authorization Reference*.	August 14, 2024
Updated managed policy for Quick Setup: SSMQuickSetupRolePolicy	Systems Manager has updated the managed policy `SSMQuickSetupRolePolicy` to provide access to additional AWS CloudFormation stack sets. For information, see Systems Manager updates to AWS managed policies.	August 13, 2024

Support for provisioning and managing Systems Manager resources using Terraform	We have added HashiCorp Terraform to the list of supported third-party integrations with Systems Manager. Terraform is an open-source *infrastructure as code* (IaC) software tool that provides a command line interface (CLI) workflow to manage various cloud services. You can use Terraform to provision and manage a number of commonly used Systems Manager resources and data sources. For information about this and other third-par ty integrations with Systems Manager, see Integration with other products and services.	August 1, 2024

New Quick Setup console experience and API	Systems Manager Quick Setup has released a new console experience and API. Now you can interact with this API using the console, AWS CLI, AWS CloudFormation, and SDKs. You can opt in to the new console using the Quick Setup console. For more information about onboarding to the new Quick Setup experience, see Getting started with Quick Setup. For more details about the API operations available through the Quick Setup API, see the Quick Setup API Reference.	August 1, 2024

New topic: Rejected patch list options in custom patch baselines	For patching operations that use a custom patch baseline in Patch Manager, we have clarified the behavior when a patch added to the **Rejected patch** list is assigned the action `Allow as dependency` . Because Windows Server doesn't support the concept of patch dependencies, patches not already installed on a managed node are skipped. Patches that are already installed on the node are assigned the status `INSTALLED_REJECTED` . For more information, see Rejected patch list options in custom patch baselines and Patch compliance values for other operating systems.	July 23, 2024
New topic: Configuring SSM Agent for use with the Federal Information Processing Standard (FIPS)	We have provided instructions for configuring SSM Agent for use with the Federal Information Processing Standard (FIPS). For information, see Configuring SSM Agent for use with the Federal Information Processing Standard (FIPS).	July 22, 2024

Update: Clarified support for @ symbol in Fleet Manager user names	If an IAM Identity Center user name contains one or more @ symbols, Fleet Manager RDP disregards the first @ symbol and all characters that follow it, whether or not the @ introduces the domain portion of an email address. For more informati on about supported character s for user names in Fleet Manager RDP connections, see Authenticating Remote Desktop connections.	July 21, 2024
Updated managed policy: AmazonSSMManagedEC 2InstanceDefaultPolicy	Systems Manager has updated the managed policy `AmazonSSMManagedEC 2InstanceDefaultPo licy` by providing inline statement IDs (Sids) to clarify the purpose of each policy statment. For information, see Systems Manager updates to AWS managed policies.	July 18, 2024

Name changes to AWS managed buckets for Patch Manager patching operations	AWS owns and maintains a number of Amazon S3 buckets that SSM Agent accesses in the course of performing various Patch Manager patching operations. 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 operation s, you must provide explicit permission in an Amazon EC2 instance profile for Systems Manager, or in a service role for non-EC2 machines in a hybrid and multiclou d environment. Otherwise, your resources can't access these public buckets. In most cases, we are changing the names of these buckets. For example, for patching operations, the bucket `aws-patchmanager-macos-us-east-2` is replaced by `aws-patchmanager-macos-us-east-2-552881074` , and the bucket `aws-ssm-us-east-2` is replaced by `aws-patch-manager-us-east-2-552881074` . For more	July 18, 2024

	information, see the following topics: • SSM Agent communications with AWS managed S3 buckets • Reference: Amazon S3 buckets for patching operations	
New service-linked role for Quick Setup	Systems Manager has released a new service-linked role, `AWSServiceRoleForSSMQuickSetup` . Systems Manager uses this role to check configuration health of resources set up using Quick Setup, to ensure consistent use of parameters and provisioned resources, and to remediate resources when drift is detected. The managed policy associated with this role is `SSMQuickSetupRolePolicy` . For more information, see AWSServiceRoleForSSMQuickSetup service-linked role permissions for Systems Manager.	July 3, 2024

New managed policies for Quick Setup configuration types	Systems Manager has released an additional 12 new managed policies to support various Quick Setup configuration types and processes. For information, see Systems Manager updates to AWS managed policies.	July 3, 2024
Support for RHEL 8.10 and 9.4	Systems Manager and Patch Manager now support Red Hat Enterprise Linux versions 8.10 and 9.4. For more information, see Supported operating systems and machine types and Supported operating systems for Patch Manager.	June 26, 2024
Patch Manager support for 8.8 and 8.9 versions: AlmaLinux, Oracle Linux, and Rocky Linux	Patch Manager now supports versions 8.8 and 8.9 of AlmaLinux, Oracle Linux, and Rocky Linux, in addition to earlier 8.x versions. For complete lists of supported OSs and versions for Patch Manager, see Supported operating systems for Patch Manager.	June 17, 2024

New public parameters for macOS Amazon EC2 AMIs	Public parameters have been released to support Amazon Machine Images for macOS Amazon Elastic Compute Cloud instances. For more information, see the following topics. • Finding public parameters • Calling AMI public parameters for macOS • Launch a Mac instance in the *Amazon EC2 User Guide* • Amazon EC2 Mac instances in the *Amazon EC2 User Guide* • Use AWS Systems Manager parameters instead of AMI IDs in launch templates in the *Amazon EC2 Auto Scaling User Guide*	June 17, 2024
Update: Regional availability of the `/aws/service/global-infrastructure` parameter path	We have clarified which commercial Regions the `/aws/service/global-infrastructure` public parameter path can be queried from, and how to run a query for the path if you are working in a different commerical AWS Region. For information, see Calling public parameters for AWS services, Regions, endpoints, Availability Zones, local zones, and Wavelength Zones.	June 12, 2024

New: Code examples chapter	A new chapter, Code examples for Systems Manager using AWS SDKs, provides examples in different SDK languages for how to work with the Systems Manager service.	May 8, 2024
Changes to `ec2messages:*` endpoint support	For AWS Regions launching in 2024 or later, the `ec2messages:*` endpoints are not supported by SSM Agent for sending status and execution information back to the Systems Manager service. Accounts in these Regions must use `ssmmessages:*`. In Regions launched before 2024, both `ssmmessages:*` and `ec2messages:*` are still supported, but we recommend using only the `ssmmessages:*` endpoint (Amazon Message Gateway Service) now. You can safely remove `ec2messages:*` permissions from your policies at this time. For more information, see Working with SSM Agent and Agent-related API operations (ssmmessages and ec2messages endpoints).	May 3, 2024

Additional runtimes available for running scripts in Automation runbooks	The `aws:executeScript` action now supports the Python 3.9, 3.10, and 3.11 runtimes. For more information about how to use this action, see `aws:executeScript`.	April 23, 2024
Support for 8.8 and 8.9 versions: AlmaLinux, Oracle Linux, and Rocky Linux	Systems Manager now supports versions 8.8 and 8.9 of AlmaLinux, Oracle Linux, and Rocky Linux, in addition to earlier 8.x versions. For complete lists of supported OSs and versions, see Supported operating systems for Systems Manager.	April 22, 2024

Patch Manager: Change to patching status 'INSTALLED_PENDING_REBOOT'	Previously, only patches installed by Patch Manager could be marked as `INSTALLED_PENDING_REBOOT` . Patches installed outside of Patch Manager were never given this status. Now, `INSTALLED_PENDING_REBOOT` can apply to any patch that has been applied to a managed node since it was last rebooted. This includes patches installed by Patch Manager with the `NoReboot` option selected, and to patches installed outside of Patch Manager after the node's most recent reboot. For descriptions of all Patch Manager patching status values, see Understanding patch compliance state values.	April 16, 2024
Support for RHEL 8.9 and 9.3	Systems Manager, including Patch Manager, now supports Red Hat Enterprise Linux (RHEL) versions 8.9 and 9.3, in addition to earlier 8.x and 9.x versions.	March 26, 2024

Topic update: AWS managed policies for AWS Systems Manager	The topic AWS managed policies for AWS Systems Manager has provided information about the four managed policies for Systems Manager that have been introduced or updated since March 12, 2021. We have added a section to this topic with information about the 12 other managed policies for use with Systems Manager that were created or last updated before that date. For details, see Additional managed policies for Systems Manager.	March 1, 2024

Parameter Store now supports cross-account sharing	You can now share advanced parameters securely and efficiently across AWS accounts or within your AWS Organization by setting up resource sharing. Resource sharing allows you to centralize application configuration management and reduce the operational overhead of sharing the parameters with every single account you own. Parameters can be shared across accounts using the Parameter Store console, the AWS RAM console, or the AWS CLI. For more information, see Working with shared parameters.	February 21, 2024
Automation action enhancement	You can now use the `onFailure` and `isCritical` properties with the `aws:approve` action. For more information about the `aws:approve` action, see aws:approve – Pause an automation for manual approval.	February 12, 2024

Additional operating version support for Patch Manager	We have added to the list of supported operating system versions for Patch Manager. Support has been added for the following: • Debian Server 11.x and 12.x • macOS 14.x (Sonoma) • SUSE Linux Enterprise Server (SLES) 15.5 • Ubuntu Server 23.04	January 4, 2024
Configure automated SSM Agent updates using the Application Manager console	You can now use the Applicati on Manager console to automate SSM Agent updates for your application instances . For more information, see Working with your application instances.	December 21, 2023
Updated process for registeri ng non Amazon EC2 machines in hybrid and multicloud environments	Systems Manager now provides the `ssm-setup-cli` to help you register non Amazon Elastic Compute Cloud (Amazon EC2) machines in hybrid and multicloud environments. For more information, see How to install the SSM Agent on hybrid Linux nodes and How to install the SSM Agent on hybrid Windows nodes.	December 20, 2023

Manage Amazon EBS volumes using Fleet Manager	You can now use Fleet Manager, a tool in AWS Systems Manager, to manage Amazon Elastic Block Store volumes on your managed instances. For example, you can initialize an EBS volume, format a partition, and mount the volume to make it available for use. For more information, see EBS volume management.	December 14, 2023
Session Manager plugin enhancement	Added support for passing a StartSession API response as an environment variable to session-manager-plugin.	December 4, 2023
New visual design experience for Automation runbooks	You can now create and edit runbooks using a new visual design experienc e developed by Systems Manager Automation. The visual design experience provides a low-code, drag-and-drop interface so you can create and edit runbooks more easily. For more information, see Visual design experience for Automation runbooks.	November 26, 2023

New Systems Manager Automation actions, data element, and functional enhancements for runbooks	You can now loop over multiple actions in a runbook using the `aws:loop` action. This new action supports `do while` and `for each` style loops. Additionally, using the new *variables* data element, you can define, reference and update values dynamically within the context of a runbook. To update the value of a variable in your runbook, use the new `aws:updateVariable` action. Automation has also added support for dynamic data type conversions for outputs. This means that if the value of an output doesn't match the data type you've specified, Automation tries to convert the data type. For example, if the value returned is an `Integer`, but the `Type` specified is `String`, the final output value is a `String` value. Lastly, Automation now supports JSONPath filter expressions for selectors. For more information, see the following topics: • aws:loop – Iterate over steps in an automation	November 17, 2023

	• aws:updateVariable – Updates a value for a runbook variable • Data elements and parameters – Top-level data elements • Using action outputs as inputs. • Using JSONPath in runbooks.	
Updated Region support for Remote Desktop Protocol (RDP) connections	Fleet Manager Remote Desktop, which is powered by Amazon DCV, provides you with secure connectivity to your Windows Server instances directly from the Systems Manager console. The following three additional Regions have been enabled for Fleet Manager Remote Desktop connections: • Africa (Cape Town) (af-south-1) • Asia Pacific (Jakarta) (ap-southeast-3) • Israel (Tel Aviv) (il-central-1)	November 15, 2023

Patch Manager: Expanded OS version support for RHEL and macOS	Patch Manager now supports the following additional operating system versions: • Red Hat Enterprise Linux: version 8.8 • macOS: 11.5–11:7 (Big Sur) • macOS: 12.0–12.6 (Monterey) • macOS: 13.0–13.5 (Ventura)	October 23, 2023
New OpsCenter API - DeleteOpsItem	OpsCenter now offers the DeleteOpsItem API for deleting individual OpsItems. For more information, see DeleteOpsItem in the *AWS Systems Manager API Reference*.	October 20, 2023
New Quick Setup configuration type: SSM Agent updates for entire organization	The new configuration type **Default Host Management Configuration** makes it possible for an organization administrator, as defined in AWS Organizations, to prompt automatic check and updates of SSM Agent on *all* EC2 instances in the organization's accounts and Regions. For more information, see Default Host Management for an organization.	October 16, 2023

New title and description format for OpsItems created by CloudWatch Application Insights	The title and description for OpsItems created by CloudWatch Application Insights is changing to an improved format on October 16, 2023. To view the new format, see Amazon CloudWatch Application Insights.	September 29, 2023
Support for multiple display resolutions in Fleet Manager RDP connections	When you connect to Windows Server managed nodes using the Remote Desktop protocol (RDP) option in Fleet Manager, you can now choose the display resolution. Previously, all connections used a fixed 720P (1366 x 768) resolution. You can now choose from the following for each connectio n: • **Adapt Automatically** (determines optimum resolution based on your detected screen size) • **1920 x 1080** • **1400 x 900** • **1366 x 768** • **800 x 600** For information, see Connect to a managed node using Remote Desktop.	September 22, 2023

New topic: Random patch baseline IDs in patch policy operations	We have added content to describe how Quick Setup patch policies use the `BaselineOverride` parameter in the `AWS-RunPatchBaseline` SSM Command document to generate random IDs for patch baselines each time a patch policy operation is run. For information, see Random patch baseline IDs in patch policy operations.	September 22, 2023
A new operational insight for managing OpsItems	OpsCenter now includes an operational insight called **Resources generating the most OpsItems**. An insight of this type is generated when an AWS resource has more than 10 open OpsItems. Use this insight to locate problematic resources. Use the `AWS-BulkResolveOpsItems` runbook from within an insight to quickly resolve OpsItems associated with a resource. For more informati on, see Analyzing operational insights to reduce OpsItems.	September 22, 2023
GPG public key updated	A new public key has been created to verify the signature of SSM Agent. For more information, see Verifying the signature of SSM Agent.	September 5, 2023

Support added for additional versions of AlmaLinux, Oracle Linux, RHEL, and Rocky Linux	The lists of supported operating systems for AWS Systems Manager and Patch Manager have been updated to reflect support the following additional OS versions: • AlmaLinux: 9.2 • Oracle Linux: 8.7 and 9.2 • Red Hat Enterprise Linux (RHEL): 8.7, 9.1, and 9.2 • Rocky Linux: 8.6 and 8.7, 9.0–9.2	August 30, 2023
OpsCenter added support for Markdown formatting in the OpsItem description field.	OpsCenter now supports Markdown formatting in the OpsItem description field. The following types of Markdown formatting are supported: • Paragraphs • Line spacing • Horizontal Lines • Headings • Text Formatting • Links • Lists For more information, see Using Markdown in the Console in the *Getting Started with the AWS Management Console Getting Started Guide.*	August 18, 2023

New versions of the AWS Parameters and Secrets Lambda Extension	New versions of the AWS Parameters and Secrets Lambda Extension are now available. In addition, extension support has been added for the Asia Pacific (Melbourne) (ap-southeast-4) and Israel (Tel Aviv) (il-centr al-1) Regions (x86_64 and x86 architectures only.) For more information, see Using Parameter Store parameters in AWS Lambda functions.	August 16, 2023

Update: Added information about required permissions for Quick Setup patch policy buckets	When you create a patch policy, Quick Setup creates an Amazon S3 bucket that contains a file named `baseline_overrides.json` . This file stores information about the patch baselines that you specified for your patch policy. When configuring the patch policy, you have the option of selecting an **Add required IAM policies to existing instance profiles attached to your instances** check box. If you choose not to select this option, then you must manually provide certain resources with permissions to access this bucket or your policy operations might fail. For more information, see the following topics: • Permissions for the patch policy S3 bucket • Issue: "Invoke-PatchBaselineOperation : Access Denied" error or "Unable to download file from S3" error for `baseline_overrides.json`	July 6, 2023

Use Quick Setup to configure OpsCenter for multi-account OpsItem management	Quick Setup for OpsCenter helps you complete the following tasks for managing OpsItems across accounts: • Specifying the delegated administrator account • Creating required AWS Identity and Access Management (IAM) policies and roles • Specifying an AWS Organizations organizat ion, or a subset of member accounts, where a delegated administrator can manage OpsItems across accounts For more information, see (Optional) Configure OpsCenter to manage OpsItems across accounts by using Quick Setup.	June 19, 2023
Update Amazon EC2 launch agents using Quick Setup	You can now allow Systems Manager to check every 30 days for a new version of the launch agent installed on your instance. If a new version is available, Systems Manager updates the agent on your instance. For more informati on, see Quick Setup Host Management.	June 19, 2023

Patch Manager now supports Ubuntu Server 22.04 LTS	You can now use Patch Manager to patch Ubuntu Server 22.04 LTS nodes. Like other supported versions of Ubuntu Server, version 22.04 LTS, uses the AWS managed `AWS-UbuntuDefaultPatchBaseline` patch baseline.	May 15, 2023
Systems Manager now supports AlmaLinux, including Patch Manager	You can now use Systems Manager to manage AlmaLinux 8.3-8.7; 9.0-9.1 nodes. Many of the rules that apply to RHEL 8 for patching also apply to AlmaLinux . AlmaLinux uses the new `AWS-DefaultAlmaLinuxPatchBaseline` . For more information, see the following topics: • Manually install SSM Agent on AlmaLinux instances • How security patches are selected • How patches are installed • How patch baseline rules work on AlmaLinux, RHEL, and Rocky Linux.	May 8, 2023

Deploy the EC2Launch v2 agent using Quick Setup	You can now deploy the EC2Launch v2 agent using Quick Setup. For more information, see Deploy Distributor packages with Quick Setup.	April 13, 2023
Systems Manager now supports Amazon Linux 2023	Systems Manager now supports the new Amazon Linux 2023 (AL2023) EC2 instance type, including support for Patch Manager operations. Many of the rules for patching that apply to Amazon Linux 2 also apply to Amazon Linux 2023. (Patch Manager also continues to support the preview release Amazon Linux 2022.) For more information, see the following topics: • How security patches are selected • How patches are installed • How patch baseline rules work on Amazon Linux 1, Amazon Linux 2, Amazon Linux 2022, and Amazon Linux 2023	March 23, 2023

Revised setting up content for Amazon EC2 instances	We have revised the setting up content for Amazon EC2 instances. It is now recommended to use the newly released Default Host Management Configuration for instance permissions. For more information, see Configure instance permissions required for Systems Manager.	February 15, 2023
Automatic instance management with the Default Host Management Configuration	You can now automatically manage Amazon EC2 instances in an entire AWS Region using Systems Manager. For more information, see Default Host Management Configuration.	February 15, 2023
Add SSM documents to your favorites	To help you find frequently used SSM documents, you can now add documents to your favorites. You can favorite up to 20 documents per document type, per AWS account and AWS Region. You can choose, modify, and view your favorites from the Systems Manager **Documents** console. For more information, see Adding documents to your favorites.	February 7, 2023

Implement change controls for Automation using Change Calendar	By integrating Automation with Change Calendar, you can now implement change controls to all automatio ns in your AWS account. For more information, see Implement change controls for Automation.	January 24, 2023
New Change Manager approval workflow	The Change Manager approval workflow now supports *per-level* approvals instead of *per-line approvals* . Previously, every approver you added to an approval level had to approve a change request. Otherwise, the level was not approved. Now, you specify how many approvals are required for the level and can add that many or more approvers. For example, you can require three approvals for a level but specify up to five approvers. Approvals from any three of those approvers are sufficien t to approve the level. For more information, see About approvals in your change templates.	January 23, 2023

New: Configure patching for an entire organization using a patch policy in Quick Setup	With Quick Setup, a tool in Systems Manager, you can now create *patch policies* powered by Patch Manager. A patch policy defines the schedule and patch baseline to use when automatically patching your managed nodes. Using a single patch policy configuration, you can define patching for all accounts in all Regions in your organization, for only the accounts and Regions you choose, or for a single account-Region pair. For more information, see the following topics. • Using Quick Setup patch policies • Automate organization-wide patching using a Quick Setup patch policy	December 22, 2022

Application Manager integrates with Amazon EC2 to display information about your instances in the context of an application.	Application Manager displays instance state, status, and Amazon EC2 Auto Scaling health for a selected application in a graphical format. The **Instances** tab also includes a table with the following information for each instance in your application.
	• Instance state (Pending, Stopping, Running, Stopped)
	• Ping status for SSM Agent
	• Status and name of the last Systems Manager Automation runbook processed on the instance
	• A count of Amazon CloudWatch Logs alarms per state.
	  • `ALARM` – The metric or expression is outside of the defined threshold.
	  • `OK` – The metric or expression is within the defined threshold.
	  • `INSUFFICIENT_DATA` – The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.
		December 22, 2022

	• Auto Scaling group health for the parent and individua l autoscaling groups	
Schedule the starting and stopping of your Amazon EC2 instances using Quick Setup	You can now deploy the Resource Scheduler solution to automate the starting and stopping of your Amazon EC2 instances using Quick Setup. For more information, see Resource Scheduler.	December 19, 2022

OpsCenter now supports working with OpsItems across accounts	OpsCenter supports working with OpsItems from a management account (either an AWS Organizations management account or a Systems Manager delegated administrator account) and a member account during a session. Once configure d, users can perform the following types of actions: • Create, view, and update OpsItems in a member account • View detailed informati on about AWS resources specified in OpsItems in a member account • Start Systems Manager Automation runbooks to remediate issues with AWS resources in a member account For more information, see Setting up OpsCenter to work with OpsItems across accounts.	November 16, 2022

Track details of Change Manager change requests using AWS CloudTrail Lake	You can now use an event data store in AWS CloudTrail Lake to capture and review details about the change requests that are run in Change Manager for your organization or account. This information includes auditable details about the user identity that created the change request, the IP address from which the request was made, the AWS Regions in which the changes were made, the targeted resources, and more. For information, see Monitoring your change request events and Reviewing change request details, tasks, and timelines.	November 11, 2022

Additional Systems Manager Automation task controls using CloudWatch alarms	You can now implement additional control when running automations across multiple accounts and Regions by using CloudWatch alarms. By applying a metric or composite CloudWatc h alarm to an automation, you can control when an automation stops based on the metrics you define. For more information about applying a CloudWatch alarm to an automation running across multiple accounts and Regions see Run an automatio n in multiple Regions and accounts (console)	November 9, 2022
Updated: 'Using Parameter Store parameters in AWS Lambda functions'	We have provided additiona l information to help you use the AWS Parameters and Secrets Lambda Extension to retrieve parameter values and cache them for future use in Lambda functions. Using the Lambda extension can reduce your costs by reducing the number of API calls to Parameter Store. For information, see Using Parameter Store parameters in AWS Lambda functions.	October 25, 2022

Additional Systems Manager task controls using CloudWatch alarms	You can now implement additional control when running automations and commands by using CloudWatch alarms. A CloudWatch alarm can also be added to an automation or command when it is registered with a State Manager association or maintenance window task. By applying a composite CloudWatch alarm to an automation or command, you can control when an automation or command stops based on the metric you define. For more information about applying a CloudWatch alarm to an automation or command see the following procedures: • How security patches are selected • How patches are installed • How patch baseline rules work on Amazon Linux 1, Amazon Linux 2, and Amazon Linux 2022.	September 26, 2022

Additional Systems Manager task controls using CloudWatch alarms	You can now implement additional control when running automations and commands by using CloudWatch alarms. A CloudWatch alarm can also be added to an automation or command when it is registered with a State Manager association or maintenance window task. By applying a composite CloudWatch alarm to an automation or command, you can control when an automation or command stops based on the metric you define. For more information about applying a CloudWatch alarm to an automation or command see the following procedures: • Running a simple automation • Running commands from the console • Create an association • Assign tasks to a maintenance window	September 26, 2022
Clarifying advanced-instances tier requirements	Based on customer feedback, we have clarified the scenarios that require you to activate the advanced-instances tier in Configuring instance tiers.	September 21, 2022

Deploy the Amazon CloudWatch Agent using Quick Setup	You can now deploy the Amazon CloudWatch agent using Quick Setup. For more information, see Deploy Distributor packages with Quick Setup.	September 20, 2022
'PatchGroup' key now supported for patch groups when EC2 instance metadata is allowed	When you allow tags in EC2 instance metadata, the tag keys you create must not contain any spaces. Previously, this prevented customers from adding some of their EC2 instances to patch groups in Patch Manager because the tag key `Patch Group` had to be applied to the instances. Patch Manager now supports both `Patch Group` (with a space) and `PatchGroup` (without a space) as the tag key for identifying instances for a patch group. EC2 instances where tags are allowed in instance metadata can now be added to patch groups in Patch Manager. For information, see About patch groups.	August 31, 2022

New topic: "How package release dates and update dates are calculated"	In patch baselines managed by AWS, new patches are auto-approved 7 days after they are released or updated. In custom patch baselines you create, you can optionall y specify how many days to wait after they are released or updated to auto-approve their installation. For Amazon Linux 1 and Amazon Linux 2, various factors influence how the latest release dates and update dates are calculated. To help you avoid unexpecte d results when choosing auto-approval delays, these factors are explained in the topic How package release dates and update dates are calculated.	August 24, 2022
Updated content: Patch an AMI and update an Auto Scaling group	We have updated the Updating AMIs for Auto Scaling groups walkthrou gh to use launch templates instead of launch configura tions. Additionally, we've implemented the latest Automation actions and runtimes in the runbook content.	June 22, 2022

Change Manager: Prevent users from creating auto-approvable requests	You can configure change templates in Change Manager to support automatic approvals, meaning that users with the necessary IAM permissions can choose to start the change request without requiring additional approval. Now, you can also restrict individual users, groups, or IAM roles from submitting auto-approval requests, even if a change template supports them. This is achieved through the use of a new IAM condition key, `ssm:AutoApprove` . For more information, see Controlling access to auto-approval runbook workflows	June 15, 2022

Updated guidance for maintenance window task roles	Previously, the Systems Manager console provided you with the ability to choose the AWS managed IAM service-linked role `AWSServiceRoleForAmazonSSM` to use as the maintenance role for your tasks. Using this role and its associated policy, `AmazonSSMServiceRolePolicy`, for maintenance window tasks is no longer recommended. You should create a custom policy and role for maintenance window tasks instead. For more information, see Setting up Maintenance Windows.	June 9, 2022
Port forwarding to remote hosts support for Session Manager	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 Starting a session (port forwarding to remote host).	May 25, 2022

Updated content: Instructions for manually installing SSM Agent on Amazon EC2 Linux instances	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 Manually installing SSM Agent on EC2 instances for Linux.	May 9, 2022

New topic: Amazon Machine Images (AMIs) with SSM Agent preinstalled	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 Amazon Machine Images (AMIs) with SSM Agent preinstalled.	May 8, 2022
New State Manager section	Added a new section that describes the details of when State Manager runs associations. For more information, see About association scheduling.	April 27, 2022

Patch Manager now supports Rocky Linux	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 `AWS-DefaultRockyLinuxPatchBaseline` . For more information, see the following topics: • How security patches are selected • How patches are installed • How patch baseline rules work on RHEL, CentOS Stream, and Rocky Linux.	April 14, 2022

Patch Manager now supports CentOS Stream 8	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 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	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	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	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 procedure s, including procedures for hybrid environments or edge devices, if applicable. For more information, see Setting up Systems Manager. For Rocky Linux machines, you must also manually install SSM Agent. For more information, see Manually install SSM Agent on Rocky Linux instances. 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	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 Allowing Automation to adapt to your concurrency needs. For more information about how to view Automation usage metrics, see Monitoring Automation metrics using Amazon CloudWatch.	January 27, 2022
Allow Automation to adapt to your concurrency needs and view Automation usage metrics	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 Allowing Automation to adapt to your concurrency needs. For more information about how to view Automation usage metrics, see Monitoring Automation metrics using Amazon CloudWatch.	January 27, 2022

Systems Manager documents organized by categories	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	You can now send messages using webhooks during an automation by creating an integration. Integrations can be invoked during an automation using the new `aws:invokeWebhook` action in your runbook. For more information about creating integrations, see Creating webhook integrations for Automation. To learn more about the `aws:invokeWebhook` action, see aws:invokeWebhook – Invoke an Automation webhook integration.	January 13, 2022
Capabilities not available in new AWS Region	The following Systems Manager tools currently aren't available in the new Asia Pacific (Jakarta) Region. • Application Manager • Change Calendar • Change Manager • Explorer • Fleet Manager • Incident Manager • Quick Setup	December 13, 2021

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. For more information, see Viewing overview information about an application.	December 7, 2021
Manage processes using Fleet Manager	You can now use Fleet Manager to manage processes on your nodes. For more information, see Working with processes.	December 6, 2021

Terminology change: managed instances are now managed nodes	With support for AWS IoT Greengrass core devices, the phrase *managed instance* has been changed to *managed node* 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	Systems Manager supports the following edge device configurations. • **AWS IoT Greengrass**: 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 Greengras s core devices, you must create an AWS Identity and Access Managemen t (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 Setting up AWS Systems Manager for edge devices. • **Edge devices in a hybrid environment**: 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-n ode activation for a hybrid environment, and manually	November 29, 2021

	install SSM Agent on your devices. For more informati on, see Setting up AWS Systems Manager for hybrid environments	
Connect to managed instances using Remote Desktop	You can now use Fleet Manager to connect to managed Windows instances using the Remote Desktop Protocol (RDP). These Remote Desktop sessions powered by Amazon DCV provide secure connections to your instances directly from your browser. For more informati on, see Connect using Remote Desktop.	November 23, 2021
Specify a maximum session duration and provide reasons for sessions	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 informati on, see Specify maximum session duration.	November 16, 2021

Patch Manager now supports the Raspberry Pi OS operating system	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: • How security patches are selected • How patches are installed • How patch baseline rules work on Debian Server	November 16, 2021
Access the Red Hat Knowledgebase portal	Use Fleet Manager to access the RHEL Knowledgebase portal to find solutions, articles, documentation, and videos about using Red Hat products. For more informati on, see Accessing the Red Hat Knowledge base portal.	November 3, 2021
Bulk edit OpsItems	OpsCenter now supports bulk editing OpsItems. You can select multiple OpsItems and edit one of the following fields: **Status**, **Priority**, **Severity**, **Category**. For more information, see Editing OpsItems.	October 15, 2021

Create input parameters that populate AWS resources	You can now create input parameters in Automatio n runbooks that populate AWS resources in the AWS Management Console. For information, see Creating input parameters that populate AWS resources.	October 14, 2021
New task invocation cutoff option for maintenance windows	You can now choose to block any new task invocatio ns from starting after the cutoff time specified for a maintenance window is reached. For information, see Assign tasks to a maintenance window (console).	October 13, 2021
Patch Manager support for macOS 11.3.1 and 11.4 (Big Sur)	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 AWS Systems Manager Patch Manager.	October 1, 2021

Application insights in Application Manager	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 continually 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 notificat ions or take actions. You can enable and view Applicati on Insights on the **Overview** and **Monitoring** tabs in Application Manager. For more information about Application Insights, see What is Amazon CloudWatc h Application Insights in the *Amazon CloudWatch User Guide*.	September 21, 2021

Import events from other calendars into Change Calendar	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-par ty 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 informati on, see Importing and managing events from third-party calendars.	September 8, 2021

New tagging and runbook features in Application Manager	Tagging enhancements include the ability to add tags to or delete tags from a specific resource or all resources in an Applicati on 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 Working with tags in Applicati on Manager and Working with runbooks in Application Manager.	August 31, 2021
New example: Create a change request using the AWS CLI	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-HelloWorldChan geTemplate` change template and `AWS-Hello World runbook` : • Creating change requests (AWS CLI)	August 20, 2021

New section: Use parameters in Amazon EKS	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 Use Parameter Store parameters in Amazon Elastic Kubernetes Service.	August 19, 2021
Updated Patch Manager lifecycle hooks	Patch Manager now provides a lifecycle hook–the ability to run a Systems Manager Command document–for an additional point during a **Patch now** patching operation. If you schedule instance reboots after running **Patch now**, you can specify a lifecycle hook to run after the reboot is complete. For more information, see Using 'Patch now' lifecycle hooks and About the `AWS-RunPatchBaselineWithHooks` SSM document.	August 9, 2021

Auto-approvals now supported for Change Manager requests	You can now configure change templates in Change Manager to support automatic approvals, meaning that users with the necessary IAM permissions can choose to start the change request without requiring additiona l approval. Users who have access to auto-approval templates can still choose to specify approvers if they choose. To help you control your Change Manager processes, approvals are still required for all requests during change freeze periods. For more information, see the following topics: • Creating change templates • Creating change requests • Try out the AWS managed `Hello World` change template	July 30, 2021

OpsCenter operational insights	OpsCenter automatically analyzes OpsItems in your account and generates *insights*. 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 informati on, see Working with operational insights.	July 13, 2021
View stopped instances in Fleet Manager	You can now view which instances are `running` and which instances are `stopped` from the Fleet Manager console. For more information, see AWS Systems Manager Fleet Manager.	July 12, 2021
New topic: Authoring Automation runbooks	A new topic, Authoring Automation runbooks, provides guidance and narrative examples of how to author content for custom Automation runbooks.	July 8, 2021

AWS CloudFormation stack and template creation in Application Manager	Application Manager helps you provision and manage resources for your applicati ons by integrating with CloudFormation. You can create, edit, and delete AWS CloudFormation templates and stacks in Applicati on Manager. Application Manager also includes a template library where you can clone, create, and store templates. Application Manager and CloudFormation display the same informati on 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 Working with AWS CloudForm ation Stacks in Application Manager.	July 8, 2021
New topic: Automatically rotate private keys for SSM Agent on hybrid instances	A new topic, Setting up private key auto rotation, 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	A new version of the Session Manager plugin for the AWS CLI has been released. For more information, see Session Manager plugin latest version and release history.	June 10, 2021
New IAM service-linked role	When you enable OpsCenter operational insights, Systems Manager creates a new AWS Identity and Access Management (IAM) service-linked role called `AWSSSMOpsInsightsServiceRolePolicy`. For more information about this role, see Using roles to create operational insight OpsItems in Systems Manager OpsCenter: `AWSSSMOpsInsightsServiceRolePolicy`.	June 9, 2021
New Patch Manager troubleshooting content for Linux	A new topic, Errors when running `AWS-RunPatchBaseline` on Linux, 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)	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 Registering maintenance window tasks without targets, Assign tasks to a maintenance window (console), and Schedule automations with maintenance windows.	May 28, 2021
Automation Runbook Reference relocated	The Automation Runbook Reference has been moved to a new location. For more information, see Systems Manager Automation Runbook Reference.	May 10, 2021

AWS Systems Manager Incident Manager launch	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 AWS Systems Manager Incident Manager User Guide.	May 10, 2021
State Manager supports Change Calendar	You can now specify Change Calendar names or Amazon Resource Names (ARNs) when you create or update a State Manager association. State Manager applies associati ons only when the change calendar is open, not when it's closed. For more information, see Creating associations and Editing and creating a new version of an association.	May 6, 2021
Clone Systems Manager documents	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 Cloning an SSM document.	May 4, 2021

Integrate Security Hub with Explorer and OpsCenter	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 integrate d 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 Receiving findings from AWS Security Hub in Explorer and Receiving findings from AWS Security Hub in OpsCenter.	April 27, 2021
New topic: Document conventions	We've added a new topic to help users understand the common typographical conventions for the *AWS Systems Manager User Guide*. For more information, see Document conventions.	April 21, 2021

Updated topic: About patching applications released by Microsoft on Windows Server	The topic About patching applications released by Microsoft on Windows Server 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 **Give me updates for other Microsoft products when I update Windows** must be allowed on the instance.	April 12, 2021
Automation Runbook Reference reorganization	To help you find the runbooks you need and navigate the reference more efficient ly, we reorganized the content in the Automatio n Runbook Reference by the relevant AWS service. To view these changes, see Systems Manager Automation Runbook Reference.	April 12, 2021

Patch Manager: Generate .csv patch compliance reports	Patch Manager now supports the ability to generate patch compliance reports for your instances and save the report in an S3 bucket of your choice, in .csv format. Then, using a tool like Amazon QuickSight, 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 automatic ally. You can also specify an Amazon Simple Notificat ion Service topic to provide notifications when a report is generated. For more informati on, see Generating CSV patch compliance reports.	April 9, 2021
Delete Parameter Store parameter labels	You can now delete Parameter Store parameter labels by using either the Systems Manager console or the AWS CLI. For more information, see Working with parameter labels.	April 6, 2021

Schedule instance reboots when using Patch Now	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 Patching instances on demand.	April 1, 2021
New topic: Discover public parameters	Parameter Store public parameters can now be found using the AWS CLI or Systems Manager console. For more information, see Finding public parameters.	April 1, 2021

Patch now updates: Store logs in S3 & and run lifecycle hooks	When you run the Patch Manager **Patch now** 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 : *Before installation*, *After installation*, and *On exit*. For more information, see Patching instances on demand.	March 31, 2021
Systems Manager now reports changes to its AWS managed policies	Beginning March 24, 2021, changes to managed policies are reported in the topic Systems Manager updates to AWS managed policies. The first change listed is the addition of support for the Explorer tool 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	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 About multiple account and Region resource data syncs.	March 24, 2021
Systems Manager Automation provides a new system variable for your runbooks	With the new `global:AWS_PARTITION` system variable, you can specify the AWS partition a resource is located in when authoring your runbooks. For more information, see Automation system variables.	March 18, 2021

Allow multiple levels of approval for Change Manager change requests	When you create a Change Manager change template, you can now require that more than one level of approvers grant permissio n 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 informati on, see Creating change templates.	March 4, 2021
Patch Manager now supports Oracle Linux 8.x	You can now use Patch Manager to patch Oracle Linux 8.x instances, through version 8.3. For more information, see the following topics: • How security patches are selected • How patches are installed • How patch baseline rules work on Oracle Linux	March 1, 2021

OpsCenter displays other OpsItems for a selected 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. For more information, see Viewing other OpsItems for a specific resource.	March 1, 2021
Define patching preferences at runtime	You can now define patching preferences at runtime using the baseline override feature. For information more, see Using the BaselineOverride parameter.	February 25, 2021
New Systems Manager document type	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 AWS Systems Manager documents.	February 9, 2021

Patch instances using optional hooks	The new SSM document `AWS-RunPatchBaselineWithHooks` provides hooks you can use to run SSM documents at three points during the instance patching cycle. For information about `AWS-RunPatchBaselineWithHooks`, see About the `AWS-RunPatchBaselineWithHooks` SSM document. For a sample walkthrough of a patching operation that uses all three hooks, see Walkthrough: Update application dependencies, patch an instance, and perform an application-specific health check.	February 2, 2021

New topic: Validating on-premises servers and virtual machines using a hardware fingerprint	SSM Agent verifies the identify of on-premises servers and virtual machines and VMs that you register with the service by using a computed *fingerprint*. The fingerprint is an opaque string, stored in the *Vault* that the agent passes to certain Systems Manager APIs. For information about the hardware fingerprint and instructions for configuri ng a *similarity threshold* to assist in machine verification, see Validating on-premises servers and virtual machines using a hardware fingerprint.	January 25, 2021
New topic: SSM Agent technical reference	The topic SSM Agent technical reference 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, SSM Agent rolling updates by AWS Regions.	January 21, 2021

SSM Agent on Windows Server 2008	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
Improved support for maintenance window tasks that don't require specified targets (AWS CLI and API only)	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 Automatio n runbook parameters, you no longer need to specify a target in the task itself. For more information, see Registering maintenance window tasks without targets and Schedule automations with maintenance windows.	December 23, 2020

New Automation features	A new shared property has been added to Systems Manager Automation runbooks. The `onCancel` property allows you to specify which step the automation should go to in the event that a user cancels the automatio n. For more information, see Properties shared by all actions.	December 21, 2020
New topic: Working with associations using IAM	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 Working with associations using IAM.	December 18, 2020
State Manager now supports multi-regions and multi-acc ounts	Associations can now be created or updated with multiple regions or accounts. For more information, see Creating associations.	December 15, 2020

New tool: Fleet Manager	Fleet Manager, a tool in 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 individua l instances to perform common troubleshooting and management tasks from the console. For information, see AWS Systems Manager Fleet Manager.	December 15, 2020

New tool: Change Manager	Amazon Web Services has released Change Manager, an enterprise change management framework for requesting, approving, implementing, and reporting on operational changes to your application configura tion 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 Manager Change Manager.	December 15, 2020

New tool: Application Manager	Application Manager helps you investigate and remediate issues with your AWS resources in the context of your applicati ons. Application Manager aggregates operations information from multiple AWS services and Systems Manager tools to a single AWS Management Console. For information, see AWS Systems Manager Application Manager.	December 15, 2020

AWS Systems Manager supports Amazon EC2 instances for macOS	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 informati on, see the following topics. • For information about installing SSM Agent on EC2 instances for macOS, see Installing and configuri ng SSM Agent on EC2 instances for macOS.. • For information about patching EC2 instances for macOS, see How patches are installed, and Creating a custom patch baseline (macOS). • For general information about support for EC2 instances for macOS, see Amazon EC2 Mac instances in the *Amazon EC2 User Guide*.	November 30, 2020

Maintenance window pseudo parameters: New resource type supported for {{TARGET_ID}} and {{RESOURCE_ID}}	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 AWS::RDS::DBCluster with both these pseudo parameters. For information about maintenance window pseudo parameters, see Using pseudo parameters when registering maintenance window tasks.	November 27, 2020
Session Manager plugin for the AWS CLI version 1.2.30.0	A new version of the Session Manager plugin for the AWS CLI has been released. For more information, see Session Manager plugin latest version and release history.	November 24, 2020
New topic: Comparing SSM document versions	You can now compare the differences in content between versions of SSM documents in the Systems Manager Documents console. For more information, see Comparing SSM document versions.	November 24, 2020
Systems Manager now supports VPC endpoint policies	You can now create policies for VPC interface endpoints for Systems Manager. For more information, see Create an interface VPC endpoint policy.	November 18, 2020

New topic: Specify an idle session timeout value	You can now specify the amount of time to allow a user to be inactive before a session ends with Session Manager. For more informati on, see Specify an idle session timeout value.	November 18, 2020
New Session Manager logging feature	You can now send a continual stream of JSON-formatted session data logs to Amazon CloudWatch Logs. For more information, see Streaming session data using Amazon CloudWatch Logs.	November 18, 2020
New topic: Verify the signature of the SSM Agent	You can now verify the cryptographic signature of the installer package for the SSM Agent on Linux instances . For more information, see SSM document schemas and features.	November 17, 2020
New topic: Understanding automation statuses	A new topic has been added to the Systems Manager Automation chapter that describes the statuses for actions and automations. For more information, see Understanding automation statuses.	November 17, 2020

New source types for the `aws:downloadContent` plugin	Git and HTTP are now supported as source types for the `aws:downloadContent` plugin. For more information, see `aws:downloadContent`.	November 17, 2020
New Systems Manager document (SSM document) schema feature	In SSM documents with schema version 2.2 or later, the `precondition` parameter now supports referencing your document's input parameters. For more information, see SSM document schemas and features.	November 17, 2020
New data source in Explorer: AWS Config	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 Editing Systems Manager Explorer data sources.	November 11, 2020

New topic: Running Auto Scaling groups with associations	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 Running Auto Scaling groups with associations.	November 10, 2020
Quick Setup now supports targeting a resource group	Quick Setup now supports choosing a resource group as a target for the local setup type. For more information, see Choosing Targets for Quick Setup.	November 5, 2020
Patch Manager adds support for Debian Server 10 LTS, Oracle Linux 7.9 LTS, and Ubuntu Server 20.10 STR	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: • Patch Manager prerequisites • How security patches are selected • How patches are installed • How patch baseline rules work on Debian Server • How patch baseline rules work on Oracle Linux • How patch baseline rules work on Ubuntu Server	November 4, 2020

New EventBridge support for AWS Systems Manager Change Calendar	Amazon EventBridge now provides support for Change Calendar events events in event rules. When the state of a calendar changes, EventBridge can initiate the target action you defined an EventBridge rule. For information about working with EventBridge and Systems Manager events, see the following topics. • Configuring EventBridge for Systems Manager events • Reference: Amazon EventBridge event patterns and types for Systems Manager	November 4, 2020
Configure CloudWatch to create OpsItems from alarms	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 Configuring CloudWatch to create OpsItems from alarms.	November 4, 2020

Support for Ubuntu Server 20.10	AWS Systems Manager now supports Ubuntu Server 20.10 short-term release (STR). For more information, see the following topics: • Supported operating systems • Install SSM Agent for a hybrid environment (Linux) • Manually install SSM Agent on Ubuntu Server instances • Checking SSM Agent status and starting the agent	October 22, 2020
New topic: Allow configurable shell profiles	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 Allow configurable shell profiles.	October 21, 2020

Patch compliance results now report which CVEs are resolved by which patches	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 Viewing patch compliance results.	October 20, 2020

Expanded support for Linux patch metadata	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 DescribeAvailablePatches 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: • Viewing available patches • DescribeAvailablePatches and Patch in the *AWS Systems Manager API Reference* • describe-available-patches in the *AWS Systems Manager section of the AWS CLI Command Reference*	October 16, 2020
Session Manager plugin for the AWS CLI version 1.2.7.0	A new version of the Session Manager plugin for the AWS CLI has been released. For more information, see Session Manager plugin latest version and release history.	October 15, 2020

New topic: Session document schema	The new topic Session document schema 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
New topic: Free text search for SSM documents	The search box on the Systems Manager **Documents** 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 Using free text search.	October 15, 2020
New topic: Troubleshooting Amazon EC2 managed instance availability	The new topic Troublesh ooting Amazon EC2 managed instance availability 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

Parameter Store chapter reorganization	To help you find the information you need more efficiently, we reorganized content in the Parameter Store chapter of the *AWS Systems Manager User Guide*. Most content is now organized in the sections Setting up Parameter Store and Working with Parameter Store. In addition, the topic AWS Systems Manager Parameter Store has been expanded to include the following sections: • How can Parameter Store benefit my organization? • Who should use Parameter Store? • What are the features of Parameter Store? • What is a parameter?	October 1, 2020

New patch compliance-related topics	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. • Identifying noncompliant instances • Patching noncompliant instances • Viewing patch compliance results	September 24, 2020
SSM Agent version 3.0	Systems Manager launched a new version of SSM Agent.	September 21, 2020

New and updated topics: Amazon EventBridge replaces CloudWatch Events for event management	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 *AWS Systems Manager User Guide* have been updated to reflect EventBridge support. In addition, the following new topics have been added. • Monitoring Systems Manager events • Configuring EventBridge for Systems Manager events • Systems Manager target type examples • Reference: Amazon EventBridge event patterns and types for Systems Manager	September 18, 2020

Integrating AWS Security Hub and Patch Manager	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}}	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 Using pseudo parameters when registering maintenance window tasks.	September 14, 2020

Patch instances on demand with new '**Patch now**' option	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 Patching instances on demand.	September 9, 2020
New topic: Checking SSM Agent status and starting the agent	The new topic Checking SSM Agent status and starting the agent provides commands to check whether SSM Agent is running on each supportin g 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	You can now use Patch Manager to patch Ubuntu Server 20.04 LTS instances. For more information, see the following topics: • How security patches are selected • How patches are installed • How patch baseline rules work on Ubuntu Server	August 31, 2020
New topic for Use cases and best practices	We've added a new topic to help users quickly understan d the differences between Maintenance Windows and State Manager. For more information, see Choosing between State Manager and Maintenance Windows.	August 28, 2020
New OpsCenter features	OpsCenter include new features to help you quickly locate and run Automation runbooks to remediate issues. For more information, see Automation runbook features in OpsCenter.	August 19, 2020
New data source in Explorer: AWS Support cases	Explorer now displays information about Support cases. You must have either an Enterprise or Business account set up with Support. For more information, see Editing Systems Manager Explorer data sources.	August 13, 2020

Distributor now provides a third-party package from Trend Micro.	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 informati on, see AWSDistributor.	August 12, 2020
The `aws:configurePacka ge` document plugin now includes the additiona lArguments parameter.	The Systems Manager Command document plugin `aws:configurePacka ge` now supports providing additional parameters to your scripts (install, uninstall , and update) with the new `additionalArgument s` parameter. For more information, see the topic `aws:configurePackage` .	August 11, 2020
AppConfig content moved into a separate user guide	Information about AWS AppConfig has been moved into a separate user guide. For more information, see What Is AWSAppConfig? AppConfig also has a separate documentation landing page with links to the user guide, the AppConfig API reference , and a new AppConfig workshop.	August 3, 2020

Quick Setup now supports AWS Organizations	Quick Setup now supports AWS Organizations allowing you to quickly configure required security roles and commonly used Systems Manager tools across multiple accounts and Regions. For more information, see AWS Systems Manager Quick Setup.	July 23, 2020
New data source in Explorer: association compliance	Explorer now displays association compliance data from State Manager. For more information, see Editing Systems Manager Explorer data sources .	July 23, 2020
New Systems Manager Command document to turn on and turn off Kernel Live Patching	The document `AWS-ConfigureKernelLivePatching` 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 Use Kernel Live Patching on Amazon Linux 2 instances	July 22, 2020

Updated Automation quotas	Service quotas for Automation have been updated including a separate queue for rate control automations. For more information, see AWS Systems Manager Automation.	July 20, 2020
Specify the number of schedule offset days for a maintenance window using the console	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 Cron and rate expressions for Systems Manager and Specify the number of schedule offset days for a maintenance window.	July 17, 2020

Update PowerShell using Run Command	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 Update PowerShell using Run Command.	June 30, 2020
Patch Manager now supports CentOS 8.0 and 8.1	You can now use Patch Manager to patch CentOS 8.0 and 8.1 instances. For more information, see the the following topics: • How security patches are selected • How patches are installed • How to install the SSM Agent on hybrid Linux nodes	June 27, 2020

AppConfig integrates with AWS CodePipeline	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. • 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	June 25, 2020

	configuration store, now have additional options. CodePipeline supports AWS CodeCommit, GitHub, and BitBucket (to name a few).	
New chapter: Product and service integrations	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	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	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	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	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 `InstallOverrideList` parameter in patching operations	The new topic Sample scenario for using the `InstallOverrideList` parameter describes a strategy for using the `InstallOverrideList` parameter in the `AWS-RunPatchBaseline` document to apply different types of patches to a target group, on different maintenance window schedules, while still using a single patch baseline.	June 11, 2020
Predefined deployment strategies for AppConfig	AppConfig now offers predefined deployment strategies. For more information, see Creating a deployment strategy.	June 10, 2020

Patch Manager now supports Red Hat Enterprise Linux (RHEL) 7.8-8.2	You can now use Patch Manager to patch RHEL 7.8–8.2 instances. For more information, see the the following topics: • How security patches are selected • How patches are installed • How patch baseline rules work on RHEL • Manually install SSM Agent on Red Hat Enterprise Linux instances • How to install the SSM Agent on hybrid Linux nodes	June 9, 2020

Explorer supports delegated administration	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 Configuring a Delegated Administrator.	June 3, 2020
Apply State Manager association only at the next specified Cron interval	If you don't want a State Manager association to run immediately after you create it, you can choose the **Apply association only at the next specified Cron interval** option in the Systems Manager console. For more information, see Creating associations.	June 3, 2020

New data source in Explorer: AWS Compute Optimizer	Explorer now displays data from AWS Compute Optimizer. This includes a count of **Under provisioned** and **Over provisioned** EC2 instances, optimization findings, on-demand pricing details, and recommendations for instance type and price. For more information, see the details for setting up AWS Compute Optimizer in Setting up related services.	May 26, 2020
Install Windows Service Packs and Linux minor version upgrades using Patch Manager	The new topic Tutorial: Create a patch baseline for installing Windows Service Packs (console) demonstrates how you can create a patch baseline devoted exclusively to installing Windows Service Packs. The topic Create a custom patch baseline (Linux) has been updated with information about including minor version upgrades for Linux operating systems in patch baselines.	May 21, 2020

Parameter Store chapter reorganization	All topics that deal with configuring or setting options for Parameter Store operations have been consolidated into the Setting up Parameter Store section. This includes the topics Managing parameter tiers and Increasing Parameter Store throughput, which have been relocated from other parts of the chapter.	May 18, 2020
New topic for creating date and time strings for interacting with Systems Manager API operations.	The new topic Creating formatted date and time strings for Systems Manager describes how to create formatted date and time strings for interacting with Systems Manager API operations.	May 13, 2020
About permissions for encrypting SecureString parameters	The new topic Restricting access to Systems Manager parameters using IAM policies 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

Patch Manager now supports the Debian Server and Oracle Linux 7.6 operating systems	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: • How security patches are selected • How patches are installed • How patch baseline rules work on Debian Server • How patch baseline rules work on Oracle Linux	May 7, 2020
Create State Manager associations that target AWS Resource Groups	In addition to targeting tags, individual instances, and all instances in your AWS account, you can now create State Manager associations that target instances in AWS Resource Groups. For more information, see About targets and rate controls in State Manager associations	May 7, 2020

New `aws:ec2:image` data type in Parameter Store to validate AMI IDs	When you create a `String` parameter, you can now specify a *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 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 `aws:ec2:image` , 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 informati on, see Native parameter support for Amazon Machine Image IDs.	May 5, 2020

Managing exit codes in Run Command commands	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 Managing exit codes in Run Command commands.	May 5, 2020
New public parameters released for availability zones and local zones	Public parameters have been released to make information about AWS *availability zones* and *local zones* available programmatically. These are in addition to existing global infrastructure public parameters for AWS services and AWS Regions. For more information, see Calling public parameters for AWS services, Regions, endpoints, Availability Zones, local zones, and Wavelength Zones .	May 4, 2020

New data source in Explorer: AWS Trusted Advisor	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 Setting up related services.	May 4, 2020

Create State Manager associations that run Chef recipes	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

	For more information, see Creating associations that run Chef recipes	
Synchronize inventory data from multiple AWS accounts to a central Amazon S3 bucket	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
Store AppConfig configurations in Amazon S3	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 About configurations stored in Amazon S3.	March 13, 2020
SSM Agent installed by default on Amazon ECS-optimized AMIs	SSM Agent is now installed by default on Amazon ECS-Optimized AMIs. For more information, see Working with SSM Agent.	February 25, 2020

Create AppConfig configurations in the console	AppConfig now allows you to create an application configuration in the console at the time you create a configuration profile. For more information, see Creating a configuration and a configuration profile.	February 13, 2020
Auto-approve only patches released up to a specified date	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 About custom baselines and Working with custom patch baselines (console).	February 12, 2020

Use the {{RESOURCE_ID}} pseudo parameter in maintenance window tasks	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): • Using pseudo parameters when registering maintenance window tasks • Examples: Register tasks with a maintenance window	February 6, 2020

Quickly rerun commands	Systems Manager includes two options to help you rerun a command from the **Run Command** page in the AWS Systems Manager console. **Rerun**: This button 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 5, 2020
Reverting from the advanced-instances tier to the standard-instances tier	If you previously configure d 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 tools. For more information, see Reverting from the advanced-instances tier to the standard-instances tier.	January 16, 2020

New option to skip instance reboots after patch installation	Previously, managed instances were always rebooted after Patch Manager installed patches on them. A new `RebootOption` parameter in the SSM document `AWS-RunPatchBaseline` allows you to specify whether or not you want your instances to reboot automatically after new patches are installed. For more information, see Parameter name: RebootOption in the topic About the SSM document `AWS-RunPatchBaseline`.	January 15, 2020
New topic: 'Running PowerShell scripts on Linux instances'	A new topic that describes how to use Run Command to run PowerShell scripts on Linux instances. For more information, see Running PowerShell scripts on Linux instances.	January 10, 2020
Updates to 'configure SSM Agent to use a proxy'	The values to specify when configuring SSM Agent to use a proxy have been updated to reflect options for both HTTP proxy servers and HTTPS proxy servers. For more information, see Configure SSM Agent to use a proxy.	January 9, 2020

New "Security" chapter outlines practices for securing Systems Manager resources	A new Security chapter in the *AWS Systems Manager User Guide* helps you understand how to apply the shared responsibility model when using Systems Manager. Topics in the chapter show you how to configure Systems Manager to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Systems Manager resources. **Note** As part of this update, the user guide chapter "Authentication and Access Control" has been replaced by a new, simpler section, Identity and access management for AWS Systems Manager.	December 24, 2019

New sample custom Automation runbooks	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 Custom Automation runbook samples. You can also view Amazon managed Automation runbook content in the Systems Manager console. For more information, see Systems Manager Automation Runbook Reference.	December 23, 2019
Support for the Oracle Linux	Systems Manager now supports Oracle Linux 7.5 and 7.7. For information about manually installing SSM Agent on EC2 instances for Oracle Linux instances, see Oracle Linux. For information about installing SSM Agent on Oracle Linux servers in a hybrid environment, see How to install the SSM Agent on hybrid Linux nodes.	December 19, 2019

Launch Session Manager sessions from the Amazon EC2 console	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. • Quickstart default IAM policies for Session Manager • Starting a session (Amazon EC2 console)	December 18, 2019

CloudWatch support for Run Command metrics and alarms	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 Monitoring Run Command metrics using Amazon CloudWatch.	December 17, 2019
New Systems Manager tool: Change Calendar	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 tool: AWSAppConfig	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 AWSAppConfig.	November 25, 2019

New Systems Manager tool: 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 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	November 18, 2019

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 tools. For more information, see Getting started with Explorer and OpsCenter.

Improved parameter search capabilities	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	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. For more information, see the following topics. • Walkthrough: Using Document Builder to create a custom Automation runbook • `aws:executeScript` (Automation actions reference) • Creating Automation runbooks using Document Builder • New Automation Features In Systems Manager on the *AWS News Blog*	November 14, 2019

Perform an in-place package update using Distributor	Previously, when you wanted to install an update to a package using Distribut or, 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 applicati on can remain available and not be taken offline during the update. For more information, see the following topics. • Create a package • Install or update packages	November 11, 2019
New SSM Agent auto update feature	With one click, you can configure all instances in your AWS account to automatic ally check for and download new versions of SSM Agent. To do this, choose **Agent auto update** on the **Managed instances** page in the AWS Systems Manager console. For information, see Automate updates to SSM Agent.	November 5, 2019

Restrict Session Manager access using AWS-supplied tags	A second method for controlling user access to session actions is now available. 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	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 For more information, see Creating associations that run Ansible playbooks	September 24, 2019

Port forwarding support for Session Manager	Session Manager now supports port forwarding sessions. Port forwarding allows you to securely create tunnels between your instances deployed in private subnets, without the need to start the SSH service on the server, to open the SSH port in the security group, or to use a bastion host. Similar to SSH tunnels, port forwarding allows you to forward traffic between your laptop to open ports on your instance. Once port forwarding is configured, you can connect to the local port and access the server application running inside the instance. For more information, see the following topics: • Port Forwarding Using AWS Systems Manager Session Manager on the *AWS News Blog* • Starting a session (port forwarding)	August 29, 2019

Specify a default parameter tier or automate tier selection	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 Specifying a default parameter tier.	August 27, 2019
Working with associations section updated with CLI and PowerShell procedures	The Working with Associations section has been updated to include procedural documentation for managing associations using the AWS CLI or AWS Tools for PowerShell. For information see, Working with associations in Systems Manager.	August 26, 2019

Working with Automation executions section updated with CLI and PowerShell procedures	The Working with Automation Executions section has been updated to include procedural documentation for running Automation workflows using the AWS CLI or AWS Tools for PowerShell. For information see, Working with Automation executions.	August 20, 2019
OpsCenter integrates with application insights	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	Quick Setup is a new feature in the Systems Manager console that helps you quickly configure several Systems Manager components on your EC2 instances. Specifically, Quick Setup helps you configure the following components on the instances you choose or target by using tags: • An AWS Identity and Access Management (IAM) instance profile role for Systems Manager. • A scheduled, bi-monthly update of SSM Agent. • A scheduled collection of Inventory metadata every 30 minutes. • A daily scan of your instances to identify missing patches. • A one-time installation and configuration of the Amazon CloudWatch agent. • A scheduled, monthly update of the CloudWatch agent. For more information, see AWS Systems Manager Quick Setup.	August 7, 2019

Register a resource group as a maintenance window target	In addition to registering managed instances as the target of a maintenance window, you can now register a resource group as a maintenance window target. Maintenance Windows supports all the AWS resource types that are supported by AWS Resource Groups including `AWS::EC2::Instance`, `AWS::DynamoDB::Table`, `AWS::OpsWorks::Instance`, `AWS::Redshift::Cluster`, and more. With this release you can also send commands to a resource group, for example by using the Run Command console or the AWS CLI send-command command. For more information, see the following topics: • Assign targets to a maintenance window (console) • Examples: Register targets with a maintenance window • Using targets and rate controls to send commands to a fleet	July 23, 2019

Simplified package creation and versioning with AWS Systems Manager Distributor	Distributor has a new, simplified package creation workflow that can generate a package manifest, scripts, and file hashes for you. You can also use the simplified workflow when you add a version to an existing package.	July 22, 2019
New document categories pane for Systems Manager Automation	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.	July 18, 2019
Support for starting Session Manager sessions using operating system user credentials	By default, Session Manager sessions are launched using the credentials of a system-generated `ssm-user` 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 Turn on Run As support for Linux instances.	July 9, 2019

Support for starting Session Manager sessions using SSH	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 (Optional) Turn on SSH Session Manager sessions. For information about starting an SSH session using Session Manager, see Starting a session (SSH).	July 9, 2019
Support for changing passwords on managed instances	You can now reset passwords on machines that you manage using Systems Manager (managed instances). You can reset the password using the Systems Manager console or the AWS CLI. For information, see Resetting passwords on managed instances.	July 9, 2019
Revisions to "What is AWS Systems Manager?"	The introductory content in What is AWS Systems Manager? has been expanded to provide a broader introduct ion to the service and reflect Systems Manager tools that have been released recently. In addition, other content in the section has been moved into individual topics for better discoverability.	June 10, 2019

New Systems Manager tool: OpsCenter	OpsCenter provides a central location where operations engineers and IT professionals can view, investigate, and resolve operational work items (OpsItems) related to AWS resources. OpsCenter is designed to reduce mean time to resolution for issues impacting AWS resources. This Systems Manager tool 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 AWS Systems Manager OpsCenter.	June 6, 2019

Changes to Systems Manager left navigation pane in the AWS Management Console	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 tools.	June 6, 2019

Revised tutorial for creating and configuring a maintenance window using the AWS CLI	Tutorial: Create and configure a maintenance window (AWS CLI) has been overhauled to provide a simple path through the practice steps. You create a single maintenance window, identify a single target, and set up a simple task for the maintenance window to run. Along the way, we provide information and examples you can use to create your own task registration commands, including information for using pseudo parameters such as `{{TARGET_ID}}` . For additional information and examples, see the following topics: • Examples: Register targets with a maintenance window • Examples: Register tasks with a maintenance window • About register-task-with-maintenance-windows options • Using pseudo parameters when registering maintenance window tasks	May 31, 2019
Notifications about SSM Agent updates	To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.	May 24, 2019

Receive notifications or trigger actions based on changes in Parameter Store	The topic Set up notifications or trigger actions based on Parameter Store events 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: • A parameter is created, updated, or deleted. • A parameter label version is created, updated, or deleted. • A parameter expires, is going to expire, or hasn't changed in a specified period of time.	May 22, 2019

Major revisions to setting up and getting started content	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 recommend ed order of completion. A new *Getting Started* chapter focuses on helping end-users get started with Systems Manager after account and service configuration tasks have been completed. • Setting up AWS Systems Manager • Setting up AWS Systems Manager for hybrid environments • Getting started with AWS Systems Manager	May 15, 2019

Include patches for applications released by Microsoft in patch baselines (Windows)	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 `AWS-WindowsPredefinedPatchBaseline-OS` applies to operating system patches only. `AWS-WindowsPredefinedPatchBaseline-OS-Applications` 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 Create a custom patch baseline. Also, as part of this update, the names of AWS-provided predefined patch baselines are being changed. For more information, see Predefined baselines.	May 7, 2019

Examples for registering maintenance window targets using the AWS CLI	The new topic Examples: Register targets with a maintenance window provides three sample commands to demonstra te different ways you can specify the targets for a maintenance window when you use the AWS CLI. The topic also explains the best use case for each of the sample commands.	May 3, 2019
Updates to patch group topics	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 operation s. Additionally, instructions have been added for using the AWS CLI or Systems Manager console to add **Patch Group** or **PatchGrou p** tags to your managed instances, and how to add a **Patch Group** or **PatchGrou p** to a patch baseline. (You must use **`PatchGroup`** , without a space, if you have allowed tags in EC2 instance metadata.) For more information see Create a patch group and Add a patch group to a patch baseline.	May 1, 2019

New Parameter Store features	Parameter Store offers the following new features: • **Advanced parameter s:** Parameter Store now allows you to individua lly 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	April 25, 2019

	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	The Automation section has been updated for improved discoverability. In addition, three new topics have been added to the Automation section: • Run an automation step by step • Run an automation that requires approvals • Scheduling automations with State Manager associations	April 17, 2019

Encrypt session data using an AWS KMS key	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 Turn on AWS KMS key encryption of session data (console), Create Session Manager preferences (AWS CLI), or Update Session Manager preferences (AWS CLI).	April 4, 2019
Configuring Amazon SNS notifications for AWS Systems Manager	Added instructions for using the AWS CLI or Systems Manager console to configure Amazon SNS notifications for Run Command and Run Command tasks registered to a maintenance window. For more information see Configuring Amazon SNS notifications for AWS Systems Manager.	March 6, 2019

Advanced instances for servers and VMs in hybrid environments	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 configure d 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 Using the advanced-instances tier.	March 4, 2019

Create State Manager associations that use shared SSM documents	You can create State Manager associations that use SSM Command and Automatio n runbooks shared from other AWS accounts. Creating associations by using shared SSM documents helps to keep your Amazon EC2 and hybrid infrastructure in a consisten t state even when instances aren't in the same account. For information about sharing SSM documents, see AWS Systems Manager Documents . For information about creating a State Manager association, see Create an association.	February 28, 2019
View lists of Systems Manager events supported for Amazon EventBridge rules	The new topic Monitoring Systems Manager events with Amazon EventBridge 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

Add tags when you create Systems Manager resources	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

Automatic IAM role creation for Systems Manager Inventory	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 **Inventory Detail View** 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 **Inventory Detail View** page, Systems Manager automatically creates the `Amazon-GlueService PolicyForSSM` role and assigns the **Amazon-GlueServicePolicyForSSM-{*S3 bucket name*}** policy and the **AWSGlueServiceRole** policy to it. For more information, see Querying inventory data from multiple Regions and accounts.	February 14, 2019
Maintenance Windows walkthroughs to update SSM Agent	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 Maintenance Windows walkthroughs.	February 11, 2019

Using Parameter Store public parameters	Added short section describing Parameter Store public parameters. For more information, see Using Systems Manager public parameters.	January 31, 2019
Use the AWS CLI to create Session Manager preferences	Added instructions for using the AWS CLI to create Session Manager preferences, such as CloudWatch Logs, S3 bucket logging options, and session encryption settings. For more information, see Use the AWS CLI to create Session Manager preferences.	January 22, 2019
Executing Systems Manager automation workflows by using State Manager	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 Executing Systems Manager Automation workflows by using State Manager.	January 22, 2019

Reference updates for cron and rate expressions and maintenance window scheduling options	The reference topic Cron and rate expressions for Systems Manager has been revised. The new version provides more examples and improved explanations of how to use cron and rate expressions to schedule your maintenance windows and State Manager associations. In addition, the new topic Maintenance Windows scheduling and active period options explains how the various schedule-related options for maintenance windows (Start date, End date, Time zone, Schedule frequency) relate to one another.	December 6, 2018
Turn on SSM Agent debug logging	You can turn on SSM Agent debug logging by editing the seelog.xml.template file on the managed instance. For more information, see Turn on SSM Agent debug logging.	November 30, 2018

Support for ARM64 processor architectures	AWS Systems Manager now supports ARM64 versions of the Amazon Linux 2, Red Hat Enterprise Linux 7.6, and Ubuntu Server (18.04 LTS and 16.04 LTS) operating systems. For more information, see the instructions for installing Amazon Linux 2, RHEL, and Ubuntu Server 18.04 and 16.04 LTS with Snap packages. For more information about the A1 instance type, see General purpose instances in the *Amazon EC2 User Guide*.	November 26, 2018

Create and deploy packages by using AWS Systems Manager Distributor	Using AWS Systems Manager Distributor, you package your own software—or find AWS-provided agent software packages, such as `AmazonCloudWatchAgent` —to install on AWS Systems Manager managed instances. Distributor publishes resources, such as software packages, to AWS Systems Manager managed instances. Publishing a package advertises specific versions of the package's document—a Systems Manager document that you create when you add the package in Distributor—to managed instances that you identify by managed instance IDs, AWS account IDs, tags, or an AWS Region. For more information, see AWS Systems ManagerDistributor.	November 20, 2018

Concurrently run AWS Systems Manager Automation workflows across multiple AWS Regions and AWS accounts from a central account	You can concurrently run AWS Systems Manager automation workflows across multiple AWS Regions and AWS accounts or AWS Organizational Units (OUs) from an Automation management account. Concurrently executing Automations in multiple Regions and accounts or OUs reduces the time required to administer your AWS resources while enhancing the security of your computing environment. For more information see Executing Automation workflows in multiple AWS Regions and AWS accounts.	November 19, 2018
Query inventory data from multiple AWS Regions and AWS accounts	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 **Inventory Detail View** page in the AWS Systems Manager console. For more information see Querying Inventory data from multiple Regions and accounts.	November 15, 2018

Create State Manager associations that run MOF files	You can run Managed Object Format (MOF) files to enforce a targeted 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 associati on to scan and report if the managed instances are currently in the targeted 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 Creating associations that run MOF files.	November 15, 2018

Restrict administrative access in Session Manager sessions	Session Manager sessions are launched using the credentials of a user account that is created with default root or administrator permissions called `ssm-user`. Information about restricting administrative control for this account is now available in the topic Turn on or turn off ssm-user account administrative permissions.	November 13, 2018
YAML examples in Automation actions reference	The Automations actions reference now includes a YAML sample for each action that already includes a JSON sample.	October 31, 2018
Assign compliance severity levels to associations	You can now assign compliance severity levels to State Manager associations. These severity levels are reported in the Compliance Dashboard and can also be used to filter your compliance reports. The severity levels you can assign include Critical, High, Medium, Low, and Unspecified. For more information, see Create an association (console).	October 26, 2018

Use targets and rate controls with Automation and State Manager	Control the execution of Automations and State Manager associations across your fleet of resources by using targets, concurrency, and error thresholds. For more information see Using targets and rate controls to run Automation workflows on a fleet and Using targets and rate controls with State Manager associations.	October 23, 2018
Specify active time ranges and international time zones for maintenance windows	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 Create a maintenance window (console) and Update a maintenance window (AWS CLI).	October 9, 2018

Maintain a custom list of patches for your patch baseline in an S3 bucket	With the new 'InstallOverrideList' parameter in the SSM command document `AWS-RunPatchBaseline`, 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 Parameter name: InstallOverrideList.	October 5, 2018
Expanded control over whether patch dependencies are installed	Previously, if a patch in your Rejected patches list was identified as a dependency of another patch, it would still be installed. Now you can choose whether to install these dependencies or block them from being installed. For more information, see Create a patch baseline.	October 5, 2018

Create dynamic automation workflows with conditional branching	The `aws:branch` 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 Using conditional statements in runbooks.	September 26, 2018
Use the AWS CLI to update Session Manager preferences	Instructions for using the CLI to update Session Manager preferences, such as CloudWatch Logs and S3 bucket logging options, have been added to the *AWS Systems Manager User Guide*. For information, see Use the AWS CLI to update Session Manager preferences.	September 25, 2018
Updated SSM Agent requirement for Session Manager	Session Manager now requires SSM Agent version 2.3.68.0 or later. For more information about Session Manager prerequisites, see Complete Session Manager prerequisites.	September 17, 2018

Manage instances without opening inbound ports or maintaining bastion hosts using Session Manager	Using Session Manager, a fully managed tool in 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 Learn more about Session Manager.	September 11, 2018

Invoking other AWS services from a Systems Manager Automation workflow	You can invoke other AWS services and other Systems Manager tools in your Automation workflow by using three new Automatio n actions (or plugins) in your Automation runbooks. For more information, see For more information, see Using action outputs as inputs.	August 28, 2018
Use Systems Manager-specific condition keys in IAM policies	The topic Specifying conditions in a policy has been updated to list the IAM condition keys for Systems Manager that you can incorporate in policies. You can use these keys to specify the conditions under which a policy should take effect. The topic also includes links to example policies and other related topics.	August 18, 2018
Aggregate inventory data with groups to see which instances are and aren't configured to collect an inventory type	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 informati on, see Aggregating Inventory data.	August 16, 2018

View history and change tracking for Inventory and Configuration Compliance	You can now view history and change tracking for Inventory collected from your managed instances. You can also viewing history and changing tracking for Patch Manager patching and State Manager associations reported by Configuration Compliance. For more information, see Viewing Inventory history and change tracking.	August 9, 2018

Parameter Store integrates with Secrets Manager	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 tools. By using Parameter Store to reference Secrets Manager secrets, you create a consistent and secure process for calling and using secrets and reference data in your code and configuration scripts. For information, see Referencing AWS Secrets Manager secrets from Parameter Store parameters.	July 26, 2018

Attach labels to Parameter Store parameters	A parameter label is a user-defined alias to help you manage different versions of a parameter. When you modify a parameter, Systems Manager automatically saves a new version and increments the version number by one. A label can help you remember the purpose of a parameter version when there are multiple versions. For information, see Labeling parameters.	July 26, 2018
Create dynamic Automation workflows	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 *conditional branching*. This means that you can create Automation workflows that dynamically respond to condition changes and jump to a specified step. For information, see Using conditional statements in runbooks.	July 18, 2018

SSM Agent now pre-installed on Ubuntu Server 16.04 AMIs using Snap	Beginning with instances created from Ubuntu Server 16.04 AMIs identified with `20180627`, the SSM Agent is pre-installed using Snap packages. On instances created from earlier AMIs, you should continue using deb installer packages. For information, see About SSM Agent installations on 64-bit Ubuntu Server 16.04 instances.	July 7, 2018
Review minimum S3 permissions required by SSM Agent	The new topic Minimum S3 bucket permissions for SSM Agent provides information about the Amazon Simple Storage Service (Amazon S3) buckets that resources might need to access to perform Systems Manager operations. You can specify these buckets in a custom policy if you want to limit S3 bucket access for an instance profile or VPC endpoint to the minimum required to use Systems Manager.	July 5, 2018
View complete execution history for a specific State Manager association ID	The new topic Viewing association histories describes how to view all executions for a specific association ID and then view execution details for one or more resources.	July 2, 2018

Patch Manager introduces support for Amazon Linux 2	You can now use Patch Manager to apply patches to Amazon Linux 2 instances . For general informati on about Patch Manager operating system support, see Patch Manager prerequis ites. For information about the supported key-value pairs for Amazon Linux 2 when defining a patch filter, see PatchFilter in the *AWS Systems Manager API Reference*.	June 26, 2018
Send command output to Amazon CloudWatch Logs	The new topic Configuring Amazon CloudWatch Logs for Run Command 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	You can use AWS CloudFormation to create or delete a resource data sync for Systems Manager Inventory. To use AWS CloudFormation, add the AWS::SSM::ResourceDataSync resource to your AWS CloudFormation template. For more information, see Working with AWS CloudFormation Templates in the *AWS CloudFormation User Guide*. You can also manually create a resource data sync for Inventory as described in Creating a resource data sync for Inventory.	June 11, 2018
AWS Systems Manager User Guide update notifications now available through RSS	The HTML version of the Systems Manager User Guide now supports an RSS feed of updates that are documented in the Systems Manager Documentation update history page. The RSS feed includes updates made in June, 2018, and later. Previously announced updates are still available in the **Systems Manager documentation update history** page. Use the RSS button in the top menu panel to subscribe to the feed.	June 6, 2018

Specify an exit code in scripts to reboot managed instances	The new topic Rebooting managed instances from scripts describes how to instruct Systems Manager to reboot managed instances by specifying an exit code in scripts that you run with Run Command.	June 3, 2018
Create an event in Amazon EventBridge whenever custom inventory is deleted	The new topic Viewing inventory delete actions in EventBridge 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	You can inventory all managed instances in your AWS account by creating a global inventory association. For more information, see Inventory all managed nodes in your AWS account. **Note** Global inventory associations are available in SSM Agent version 2.0.790.0 or later. For informati on about how to update SSM Agent on your	May 3, 2018

Change	Description	Release date
	instances, see Updating the SSM Agent using Run Command.	
SSM Agent installed by default on Ubuntu Server 18	SSM Agent is installed, by default, on Ubuntu Server 18.04 LTS 64-bit and 32-bit AMIs.	May 2, 2018
New topic	The new topic Running commands using a specific document version describes how to use the document-version parameter to specify which version of an SSM document to use when the command runs.	May 1, 2018
New topic	The new topic Deleting custom inventory 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 DeleteInventory 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 Subscribing to SSM Agent notifications.	April 9, 2018
CentOS patching support	Systems Manager now supports patching CentOS instances. For information about supported CentOS versions, see Patch Manager prerequisites. For more information about how patching works, see How Patch Manager operations work.	March 29, 2018

Change	Description	Release date
New section	To provide a single source for reference information in the AWS Systems Manager User Guide, a new section has been introduced, AWS Systems Manager reference . Additional content will be added to this section as it becomes available.	March 15, 2018
New topic	The new topic Package name formats for approved and rejected patch lists 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 ChefChef InSpec. 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 information, see Using Chef InSpec profiles with Systems Manager Compliance.	March 7, 2018
New topic	The new topic Using service-linked roles for Systems Manager 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

Change	Description	Release date
New and updated topics	You can now use Patch Manager to install patches that are in a different source repository than the default one configured on the instance. This is useful for patching instances with updates not related to security; with the content of Personal Package Archives (PPA) for Ubuntu Server; with updates for internal corporate applications; and so on. You specify alternative patch source repositor ies when you create a custom patch baseline. For more information, see the following topics: • How to specify an alternative patch source repository (Linux) • Working with custom patch baselines • Create a patch baseline with custom repositories for different OS versions	February 6, 2018
New topic	The new topic SSM Command documents for patching managed nodes describes the seven SSM documents available to help you keep your managed instances patched with the latest security-related updates.	January 10, 2018
Important updates regarding Linux support	Updated various topics with the following information: • SSM Agent is installed, by default, on Amazon Linux 1 *base* AMIs dated 2017.09 and later. • Manually install SSM Agent on other versions of Linux, including non-base images like *Amazon ECS-Optimized AMIs*.	January 9, 2018

Change	Description	Release date
New topic	A new topic, SSM Command document for patching: `AWS-RunPatchBaseline`, 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, How Patch Manager operations work, 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 Command document plugin reference. For more information, see Systems Manager Automation actions reference.	December 20, 2017
New Monitoring chapter and content	A new chapter, Logging and monitoring in AWS Systems Manager, provides instructions for sending metrics and log data to Amazon CloudWatch Logs. A new topic, Sending node logs to unified CloudWatch Logs (CloudWatch agent), 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

Change	Description	Release date
New chapter	A new chapter, Identity and access management for AWS Systems Manager, provides comprehensive informati on about using AWS Identity and Access Management (IAM) and AWS Systems Manager to help secure access to your resources through the use of credentials. These credentials provide the permissions required to access AWS resources, such as accessing data stored in S3 buckets and sending commands to and reading the tags on EC2 instances.	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 AWS Systems Manager console.	December 8, 2017
Multiple changes for re:Invent 2017	• **Official launch of AWS Systems Manager**: 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 here. For more information, see What is AWS Systems Manager? • **YAML Support**: You can create SSM documents in YAML. For more information, see AWS Systems Manager Documents.	November 29, 2017

Change	Description	Release date
Using Run Command to Take VSS-Enabled Snapshots of EBS Volumes	Using Run Command, you can take application-consistent snapshots of all Amazon Elastic Block Store (Amazon EBS) volumes attached to your Amazon EC2 Windows instances. The snapshot process uses the Windows Volume Shadow Copy Service (VSS) to take image-level backups of VSS-aware applications, including data from pending transactions between these applications and the disk. Furthermore, you don't need to shut down your instances or disconnect them when you need to back up all attached volumes. For more information, see Take Microsoft VSS-Enabled Snapshots Using AWS Systems Manager in the *Amazon EC2 User Guide*.	November 20, 2017
Enhanced Systems Manager Security Available By Using VPC Endpoints	You can improve the security posture of your managed instances (including managed instances in your hybrid environment) by configuring Systems Manager to use an interface VPC endpoint. Interface endpoints are powered by PrivateLink, a technology that 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 Improve the security of EC2 instances by using VPC endpoints for Systems Manager.	November 7, 2017

Change	Description	Release date
Inventory Support for Files, Services, Windows Roles, and the Windows Registry	SSM Inventory now supports gathering the following information from your managed instances. • **Files**: Name, size, version, installed date, modification and last accessed times, and so on. • **Services**: Name, display name, status, dependent services, service type, start type, and so on. • **Windows Registry**: Registry key path, value name, value type, and value. • **Windows roles**: Name, display name, path, feature type, installed state, and so on. 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 Walkthrough: Automatically update SSM Agent with the AWS CLI. For more information Inventory, see Learn more about Systems Manager Inventory.	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 Setting up Automation.	October 31, 2017

Change	Description	Release date
GitHub and Amazon S3 Integration	**Run remote scripts**: Systems Manager now supports downloading and running scripts from a private or public GitHub repository, and from Amazon S3. Using either the `AWS-RunRemoteScript` pre-defined SSM document or the `aws:downloadContent` plugin in a custom SSM document, you can run Ansible Playbooks and scripts in Python, Ruby, or PowerShell, to name a few. These changes further enhance *infrastructure as code* when you use Systems Manager to automate configuration and deployment of EC2 instances and on-premises managed instances in your hybrid environment. For more informati on, see Running scripts from GitHub and Running scripts from Amazon S3. **Create composite SSM documents**: Systems Manager now supports running one or more secondary SSM documents from a primary SSM document. These primary documents that run other documents are called *composite* documents. Composite documents allow you to create and share a standard set of secondary SSM documents across AWS accounts for common tasks such as boot-stra pping anti-virus software or domain-joining instances. You can run composite and secondary documents stored in Systems Manager, GitHub, or Amazon S3. After you create a composite document, you can run it by using the `AWS-RunDocument` pre-defined SSM document. For more information, see Creating composite documents and Running documents from remote locations. **SSM document plugin reference**: For easier access, we moved the SSM Plugin Reference for SSM documents out of the Systems Manager API Reference and into the User Guide. For more information, see Command document plugin reference.	October 26, 2017

Change	Description	Release date
Support for Parameter Versions in Parameter Store	When you edit a parameter, Parameter Store now automatically iterates the version number by 1. You can specify a parameter name and a specific version number in API calls and SSM documents. If you don't specify a version number, the system automatically uses the latest version. Parameter versions provide a layer of protection in the event that a parameter is accidentally changed. You can view the values of all versions, and reference older versions if necessary. You can also use parameter versions to see how many times a parameter changed over a period of time. For more information, see Working with parameter versions in Parameter Store.	October 24, 2017
Support for Tagging Systems Manager Documents	You can now use the AddTagsToResource API, the AWS CLI, or the AWS Tools for PowerShell to tag Systems Manager documents with key-value pairs. Tagging helps you quickly identify specific resources based on the tags you've assigned to them. This is in addition to existing tagging support for managed instances, maintenan ce windows, Parameter Store parameters, and patch baselines.	October 3, 2017

Change	Description	Release date
Various Documentation Updates to Fix Errors or Update Content Based on Feedback	• Updated Managing nodes in hybrid and multicloud environments with Systems Manager with information for Raspbian Linux. • Updated Managing EC2 instances with Systems Manager with new requirement for Windows Server instances. SSM Agent requires Windows PowerShell 3.0 or later to run certain SSM Documents on Windows Server instances (for example, the legacy `AWS-ApplyPatchBaseline` SSM document). Verify that your Windows Server instances are running Windows Management Framework 3.0 or later. The framework includes PowerShell. For more information, see Windows Management Framework 3.0.	October 2, 2017
Troubleshoot Unreachable Windows Instances by Using the EC2Rescue Automation Workflow	EC2Rescue can help you diagnose and troubleshoot problems on Amazon EC2 Windows Server instances. You can run the tool as a Systems Manager Automation workflow by using the **AWSSupport-ExecuteEC2Rescue** document. The **AWSSupport-ExecuteEC2Rescue** document is designed to perform a combination of Systems Manager actions, AWS CloudFormation actions, and Lambda functions that automate the steps normally required to use EC2Rescue. For more information, see Run the EC2Rescue tool on unreachable instances.	September 29, 2017
SSM Agent Installed By Default on Amazon Linux	SSM Agent is installed, by default, on Amazon Linux AMIs dated 2017.09 and later. Manually install SSM Agent on other versions of Linux, as described in Working with SSM Agent on EC2 instances for Linux.	September 27, 2017

Change	Description	Release date
Run Command Enhancements	Run Command includes the following enhancements. • You can restrict command execution to specific instances by creating and assigning an IAM policy that includes a condition that the user can only run commands on instances that are tagged with specific Amazon EC2 tags. For more information, see Restricting Run Command access based on tags. • 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 Run commands at scale.	September 12, 2017
Systems Manager Supported on Raspbian	Systems Manager can now run on Raspbian Jessie and Raspbian Stretch devices, including Raspberry Pi (32-Bit).	September 7, 2017
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 Sending SSM Agent logs to CloudWatch Logs.	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 Walkthrough: Using resource data sync to aggregate inventory data.	September 1, 2017

Change	Description	Release date
New State Manager Walkthroughs	Added two new walkthroughs to the State Manager documentation: Walkthrough: Automatically update SSM Agent with the AWS CLI Walkthrough: Automatically update PV drivers on EC2 instances for Windows Server	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 AWS Systems Manager Compliance.	August 28, 2017
New Automation Action: `aws:executeAutomation`	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 `aws:executeAutomation` – Run another automation.	August 22, 2017

Change	Description	Release date
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 Run automations based on EventBridge events.	August 21, 2017
State Manager Association Versioning and General Updates	You can now create different State Manager associati on 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 AWS Systems Manager State Manager.	August 21, 2017

Change	Description	Release date
Changes to Maintenance Windows	Maintenance Windows include the following changes or enhancements: • Previously, Maintenance Windows could only perform tasks by using Run Command. You can now perform tasks by using Systems Manager Automation, AWS Lambda, and AWS Step Functions. • You can edit the targets of a maintenance window, specify a target name, description, and owner. • You can edit tasks in a maintenance window, including specifying a new SSM document for Run Command and Automation tasks. • All Run Command parameters are now supported , including DocumentHash, DocumentHashType, TimeoutSeconds, Comment, and NotificationConfig. • You can now use a `safe` flag when you attempt to deregister a target. If turned on, the system returns an error if the target is referenced by any task. For more information, see AWS Systems Manager Maintenance Windows.	August 16, 2017
New Automation Action: `aws:approve`	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. For more information, see Systems Manager Automation actions reference.	August 10, 2017

Change	Description	Release date
Automation assume role no longer required	Automation previously required that you specify a service role (or *assume role*) so that the service had permission to perform actions on your behalf. Automation no longer requires this role because the service now operates by using the context of the user who invoked the execution. However, the following situations still require that you specify a service role for Automation: • When you want to restrict a user's permissions on a resource, but you want the user to run an Automatio n 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. For more information, see Setting up Automation.	August 3, 2017
Configuration Compliance	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 AWS Systems Manager Compliance.	August 8, 2017

Change	Description	Release date
SSM Document Enhancements	SSM Command and Policy documents now offer cross-platform support. This means that a single SSM document can process plugins for Windows and Linux operating systems. Cross-platform support 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. SSM Command documents that use schema version 2.0 or later can now include multiple plugins of the same type. For example, you can create a Command document that calls the `aws:runRunShellScript` plugin multiple times. For more information about schema version 2.2 changes, see AWS Systems Manager documents. For more information about SSM plugins, see Command document plugin reference.	July 12, 2017

Change	Description	Release date
Linux Patching	Patch Manager can now patch the following Linux distributions: **64-bit and 32-bit systems** • Amazon Linux 2014.03, 2014.09, or later • Ubuntu Server 16.04 LTS, 14.04 LTS, or 12.04 LTS • Red Hat Enterprise Linux (RHEL) 6.5 or later **64-bit systems only** • Amazon Linux 2015.03, 2015.09, or later • Red Hat Enterprise Linux (RHEL) 7.x or later For more information, see AWS Systems Manager Patch Manager. **Note** • To patch Linux instances, your instances must be running SSM Agent version 2.0.834.0 or later. For information about updating the agent, see the section titled *Example: Update SSM Agent* in Running commands from the console. • The `AWS-ApplyPatchBaseline` SSM document is being replaced by the `AWS-RunPatchBaseline` document.	July 6, 2017

Change	Description	Release date
Resource data sync	You can use Systems Manager resource data sync to send Inventory data collected from all of your managed instances to a single 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 Creating a resource data sync for Inventory. For an example of how to work with resource data sync, see Walkthrough: Using resource data sync to aggregate inventory data.	June 29, 2017
Systems Manager Parameter Hierarchies	Managing dozens or hundreds of Systems Manager parameters as a flat list is time-consuming and prone to errors. You can use parameter hierarchies to help you organize and manage Systems Manager parameters. A hierarchy is a parameter name that includes a path that you define by using forward slashes. Here is an example that uses three hierarchy levels in the name to identify the following: /Environment/Type of computer/Application/Data `/Dev/DBServer/MySQL/db-string13` For more information, see Working with parameter hierarchies in Parameter Store.	June 22, 2017
SSM Agent Support for SUSE Linux Enterprise Server	You can install SSM Agent on 64-bit SUSE Linux Enterprise Server (SLES). For more information, see Working with SSM Agent on EC2 instances for Linux.	June 14, 2017

Document conventions

The following are common typographical conventions for the *AWS Systems Manager User Guide*.

Differentiated examples for local operating systems or command line languages

We use tabs to present different examples of commands based on a user's local operating system type. For Linux and macOS examples, we use the backslash (\ ) character to break long commands into multiple lines. For Windows Server examples, we use the caret (^) character to break commands into multiple lines.

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
```

Elements in the user interface

Formatting: Text in bold

Example: Choose **File**, **Properties**.

User input (text that a user types)

Formatting: Text in a monospace font

Example: For the name, type **`my-new-resource`**.

Placeholder text for a required value

Formatting: Text in *`italics`*

Example:

```
aws ec2 register-image --image-location amzn-s3-demo-bucket/image.manifest.xml
```