AWS Systems Manager
User Guide
# Table of Contents

What Is AWS Systems Manager? ................................................................. 1
Features ........................................................................................................... 1
  Resource Groups ..................................................................................... 1
  Insights ..................................................................................................... 1
  Actions ...................................................................................................... 1
  Shared Resources .................................................................................... 2
Getting Started ......................................................................................... 2
Accessing Systems Manager ...................................................................... 3
Pricing .......................................................................................................... 3
Related Content ......................................................................................... 3

Setting Up .................................................................................................. 4
Prerequisites ............................................................................................... 4

Ec2messages and Undocumented API Calls .............................................. 7

Configuring Access ................................................................................... 8
  Task 1: Configure User Access for Systems Manager ................................ 8
  Task 2: Create an Instance Profile Role for Systems Manager ............... 8
  Task 3: Create an Amazon EC2 Instance that Uses the Systems Manager Role ......................................................... 9
  Optional Access Configurations .............................................................. 10

Setting Up VPC Endpoints for Systems Manager .................................... 12
  Creating VPC EndPoints for Systems Manager ....................................... 12

Installing and Configuring SSM Agent ..................................................... 13
  Installing and Configuring SSM Agent on Windows Instances ............ 13
  Installing and Configuring SSM Agent on Linux Instances ................. 15

Hybrid Environments ................................................................................. 27
  Create an IAM Service Role .................................................................... 28
  Create a Managed-Instance Activation ................................................... 29
  Install the SSM Agent on Servers and VMs in Your Windows Hybrid Environment ......................................................... 30
  Install the SSM Agent on Servers and VMs in Your Linux Hybrid Environment ................................................................. 31

Cron Schedules .......................................................................................... 33
  Cron and Rate Expressions for Associations ......................................... 34
  Cron and Rate Expressions for Maintenance Windows ....................... 35
  General Information About Cron and Rate Expressions ..................... 35

Partner and Product Integration ............................................................... 38
Running Scripts from GitHub and Amazon S3 .......................................... 38
  Running Scripts from GitHub .................................................................. 38
  Running Scripts from Amazon S3 ............................................................ 44
Using Run Command to Take VSS-Enabled Snapshots of EBS Volumes ............................................................................... 50
  How It Works .......................................................................................... 50
  Before You Begin .................................................................................... 51
  Creating VSS-enabled EBS snapshots ..................................................... 54
  Creating VSS-Enabled EBS Snapshots by Using the AWSEC2-ManageVssIO SSM Document (Advanced) ................................................................. 57
  Restoring Volumes from VSS-enabled EBS snapshots ......................... 58

Resource Groups ....................................................................................... 60
Insights ....................................................................................................... 61
Inventory .................................................................................................... 61
  Getting Started with Inventory .............................................................. 61
  About Systems Manager Inventory ....................................................... 61
  Configuring Inventory Collection ........................................................ 69
  Configuring Resource Data Sync for Inventory .................................... 71
  Querying Inventory Collection .............................................................. 74
  Inventory Walkthroughs ........................................................................ 74

Compliance ................................................................................................. 84
  Getting Started with Configuration Compliance ................................. 85
What Is AWS Systems Manager?

AWS Systems Manager (formerly Amazon EC2 Systems Manager) is a unified interface that allows you to easily centralize operational data and automate tasks across your AWS resources. Systems Manager shortens the time to detect and resolve operational problems in your infrastructure. Systems Manager gives you a complete view of your infrastructure performance and configuration, simplifies resource and application management, and makes it easy to operate and manage your infrastructure at scale.

Features

Systems Manager includes the following features:

Resource Groups

AWS Resource Groups: A resource group is a collection of AWS resources that are all in the same AWS region, and that match criteria provided in a query. You build queries in the Resource Groups console, or pass them as arguments to Resource Groups commands in the AWS CLI. With Resource Groups, you can create a custom console that organizes and consolidates information based on criteria that you specify in tags. After you've created groups in Resource Groups, use AWS Systems Manager tools such as Automation, Run Command, Patch Manager, and Maintenance Windows to simplify management tasks on your groups of resources. You can also use groups as the basis for viewing monitoring and configuration insights in AWS Systems Manager.

Insights

Systems Manager provides the following features and capabilities for centrally viewing data about your AWS resources.

- Built-in Insights: Insights show detailed information about the resources in your AWS Resource Groups, such as AWS CloudTrail logs, results of evaluations against AWS Config rules, and AWS Trusted Advisor reports. Insights show information about a single, selected resource group at a time.
- Amazon CloudWatch Dashboard: CloudWatch dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those resources that are spread across different regions. You can use CloudWatch dashboards to create customized views of the metrics and alarms for your AWS resources.
- Inventory Management (p. 61): Inventory Manager automates the process of collecting software inventory from managed instances. You can use Inventory Manager to gather metadata about OS and system configurations and application deployments.
- Configuration Compliance (p. 84): Use Configuration Compliance to scan your fleet of managed instances for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren’t compliant. By default, Configuration Compliance displays compliance data about Patch Manager patching and State Manager associations. You can also customize the service and create your own compliance types based on your IT or business requirements.

Actions

Systems Manager provides the following capabilities for taking action against your AWS resources.
• **Automation (p. 92):** Automation automates common maintenance and deployment tasks. You can use Automation to create and update Amazon Machine Images, apply driver and agent updates, and apply OS patches or application updates.

• **Run Command (p. 160):** Run Command helps you remotely and securely manage the configuration of your managed instances at scale. Use Run Command to perform on-demand changes like updating applications or running Linux shell scripts and Windows PowerShell commands on a target set of dozens or hundreds of instances.

• **Patch Management (p. 192):** Patch Manager automates the process of patching your managed instances. This feature enables you to scan instances for missing patches and apply missing patches individually or to large groups of instances by using Amazon EC2 instance tags. Patch Manager uses patch baselines that include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. You can install patches on a regular basis by scheduling patching to run as a Systems Manager Maintenance Window task.

• **Maintenance Windows (p. 218):** Maintenance Windows let you set up recurring schedules for managed instances to execute administrative tasks like installing patches and updates without interrupting business-critical operations.

• **State Management (p. 249):** State Manager automates the process of keeping your managed instances in a defined state. You can use State Manager to ensure that your instances are bootstrapped with specific software at startup, joined to a Windows domain (Windows instances only), or patched with specific software updates.

## Shared Resources

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

• **Managed Instances (p. 4):** A managed instance is any Amazon EC2 instance or on-premise machine (server or virtual machine [VM]) in your hybrid environment that is configured for Systems Manager. To set up managed instances, you need to install SSM agent on your machines (if not installed by default) and configure AWS Identity and Access Management (IAM) permissions. On-premises machines also require an activation code.

• **Activations (p. 27):** To set up servers and VMs in your hybrid environment as managed instances, you need to create a managed-instance activation. After you complete the activation, you receive an activation code and ID. This code/ID combination functions like an Amazon EC2 access ID and secret key to provide secure access to the Systems Manager service from your managed instances.

• **Systems Manager Documents (p. 258):** A Systems Manager document (SSM document) defines the actions that Systems Manager performs on your managed instances. Systems Manager includes more than a dozen pre-configured documents that you can use by specifying parameters at runtime. Documents use JavaScript Object Notation (JSON) or YAML, and include steps and parameters that you specify.

• **Parameter Store (p. 330):** Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, 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.

## Getting Started

To get started with Systems Manager, verify prerequisites, configure roles and permissions, and install the SSM Agent on your instances. If you want to manage your on-premises servers and VMs with Systems Manager, then you must also create a managed instance activation. These tasks are described in Setting Up AWS Systems Manager (p. 4).
Accessing Systems Manager

You can access Systems Manager using any of the following interfaces:

- The [AWS Systems Manager console](#) — Provides a web interface that you can use to access Systems Manager.
- [AWS Command Line Interface (AWS CLI)](#) — Provides commands for a broad set of AWS services, including Systems Manager, and is supported on Windows, Mac, and Linux. For more information, see [AWS Command Line Interface](#).
- [AWS SDKs](#) — Provides language-specific APIs and takes care of many of the connection details, such as calculating signatures, handling request retries, and error handling. For more information, see [AWS SDKs](#).
- [Query API](#) — Provides low-level API actions that you call using HTTPS requests. Using the Query API is the most direct way to access Systems Manager, but it requires that your application handle low-level details such as generating the hash to sign the request, and error handling. For more information, see the [Amazon EC2 Systems Manager API Reference](#).

Pricing

Systems Manager features and shared components are offered at no additional cost. You pay only for the AWS resources that you use.

Related Content

Systems Manager is also documented in the following references.

- [Amazon EC2 Systems Manager API Reference](#)
- [Systems Manager AWS Tools for Windows PowerShell](#)
- [Systems Manager AWS CLI Reference](#)
- [AWS SDKs](#)
- [AWS Systems Manager Limits](#)
## Setting Up AWS Systems Manager

This section describes tasks and prerequisites for setting up AWS Systems Manager. Use the following table to help you get started.

<table>
<thead>
<tr>
<th>What do you want to do with Systems Manager?</th>
<th>Set up tasks</th>
</tr>
</thead>
</table>
| Test it out / Play with it                  | 1. Verify permissions and create an instance profile role (p. 8).  
2. Create a few Amazon EC2 test instances (free tier) from recent Amazon Linux or Windows AMIs.  
3. Test out Systems Manager. Here are some walkthroughs to help you get started.  
   **Note**  
   Some of these walkthroughs require additional setup tasks, such as additional permissions, before you can complete them.  
   - Run Command (EC2 console): Linux or Windows  
   - Run Command Walkthroughs (p. 180) (AWS CLI or AWS Tools for Windows PowerShell)  
   - State Manager Walkthroughs (p. 254)  
   - Parameter Store Walkthroughs (p. 351)  
   - Inventory Manager Walkthroughs (p. 74)  
   - Automation Walkthroughs (p. 103)  
   - Maintenance Window Walkthroughs (p. 231)  
   - Patch Manager Walkthroughs (p. 198) |
| Use Systems Manager to manage and configure my existing EC2 instances | 1. Verify permissions and create an instance profile role (p. 8).  
2. Verify that your EC2 instances meet Systems Manager requirements (p. 4).  
3. (Linux only) Install SSM Agent (p. 8). |
| Use Systems Manager to manage and configure my servers and VMs in a hybrid environment | 1. Verify permissions and create an instance profile role (p. 8).  
2. Verify that your servers and VMs in your hybrid environment meet Systems Manager requirements (p. 4).  
3. Perform setup and activation tasks for managed instances in a hybrid environment. |

## Systems Manager Prerequisites

Systems Manager includes the following prerequisites.
## Prerequisites

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Operating System (Linux)</td>
<td>Instances must run a supported version of Linux.</td>
</tr>
<tr>
<td><strong>64-Bit and 32-Bit Systems</strong></td>
<td></td>
</tr>
<tr>
<td>• Amazon Linux 2014.09, 2014.03 or later</td>
<td></td>
</tr>
<tr>
<td>• Ubuntu Server 16.04 LTS, 14.04 LTS, or 12.04 LTS</td>
<td></td>
</tr>
<tr>
<td>• Red Hat Enterprise Linux (RHEL) 6.5</td>
<td></td>
</tr>
<tr>
<td>• CentOS 6.3 or later</td>
<td></td>
</tr>
<tr>
<td><strong>32-Bit Systems Only</strong></td>
<td></td>
</tr>
<tr>
<td>• Raspbian Jessie</td>
<td></td>
</tr>
<tr>
<td>• Raspbian Stretch</td>
<td></td>
</tr>
<tr>
<td><strong>64-Bit Systems Only</strong></td>
<td></td>
</tr>
<tr>
<td>• Amazon Linux 2015.09, 2015.03 or later</td>
<td></td>
</tr>
<tr>
<td>• Red Hat Enterprise Linux (RHEL) 7.4</td>
<td></td>
</tr>
<tr>
<td>• CentOS 7.1 or later</td>
<td></td>
</tr>
<tr>
<td>• SUSE Linux Enterprise Server (SLES) 12 or higher</td>
<td></td>
</tr>
<tr>
<td>Supported Regions</td>
<td>Systems Manager is available in these regions.</td>
</tr>
<tr>
<td>For servers and VMs in your hybrid environment,</td>
<td>we recommend that you choose the region closest to your data center or computing environment.</td>
</tr>
<tr>
<td>Access to Systems Manager</td>
<td>Systems Manager requires an IAM role for instances that will process commands and a separate role for users executing commands. Both roles require permission policies that enable them to communicate with the Systems Manager API. You can choose to use Systems Manager managed policies or you can create your own roles and specify permissions. For more information, see Configuring Access to Systems Manager (p. 8). If you are configuring on-premises servers or VMs that you want to configure using Systems Manager, you must also configure an IAM service role. For more information, see Create an IAM Service Role (p. 28).</td>
</tr>
<tr>
<td>SSM Agent (EC2 Windows instances)</td>
<td>SSM Agent processes Systems Manager requests and configures your machine as specified in the request. The SSM Agent is installed by default on Windows Server 2016 instances and instances</td>
</tr>
<tr>
<td>Requirement</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>created from Windows Server 2003-2012 R2 AMIs published in November 2016 or later. Windows AMIs published before November 2016 use the EC2Config service to process requests and configure instances. Unless you have a specific reason for using the EC2Config service or an earlier version of the SSM Agent to process Systems Manager requests, we recommend that you download and install the latest version of the SSM Agent to each of your Amazon EC2 instances or managed instances (servers and VMs in a hybrid environment). For more information, see Installing and Configuring SSM Agent on Windows Instances (p. 13).</td>
</tr>
<tr>
<td>SSM Agent (EC2 Linux instances)</td>
<td>SSM Agent processes Systems Manager requests and configures your machine as specified in the request. You must download and install SSM Agent to your EC2 Linux instances. For more information, see Installing and Configuring SSM Agent on Linux Instances (p. 15). 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 does not currently provide support for running modified copies of this software.</td>
</tr>
<tr>
<td>SSM Agent (hybrid environment)</td>
<td>The SSM Agent download and installation process for managed instances in a hybrid environment is different than Amazon EC2 instances. For more information, see Install the SSM Agent on Servers and VMs in Your Windows Hybrid Environment (p. 30).</td>
</tr>
<tr>
<td>Windows PowerShell 3.0 or Later</td>
<td>SSM Agent requires Windows PowerShell 3.0 or later to execute certain SSM Documents on Windows instances (for example, the AWS-ApplyPatchBaseline document). Verify that your Windows instances are running Windows Management Framework 3.0 or later. The framework includes PowerShell. For more information, see Windows Management Framework 3.0.</td>
</tr>
<tr>
<td>Internet Access</td>
<td>Verify that your EC2 instances have outbound Internet access. Inbound Internet access is not required.</td>
</tr>
</tbody>
</table>
Requirement | Description
--- | ---
Configure Monitoring and Notifications (Optional) | You can configure Amazon CloudWatch Events to log status execution changes of the commands you send using Systems Manager. You can also configure Amazon Simple Notification Service (Amazon SNS) to send you notifications about specific command status changes. For more information, see Setting Up Events and Notifications (p. 164).
Amazon S3 Bucket (Optional) | You can store System Manager output in an Amazon Simple Storage Service (Amazon S3) bucket. Output in the Amazon EC2 console is truncated after 2500 characters. Additionally, you might want to create an Amazon S3 key prefix (a subfolder) to help you organize output. For more information, see Create a Bucket.

For information about Systems Manager limits, see AWS Systems Manager Limits. To increase limits, go to AWS Support Center and submit a limit increase request form.

Ec2messages and Undocumented API Calls

If you monitor API calls, you will see calls to the following APIs.

- ec2messages:AcknowledgeMessage
- ec2messages:DeleteMessage
- ec2messages:FailMessage
- ec2messages:GetEndpoint
- ec2messages:GetMessages
- ec2messages:SendReply
- UpdateInstanceInformation
- ListInstanceAssociations
- DescribeInstanceProperties
- DescribeDocumentParameters

Calls to ec2messages:* APIs are calls to the ec2messages endpoint. Systems Manager uses this endpoint to make calls from the SSM Agent to the Systems Manager service in the cloud. This endpoint is required to send and receive commands.

UpdateInstanceInformation: SSM Agent calls the Systems Manager service in the cloud every five minutes to provide heartbeat information. This call is necessary to maintain a heartbeat with the agent so that the service knows the agent is functioning as expected.

ListInstanceAssociations: The agent calls this API to see if a new Systems Manager State Manager association is available. This API is required for State Manager to function.

DescribeInstanceProperties and DescribeDocumentParameters: Systems Manager calls these APIs to render specific nodes in the Amazon EC2 console. The DescribeInstanceProperties API
Configuring Access displays the Managed Instances node in the left navigation. The DescribeDocumentParameters API displays the Documents node in the left navigation.

Configuring Access to Systems Manager

Complete the following tasks to configure access for AWS Systems Manager.

**Note**
For more information about access permissions for Systems Manager, see Authentication and Access Control for AWS Systems Manager (p. 367).

Contents
- Task 1: Configure User Access for Systems Manager (p. 8)
- Task 2: Create an Instance Profile Role for Systems Manager (p. 8)
- Task 3: Create an Amazon EC2 Instance that Uses the Systems Manager Role (p. 9)
- Optional Access Configurations (p. 10)

Task 1: Configure User Access for Systems Manager

If your IAM user account, group, or role is assigned administrator permissions, then you have access to Systems Manager. You can skip this task. If you don't have administrator permissions, then an administrator must update your IAM user account, group, or role to include the following permissions:

- **To access Resource Groups**: You must add the resource-groups:* permissions entity to your IAM user account, group, or role. For more information, see Setting Up Permissions in the AWS Resource Groups user guide.
- **To access Insights**: You must add the following managed policies to your user account, group, or role:
  - AWSHealthFullAccess
  - AWSConfigUserAccess
  - CloudWatchReadOnlyAccess

  **Note**
  Access to Inventory and Compliance are covered by the next policies.

- **To access Actions and Shared Resources**: You must add either the AmazonSSMFullAccess policy or the AmazonSSMReadOnlyAccess policy.

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

Task 2: Create an Instance Profile Role for Systems Manager

By default, Systems Manager doesn't have permission to perform actions on your instances. You must enable access by creating an IAM instance profile role, as described here. After you create the role, you can assign it to existing instances, or you can create new instances that uses this role, as described in Task 3.

**Note**
If you are configuring servers or virtual machines (VMs) in a hybrid environment for Systems Manager, you don't need to create the instance profile role. Instead, you must configure your servers and VMs to use an IAM service role. For more information, see Create an IAM Service Role (p. 28).
To create an instance profile role for Systems Manager managed instances

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles, and then choose Create role.
4. In the Select your use case section, choose EC2 Role for Simple Systems Manager, and then choose Next: Permissions.
5. On the Attached permissions policy page, verify that AmazonEC2RoleforSSM is listed, and then choose Next: Review.
6. On the Review page, type a name in the Role name box, and then type a description.

   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.
7. Choose Create role. The system returns you to the Roles page.

   Note
   This procedure created a new role from a pre-existing IAM policy or managed policy. If you choose to create a role from a custom policy, you must add ssm.amazonaws.com as a trusted entity to your role (after you create it). You add trusted entities on the Trust Relationship tab when viewing the role. For example, you must add the following JSON block to the policy as a trusted entity. For information about how to update a role to include a trusted entity, see Modifying a Role.

   ```json
   {
   "Version":"2012-10-17",
   "Statement": [
   {
   "Sid":"
   "Effect":"Allow",
   "Principal": {
   "Service": [
   "ec2.amazonaws.com",
   "ssm.amazonaws.com"
   ],
   "Action": "sts:AssumeRole"
   }
   ]
   }
   }
   ```

For information about how to attach the role you just created to a new instance or an existing instance, see the next task.

**Task 3: Create an Amazon EC2 Instance that Uses the Systems Manager Role**

This procedure describes how to launch an Amazon EC2 instance that uses the role you just created. You can also attach the role to an existing instance. For more information, see Attaching an IAM Role to an Instance in the Amazon EC2 User Guide.

**To create an instance that uses the Systems Manager instance role**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Select a supported region.
3. Choose **Launch Instance** and select an instance.
4. Choose your instance type and then choose **Next: Configure Instance Details**.
5. In the **IAM role** drop-down list choose the EC2 instance role you created earlier.
6. Complete the wizard.

If you create other instances that you want to configure using Systems Manager, you must specify the instance profile role for each instance.

## Optional Access Configurations

Task 1 in this section enabled you to grant access to a user by choosing a pre-existing or *managed* IAM user policy. If you want to limit user access to Systems Manager and SSM documents, you can create your own restrictive user policies, as described in this section. For more information about how to create a custom policy, see [Creating a New Policy](#).

The following example IAM policy allows a user to do the following.

- List Systems Manager documents and document versions.
- View details about documents.
- Send a command using the document specified in the policy.

The name of the document is determined by this entry:


- Send a command to three instances.

The instances are determined by the following entries in the second **Resource** section:

```
"arn:aws:ec2:us-east-1:*:instance/i-1234567890abcdef0",
"arn:aws:ec2:us-east-1:*:instance/i-0598c7d356eba48d7",
"arn:aws:ec2:us-east-1:*:instance/i-345678abcdef12345",
```

- View details about a command after it has been sent.
- Start and stop Automation executions.
- Get information about Automation executions.

If you want to give a user permission to use this document to send commands on any instance for which the user currently has access (as determined by their AWS user account), you could specify the following entry in the **Resource** section and remove the other instance entries.

```
"arn:aws:ec2:us-east-1:*:instance/**
```

Note that the **Resource** section includes an Amazon S3 ARN entry:

`arn:aws:s3:::bucket_name`

You can also format this entry as follows:

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

- or -

---

10
Optional Access Configurations

```
{  
  "Version":"2012-10-17",  
  "Statement":[  
    {  
      "Action":[  
        "ssm:ListDocuments",  
        "ssm:ListDocumentsVersions",  
        "ssm:DescribeDocument",  
        "ssm:GetDocument",  
        "ssm:DescribeInstanceInformation",  
        "ssm:DescribeDocumentParameters",  
        "ssm:DescribeInstanceProperties"
      ],  
      "Effect":"Allow",  
      "Resource":"*
    },  
    {  
      "Action":"ssm:SendCommand",  
      "Effect":"Allow",  
      "Resource": [  
        "arn:aws:ec2:us-east-1:*:instance/i-123456789abcdef0",  
        "arn:aws:ec2:us-east-1:*:instance/i-0598c7d356eba48d7",  
        "arn:aws:ec2:us-east-1:*:instance/i-345678abcdef12345",  
        "arn:aws:s3:::bucket_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:::automation-definition/
      ]
    },  
    {  
      "Action": "ssm:DescribeAutomationExecutions ",  
      "Effect":"Allow",  
      "Resource": [
        "*
      ]
    },  
    {  
      "Action": [  
        "ssm:StopAutomationExecution",  
        "ssm:GetAutomationExecution"
      ],  
      "Effect":"Allow",  
      "Resource": [  
        "*
      ]
    },  
    {  
      "Action": [  
        "ssm:DescribeAutomationExecutions ",  
        "ssm:GetAutomationExecution"
      ],  
      "Effect":"Allow",  
      "Resource": [  
        "*
      ]
    },
    {  
      "Action": [  
        "ssm:CancelAutomationExecution"
      ],  
      "Effect": "Allow",  
      "Resource": [  
        "*
      ]
    }
  ]
}
```
Setting Up VPC Endpoints for Systems Manager

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

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

Before You Begin

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

- VPC endpoints do not support Active Directory directory service, Amazon CloudWatch Events, or Amazon CloudWatch Logs. If you configure your managed instances to use a VPC endpoint, you won't be able to use these services.
- VPC endpoints currently do not support cross-region requests—ensure that you create your endpoint in the same region as your bucket. You can find the location of your bucket by using the Amazon S3 console, or by using the get-bucket-location command. Use a region-specific Amazon S3 endpoint to access your bucket; for example, mybucket.s3-us-west-2.amazonaws.com. For more information about region-specific endpoints for Amazon S3, see Amazon Simple Storage Service (S3) in Amazon Web Services General Reference. If you use the AWS CLI to make requests to Amazon S3, set your default region to the same region as your bucket, or use the --region parameter in your requests.
- VPC endpoints only support Amazon-provided DNS through Route 53. If you want to use your own DNS, you can use conditional DNS forwarding. For more information, see DHCP Options Sets in the Amazon VPC User Guide.

Creating VPC Endpoints for Systems Manager

Use the following procedure to create three separate VPC endpoints for Systems Manager. All three endpoints are required for Systems Manager to work in a VPC. This procedure links to related procedures in the Amazon VPC User Guide.

To create VPC endpoints for Systems Manager

1. Use Creating an Interface Endpoint to create the following endpoints:
   - com.amazonaws.region_code.ssm: The endpoint for the Systems Manager service.
   - com.amazonaws.region_code.ec2messages: Systems Manager uses this endpoint to make calls from SSM Agent to the Systems Manager service.

2. Use Creating a Gateway Endpoint to create an endpoint for Amazon S3. Systems Manager uses this endpoint to upload Amazon S3 output logs, and to update SSM Agent.
Installing and Configuring SSM Agent

AWS Systems Manager Agent is Amazon software that runs on your Amazon EC2 instances and your hybrid instances that are configured for Systems Manager (hybrid instances). SSM Agent processes requests from the Systems Manager service in the cloud and configures your machine as specified in the request. SSM Agent sends status and execution information back to the Systems Manager service by using the EC2 Messaging service. If you monitor traffic, you will see your instances communicating with ec2messages.* endpoints. For more information, see Ec2messages and Undocumented API Calls (p. 7).

SSM Agent is installed, by default, on Amazon EC2 Windows instances and Amazon Linux instances. You must manually install the agent on other versions of Linux and hybrid instances.

Note
The SSM Agent download and installation process for hybrid instances is different than Amazon EC2 instances. For more information, see Install the SSM Agent on Servers and VMs in Your Windows Hybrid Environment (p. 30).

Use the following procedures to install, configure, or uninstall SSM Agent. This section also includes information about configuring SSM Agent to use a proxy. For information about porting SSM Agent logs to Amazon CloudWatch Logs, see Monitoring Instances with AWS Systems Manager (p. 359).

Contents
- Installing and Configuring SSM Agent on Windows Instances (p. 13)
- Installing and Configuring SSM Agent on Linux Instances (p. 15)

Installing and Configuring SSM Agent on Windows Instances

SSM Agent is installed by default on Windows Server 2016 instances and instances created from Windows Server 2003-2012 R2 AMIs published in November 2016 or later.

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

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

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

Topics
- Install and Configure SSM Agent on Windows Instances (p. 13)
- View SSM Agent Logs on Windows Instances (p. 14)
- Configure SSM Agent to Use a Proxy for Windows Instances (p. 14)

Install and Configure SSM Agent on Windows Instances

If your instance is a Windows Server 2003-2012 R2 instance created before November 2016, then EC2Config processes Systems Manager requests on your instance. We recommend that you upgrade your existing instances to use the latest version of EC2Config. By upgrading, you install SSM Agent side-by-
side with EC2Config. This version of SSM Agent is compatible with your instances created from earlier Windows AMIs and enables you to use SSM features published after November 2016.

You can automatically update SSM Agent on your instances when new versions become available by using Systems Manager State Manager. For more information, see Walkthrough: Automatically Update the SSM Agent (p. 254).

Alternatively, you can manually download and install the latest version of SSM Agent using the following procedure.

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

1. Login to your instance.
2. Download the latest version of SSM Agent to your instance.
   
   ![Download URL](https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/AmazonSSMAgentSetup.exe)

   This URL lets 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:
   
   ![Region-specific URL](https://amazon-ssm-region.s3.amazonaws.com/latest/windows_amd64/AmazonSSMAgentSetup.exe)

3. Start or restart SSM Agent (AmazonSSMAgent.exe) using the Windows Services control panel or by sending the following command in PowerShell:

   ```powershell
   Restart-Service AmazonSSMAgent
   ```

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

**View SSM Agent Logs on Windows Instances**

SSM Agent writes information about executions, scheduled actions, errors, and health statuses to log files on each instance. You can view log files by manually connecting to an instance, or you can automatically send logs to Amazon CloudWatch Logs. For more information about sending logs to CloudWatch, see Monitoring Instances with AWS Systems Manager (p. 359).

You can view SSM Agent log files on Windows instances in the following locations.

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

**Configure SSM Agent to Use a Proxy for Windows Instances**

For information about configuring EC2Config to use a proxy, see Configure Proxy Settings for the EC2Config Service.

**To configure SSM Agent to use a proxy**

1. Using Remote Desktop or Windows PowerShell, connect to the instance that you would like to configure to use a proxy. For Windows Server 2016 instances that use the Nano installation option
(Nano Server), you must connect using PowerShell. For more information, see Connect to a Windows Server 2016 Nano Server Instance.

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

```powershell
$serviceKey = "HKLM:\SYSTEM\CurrentControlSet\Services\AmazonSSMAgent"
$keyInfo = (Get-Item -Path $serviceKey).GetValue("Environment")
$proxyVariables = @("http_proxy=hostname:port", "no_proxy=169.254.169.254")

If($keyInfo -eq $null)
{
    New-ItemProperty -Path $serviceKey -Name Environment -Value $proxyVariables -PropertyType MultiString -Force
} else {
    Set-ItemProperty -Path $serviceKey -Name Environment -Value $proxyVariables
}
Restart-Service AmazonSSMAgent
```

To reset SSM Agent proxy configuration

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

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

Installing and Configuring SSM Agent on Linux Instances

The Amazon EC2 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.

**Note**

SSM Agent is installed, by default, on Amazon Linux AMIs dated 2017.09 and later. You must manually install SSM Agent on other versions of Linux, as described in this section. For backwards compatibility, this section still includes information about manually installing the agent on Amazon Linux.

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

**Note**

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

**Topics**

- Install SSM Agent on Amazon EC2 Linux Instances at Launch (p. 16)
- Manually Install SSM Agent on Amazon EC2 Linux Instances (p. 18)
- Configure SSM Agent to Use a Proxy (p. 25)
Install SSM Agent on Amazon EC2 Linux Instances at Launch

You can install SSM Agent when you launch an instance for the first time by using Amazon EC2 User Data. On the **Configure Instance Details** page of the launch wizard, expand **Advanced Details** and then copy and paste one of the following scripts into the **User data** field. For example:

```bash
#!/bin/bash
cd /tmp
sudo yum install -y https://s3.amazonaws.com/ec2-downloads甸/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
sudo systemctl start amazon-ssm-agent
```

**Note**

- The process for installing SSM Agent on Raspbian Jessie devices is different than other Linux versions. For more information, see Raspbian (p. 24).
- The URLs in the following scripts let you download the SSM Agent from **any** AWS region. If you want to download the agent from a **specific** region, copy the URL for your operating system, and then replace `region` with a **region where SSM is available**.

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

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

  If the download fails, try replacing `https://s3-region` with `https://s3.region`.

- Amazon Linux, RHEL, CentOS, and SLES 64-bit:
  ```bash
  https://s3-region.amazonaws.com/amazon-ssm-region/latest/linux_amd64/amazon-ssm-agent.rpm
  ```

- Amazon Linux, RHEL, and CentOS 32-bit:
  ```bash
  ```

- Ubuntu Server 64-bit:
  ```bash
  ```

- Ubuntu Server 32-bit:
  ```bash
  ```

- Raspbian:
Installing and Configuring SSM Agent on Linux Instances

---

**RHEL 7.x and CentOS 7.x, 64-bit**

```bash
#!/bin/bash
cd /tmp
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
sudo systemctl start amazon-ssm-agent
```

**RHEL 7.x and CentOS 7.x, 32-bit**

```bash
#!/bin/bash
cd /tmp
sudo systemctl start amazon-ssm-agent
```

**Amazon Linux, RHEL 6.x, and CentOS 6.x, 64-bit**

*Note*
SSM Agent is installed, by default, on Amazon Linux AMIs dated 2017.09 and later.

```bash
#!/bin/bash
cd /tmp
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
sudo systemctl start amazon-ssm-agent
```

**Amazon Linux, RHEL 6.x, and CentOS 6.x, 32-bit**

*Note*
SSM Agent is installed, by default, on Amazon Linux AMIs dated 2017.09 and later.

```bash
#!/bin/bash
cd /tmp
sudo systemctl start amazon-ssm-agent
```

**Ubuntu Server 16 64-bit**

```bash
#!/bin/bash
cd /tmp
wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_amd64/amazon-ssm-agent.deb
sudo dpkg -i amazon-ssm-agent.deb
sudo systemctl enable amazon-ssm-agent
```

**Ubuntu Server 16 32-bit**

```bash
#!/bin/bash
cd /tmp
```
## Installing and Configuring SSM Agent on Linux Instances

### SLES 64-bit

```bash
#!/bin/bash
cd /tmp
wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
sudo rpm --install amazon-ssm-agent.rpm
sudo systemctl start amazon-ssm-agent
```

### Ubuntu Server 14 64-bit

```bash
#!/bin/bash
cd /tmp
wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_amd64/amazon-ssm-agent.deb
sudo dpkg -i amazon-ssm-agent.deb
sudo start amazon-ssm-agent
```

### Ubuntu Server 14 32-bit

```bash
#!/bin/bash
cd /tmp
sudo dpkg -i amazon-ssm-agent.deb
sudo start amazon-ssm-agent
```

Save your changes, and complete the wizard. When the instance launches, the system copies SSM Agent to the instance and starts it. When the instance is online, you can configure it using Run Command. For more information, see Executing Commands Using Systems Manager Run Command (p. 174).

### Note

You can automatically update SSM Agent on your instances when new versions become available by using Systems Manager State Manager. For more information, see Walkthrough: Automatically Update the SSM Agent (p. 254).

## Manually Install SSM Agent on Amazon EC2 Linux Instances

Use one of the following scripts to install SSM Agent on one of the following Linux instances.

- Amazon Linux (p. 19)
- Ubuntu (p. 20)
- Red Hat Enterprise Linux (p. 21)
- CentOS (p. 22)
- SLES (p. 23)
- Raspbian (p. 24)

### Note

The URLs in the following scripts let you download the SSM Agent from any AWS region. If you want to download the agent from a specific region, copy the URL for your operating system, and then replace `region` with a region where SSM is available.
For example, to download the SSM Agent for Amazon Linux, RHEL, CentOS, and SLES 64-bit from the US West 1 Region, use the following URL:

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

If the download fails, try replacing https://s3-region with https://s3.region.

- Amazon Linux, RHEL, CentOS, and SLES 64-bit:
  https://s3-region.amazonaws.com/amazon-ssm-region/latest/linux_amd64/amazon-ssm-agent.rpm

- Amazon Linux, RHEL, and CentOS 32-bit:

- Ubuntu Server 64-bit:

- Ubuntu Server 32-bit:

After you manually install SSM Agent, you can automatically update SSM Agent on your instances when new versions become available by using Systems Manager State Manager. For more information, see Walkthrough: Automatically Update the SSM Agent (p. 254).

**Amazon Linux**

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

*Note*

SSM Agent is installed, by default, on Amazon Linux AMIs dated 2017.09 and later.

**To install SSM Agent on Amazon Linux**

1. Create a temporary directory on the instance.

   mkdir /tmp/ssm

2. Change to the temporary directory.

   cd /tmp/ssm

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

   64-Bit

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

   32-Bit

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

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

5. Execute the following commands if the previous command returns, "amazon-ssm-agent is stopped."
   
a. Start the service.
   ```bash
   sudo start amazon-ssm-agent
   ```
   
b. Check the status of the agent.
   ```bash
   sudo status amazon-ssm-agent
   ```

**Ubuntu**

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

**To install SSM Agent on Ubuntu**

1. Create a temporary directory on the instance.

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

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

   **64-Bit**

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

   **32-Bit**

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

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

   **Ubuntu Server 14**

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

   **Ubuntu Server 16**

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

4. Execute the following commands if the previous command returned "amazon-ssm-agent is stopped," "inactive," or "disabled."
Installing and Configuring SSM Agent on Linux Instances

a. Start the service.

Ubuntu Server 14

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

Ubuntu Server 16

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

sudo systemctl start amazon-ssm-agent
```

b. Check the status of the agent.

Ubuntu Server 14

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

Ubuntu Server 16

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

Red Hat Enterprise Linux

Connect to your Red Hat Enterprise Linux (RHEL) instance and perform the following steps to install SSM Agent. Perform these steps on each instance that will execute commands using Systems Manager.

To install SSM Agent on Red Hat Enterprise Linux

1. Create a temporary directory on the instance.

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

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

   **64-Bit**

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

   **32-Bit**

   ```bash
   ```

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

   RHEL 7.x

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

   RHEL 6.x

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

4. Execute the following commands if the previous command returned "amazon-ssm-agent is stopped."
   a. Start the service.
      RHEL 7.x
      ```
      sudo systemctl enable amazon-ssm-agent
      sudo systemctl start amazon-ssm-agent
      ```
      RHEL 6.x
      ```
      sudo start amazon-ssm-agent
      ```
   b. Check the status of the agent.
      RHEL 7.x
      ```
      sudo systemctl status amazon-ssm-agent
      ```
      RHEL 6.x
      ```
      sudo status amazon-ssm-agent
      ```

CentOS

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

To install SSM Agent on CentOS

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

2. Use one of the following commands to download and run the SSM installer.
   64-Bit
   ```
   sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
   ```
   32-Bit
   ```
   ```

3. Run one of the following commands to determine if SSM Agent is running. The command should return "amazon-ssm-agent is running."
   CentOS 7.x
   ```
   ```
sudo systemctl status amazon-ssm-agent

CentOS 6.x

sudo status amazon-ssm-agent

4. Execute the following commands if the previous command returned "amazon-ssm-agent is stopped."
   a. Start the service.
      CentOS 7.x
      ```
      sudo systemctl enable amazon-ssm-agent
      sudo systemctl start amazon-ssm-agent
      ```
      CentOS 6.x
      ```
      sudo start amazon-ssm-agent
      ```
   b. Check the status of the agent.
      CentOS 7.x
      ```
      sudo systemctl status amazon-ssm-agent
      ```
      CentOS 6.x
      ```
      sudo status amazon-ssm-agent
      ```

SLES

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

To install SSM Agent on SLES

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

2. Change to the temporary directory.
   ```
   cd /tmp/ssm
   ```

3. Use the following command to download and run the SSM installer.
   ```
   64-Bit
   ```
   ```
   wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
   sudo rpm --install amazon-ssm-agent.rpm
   ```
4. Run the following command to determine if SSM Agent is running. The command should return "amazon-ssm-agent is running."

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

5. Execute the following commands if the previous command returns, "amazon-ssm-agent is stopped."

   a. Start the service.

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

   b. Check the status of the agent.

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

**Raspbian**

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

**Before You Begin**

To set up your Raspbian devices as Systems Manager managed instances, you need to create a managed-instance activation. After you complete the activation, you receive an activation code and ID. This code/ID combination functions like an Amazon EC2 access ID and secret key to provide secure access to the Systems Manager service from your managed instances. Store the activation code and ID in a safe place. For more information about the activation process, see Setting Up AWS Systems Manager in Hybrid Environments (p. 27).

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

**To install SSM Agent on Raspbian devices**

1. Create a temporary directory on the instance.

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

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

   ```bash
   ```

3. Execute the following command to install SSM Agent.

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

4. Execute the following command to stop SSM Agent.

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

5. Execute the following command to register the agent using the managed-instance activation code and ID you received when you completed the managed-instance activation process.

   ```bash
   sudo amazon-ssm-agent -register -code "code" -id "ID" -region "Region"
   ```
6. Execute the following command to start SSM Agent.

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

**Note**
If you see the following error in the SSM Agent error logs, then the machine ID did not persist after a reboot:
Unable to load instance associations, unable to retrieve associations unable to retrieve associations error occurred in RequestManagedInstanceRoleToken: MachineFingerprintDoesNotMatch: Fingerprint does not match
Execute the following command to make the machine ID persist after a reboot.

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

**Configure SSM Agent to Use a Proxy**

You can configure SSM Agent to communicate through an HTTP proxy by adding the `http_proxy` and `no_proxy` settings to the SSM Agent configuration file. This section includes procedures for `upstart` and `systemd` environments.

**To configure SSM Agent to use a proxy (Upstart)**

1. Connect to the instance where you installed SSM Agent.
2. Open the `amazon-ssm-agent.conf` file in an editor such as VIM. By default, the file is located here:

   ```bash
   /etc/init/amazon-ssm-agent.conf
   ```

3. Add the following settings to the file in the following format:

   ```bash
   env http_proxy=http://hostname:port
   env https_proxy=http://hostname:port
   env HTTP_PROXY=http://hostname:port
   env HTTPS_PROXY=http://hostname:port
   ```

4. Add the `no_proxy` setting to the file in the following format. You must specify the IP address listed here. It is the instance metadata endpoint for SSM and without this IP address calls to SSM fail:

   ```bash
   env no_proxy=169.254.169.254
   ```

5. Save your changes and close the editor.
6. Stop and restart SSM Agent using the following commands:

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

The following Upstart example includes the `http_proxy` and `no_proxy` settings in the `amazon-ssm-agent.conf` file:

```bash
description "Amazon SSM Agent"
author "Amazon.com"
start on (runlevel [345] and started network)
```
To configure SSM Agent to use a proxy (systemd)

1. Connect to the instance where you installed SSM Agent.
2. Open the amazon-ssm-agent.service file in an editor such as VIM. By default, the file is located here:

   /etc/systemd/system/amazon-ssm-agent.service

3. Add the following settings to the file:

   ```
   Environment="http_proxy=http://hostname:port"
   Environment="https_proxy=http://hostname:port"
   Environment="HTTP_PROXY=http://hostname:port"
   Environment="HTTPS_PROXY=http://hostname:port"
   ```

4. Add the `no_proxy` setting to the file in the following format. You must specify the IP address listed here. It is the instance metadata endpoint for SSM and without this IP address calls to SSM fail:

   ```
   Environment="no_proxy=169.254.169.254"
   ```

5. Save your changes and close the editor.
6. Restart SSM Agent using the following commands:

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

The following systemd example includes the `http_proxy` and `no_proxy` settings in the `amazon-ssm-agent.service` file:

```
Type=simple
Environment="HTTP_PROXY=http://i-1234567890abcdef0:443"
Environment="no_proxy=169.254.169.254"
WorkingDirectory=/opt/amazon/ssm/
ExecStart=/usr/bin/amazon-ssm-agent
KillMode=process
Restart=on-failure
RestartSec=15min
```

View SSM Agent Logs

SSM Agent writes information about executions, scheduled actions, errors, and health statuses to log files on each instance. You can view log files by manually connecting to an instance, or you can automatically send logs to Amazon CloudWatch Logs. For more information about sending logs to CloudWatch, see Monitoring Instances with AWS Systems Manager (p. 359).

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

- `/var/log/amazon/ssm/amazon-ssm-agent.log`
You can enable extended logging by updating the seelog.xml file. By default, the configuration file is located here: /etc/amazon/ssm/seelog.xml.

For more information about cihub/seelog configuration, go to the cihub/seelog Wiki. For examples of cihub/seelog configurations, go to cihub/seelog examples.

### Uninstall SSM Agent from Linux Instances

Use the following commands to uninstall SSM Agent.

**Amazon Linux, RHEL, or Cent OS**

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

**Ubuntu**

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

**SLES**

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

### Setting Up AWS Systems Manager in Hybrid Environments

AWS Systems Manager lets you remotely and securely manage on-premises servers and virtual machines (VMs) in your hybrid environment. Configuring your hybrid environment for Systems Manager provides the following benefits.

- Create a consistent and secure way to remotely manage your on-premises workloads from one location using the same tools or scripts.
- Centralize access control for actions that can be performed on your servers and VMs by using AWS Identity and Access Management (IAM).
- Centralize auditing and your view into the actions performed on your servers and VMs because all actions are recorded in AWS CloudTrail.
- Centralize monitoring because you can configure CloudWatch Events and Amazon SNS to send notifications about service execution success.

Complete the procedures in this topic to configure your hybrid machines for Systems Manager.

**Important**

After you finish, your hybrid machines that are configured for Systems Manager are listed in the Amazon EC2 console and described as managed instances. Amazon EC2 instances configured for Systems Manager are also managed instances. In the Amazon EC2 console, however, your on-premise instances are distinguished from Amazon EC2 instances with the prefix "mi-".
Create an IAM Service Role

Servers and VMs in a hybrid environment require an IAM role to communicate with the Systems Manager SSM service. The role grants AssumeRole trust to the SSM service.

**Note**
You only need to create the service role once for each AWS account.

**To create an IAM service role using AWS Tools for Windows PowerShell**

1. Create a text file (in this example it is named `SSMService-Trust.json`) with the following trust policy. Save the file with the `.json` file extension.

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

2. Use `New-IAMRole` as follows to create a service role. This example creates a role named `SSMServiceRole`.

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

3. Use `Register-IAMRolePolicy` as follows to enable the `SSMServiceRole` to create a session token. The session token gives your managed instance permission to execute commands using Systems Manager.

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

**To create an IAM service role using the AWS CLI**

1. Create a text file (in this example it is named `SSMService-Trust.json`) with the following trust policy. Save the file with the `.json` file extension.

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

2. Use the `create-role` command to create the service role. This example creates a role named `SSMServiceRole`. 
Create a Managed-Instance Activation

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

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

**To create a managed-instance activation (AWS Systems Manager)**

2. In the left navigation, choose Activations, and then choose Create an Activation.
3. (Optional) In the Activation description field, enter a description for this activation. The description is optional, but we recommend that you enter a description if you plan to activate large numbers of servers and VMs.
4. In the Instance limit field, specify the total number of servers or VMs that you want to register with AWS.
5. In the IAM role name section, choose a service role option that enables your servers and VMs to communicate with AWS Systems Manager in the cloud:
   a. Choose Use the system created default command execution role to use a role and managed policy created by AWS.
   b. Choose Select an existing custom IAM role that has the required permissions to use the optional custom role you created earlier.
6. In the Activation expiry date field, specify an expiration date for the activation.
   **Note**
   If you want to register additional managed instances after the expiry date, you must create a new activation. The expiry date has no impact on registered and running instances.
7. (Optional) In the Default instance name field, specify a name.
8. Choose Create activation.
   **Important**
   Store the managed-instance activation code and ID in a safe place. You specify this code and ID when you install SSM Agent on servers and VMs in your hybrid environment. If you lose the code and ID, you must create a new activation.
To create a managed-instance activation using the console (Amazon EC2 Systems Manager)

1. Open the Amazon EC2 console, expand Systems Manager Shared Resources in the navigation pane, and choose Activations.
2. Choose Create an Activation.
3. Fill out the form and choose Create Activation.

   Note that you can specify a date when the activation expires. If you want to register additional managed instances after the expiry date, you must create a new activation. The expiry date has no impact on registered and running instances.
4. Store the managed-instance activation code and ID in a safe place. You specify this code and ID when you install SSM Agent on servers and VMs in your hybrid environment. If you lose the code and ID, you must create a new activation.

To create a managed-instance activation using the AWS Tools for Windows PowerShell

1. On a machine with where you have installed AWS Tools for Windows PowerShell, execute the following command in AWS Tools for Windows PowerShell.

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

   For example:

   ```powershell
   New-SSMActivation -DefaultInstanceName MyWebServers -IamRole RunCommandServiceRole -RegistrationLimit 10 -Region us-east-1
   ```

2. Press Enter. If the activation is successful, the system returns an activation code and an ID. Store the activation code and ID in a safe place.

To create a managed-instance activation using the AWS CLI

1. On a machine where you have installed the AWS Command Line Interface (AWS CLI), execute the following command in the CLI.

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

   For example:

   ```bash
   aws ssm create-activation --default-instance-name MyWebServers --iam-role RunCommandServiceRole --registration-limit 10 --region us-east-1
   ```

2. Press Enter. If the activation is successful, the system returns an activation code and an ID. Store the activation code and ID in a safe place.

Install the SSM Agent on Servers and VMs in Your Windows Hybrid Environment

Before you begin, locate the activation code and ID that was sent to you after you completed the managed-instance activation in the previous section. You will specify the code and ID in the following procedure.
Important
This procedure is for servers and VMs in an on-premises or hybrid environment. To download and install the SSM Agent on an Amazon EC2 Windows instance, see Installing and Configuring SSM Agent on Windows Instances (p. 13).

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

1. Log on to a server or VM in your hybrid environment.
2. Open Windows PowerShell.
3. Copy and paste the following command block into AWS Tools for Windows PowerShell. Specify your activation code, activation ID, and the region where you want to download the SSM Agent from. For region, choose a region where SSM is available. For example, us-west-2.

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

The command downloads and installs the SSM Agent onto the server or VM. The command also registers the server or VM with the SSM service. The server or VM is now a managed instance. In the console, these instances are listed with the prefix "mi-". You can view all instances using a List command. For more information, see the Amazon EC2 Systems Manager API Reference.

Install the SSM Agent on Servers and VMs in Your Linux Hybrid Environment

Before you begin, locate the activation code and ID that was sent to you after you completed the managed-instance activation. You will specify the code and ID in the following procedure.

Important
This procedure is for servers and VMs in an on-premises or hybrid environment. To download and install the SSM Agent on an Amazon EC2 Linux instance, see Installing and Configuring SSM Agent on Linux Instances (p. 15).

The URLs in the following scripts let you download the SSM Agent from any AWS region. If you want to download the agent from a specific region, copy the URL for your operating system, and then replace region with a region where SSM is available.

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

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

If the download fails, try replacing https://s3-region with https://s3.region.
Install the SSM Agent on Servers and VMs in Your Linux Hybrid Environment

- Amazon Linux, RHEL, CentOS, and SLES 64-bit

- Amazon Linux, RHEL, and CentOS 32-bit

- Ubuntu Server 64-bit

- Ubuntu Server 32-bit

- Raspbian

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

1. Log on to a server or VM in your hybrid environment.
2. Copy and paste one of the following command blocks into SSH. Specify your activation code, activation ID, and the region where you want to download the SSM Agent from. Note that sudo is not necessary if you are a root user.

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

```
mkdir /tmp/ssm
sudo yum install -y /tmp/ssm/amazon-ssm-agent.rpm
```

On RHEL 7.x and CentOS 7.x

```
mkdir /tmp/ssm
```

On SLES

```
```

On Ubuntu

```
```
### Working with Cron and Rate Expressions for Systems Manager

When you create a AWS Systems Manager Maintenance Window or a State Manager association, you specify a schedule for when the window or the association should run. You can specify a schedule in the form of either a time-based entry, called a **cron expression**, or a frequency-based entry, called a **rate expression**.

If you create a Maintenance Window or an association by using the Amazon EC2 console, then you can use tools in the user interface to create your schedule. If you want to create the Maintenance Window or the association programmatically or from the command line by using, for example, the AWS CLI, then you must specify a schedule parameter with a cron or rate expression in the correct format.
Note
To create a Maintenance Windows from the AWS CLI, you use the `--schedule` parameter with a cron or rate expression. To create a State Manager associations from the AWS CLI, you use the `--scheduleExpression` parameter with a cron or rate expression.

Here are some examples that show the `schedule` parameter with cron and rate expressions:

**Cron example:** This cron expression runs the association at 4 PM (16:00) every Tuesday.

```
--scheduleExpression "cron(0 16 ? * TUE *)"
```

**Rate example:** This rate expression runs the Maintenance Window or the association every other day.

```
--schedule "rate(2 days)"
```

Important
State Manager associations have limited options for cron and rate expressions compared to Maintenance Windows. Before you create one of these expressions for an association, review the restrictions in the following section.

If you are not familiar with cron and rate expressions, we suggest that you read General Information About Cron and Rate Expressions (p. 35).

### Cron and Rate Expressions for Associations

This section includes examples of cron and rate expressions for State Manager associations. Before you create one of these expressions, be aware of the following restrictions.

- Associations only support the following cron expressions: every 1/2, 1, 2, 4, 8, or 12 hours; every day or every week at a specific time.
- Associations only support the following rate expressions: intervals of 30 minutes or greater and less than 31 days.

Here are some cron examples for associations.

#### Cron Examples for Associations

<table>
<thead>
<tr>
<th>Example</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>cron(0/30 * * * ? *)</td>
<td>Every 30 minutes</td>
</tr>
<tr>
<td>cron(0 0/1 * * ? *)</td>
<td>Every hour</td>
</tr>
<tr>
<td>cron(0 0/2 * * ? *)</td>
<td>Every 2 hours</td>
</tr>
<tr>
<td>cron(0 0/4 * * ? *)</td>
<td>Every 4 hours</td>
</tr>
<tr>
<td>cron(0 0/8 * * ? *)</td>
<td>Every 8 hours</td>
</tr>
<tr>
<td>cron(0 0/12 * * ? *)</td>
<td>Every 12 hours</td>
</tr>
<tr>
<td>cron(15 13 * * ? *)</td>
<td>Every day at 1:15 PM</td>
</tr>
<tr>
<td>cron(15 13 ? * MON *)</td>
<td>Every Monday at 1:15 PM</td>
</tr>
</tbody>
</table>

Here are some rate examples for associations.
## Cron and Rate Expressions for Maintenance Windows

Unlike State Manager associations, Maintenance Windows support all cron and rate expressions. Here are some cron examples for Maintenance Windows.

### Cron Examples for Maintenance Windows

<table>
<thead>
<tr>
<th>Example</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 2 ? 1/1 THU#3 *</td>
<td>02:00 AM the third Thursday of every month</td>
</tr>
<tr>
<td>15 10 ? * * *</td>
<td>10:15 AM every day</td>
</tr>
<tr>
<td>0 15 10 ? * MON-FRI</td>
<td>10:15 AM every Monday, Tuesday, Wednesday, Thursday and Friday</td>
</tr>
<tr>
<td>0 0 2 L * ?</td>
<td>02:00 AM on the last day of every month</td>
</tr>
<tr>
<td>0 15 10 ? * 6L</td>
<td>10:15 AM on the last Friday of every month</td>
</tr>
</tbody>
</table>

Here are some rate examples for Maintenance Windows.

### Rate Examples for Maintenance Windows

<table>
<thead>
<tr>
<th>Example</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate(30 minutes)</td>
<td>Every 30 minutes</td>
</tr>
<tr>
<td>rate(1 hour)</td>
<td>Every hour</td>
</tr>
<tr>
<td>rate(5 hours)</td>
<td>Every 5 hours</td>
</tr>
<tr>
<td>rate(15 days)</td>
<td>Every 15 days</td>
</tr>
<tr>
<td>rate(25 days)</td>
<td>Every 25 days</td>
</tr>
</tbody>
</table>

## General Information About Cron and Rate Expressions

Cron expressions for Systems Manager have six required fields. Fields are separated by a space.
### General Information About Cron and Rate Expressions

<table>
<thead>
<tr>
<th>Minutes</th>
<th>Hours</th>
<th>Day of month</th>
<th>Month</th>
<th>Day of week</th>
<th>Year</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run at 10:00 am (UTC) every day</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run at 12:15 PM (UTC) every day</td>
</tr>
<tr>
<td>0</td>
<td>18</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run at 6:00 PM (UTC) every Monday through Friday</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>1</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run at 8:00 AM (UTC) every 1st day of the month</td>
</tr>
<tr>
<td>0/15</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run every 15 minutes</td>
</tr>
<tr>
<td>0/10</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run every 10 minutes Monday through Friday</td>
</tr>
<tr>
<td>0/5</td>
<td>8-17</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run every 5 minutes Monday through Friday between 8:00 AM and 5:55 PM (UTC)</td>
</tr>
</tbody>
</table>

The following table shows supported values for required cron entries:

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Wildcards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minutes</td>
<td>0-59</td>
<td>, - * /</td>
</tr>
<tr>
<td>Hours</td>
<td>0-23</td>
<td>, - * /</td>
</tr>
<tr>
<td>Day-of-month</td>
<td>1-31</td>
<td>, - * ? / L W</td>
</tr>
<tr>
<td>Month</td>
<td>1-12 or JAN-DEC</td>
<td>, - * /</td>
</tr>
<tr>
<td>Day-of-week</td>
<td>1-7 or SUN-SAT</td>
<td>, - * ? / L</td>
</tr>
</tbody>
</table>
### General Information About Cron and Rate Expressions

#### Field | Values | Wildcards
---|---|---
Year | 1970-2199 | , - */

**Note**
You cannot 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, you must use a ? (question mark) in the other field.

#### Wildcards
Cron expressions support the following wildcards:

- The , (comma) wildcard includes additional values. In the Month field, JAN,FEB,MAR would include January, February, and March.
- The - (dash) wildcard specifies ranges. In the Day field, 1-15 would include days 1 through 15 of the specified month.
- The * (asterisk) wildcard includes all values in the field. In the Hours field, * would include every hour.
- The / (forward slash) wildcard specifies increments. In the Minutes field, you could enter 1/10 to specify every tenth minute, starting from the first minute of the hour (for example, the 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.
- The L wildcard in the Day-of-month or Day-of-week fields specifies the last day of the month or week.
- 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.

**Note**
Cron expressions that lead to rates faster than 5 minute are not supported. Support for specifying both a day-of-week and a day-of-month value is not complete. You must currently use the '?' character in one of these fields.

For more information about cron expressions, see [CRON expression at the Wikipedia website](https://en.wikipedia.org/wiki/Cron).

#### Rate Expressions
Rate expressions have the following two required fields. Fields are separated by white space.

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>positive number</td>
</tr>
<tr>
<td>Unit</td>
<td>minute(s) OR hour(s) OR day(s)</td>
</tr>
</tbody>
</table>

**Note**
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) are not valid, but rate(1 hour) and rate(5 hours) are valid.
Partner and Product Integration

You can use AWS Systems Manager with partner and product technologies such as GitHub, Amazon S3, and the Volume Shadow Copy Service (VSS) to automate the deployment, configuration, and maintenance of your managed instances.

Contents
- Running Scripts from GitHub and Amazon S3 (p. 38)
- Using Run Command to Take VSS-Enabled Snapshots of EBS Volumes (p. 50)

Running Scripts from GitHub and Amazon S3

This section describes how to use the AWS-RunRemoteScript pre-defined SSM document to download scripts from GitHub and Amazon S3, including Ansible Playbooks, Python, Ruby, and PowerShell scripts. By using this document, you no longer need to manually port scripts into Amazon EC2 or wrap them in SSM documents. Systems Manager integration with GitHub and Amazon S3 promotes infrastructure as code, which reduces the time it takes to manage instances while standardizing configurations across your fleet.

You can also create custom SSM documents that enable you to download and run scripts or other SSM documents from remote locations. For more information, see Creating Composite Documents (p. 279).

Topics
- Running Scripts from GitHub (p. 38)
- Running Scripts from Amazon S3 (p. 44)

Running Scripts from GitHub

This section describes how to download and run scripts from a private or public GitHub repository. You can run different types of scripts, including Ansible Playbooks, Python, Ruby, and PowerShell scripts.

You can also download a directory that includes multiple scripts. When you run the primary script in the directory, Systems Manager also runs any referenced scripts (as long as the referenced scripts are included in the directory).

Note the following important details about running scripts from GitHub.

- Systems Manager does not check to see if your script is capable of running on an instance. Before you download and run the script, you must verify that the required software is installed on the instance. Or, you can create a composite document that installs the software by using either Run Command or State Manager, and then downloads and runs the script.
- You are responsible for ensuring that all GitHub requirements are met. This includes refreshing your access token, as needed. You must also ensure that you don't surpass the number of authenticated or unauthenticated requests. For more information, see the GitHub documentation.

Topics
- Run Ansible Playbooks from GitHub (p. 39)
Run Ansible Playbooks from GitHub

This section includes procedures to help you run Ansible Playbooks from GitHub by using either the console or the AWS CLI.

Before You Begin

If you plan to run a script that is stored in a private GitHub repository, then you must create a Systems Manager SecureString parameter for your GitHub security access token. You can’t access a script in a private GitHub repository by manually passing your token over SSH. The access token must be passed as a Systems Manager SecureString parameter. For more information about creating a SecureString parameter, see Creating Systems Manager Parameters (p. 337).

Run an Ansible Playbook from GitHub (Console)

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

Run an Ansible Playbook from GitHub (AWS Systems Manager)

2. In the navigation pane, choose Run Command.
   -or-
   If the AWS Systems Manager home page opens, scroll down and choose Explore Run Command.
3. Choose Run command.
5. In the Targets section, identify the instances to which you want to download and run the script by specifying tags or selecting instances manually.
6. In Command parameters, do the following:
   - In Source Type, select GitHub.
   - In the Source Info box, type the required information to access the source in the following format:

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

     For example:

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

     This example downloads a directory of scripts named complex-script.
   - In the Command Line field, type parameters for the script execution. Here is an example.

     ```bash
     ansible-playbook -i "localhost," --check --c local webserver.yml
     ```
   - (Optional) In the Working Directory field, type the name of a directory on the instance where you want to download and run the script.
   - (Optional) In Execution Timeout, specify the number of seconds for the system to wait before failing the script command execution.
7. In **Other parameters**:
   - In the **Comment** box, type information about this command.
   - In **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. (Optional) In **Rate control**:
   - In **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.
   - **Note**
     If you selected targets by choosing Amazon EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document at the same time by specifying a percentage.
   - In **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify 3 errors, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.

9. In the **Output options** section, if you want to save the command output to a file, select the **Write command output to an Amazon S3 bucket**. Type the bucket and prefix names in the boxes.

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 Configuring Amazon SNS Notifications for Run Command (p. 169).

11. Choose **Run**.

Run an Ansible Playbook from GitHub (Amazon EC2 Systems Manager)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Run Command**, and then choose **Run a command**.
3. In the **Document** list, choose **AWS-RunRemoteScript**.
4. In the **Select Targets by** section, choose an option and select the instances where you want to download and run the script.
5. (Optional) In the **Execute on** field, specify a number of **Targets** that can run the AWS-RunRemoteScript document concurrently (for example, 10). Or, specify a percentage of the number of targets that can run the document concurrently (for example, 10%).
   - **Note**
     If you selected targets by choosing EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document by specifying a percentage.
6. (Optional) In the **Stop after** field, specify the maximum number of errors allowed before the system stops sending the command to other instances. For example, if you specify 3, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.
7. In the **Source Type** list, choose **GitHub**
8. In the **Source** text box, type the required information to access the source in the following format:

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

   For example:
9. In the **Command Line** field, type parameters for the script execution. Here is an example.

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

10. In the **Working Directory** field, type the name of a directory on the instance where you want to download and run the script.

11. In the **Comments** field, type information about this command.

12. In the **Advanced Options** section, choose **Write to S3** to store command output in an Amazon S3 bucket. Type the bucket and prefix names in the text boxes.

13. Choose **Enable SNS notifications** to receive notifications and status about the command execution. For more information about configuring SNS notifications for Run Command, see Configuring Amazon SNS Notifications for Run Command (p. 169).

14. Choose **Run**.

### Run an Ansible Playbook from GitHub by Using the AWS CLI

1. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2 or you must have been granted the appropriate permission in IAM. For more information, see **Systems Manager Prerequisites (p. 4)**.

    ```bash
    aws configure
    ```

    The system prompts you to specify the following.

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

2. Execute the following command to download and run a script from GitHub.

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

    Here is an example.

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

### Run Python Scripts from GitHub

This section includes procedures to help you run Python scripts from GitHub by using either the console or the AWS CLI.
Run a Python Script from GitHub (Console)

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

**Run a Python Script from GitHub (AWS Systems Manager)**

2. In the navigation pane, choose **Run Command**.
   - or-
   If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.
3. Choose **Run command**.
4. In the **Command document** list, choose **AWS-RunRemoteScript**.
5. In the **Targets** section, identify the instances to which you want to download and run the script by specifying tags or selecting instances manually.
6. In **Command parameters**, do the following:
   - In **Source Type**, select **GitHub**.
   - In the **Source Info** text box, type the required information to access the source in the following format:

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

   For example:

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

   This example downloads a directory of scripts named **complex-script**.
   - In the **Command Line** field, type parameters for the script execution. Here is an example.

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

   This example runs **mainFile.py**, which can then run other scripts in the **complex-script** directory.
   - (Optional) In the **Working Directory** field, type the name of a directory on the instance where you want to download and run the script.
   - (Optional) In **Execution Timeout**, specify the number of seconds for the system to wait before failing the script command execution.
7. In **Other parameters**:
   - In the **Comment** box, type information about this command.
   - In **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.
8. (Optional) In **Rate control**:
   - In **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.
Note
If you selected targets by choosing Amazon EC2 tags, and you are not certain how many
instances use the selected tags, then limit the number of instances that can run the
document at the same time by specifying a percentage.

• In Error threshold, specify when to stop running the command on other instances after it fails on
either a number or a percentage of instances. For example, if you specify 3 errors, then Systems
Manager stops sending the command when the 4th error is received. Instances still processing the
command might also send errors.

9. In the Output options section, if you want to save the command output to a file, select the Write
command output to an Amazon S3 bucket. Type the bucket and prefix names in the boxes.

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 Configuring Amazon SNS Notifications for Run
Command (p. 169).


Run a Python Script from GitHub (Amazon EC2 Systems Manager)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Run Command, and then choose Run a command.
4. In the Select Targets by section, choose an option and select the instances where you want to
download and run the script.
5. (Optional) In the Execute on field, specify a number of Targets that can run the AWS-
RunRemoteScript document concurrently (for example, 10). Or, specify a percentage of the number
of targets that can run the document concurrently (for example, 10%).

Note
If you selected targets by choosing EC2 tags, and you are not certain how many instances
use the selected tags, then limit the number of instances that can run the document by
specifying a percentage.

6. (Optional) In the Stop after field, specify the maximum number of errors allowed before the system
stops sending the command to other instances. For example, if you specify 3, then Systems Manager
stops sending the command when the 4th error is received. Instances still processing the command
might also send errors.

7. In the Source Type list, choose GitHub
8. In the Source text box, type the required information to access the source in the following format:

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

For example:

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

This example downloads a directory of scripts named complex-script.

9. In the Command Line field, type 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.

10. In the **Working Directory** field, type the name of a directory on the instance where you want to download and run the script.

11. In the **Comments** field, type information about this command.

12. In the **Advanced Options** section, choose **Write to S3** to store command output in an Amazon S3 bucket. Type the bucket and prefix names in the text boxes.

13. Choose **Enable SNS notifications** to receive notifications and status about the command execution. For more information about configuring SNS notifications for Run Command, see Configuring Amazon SNS Notifications for Run Command (p. 169).

14. Choose **Run**.

### Run a Python Script from GitHub by Using the AWS CLI

1. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2 or you must have been granted the appropriate permission in IAM. For more information, see *Systems Manager Prerequisites (p. 4).*

   ```bash
   aws configure
   ```

   The system prompts you to specify the following.

<table>
<thead>
<tr>
<th>AWS Access Key ID [None]: key_name</th>
<th>AWS Secret Access Key [None]: key_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default region name [None]: region</td>
<td>Default output format [None]: ENTER</td>
</tr>
</tbody>
</table>

2. Execute the following command to download and run a script from GitHub.

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

   Here is an example.

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

   This example downloads a directory of scripts name complex-script. The **commandLine** entry runs mainFile.py, which can then run other scripts in the complex-script directory.

### Running Scripts from Amazon S3

This section describes how to download and run scripts from Amazon S3. You can run different types of scripts, including Ansible Playbooks, Python, Ruby, and PowerShell.

You can also download a directory that includes multiple scripts. When you run the primary script in the directory, Systems Manager also runs any referenced scripts (as long as the referenced scripts are included in the directory).

Note the following important details about running scripts from Amazon S3.
• Systems Manager does not check to see if your script is capable of running on an instance. Before you download and run the script, you must verify that the required software is installed on the instance. Or, you can create a composite document that installs the software by using either Run Command or State Manager, and then downloads and runs the script.
• Verify that your AWS Identity and Access Management (IAM) user account, role, or group has permission to read from the S3 bucket.

Topics
• Run Ruby Scripts from Amazon S3 (p. 45)
• Run PowerShell Script from Amazon S3 (p. 47)

Run Ruby Scripts from Amazon S3

This section includes procedures to help you run Ruby scripts from Amazon S3 by using either the EC2 console or the AWS CLI.

Run a Ruby Script from Amazon S3 (Console)

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

Run a Ruby Script from Amazon S3 (AWS Systems Manager)

2. In the navigation pane, choose Run Command.
   -or-
   If the AWS Systems Manager home page opens, scroll down and choose Explore Run Command.
3. Choose Run command.
5. In the Targets section, identify the instances to which you want to download and run the script by specifying tags or selecting instances manually.
6. In Command parameters, do the following:
   • In Source Type, select S3.
   • In the Source Info text box, type the required information to access the source in the following format:

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

```
{"path":"https://s3.amazonaws.com/rubytest/scripts/ruby/helloWorld.rb"}
```
   • In the Command Line field, type parameters for the script execution. Here is an example.

```
helloWorld.rb argument-1 argument-2
```
   • (Optional) In the Working Directory field, type the name of a directory on the instance where you want to download and run the script.
   • (Optional) In Execution Timeout, specify the number of seconds for the system to wait before failing the script command execution.
7. In **Other parameters**:
   - In the **Comment** box, type information about this command.
   - In **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. (Optional) In **Rate control**:
   - In **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.
     
     **Note**
     
     If you selected targets by choosing Amazon EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document at the same time by specifying a percentage.

   - In **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify 3 errors, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.

9. In the **Output options** section, if you want to save the command output to a file, select the **Write command output to an Amazon S3 bucket**. Type the bucket and prefix names in the boxes.

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 Configuring Amazon SNS Notifications for Run Command (p. 169).

11. Choose **Run**.

---

**Run a Ruby Script from Amazon S3 (Amazon EC2 Systems Manager)**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Run Command**, and then choose **Run a command**.
3. In the **Document** list, choose **AWS-RunRemoteScript**.
4. In the **Select Targets by** section, choose an option and select the instances where you want to download and run the script.
5. (Optional) In the **Execute on** field, specify a number of **Targets** that can run the AWS-RunRemoteScript document concurrently (for example, 10). Or, specify a percentage of the number of targets that can run the document concurrently (for example, 10%).

   **Note**
   
   If you selected targets by choosing EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document by specifying a percentage.

6. (Optional) In the **Stop after** field, specify the maximum number of errors allowed before the system stops sending the command to other instances. For example, if you specify 3, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.

7. In the **Source Type** list, choose **S3**
8. In the **Source** text box, type the required information to access the source in the following format:

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

   ```json
   {"path":"https://s3.amazonaws.com/rubytest/scripts/ruby/helloWorld.rb"}
   ```
9. In the **Command Line** field, type parameters for the script execution. Here is an example.

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

10. In the **Working Directory** field, type the name of a directory on the instance where you want to download and run the script.

11. In the **Comments** field, type information about this command.

12. In the **Advanced Options** section, choose **Write to S3** to store command output in an Amazon S3 bucket. Type the bucket and prefix names in the text boxes.

13. Choose **Enable SNS notifications** to receive notifications and status about the command execution. For more information about configuring SNS notifications for Run Command, see **Configuring Amazon SNS Notifications for Run Command** (p. 169).

14. Choose **Run**.

### Run a Ruby Script from S3 by using the AWS CLI

1. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2 or you must have been granted the appropriate permission in IAM. For more information, see **Systems Manager Prerequisites** (p. 4).

```
aws configure
```

The system prompts you to specify the following.

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

2. Execute the following command to download and run a script from Amazon S3.

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

Here is an example.

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

### Run a PowerShell Script from Amazon S3

This section includes procedures to help you run PowerShell scripts from Amazon S3 by using either the EC2 console or the AWS CLI.

### Run a PowerShell Script from Amazon S3 (Console)

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:
Run a PowerShell Script from Amazon S3 (AWS Systems Manager)

   -or-
   If the AWS Systems Manager home page opens, scroll down and choose Explore Run Command.
2. Choose Run command.
3. In the Command document list, choose AWS-RunRemoteScript.
4. In the Targets section, identify the instances to which you want to download and run the script by
   specifying tags or selecting instances manually.
5. In Command parameters, do the following:
   - In Source Type, select S3.
   - In the Source Info text box, type the required information to access the source in the following
     format:
     
     ```json
     {"path": "https://s3.amazonaws.com/path_to_script"}
     ```
     For example:
     
     ```json
     {"path": "https://s3.amazonaws.com/PowerShellTest/powershell/helloPowershell.ps1"}
     ```
   - In the Command Line field, type parameters for the script execution. Here is an example.
     ```bash
     helloPowershell.ps1 argument-1 argument-2
     ```
   - (Optional) In the Working Directory field, type the name of a directory on the instance where you
     want to download and run the script.
   - (Optional) In Execution Timeout, specify the number of seconds for the system to wait before
     failing the script command execution.
6. In Other parameters:
   - In the Comment box, type information about this command.
   - In Timeout (seconds), specify the number of seconds for the system to wait before failing the
     overall command execution.
7. In Output options:
   - In the Write command output to an Amazon S3 bucket section, type the bucket and prefix names in the boxes.
8. In Rate control:
   - In Concurrency, specify either a number or a percentage of instances on which to run the
     command at the same time.
     
     **Note**
     If you selected targets by choosing Amazon EC2 tags, and you are not certain how many
     instances use the selected tags, then limit the number of instances that can run the
     document at the same time by specifying a percentage.
   - In Error threshold, specify when to stop running the command on other instances after it fails on
     either a number or a percentage of instances. For example, if you specify 3 errors, then Systems
     Manager stops sending the command when the 4th error is received. Instances still processing the
     command might also send errors.
9. In the Output options section, if you want to save the command output to a file, select the Write
   command output to an Amazon S3 bucket. Type the bucket and prefix names in the boxes.
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 Configuring Amazon SNS Notifications for Run Command (p. 169).


Run a PowerShell Script from Amazon S3 (Amazon EC2 Systems Manager)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Run Command, and then choose Run a command.
4. In the Select Targets by section, choose an option and select the instances where you want to download and run the script.
5. (Optional) In the Execute on field, specify a number of Targets that can run the AWS-RunRemoteScript document concurrently (for example, 10). Or, specify a percentage of the number of targets that can run the document concurrently (for example, 10%).
   
   Note
   If you selected targets by choosing EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document by specifying a percentage.

6. (Optional) In the Stop after field, specify the maximum number of errors allowed before the system stops sending the command to other instances. For example, if you specify 3, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.
7. In the Source Type list, choose S3
8. In the Source text box, type the required information to access the source in the following format:

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

   ```json
   {"path": "https://s3.amazonaws.com/PowerShellTest/powershell/helloPowershell.ps1"}
   ```
9. In the Command Line field, type parameters for the script execution. Here is an example.

   ```bash
   helloPowershell.ps1 argument-1 argument-2
   ```
10. In the Working Directory field, type the name of a directory on the instance where you want to download and run the script.
11. In the Comments field, type information about this command.
12. In the Advanced Options section, choose Write to S3 to store command output in an Amazon S3 bucket. Type the bucket and prefix names in the text boxes.
13. Choose Enable SNS notifications to receive notifications and status about the command execution. For more information about configuring SNS notifications for Run Command, see Configuring Amazon SNS Notifications for Run Command (p. 169).

Run a PowerShell Script from S3 by Using the AWS CLI

1. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2 or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).
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.

There is no additional cost to use VSS-enabled EBS snapshots. You only pay for EBS snapshots created by the backup process. For more information, see How is my EBS snapshot bill calculated?

How It Works

Here is how the process of taking application-consistent, VSS-enabled EBS snapshots works.

1. You verify and configure Systems Manager prerequisites.
2. You enter parameters for the AWSEC2-CreateVssSnapshots SSM document and execute this document by using Run Command. You can't create a VSS-enabled EBS snapshot for a specific volume. You can, however, specify a parameter to exclude the boot volume from the backup process.
3. The VSS agent on your instance coordinates all ongoing I/O operations for running applications.
4. The system flushes all I/O buffers and temporarily pauses all I/O operations. The pause lasts, at most, ten seconds.
5. During the pause, the system creates snapshots of all volumes attached to the instance.
6. The pause is lifted and I/O resumes operation.
7. The system adds all newly-created snapshots to the list of EBS snapshots. The system tags all VSS-enabled EBS snapshots successfully created by this process with AppConsistent:true. This tag

aws configure

The system prompts you to specify the following.

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

2. Execute the following command to download and run a script from Amazon S3.

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

Here is an example.

aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids "i-1234abcd" --parameters '{"sourceType": ["S3"], "sourceInfo": {"path": "https://s3.amazonaws.com/TestPowershell/powershell/helloPowershell.ps1"}, "commandLine": ["helloPowershell.ps1 argument-1 argument-2"]}'
helps you identify snapshots created by this process, as opposed to other processes. If the system encounters an error, the snapshot created by this process does not include the `AppConsistent:true` tag.

8. In the event that you need to restore from a snapshot, you can restore by using the standard EBS process of creating a volume from a snapshot, or you can restore all volumes to an instance by using a sample script, which is described later in this section.

**Before You Begin**

Before you create VSS-enabled EBS snapshots by using Run Command, review the following requirements and limitations, and complete the required tasks.

- VSS-enabled EBS snapshots are supported for instances running Windows Server 2008 R2 or later. (Windows Server 2008 R2 Core is currently not supported.) Verify that your instances meet all requirements for Amazon EC2 Windows. For more information, see Setting Up AWS Systems Manager (p. 4).
- Update your instances to use SSM Agent version 2.2.58.0 or later. If you are using an older version of SSM Agent, you can update it by using Run Command. For more information, see Example: Update the SSM Agent (p. 176).
- Systems Manager requires permission to perform actions on your instances. You must configure each instance with an AWS Identity and Access Management (IAM) instance profile role for Systems Manager. For more information, see Configuring Access to Systems Manager (p. 8).
- Systems Manager needs permissions to create and tag VSS-enabled EBS snapshots. You can configure an IAM role that enables these permissions. You must configure each instance with a role for creating and tagging snapshots. For more information, see Create an IAM Role for VSS-Enabled Snapshots (p. 51) in Set Up Tasks.

  **Note**

  If you don't want to assign the snapshot role to your instances, you can use Run Command and the pre-defined AWSEC2-ManageVssIO SSM document to temporarily pause I/O, create VSS-enabled EBS snapshots, and restart I/O. This process runs in the context of the user who executes the command. If the user has sufficient permission to create and tag snapshots, then Systems Manager can create and tag VSS-enabled EBS snapshots without the need for the additional IAM snapshot role on the instance. Instances still must be configured with the instance profile role. For more information, see Creating VSS-Enabled EBS Snapshots by Using the AWSEC2-ManageVssIO SSM Document (Advanced) (p. 57).

- Systems Manager requires VSS components to be installed on your instances. By default, the components are pre-installed on all instances created from Amazon EC2 Windows Amazon Machine Images (AMIs) dated 2017.11 or later. If you need to install the required VSS components, then you can download and run a VSS package for Systems Manager on your instances. If you plan to use your own Microsoft licenses for VSS (BYOL), you still need to install the VSS components for Systems Manager. For more information, see Download and Install VSS Components for Systems Manager (p. 52) in Set Up Tasks.

**Set Up Tasks**

- Create an IAM Role for VSS-Enabled Snapshots (p. 51)
- Download and Install VSS Components for Systems Manager (p. 52)

**Create an IAM Role for VSS-Enabled Snapshots**

This section includes a procedure for creating an IAM policy, and a separate procedure for creating an IAM role that uses the policy you create in the first procedure. The policy enables Systems Manager to
create snapshots, tags snapshots, and attach metadata like a device ID to the default snapshot tags the system creates.

To create an IAM policy for VSS-enable snapshots

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Policies, and then choose Create policy.
4. In the Select your use case section, choose EC2, and then choose Next: Permissions.
5. On the Attach permissions policy page, choose Create policy.
7. On the Edit Permissions page, verify that Effect is set to Allow.
8. Choose AWS Service, and then choose Amazon EC2.
9. In the Actions list, choose CreateSnapshot, CreateTags, and DescribeInstances.
10. In the Amazon Resource Name (ARN) field, type an asterisk (*).
11. Choose Add Statement, and then choose Next Step.
12. On the Review Policy page, type a name in the Policy Name field, and then choose Create Policy. The system returns you to the IAM console.

To create an IAM role for VSS-enabled EBS snapshots

1. In the navigation pane, choose Roles, and then choose Create role.
2. On the Select type of trusted entity page, under AWS Service, choose EC2.
3. In the Select your use case section, choose EC2, and then choose Next: Permissions.
4. On the Attach permissions policy page, choose the policy you created earlier in this procedure, and then choose Next: Review.
5. On the Review page, type a name in the Role name box, and then type a description.
6. Choose Create role. The system returns you to the Roles page.
7. Attach the role to the instances for which you want to create VSS-enabled EBS snapshots. For more information, see Attaching an IAM Role to an Instance in the Amazon EC2 User Guide.

Download and Install VSS Components for Systems Manager

Systems Manager requires VSS components to be installed on your instances. By default, the components are pre-installed on all instances created from Amazon EC2 Windows Amazon Machine Images (AMIs) dated 2017.11 or later. If you need to install the required VSS components, then you can use the following procedure to configure a State Manager association that automatically downloads and installs the components by using the AwsVssComponents package. The State Manager association will automatically download and install new versions of the package when they are published by AWS. The package installs two components: a VSS requestor and a VSS provider. The system copies these components to a directory on the instance, and then registers the provider DLL as a VSS provider.

If you don’t want Systems Manager to automatically download and install new versions of this package when they become available, you can use Run Command and the AWS-ConfigureAWSPackage SSM document to install the package on your instances, as described later in this section.

Note
The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.
To create a State Manager association that automatically downloads and installs the AwsVssComponents package

1. Open the Amazon EC2 console, expand Systems Manager Services in the navigation pane, and then choose State Manager.
2. Choose Create Association.
3. In the Association Name field, type a descriptive name.
4. In the Select Document list, choose AWS-ConfigureAWSPackage.
5. In the Select Targets by section, choose an option and select the instances where you want to download and run the script.
6. In the Schedule section, choose an option.
7. In the Parameters section, choose Install from the Action list.
8. From the Name field, type AwsVssComponents.
9. In the Version field, verify that latest is auto-populated.
10. In the Advanced section, choose Write to S3 if you want to write association details to an Amazon S3 bucket.
11. Choose Create Association, and then choose Close. The system attempts to create the association on the instances and immediately apply the state. The association status shows Pending.
12. In the right corner of the Association page, choose the refresh button. If you created the association on one or more EC2 Windows instances, the status changes to Success. If your instances are not properly configured for Systems Manager, the status shows Failed.
13. If the status is Failed, verify that the SSM Agent is running on the instance, and verify that the instance is configured with an IAM role for Systems Manager. For more information, see Systems Manager Prerequisites (p. 4).

Install the VSS Package by Using the AWS CLI

Use the following procedure to download and install the AwsVssComponents package on your instances by using Run Command from the AWS CLI. The package installs two components: a VSS requestor and a VSS provider. The system copies these components to a directory on the instance, and then registers the provider DLL as a VSS provider.

To install the VSS package by using the AWS CLI

1. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2 or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

```bash
aws configure
```

The system prompts you to specify the following:

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

2. Execute the following command to download and install the required VSS components for Systems Manager.

```bash
aws ssm send-command --document-name "AWS-ConfigureAWSPackage" --instance-ids "i-12345678" --parameters '{"action":["Install"],"name":["AwsVssComponents"]}'
```
Install the VSS Package by Using Tools for Windows PowerShell

Use the following procedure to download and install the AwsVssComponents package on your instances by using Run Command from the Tools for Windows PowerShell. The package installs two components: a VSS requestor and a VSS provider. The system copies these components to a directory on the instance, and then registers the provider DLL as a VSS provider.

**To install the VSS package by using AWS Tools for Windows PowerShell**

1. Open AWS Tools for Windows PowerShell and execute the following command to specify your credentials. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).
   ```powershell
   Set-AWSCredentials -AccessKey key_name -SecretKey key_name
   ```

2. Execute the following command to set the Region for your PowerShell session. The example uses the us-east-2 Region.
   ```powershell
   Set-DefaultAWSRegion -Region us-east-2
   ```

3. Execute the following command to download and install the required VSS components for Systems Manager.
   ```powershell
   Send-SSMCommand -DocumentName AWS-ConfigureAWSPackage -InstanceId "$instance" -Parameter @{'action'='Install';'name'='AwsVssComponents'}
   ```

Creating VSS-enabled EBS snapshots

This section includes procedures for creating VSS-enabled EBS snapshots by using the Amazon EC2 console, the AWS CLI, and Tools for Windows PowerShell.

**Topics**

- Create VSS-enabled EBS snapshots by Using the EC2 Console (p. 54)
- Create VSS-enabled EBS snapshots by Using the AWS CLI (p. 55)
- Create VSS-enabled EBS snapshots by Using AWS Tools for Windows PowerShell (p. 56)

Create VSS-enabled EBS snapshots by Using the EC2 Console

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Run Command, and then choose Run a Command.
3. In the Command document list, choose AWSEC2-CreateVssSnapshot.
4. In the Targets section, identify the instances to which you want to download and run the script by specifying tags or selecting instances manually.
5. (Optional) In Rate control:
   - In Concurrency, specify either a number or a percentage of instances on which to run the command at the same time.
Note
If you selected targets by choosing Amazon EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document at the same time by specifying a percentage.

- In Error threshold, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify 3 errors, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.

6. In the Exclude Boot Volume list, choose an option. Use this parameter to exclude boot volumes from the backup process.

7. In the Description field, type a description. This description is applied to any snapshot created by this process (optional, but recommended).

8. In the Tags field, type keys and values for tags that you want to apply to any snapshot created by this process. Tags can help you locate, manage, and restore volumes from a list of snapshots. By default, the system populates the tag parameter with a Name key. For the value of this key, specify a name that you want to apply to snapshots created by this process. If you want to specify additional tags, separate tags by using a semicolon. For example, Key=Environment,Value=Test;Key=User,Value=TestUser1.

This parameter is optional, but we recommended that you tag snapshots. By default, the systems tags the snapshots with the device ID, and AppConsistent (for indicating successful, application-consistent VSS-enabled EBS snapshots).

9. In the Comments field, type information about this command.

10. In the Output options section, if you want to save the command output to a file, select the Write command output to an Amazon S3 bucket. Type the bucket and prefix names in the boxes.

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 Configuring Amazon SNS Notifications for Run Command (p. 169).


If successful, the command populates the list of EBS snapshots with the new snapshots. You can locate these snapshots in the list of EBS snapshots by searching for the tags you specified, or by searching for AppConsistent. If the command execution failed, view the Systems Manager command output for details about why the execution failed. If the command successfully completed, but a specific volume backup failed, you can troubleshoot the failure in the list of EBS volumes.

You can automate backups by creating a Maintenance Windows task that uses the AWSEC2-CreateVssSnapshot SSM document. For more information, see Working with Maintenance Windows (p. 225).

Create VSS-enabled EBS snapshots by Using the AWS CLI

Use the following procedure to create VSS-enabled EBS snapshots by using the AWS CLI. When you execute the command, you can specify the following parameters:

- Instance (Required): Specify one or more Amazon EC2 Windows instances. You can either manually specify instances, or you can specify tags.
- Description (Optional): Specify details about this backup.
- Tags (Optional): Specify key-value tag pairs that you want to assign to the snapshots. Tags can help you locate, manage, and restore volumes from a list of snapshots. By default, the system populates the tag parameter with a Name key. For the value of this key, specify a name that you want to apply to snapshots created by this process. You can also add custom tags to this list by using the following format: Key=Environment,Value=Test;Key=User,Value=TestUser1.
Creating VSS-enabled EBS snapshots

This parameter is optional, but we recommended that you tag snapshots. By default, the systems tags snapshots with the device ID, and AppConsistent (for indicating successful, application-consistent VSS-enabled EBS snapshots).

- Exclude Boot Volume (Optional): Use this parameter to exclude boot volumes from the backup process.

**To create VSS-enabled EBS snapshots by using the AWS CLI**

1. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2 or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

   ```bash
   aws configure
   ``

   The system prompts you to specify the following.

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

2. Execute the following command to create VSS-enabled EBS snapshots.

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

   If successful, the command populates the list of EBS snapshots with the new snapshots. You can locate these snapshots in the list of EBS snapshots by searching for the tags you specified, or by searching for AppConsistent. If the command execution failed, view the command output for details about why the execution failed.

   You can automate backups by creating a Maintenance Windows task that uses the AWSEC2-CreateVssSnapshot SSM document. For more information, see Working with Maintenance Windows (p. 225).

**Create VSS-enabled EBS snapshots by Using AWS Tools for Windows PowerShell**

Use the following procedure to create VSS-enabled EBS snapshots by using the AWS Tools for Windows PowerShell. When you execute the command, you can specify the following parameters:

- **Instance (Required):** Specify one or more Amazon EC2 Windows instances. You can either manually specify instances, or you can specify tags.
- **Description (Optional):** Specify details about this backup.
- **Tags (Optional):** Specify key-value tag pairs that you want to assign to the snapshots. Tags can help you locate, manage, and restore volumes from a list of snapshots. By default, the system populates the tag parameter with a Name key. For the value of this key, specify a name that you want to apply to snapshots created by this process. You can also add custom tags to this list by using the following format: Key=Environment,Value=Test;Key=User,Value=TestUser1.

   This parameter is optional, but we recommended that you tag snapshots. By default, the systems tags snapshots with the device ID, and AppConsistent (for indicating successful, application-consistent VSS-enabled EBS snapshots).
• Exclude Boot Volume (Optional): Use this parameter to exclude boot volumes from the backup process.

To create VSS-enabled EBS snapshots by using AWS Tools for Windows PowerShell

1. Open AWS Tools for Windows PowerShell and execute the following command to specify your credentials. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

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

2. Execute the following command to set the Region for your PowerShell session. The example uses the us-east-2 Region.

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

3. Execute the following command to create VSS-enabled EBS snapshots.

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

If successful, the command populates the list of EBS snapshots with the new snapshots. You can locate these snapshots in the list of EBS snapshots by searching for the tags you specified, or by searching for AppConsistent. If the command execution failed, view the command output for details about why the execution failed. If the command successfully completed, but a specific volume backup failed, you can troubleshoot the failure in the list of EBS snapshots.

You can automate backups by creating a Maintenance Windows task that uses the AWSEC2-CreateVssSnapshot SSM document. For more information, see Working with Maintenance Windows (p. 225).

Creating VSS-Enabled EBS Snapshots by Using the AWSEC2-ManageVssIO SSM Document (Advanced)

You can use the following script and the pre-defined AWSEC2-ManageVssIO SSM document to temporarily pause I/O, create VSS-enabled EBS snapshots, and restart I/O. This process runs in the context of the user who executes the command. If the user has sufficient permission to create and tag snapshots, then Systems Manager can create and tag VSS-enabled EBS snapshots without the need for the additional IAM snapshot role on the instance.

In contrast, the AWSEC2-CreateVssSnapshot document requires that you assign the IAM snapshot role to each instance for which you want to create EBS snapshots. If you don’t want to provide additional IAM permissions to your instances for policy or compliance reasons, then you can use the following script.

Before You Begin

Note the following important details about this process:

• This process uses a PowerShell script (CreateVssSnapshotAdvancedScript.ps1) to take snapshots of all volumes on the instances you specify, except root volumes. If you need to take snapshots of root volumes, then you must use the AWSEC2-CreateVssSnapshots SSM document.

• The script calls the AWSEC2-ManageVssIO document twice. The first time with the **Action** parameter set to **Freeze**, which pauses all I/O on the instances. The second time, the **Action** parameter is set to **Thaw**, which forces I/O to resume.

57
To create VSS-enabled EBS snapshots by using the AWSEC2-ManageVssIO SSM document

1. Open AWS Tools for Windows PowerShell and execute the following command to specify your credentials. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

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

2. Execute the following command to set the Region for your PowerShell session. The example uses the us-east-2 Region.

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

3. Download the CreateVssSnapshotAdvancedScript.zip file and extract the file contents.

4. Open the script in a simple text editor, edit the sample call at the bottom of the script, and then run it.

If successful, the command populates the list of EBS snapshots with the new snapshots. You can locate these snapshots in the list of EBS snapshots by searching for the tags you specified, or by searching for AppConsistent. If the command execution failed, view the command output for details about why the execution failed. If the command was successfully completed, but a specific volume backup failed, you can troubleshoot the failure in the list of EBS volumes.

Restoring Volumes from VSS-enabled EBS snapshots

You can use the RestoreVssSnapshotSampleScript.ps1 script to restore volumes on an instance from VSS-enabled EBS snapshots. This script performs the following tasks:

- Stops an instance
- Removes all existing drives from the instance (except the boot volume, if it was excluded)
- Creates new volumes from the snapshots
- Attaches the volumes to the instance by using the device ID tag on the snapshot
- Restarts the instance

**Important**

The following script detaches all volumes attached to an instance, and then creates new volumes from a snapshot. Make sure that you have properly backed-up the instance. The old volumes are not deleted. If you want, you can edit the script to delete the old volumes.

To restore volumes from VSS-enabled EBS snapshots

1. Open AWS Tools for Windows PowerShell and execute the following command to specify your credentials. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).
2. Execute the following command to set the Region for your PowerShell session. The example uses the us-east-2 Region.

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

3. Download the `RestoreVssSnapshotSampleScript.zip` file and extract the file contents.
4. Open the script in a simple text editor, edit the sample call at the bottom of the script, and then run it.
AWS Systems Manager Resource Groups

A resource group is a collection of AWS resources that are all in the same AWS region, and that match criteria provided in a query. You build queries in the Resource Groups console, or pass them as arguments to Resource Groups commands in the AWS CLI. With Resource Groups, you can create a custom console that organizes and consolidates information based on criteria that you specify in tags. After you've created groups in Resource Groups, use AWS Systems Manager tools such as Automation, Run Command, Patch Manager, and Maintenance Windows to simplify management tasks on your groups of resources. You can also use groups as the basis for viewing monitoring and configuration insights in AWS Systems Manager. For more information, see AWS Resource Groups.
AWS Systems Manager Insights

AWS Systems Manager (formerly Amazon EC2 Systems Manager) provides the following features and capabilities for centrally viewing data about your AWS resources.

Topics
- AWS Systems Manager Inventory Manager (p. 61)
- AWS Systems Manager Configuration Compliance (p. 84)

AWS Systems Manager Inventory Manager

You can use AWS Systems Manager Inventory to collect operating system (OS), application, and instance metadata from your Amazon EC2 instances and your on-premises servers or virtual machines (VMs) in your hybrid environment. You can query the metadata to quickly understand which instances are running the software and configurations required by your software policy, and which instances need to be updated.

Contents
- Getting Started with Inventory (p. 61)
- About Systems Manager Inventory (p. 61)
- Configuring Inventory Collection (p. 69)
- Configuring Resource Data Sync for Inventory (p. 71)
- Querying Inventory Collection (p. 74)
- Systems Manager Inventory Manager Walkthroughs (p. 74)

Getting Started with Inventory

To get started with Inventory, complete the following tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify Systems Manager prerequisites.</td>
<td>Systems Manager Prerequisites (p. 4)</td>
</tr>
<tr>
<td>Configure Inventory by creating a Systems Manager State Manager association.</td>
<td>Configuring Inventory Collection (p. 69) (Amazon EC2 console) Collecting Inventory by Using the AWS CLI (p. 76)</td>
</tr>
</tbody>
</table>

About Systems Manager Inventory

When you configure Systems Manager Inventory, you specify the type of metadata to collect, the instances from where the metadata should be collected, and a schedule for metadata collection. These configurations are saved with your AWS account as a State Manager association.
**Note**

Inventory only collects metadata. It does not collect any personal or proprietary data.

The following table describes the different aspects of Inventory collection that you configure.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of information to collect</td>
<td>You can configure Inventory to collect the following types of metadata:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Applications</strong>: Application names, publishers, versions, etc.</td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS components</strong>: EC2 driver, agents, versions, etc.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Files</strong>: Name, size, version, installed date, modification and last accessed times, etc.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Network configuration</strong>: IP address, MAC address, DNS, gateway, subnet mask, etc.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Windows updates</strong>: Hotfix ID, installed by, installed date, etc.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Instance details</strong>: System name, operating systems (OS) name, OS version, last boot, DNS, domain, work group, OS architecture, etc.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Services</strong>: Name, display name, status, dependent services, service type, start type, etc.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Windows Registry</strong>: Registry key path, value name, value type, and value.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Windows roles</strong>: Name, display name, path, feature type, installed state, etc.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Custom inventory</strong>: Metadata that was assigned to a managed instance as described in Working with Custom Inventory (p. 68).</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>                                                                            To view a list of all metadata collected by Inventory, see Metadata Collected by Inventory (p. 63).</td>
</tr>
<tr>
<td>Instances to collect information from</td>
<td>You can individually select instances or target groups of instances by using Amazon EC2 tags.</td>
</tr>
<tr>
<td>When to collect information</td>
<td>You can specify a collection interval in terms of minutes, hours, days, and weeks. The shortest collection interval is every 30 minutes.</td>
</tr>
</tbody>
</table>

Depending on the amount of data collected, the system can take several minutes to report the data to the output you specified. After the information is collected, the metadata is sent over a secure HTTPS channel to a plain-text AWS store that is accessible only from your AWS account. You can view the data in the Amazon S3 bucket you specified, or in the Amazon EC2 console on the **Inventory** tab for your managed instance. The **Inventory** tab includes several predefined filters to help you query the data.

To start collecting inventory on your managed instance, see Configuring Inventory Collection (p. 69) and Collecting Inventory by Using the AWS CLI (p. 76).
Metadata Collected by Inventory

The following sample shows the complete list of metadata collected by each Inventory plugin.

```json
[  
    {
      "typeName": "AWS:InstanceInformation",
      "version": "1.0",
      "attributes": [
        { "name": "AgentType", "dataType": "STRING" },
        { "name": "AgentVersion", "dataType": "STRING" },
        { "name": "ComputerName", "dataType": "STRING" },
        { "name": "IamRole", "dataType": "STRING" },
        { "name": "InstanceId", "dataType": "STRING" },
        { "name": "IpAddress", "dataType": "STRING" },
        { "name": "PlatformName", "dataType": "STRING" },
        { "name": "PlatformType", "dataType": "STRING" },
        { "name": "PlatformVersion", "dataType": "STRING" },
        { "name": "ResourceType", "dataType": "STRING" }
      ]
    },
    {
      "typeName": "AWS:Application",
      "version": "1.1",
      "attributes": [
        { "name": "Name", "dataType": "STRING" },
        { "name": "ApplicationType", "dataType": "STRING" },
        { "name": "Publisher", "dataType": "STRING" },
        { "name": "Version", "dataType": "STRING" },
        { "name": "InstalledTime", "dataType": "STRING" },
        { "name": "Architecture", "dataType": "STRING" },
        { "name": "URL", "dataType": "STRING" },
        { "name": "Summary", "dataType": "STRING" },
        { "name": "PackageId", "dataType": "STRING" }
      ]
    },
    {
      "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": "InstalledOtherCount",  "dataType": "NUMBER"}, { "name": "MissingCount",  "dataType": "NUMBER"}, { "name": "NotApplicableCount",  "dataType": "NUMBER"}, { "name": "FailedCount",  "dataType": "NUMBER"}, { "name": "OperationType",  "dataType": "STRING"}, { "name": "OperationStartTime",  "dataType": "STRING"}, { "name": "OperationEndTime",  "dataType": "STRING"} ] },

{ "typeName": "AWS:PatchCompliance",  "version": "1.0",  "attributes": [ { "name": "Title",  "dataType": "STRING"}, { "name": "KBIID",  "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"}, { "name": "ExecutionId",  "dataType": "STRING"}, { "name": "ExecutionType",  "dataType": "STRING"}, { "name": "ExecutionTime",  "dataType": "STRING"}, { "name": "Id",  "dataType": "STRING"},
### About Systems Manager Inventory

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>STRING</td>
</tr>
<tr>
<td>Status</td>
<td>STRING</td>
</tr>
<tr>
<td>Severity</td>
<td>STRING</td>
</tr>
<tr>
<td>DocumentName</td>
<td>STRING</td>
</tr>
<tr>
<td>DocumentVersion</td>
<td>STRING</td>
</tr>
<tr>
<td>Classification</td>
<td>STRING</td>
</tr>
<tr>
<td>PatchBaselineId</td>
<td>STRING</td>
</tr>
<tr>
<td>PatchSeverity</td>
<td>STRING</td>
</tr>
<tr>
<td>PatchState</td>
<td>STRING</td>
</tr>
<tr>
<td>PatchGroup</td>
<td>STRING</td>
</tr>
<tr>
<td>InstalledTime</td>
<td>STRING</td>
</tr>
</tbody>
</table>

### ComplianceSummary

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComplianceType</td>
<td>STRING</td>
</tr>
<tr>
<td>PatchGroup</td>
<td>STRING</td>
</tr>
<tr>
<td>PatchBaselineId</td>
<td>STRING</td>
</tr>
<tr>
<td>Status</td>
<td>STRING</td>
</tr>
<tr>
<td>OverallSeverity</td>
<td>STRING</td>
</tr>
<tr>
<td>ExecutionId</td>
<td>STRING</td>
</tr>
<tr>
<td>ExecutionType</td>
<td>STRING</td>
</tr>
<tr>
<td>ExecutionTime</td>
<td>STRING</td>
</tr>
<tr>
<td>CompliantCriticalCount</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CompliantHighCount</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CompliantMediumCount</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CompliantLowCount</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CompliantInformationalCount</td>
<td>NUMBER</td>
</tr>
<tr>
<td>NonCompliantCriticalCount</td>
<td>NUMBER</td>
</tr>
<tr>
<td>NonCompliantHighCount</td>
<td>NUMBER</td>
</tr>
<tr>
<td>NonCompliantMediumCount</td>
<td>NUMBER</td>
</tr>
<tr>
<td>NonCompliantLowCount</td>
<td>NUMBER</td>
</tr>
<tr>
<td>NonCompliantInformationalCount</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

### InstanceDetailedInformation

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUModel</td>
<td>STRING</td>
</tr>
<tr>
<td>CPUCores</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CPUs</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CPUSpeedMHz</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CPUSockets</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CPUHyperThreadEnabled</td>
<td>STRING</td>
</tr>
<tr>
<td>OSServicePack</td>
<td>STRING</td>
</tr>
</tbody>
</table>

### Service

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>STRING</td>
</tr>
<tr>
<td>DisplayName</td>
<td>STRING</td>
</tr>
<tr>
<td>ServiceType</td>
<td>STRING</td>
</tr>
<tr>
<td>Status</td>
<td>STRING</td>
</tr>
<tr>
<td>DependentServices</td>
<td>STRING</td>
</tr>
<tr>
<td>ServicesDependedOn</td>
<td>STRING</td>
</tr>
<tr>
<td>StartType</td>
<td>STRING</td>
</tr>
</tbody>
</table>
AWS Systems Manager User Guide
About Systems Manager Inventory

"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" }
]

Working with File and Windows Registry Inventory

Systems Manager Inventory enables you to search and inventory files on Windows and Linux operating systems. You can also search and inventory the Windows Registry.

Files: You can collect metadata information about files, including file names, the time files were created, the 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 instances. By default, Inventory scans a maximum of 5,000 directories.

**Note**

Inventory collects metadata for a maximum of 500 files across all specified paths.
Here are some examples of how to specify the parameters when performing an inventory of files.

- On Linux, collect metadata of .sh files in the /home/ec2-user directory, excluding all subdirectories.

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

- On Windows, collect metadata of all *.exe files in the Program Files folder, including subdirectories recursively.

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

- On Windows, collect metadata of specific log patterns.

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

- Limit the directory count when performing recursive collection.

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

**Windows Registry**: You can collect Windows Registry keys and values. You can choose a key path and collect all keys and values recursively. You can also collect a specific registry key and its value for a specific path. Inventory collects the key path, name, type, and the value.

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

- **Path**: The path to the Registry key.
- **Recursive**: A Boolean value indicating whether Inventory should recursively traverse Registry paths.
- **ValueNames**: An array of value names for performing inventory of Registry keys. If you use this parameter, Systems Manager will inventory only the specified value names for the specified path.

  **Note**

  Inventory collects a maximum of 250 Registry key values for all specified paths.

Here are some examples of how to specify the parameters when performing an inventory of the Windows Registry.

- Collect all keys and values recursively for a specific path.

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

- Collect all keys and values for a specific path (recursive search disabled).

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

- Collect a specific key by using the ValueNames option.

  ```json
  {"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon\MachineImage","ValueNames": ["AMIName"]}
  ```
Working with Custom Inventory

You can assign any metadata you want to your instances by creating custom inventory. For example, let's say you manage a large number of servers in racks in your data center, and these servers have been configured as Systems Manager managed instances. Currently, you store information about server rack location in a spreadsheet. With custom inventory, you can specify the rack location of each instance as metadata on the instance. When you collect Inventory by using Systems Manager, the metadata is collected with other Inventory metadata. You can then port all Inventory metadata to a central Amazon S3 bucket by using Resource Data Sync and query the data.

To assign custom inventory to an instance, you can either use the Systems Manager PutInventory API action, as described in Assigning Custom Inventory Metadata to an Instance (p. 75). Or, you can create a custom inventory JSON file and upload it to the instance. This section describes how to create the JSON file.

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

You can also specify multiple items in the file, as shown in the following example.

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

The JSON schema for custom inventory requires SchemaVersion, TypeName, and Content sections, but you can define the information in those sections.

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

TypeName is limited to 100 characters. Also, the TypeName section must start with Custom. For example, Custom:PuppetModuleInfo. Both Custom and the Data you specify must begin with
a capital letter. The following examples would cause an exception: "CUSTOM:RackInformation", "custom:rackinformation".

The Content section includes attributes and data. These items are not case-sensitive. However, if you define an attribute (for example: "Vendor": "DELL"), then you must consistently reference this attribute in your custom inventory files. If you specify "Vendor": "DELL" (using a capital “V” in vendor) in one file, and then you specify "vendor": "DELL" (using a lowercase “v” in vendor) in another file, the system returns an error.

**Note**
You must save the file with a .json extension.

After you create the file, you must save it on the instance. The following table shows the location where custom inventory JSON files must be stored on the instance:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>%SystemDrive%\ProgramData\Amazon\SSM \InstanceData&lt;instance-id&gt;\inventory\custom</td>
</tr>
<tr>
<td>Linux</td>
<td>/var/lib/amazon/ssm/&lt;instance-id&gt;/inventory/custom</td>
</tr>
</tbody>
</table>

For an example of how to use custom inventory, see [Get Disk Utilization of Your Fleet Using EC2 Systems Manager Custom Inventory Types](#).

**Related AWS Services**

Systems Manager Inventory provides a snapshot of your current inventory to help you manage software policy and improve the security posture of your entire fleet. You can extend your inventory management and migration capabilities using the following AWS services.

- AWS Config provides a historical record of changes to your inventory, along with the ability to create rules to generate notifications when a configuration item is changed. For more information, see, [Recording Amazon EC2 managed instance inventory](#) in the AWS Config Developer Guide.
- AWS Application Discovery Service is designed to collect inventory on OS type, application inventory, processes, connections, and server performance metrics from your on-premises VMs to support a successful migration to AWS. For more information, see the Application Discovery Service User Guide.

**Configuring Inventory Collection**

This section describes how to configure inventory collection on one or more managed instances by using the Amazon EC2 console. This section also describes how to aggregate inventory data from multiple AWS accounts and regions in a single Amazon S3 bucket by using Systems Manager Resource Data Sync. For an example of how to configure inventory collection using the AWS CLI, see Systems Manager Inventory Manager Walkthroughs (p. 74).

**Before You Begin**

Before you configure inventory collection, complete the following tasks.

- Update SSM Agent on the instances you want to inventory. By running the latest version of SSM Agent, you ensure that you can collect metadata for all supported inventory types. For information about how to update SSM Agent by using State Manager, see Walkthrough: Automatically Update the SSM Agent (p. 254).
- Verify that your instances meet Systems Manager prerequisites. For more information, see Systems Manager Prerequisites (p. 4).
- (Optional) Create a JSON file to collect custom inventory. For more information, see Working with Custom Inventory (p. 68).

Configuring Collection

Use the following procedure to configure inventory collection on a managed instance using the Amazon EC2 console.

**Note**
When you configure inventory collection, you start by creating a Systems Manager State Manager association. Systems Manager collects the inventory data when the association is run. If you don't create the association first, and attempt to invoke the aws:softwareInventory plugin by using, for example, Run Command, the system returns the following error:

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

Also note that an instance can have only one inventory association configured at a time. If you configure an instance with two or more associations, the association doesn't run and no inventory data is collected.

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

**To configure inventory collection**

1. Open the Amazon EC2 console, expand **Systems Manager Shared Resources** in the navigation pane, and then choose **Managed Instances**.
2. Choose **Setup Inventory**.
3. In the **Targets** section, choose **Specify a Tag** if you want to configure inventory on multiple instances using EC2 tags. Choose **Manually Select Instances** if you want to individually choose which instances are configured for inventory.
   
   **Note**
   If you use tags, any instances created in the future with the same tag will also report inventory.

4. In the **Schedule** section, choose how often you want the system to collect inventory metadata from your instances.
5. In the **Parameters** section, use the lists to enable or disable different types of inventory collection. See the following samples if you want to create an inventory search for **Files** or the **Windows Registry**.

**Files**

- On Linux, collect metadata of .sh files in the `/home/ec2-user` directory, excluding all subdirectories.

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

- On Windows, collect metadata of all `.exe` files in the Program Files folder, including subdirectories recursively.

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

70
• On Windows, collect metadata of specific log patterns.

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

• Limit the directory count when performing recursive collection.

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

### Windows Registry

• Collect all keys and values recursively for a specific path.

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

• Collect all keys and values for a specific path (recursive search disabled).

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

• Collect a specific key by using the ValueNames option.

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

For more information about collecting File and Windows Registry inventory, see [Working with File and Windows Registry Inventory](p. 66).

6. In the Advanced section, choose Write to S3 if you want to store the association execution status in an Amazon S3 bucket.

7. Choose Setup Inventory and then choose OK.

8. In the Managed Instances page, choose an instance that you just configured for inventory and choose the Description tab. The Association Status shows Pending until the association is created. If the status is Failed, verify that you have the latest version of the SSM Agent installed on your instances.

9. After the collection time-frame has passed, choose a managed instance, and then choose the Inventory tab.

10. Use the Inventory Type list to filter on different types of inventory data.

After you configure Inventory collection, we recommend that you configure Systems Manager Resource Data Sync. Resource Data Sync centralizes all Inventory data in a target Amazon S3 bucket and automatically updates the central storage when new Inventory data is collected. With all Inventory data stored in a target Amazon S3 bucket, you can then use services like Amazon Athena and Amazon QuickSight to query and analyze the aggregated data. For more information, see [Configuring Resource Data Sync for Inventory](p. 71).

## Configuring Resource Data Sync for Inventory

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 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 instances. Some of these instances are located in a
hybrid data center, and others are running in Amazon EC2 across multiple AWS Regions. If you have not configured Resource Data Sync for Inventory, you either need to manually gather the collected inventory data for each instance, or you have to create scripts to gather this information. You would then need to port the data into an application so that you can run queries and analyze it.

With Resource Data Sync, you perform a one-time operation that synchronizes all Inventory data from all of your managed instances. When you create the sync, you can specify managed instances from multiple AWS accounts and AWS Regions. After the sync is successfully created, Systems Manager creates a baseline of all Inventory data and saves it in the target Amazon S3 bucket. When new inventory data is collected, Systems Manager automatically updates the data in the Amazon S3 bucket. You can then quickly and cost-effectively port the data to Amazon Athena and Amazon QuickSight.

Diagram 1 shows how Resource Data Sync aggregates inventory data from managed instances in Amazon EC2 and a hybrid environment to a target Amazon S3 bucket. This diagram also shows how Resource Data Sync works with multiple AWS accounts and AWS Regions.

Diagram 1: Resource Data Sync with Multiple AWS Accounts and AWS Regions

If you delete a managed instance, Resource Data Sync preserves the Inventory file for the deleted instance. For running instances, however, Resource Data Sync automatically overwrites old inventory files when new files are created and written to the Amazon S3 bucket. If you want to track inventory changes over time, you can use the AWS Config service to track the ManagedInstanceInventory resource type. For more information, see Getting Started with AWS Config.

Related Content

- Resource Data Sync uses the following API actions: CreateResourceDataSync, ListResourceDataSync, and DeleteResourceDataSync.
- Amazon QuickSight is a business analytics service that makes it easy to build visualizations so that you can analyze and gather insights from your data. Connecting to Athena from QuickSight is a one-click process. You don’t need to provide endpoints or a user name and password. You can simply choose Athena as your data source, choose the database and tables to analyze, and start visualizing the data in QuickSight. For more information, see Amazon QuickSight User Guide.
- Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL queries. Athena doesn’t require an Amazon EC2 instance to run, so there is no
infrastructure to manage. You pay only for the queries that you run. For more information, see Amazon Athena User Guide.

Creating a Resource Data Sync for Inventory

Use the following procedure to create a Resource Data Sync for Inventory by using the Amazon EC2 console.

**Note**

You can use AWS Key Management Service (AWS Key Management Service) to encrypt the data sent by Systems Manager to the centralized Amazon S3 bucket. Currently, you can only configure encryption by using the AWS CLI. For an example of how to create an encrypted sync by using the AWS CLI and how to work with the centralized data in Amazon Athena and Amazon QuickSight, see Using Resource Data Sync to Aggregate Inventory Data (p. 78).

To create a Resource Data Sync for Inventory

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create a bucket to store your aggregated Inventory data. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide. Make a note of the bucket name and the AWS Region where you created it.
3. Choose the Permissions tab, and then choose Bucket Policy.
4. Copy and paste the following bucket policy into the policy editor. Replace `Bucket-Name` and `Account-ID` with the name of the Amazon S3 bucket you created and a valid AWS account ID. Optionally, replace `Bucket-Prefix` with the name of an Amazon S3 prefix (subdirectory). If you didn't create a prefix, remove `Bucket-Prefix` from the ARN in the policy.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SSMBucketPermissionsCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "ssm.amazonaws.com"
            },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::
Bucket-Name"
        },
        {
            "Sid": "SSMBucketDelivery",
            "Effect": "Allow",
            "Principal": {
                "Service": "ssm.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": ["arn:aws:s3:::
Bucket-Name/Bucket-Prefix/*accountid=
Account-ID/*"],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

5. Open the Amazon EC2 console, expand Systems Manager Shared Resources in the navigation pane, and choose Managed Instances.
6. Choose Resource Data Syncs, and then choose Create a Resource Data Sync.
7. In the **Sync Name** field, type a name for the sync configuration.
8. In the **Bucket Name** field, type the name of the Amazon S3 bucket you created at the start of this procedure.
9. (Optional) In the **Bucket Prefix** field, type the name of an Amazon S3 bucket prefix (subdirectory).
10. 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 type the name of the Region.

    **Note**
    If the sync and the target Amazon S3 bucket are located in different regions, you may be subject to data transfer pricing. For more information, see Amazon S3 Pricing.
11. Choose **Create**.

### Querying Inventory Collection

After you collect inventory data, you can use the filter capability on the **Inventory** tab to filter on or filter out the instances you want.

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

#### To filter managed instance metadata

1. Open the Amazon EC2 console, expand **Systems Manager Shared Resources** in the navigation pane, and then choose **Managed Instances**.
2. Choose the **Inventory** tab.
3. In the **Inventory Type** list, choose an attribute to filter on. For example: **AWS:Application**.
4. Choose the filter bar below the **Inventory Type** list to view a list of attributes on which to filter.
5. Choose a delimiter from the list. For example, choose **begins-with**.
6. Type a value. For example, type "ssm" and then choose the search icon at the left of the filter bar. The system returns all relevant managed instances.

    **Note**
    You can combine multiple filters to refine your search.

### Systems Manager Inventory Manager Walkthroughs

Use the following walkthroughs to collect and manage Inventory data. We recommend that you initially perform these walkthroughs with managed instances in a test environment.

#### Before You Begin

Before you start these walkthroughs, complete the following tasks.

- Update SSM Agent on the instances you want to inventory. By running the latest version of SSM Agent, you ensure that you can collect metadata for all supported inventory types. For information about how to update SSM Agent by using State Manager, see Walkthrough: Automatically Update the SSM Agent (p. 254).
- Verify that your instances meet Systems Manager prerequisites. For more information, see Systems Manager Prerequisites (p. 4).
- (Optional) Create a JSON file to collect custom inventory. For more information, see Working with Custom Inventory (p. 68).
Assigning Custom Inventory Metadata to an Instance

The following procedure walks you through the process of using the `PutInventory` API action to assign custom Inventory metadata to a managed instance. This example assigns rack location information to an instance. For more information about custom Inventory, see Working with Custom Inventory (p. 68).

To assign custom inventory metadata to an instance

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 a Region. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM).

   ```bash
   aws configure
   ``

   The system prompts you to specify the following.

   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. Execute the following command to assign rack location information to an instance.

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

4. Execute the following command to view custom inventory entries for this instance.

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

   The system responds with information like the following.

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

5. Execute the following command to view the custom metadata.

   ```bash
   aws ssm get-inventory
   ```
Collecting Inventory by Using the AWS CLI

The following procedure walks you through the process of using Inventory to collect metadata from an Amazon EC2 instance. When you configure Inventory collection, you start by creating a Systems Manager State Manager association. Systems Manager collects the Inventory data when the association is run. If you don't create the association first, and attempt to invoke the aws:softwareInventory plugin by using, for example, Run Command, the system returns the following error:

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

**Note**
An instance can have only have one Inventory association configured at a time. If you configure an instance with two or more Inventory associations, the association doesn't run and no inventory data is collected.

To gather inventory from an instance

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 a Region. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM).

```bash
aws configure
```

The system prompts you to specify the following.

- 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. Execute the following command to create a State Manager association that runs Inventory on the instance. This command configures the service to run every six hours and to collect network configuration, Windows Update, and application metadata from an instance.

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

The system responds with information like the following.

```json
{
  "AssociationDescription": {
    "ScheduleExpression": "cron(0 0/30 * 1/1 * ? *)",
    "OutputLocation": {
      "S3Location": {
        "OutputS3KeyPrefix": "Test",
        "OutputS3BucketName": "Test bucket",
        "OutputS3Region": "us-east-1"
      }
    },
    "Name": "The name you specified",
    "Parameters": {
      "applications": ["Enabled"],
      "networkConfig": ["Enabled"],
      "windowsUpdates": ["Enabled"]
    }
  },
  "Overview": {
    "Status": "Pending",
    "DetailedStatus": "Creating"
  },
  "AssociationId": "1a2b3c4d5e6f7g-1a2b3c-1a2b3c-1a2b3c-1a2b3c4d5e6f7g",
  "DocumentVersion": "$DEFAULT",
  "LastUpdateAssociationDate": 1480544990.06,
  "Date": 1480544990.06,
  "Targets": [{
    "Values": ["i-1a2b3c4d5e6f7g"],
    "Key": "InstanceIds"
  }]
}
```

You can target large groups of instances by using the `Targets` parameter with EC2 tags.

```bash
aws ssm create-association --name "AWS-GatherSoftwareInventory" --targets "Key=tag:Environment,Values=Production" --schedule-expression "cron(0 0/30 * 1/1 * ? *)" --output-location "{ "S3Location": { "OutputS3Region": "us-east-1", "OutputS3BucketName": "Test bucket", "OutputS3KeyPrefix": "Test" }, "Parameters": { "networkConfig": "Enabled", "windowsUpdates": "Enabled", "applications": "Enabled" } }"
```
You can also inventory files and Windows Registry keys on a Windows instance by using the `files` and `windowsRegistry` inventory types with expressions. For more information about these inventory types, see Working with File and Windows Registry Inventory (p. 66).

```bash
aws ssm create-association --name "AWS-GatherSoftwareInventory" --targets "Key=instanceids,Values=i-0704358e3a9da9eb1" --schedule-expression "cron(0 0/30 * 1/1 * ? *)" --parameters '"files":"[\"Path\": "C:\Program Files\", \"Pattern\": \"*.exe\", \"Recursive\": true\\"]", "windowsRegistry": "[\"HKEY_LOCAL_MACHINE\Software\Amazon\", \"Recursive\\":true\\"]"'} --profile dev-pdx
```

4. Execute the following command to view the association status.

```bash
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"  
    }  
  ]
}
```

Using Resource Data Sync to Aggregate Inventory Data

The following walkthrough describes how to create a Resource Data Sync configuration by using the AWS CLI. A Resource Data Sync automatically ports Inventory data from all of your managed instances to a central 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 the Amazon EC2 console, see Configuring Resource Data Sync for Inventory (p. 71).

**Note**

This walkthrough includes information about how to encrypt the sync by using AWS Key Management Service (AWS KMS). Inventory does not collect any user-specific, proprietary, or sensitive data so encryption is optional. For more information about AWS KMS, see AWS Key Management Service Developer Guide.

**Before You Begin**

Before you start this walkthrough, you must collect Inventory metadata from your managed instances. For the purpose of the Amazon Athena and Amazon QuickSight sections in this walkthrough, we recommend that you collect Application metadata. For more information about how to collect Inventory data, see Collecting Inventory by Using the AWS CLI (p. 76).

(Optional) If you want to encrypt the sync by using AWS KMS, then you must either create a new key that includes the following policy, or you must update an existing key and add this policy to it.

```json
{"Version":"2012-10-17",  
"Id":"ssm-access-policy",  
"Statement": [{  
  "Sid": "Use the AWS Systems Manager Service",  
  "Effect": "Allow",  
  "Action": [  
    "ssm:CreateInstanceAssociation",  
    "ssm:DescribeInstanceAssociation",  
    "ssm:DescribeInstanceAssociationsStatus",  
    "ssm:GetInventory",  
    "ssm:GetInventoryValues",  
    "ssm:GetResourceDataSyncConfigurations",  
    "ssm:GetResourceDataSyncrowserAssociations",  
    "ssm:GetResourceDataSyncOperations",  
    "ssm:GetResourceDataSyncStatus",  
    "ssm:GetResourceGroups",  
    "ssm:GetReportSummaries",  
    "ssm:GetUpdates",  
    "ssm:GetUpdateSet",  
    "ssm:ListAssociations",  
    "ssm:ListDocumentProvisioningStatus",  
    "ssm:ListInstanceAssociations",  
    "ssm:ListInstanceAssociationStatusInfos",  
    "ssm:ListInstanceAssociationTypes",  
    "ssm:ListImages",  
    "ssm:ListInstanceDataSyncOperations",  
    "ssm:ListInstanceDataSyncStatus",  
    "ssm:ListInstanceDataSyncs",  
    "ssm:ListInstanceProvisioningStatus",  
    "ssm:ListPatches",  
    "ssm:PutInventory",  
    "ssm:PutResourceDataSyncConfigurations",  
    "ssm:PutResourceDataSyncs",  
    "ssm:PutUpdateSet",  
    "ssm:UpdateAssociationSchedule",  
    "ssm:UpdateInstanceAssociationSchedule",  
    "ssm:UpdateInventoryValues",  
    "ssm:UpdatePatches",  
    "ssm:UpdateResourceDataSyncOperations",  
    "ssm:UpdateResourceDataSyncStatus",  
    "ssm:UpdateResourceDataSyncs"  
  ],  
  "Resource": [  
    "ssm:/*"  
  ]  
}]
```
To create a Resource Data Sync for Inventory

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create a bucket to store your aggregated Inventory data. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide. Make a note of the bucket name and the AWS Region where you created it.
3. After you create the bucket, choose the Permissions tab, and then choose Bucket Policy.
4. Copy and paste the following bucket policy into the policy editor. Replace Bucket-Name and Account-ID with the name of the Amazon S3 bucket you created and a valid AWS account ID. Optionally, replace Bucket-Prefix with the name of an Amazon S3 prefix (subdirectory). If you did not create a prefix, remove Bucket-Prefix/ from the ARN in the policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SSMBucketPermissionsCheck",
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::Bucket-Name"
    },
    {
      "Sid": "SSMBucketDelivery",
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": ["arn:aws:s3:::Bucket-Name/Bucket-Prefix/*/accountid=Account-ID/*"],
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    }
  ]
}
```

5. (Optional) If you want to encrypt the sync, then you must add the following policy to the bucket. Repeat the previous step to add the following policy to the bucket.

```
{
  "Statement": [
    {
      "Sid": "ssm-access-policy-statement",
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      },
      "Resource": "arn:aws:kms:region:AWS-account-ID:key/KMS-key-id"
    }
  ]
}
```
6. **Download** the latest version of the AWS CLI to your local machine.

7. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM).

   ```bash
   aws configure
   ```

   The system prompts you to specify the following.

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

8. (Optional) If you want to encrypt the sync, execute the following command to verify that the bucket policy is enforcing the KMS key requirement.

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

9. Execute 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 are currently logged into.

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

   You can use the `region` parameter to specify where the sync configuration should be created. In the following example, Inventory data from the us-west-1 Region, will be synchronized in the Amazon S3 bucket in the us-west-2 Region.

   ```bash
   aws ssm create-resource-data-sync --sync-name InventoryDataWest --s3-destination "BucketName=InventoryData,Prefix=HybridEnv,SyncFormat=JsonSerDe,Region=us-west-2" --region us-west-1
   ```
(Optional) If you want to encrypt the sync by using AWS KMS, execute 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.

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

10. Execute 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
```

11. After the sync configuration is created successfully, browse the target bucket in Amazon S3. Inventory data should appear within a few minutes.

**Working with the Data in Amazon Athena**

The following section describes how to view and query the data in Amazon Athena.

**To view and query the data in Amazon Athena**

1. In the Amazon Athena console, 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.

2. Copy and paste the following statement into the query editor and then choose Run Query. Replace `Bucket-Name` and `Bucket-Prefix` with the name and prefix of the Amazon S3 target.

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

The system creates a table with partitioning information for the Application inventory type.

3. Copy and paste the following statement into the query editor and then choose Run Query.
MSCK REPAIR TABLE ssminventory.AWS_Application

The system partitions the table.

**Note**
If you create Resource Data Syncs from additional AWS Regions or accounts, then you must run this command again to update the partitions. You may also need to update your Amazon S3 bucket policy.

4. To preview your data, choose the view icon next to the AWS_Application table.

5. Copy and paste the following statement into the query editor and then choose Run Query.

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

The query returns a count of different versions of aws-cfn-bootstrap, which is an AWS application present on Amazon EC2 Linux and Windows instances.

6. Individually copy and paste the following statements into the query editor, replace `Bucket-Name` and `Bucket-Prefix` with information for Amazon S3, and then choose Run Query. These statements set up additional Inventory tables in Athena.

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

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

82
CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_InstanceInformation ( 
  `AgentType` string,
  `AgentVersion` string,
  `ComputerName` string,
  `IamRole` string,
  `InstanceId` string,
  `IpAddress` string,
  `PlatformName` string,
  `PlatformType` string,
  `PlatformVersion` string
)
PARTITIONED BY (AccountId string, Region string, ResourceType string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ( 
  'serialization.format' = '1'
) LOCATION 's3://Bucket-Name/Bucket-Prefix/AWS:InstanceInformation/

MSCK REPAIR TABLE ssminventory.AWS_InstanceInformation

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

MSCK REPAIR TABLE ssminventory.AWS_Network

CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_PatchCompliance ( 
  `ResourceId` string,
  `Title` string,
  `KBId` string,
  `Classification` string,
  `Severity` string,
  `State` string,
  `InstalledTime` string
)
PARTITIONED BY (AccountId string, Region string, ResourceType string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ( 
  'serialization.format' = '1'
) LOCATION 's3://Bucket-Name/Bucket-Prefix/AWS:PatchCompliance/

MSCK REPAIR TABLE ssminventory.AWS_PatchCompliance

CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_PatchSummary ( 
  `ResourceId` string,
AWS Systems Manager Configuration Compliance

You can use AWS Systems Manager Configuration Compliance to scan your fleet of managed instances for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren't compliant. By default, Configuration Compliance displays compliance data about Systems Manager Patch Manager patching and Systems Manager State Manager associations. You can also customize the service and create your own compliance types based on your IT or business requirements. You can quickly remediate issues using Systems Manager Run Command, State Manager, or Amazon CloudWatch Events. You can also port data to Amazon Athena and Amazon QuickSight to generate fleet-wide reports.

Configuration Compliance is offered at no additional charge. You only pay for the AWS resources that you use.

Contents
- Getting Started with Configuration Compliance (p. 85)
- About Configuration Compliance (p. 87)
- Remediating Compliance Issues (p. 89)
- Systems Manager Configuration Compliance Manager Walkthrough (p. 90)
Getting Started with Configuration Compliance

To get started with Configuration Compliance, complete the following tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Compliance works with Patch Manager patch data, State Manager associations, and custom compliance types on Systems Manager managed instances. Verify that your Amazon EC2 instances and hybrid machines are configured as managed instances by verifying Systems Manager prerequisites.</td>
<td>Systems Manager Prerequisites (p. 4)</td>
</tr>
<tr>
<td>Update the SSM Agent on your managed instances to the latest version.</td>
<td>Installing and Configuring SSM Agent (p. 13)</td>
</tr>
<tr>
<td>If you plan to monitor patch compliance, verify that you've configured Systems Manager Patch Manager. You must perform patching operations by using Patch Manager before Configuration Compliance can display patch compliance data.</td>
<td>AWS Systems Manager Patch Manager (p. 192)</td>
</tr>
<tr>
<td>If you plan to monitor association compliance, verify that you've created State Manager associations. You must create associations before Configuration Compliance can display association compliance data.</td>
<td>AWS Systems Manager State Manager (p. 249)</td>
</tr>
<tr>
<td>(Optional) Create custom compliance types.</td>
<td>Systems Manager Configuration Compliance Manager Walkthrough (p. 90)</td>
</tr>
<tr>
<td>(Optional) Create a Resource Data Sync to aggregate all compliance data in a target Amazon S3 bucket.</td>
<td>Creating a Resource Data Sync for Configuration Compliance (p. 85)</td>
</tr>
</tbody>
</table>

Creating a Resource Data Sync for Configuration Compliance

You can use Systems Manager Resource Data Sync to send compliance data from all of your managed instances to a target Amazon S3 bucket. When you create the sync, you can specify managed instances from multiple AWS accounts, AWS Regions, and your on-premises hybrid environment. Resource Data Sync then automatically updates the centralized data when new compliance data is collected. With all compliance data stored in a target Amazon S3 bucket, you can use services like Amazon Athena and Amazon QuickSight to query and analyze the aggregated data. Configuring Resource Data Sync for configuration compliance is a one-time operation.

The following graphic shows how Resource Data Sync aggregates all data from different accounts, Regions, and your hybrid environment to a central repository.
Use the following procedure to create a Resource Data Sync for Configuration Compliance by using the Amazon EC2 console.

**To create a Resource Data Sync for Configuration Compliance**

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 Getting Started Guide. Make a note of the bucket name and the AWS Region where you created it.
3. Choose the Permissions tab, and then choose Bucket Policy.
4. Copy and paste the following bucket policy into the policy editor. Replace `Bucket-Name` and `Account-ID` with the name of the Amazon S3 bucket you created and a valid AWS account ID. Optionally, replace `Bucket-Prefix` with the name of an Amazon S3 prefix (subdirectory). If you didn't create a prefix, remove `Bucket-Prefix/` from the ARN in the policy.

```json
{
   "Version": "2012-10-17",
   "Statement": [
   {
      "Sid": "SSMBucketPermissionsCheck",
      "Effect": "Allow",
      "Principal": {
         "Service": "ssm.amazonaws.com"
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::Bucket-Name"
    },
    {
      "Sid": " SSMBucketDelivery",
      "Effect": "Allow",
      "Principal": {
         "Service": "ssm.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": ["arn:aws:s3:::Bucket-Name/Bucket-Prefix/*/accountid=Account-ID/**"]
    }
   ]
}
```
5. Open the Amazon EC2 console, expand Systems Manager Shared Resources in the navigation pane, and choose Managed Instances.

6. Choose Resource Data Syncs, and then choose Create a Resource Data Sync.

7. In the Sync Name field, type a name for the sync configuration.

8. In the Bucket Name field, type the name of the Amazon S3 bucket you created at the start of this procedure.

9. (Optional) In the Bucket Prefix field, type the name of an Amazon S3 bucket prefix (subdirectory).

10. 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 type the name of the Region.

    Note
If the sync and the target Amazon S3 bucket are located in different regions, you may be subject to data transfer pricing. For more information, see Amazon S3 Pricing.

11. Choose Create.

### About Configuration Compliance

This section includes information about the different types of information, compliance types, that you can view by using Configuration Compliance. Configuration Compliance currently supports Patch Manager patching data, State Manager associations, and custom compliance types.

Topics
- About Patch Compliance (p. 87)
- About Association Compliance (p. 88)
- About Custom Compliance (p. 88)

### About Patch Compliance

After you configure and execute patching by using Patch Manager, Configuration Compliance automatically reports patch compliance status. You don't need to perform any additional steps to view these statuses. If you want to assign a specific patch compliance status to an instance, you can use the PutComplianceItems API action to explicitly assign a status. You can use this API action from the AWS CLI, AWS Tools for Windows PowerShell, or the SDK. You currently can't assign compliance status by using the Amazon EC2 console.

You can view and drill down into patch compliance details in the Amazon EC2 console on the Compliance Configuration page, or you can use the following API actions:

- **ListComplianceSummaries**: Returns a summary count of compliant and non-compliant patch statuses according to the filter you specify.
- **ListResourceComplianceSummaries**: 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.
• **DescribePatchGroupState**: Returns high-level aggregated patch compliance state for a patch group.
• **DescribeInstancePatchStatesForPatchGroup**: Returns the high-level patch state for the instances in the specified patch group.

The results for each patch show one of the following states.

• **Installed**: Either the patch was already installed, or Patch Manager installed it when the AWS-RunPatchBaseline document was run on the instance.
• **Installed_Other**: The patch is not in the baseline, but it is installed on the instance. An individual might have installed it manually.
• **Missing**: The patch is approved in the baseline, but it's not installed on the instance. If you configure the AWS-RunPatchBaseline document task to scan (instead of install) the system reports this status for patches that were located during the scan, but have not been installed.
• **Not_Applicable**: The patch is approved in the baseline, but the service or feature that uses the patch is not installed on the instance. For example, a patch for a web server service would show Not_Applicable if it was approved in the baseline, but the web service is not installed on the instance.
• **Failed**: The patch is approved in the baseline, but it could not be installed. To troubleshoot this situation, review the command output for information that might help you understand the problem.

To view an example of how to configure patching and how to view patch compliance details by using the AWS CLI, see Systems Manager Patch Manager Walkthroughs (p. 198).

**About Association Compliance**

After you create one or more State Manager associations, Configuration Compliance automatically reports association compliance status. You don't need to perform any additional steps to view these statuses. If you want to assign a specific association compliance status to an instance, you can use the PutComplianceItems API action to explicitly assign a status. You can use this API action from the AWS CLI, AWS Tools for Windows PowerShell, or the SDK. You currently can't assign compliance status by using the Amazon EC2 console.

You can view association compliance details in the Amazon EC2 console on the Compliance Configuration page, or you can use the following API actions to view compliance details:

  **Note**
  Currently, Configuration Compliance shows compliance statuses of Compliant or Non-compliant and severity of Unspecified.

• **ListComplianceSummaries**: Returns a summary count of compliant and non-compliant association statuses according to the filter you specify.
• **ListResourceComplianceSummaries**: Returns a resource-level summary count. The summary includes information about compliant and non-compliant statuses and Unspecified counts, according to the filter criteria you specify.

**About Custom Compliance**

You can assign compliance metadata to a managed instance. This metadata can then be aggregated with other compliance data for compliance reporting purposes. For example, say that your business runs versions 2.0, 3.0, and 4.0 of software X on your managed instances. The company wants to standardize on version 4.0, meaning that instances running versions 2.0 and 3.0 are non-compliant. You can use the PutComplianceItems API action to explicitly note which managed instances are running older versions of software X. Currently you can only assign compliance metadata by using the AWS CLI, AWS Tools for Windows PowerShell, or the SDKs. The following CLI sample command assigns compliance metadata to a managed instance and specifies the compliance type in the required format Custom.
aws ssm put-compliance-items --resource-id i-1234567890 --resource-type ManagedInstance --compliance-type Custom:SoftwareXCheck --execution-summary ExecutionTime=AnyStringToDenoteTimeOrDate, --items Id=Version2.0,Title=SoftwareXVersion,Severity=CRITICAL,Status=NON_COMPLIANT

Compliance managers can then view summaries or create reports about which instances are or aren't compliant. You can assign a maximum of 10 different custom compliance types to an instance.

For an example of how to create a custom compliance type and view compliance data, see Systems Manager Configuration Compliance Manager Walkthrough (p. 90).

Remediating Compliance Issues

You can quickly remediate patch and association compliance issues by using Systems Manager Run Command. You can target either instance IDs or Amazon EC2 tags and execute the AWS-RefreshAssociation document or the AWS-RunPatchBaseline document. If refreshing the association or re-running the patch baseline fails to resolve the compliance issue, then you need to investigate your associations, patch baselines, or instance configurations to understand why the Run Command executions did not resolve the problem. For more information about running a command, see Executing Commands Using Systems Manager Run Command (p. 174).

You can also configure CloudWatch Events to perform an action in response to Configuration Compliance events. For example, if one or more instances fail to install Critical patch updates or run an association that installs anti-virus software, then you can configure CloudWatch to run the AWS-RunPatchBaseline document or the AWS-RefreshAssociation document when the Configuration Compliance event occurs. Use the following procedure to configure Configuration Compliance as the target of a CloudWatch event.

To configure Configuration Compliance as the target of a CloudWatch event

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose Events, and then choose Create rule.
3. Choose Event Pattern. Event Pattern lets you build a rule that generates events for specific actions in AWS services.
4. In the Service Name field, choose EC2 Simple Systems Manager (SSM)
5. In the Event Type field, choose Configuration Compliance.
6. Choose Add target.
7. In the Select target type list, choose SSM Run Command.
8. In the Document list, choose an SSM document to run when your target is invoked. For example, choose AWS-RefreshAssociation for a non-compliant association event, or choose AWS-RunPatchBaseline for a non-compliant patch event.
9. Specify information for the remaining fields and parameters.
   
   **Note**
   Required fields and parameters have an asterisk (*) next to the name. To create a target, you must specify a value for each required parameter or field. If you don’t, the system creates the rule, but it won’t execute.
10. Choose Configure details and complete the wizard.
Systems Manager Configuration Compliance Manager Walkthrough

The following procedure walks you through the process of using the PutComplianceItems API action to assign custom compliance metadata to a resource. You can also use this API action to manually assign patch or association compliance metadata to an instance, as shown in the following walkthrough. For more information about custom compliance, see About Custom Compliance (p. 88).

To assign custom compliance metadata to a managed instance

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 a Region. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM).

   ```
   aws configure
   ```

   The system prompts you to specify the following.

   - 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. Execute the following command to assign custom compliance metadata to an instance. Currently the only supported resource type is `ManagedInstance`.

   ```
   aws ssm put-compliance-items --resource-id Instance ID --resource-type ManagedInstance --compliance-type Custom:User-defined string --execution-summary ExecutionTime=User-defined time and/or date value --items Id=User-defined ID,Title=User-defined title,Severity=One or more comma-separated severities:CRITICAL,MAJOR,MINOR,INFORMATIONAL, or UNSPECIFIED,Status=COMPLIANT or NON_COMPLIANT
   ```

4. Repeat the previous step to assign additional custom compliance metadata to one or more instances. You can also manually assign patch or association compliance metadata to managed instances by using the following commands:

   Association compliance metadata

   ```
   aws ssm put-compliance-items --resource-id Instance ID --resource-type ManagedInstance --compliance-type Association --execution-summary ExecutionTime=User-defined time and/or date value --items Id=User-defined ID,Title=User-defined title,Severity=One or more comma-separated severities:CRITICAL,MAJOR,MINOR,INFORMATIONAL, or UNSPECIFIED,Status=COMPLIANT or NON_COMPLIANT,Details="\{DocumentName=The SSM document for the association,DocumentVersion=A version number\}"
   ```

   Patch compliance metadata

   ```
   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
   ```
5. Execute the following command to view a list of compliance items for a specific managed instance. Use filters to drill-down into specific compliance data.

```
aws ssm list-compliance-items --resource-ids Instance ID --resource-types ManagedInstance --filters One or more filters.
```

The following examples show you how to use this command with filters.

```
aws ssm list-compliance-items --resource-ids i-1234567890abcdef0 --resource-type ManagedInstance --filters Key=DocumentName,Values=AWS-RunPowerShellScript Key=Status,Values=NON_COMPLIANT,Type=NotEqual Key=Id,Values=cee20ae7-6388-488e-8be1-a88cc6c46dcc Key=Severity,Values=UNSPECIFIED
```

```
aws ssm list-resource-compliance-summaries --filters Key=OverallSeverity,Values=UNSPECIFIED
```

```
aws ssm list-resource-compliance-summaries --filters Key=OverallSeverity,Values=UNSPECIFIED Key=ComplianceType,Values=Association Key=InstanceId,Values=i-1234567890abcdef0
```

6. Execute 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.

```
aws ssm list-resource-compliance-summaries --filters Key=ExecutionType,Values=Command
```

```
aws ssm list-resource-compliance-summaries --filters Key=AWS:InstanceInformation.PlatformType,Values=Windows Key=OverallSeverity,Values=CRITICAL
```

7. Execute 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.

```
aws ssm list-compliance-summaries --filters Key=AWS:InstanceInformation.PlatformType,Values=Windows Key=PatchGroup,Values=TestGroup
```

```
aws ssm list-compliance-summaries --filters Key=AWS:InstanceInformation.PlatformType,Values=Windows Key=ExecutionId,Values=4adf0526-6aedd-4694-97a5-14522f4c2b6
```
AWS Systems Manager Actions

AWS Systems Manager (formerly Amazon EC2 Systems Manager) provides the following capabilities for taking action against your AWS resources.

Topics
- AWS Systems Manager Automation (p. 92)
- AWS Systems Manager Run Command (p. 160)
- AWS Systems Manager Patch Manager (p. 192)
- AWS Systems Manager Maintenance Windows (p. 218)
- AWS Systems Manager State Manager (p. 249)

AWS Systems Manager Automation

Systems Manager Automation is an AWS-hosted service that simplifies common instance and system maintenance and deployment tasks. For example, you can use Automation as part of your change management process to keep your Amazon Machine Images (AMIs) up-to-date with the latest application build. Or, let's say you want to create a backup of a database and upload it nightly to Amazon S3. With Automation, you can avoid deploying scripts and scheduling logic directly to the instance. Instead, you can run maintenance activities through Systems Manager Run Command and AWS Lambda steps orchestrated by the Automation service.

Automation enables you to do the following.

- Pre-install and configure applications and agents in your Amazon Machine Images (AMIs) using a streamlined and repeatable process that you can audit.
- Build workflows to configure and manage instances and AWS resources.
- Create your own custom workflows, or use pre-defined workflows maintained by AWS.
- Receive notifications about Automation tasks and workflows by using Amazon CloudWatch Events
- Monitor Automation progress and execution details by using the Amazon EC2 or the AWS Systems Manager console.

Contents
- Automation Use Cases (p. 92)
- Automation QuickStart (p. 95)
- Setting Up Automation (p. 96)
- Systems Manager Automation Walkthroughs (p. 103)
- Working with Automation Documents (p. 107)
- Systems Manager Automation Examples (p. 119)
- Automation System Variables (p. 140)
- Troubleshooting Systems Manager Automation (p. 149)

Automation Use Cases

This section includes common uses cases for AWS Systems Manager Automation.

Perform common IT tasks
Automation can simplify common IT tasks such as changing the state of one or more instances (using an approval workflow) and managing instance states according to a schedule. Here are some examples:

- Use the AWS-StopInstance document to request that one or more AWS Identity and Access Management (IAM) users approve the instance stop action. After the approval is received, Automation stops the instance.
- Use the AWS-StopInstance document to automatically stop instances on a schedule by using Amazon CloudWatch Events or by using a Maintenance Window task. For example, you can configure an Automation workflow to stop instances every Friday evening, and then restart them every Monday morning.
- Use the AWS-UpdateCloudFormationStackWithApproval document to update resources that were deployed by using CloudFormation template. The update applies a new template. You can configure the Automation to request approval by one or more IAM users before the update begins.

**Safely perform disruptive tasks in bulk**

Systems Manager includes features that help you target large groups of instances by using EC2 tags, and velocity controls that help you roll out changes according to the limits you define.

Use the AWS-RestartInstanceWithApproval document to target an AWS Resource Group that includes multiple instances. You can configure the Automation workflow to use velocity controls. For example, you can specify the number of instances that should be restarted concurrently. You can also specify a maximum number of errors that are allowed before the Automation workflow is cancelled.

**Simplify complex tasks**

Automation offers one-click automations for simplifying complex tasks such as creating golden Amazon Machine Images (AMIs), and recovering unreachable EC2 instances. Here are some examples:

- Use the AWS-UpdateLinuxAMI and AWS-UpdateWindowsAMI documents to create golden AMIs from a source AMI. You can run custom scripts before and after updates are applied. You can also include or exclude specific packages from being installed.
- Use the AWSSupport-ExecuteEC2Rescue document to recover impaired instances. An instance can become unreachable for a variety of reasons, including network misconfigurations, RDP issues, or firewall settings. Troubleshooting and regaining access to the instance previously required dozens of manual steps before you could regain access. The AWSSupport-ExecuteEC2Rescue document lets you regain access by specifying an instance ID and clicking a button.

**Enhance operations security**

Using delegated administration, you can restrict or elevate user permissions for various types of tasks.

Delegated administration enables you to provide permissions for certain tasks on certain resource without having to give a user direct permission to access the resources. This improves your overall security profile. For example, assume that User1 doesn’t have permissions to restart EC2 instances, but you would like to authorize the user to do so. Instead of allowing User1 direct permissions, you can:

- Create an IAM role with the permissions required to successfully stop and start EC2 instances.
- Create an Automation document and embed the role in the document. (The easiest way to do this is to customize the AWS-RestartEC2Instance document and embed the role in the document instead of assigning an Automation service role [or assume role]).
- Modify IAM permissions for User1 and allow the user permission to execute the document. Here is an example of how you would alter the document:

```json
"Version": "2012-10-17",
"Statement": [
```
User1 can now execute this document and restart instances.

**Share best practices**

Automation lets you share best practices with rest of your organization.

You can create best practices for resource management in Automation documents and easily share the documents across AWS Regions and groups. You can also constrain the allowed values for the parameters the document accepts. Here is an example of an Automation document that launches new EC2 instances and constrains users to specify only certain AMI IDs:

**Note**
Replace the AMI value, and the value of MySecurityKeyName and IamInstanceProfileName, as needed.

```json
{
    "description":"Launch Allowed EC2 instances",
    "schemaVersion":"0.3",
    "assumeRole": "{{ AutomationAssumeRole }}",
    "parameters":{
        "AMIID":{
            "type":"String",
            "description":"Instance to value",
            "allowedValues" : [ "ami-e3bb7399","ami-55ef662f" ]
        },
        "AutomationAssumeRole": {
            "type": "String",
            "description": "(Optional) The ARN of the role that allows Automation to perform the actions on your behalf.",
            "default": ""
        }
    },
    "mainSteps":[
        {
            "name":"launchInstances",
            "action":"aws:runInstances",
            "timeoutSeconds":1200,
            "maxAttempts":1,
            "onFailure":"Abort",
            "inputs":{
                "ImageId": "{{ AMIID }}",
                "InstanceType": "t2.micro",
                "KeyName": "MySecurityKeyName",
                "MinInstanceCount": 1,
                "MaxInstanceCount": 1,
                "IamInstanceProfileName": "ManagedInstanceProfile"
            }
        }
    ]
}
```

This document ensures that users are only able to launch t2.micro instances from the specified AMI IDs.
Automation QuickStart

This section includes two walkthroughs to help you execute a simple Systems Manager Automation workflow within minutes. Each walkthrough offers a different approach for setting up and executing Automation workflows. We suggest that you perform these walkthroughs in a test environment where you have administrator permissions in AWS Identity and Access Management (IAM).

Contents

- QuickStart #1: Executing an Automation Workflow as the Current Authenticated User (p. 95)
- QuickStart #2: Executing an Automation Workflow by Using an IAM Service Role (p. 95)

QuickStart #1: Executing an Automation Workflow as the Current Authenticated User

This walkthrough shows you how to execute an Automation workflow that restarts a managed instance by using the AWS-RestartEC2Instance document. The workflow executes in the context of the current IAM user. This means that you don't need to configure additional IAM permissions as long as you have permission to run the Automation document and any actions called by the document. If you have administrator permissions in IAM, then you have permission to run this Automation.

To execute the Automation document as the current authenticated user

2. In the navigation pane, choose Managed Instances.
3. Copy the instance ID of one more managed instances that you want to restart.
4. In the navigation pane, choose Execute automation.
5. In the Automation document list choose AWS-RestartEC2Instance.
6. In the Document details section, verify that Document version is set to 1 (Default).
7. In the Input parameters section, paste one or more IDs in the Instance ID box. Separate instance IDs by using a semicolon (;).
8. Choose Execute automation. The console displays the status of the Automation execution.

QuickStart #2: Executing an Automation Workflow by Using an IAM Service Role

This walkthrough shows you how to execute an Automation workflow that restarts a managed instance by using the AWS-RestartEC2Instance document. The workflow executes the Automation by using an IAM service role (or assume role). The service role gives the Automation service permission to perform actions on your behalf. Configuring a service role is useful when you want restrict permissions and execute actions with least privilege. For example, if you want to restrict a user's privileges on a resource, such as an EC2 instance, but you want the user to execute an Automation workflow that performs a specific and allowable set of actions. In this scenario, you can create a service role with higher privileges and allow the user to execute the Automation workflow.

Before You Begin

Before you complete the following procedure, you must create the IAM service role and configure a trust relationship for Automation. For more information, see the following procedures: Task 1: Create a Service Role for Automation (p. 99) and Task 2: Add a Trust Relationship for Automation (p. 100).
To execute the Automation document by using a service role

2. In the navigation pane, choose Managed Instances.
3. Copy the instance ID of one more managed instances that you want to restart.
4. In the navigation pane, choose Execute automation.
5. In the Automation document list choose AWS-RestartEC2Instance.
6. In the Document details section, verify that Document version is set to 1 (Default).
7. In the Input parameters section, paste one or more IDs in the Instance ID box. Separate instance IDs by using a semicolon (;).
8. In the Automation Assume Role box, paste the ARN of the IAM service role.

For more examples of how use Systems Manager Automation, see Systems Manager Automation Walkthroughs (p. 103). For information about how to get started with Automation, see Setting Up Automation (p. 96).

Setting Up Automation

Setting up Automation requires that you verify user access to the Automation service and configure roles so that the service can perform actions on your instances. Optionally, you can also configure Automation to send events to Amazon CloudWatch Events. You can also configure an Automation workflow to run when a specific CloudWatch event occurs. This is called specifying Automation as the target of a CloudWatch event.

Contents
- Configuring Access for Systems Manager Automation (p. 96)
- Configuring CloudWatch Events for Systems Manager Automation (Optional) (p. 101)
- Configuring Automation as a CloudWatch Events Target (Optional) (p. 102)

Configuring Access for Systems Manager Automation

Configuring access to Systems Manager Automation requires that you complete the following tasks.

1. **Verify user access**: Verify that you have permission to run Automation workflows. If your AWS Identity and Access Management (IAM) user account, group, or role is assigned administrator permissions, then you have access to Systems Manager Automation. If you don’t have administrator permissions, then an administrator must give you permission by assigning the AmazonSSMFullAccess managed policy, or a policy that provides comparable permissions, to your IAM account, group, or role.

2. **Configure instance access by creating and assigning an instance profile role (Required)**: Each instance that runs an Automation workflow requires an IAM instance profile role. This role gives Automation permission to perform actions on your instances, such as executing commands or starting and stopping services. If you previously created an instance profile role for Systems Manager, as described in Task 2: Create an Instance Profile Role for Systems Manager (p. 8) in the Configuring Access to Systems Manager topic, then you can use this same instance profile role for Automation. For information about how to attach this role to an existing instance, see Attaching an IAM Role to an Instance in the Amazon EC2 User Guide.

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

However, the following situations still require that you specify a service role for Automation:

- When you want to restrict a user’s privileges on a resource, but you want the user to execute an Automation workflow that requires higher privileges. In this scenario, you can create a service role with higher privileges and allow the user to execute the workflow.
- Operations that you expect to run longer than 12 hours require a service role.

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

**Topics**
- Method 1: Using AWS CloudFormation to Configure Roles for Automation (p. 97)
- Method 2: Using IAM to Configure Roles for Automation (p. 98)

**Method 1: Using AWS CloudFormation to Configure Roles for Automation**

You can create an IAM instance profile role and a service role for Automation from an AWS CloudFormation template.

After you create the instance profile role, you must assign it to any instance that you plan to configure using Automation. For information about how to assign the role to an existing instance, see Attaching an IAM Role to an Instance in the Amazon EC2 User Guide. For information about how to assign the role when you create a new instance, see Task 3: Create an Amazon EC2 Instance that Uses the Systems Manager Role (p. 9) in the Configuring Access to Systems Manager topic.

**Note**
You can also use these roles and their Amazon Resource Names (ARNs) in Automation documents, such as the AWS-UpdateLinuxAmi document. Using these roles or their ARNs in Automation documents enables Automation to perform actions on your managed instances, launch new instances, and perform actions on your behalf. To view an example, see Automation CLI Walkthrough: Patch a Linux AMI (p. 105).

**Create the Instance Profile Role and Service Role Using AWS CloudFormation**

Use the following procedure to create the required IAM roles for Systems Manager Automation by using AWS CloudFormation.

**To create the required IAM roles**

1. Choose the **Launch Stack** button. The button opens the AWS CloudFormation console and populates the **Specify an Amazon S3 template URL** field with the URL to the Systems Manager Automation template.

   **Note**
   Choose **View** to view the template.

<table>
<thead>
<tr>
<th>View</th>
<th>Launch</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td></td>
</tr>
</tbody>
</table>

2. Choose **Next**.
3. On the **Specify Details** page, in the **Stack Name** field, either choose to keep the default value or specify your own value. Choose **Next**.
4. On the **Options** page, you don’t need to make any selections. Choose **Next**.
5. On the Review page, scroll down and choose the **I acknowledge that AWS CloudFormation might create IAM resources** option.

6. Choose Create.

AWS CloudFormation shows the **CREATE_IN_PROGRESS** status for approximately three minutes. The status changes to **CREATE_COMPLETE** after the stack has been created and your roles are ready to use.

**Copying Role Information for Automation**

Use the following procedure to copy information about the instance profile role and Automation service role from the AWS CloudFormation console. You must specify these roles when you run an Automation document.

**Note**

You do not need to copy role information using this procedure if you run the AWS-UpdateLinuxAmi or AWS-UpdateWindowsAmi documents. These documents already have the required roles specified as default values. The roles specified in these documents use IAM managed policies.

**To copy the role names**

2. Choose the check-box beside the Automation stack you created in the previous procedure.
3. Choose the **Resources** tab.
4. The **Resources** table includes three items in the **Logical ID** column: **AutomationServiceRole**, **ManagedInstanceProfile**, and **ManagedInstanceRole**.
5. Copy the **Physical ID** for **ManagedInstanceProfile**. The physical ID will be similar to **Automation-ManagedInstanceProfile-1a2b3c4**. This is the name of your instance profile role.
6. Paste the instance profile role into a text file to use later.
7. Choose the **Physical ID** link for **AutomationServiceRole**. The IAM console opens to a summary of the Automation Service Role.
8. Copy the Amazon Resource Name (ARN) beside **Role ARN**. The ARN is similar to the following: `arn:aws:iam::12345678:role/Automation-AutomationServiceRole-1A2B3C4D5E`
9. Paste the ARN into a text file to use later.

You have finished configuring the required roles for Automation. You can now use the instance profile role and the Automation service role ARN in your Automation documents. For more information, see [Automation Console Walkthrough: Patch a Linux AMI](p. 103) and [Automation CLI Walkthrough: Patch a Linux AMI](p. 105).

**Method 2: Using IAM to Configure Roles for Automation**

Configuring access to Systems Manager Automation requires that you complete the following tasks.

1. **Verify user access**: Verify that you have permission to run Automation workflows. If your AWS Identity and Access Management (IAM) user account, group, or role is assigned administrator permissions, then you have access to Systems Manager Automation. If you don't have administrator permissions, then an administrator must give you permission by assigning the **AmazonSSMFullAccess** managed policy, or a policy that provides comparable permissions, to your IAM account, group, or role.

2. **Configure instance access by creating and assigning an instance profile role (Required)**: Each instance that runs an Automation workflow requires an IAM instance profile role. This role gives Automation permission to perform actions on your instances, such as executing commands or starting and stopping services. If you previously created an instance profile role for Systems Manager, as described in [Task 2: Create an Instance Profile Role for Systems Manager](p. 8) in the **Configuring**
Access to Systems Manager topic, then you can use this same instance profile role for Automation. For information about how to attach this role to an existing instance, see Attaching an IAM Role to an Instance in the Amazon EC2 User Guide.

**Note**  
Automation previously required that you specify a service role (or assume role) so that the service had permission to perform actions on your behalf. Automation no longer requires this role because the service now operates by using the context of the user who invoked the execution.  
However, the following situations still require that you specify a service role for Automation:

- When you want to restrict a user's privileges on a resource, but you want the user to execute an Automation workflow that requires higher privileges. In this scenario, you can create a service role with higher privileges and allow the user to execute the workflow.
- Operations that you expect to run longer than 12 hours require a service role.

If you need to create an instance profile role and a service role for Systems Manager Automation, complete the following tasks.

**Tasks**
- Task 1: Create a Service Role for Automation (p. 99)
- Task 2: Add a Trust Relationship for Automation (p. 100)
- Task 3: Attach the iam:PassRole Policy to Your Automation Role (p. 100)
- Task 4: Configure User Access to Automation (p. 101)
- Task 5: Create an Instance Profile Role (p. 101)

**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 these roles and their Amazon Resource Names (ARNs) in Automation documents, such as the AWS-UpdateLinuxAmi document. Using these roles or their ARNs in Automation documents enables Automation to perform actions on your managed instances, launch new instances, and perform actions on your behalf. To view an example, see Automation CLI Walkthrough: Patch a Linux AMI (p. 105).

**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. On the Select type of trusted entity page, under AWS Service, choose EC2, and then choose Next: Permissions.
4. On the Attached permissions policy page, search for the AmazonSSMAutomationRole policy, choose it, and then choose Next: Review.
5. On the Review page, type a name in the Role name box, and then type a description.
6. Choose Create role. The system returns you to the Roles page.
7. On the Roles page, choose the role you just created to open the Summary page. Make a note of the Role Name and Role ARN. You will specify the role ARN when you attach the iam:PassRole policy to your IAM account in the next procedure. You can also specify the role name and the ARN in Automation documents.

Leave the Summary page open.
Note
The AmazonSSMAutomationRole policy assigns the Automation role permission to a subset of AWS Lambda functions within your account. These functions begin with "Automation". If you plan to use Automation with Lambda functions, the Lambda ARN must use the following format:

"arn:aws:lambda::*::*:function:Automation*"

If you have existing Lambda functions whose ARNs do not use this format, then you must also attach an additional Lambda policy to your automation role, such as the AWSLambdaRole policy. The additional policy or role must provide broader access to Lambda functions within the AWS account.

Task 2: Add a Trust Relationship for Automation

Use the following procedure to configure the service role policy to trust Automation.

To add a trust relationship for Automation

1. In the Summary page for the role you just created, choose the Trust Relationships tab, and then choose Edit Trust Relationship.
2. Add "ssm.amazonaws.com", as shown in the following example:

```json
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":[
               "ec2.amazonaws.com",
               "ssm.amazonaws.com"
            ],
            "Action":"sts:AssumeRole"
         }
      }
   ]
}
```
3. Choose Update Trust Policy.
4. Leave the Summary page open.

Task 3: Attach the iam:PassRole Policy to Your Automation Role

Use the following procedure to attach the iam:PassRole policy to your Automation service role. This enables the Automation service to pass the role to other services or Systems Manager capabilities when running Automation workflows.

To attach the iam:PassRole policy to your Automation role

1. In the Summary page for the role you just created, choose the Permissions tab.
2. Choose Add inline policy.
3. On the Set Permissions page, choose Policy Generator, and then choose Select.
4. Verify that Effect is set to Allow.
5. From AWS Services, choose AWS Identity and Access Management.
6. From Actions, choose PassRole.
7. In the Amazon Resource Name (ARN) field, paste the Automation role ARN that you copied at the end of Task 1.
8. Choose Add Statement, and then choose Next Step.

Task 4: Configure User Access to Automation

If your AWS Identity and Access Management (IAM) user account, group, or role is assigned administrator permissions, then you have access to Systems Manager Automation. If you don’t have administrator permissions, then an administrator must give you permission by assigning the AmazonSSMFullAccess managed policy, or a policy that provides comparable permissions, to your IAM account, group, or role.

Use the following procedure to configure a user account to use Automation. The user account you choose will have permission to configure and execute Automation. If you need to create a new user account, see Creating an IAM User in Your AWS Account in the IAM User Guide.

To configure user access and attach the iam:PassRole policy to a user account

1. In the IAM navigation pane, choose Users, and then choose the user account you want to configure.
2. On the Permissions tab, in the policies list, verify that either the AmazonSSMFullAccess policy is listed or there is a comparable policy that gives the account permissions to access Systems Manager.
3. Choose Add inline policy.
4. On the Set Permissions page, choose Policy Generator, and then choose Select.
5. Verify that Effect is set to Allow.
6. From AWS Services, choose AWS Identity and Access Management.
7. From Actions, choose PassRole.
8. In the Amazon Resource Name (ARN) field, paste the ARN for the Automation service role you copied at the end of Task 1.
9. Choose Add Statement, and then choose Next Step.

Task 5: Create an Instance Profile Role

Each instance that runs an Automation workflow requires an IAM instance profile role. This role gives Automation permission to perform actions on your instances, such as executing commands or starting and stopping services. If you previously created an instance profile role for Systems Manager, as described in Task 2: Create an Instance Profile Role for Systems Manager (p. 8) in the Configuring Access to Systems Manager topic, then you can use this same instance profile role for Automation. If you have not created an instance profile role as described in that topic, please do so now. For information about how to attach this role to an existing instance, see Attaching an IAM Role to an Instance in the Amazon EC2 User Guide.

You have finished configuring the required roles for Automation. You can now use the instance profile role and the Automation service role ARN in your Automation documents. For more information, see Automation Console Walkthrough: Patch a Linux AMI (p. 103) and Automation CLI Walkthrough: Patch a Linux AMI (p. 105).

Configuring CloudWatch Events for Systems Manager Automation (Optional)

You can configure Amazon CloudWatch Events to notify you of Systems Manager Automation events. For example, you can configure CloudWatch Events to send notifications when an Automation step succeeds.
or fails. You can also configure CloudWatch Events to send notifications if the Automation workflow succeeds or fails. Use the following procedure to configure CloudWatch Events to send notification about Automation events.

**To configure CloudWatch Events for Automation**

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Choose Events in the left navigation, and then choose Create rule.
3. Under Event Source, verify that Event Pattern is selected.
4. In the Service Name field, choose EC2 Simple Systems Manager (SSM).
5. In the Event Type field, choose Automation.
6. Choose the detail types and statuses for which you want to receive notifications, and then choose Add targets.
7. In the Select target type list, choose a target type. For information about the different types of targets, see the corresponding AWS Help documentation.
8. Choose Configure details.
9. Specify the rule details, and then choose Create rule.

The next time you run Automation, CloudWatch Events sends event details to the target you specified.

**Configuring Automation as a CloudWatch Events Target (Optional)**

You can start an Automation workflow by specifying an Automation document as the target of an Amazon CloudWatch event. You can start workflows according to a schedule, or when a specific AWS system event occurs. For example, let’s say you create an Automation document named BootStrapInstances that installs software on an instance when an instance starts. To specify the BootStrapInstances document (and corresponding workflow) as a target of a CloudWatch event, you first create a new CloudWatch Events rule. (Here’s an example rule: Service name: EC2, Event Type: EC2 Instance State-change Notification, Specific state(s): running, Any instance.) Then you use the following procedure to specify the BootStrapInstances document as the target of the event. When a new instances starts, the system runs the workflow and installs software.

For information about creating Automation documents, see Working with Automation Documents (p. 107).

Use the following procedure to configure an Automation workflow as the target of a CloudWatch event.

**To configure Automation as a target of a CloudWatch event**

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose Events, and then choose Create rule.
3. Choose Event Pattern or Schedule. Event Pattern lets you build a rule that generates events for specific actions in AWS services. Schedule lets you build a rule that generates events according to a schedule that you specify by using the cron format.
4. Choose the remaining options for the rule you want to create, and then choose Add target.
5. In the Select target type list, choose SSM Automation.
6. In the Document list, choose an SSM document to run when your target is invoked.
7. Expand Configure document version, and choose a version. $DEFAULT was explicitly set as the default document in Systems Manager. You can choose a specific version, or use the latest version.
8. Expand **Configure automation parameter(s)**, and either keep the default parameter values (if available) or enter your own values.

**Note**
Required parameters have an asterisk (*) next to the parameter name. To create a target, you must specify a value for each required parameter. If you don’t, the system creates the rule, but it won't execute.

9. In the permissions section, choose an option. CloudWatch uses the role to start the Automation workflow.

10. Choose **Configure details** and complete the wizard.

### Systems Manager Automation Walkthroughs

The following walkthroughs help you get started with Systems Manager Automation using a predefined Automation document.

Before you begin, you must configure Automation roles and permissions. For more information, see [Setting Up Automation](p. 96). For information about creating a custom Automation document, see [Creating an Automation Document](p. 114).

**Warning**
If you create an AMI from a running instance, there is a risk that credentials, sensitive data, or other confidential information from the instance may be recorded to the new image. Use caution when creating AMIs.

#### Walkthroughs
- **Automation Console Walkthrough: Patch a Linux AMI** (p. 103)
- **Automation CLI Walkthrough: Patch a Linux AMI** (p. 105)

### Automation Console Walkthrough: Patch a Linux AMI

This Systems Manager Automation walkthrough shows you how to use the Amazon EC2 console and the Systems Manager AWS-UpdateLinuxAmi document to automatically patch a Linux AMI. You can update any of the following Linux versions using this walkthrough: Ubuntu, CentOS, RHEL, or Amazon Linux AMIs. The AWS-UpdateLinuxAmi document also automates the installation of additional site-specific packages and configurations.

The walkthrough shows you how to specify parameters for the AWS-UpdateLinuxAmi document at runtime. If you want to add steps to your automation or specify default values, you can use the AWS-UpdateLinuxAmi document as a template.

For more information about working with Systems Manager documents, see [AWS Systems Manager Documents](p. 258). For information about actions you can add to a document, see [Systems Manager Automation Document Reference](p. 305).

When you run the AWS-UpdateLinuxAmi document, Automation performs the following tasks.

1. Launches a temporary Amazon EC2 instance from a Linux AMI. The instance is configured with a User Data script that installs the SSM Agent. The SSM Agent executes scripts sent remotely from Systems Manager Run Command.

2. Updates the Instance by performing the following actions:
   a. Invokes a user-provided pre-update script on the instance.
   b. Updates AWS tools on the instance, if any tools are present.
   c. Updates distribution packages on the instance by using the native package manager.
   d. Invokes a user-provided post-update script on the instance.
3. Stops the temporary instance.
4. Creates a new AMI from the stopped instance.
5. Terminates the instance.

After Automation successfully completes this workflow, the new AMI is available in the console on the AMIs page.

**Important**
If you use Automation to create an AMI from an instance, be aware that credentials, passwords, data, or other confidential information on the instance are recorded on the new image. Use caution when creating an AMI from an instance.

As you get started with Automation, note the following restrictions.

- Automation does not perform resource clean-up. In the event your workflow stops before reaching the final instance-termination step in the example workflow, you might need to stop instances manually or disable services that were started during the Automation workflow.
- If you use userdata with Automation, the userdata must be base-64 encoded.
- Automation retains execution records for 30 days.
- Systems Manager and Automation have the following service limits.

**Before you begin**

Create an AWS Identity and Access Management (IAM) instance profile role and Automation service role (or assume role). For more information about these roles and how to quickly create them from an AWS CloudFormation template, see Method 1: Using AWS CloudFormation to Configure Roles for Automation (p. 97).

We recommend that you also collect the following information before you begin.

- The source ID of the AMI to update.
- (Optional) The URL of a script to run before updates are applied.
- (Optional) The URL of a script to run after updates are applied.
- (Optional) The names of specific packages to update. By default, Automation updates all packages.
- (Optional) The names of specific packages to exclude from updating.

**Note**
By default, when Automation runs the AWS-UpdateLinuxAmi document, the system creates a temporary instance in the default VPC (172.30.0.0/16). If you deleted the default VPC, you will receive the following error:

```
VPC not defined 400
```

To solve this problem, you must make a copy of the AWS-UpdateLinuxAmi document and specify a subnet ID. For more information, see VPC not defined 400 (p. 150).

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

**To create a patched AMI using Automation**

1. Open the Amazon EC2 console, expand Systems Manager Services in the navigation pane, and then choose Automations.
2. Choose Run automation.
3. In the **Document name** list, choose AWS-UpdateLinuxAmi.
4. In the Version list, choose 1.
5. In the Input parameters section, enter the information you collected in the Before You Begin section.
6. Choose Run automation. The system displays an automation execution ID. Choose OK.
7. In the execution list, choose the execution you just ran and then choose the Steps tab. This tab shows you the status of the workflow actions. The update process can take 30 minutes or more to complete.

After the workflow finishes, launch a test instance from the updated AMI to verify changes.

**Note**
If any step in the workflow fails, information about the failure is listed on the Automation Executions page. The workflow is designed to terminate the temporary instance after successfully completing all tasks. If a step fails, the system might not terminate the instance. So if a step fails, manually terminate the temporary instance.

### Automation CLI Walkthrough: Patch a Linux AMI

This Systems Manager Automation walkthrough shows you how to use the AWS CLI and the Systems Manager AWS-UpdateLinuxAmi document to automatically patch a Linux AMI. You can update any of the following Linux versions using this walkthrough: Ubuntu, CentOS, RHEL, or Amazon Linux AMIs. The AWS-UpdateLinuxAmi document also automates the installation of additional site-specific packages and configurations.

When you run the AWS-UpdateLinuxAmi document, Automation performs the following tasks.

1. Launches a temporary Amazon EC2 instance from a Linux AMI. The instance is configured with a User Data script that installs the SSM Agent. The SSM Agent executes scripts sent remotely from Systems Manager Run Command.
2. Updates the Instance by performing the following actions:
   a. Invokes a user-provided pre-update script on the instance.
   b. Updates AWS tools on the instance, if any tools are present.
   c. Updates distribution packages on the instance by using the native package manager.
   d. Invokes a user-provided post-update script on the instance.
3. Stops the temporary instance.
4. Creates a new AMI from the stopped instance.
5. Terminates the instance.

After Automation successfully completes this workflow, the new AMI is available in the console on the AMIs page.

**Important**
If you use Automation to create an AMI from an instance, be aware that credentials, passwords, data, or other confidential information on the instance are recorded on the new image. Use caution when creating an AMI from an instance.

As you get started with Automation, note the following restrictions.

- Automation does not perform resource clean-up. In the event your workflow stops before reaching the final instance-termination step in the example workflow, you might need to stop instances manually or disable services that were started during the Automation workflow.
- If you use userdata with Automation, the userdata must be base-64 encoded.
- Automation retains execution records for 30 days.
- Systems Manager and Automation have the following service limits.
Before you begin

Create an AWS Identity and Access Management (IAM) instance profile role and Automation service role (or assume role). For more information about these roles and how to quickly create them from an AWS CloudFormation template, see Method 1: Using AWS CloudFormation to Configure Roles for Automation (p. 97).

We recommend that you also collect the source ID of the AMI to update.

**Note**
By default, when Automation runs the AWS-UpdateLinuxAmi document, the system creates a temporary instance in the default VPC (172.30.0.0/16). If you deleted the default VPC, you will receive the following error:
VPC not defined 400
To solve this problem, you must make a copy of the AWS-UpdateLinuxAmi document and specify a subnet ID. For more information, see VPC not defined 400 (p. 150).

To create a patched AMI using Automation

1. **Download** the AWS CLI to your local machine.
2. Execute the following command to run the AWS-UpdateLinuxAmi document and run the Automation workflow. In the parameters section, specify your Automation role, an AMI source ID, and an Amazon EC2 instance profile role.

   ```bash
   aws ssm start-automation-execution \
   --document-name "AWS-UpdateLinuxAmi" \
   --parameters \
   SourceAmiId=ami-e6d5d2f1
   ```

   The command returns an execution ID. Copy this ID to the clipboard. You will use this ID to view the status of the workflow.

   ```json
   {
     "AutomationExecutionId": "ID"
   }
   ```

3. To view the workflow execution using the CLI, execute the following command:

   ```bash
   aws ssm describe-automation-executions
   ```

4. To view details about the execution progress, execute the following command.

   ```bash
   aws ssm get-automation-execution --automation-execution-id ID
   ```

   The update process can take 30 minutes or more to complete.

   **Note**
   You can also monitor the status of the workflow in the console. In the execution list, choose the execution you just ran and then choose the **Steps** tab. This tab shows you the status of the workflow actions.

After the workflow finishes, launch a test instance from the updated AMI to verify changes.

**Note**
If any step in the workflow fails, information about the failure is listed on the Automation Executions page. The workflow is designed to terminate the temporary instance after successfully completing all tasks. If a step fails, the system might not terminate the instance. So if a step fails, manually terminate the temporary instance.
**Additional Automation CLI Examples**

You can manage other aspects of Automation execution using the following tasks.

**Stop an Execution**

Execute the following to stop a workflow. The command doesn't terminate associated instances.

```
aws ssm stop-automation-execution --automation-execution-id ID
```

**Create Versions of Automation Documents**

You can't change an existing automation document, but you can create a new version using the following command:

```
aws ssm update-document --name "patchWindowsAmi" --content file:///Users/test-user/Documents/patchWindowsAmi.json --document-version "\$LATEST"
```

Execute the following command to view details about the existing document versions:

```
aws ssm list-document-versions --name "patchWindowsAmi"
```

The command returns information like the following:

```
{
   "DocumentVersions": [
   {
      "IsDefaultVersion": false,
      "Name": "patchWindowsAmi",
      "DocumentVersion": "2",
      "CreatedDate": 1475799950.484
   },
   {
      "IsDefaultVersion": false,
      "Name": "patchWindowsAmi",
      "DocumentVersion": "1",
      "CreatedDate": 1475799931.064
   }
   ]
}
```

Execute the following command to update the default version for execution. The default execution version only changes when you explicitly set it to a new version. Creating a new document version does not change the default version.

```
aws ssm update-document-default-version --name patchWindowsAmi --document-version 2
```

**Delete a Document**

Execute the following command to delete an automation document:

```
aws ssm delete-document --name patchWindowsAMI
```

**Working with Automation Documents**

A Systems Manager Automation document defines the actions that Systems Manager performs on your managed instances and AWS resources. Documents use JavaScript Object Notation (JSON) or YAML, and they include steps and parameters that you specify. Steps execute in sequential order.
Automation documents are Systems Manager documents of type Automation, as opposed to Command and Policy documents. Automation documents currently support schema version 0.3. Command and Policy documents use schema version 1.2 or 2.0.

**Note**
To view information about the actions or plugins that you can specify in a Systems Manager Automation document, see Systems Manager Automation Document Reference (p. 305). To view information about the plugins for all other SSM documents, see SSM Document Plugin Reference (p. 284).

Contents
- Working with Predefined Automation Documents (p. 108)
- Creating an Automation Document (p. 114)

**Working with Predefined Automation Documents**

To help you get started quickly, Systems Manager provides the following pre-defined Automation documents. These documents are maintained by Amazon Web Services.

- Customizing and Updating Linux AMIs Using AWS-UpdateLinuxAmi (p. 108)
- Customizing and Updating Windows AMIs Using AWS-UpdateWindowsAmi (p. 111)

**Customizing and Updating Linux AMIs Using AWS-UpdateLinuxAmi**

The AWS-UpdateLinuxAmi document enables you to automate image-maintenance tasks without having to author the workflow in JSON or YAML. You can use the AWS-UpdateLinuxAmi document to perform the following types of tasks.

- Upgrade all distribution packages and Amazon software on an Amazon Linux, Red Hat, Ubuntu, or CentOS Amazon Machine Image (AMI). This is the default document behavior.
- Install the SSM Agent on an existing image to enable Systems Manager capabilities, such as remote command execution using Run Command or software inventory collection using Inventory.
- Install additional software packages.

You can view the JSON or YAML for this document in the Amazon EC2 console. Expand Systems Manager Shared Resources, and then choose Documents. Choose the option beside the AWS-UpdateLinuxAmi document, and then use the tabs in the lower pane to view the JSON or YAML and other information about the document, as shown in the following image.
Before You Begin

Before you begin working with Automation documents, configure roles and, optionally, CloudWatch Events for Automation. For more information, see Setting Up Automation (p. 96).

The AWS-UpdateLinuxAmi document accepts the following input parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceAmiId</td>
<td>String</td>
<td>(Required) The source AMI ID.</td>
</tr>
<tr>
<td>InstanceRole</td>
<td>String</td>
<td>(Optional) The name of the AWS Identity and Access Management (IAM) instance profile role you created in Setting Up Automation (p. 96). The instance profile role gives Automation permission to perform actions on your instances, such as executing commands or starting and stopping services. The Automation document uses only the name of the instance profile role. If you specify the Amazon Resource Name (ARN), the Automation execution fails.</td>
</tr>
<tr>
<td>AutomationAssumeRole</td>
<td>String</td>
<td>(Optional) The name of the IAM service role you created in Setting Up Automation (p. 96). 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 executing the aws:createImage action in an Automation document. For this parameter, the complete ARN must be specified.</td>
</tr>
<tr>
<td>TargetAmiName</td>
<td>String</td>
<td>(Optional) The name of the new AMI after it is created. The default name is a system-generated string that includes the source AMI ID, and the creation time and date.</td>
</tr>
<tr>
<td>InstanceType</td>
<td>String</td>
<td>(Optional) The type of instance to launch as the workspace host. Instance types vary by region. The default type is t2.micro.</td>
</tr>
<tr>
<td>PreUpdateScript</td>
<td>String</td>
<td>(Optional) URL of a script to run before updates are applied.</td>
</tr>
</tbody>
</table>


### Automation Steps

The AWS-UpdateLinuxAmi document includes the following Automation steps, by default.

**Step 1: launchInstance (aws:runInstances action)**

This step launches an instance using Amazon EC2 userdata and an IAM instance profile role. Userdata installs the appropriate SSM Agent, based on the operating system. Installing the SSM Agent enables you to utilize Systems Manager capabilities such as Run Command, State Manager, and Inventory.

**Step 2: updateOSSoftware (aws:runCommand action)**

This step executes the following commands on the launched instance:

- Downloads an update script from Amazon S3.
- Executes an optional pre-update script.
- Updates distribution packages and Amazon software.
- Executes an optional post-update script.

The execution log is stored in the /tmp folder for the user to view later.

If you want to upgrade a specific set of packages, you can supply the list using the IncludePackages parameter. When provided, the system attempts to update only these packages and their dependencies. No other updates are performed. By default, when no *include* packages are specified, the program updates all available packages.

If you want to exclude upgrading a specific set of packages, you can supply the list to the ExcludePackages parameter. If provided, these packages remain at their current version, independent of any other options specified. By default, when no *exclude* packages are specified, no packages are excluded.

**Step 3: stopInstance (aws:changeInstanceState action)**

This step stops the updated instance.

**Step 4: createImage (aws:createImage action)**

This step creates a new AMI with a descriptive name that links it to the source ID and creation time. For example: “AMI Generated by EC2 Automation on {{global:DATE_TIME}} from {{SourceAmiId}}” where DATE_TIME and SourceId represent Automation variables.
Step 5: terminateInstance (aws:changeInstanceState action)

This step cleans up the execution by terminating the running instance.

Output

The execution returns the new AMI ID as output.

You can use the AWS-UpdateLinuxAmi document as a template to create your own document, as described in the next section. For information about actions (steps) that are supported in Automation documents, see Systems Manager Automation Document Reference (p. 305). For information about how to use Automation documents, see Systems Manager Automation Walkthroughs (p. 103).

Customizing and Updating Windows AMIs Using AWS-UpdateWindowsAmi

The AWS-UpdateWindowsAmi document enables you to automate image-maintenance tasks on your Amazon Windows AMIs without having to author the workflow in JSON or YAML. This document is supported for Windows Server 2008 R2 or later. You can use the AWS-UpdateWindowsAmi document to perform the following types of tasks.

- Install all Windows updates and upgrade Amazon software (default behavior).
- Install specific Windows updates and upgrade Amazon software.
- Customize an AMI using your scripts.

Before You Begin

Before you begin working with Automation documents, configure roles and, optionally, CloudWatch Events for Automation. For more information, see Setting Up Automation (p. 96).

Note

Updates to the SSM Agent are typically rolled out to different regions at different times. When you customize or update an AMI, use only source AMIs published for the region that you are working in. This will ensure that you are working with the latest SSM Agent released for that region and avoid compatibility issues.

The AWS-UpdateWindowsAmi document accepts the following input parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceAmiId</td>
<td>String</td>
<td>(Optional) The source AMI ID.</td>
</tr>
<tr>
<td>InstanceIamRole</td>
<td>String</td>
<td>(Optional) The name of the AWS Identity and Access Management (IAM) instance profile role you created in Setting Up Automation (p. 96). The instance profile role gives Automation permission to perform actions on your instances, such as executing commands or starting and stopping services. The Automation document uses only the name of the instance profile role. If you specify the Amazon Resource Name (ARN), the Automation execution fails.</td>
</tr>
<tr>
<td>InstancelamRole</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AutomationAssumeRole</td>
<td>String</td>
<td>(Optional) The name of the IAM service role you created in Setting Up Automation (p. 96). 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 executing the <code>aws:createImage</code> action in an Automation document. For this parameter, the complete ARN must be specified.</td>
</tr>
<tr>
<td>TargetAmiName</td>
<td>String</td>
<td>(Optional) The name of the new AMI after it is created. The default name is a system-generated string that includes the source AMI ID, and the creation time and date.</td>
</tr>
<tr>
<td>InstanceType</td>
<td>String</td>
<td>(Optional) The type of instance to launch as the workspace host. Instance types vary by region. The default type is <code>t2.medium</code>.</td>
</tr>
<tr>
<td>PreUpdateScript</td>
<td>String</td>
<td>(Optional) A script to run before updating the AMI. Enter a script in the Automation document or at runtime as a parameter.</td>
</tr>
<tr>
<td>PostUpdateScript</td>
<td>String</td>
<td>(Optional) A script to run after updating the AMI. Enter a script in the Automation document or at runtime as a parameter.</td>
</tr>
<tr>
<td>IncludeKbs</td>
<td>String</td>
<td>(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.</td>
</tr>
<tr>
<td>ExcludeKbs</td>
<td>String</td>
<td>(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.</td>
</tr>
</tbody>
</table>
### Parameter Types and Descriptions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categories</td>
<td>String</td>
<td>(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 Definition Update.</td>
</tr>
<tr>
<td>SeverityLevels</td>
<td>String</td>
<td>(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.</td>
</tr>
</tbody>
</table>

### Automation Steps

The AWS-UpdateWindowsAmi document includes the following Automation steps, by default.

**Step 1: launchInstance (aws:runInstances action)**

This step launches an instance with an IAM instance profile role from the specified SourceAmiID.

**Step 2: runPreUpdateScript (aws:runCommand action)**

This step enables you to specify a script as a string that executes before updates are installed.

**Step 3: updateEC2Config (aws:runCommand action)**

This step uses the AWS-InstallPowerShellModule public document to download an AWS public PowerShell module. Systems Manager verifies the integrity of the module by using an SHA-256 hash. Systems Manager then checks the operating system to determine whether to update EC2Config or EC2Launch. EC2Config runs on Windows Server 2008 R2 through Windows Server 2012 R2. EC2Launch runs on Windows Server 2016.

**Step 4: updateSSMAgent (aws:runCommand action)**

This step updates the SSM Agent by using the AWS-UpdateSSMAgent public document.

**Step 5: updateAWSPVDriver (aws:runCommand action)**

This step updates AWS PV drivers by using the AWS-ConfigureAWSPackage public document.

**Step 6: installWindowsUpdates (aws:runCommand action)**

This step installs Windows updates by using the AWS-InstallWindowsUpdates public document. By default, Systems Manager searches for and installs all missing updates. You can change the default behavior by specifying one of the following parameters: IncludeKbs, ExcludeKbs, Categories, or SeverityLevels.
Step 7: **runPostUpdateScript (aws:runCommand action)**

This step enables you to specify a script as a string that executes after the updates have been installed.

**Step 8: runSysprepGeneralize (aws:runCommand action)**

This step uses the AWS-InstallPowerShellModule public document to download an AWS public PowerShell module. Systems Manager verifies the integrity of the module by using an SHA-256 hash. Systems Manager then executes sysprep using AWS-supported methods for either EC2Launch (Windows Server 2016) or EC2Config (Windows Server 2008 R2 through 2012 R2).

**Step 9: stopInstance (aws:changeInstanceState action)**

This step stops the updated instance.

**Step 10: 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 11: TerminateInstance (aws:changeInstanceState action)**

This step cleans up the execution by terminating the running instance.

**Output**

This section enables you to designate the outputs of various steps or values of any parameter as the Automation output. By default, the output is the ID of the updated Windows AMI created by the execution.

You can use the AWS-UpdateWindowsAmi document as a template to create your own document, as described in the next section. For information about actions (steps) that are supported in Automation documents, see Systems Manager Automation Document Reference (p. 305). For information about how to use Automation documents, see Systems Manager Automation Walkthroughs (p. 103)

**Creating an Automation Document**

This walkthrough shows you how to create and execute a custom Automation document. After you run Automation, the system performs the following tasks.

- Launches a Windows instance from a specified AMI.
- Executes a command using Run Command that applies Windows updates to the instance.
- Stops the instance.
- Creates a new Windows AMI.
- Tag the Windows AMI.
- Terminates the original instance.

**Automation Sample Document**

Automation executes Systems Manager automation documents written in JSON or YAML. Automation documents include the actions to be performed during workflow execution. For more information about Systems Manager documents, see AWS Systems Manager Documents (p. 258). For information about actions you can add to a document, see Systems Manager Automation Document Reference (p. 305)

**Note**

By default, when Automation runs the AWS-UpdateWindowsAmi document and creates a temporary instance, the system uses the default VPC (172.30.0.0/16). If you deleted the default VPC, you will receive the following error:
VPC not defined 400
To solve this problem, you must make a copy of the AWS-UpdateWindowsAmi document and specify a subnet ID. For more information, see VPC not defined 400 (p. 150).

To create a patched AMI using Automation

1. Collect the following information. You will specify this information later in this procedure.
   - The source ID of the AMI to update.
   - Create an AWS Identity and Access Management (IAM) instance profile role and Automation service role (or assume role). For more information about these roles and how to quickly create them from an AWS CloudFormation template, see Method 1: Using AWS CloudFormation to Configure Roles for Automation (p. 97). Be sure to copy the name of the instance profile role and the Amazon Resource Name (ARN) of the Automation service role, as described in Copying Role Information for Automation (p. 98).

2. Copy the following example document into a text editor such as Notepad. Change the value of assumeRole to the role ARN you created earlier when you created an IAM role for Automation and change the value of IamInstanceProfileName to the name of the role you created earlier. Save the document on a local drive as patchWindowsAmi.json or patchWindowsAmi.yaml.

```json
{
    "description":"Systems Manager Automation Demo - Patch and Create a New AMI",
    "schemaVersion":"0.3",
    "assumeRole":"the role ARN you created",
    "parameters":{
        "sourceAMIid":{
            "type":"String",
            "description":"AMI to patch"
        },
        "targetAMIname":{
            "type":"String",
            "description":"Name of new AMI",
            "default":"patchedAMI-{{global:DATE_TIME}}"
        }
    },
    "mainSteps":[
        {
            "name":"startInstances",
            "action":"aws:runInstances",
            "timeoutSeconds":1200,
            "maxAttempts":1,
            "onFailure":"Abort",
            "inputs":{
                "ImageId":{{ sourceAMIid }}",
                "InstanceType":"m3.large",
                "MinInstanceCount":1,
                "MaxInstanceCount":1,
                "IamInstanceProfileName":"the name of the IAM role you created"
            }
        },
        {
            "name":"installMissingWindowsUpdates",
            "action":"aws:runCommand",
            "maxAttempts":1,
            "onFailure":"Continue",
            "inputs":{
                "DocumentName":"AWS-InstallMissingWindowsUpdates",
                "InstanceId":{
                    "{{ startInstances.InstanceIds }}"
                },
                "Parameters":{
                    "UpdateLevel":"Important"
                }
            }
        }
    ]
}
```
```json
{
  "name": "stopInstance",
  "action": "aws:changeInstanceState",
  "maxAttempts": 1,
  "onFailure": "Continue",
  "inputs": {
    "InstanceIds": [
      "{{ startInstances.InstanceIds }}"
    ],
    "DesiredState": "stopped"
  }
},
{
  "name": "createImage",
  "action": "aws:createImage",
  "maxAttempts": 1,
  "onFailure": "Continue",
  "inputs": {
    "InstanceId": "{{ startInstances.InstanceIds }}"
  },
  "ImageName": "{{ targetAMIname }}",
  "NoReboot": true,
  "ImageDescription": "AMI created by EC2 Automation"
},
{
  "name": "createTags",
  "action": "aws:createTags",
  "maxAttempts": 1,
  "onFailure": "Continue",
  "inputs": {
    "ResourceType": "EC2",
    "ResourceIds": [
      "{{createImage.ImageId}}"
    ],
    "Tags": [
      {
        "Key": "Generated By Automation",
        "Value": "{{automation:EXECUTION_ID}}"
      },
      {
        "Key": "From Source AMI",
        "Value": "{{sourceAMIid}}"
      }
    ]
  }
},
{
  "name": "terminateInstance",
  "action": "aws:changeInstanceState",
  "maxAttempts": 1,
  "onFailure": "Continue",
  "inputs": {
    "InstanceIds": [
      "{{ startInstances.InstanceIds }}"
    ],
    "DesiredState": "terminated"
  }
},
"outputs": [
  "createImage.ImageId"
]
```
3. **Download** the AWS CLI to your local machine.
4. Edit the following command, and specify the path to the `patchWindowsAmi.json/yaml` file on your local machine. Execute the command to create the required Automation document.

   **Note**
   For the `name` parameter, you can't prefix documents with AWS. If you specify `AWS-name` or `AWSname`, you will receive an error.

   ```shell
   aws ssm create-document --name "patchWindowsAmi" --content file:///Users/test-user/Documents/patchWindowsAmi.json/yaml --document-type Automation
   ``

   The system returns information about the command progress.

   ```json
   {
     "DocumentDescription": {
       "Status": "Creating",
       "Hash": "bce98f80b89668b092cd094d2f2895f57e40942bcc1598d85338dc9516b0b7f1",
       "Name": "test",
       "Parameters": [
         {
           "Type": "String",
           "Name": "sourceAMIid",
           "Description": "AMI to patch"
         },
         {
           "DefaultValue": "patchedAMI-{{global:DATE_TIME}}",
           "Type": "String",
           "Name": "targetAMIname",
           "Description": "Name of new AMI"
         }
       ],
       "DocumentType": "Automation",
       "PlatformTypes": ["Windows", "Linux"],
       "DocumentVersion": "1",
       "HashType": "Sha256",
       "CreatedDate": 1488303738.572,
       "Owner": "12345678901",
       "SchemaVersion": "0.3",
       "DefaultVersion": "1",
       "LatestVersion": "1",
       "Description": "Systems Manager Automation Demo - Patch and Create a New AMI"
     }
   }
   ```

5. Execute the following command to view a list of documents that you can access.

   ```shell
   aws ssm list-documents --document-filter-list key=Owner,value=Self
   ``

   The system returns information like the following:

   ```json
   {
     "DocumentIdentifiers": [
       {
         "Name": "patchWindowsAmi",
         "PlatformTypes": [
           "Windows",
           "Linux"
         ]
       }
     ]
   }
   ```
6. Execute the following command to view details about the patchWindowsAmi document.

```
aws ssm describe-document --name patchWindowsAmi
```

The system returns information like the following:

```
{
   "Document": {
      "Status": "Active",
      "Hash": "99d5b2e3571a16bb52c629283bca01a14026cd201876adf0a76de16766fb98ac",
      "Name": "patchWindowsAmi",
      "Parameters": [
         {
            "DefaultValue": "ami-3f0c4628",
            "Type": "String",
            "Name": "sourceAMIid",
            "Description": "AMI to patch"
         },
         {
            "DefaultValue": "patchedAMI-{{global:DATE_TIME}}",
            "Type": "String",
            "Name": "targetAMIname",
            "Description": "Name of new AMI"
         }
      ],
      "DocumentType": "Automation",
      "PlatformTypes": [
      ],
      "DocumentVersion": "5",
      "HashType": "Sha256",
      "CreatedDate": 1478904417.477,
      "Owner": "12345678901",
      "SchemaVersion": "0.3",
      "DefaultVersion": "5",
      "LatestVersion": "5",
      "Description": "Automation Demo - Patch and Create a New AMI"
   }
}
```

7. Execute the following command to run the patchWindowsAmi document and run the Automation workflow. This command takes two input parameters: the ID of the AMI to be patched, and the name of the new AMI. The example command below uses a recent 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="patchWindowsAmi" --parameters
sourceAMIid=ami-bd3ba0aa
```

The command returns an execution ID. Copy this ID to the clipboard. You will use this ID to view the status of the workflow.
You can monitor the status of the workflow in the console. Check the console to verify that a new instance is launching. After the instance launch is complete, you can confirm that the Run Command action was executed. After Run Command execution is complete, you should see a new AMI in your list of AMI images.

8. To view the workflow execution using the CLI, execute the following command:

```bash
aws ssm describe-automation-executions
```

9. To view details about the execution progress, execute the following command.

```bash
aws ssm get-automation-execution --automation-execution-id ID
```

**Note**
Depending on the number of patches applied, the Windows patching process executed in this sample workflow can take 30 minutes or more to complete.

For more examples of how to use Automation, including examples that build on the walkthrough you just completed, see Systems Manager Automation Examples (p. 119).

**Systems Manager Automation Examples**

The following are examples of how to use Systems Manager Automation to simplify common instance and system maintenance tasks. Note that some of these examples expand on the example of how to update a Windows AMI, which is described in Creating an Automation Document (p. 114).

**Examples**

- Run the EC2Rescue Tool on Unreachable Instances (p. 119)
- Reset the Local Administrator Password on Amazon EC2 Windows Instances (p. 123)
- Simplify AMI Patching Using Automation, Lambda, and Parameter Store (p. 128)
- Using Automation with Jenkins (p. 133)
- Patch an AMI and Update an Auto Scaling Group (p. 135)

**Run the EC2Rescue Tool on Unreachable Instances**

EC2Rescue can help you diagnose and troubleshoot problems on Amazon EC2 Windows Server instances. You can run the tool manually, as described in Using EC2Rescue for Windows Server, or you can run the tool automatically by using Systems Manager Automation and the `AWSSupport-ExecuteEC2Rescue` document. The `AWSSupport-ExecuteEC2Rescue` document is designed to perform a combination of Systems Manager actions, AWS CloudFormation actions, and Lambda functions that automate the steps normally required to use EC2Rescue.

**Note**
The `AWSSupport-ExecuteEC2Rescue` document is currently not supported for Linux instances. If you execute the automation on a Linux instance, the automation fails without making changes to your instance.

You can use Automation with the `AWSSupport-ExecuteEC2Rescue` document to troubleshoot and potentially remediate the following types of problems:
• Misconfigured network adapter (for example, incorrect static IP, DHCP disabled, and DHCP client disabled)
• Issues with the Remote Desktop Protocol (RDP) service (for example, when the service is disabled or configured with a non-default configuration)
• Issues with Windows Firewall (for example, when the firewall is blocking RDP traffic)

How It Works

Troubleshooting an instance with Automation and the **AWSSupport-ExecuteEC2Rescue** document works as follows:

• You specify the ID of the unreachable instance and execute the Automation workflow.
• The system creates a temporary VPC, and then executes 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 Windows Server 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 execute the following Automation, do the following:

• Copy the instance ID of the unreachable instance. You will specify this ID in the procedure.
• Optionally, collect the ID of a subnet in the same availability zone as your unreachable instance. The EC2Rescue instance will be created in this subnet. If you don't specify a subnet, then Automation creates a new temporary VPC in your AWS account. Verify that your AWS account has at least one VPC available. By default, you can create five VPCs in a Region. If you already created five VPCs in the Region, the automation fails without making changes to your instance. For more information, see VPC and Subnets.
• Optionally, you can create and specify an AWS Identity and Access Management (IAM) role for Automation. If you don't specify this role, then Automation runs in the context of the user who executed the automation. For more information about creating roles for Automation, see QuickStart #2: Executing an Automation Workflow by Using an IAM Service Role (p. 95).

Granting **AWSSupport-EC2Rescue** Permissions to Perform Actions On Your Instances

EC2Rescue needs permission to perform a series of actions on your instances during the Automation execution. These actions invoke the AWS Lambda, IAM, and Amazon EC2 services to safely and securely attempt to remediate issues with your instances. If you have Administrator-level permissions in your AWS account and/or VPC, you might be able to execute the automation without configuring permissions, as described in this section. If you don't have Administrator-level permissions, then you or an administrator must configure permissions by using one of the following options.

• Granting Permissions By Using IAM Policies (p. 121)
• Granting Permissions By Using An AWS CloudFormation Template (p. 122)
Granting Permissions By Using IAM Policies

You can either attach the following IAM policy to your IAM user account, group, or role as an inline policy; or, you can create a new IAM managed policy and attach it to your user account, group, or role. For more information about adding an inline policy to your user account, group, or role see Working With Inline Policies. For more information about creating a new managed policy, see Working With Managed Policies.

**Note**
If you create a new IAM managed policy, you must also attach the AmazonEC2RoleforSSM managed policy to it so that your instances can communicate with the Systems Manager API.

IAM Policy for AWSSupport-EC2Rescue

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunction"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::awssupport-ssm.*/*.template",
                "arn:aws:s3:::awssupport-ssm.*/*.zip"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:CreateRole",
                "iam:CreateInstanceProfile",
                "iam:GetRole",
                "iam:GetInstanceProfile",
                "iam:PutRolePolicy",
                "iam:DetachRolePolicy",
                "iam:AttachRolePolicy",
                "iam:PassRole",
                "iam:AddRoleToInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy",
                "iam:DeleteInstanceProfile"
            ],
            "Resource": [
                "arn:aws:iam::An-AWS-Account-ID:role/AWSSupport-EC2Rescue-*",
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:CreateFunction",
                "ec2:CreateVpc",
                "ec2:ModifyVpcAttribute",
            ]
        }
    ]
}
```
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. Choose the Launch Stack button. The button opens the AWS CloudFormation console and populates the Specify an Amazon S3 template URL field with the URL to the EC2Rescue template.

   **Note**
   Choose View to view the template.

2. Choose Next.

3. On the Specify Details page, in the Stack Name field, either choose to keep the default value or specify your own value. Choose Next.

4. On the Options page, you don’t need to make any selections. Choose Next.

5. On the Review page, scroll down and choose the I acknowledge that AWS CloudFormation might create IAM resources option.

6. Choose Create.

   AWS CloudFormation shows the CREATE_IN_PROGRESS status for approximately three minutes. The status changes to CREATE_COMPLETE after the stack has been created.

7. In the stack list, choose the option beside the stack you just created, and then choose the Outputs tab.

8. Copy the Value. The is the ARN of the AssumeRole. You will specify this ARN when you execute the Automation.
Executing the Automation

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

**Important**
The following Automation execution 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 execute the AWSSupport-ExecuteEC2Rescue Automation**

1. Open the Amazon EC2 console, expand Systems Manager Services in the navigation pane, and then choose Automations.
2. Choose Run Automation.
3. In the Document name section, choose Owned by Me or Amazon from the list.
4. In the documents list, choose AWSSupport-ExecuteEC2Rescue. The document owner is Amazon.
5. In the Input parameters section, specify the following parameters:
   a. For UnreachableInstanceId, specify the ID of the unreachable instance.
   b. For LogDestination, specify an Amazon S3 bucket if you want to collect operating system-level logs while troubleshooting your instance. Logs are automatically uploaded to the specified bucket.
   c. For EC2RescueInstanceType, specify an instance type for the EC2Rescue instance. The default instance type is t2.small.
   d. For SubnetId, specify a subnet in an existing VPC in the same availability zone as the unreachable instance. By default, Systems Manager creates a new VPC, but you can specify a subnet in an existing VPC if you want.
      + **Note**
      If you don’t see the option to specify a bucket or a subnet ID, verify that you are using the latest Default version of the document.
   e. For AssumeRole, if you created roles for this Automation by using the CloudFormation procedure described earlier in this topic, then specify the AssumeRole ARN that you copied from the CloudFormation console.
6. Choose Run Automation.
7. To monitor the execution progress, choose the running Automation, and then choose the Steps tab. When the execution is finished, choose the Descriptions tab, and then choose View output to view the results. To view the output of individual steps, choose the Steps tab, and then choose View Outputs beside a step.

The Automation creates a backup AMI as part of the workflow. All other resources created by the Automation workflow are automatically deleted, but this AMI remains in your account. The AMI is named using the following convention:

Backup AMI: AWSSupport-EC2Rescue:UnreachableInstanceId

You can locate this AMI in the Amazon EC2 console by searching on the Automation execution ID.

**Reset the Local Administrator Password on Amazon EC2 Windows Instances**

You can use the AWSSupport-ResetAccess document to automatically reenable local Administrator password generation on Amazon EC2 Windows instances. The AWSSupport-ResetAccess document
is designed to perform a combination of Systems Manager actions, AWS CloudFormation actions, and Lambda functions that automate the steps normally required to reset the local administrator password.

**Note**
The **AWSSupport-ResetAccess** document is currently not supported for Linux instances. If you execute the automation on a Linux instance, the automation fails without making changes to your instance.

You can use Automation with the **AWSSupport-ResetAccess** document to solve the following problems:

- You lost your EC2 key pair: and want to create a password-enabled AMI from your current instance, so that you can launch a new EC2 instance and select a key pair you own
- You lost your local Administrator password: you want to generate a new password you can decrypt with the current EC2 key pair.

**Note**
If your EC2 Windows instance is configured for Systems Manager, you can also reset your local Administrator password by using EC2Rescue and Run Command. For more information, see [Using EC2Rescue for Windows Server with Systems Manager Run Command](https://docs.aws.amazon.com/AmazonEC2/latest/UserGuide/Using_EC2Rescue_for_Windows.html) in the *Amazon EC2 User Guide for Windows Instances*.

**How It Works**

Troubleshooting an instance with Automation and the **AWSSupport-ResetAccess** document works as follows:

- You specify the ID of the instance and execute the Automation workflow.
- The system creates a temporary VPC, and then executes 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 Windows Server 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 enables password generation for the local Administrator by using EC2Config or EC2Launch on the attached, original root volume. 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.
- Your instance generates a new password you can decode from the EC2 console using the current key pair assigned to the instance.

**Before You Begin**

Before you execute the following Automation, do the following:

- Copy the instance ID of the instance on which you want to reset the Administrator password. You will specify this ID in the procedure.
- Optionally, collect the ID of a subnet in the same availability zone as your unreachable instance. The EC2Rescue instance will be created in this subnet. If you don't specify a subnet, then Automation
creates a new temporary VPC in your AWS account. Verify that your AWS account has at least one VPC available. By default, you can create five VPCs in a Region. If you already created five VPCs in the Region, the automation fails without making changes to your instance. For more information, see VPC and Subnets.

- Optionally, you can create and specify an AWS Identity and Access Management (IAM) role for Automation. If you don't specify this role, then Automation runs in the context of the user who executed the automation. For more information about creating roles for Automation, see QuickStart #2: Executing an Automation Workflow by Using an IAM Service Role (p. 95).

Granting AWSSupport-EC2Rescue Permissions to Perform Actions On Your Instances

EC2Rescue needs permission to perform a series of actions on your instances during the Automation execution. These actions invoke the AWS Lambda, IAM, and Amazon EC2 services to safely and securely attempt to remediate issues with your instances. If you have Administrator-level permissions in your AWS account and/or VPC, you might be able to execute the automation without configuring permissions, as described in this section. If you don't have Administrator-level permissions, then you or an administrator must configure permissions by using one of the following options.

- Granting Permissions By Using IAM Policies (p. 121)
- Granting Permissions By Using An AWS CloudFormation Template (p. 122)

Granting Permissions By Using IAM Policies

You can either attach the following IAM policy to your IAM user account, group, or role as an inline policy; or, you can create a new IAM managed policy and attach it to your user account, group, or role. For more information about adding an inline policy to your user account, group, or role see Working With Inline Policies. For more information about creating a new managed policy, see Working With Managed Policies.

**Note**

If you create a new IAM managed policy, you must also attach the AmazonEC2RoleforSSM managed policy to it so that your instances can communicate with the Systems Manager API.

**IAM Policy for AWSSupport-ResetAccess**

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lambda:InvokeFunction",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::awssupport-ssm.*/*.template",
        "arn:aws:s3:::awssupport-ssm.*/*.zip"
      ],
      "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. Choose the Launch Stack button. The button opens the AWS CloudFormation console and populates the Specify an Amazon S3 template URL field with the URL to the EC2Rescue template.

   **Note**
   Choose View to view the template.
### Automation Examples

<table>
<thead>
<tr>
<th>View</th>
<th>Launch</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td><img src="image" alt="Launch Stack" /></td>
</tr>
</tbody>
</table>

2. Choose **Next**.
3. On the **Specify Details** page, in the **Stack Name** field, either choose to keep the default value or specify your own value. Choose **Next**.
4. On the **Options** page, you don’t need to make any selections. Choose **Next**.
5. On the **Review** page, scroll down and choose the I **acknowledge that AWS CloudFormation might create IAM resources** option.
6. Choose **Create**.

AWS CloudFormation shows the **CREATE_IN_PROGRESS** status for approximately three minutes. The status changes to **CREATE_COMPLETE** after the stack has been created.

7. In the stack list, choose the option beside the stack you just created, and then choose the **Outputs** tab.
8. Copy the **Value**. The is the ARN of the AssumeRole. You will specify this ARN when you execute the Automation.

### Executing the Automation

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new **AWS Systems Manager console**. The steps in the new console will differ from the steps below.

The following procedure describes how to execute the **AWSSupport-ResetAccess** document by using the Amazon EC2 console.

**Important**

The following Automation execution stops the instance. Stopping the instance can result in lost data on attached instance store volumes (if present). Stopping the instance can also cause the public IP to change, if no Elastic IP is associated. To avoid these configuration changes, use Run Command to reset access. For more information, see Using EC2Rescue for Windows Server with Systems Manager Run Command in the **Amazon EC2 User Guide for Windows Instances**.

### To execute the AWSSupport-ResetAccess Automation

1. Open the **Amazon EC2 console**, expand **Systems Manager Services** in the navigation pane, and then choose **Automations**.
2. Choose **Run Automation**.
3. In the **Document name** section, choose **Owned by Me or Amazon** from the list.
4. In the documents list, choose **AWSSupport-ResetAccess**. The document owner is Amazon.
5. In the **Input parameters** section, specify the following parameters:

   a. For **InstanceID**, specify the ID of the unreachable instance.

   b. For **EC2RescueInstanceType**, specify an instance type for the EC2Rescue instance. The default instance type is t2.small.

   c. For **SubnetId**, specify a subnet in an existing VPC in the same availability zone as the instance you specified. By default, Systems Manager creates a new VPC, but you can specify a subnet in an existing VPC if you want.
**Note**  
If you don’t see the option to specify a subnet ID, verify that you are using the latest **Default** version of the document.

d. For **Assume Role**, if you created roles for this Automation by using the CloudFormation procedure described earlier in this topic, then specify the AssumeRole ARN that you copied from the CloudFormation console.

6. Choose **Run Automation**.

7. To monitor the execution progress, choose the running Automation, and then choose the **Steps** tab. When the execution is finished, choose the **Descriptions** tab, and then choose **View output** to view the results. To view the output of individual steps, choose the **Steps** tab, and then choose **View Outputs** beside a step.

The Automation creates a backup AMI and a password-enabled AMI as part of the workflow. All other resources created by the Automation workflow are automatically deleted, but these AMIs remain in your account. The AMIs are named using the following conventions:

- Backup AMI: AWSSupport-EC2Rescue:`InstanceId`
- Password-enabled AMI: AWSSupport-EC2Rescue: Password-enabled AMI from `InstanceId`

You can locate these AMIs by searching on the Automation execution ID.

**Simplify AMI Patching Using Automation, Lambda, and Parameter Store**

The following example expands on how to update a Windows AMI, as described in **Creating an Automation Document** (p. 114). This example uses the model where an organization maintains and periodically patches their own, proprietary AMIs rather than building from Amazon EC2 AMIs.

The following procedure shows how to automatically apply operating system (OS) patches to a Windows AMI that is already considered to be the most up-to-date or *latest* AMI. In the example, the default value of the parameter `SourceAmiId` is defined by a Systems Manager Parameter Store parameter called `latestAmi`. The value of `latestAmi` is updated by an AWS Lambda function invoked at the end of the Automation workflow. As a result of this Automation process, the time and effort spent patching AMIs is minimized because patching is always applied to the most up-to-date AMI.

**Before You Begin**

Configure Automation roles and, optionally, CloudWatch Events for Automation. For more information, see **Setting Up Automation** (p. 96).

**Contents**

- **Task 1: Create a Parameter in Systems Manager Parameter Store** (p. 128)
- **Task 2: Create an IAM Role for AWS Lambda** (p. 129)
- **Task 3: Create an AWS Lambda Function** (p. 129)
- **Task 4: Create an Automation Document and Patch the AMI** (p. 131)

**Task 1: Create a Parameter in Systems Manager Parameter Store**

Use the following procedure to create a parameter in Systems Manager Parameter Store. Parameter Store lets you reference parameters (called Systems Manager parameters) across Systems Manager features, including Run Command, State Manager, and Automation.
Note
The following procedure describes steps that you perform in the Amazon EC2 console. You
can also perform these steps in the new AWS Systems Manager console. The steps in the new
console will differ from the steps below.

To create a parameter using Parameter Store
1. Open the Amazon EC2 console, expand Systems Manager Shared Resources in the navigation pane,
and then choose Parameter Store.
2. Choose Create Parameter.
3. For Name, type latestAmi.
4. In the Description field, type a description that identifies this parameter’s use.
5. For Type, choose String.
6. In the Value field, enter a Windows AMI ID. For example: ami-188d6e0e.
7. Choose Create Parameter, and then choose OK.

Task 2: Create an IAM Role for AWS Lambda

Use the following procedure to create an IAM service role for AWS Lambda. This role includes the
AWSLambdaExecute and AmazonSSMFullAccess managed policies. These policies give Lambda
permission to update the value of the latestAmi parameter using a Lambda function and Systems
Manager.

To create an IAM service role for Lambda
1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles, and then choose Create New Role.
3. For Role name, type a role name that can help you identify the purpose of this role, for example,
lambda-ssm-role. Role names must be unique within your AWS account. After you type the name,
choose Next Step at the bottom of the page.

Note
Because various entities might reference the role, you cannot change the name of the role
after it has been created.

4. On the Select Role Type page, choose the AWS Service Roles section, and then choose AWS
Lambda.
5. On the Attach Policy page, choose AWSLambdaExecute and AmazonSSMFullAccess, and then
choose Next Step.
6. 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 a Lambda function.
4. On the Configure triggers page, choose Next.
5. On the **Configure function** page, type Automation-UpdateSsmParam in the **Name** field, and enter a description, if you want.

6. In the **Runtime list**, choose **Python 2.7**.

7. In the **Lambda function code** section, delete the pre-populated code in the field, and then paste the following code sample.

```python
from __future__ import print_function
import json
import boto3

print('Loading function')

# Updates an SSM parameter
# Expects parameterName, parameterValue
def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))

    # get SSM client
    client = boto3.client('ssm')

    # confirm parameter exists before updating it
    response = client.describe_parameters(
        Filters=[
            {'Key': 'Name',
             'Values': [ event['parameterName'] ]}
        ],
    )

    if not response['Parameters']:
        print('No such parameter')
        return 'SSM parameter not found.'

    # if parameter has a Description field, update it PLUS the Value
    if 'Description' in response['Parameters'][0]:
        description = response['Parameters'][0]['Description']
        response = client.put_parameter(
            Name=event['parameterName'],
            Value=event['parameterValue'],
            Description=description,
            Type='String',
            Overwrite=True
        )

    # otherwise just update Value
    else:
        response = client.put_parameter(
            Name=event['parameterName'],
            Value=event['parameterValue'],
            Type='String',
            Overwrite=True
        )

    responseString = 'Updated parameter %s with value %s.' % (event['parameterName'], event['parameterValue'])

    return responseString
```
8. In the **Lambda function handler and role** section, in the **Role** list, choose the service role for Lambda that you created in Task 2.

9. Choose **Next**, and then choose **Create function**.

10. To test the Lambda function, from the **Actions** menu, choose **Configure Test Event**.

11. Replace the existing text with the following JSON.

   ```json
   {
   "parameterName":"latestAmi",
   "parameterValue":"your AMI ID"
   }
   ```

12. Choose **Save and test**. The output should state that the parameter was successfully updated and include details about the update. For example, "Updated parameter latestAmi with value ami-123456".

**Task 4: Create an Automation Document and Patch the AMI**

Use the following procedure to create and run an Automation document that patches the AMI you specified for the **latestAmi** parameter. After the Automation workflow completes, the value of **latestAmi** is updated with the ID of the newly-patched AMI. Subsequent executions use the AMI created by the previous execution.

**Note**

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. In the navigation pane, choose **Documents**.

3. Choose **Create Document**.

4. In the **Name** field, type UpdateMyLatestWindowsAmi.

5. In the **Document Type** list, choose **Automation**.

6. Delete the brackets in the **Content** field, and then paste the following JSON sample document.

   ```json
   {  
   "description":"Systems Manager Automation Demo – Patch AMI and Update SSM Param",
   "schemaVersion":"0.3",
   "assumeRole":"the role ARN you created",
   "parameters":{
   "sourceAMIid":{
   "type":"String",
   "description":"AMI to patch",
   "default":"{{ssm:latestAmi}}"
   },
   "targetAMIname":{
   "type":"String",
   "description":"Name of new AMI",
   "default":"patchedAMI-{{global:DATE_TIME}}"
   }
   },
   "mainSteps":[
   
   ]
   }
   ```

You must change the values of **assumeRole** and **IamInstanceProfileName** in this sample with the service role ARN and instance profile role you created when Setting Up Automation (p. 96).
"name":"startInstances",
"action":"aws:runInstances",
"timeoutSeconds":1200,
"maxAttempts":1,
"onFailure":"Abort",
"inputs":{
  "ImageId":"{{ sourceAMIid }}",
  "InstanceType":"m3.large",
  "MinInstanceCount":1,
  "MaxInstanceCount":1,
  "IamInstanceProfileName":"the name of the IAM role you created"
},
{
"name":"installMissingWindowsUpdates",
"action":"aws:runCommand",
"maxAttempts":1,
"onFailure":"Continue",
"inputs":{
  "DocumentName":"AWS-InstallMissingWindowsUpdates",
  "InstanceIds":[
    "{{ startInstances.InstanceIds }}"
  ],
  "Parameters":{
    "UpdateLevel":"Important"
  }
},
{
"name":"stopInstance",
"action":"aws:changeInstanceState",
"maxAttempts":1,
"onFailure":"Continue",
"inputs":{
  "InstanceIds":[
    "{{ startInstances.InstanceIds }}"
  ],
  "DesiredState":"stopped"
},
{
"name":"createImage",
"action":"aws:createImage",
"maxAttempts":1,
"onFailure":"Continue",
"inputs":{
  "InstanceId":"{{ startInstances.InstanceIds }}",
  "ImageName":"{{ targetAMIname }}",
  "NoReboot":true,
  "ImageDescription":"AMI created by EC2 Automation"
},
{
"name":"terminateInstance",
"action":"aws:changeInstanceState",
"maxAttempts":1,
"onFailure":"Continue",
"inputs":{
  "InstanceIds":[
    "{{ startInstances.InstanceIds }}"
  ],
  "DesiredState":"terminated"
},
{
"name":"updateSsmParam",
"action":null,
"action":"aws:invokeLambdaFunction",
"timeoutSeconds":1200,
"maxAttempts":1,
"onFailure":"Abort",
"inputs":{
  "FunctionName":"Automation-UpdateSsmParam",
  "Payload":{"\"parameterName\":\"latestAmi\", \"parameterValue\": \"{{createImage.ImageId}}\"}
}
],
"outputs":[
  "createImage.ImageId"
]}

7. Choose Create Document to save the document.
8. Expand Systems Manager Services in the navigation pane, choose Automations, and then choose Run automation.
10. In the Version list, choose 1, and then choose Run automation.
11. After execution completes, in the Amazon EC2 console, choose Parameter Store and confirm that the new value for latestAmi matches the value returned by the Automation workflow. You can also verify the new AMI ID matches the Automation output in the AMIs section of the EC2 console.

Using Automation with Jenkins

If your organization uses Jenkins software in a CI/CD pipeline, you can add Automation as a post-build step to pre-install application releases into Amazon Machine Images (AMIs). You can also use the Jenkins scheduling feature to call Automation and create your own operating system (OS) patching cadence.

The example below shows how to invoke Automation from a Jenkins server that is running either on-premises or in Amazon EC2. For authentication, the Jenkins server uses AWS credentials based on an AWS Identity and Access Management (IAM) user that you create in the example. If your Jenkins server is running in Amazon EC2, you can also authenticate it using an IAM instance profile role.

Note
Be sure to follow Jenkins security best-practices when configuring your instance.

Before You Begin

Complete the following tasks before you configure Automation with Jenkins.

• Complete the Simplify AMI Patching Using Automation, Lambda, and Parameter Store (p. 128) example. The following example uses the UpdateMyLatestWindowsAmi automation document created in that example.

• Configure IAM roles for Automation. Systems Manager requires an instance profile role and a service role ARN to process Automation workflows. For more information, see Setting Up Automation (p. 96).

• After you configure IAM roles for Automation, use the following procedure to create an IAM user account for your Jenkins server. The Automation workflow uses the IAM user account's Access key and Secret key to authenticate the Jenkins server during execution.

To create a user account for the Jenkins server

1. From the Users page on the IAM console, choose Add User.
2. In the **Set user details** section, specify a user name (for example, *Jenkins*).
3. In the **Select AWS access type** section, choose **Programmatic Access**.
4. Choose **Next:Permissions**.
5. In the **Set permissions for** section, choose **Attach existing policies directly**.
6. In the filter field, type **AmazonSSMFullAccess**.
7. Choose the checkbox beside the policy, and then choose **Next:Review**.
8. Verify the details, and then choose **Create**.
9. Copy the Access and Secret keys to a text file. You will specify these credentials in the next procedure.

Use the following procedure to configure the AWS CLI on your Jenkins server.

**To configure the Jenkins server for Automation**

1. If it’s not already installed, download the AWS CLI to your Jenkins server. For more information, see [Installing the AWS Command Line Interface](#).
2. In a terminal window on your Jenkins server, execute the following commands to configure the AWS CLI.

   ```bash
   sudo su - jenkins
   aws configure
   ```
3. When prompted, enter the AWS Access key and Secret key you received when you created the Jenkins user in IAM. Specify a default region. For more information about configuring the AWS CLI see [Configuring the AWS Command Line Interface](#).

Use the following procedure to configure your Jenkins project to invoke Automation.

**To configure your Jenkins server to invoke Automation**

1. Open the Jenkins console in a web browser.
2. Choose the project that you want to configure with Automation, and then choose **Configure**.
3. On the **Build** tab, choose **Add Build Step**.
4. Choose **Execute shell** or **Execute Windows batch command** (depending on your operating system).
5. In the **Command** box, execute an AWS CLI command like the following:

   ```bash
   aws --region the region of your source AMI ssm start-automation-execution --document-name your document name --parameters parameters for the document
   ```

   The following example command uses the **UpdateMyLatestWindowsAmi** document and the Systems Manager Parameter `latestAmi` created in [Simplify AMI Patching Using Automation, Lambda, and Parameter Store](#) (p. 128):

   ```bash
   aws --region us-east-1 ssm start-automation-execution
   --document-name UpdateMyLatestWindowsAmi
   --parameters
   "sourceAMIid='{{ssm:latestAmi}}'"
   ```

   In Jenkins, the command looks like the example in the following screenshot.
6. In the Jenkins project, choose **Build Now**. Jenkins returns output similar to the following example.

```
Console Output
Started by user admin
Building in workspace /var/lib/jenkins/workspace/build AMI
[Build AMI] $ /bin/sh -c '/path/to/lambda/func.sh'
+ aws --region us-east-1 ssm start-automation-execution --document-name UpdateMyLatestWindowsAmi --parameters 'sourceAmiId='"{ssm:latestAmi}"'
       'AutomationExecutionId': "79affa1-f8e1-1a5-95f-3e9404a844f3"
Finished: SUCCESS
```

### Patch an AMI and Update an Auto Scaling Group

The following example builds on the **Simplify AMI Patching Using Automation, Lambda, and Parameter Store (p. 128)** example by adding a step that updates an Auto Scaling group with the newly-patched AMI. This approach ensures that new images are automatically made available to different computing environments that use Auto Scaling groups.

The final step of the Automation workflow in this example uses an AWS Lambda function to copy an existing launch configuration and set the AMI ID to the newly-patched AMI. The Auto Scaling group is then updated with the new launch configuration. In this type of Auto Scaling scenario, users could terminate existing instances in the Auto Scaling group to force a new instance to launch that uses the new image. Or, users could wait and allow scale-in or scale-out events to naturally launch newer instances.

**Before You Begin**

Complete the following tasks before you begin this example.

- Complete the **Simplify AMI Patching Using Automation, Lambda, and Parameter Store (p. 128)** example. The following example uses the **UpdateMyLatestWindowsAmi** Automation document created in that example.
- Configure IAM roles for Automation. Systems Manager requires an instance profile role and a service role ARN to process Automation workflows. For more information, see **Setting Up Automation (p. 96)**.
Task 1: Create an IAM Role for AWS Lambda

Use the following procedure to create an IAM service role for AWS Lambda. This role includes the `AWSLambdaExecute` and `AutoScalingFullAccess` managed policies. These policies give Lambda permission to create a new Auto Scaling group with the latest, patched AMI using a Lambda function.

**To create an IAM service role for Lambda**

2. In the navigation pane, choose Roles, and then choose Create New Role.
3. For Role name, type a role name that can help you identify the purpose of this role, for example, lambda-ssm-role. Role names must be unique within your AWS account. After you type the name, choose Next Step at the bottom of the page.
   
   **Note**  
   Because various entities might reference the role, you cannot change the name of the role after it has been created.
4. On the Select Role Type page, choose the AWS Service Roles section, and then choose AWS Lambda.
5. On the Attach Policy page, choose `AWSLambdaExecute` and `AutoScalingFullAccess`, and then choose Next Step.
6. Choose Create Role.

Task 2: Create an AWS Lambda Function

Use the following procedure to create a Lambda function that automatically creates a new Auto Scaling group with the latest, patched AMI.

**To create a Lambda function**

1. Sign in to the AWS Management Console and open the AWS Lambda console at [https://console.aws.amazon.com/lambda/](https://console.aws.amazon.com/lambda/).
2. Choose Create a Lambda function.
4. On the Configure triggers page, choose Next.
5. On the Configure function page, type Automation-UpdateAsg in the Name field, and enter a description, if you want.
6. In the Runtime list, choose Python 2.7.
7. In the Lambda function code section, delete the pre-populated code in the field, and then paste the following code sample.

```python
from __future__ import print_function
import json
import datetime
import time
import boto3
print('Loading function')

def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))
    
    # get autoscaling client
    client = boto3.client('autoscaling')
```
# get object for the ASG we're going to update, filter by name of target ASG
response = client.describe_auto_scaling_groups(AutoScalingGroupNames=[event['targetASG']])

if not response['AutoScalingGroups']:
    return 'No such ASG'

# get name of InstanceID in current ASG that we'll use to model new Launch Configuration after
sourceInstanceId = response.get('AutoScalingGroups')[0]['Instances'][0]['InstanceId']

# create LC using instance from target ASG as a template, only diff is the name of the new LC and new AMI
timeStamp = time.time()
timeStampString = datetime.datetime.fromtimestamp(timeStamp).strftime('%Y-%m-%d %H-%M-%S')
newLaunchConfigName = 'LC ' + event['newAmiID'] + ' ' + timeStampString
client.create_launch_configuration(
    InstanceId = sourceInstanceId,
    LaunchConfigurationName=newLaunchConfigName,
    ImageId= event['newAmiID'] )

# update ASG to use new LC
response = client.update_auto_scaling_group(AutoScalingGroupName = event['targetASG'],LaunchConfigurationName = newLaunchConfigName)

return 'Updated ASG `%s` with new launch configuration `%s` which includes AMI `%s`. % (event['targetASG'], newLaunchConfigName, event['newAmiID'])

8. In the **Lambda function handler and role** section, in the **Role** list, choose the service role for Lambda that you created in Task 1.
9. Choose **Next**, and then choose **Create function**.
10. To test the Lambda function, from the **Actions** menu, choose **Configure Test Event**.
11. Replace the existing text with the following JSON, and enter an AMI ID and Auto Scaling group.

```
{
    "newAmiID": "valid AMI ID",
    "targetASG": "name of your Auto Scaling group"
}
```
12. Choose **Save and test**. The output states that the Auto Scaling group was successfully updated with a new launch configuration.

**Task 3: Create an Automation Document, Patch the AMI, and Update the Auto Scaling Group**

Use the following procedure to create and run an Automation document that patches the AMI you specified for the `latestAmi` parameter. The Automation workflow then updates the Auto Scaling group to use the latest, patched AMI.

**Note**
The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new **AWS Systems Manager console**. The steps in the new console will differ from the steps below.

1. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).
2. In the navigation pane, choose **Documents**.
3. Choose **Create Document**.
4. In the **Name** field, type `PatchAmiandUpdateAsg`.
5. In the **Document Type** list, choose **Automation**.
6. Delete the brackets in the **Content** field, and then paste the following JSON sample document.

**Note**
You must change the values of `assumeRole` and `IamInstanceProfileName` in this sample with the service role ARN and instance profile role you created when Setting Up Automation (p. 96).

```
{
  "description": "Systems Manager Automation Demo - Patch AMI and Update ASG",
  "schemaVersion": "0.3",
  "assumeRole": "the service role ARN you created",
  "parameters": {
    "sourceAMIid": {
      "type": "String",
      "description": "AMI to patch"
    },
    "targetAMIName": {
      "type": "String",
      "description": "Name of new AMI",
      "default": "patchedAMI-{{global:DATE_TIME}}"
    },
    "targetASG": {
      "type": "String",
      "description": "Autosaling group to Update"
    }
  },
  "mainSteps": [
    {
      "name": "startInstances",
      "action": "aws:runInstances",
      "timeoutSeconds": 1200,
      "maxAttempts": 1,
      "onFailure": "Abort",
      "inputs": {
        "ImageId": "{{ sourceAMIid }}",
        "InstanceType": "m3.large",
        "MinInstanceCount": 1,
        "MaxInstanceCount": 1,
        "IamInstanceProfileName": "the name of the instance IAM role you created"
      }
    },
    {
      "name": "installMissingWindowsUpdates",
      "action": "aws:runCommand",
      "maxAttempts": 1,
      "onFailure": "Continue",
      "inputs": {
        "DocumentName": "AWS-InstallMissingWindowsUpdates",
        "InstanceIds": [
          "{{ startInstances.InstanceIds }}"
        ],
        "Parameters": {
          "UpdateLevel": "Important"
        }
      }
    },
    {
      "name": "stopInstance",
      "action": "aws:changeInstanceState",
      "maxAttempts": 1,
      "onFailure": "Continue",
      "inputs": {
        "InstanceId": "{{ startInstances.InstanceIds }}"
      }
    }
  ]
}
```
7. Choose Create Document to save the document.
8. Expand Systems Manager Services in the navigation pane, choose Automations, and then choose Run automation.
10. In the Version list, choose 1, and then choose Run automation.
11. Specify a Windows AMI ID for sourceAMIId and your Auto Scaling group name for targetASG.
12. Choose Run automation.
13. After execution completes, in the Amazon EC2 console, choose Auto Scaling, and then choose Launch Configurations. Verify that you see the new launch configuration, and that it uses the new AMI ID.
14. Choose Auto Scaling, and then choose Auto Scaling Groups. Verify that the Auto Scaling group uses the new launch configuration.
15. Terminate one or more instances in your Auto Scaling group. Replacement instances will be launched with the new AMI ID.

**Note**
You can further automate deployment of the new AMI by editing the Lambda function to gracefully terminate instances. You can also invoke your own Lambda function and utilize the ability of AWS CloudFormation to update Auto Scaling groups. For more information, see UpdatePolicy Attribute.

## Automation System Variables

Systems Manager Automation documents use the following variables. For an example of how these variables are used, view the JSON source of the AWS-UpdateWindowsAmi document.

**Note**
The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

### To view the JSON source of the AWS-UpdateWindowsAmi document

1. In the Amazon EC2 console, expand **Systems Manager Shared Resources**, and then choose **Documents**.
2. Choose **AWS-UpdateWindowsAmi**.
3. In the lower pane, choose the **Content** tab.

### System Variables

Automation documents currently support the following system variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>global:ACCOUNT_ID</td>
<td>The AWS account ID of the AWS Identity and Access Management (IAM) user or role in which Automation executes.</td>
</tr>
<tr>
<td>global:DATE</td>
<td>The date (at execution time) in the format yyyy-MM-dd.</td>
</tr>
<tr>
<td>global:DATE_TIME</td>
<td>The date and time (at execution time) in the format yyyy-MM-dd_HH.mm.ss.</td>
</tr>
<tr>
<td>global:REGION</td>
<td>The region which the document is executed in. For example, us-east-1.</td>
</tr>
</tbody>
</table>

### Automation Variables

Automation documents currently support the following automation variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>automation:EXECUTION_ID</td>
<td>The unique identifier assigned to the current automation execution. For example 1a2b3c-1a2b3c-1a2b3c-1a2b3c1a2b3c1a2b3c.</td>
</tr>
</tbody>
</table>
# Terminology

The following terms describe how variables and parameters are resolved.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant ARN</td>
<td>A valid ARN without variables</td>
<td>arn:aws:iam::123456789012:role/roleName</td>
</tr>
<tr>
<td>Document Parameter</td>
<td>A parameter defined at the document level for an Automation document (for example, instanceId). The parameter is used in a basic string replace. Its value is supplied at Start Execution time.</td>
<td>{ &quot;description&quot;: &quot;Create Image Demo&quot;, &quot;version&quot;: &quot;0.3&quot;, &quot;assumeRole&quot;: &quot;Your_Automation_Assume_Role_ARN&quot;, &quot;parameters&quot;: { &quot;instanceId&quot;: { &quot;type&quot;: &quot;STRING&quot;, &quot;description&quot;: &quot;Instance to create image from&quot; } } }</td>
</tr>
<tr>
<td>System variable</td>
<td>A general variable substituted into the document when any part of the document is evaluated.</td>
<td>&quot;activities&quot;: [ { &quot;id&quot;: &quot;copyImage&quot;, &quot;activityType&quot;: &quot;AWS-CopyImage&quot;, &quot;maxAttempts&quot;: 1, &quot;onFailure&quot;: &quot;Continue&quot;, &quot;inputs&quot;: { &quot;ImageName&quot;: &quot;{{imageName}}&quot;, &quot;SourceImageId&quot;: &quot;{{sourceImageId}}&quot;, &quot;SourceRegion&quot;: &quot;{{sourceRegion}}&quot;, &quot;Encrypted&quot;: true, &quot;ImageDescription&quot;: &quot;Test CopyImage Description created on {{global:DATE}}&quot; } } ]</td>
</tr>
<tr>
<td>Automation variable</td>
<td>A variable relating to the automation execution substituted into the document when any part of the document is evaluated.</td>
<td>{ &quot;name&quot;: &quot;runFixedCmds&quot;, &quot;action&quot;: &quot;aws:runCommand&quot;, &quot;maxAttempts&quot;: 1, &quot;onFailure&quot;: &quot;Continue&quot;, &quot;inputs&quot;: { &quot;DocumentName&quot;: &quot;AWS-RunPowerShellScript&quot;, &quot;InstanceIds&quot;: [ &quot;{LaunchInstance.InstanceIds}&quot; ] }, &quot;Parameters&quot;: {</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
<td>Example</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| "commands": | [ | "dir",
| | "date",
| | "echo {Hello
| | {{ssm:administratorName}}}",
| | "{{outputFormat}}" -f
| | "left","right","{{global:DATE}}","{{automation:EXECUTION_ID}}","{{global:TIME}}"
<p>| | ] | } |
| | } | } |</p>
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
</table>
| SSM parameter    | A variable defined within the Parameter Service. It is not declared as a Document Parameter. It may require permissions to access.                                                                         | { "description": "Run Command Demo", "schemaVersion": "0.3", "assumeRole": "arn:aws:iam::123456789012:role/roleName", "parameters": { "commands": { "type": "STRING_LIST", "description": "list of commands to execute as part of first step" }, "instanceIds": { "type": "STRING_LIST", "description": "list of instances to execute commands on" } }, "mainSteps": [ { "name": "runFixedCmds", "action": "aws:runCommand", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "DocumentName": "AWS-RunPowerShellScript", "InstanceIds": [ "{LaunchInstance.InstanceIds}" ] }, "Parameters": { "commands": [ "dir", "date", "echo
{Hello {{ssm:administratorName}}}
"{{outputFormat}}" -f "left","right","{{global:DATE}}","{{automation:EXECUTION_ID}}","{{global:TIME}}"
] } } ]}
## Supported Scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Comments</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant ARN assumeRole at create</td>
<td>An authorization check will be performed to check the calling user is permitted to pass the given assume role.</td>
<td></td>
</tr>
</tbody>
</table>

```
{
  "description": "Test all Automation resolvable parameters",
  "schemaVersion": "0.3",
  "assumeRole": "arn:aws:iam::123456789012:role/roleName",
  "parameters": {
```

| Document Parameter supplied for assumeRole at create | Must be defined in the Parameter list of the document. | 

```
{
  "description": "Test all Automation resolvable parameters",
  "schemaVersion": "0.3",
  "assumeRole": "{{dynamicARN}}",
  "parameters": {
```

| Value supplied for Document Parameter at start. | Customer supplies the value to use for a parameter. Any execution inputs supplied at start time need to be defined in the parameter list of the document. | 

```
... 

```
"parameters": {
  "amiId": {
    "type": "STRING",
    "default": "ami-7f2e6015",
    "description": "list of commands to execute as part of first step"
  }
},
...

```

| SSM parameter referenced within step definition | The variable exists within the customers account and the assumeRole for the document has access to the variable. A check will be performed at create time to confirm the assumeRole has access. SSM parameters do not need to be set in the parameter list of the document. | 

```
... 

```
"mainSteps": [
  {
    "name": "RunSomeCommands",
    "action": "aws:runCommand",
    "maxAttempts": 1,
    "onFailure": "Continue",
    "inputs": {
      "DocumentName": "AWS:RunPowerShell",
      "InstanceIds": [{LaunchInstance.InstanceIds}],
      "parameters": {
```

```
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Comments</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>System variable referenced within step definition</td>
<td>A system variable is substituted into the document at execution time. The value injected into the document is relative to when the substitution occurs. e.g. The value of a time variable injected at step 1 will be different to the value injected at step 3 due to the time taken to execute the steps between. System variables do not need to be set in the parameter list of the document.</td>
<td>...</td>
</tr>
</tbody>
</table>

```bash
"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:TIME}}"
        ]
      }
    }
  }, ...
```
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Comments</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation variable referenced within step definition.</td>
<td>Automation variables do not need to be set in the parameter list of the document. The only supported Automation variable is automation:EXECUTION_ID.</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;mainSteps&quot;: [</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;name&quot;: &quot;invokeLambdaFunction&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;action&quot;: &quot;aws:invokeLambdaFunction&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;maxAttempts&quot;: 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;onFailure&quot;: &quot;Continue&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;inputs&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;FunctionName&quot;: &quot;Hello-World-LambdaFunction&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Payload&quot; : &quot;{{executionId} : {{automation:EXECUTION_ID}}}&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
<tr>
<td>Refer to output from previous step within next step definition.</td>
<td>This is parameter redirection. The output of a previous step is referenced using the syntax {{stepName.OutputName}}. This syntax cannot be used by the customer for Document Parameters. This is resolved at the time of execution for the referring step. The parameter is not listed in the parameters of the document.</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;mainSteps&quot;: [</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;name&quot;: &quot;LaunchInstance&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;action&quot;: &quot;aws:runInstances&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;maxAttempts&quot;: 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;onFailure&quot;: &quot;Continue&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;inputs&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;ImageId&quot;: &quot;{{amiId}}&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;MinInstanceCount&quot;: 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;MaxInstanceCount&quot;: 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;name&quot;: &quot;changeState&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;action&quot;: &quot;aws:changeInstanceState&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;maxAttempts&quot;: 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;onFailure&quot;: &quot;Continue&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;inputs&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;InstanceIds&quot;: [&quot;{{LaunchInstance.InstanceIds}}&quot;],</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;DesiredState&quot;: &quot;terminated&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>
## Unsupported Scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Comment</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSM Parameter supplied for assumeRole at create</td>
<td>Not supported.</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;description&quot;: &quot;Test all Automation resolvable parameters&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;schemaVersion&quot;: &quot;0.3&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;assumeRole&quot;:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;{{ssm:administratorRoleARN}}&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;parameters&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;amiId&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;type&quot;: &quot;STRING&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;default&quot;: &quot;ami-7f2e6015&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;description&quot;: &quot;list of commands to execute as part of first step&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>SSM Parameter supplied for Document Parameter at start</td>
<td>The user supplies an input parameter at start time which is an SSM parameter</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;parameters&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;amiId&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;type&quot;: &quot;STRING&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;default&quot;: &quot;ami-7f2e6015&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;description&quot;: &quot;list of commands to execute as part of first step&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Variable step definition</td>
<td>The definition of a step in the document is constructed by variables.</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;mainSteps&quot;: [</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;name&quot;: &quot;LaunchInstance&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;action&quot;: &quot;aws:runInstances&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;{{attemptModel}}&quot;: 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;onFailure&quot;: &quot;Continue&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;inputs&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;ImageId&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;type&quot;: &quot;STRING&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;default&quot;: &quot;ami-12345678&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;description&quot;: &quot;list of commands to execute as part of first step&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;MinInstanceCount&quot;: 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;MaxInstanceCount&quot;: 2,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User supplies input :</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{ &quot;attemptModel&quot; : &quot;minAttempts&quot; }</td>
</tr>
<tr>
<td>Scenario</td>
<td>Comment</td>
<td>Example</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Cross referencing Document Parameters | The user supplies an input parameter at start time which is a reference to another parameter in the document. | ```
... "parameters": {
  "amiId": { 
  "type": "STRING",
  "default":
  "ami-7f2e6015",
  "description": "list of commands to execute as part of first step"
  },
  "otherAmiId": { 
  "type": "STRING",
  "description": "The other amiId to try if this one fails".
  
  "default": "{{amiId}}"
  },
...``` |
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-level</td>
<td>The document defines a variable which 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.</td>
</tr>
<tr>
<td>expansion</td>
<td></td>
</tr>
</tbody>
</table>

```json
"parameters": {
  "param1": {
    "type": "STRING",
    "default": "param2",
    "description": "The parameter to reference"
  },
  "param2": {
    "type": "STRING",
    "default": "echo \{Hello world\}",
    "description": "What to execute"
  }
},
"mainSteps": [{
  "name": "runFixedCmds",
  "action": "aws:runCommand",
  "maxAttempts": 1,
  "onFailure": "Continue",
  "inputs": {
    "DocumentName": "AWS-RunPowerShellScript",
    "InstanceIds": "\{{{LaunchInstance.InstanceIds}}",
    "Parameters": {
      "commands": [ "\{{param1}}"]
    }
  }
}
```

Note: The customer intention here would be to execute a runCommand of "echo {Hello world}"

### Troubleshooting Systems Manager Automation

Use the following information to help you troubleshoot problems with the Automation service. This topic includes specific tasks to resolve issues based on Automation error messages.

**Topics**

- Common Automation Errors (p. 150)
- Automation Execution Failed to Start (p. 157)
- Execution Started, but Status is Failed (p. 158)
- Execution Started, but Timed Out (p. 160)
Common Automation Errors

This section includes information about common Automation errors.

VPC not defined 400

By default, when Automation runs either the AWS-UpdateLinuxAmi document or the AWS-UpdateWindowsAmi document, the system creates a temporary instance in the default VPC (172.30.0.0/16). If you deleted the default VPC, you will receive the following error:

VPC not defined 400

To solve this problem, you must create a new Automation document that includes the subnet ID. Copy a sample document below that includes the subnet ID parameter and create a new document. For information about creating a document, see Creating an Automation Document (p. 114).

AWS-UpdateLinuxAmi

```json
{
   "schemaVersion":"0.3",
   "description":"Updates AMI with Linux distribution packages and Amazon software. For details, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/sysman-ami-walkthrough.html",
   "assumeRole": "{\"AutomationAssumeRole\"}"
   "parameters":{
       "SourceAmiId":{
           "type":"String",
           "description":\"(Required) The source Amazon Machine Image ID.\"
       },
       "InstanceIamRole":{
           "type":"String",
           "description":\"(Required) The name of the role that enables Systems Manager (SSM) to manage the instance.\",
           "default":"ManagedInstanceProfile"
       },
       "AutomationAssumeRole":{
           "type":"String",
           "description":\"(Required) The ARN of the role that allows Automation to perform the actions on your behalf.\",
           "default":\"arn:aws:iam::\{\"global:ACCOUNT_ID\}\:role/AutomationServiceRole\"
       },
       "SubnetId":{
           "type":"String",
           "description":\"(Required) The subnet that the created instance will be placed into.\"
       },
       "TargetAmiName":{
           "type":"String",
           "description":\"(Optional) The name of the new AMI that will be created. Default is a system-generated string including the source AMI id, and the creation time and date.\",
           "default":\"UpdateLinuxAmi_from_{\"SourceAmiId\"}_on_{\"global:DATE_TIME\"}\"
       },
       "InstanceType":{
           "type":"String",
           "description":\"(Optional) Type of instance to launch as the workspace host. Instance types vary by region. Default is t2.micro.\",
           "default":\"t2.micro\"
       },
       "PreUpdateScript":{
           "type":"String",
           "description":\"(Optional) URL of a script to run before updates are applied. Default \"\"none\"\" is to not run a script.\"
       }
   }
}
```

"default":"none",
"PostUpdateScript":{
  "type":"String",
  "description":"(Optional) URL of a script to run after package updates are applied. Default ("none") is to not run a script.",
  "default":"none"
},
"IncludePackages":{
  "type":"String",
  "description":"(Optional) Only update these named packages. By default ("all"), all available updates are applied.",
  "default":"all"
},
"ExcludePackages":{
  "type":"String",
  "description": "(Optional) Names of packages to hold back from updates, under all conditions. By default ("none"), no package is excluded.",
  "default": "none"
}
},
"mainSteps": [
  {
    "name": "launchInstance",
    "action": "aws:runInstances",
    "maxAttempts": 3,
    "timeoutSeconds": 1200,
    "onFailure": "Abort",
    "inputs": {
      "ImageId": "{{SourceAmiId}}",
      "InstanceType": "{{InstanceType}}",
      "SubnetId": "{{ SubnetId }}",
      "MinInstanceCount": 1,
      "MaxInstanceCount": 1,
      "IamInstanceProfileName": "{{InstanceIamRole}}"
    }
  }
],
{
  "name": "updateOSSoftware",
  "action": "aws:runCommand",
  "maxAttempts": 3,
  "timeoutSeconds": 3600,
  "onFailure": "Abort",
  "inputs": {
    "DocumentName": "AWS-RunShellScript",
    "InstanceIds": [
      "{{launchInstance.InstanceIds}}"
    ],
    "Parameters": {
      "commands": [
        "set -e",
        "[ -x \"$(which wget)\" ] && get_contents='wget $1 -O -'",
        "[ -x \"$(which curl)\" ] && get_contents='curl -s -f $1'",
        "chmod +x /tmp/aws-update-linux-instance",
      ]
    }
  }
}
## Troubleshooting Systems Manager Automation

### AWS-UpdateWindowsAmi

```
{
  "schemaVersion":"0.3",
  "description":"Updates a Microsoft Windows AMI. By default it will install all Windows updates, Amazon software, and Amazon drivers. It will then sysprep and create a new AMI. Supports Windows Server 2008 R2 and greater.",
  "assumeRole": "{{ AutomationAssumeRole }}",
  "parameters":{
    "SourceAmiId":{
      "type":"String",
      "description": "(Required) The source Amazon Machine Image ID."
    },
    "IamInstanceProfileName":{
      "type":"String",
      "description": "(Required) The name of the role that enables Systems Manager to manage the instance.",
      "default": "ManagedInstanceProfile"
    },
    "AutomationAssumeRole":{
      "type": "String",
      "description": "(Required) The role that Systems Manager uses to assume the AssumeRole."
    }
  }
}
```
"type":"String",
"description": "(Required) The ARN of the role that allows Automation to perform
the actions on your behalf.",
"default": "arn:aws:iam::{{global:ACCOUNT_ID}}:role/AutomationServiceRole"
},
"SubnetId": {
"type": "String",
"description": "(Required) The subnet that the created instance will be placed
into."
},
"TargetAmiName": {
"type": "String",
"description": "(Optional) The name of the new AMI that will be created. Default is
a system-generated string including the source AMI id, and the creation time and date.",
"default": "UpdateWindowsAmi_from_{{SourceAmiId}}_on_{{global:DATE_TIME}}"
},
"InstanceType": {
"type": "String",
"description": "(Optional) Type of instance to launch as the workspace host.
Instance types vary by region. Default is t2.medium."
"default": "t2.medium"
},
"IncludeKbs": {
"type": "String",
"description": "(Optional) Specify one or more Microsoft Knowledge Base (KB)
article IDs to include. You can install multiple IDs using comma-separated values. When
specified, the categories and security level values are ignored. Valid formats: KB9876543 or
9876543.",
"default": ""
},
"ExcludeKbs": {
"type": "String",
"description": "(Optional) Specify one or more Microsoft Knowledge Base (KB)
article IDs to exclude. You can exclude multiple IDs using comma-separated values. When
specified, all these KBs are excluded from install process. Valid formats: KB9876543 or
9876543.",
"default": ""
},
"Categories": {
"type": "String",
"description": "(Optional) Specify one or more update categories. You can filter
categories using comma-separated values. By default patches for all categories are
selected. If value supplied, the update list is filtered by those values. Options:
Critical Update, Security Update, Definition Update, Update Rollup, Service Pack, Tool,
Update or Driver. Valid formats include a single entry, for example: Critical Update. Or,
you can specify a comma separated list: Critical Update,Security Update,Definition Update.
NOTE: There cannot be any spaces around the commas."
"default": ""
},
"SeverityLevels": {
"type": "String",
"description": "(Optional) Specify one or more MSRC severity levels associated with
an update. You can filter severity levels using comma-separated values. By default patches
for all security levels are selected. If value supplied, the update list is filtered by those values. Options: Critical, Important, Low, Moderate or Unspecified. Valid formats
include a single entry, for example: Critical. Or, you can specify a comma separated list:
Critical,Important,Low.",
"default": ""
},
"PreUpdateScript": {
"type": "String",
"description": "(Optional) A script provided as a string. It will execute prior to
installing OS updates."
"default": ""
},
"PostUpdateScript": {
"type": "String",
"description": "(Optional) A script provided as a string. It will execute after installing OS updates."
"default": ""
}
"mainSteps": [ 
{ 
"name": "LaunchInstance",
"action": "aws:runInstances",
"timeoutSeconds": 1800,
"maxAttempts": 3,
"onFailure": "Abort",
"inputs": { 
"ImageId": "{{ SourceAmiId }}",
"InstanceType": "{{ InstanceType }}",
"SubnetId": "{{ SubnetId }}",
"MinInstanceCount": 1,
"MaxInstanceCount": 1,
"IamInstanceProfileName": "{{ IamInstanceProfileName }}"
}
},
{ 
"name": "RunPreUpdateScript",
"action": "aws:runCommand",
"maxAttempts": 3,
"onFailure": "Abort",
"timeoutSeconds": 1800,
"inputs": { 
"DocumentName": "AWS-RunPowerShellScript",
"InstanceIds": [ 
"{{ LaunchInstance.InstanceIds }}"
],
"Parameters": {
"commands": "{{ PreUpdateScript }}"
}
}
},
{ 
"name": "UpdateSSMAgent",
"action": "aws:runCommand",
"maxAttempts": 3,
"onFailure": "Abort",
"timeoutSeconds": 600,
"inputs": { 
"DocumentName": "AWS-UpdateSSMAgent",
"InstanceIds": [ 
"{{ LaunchInstance.InstanceIds }}"
]
}
},
{ 
"name": "UpdateEC2Config",
"action": "aws:runCommand",
"maxAttempts": 3,
"onFailure": "Abort",
"timeoutSeconds": 7200,
"inputs": { 
"DocumentName": "AWS-InstallPowerShellModule",
"InstanceIds": [ 
"{{ LaunchInstance.InstanceIds }}"
],
"Parameters": { 
"executionTimeout": "7200",
}
}
"sourceHash": "14CAD416F4A054894EBD2091EA4B99E542368BE5895BDD466B567C1ABA87C87C",
"commands": [
  "Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force",
  "Import-Module AWSUpdateWindowsInstance",
  "if ([Environment]::OSVersion.Version.Major -ge 10) {",
    "Install-AwsUwiEC2Launch -Id {{ automation:EXECUTION_ID }}",
  } else {
    "Install-AwsUwiEC2Config -Id {{ automation:EXECUTION_ID }}",
  }
],

"name": "UpdateAWSPVDriver",
"action": "aws:runCommand",
"maxAttempts": 3,
"onFailure": "Abort",
"timeoutSeconds": 600,
"inputs": {
  "DocumentName": "AWS-ConfigureAWSPackage",
  "InstanceIds": [
    "{{ LaunchInstance.InstanceIds }}"
  ],
  "Parameters": {
    "name": "AWSPVDriver",
    "action": "Install"
  }
},

"name": "InstallWindowsUpdates",
"action": "aws:runCommand",
"maxAttempts": 3,
"onFailure": "Abort",
"timeoutSeconds": 14400,
"inputs": {
  "DocumentName": "AWS-InstallWindowsUpdates",
  "InstanceIds": [
    "{{ LaunchInstance.InstanceIds }}"
  ],
  "Parameters": {
    "Action": "Install",
    "IncludeKbs": "{{ IncludeKbs }}",
    "ExcludeKbs": "{{ ExcludeKbs }}",
    "Categories": "{{ Categories }}",
    "SeverityLevels": "{{ SeverityLevels }}"
  }
},

"name": "RunPostUpdateScript",
"action": "aws:runCommand",
"maxAttempts": 3,
"onFailure": "Abort",
"timeoutSeconds": 1800,
"inputs": {
  "DocumentName": "AWS-RunPowerShellScript",
  "InstanceIds": [
    "{{ LaunchInstance.InstanceIds }}"
  ],
  "Parameters": {
    "commands": "{{ PostUpdateScript }}"
  }
}
Troubleshooting Systems Manager Automation

```json
{
  "name":"RunSysprepGeneralize",
  "action":"aws:runCommand",
  "maxAttempts":3,
  "onFailure":"Abort",
  "timeoutSeconds":7200,
  "inputs":{
    "DocumentName":"AWS-InstallPowerShellModule",
    "InstanceIds":{
      "{{ LaunchInstance.InstanceIds }}"
    },
    "Parameters":{
      "executionTimeout":7200,
      "sourceHash":"14CAD416F4A054894EBD2091EA4B99E542368BE5895BD466B567C1ABA87C87C",
      "commands":[
        "Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force",
        "Import-Module AWSUpdateWindowsInstance",
        "Start-AwsUwiSysprep -Id {{ automation:EXECUTION_ID }}"
      ]
    }
  }
},
{
  "name":"StopInstance",
  "action":"aws:changeInstanceState",
  "maxAttempts":3,
  "onFailure":"Abort",
  "timeoutSeconds":7200,
  "inputs":{
    "InstanceIds":{
      "{{ LaunchInstance.InstanceIds }}"
    },
    "CheckStateOnly":false,
    "DesiredState":"stopped"
  }
},
{
  "name":"CreateImage",
  "action":"aws:createImage",
  "maxAttempts":3,
  "onFailure":"Abort",
  "inputs":{
    "InstanceId":{
      "{{ LaunchInstance.InstanceIds }}"
    },
    "ImageName":{{ TargetAmiName }},
    "NoReboot":true,
    "ImageDescription":"Test CreateImage Description"
  }
},
{
  "name":"CreateTags",
  "action":"aws:createTags",
  "maxAttempts":3,
  "onFailure":"Abort",
  "inputs":{
    "ResourceType":"EC2",
    "ResourceIds":{
      "{{ CreateImage.ImageId }}"
    },
    "Tags":{
      "Key":"Original_AMI_ID",
      "Value":"Created from {{ SourceAmiId }}"
    }
  }
}
```
Automation Execution Failed to Start

An Automation execution can fail with an access denied error or an invalid assume role error if you have not properly configured IAM users, roles, and policies for Automation.

Access Denied

The following examples describe situations when an Automation execution failed to start with an access denied error.

Access Denied to Systems Manager API


- Possible cause 1: The IAM user attempting to start the Automation execution does not have permission to invoke the StartAutomationExecution API. To resolve this issue, attach the required IAM policy to the user account that was used to start the execution. For more information, see Task 4: Configure User Access to Automation (p. 101).

- Possible cause 2: The IAM user attempting to start the Automation execution has permission to invoke the StartAutomationExecution API, but does not have permission to invoke the API by using the specific Automation document. To resolve this issue, attach the required IAM policy to the user account that was used to start the execution. For more information, see Task 4: Configure User Access to Automation (p. 101).

Access Denied Because of Missing PassRole Permissions

Error message: User: user arn is not authorized to perform: iam:PassRole on resource: automation assume role arn (Service: AWSSimpleSystemsManagement; Status Code: 400; Error Code: AccessDeniedException; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)

The IAM user attempting to start the Automation execution does not have PassRole permission for the assume role. To resolve this issue, attach the iam:PassRole policy to the role of the IAM user attempting to start the Automation execution. For more information, see Task 3: Attach the iam:PassRole Policy to Your Automation Role (p. 100).
Invalid Assume Role

When you execute an Automation, an assume role is either provided in the document or passed as a parameter value for the document. Different types of errors can occur if the assume role is not specified or configured properly.

Malformed Assume Role

**Error message:** The format of the supplied assume role ARN is invalid. The assume role is improperly formatted. To resolve this issue, verify that a valid assume role is specified in your Automation document or as a runtime parameter when executing the Automation.

Assume Role Can’t Be Assumed

**Error message:** The defined assume role is unable to be assumed.

(Service: AWSSimpleSystemsManagement; Status Code: 400; Error Code: InvalidAutomationExecutionParametersException; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)

- Possible cause 1: The assume role does not exist. To resolve this issue, create the role. For more information, see the section called “Setting Up Automation” (p. 96). Specific details for creating this role are described in the following topic, Task 1: Create a Service Role for Automation (p. 99).
- Possible cause 2: The assume role does not have a trust relationship with the Systems Manager service. To resolve this issue, create the trust relationship. For more information, see Task 2: Add a Trust Relationship for Automation (p. 100).

Execution Started, but Status is Failed

Action-Specific Failures

Automation documents contain steps and steps execute in order. Each step invokes one or more AWS service APIs. The APIs determine the inputs, behavior, and outputs of the step. There are multiple places where an error can cause a step to fail. Failure messages indicate when and where an error occurred.

To see a failure message in the EC2 console, choose the **View Outputs** link of the failed step. To see a failure message from the 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:runInstances` action failed. Each example explores a different type of error.

Missing Image

**Error message:** Automation Step Execution fails when it is launching the instance(s). Get Exception from RunInstances API of ec2 Service. Exception Message from RunInstances API: [The image id '[ami id]' does not exist (Service: AmazonEC2; Status Code: 400; Error Code: InvalidAMIID.NotFound; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)]. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

The `aws:runInstances` action received input for an `ImageId` that doesn't exist. To resolve this problem, update the automation document or parameter values with the correct AMI ID.

Assume Role Policy Doesn't Have Sufficient Permissions

**Error message:** Automation Step Execution fails when it is launching the instance(s). Get Exception from RunInstances API of ec2 Service. Exception Message from RunInstances API: [You are not authorized to perform this operation. Encoded authorization failure message: xxxxxxx (Service: AmazonEC2; Status Code: 403; Error Code: UnauthorizedOperation; Request ID: xxxxxxxx-xxxx-
The assume role doesn't have sufficient permission to invoke the RunInstances API on Amazon EC2 instances. To resolve this problem, attach an IAM policy to the assume role that has permission to invoke the RunInstances API. For more information, see the Method 2: Using IAM to Configure Roles for Automation (p. 98).

**Unexpected State**

**Error message:** Step fails when it is 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.

<table>
<thead>
<tr>
<th>Action</th>
<th>AWS Service(s) Invoked by This Action</th>
<th>For Information About This Service</th>
<th>Troubleshooting Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws:runInstances</td>
<td>Amazon EC2</td>
<td>Amazon EC2 User Guide</td>
<td>Troubleshooting EC2 Instances</td>
</tr>
<tr>
<td>aws:changeInstanceState</td>
<td>Amazon EC2</td>
<td>Amazon EC2 User Guide</td>
<td>Troubleshooting EC2 Instances</td>
</tr>
<tr>
<td>aws:runCommand</td>
<td>Systems Manager</td>
<td>Systems Manager Run Command</td>
<td>Troubleshooting Run Command</td>
</tr>
<tr>
<td>aws:createImage</td>
<td>Amazon EC2</td>
<td>Amazon Machine Images</td>
<td></td>
</tr>
<tr>
<td>aws:createStack</td>
<td>AWS CloudFormation</td>
<td>AWS CloudFormation User Guide</td>
<td>Troubleshooting AWS CloudFormation</td>
</tr>
<tr>
<td>aws:deleteStack</td>
<td>AWS CloudFormation</td>
<td>AWS CloudFormation User Guide</td>
<td>Troubleshooting AWS CloudFormation</td>
</tr>
<tr>
<td>aws:deleteImage</td>
<td>Amazon EC2</td>
<td>Amazon Machine Images</td>
<td></td>
</tr>
<tr>
<td>aws:copyImage</td>
<td>Amazon EC2</td>
<td>Amazon Machine Images</td>
<td></td>
</tr>
<tr>
<td>aws:createTag</td>
<td>Amazon EC2, Systems Manager</td>
<td>EC2 Resource and Tags</td>
<td></td>
</tr>
<tr>
<td>aws:invokeLambdaFunction</td>
<td>AWS Lambda</td>
<td>AWS Lambda Developer Guide</td>
<td>Troubleshooting Lambda</td>
</tr>
</tbody>
</table>
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 Automation document from executing 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 execute than the value specified for `timeoutSeconds` in the step. To resolve this issue, specify a longer value for `timeoutSeconds`. If that does not solve the problem, investigate why the step takes longer to execute than expected.

AWS Systems Manager Run Command

AWS Systems Manager Run Command lets you remotely and securely manage the configuration of your managed instances. A *managed instance* is any Amazon EC2 instance or on-premises machine in your hybrid environment that has been configured for Systems Manager. Run Command enables you to automate common administrative tasks and perform ad hoc configuration changes at scale. You can use Run Command from the AWS console, the AWS Command Line Interface, AWS Tools for Windows PowerShell, or the AWS SDKs. Run Command is offered at no additional cost.

Administrators use Run Command to perform the following types of tasks on their managed instances: install or bootstrap applications, build a deployment pipeline, capture log files when an instance is terminated from an Auto Scaling group, and join instances to a Windows domain, to name a few.

**Getting Started**

The following table includes information to help you get started with Run Command.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tutorial: Remotely Manage Your Amazon EC2 Instances <em>(Amazon EC2 User Guide)</em></td>
<td>(Optional) The tutorial shows you how to quickly send a command using Run Command with AWS Tools for Windows PowerShell or the AWS Command Line Interface (AWS CLI).</td>
</tr>
<tr>
<td>Systems Manager Prerequisites (p. 4)</td>
<td>(Required) Verify that your instances meet the minimum requirements for Run Command, configure required roles, and install the SSM Agent.</td>
</tr>
<tr>
<td>Setting Up AWS Systems Manager in Hybrid Environments (p. 27)</td>
<td>(Optional) Register on-premises servers and VMs with AWS so that you can manage them using Run Command.</td>
</tr>
<tr>
<td>Executing Commands Using Systems Manager Run Command (p. 174)</td>
<td>Learn how to execute a command from the EC2 console and how to execute commands to a fleet of managed instances.</td>
</tr>
</tbody>
</table>
## Run Command Walkthroughs (p. 180)

Learn how to execute commands using either AWS Tools for Windows PowerShell or the AWS CLI.

## Components and Concepts

As you get started with Systems Manager Run Command, you'll benefit from understanding the components and concepts of this feature.

<table>
<thead>
<tr>
<th>Component/Concept</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systems Manager Documents</td>
<td>A Systems Manager document defines the plugins to run and the parameters to use when a command executes on a machine. When you execute a command, you specify the Systems Manager document that Run Command uses. Run Command includes pre-defined documents that enable you to quickly perform common tasks on a machine. You can also create your own Systems Manager documents. The first time you execute a command from a new Systems Manager document, the system stores the document with your AWS account. For more information, see AWS Systems Manager Documents (p. 258).</td>
</tr>
<tr>
<td>Commands</td>
<td>You can configure managed instances by sending commands from your local machine. You don't need to log on locally to configure your instances. You can send commands using one of the following: the Amazon EC2 console, AWS Tools for Windows PowerShell, the AWS Command Line Interface (AWS CLI), the Systems Manager API, or Amazon SDKs. For more information, see Systems Manager AWS Tools for Windows PowerShell Reference, Systems Manager AWS CLI Reference, and the AWS SDKs.</td>
</tr>
<tr>
<td>SSM Agent</td>
<td>The SSM Agent is AWS software that you install on your EC2 instances and servers and VMs in your hybrid environment. The agent processes Run Command requests and configures your machine as specified in the request. For more information, see Installing and Configuring SSM Agent on Linux Instances (p. 15) (Linux) and Installing and Configuring SSM Agent on Windows Instances (p. 13) (Windows).</td>
</tr>
</tbody>
</table>

For information about Systems Manager limits, see AWS Systems Manager Limits. To increase limits, go to AWS Support Center and submit a limit increase request form.

**Contents**

- Setting Up Run Command (p. 162)
- Executing Commands Using Systems Manager Run Command (p. 174)
- Run Command Walkthroughs (p. 180)
Setting Up Run Command

This section includes recommended tasks for restricting command access to tagged instances and monitoring command execution. The tasks in this section are not required, but they can help minimize the security posture and day-to-day management of your instances. For this reason, we highly recommend you complete the tasks in this section.

Topics

• Restricting Run Command Access Based on Instance Tags (p. 162)
• Setting Up Events and Notifications (p. 164)

Restricting Run Command Access Based on Instance Tags

Before you can manage instances by using Run Command, you must configure an AWS Identity and Access Management (IAM) user policy for any user who will execute commands, and an IAM instance profile role for any instance that will process commands, as described in Configuring Access to Systems Manager (p. 8). You can further restrict command execution to specific instances by creating an IAM user policy that includes a condition that the user can only execute commands on instances that are tagged with specific Amazon EC2 tags. In the following example, the user is allowed to use Run Command (Effect: Allow, Action: ssm:SendCommand) by using any SSM document (Resource: arn:aws:ssm:*:*:document/*) on any instance (Resource: arn:aws:ec2:*:*:instance/*) with the condition that the instance is a Finance WebServer (ssm:resourceTag/Finance: WebServer). If the user sends a command to an instance that is not tagged or that has any tag other than Finance: WebServer, the execution results show AccessDenied.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ssm:SendCommand"],
            "Resource": ["arn:aws:ssm:*:*:document/*"]
        },
        {
            "Effect": "Allow",
            "Action": ["ssm:SendCommand"],
            "Resource": ["arn:aws:ec2:*:*:instance/*"],
            "Condition": {
```
You can create IAM policies that enable a user to execute commands on instances that are tagged with multiple tags. The following policy enables the user to execute commands on instances that have two tags. If a user sends a command to an instance that is not tagged with both of these tags, the execution results show AccessDenied.

```
{
    "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"
            ],
        },
        {
            "Effect":"Allow",
            "Resource": "*"
        }
    ]
}
```

You can also create IAM policies that enable a user to execute commands on multiple groups of tagged instances. The following policy enables the user to execute commands on either group of tagged instances, or both groups.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect":"Allow",
            "Action": ["ssm:SendCommand"
            ],
        },
        {
            "Effect":"Allow",
            "Resource": "*"
        }
    ]
}
```
"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-1::document/AWS-*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:UpdateInstanceInformation",
      "ssm:ListCommands",
      "ssm:ListCommandInvocations",
      "ssm:GetDocument"
    ],
    "Resource": "*"
  }
]

For more information about creating IAM user policies, see Managed Policies and Inline Policies in the IAM User Guide. For more information about tagging instances, see Tagging Your Amazon EC2 Resources in the Amazon EC2 User Guide for Linux Instances (content applies to Windows and Linux instances).

### Setting Up Events and Notifications

Systems Manager Run Command reports detailed status information about the different states a command experiences during processing and for each instance that processed the command. You can monitor command statuses using the following methods.

- Click the **Refresh** icon on the **Run Command** page in the Amazon EC2 console.
- Call `list-commands` or `list-command-invocations` using the AWS CLI. Or call `Get-SSMCommand` or `Get-SSMCommandInvocation` using AWS Tools for Windows PowerShell.
- Configure CloudWatch Events to log status changes.
- Configure Amazon SNS to send notifications for all status changes or specific statuses like Failed or TimedOut.

Contents
- Run Command Status (p. 165)
- Configuring CloudWatch Events for Run Command (p. 168)
- Configuring Amazon SNS Notifications for Run Command (p. 169)

Run Command Status

Run Command reports status details for three areas: plugins, invocations, and an overall command status. A plugin is a code-execution block that is defined in your command’s Systems Manager document. The AWS-* documents include only one plugin, but you can create your own documents that use multiple plugins. For more information about plugins, see SSM Document Plugin Reference (p. 284).

When you send a command to multiple instances at the same time, each copy of the command targeting each instance is a command invocation. For example, if you use the AWS-RunShellScript document and send an `ifconfig` command to 20 instances, that command has 20 invocations. Each command invocation individually reports status. The plugins for a given command invocation individually report status as well.

Lastly, Run Command includes an aggregated command status for all plugins and invocations. The aggregated command status can be different than the status reported by plugins or invocations, as noted in the following tables.

**Note**
If you execute commands to large numbers of instances using the `max-concurrency` or `max-errors` parameters, command status reflects the limits imposed by those parameters, as described in the following tables. For more information about these parameters, see Sending Commands to a Fleet (p. 177).

Detailed Status for Command Plugins and Invocations

<table>
<thead>
<tr>
<th>Status</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pending</td>
<td>The command was not yet received by the agent on the instance. If the command is not received by the agent before the value specified by the <strong>Timeout (seconds)</strong> parameter is reached, then the status changes to Delivery Timed Out.</td>
</tr>
<tr>
<td>In Progress</td>
<td>The command was received by the agent, or the command started executing on the instance. Depending on the result of all command plugins, the status will change to Success, Failed, or Execution Timed Out. If the agent is not available on the instance, the command status will show In Progress until the agent is available again. The status will then change to a terminal state.</td>
</tr>
<tr>
<td>Delayed</td>
<td>The system attempted to send the command to the instance but was not successful. The system will retry again.</td>
</tr>
</tbody>
</table>
## Status Details

<table>
<thead>
<tr>
<th>Status</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>The command was received by the SSM Agent on the instance and returned an exit code of zero. This status does not mean the command was successfully processed on the instance. This is a terminal state. <strong>Note</strong> To troubleshoot errors or get more information about the command execution, send a command that handles errors or exceptions by returning appropriate exit codes (non-zero exit codes for command failure).</td>
</tr>
<tr>
<td>Delivery Timed Out</td>
<td>The command was not delivered to the instance before the delivery timeout expired. Delivery timeouts do not count against the parent command's max-errors limit, but they do contribute to whether the parent command status is Success or Incomplete. This is a terminal state.</td>
</tr>
<tr>
<td>Execution Timed Out</td>
<td>Command execution started on the instance, but the execution was not complete before the execution timeout expired. Execution timeouts count against the max-errors limit of the parent command. This is a terminal state.</td>
</tr>
<tr>
<td>Failed</td>
<td>The command was not successful on the instance. For a plugin, this indicates that the result code was not zero. For a command invocation, this indicates that the result code for one or more plugins was not zero. Invocation failures count against the max-errors limit of the parent command. This is a terminal state.</td>
</tr>
<tr>
<td>Canceled</td>
<td>The command was terminated before it was completed. This is a terminal state.</td>
</tr>
<tr>
<td>Undeliverable</td>
<td>The command can't be delivered to the instance. The instance might not exist or it might not be responding. Undeliverable invocations don't count against the parent command's max-errors limit, and they don't contribute to whether the parent command status is Success or Incomplete. This is a terminal state.</td>
</tr>
<tr>
<td>Terminated</td>
<td>The parent command exceeded its max-errors limit and subsequent command invocations were canceled by the system. This is a terminal state.</td>
</tr>
</tbody>
</table>
### Detailed Status for a Command

<table>
<thead>
<tr>
<th>Status</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pending</td>
<td>The command was not yet received by an agent on any instances.</td>
</tr>
<tr>
<td>In Progress</td>
<td>The command has been sent to at least one instance but has not reached a final state on all instances.</td>
</tr>
<tr>
<td>Delayed</td>
<td>The system attempted to send the command to the instance but was not successful. The system will retry again.</td>
</tr>
</tbody>
</table>
| Success              | The command was received by the SSM Agent on all specified or targeted instances and returned an exit code of zero. All command invocations have reached a terminal state, and the value of `max-errors` was not reached. This status does not mean the command was successfully processed on all specified or targeted instances. 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). |
| Delivery Timed Out   | The command was not delivered to the instance before the delivery timeout expired. The value of `max-errors` or more command invocations shows a status of Delivery Timed Out. This is a terminal state. |
| Execution Timed Out  | Command execution started on the instance, but the execution was not complete before the execution timeout expired. The value of `max-errors` or more command invocations shows a status of Execution Timed Out. This is a terminal state. |
| Failed               | The command was not successful on the instance. The value of `max-errors` or more command invocations shows a status of Failed. This is a terminal state. |
| Incomplete           | The command was attempted on all instances and one or more of the invocations does not have a value of Success. However, not enough invocations failed for the status to be Failed. This is a terminal state. |
| Canceled             | The command was terminated before it was completed. This is a terminal state.                                                           |
### Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate Exceeded</td>
<td>The number of instances targeted by the command exceeded the account limit for pending invocations. The system has canceled the command before executing it on any instance. This is a terminal state.</td>
</tr>
</tbody>
</table>

**Configuring CloudWatch Events for Run Command**

Use Amazon CloudWatch Events to log command execution status changes. You can create a rule that runs whenever there is a state transition, or when there is a transition to one or more states that are of interest.

You can also specify Run Command as a target action when a CloudWatch event occurs. For example, say a CloudWatch event is triggered that an instance in an Auto Scaling group is about to terminate. You can configure CloudWatch so the target of that event is a Run Command script that captures the log files from the instance before it is terminated. You can also configure a Run Command action when a new instance is created in an Auto Scaling group. For example, when CloudWatch receives the instance-created event, Run Command could enable the web server role or install software on the instance.

- Configuring CloudWatch Events for Run Command (p. 168)
- Configure Run Command as a CloudWatch Events Target (p. 168)

**Configuring CloudWatch Events for Run Command**

You can configure CloudWatch Events to notify you of Run Command status changes, or a status change for a specific command invocation. Use the following procedure to configure CloudWatch Events to send notification about Run Command.

**To configure CloudWatch Events for Run Command**

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Events**, and then choose **Create rule**.
3. Under **Event Source**, verify that **Event Pattern** is selected.
4. In the **Service Name** field, choose **EC2 Simple Systems Manager (SSM)**
5. In the **Event Type** field, choose **Run Command**.
6. Choose the detail types and statuses for which you want to receive notifications, and then choose **Add targets**.
7. In the **Select target type** list, choose a target type. For information about the different types of targets, see the corresponding AWS Help documentation.
8. Choose **Configure details**.
9. Specify the rule details, and then choose **Create rule**.

**Configure Run Command as a CloudWatch Events Target**

Use the following procedure to configure a Run Command action as the target of a CloudWatch event.

**To configure Run Command as a target of a CloudWatch event**

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Events**, and then either choose to create a new rule or edit an existing rule.
3. After specifying or verifying the details of the rule, choose **Add target**.
4. In the **Select target type** list, choose **SSM Run Command**.
5. In the **Document** list, choose an SSM document. The document determines the type of actions Run Command can perform on your instances.

**Note**
Verify that the document you choose can run on the instance operating system. Some documents run only on Windows or only on Linux operating systems. For more information about SSM Documents, see [AWS Systems Manager Documents](p. 258).

6. In the **Target key** field, specify either InstanceIds or tag:*EC2_tag_name*. Here are some examples of a **Target key** that uses an EC2 tag: tag:production and tag:server-role.
7. In the **Target value(s)** field, if you chose InstanceIds in the previous step, specify one or more instance IDs separated by commas. If you chose tag:*EC2_tag_name* in the previous step, specify one or more tag values. After you type the value, for example `web-server` or `database`, choose **Add**.
8. In the **Configure parameter(s)** section, choose an option and then complete any fields populated by your choice. Use the hover text for more information about the options. For more information about the parameter fields for your document, see [Executing Commands Using Systems Manager Run Command](p. 174) and choose the procedure for your document.
9. In the permissions section, choose **Create a new role for this specific resource** to create a new role with the required instance profile role for Run Command. Or, choose **Use existing role**. For more information about roles required for Run Command, see [Configuring Access to Systems Manager](p. 8).
10. Choose **Configure details** and complete the wizard.

### Configuring Amazon SNS Notifications for Run Command

You can configure Amazon Simple Notification Service (Amazon SNS) to send notifications about the status of commands you send using Systems Manager Run Command. Amazon SNS coordinates and manages the delivery or sending of notifications to subscribing clients or endpoints. You can receive a notification whenever a command changes to a new state or changes to a specific state, such as failed or timed out. In cases where you send a command to multiple instances, you can receive a notification for each copy of the command sent to a specific instance. Each copy is called an **invocation**.

Amazon SNS can deliver notifications as HTTP or HTTPS POST, email (SMTP, either plain-text 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](https://docs.aws.amazon.com/sns/latest/dg/about-sns.html) in the [Amazon Simple Notification Service Developer Guide](https://docs.aws.amazon.com/sns/latest/dg/).

For example, if you configure Amazon SNS to send a notification when a command status changes to failed, SNS sends an email notification with the details of the command execution.

**Note**
If you prefer, you can use Amazon CloudWatch Events to configure a target to invoke an AWS Lambda function when a command changes status. For more information, see [Configuring CloudWatch Events for Run Command](p. 168).

To set up Amazon SNS notifications when a command changes status, you must complete the following tasks.

1. **Configure Account Permissions** (p. 171)
2. **Create an IAM Role for Notifications** (p. 172)
3. **Configure Amazon SNS** (p. 172)
4. **Send a Command that Returns Status Notifications** (p. 173)
Configure Amazon SNS Notifications for Systems Manager

Run Command supports sending Amazon SNS notifications for commands that enter the following statuses. For information about the conditions that cause a command to enter one of these statuses, see Setting Up Events and Notifications (p. 164).

- In Progress
- Success
- Failed
- Timed Out
- Canceled

**Note**

Commands sent using Run Command also report Cancelling and Pending status. These statuses are not captured by SNS notifications.

If you configure Run Command for SNS notifications, SNS sends summary messages that include the following information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventTime</td>
<td>String</td>
<td>The time the event was triggered. The time stamp is important because SNS does not guarantee message delivery order. Example: 2016-04-26T13:15:30Z</td>
</tr>
<tr>
<td>DocumentName</td>
<td>String</td>
<td>The name of the SSM document used to execute this command.</td>
</tr>
<tr>
<td>CommandId</td>
<td>String</td>
<td>The ID generated by Run Command after the command was sent.</td>
</tr>
<tr>
<td>ExpiresAfter</td>
<td>Date</td>
<td>If this time is reached and the command has not already started executing, it will not execute.</td>
</tr>
<tr>
<td>OutputS3BucketName</td>
<td>String</td>
<td>The Amazon Simple Storage Service (Amazon S3) bucket where the responses to the command execution should be stored.</td>
</tr>
<tr>
<td>OutputS3KeyPrefix</td>
<td>String</td>
<td>The Amazon S3 directory path inside the bucket where the responses to the command execution should be stored.</td>
</tr>
<tr>
<td>RequestedDateTime</td>
<td>String</td>
<td>The time and date the request was sent to this specific instance.</td>
</tr>
<tr>
<td>InstanceId</td>
<td>String</td>
<td>The instance targeted by the command.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Status</td>
<td>String</td>
<td>Command status for the command.</td>
</tr>
</tbody>
</table>

If you send a command to multiple instances, Amazon SNS can send messages about each copy or invocation of the command that include the following information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventTime</td>
<td>String</td>
<td>The time the event was triggered. The time stamp is important because SNS does not guarantee message delivery order. Example: 2016-04-26T13:15:30Z</td>
</tr>
<tr>
<td>DocumentName</td>
<td>String</td>
<td>The name of the Systems Manager document used to execute this command.</td>
</tr>
<tr>
<td>RequestedDateTime</td>
<td>String</td>
<td>The time and date the request was sent to this specific instance.</td>
</tr>
<tr>
<td>CommandId</td>
<td>String</td>
<td>The ID generated by Run Command after the command was sent.</td>
</tr>
<tr>
<td>InstanceId</td>
<td>String</td>
<td>The instance targeted by the command.</td>
</tr>
<tr>
<td>Status</td>
<td>String</td>
<td>Command status for this invocation.</td>
</tr>
</tbody>
</table>

**Configure Account Permissions**

When you send a command that is configured for notifications, you specify a service role Amazon Resource Name (ARN). For example: --service-role-arn=arn:aws:iam::123456789012:myrole. This service role is used by Systems Manager to trigger SNS notifications.

To receive notifications from the Amazon SNS service, you must either attach the iam:PassRole policy to your existing AWS Identity and Access Management (IAM) user account, or create a new IAM account and attach this policy to it. If you create a new account, you must also attach the AmazonSSMFullAccess policy so the account can communicate with the Systems Manager API.

Use the following procedure to attach an IAM policy to your user account. If you need to create a new user account, see Creating an IAM User in Your AWS Account in the IAM User Guide.

**To attach the iam:PassRole policy to your user account**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users** and select the user (under **User name**).
3. At the top of the page, copy your **User ARN** to the clipboard.
4. Under **Permissions**, verify that either the **AmazonSSMFullAccess** policy is listed or there is a comparable policy that gives you permission to the Systems Manager API.
5. Choose **Add inline policy**.
6. On the **Set Permissions** page, choose **Policy Generator**, and then choose **Select**.
7. Verify that **Effect** is set to **Allow**.
8. From **AWS Services** choose **AWS Identity and Access Management**.
9. From **Actions** choose **PassRole**.
10. In the **Amazon Resource Name (ARN)** field, paste your ARN.
11. Choose **Add Statement**, and then choose **Next**.
12. On the **Review Policy** page, choose **Apply Policy**.

### Create an IAM Role for Notifications

In the previous procedure, you added an IAM policy to your user account so that you could send commands that return notifications. In the following procedure, you will create a role so that the Systems Manager service can act on your behalf when sending notifications.

2. In the navigation pane, choose **Roles**, and then choose **Create New Role**.
3. In **Step 1: Set Role Name** enter a name that identifies this role as a Run Command role for notifications.
4. In **Step 2: Select Role Type** choose **Amazon EC2**. The system skips **Step 3: Establish Trust** because this is a managed policy.
5. In **Step 4: Attach Policy** choose **AmazonSNSFullAccess**.
6. Choose **Next Step** and then choose **Create Role**. The system returns you to the **Roles** page.
7. Locate the role you just created and double-click it.
8. Choose the **Trust Relationships** tab, and then choose **Edit Trust Relationship**.
9. Add "ssm.amazonaws.com" to the existing policy as the following code snippet illustrates:

   ```json
   ``

**Note**

You must add a comma after the existing entry. "Service": "sns.amazonaws.com", or the JSON will not validate.

10. Choose **Update Trust Policy**.
11. Copy or make a note of the **Role ARN**. You will specify this ARN when you send a command that is configured to return notifications.

### Configure Amazon SNS

To use Amazon SNS to send email notifications, you must first create a **topic** and then subscribe your email addresses to the topic.
Create an Amazon SNS Topic

An Amazon SNS topic is a logical access point, a communication channel that Run Command uses to send the notifications. You create a topic by specifying a name for your topic.

For more information, see Create a Topic in the Amazon Simple Notification Service Developer Guide.

Note
After you create the topic, copy or make a note of the Topic ARN. You will specify this ARN when you send a command that is configured to return status notifications.

Subscribe to the Amazon SNS Topic

To receive the notifications that Run Command sends to the topic, you must subscribe an endpoint to the topic. In this procedure, for Endpoint, specify the email address where you want to receive the notifications from Run Command.

For more information, see Subscribe to a Topic in the Amazon Simple Notification Service Developer Guide.

Confirm Your Amazon SNS Subscription

Amazon SNS sends a confirmation email to the email address that you specified in the previous step. Make sure you open the email from AWS Notifications and choose the link to confirm the subscription before you continue with the next step.

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.

Send a Command that Returns Status Notifications

This section shows you how to send a command that is configured to return status notifications using either the Amazon EC2 console or the AWS Command Line Interface (AWS CLI).

Note
The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

To send a command from the Amazon EC2 console that returns notifications

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Run Command.
3. Choose Run a command.
5. For Target instances, choose the instances where you want the command to run. If you do not see an instance in this list, it might not be configured properly for Run Command. For more information, see Systems Manager Prerequisites (p. 4).
6. Enter information in the fields required by the Systems Manager document. In the SNS Notifications section, choose Enable SNS notifications.
7. In the Role field, type or paste the IAM role ARN you created earlier.
8. In the SNS Topic field, type or paste the Amazon SNS ARN you created earlier.
9. In the Notify me on field, choose the events for which you want to receive notifications.
10. In the Notify me for field, choose to receive notifications for each copy of a command sent to multiple instances (invocations) or the command summary.
12. Check your email for a message from Amazon SNS and open the email. Amazon SNS can take a few minutes to send the mail.

To send a command that is configured for notifications from the AWS CLI

1. Open the AWS CLI.
2. Specify parameters in the following command.

   ```
   aws ssm send-command --instance-ids "ID-1, ID-2" --document-name "name" --parameters
   commands=date --service-role ServiceRole ARN --notification-config NotificationArn=SNS
   ARN
   ```

   For example

   ```
   aws ssm send-command --instance-ids "i-12345678, i-34567890" --document-name "AWS-
   RunPowerShellScript" --parameters commands=date --service-role arn:aws-cn:iam::
   123456789012:myrole --notification-config NotificationArn=arn:aws-cn:sns:cn-
   north-1:123456789012:test
   ```

3. Press Enter.
4. Check your email for a message from Amazon SNS and open the email. Amazon SNS can take a few minutes to send the mail.

For more information about configuring Run Command from the command line, see Amazon EC2 Systems Manager API Reference and the Systems Manager AWS CLI Reference.

Executing Commands Using Systems Manager Run Command

This section includes information about how to send commands from the Amazon EC2 console, and how to send commands to a fleet of instances by using the Targets parameter with EC2 tags. This section also includes information about how to cancel a command.

For information about how to send commands using Windows PowerShell, see Systems Manager Run Command Walkthrough Using the AWS Tools for Windows PowerShell (p. 183) or the examples in the Tools for Windows PowerShell Reference. For information about how to send commands using the AWS CLI, see the Systems Manager Run Command Walkthrough Using the AWS CLI (p. 181) or the examples in the SSM CLI Reference.

**Important**

If this is your first time using Run Command, we recommend executing commands against a test instance or an instance that is not being used in a production environment.

Contents

- Executing Commands from the EC2 Console (p. 174)
- Sending Commands to a Fleet (p. 177)
- Canceling a Command (p. 180)

Executing Commands from the EC2 Console

You can use Run Command from the Amazon EC2 console to configure instances without having to login to each instance. This topic includes an example that shows how to update the SSM Agent (p. 176) on an instance by using Run Command.
Before You Begin

Before you send a command using Run Command, verify that your instances meet Systems Manager requirements (p. 4).

The following procedure describes how to execute a command from the EC2 console.

Note
The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

To send a command using Run Command

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, expand Systems Manager Services, and then choose Run Command.
3. Choose Run a command.
4. In the Command document section, choose a document.
5. In the Select Targets by section, choose Manually Selecting Instances to choose individual instances. Or choose Specifying a Tag, choose a group of instances by specifying one or more Amazon EC2 tags.
6. In the Execute on field, choose either Targets or Percent in the list. If you choose Targets, then you can specify the exact number of instances that should be allowed to run the command at one time, for example, 10. If you choose Percent, then you can choose a percentage of the instances that should be allowed to run the command at one time, for example 30. Percent is a helpful option when targeting EC2 tags and you are not certain of the total number of instances that will run the command.

This feature allows you to limit the number of instances running the command at one time to avoid impacting instance performance and availability. For more information, see Sending Commands to a Fleet (p. 177).

7. In the Stop after __ errors field, specify the maximum number of errors allowed before the system stops sending the command to additional instances. For example, if you specify 1, then the system stops sending the command to additional instances when the system receives the second error.

Instances that are already running a command when this value is reached are allowed to complete, but some of these executions may fail as well. For more information, see Sending Commands to a Fleet (p. 177).

8. In the next section, specify the parameters or options for your SSM document. Parameters and options are different for each document.
9. For Comment, we recommend providing information to will help you identify this command in your list of commands.
10. For Timeout (seconds), type the number of seconds that Run Command should attempt to reach an instance before it is considered unreachable and the command execution fails. The minimum is 30 seconds, the maximum is 30 days, and the default is 10 minutes.
11. (Optional) Choose Write output to an S3 bucket if you want to write the command output to an Amazon S3 bucket. If you chose this option, specify the S3 bucket and, optionally, an S3 key prefix. An S3 key prefix is a subfolder in the S3 bucket. A subfolder can help you organize Run Command output if you execute multiple commands against multiple instances.

Important
The Run Command Output page in the Amazon EC2 console truncates output after 2500 characters. Configure an Amazon S3 bucket before executing commands using Run Command. If your command output was longer than 2500 characters, you can view the full output in your Amazon S3 bucket. For more information, see Create a Bucket.
12. (Optional) Choose **Enable SNS notifications** if you want to receive notifications about the status of the commands you execute with Run Command. For more information, see Configuring Amazon SNS Notifications for Run Command (p. 169).

   **Note**
   After you specify parameters and options for your SSM document, expand the **AWS Command Line Interface command** section. This section includes a reusable command for different command-line platforms.

13. Choose **Run**, and then choose **View results**.

14. In the commands list, choose the command you just executed. If the command is still in progress, choose the refresh icon in the top right corner of the console.

15. When the **Status** column shows **Success** or **Failed**, choose the **Output** tab.

16. Choose **View Output**. The command output page shows the results of your command execution.

For information about canceling a command, see the section called “Canceling a Command” (p. 180).

**Example: Update the SSM Agent**

You can use the AWS-UpdateSSMAgent document to update the Amazon EC2 SSM Agent running on your Windows and Linux instances. You can update to either the latest version or downgrade to an older version. When you execute 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**
The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new **AWS Systems Manager console**. The steps in the new console will differ from the steps below.

**To update the SSM Agent using Run Command**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane under **Systems Manager Services**, choose **Run Command**.
3. Choose **Run a command**.
4. For **Command document**, choose **AWS-UpdateSSMAgent**.
5. In the **Select Targets by** section, choose **Manually Selecting Instances** to choose individual instances. Or choose **Specifying a Tag** to choose a group of instances by specifying one or more Amazon EC2 tags.
6. In the **Execute on** field, choose either **Targets** or **Percent** in the list. If you choose **Targets**, then you can specify the exact number of instances that should be allowed to run the command at one time, for example, 10. If you choose **Percent**, then you can choose a percentage of the instances that should be allowed to run the command at one time, for example 30. **Percent** is a helpful option when targeting EC2 tags and you are not certain of the total number of instances that will run the command.

   This feature allows you to limit the number of instances running the command at one time to avoid impacting instance performance and availability. For more information, see Sending Commands to a Fleet (p. 177).

7. In the **Stop after __ errors** field, specify the maximum number of errors allowed before the system stops sending the command to additional instances. For example, if you specify 1, then the systems stops sending the command to additional instances when the system receives the second error. For more information, see Sending Commands to a Fleet (p. 177).

8. (Optional) For **Version**, type the version of the SSM Agent to install. You can install older versions of the agent. If you do not specify a version, the service installs the latest version.
9. (Optional) For **Allow Downgrade**, choose **true** to install an earlier version of the SSM agent. If you choose this option, you must specify the earlier version number. Choose **false** to install only the newest version of the service.

10. For **Comment**, we recommend providing information that will help you identify this command in your list of commands.

11. For **Timeout (seconds)**, type the number of seconds that Run Command should attempt to reach an instance before it is considered unreachable and the command execution fails. The minimum is 30 seconds, the maximum is 30 days, and the default is 10 minutes.

12. (Optional) Choose **Write output to an S3 bucket** if you want to write the command output to an Amazon S3 bucket. If you chose this option, specify the S3 bucket and, optionally, an S3 key prefix. An S3 key prefix is a subfolder in the S3 bucket. A subfolder can help you organize Run Command output if you execute multiple commands against multiple instances.

    **Important**
    The Run Command **Output** page in the Amazon EC2 console truncates output after 2500 characters. Configure an Amazon S3 bucket before executing commands using Run Command. If your command output was longer than 2500 characters, you can view the full output in your Amazon S3 bucket. For more information, see Create a Bucket.

13. (Optional) Choose **Enable SNS notifications** if you want to receive notifications about the status of the commands you execute with Run Command. For more information, see Configuring Amazon SNS Notifications for Run Command (p. 169).

    **Note**
    After you specify parameters and options for your SSM document, expand the AWS Command Line Interface command section. This section includes a reusable command for different command-line platforms.

14. Choose **Run**, and then choose **View results**.

15. In the commands list, choose the command you just executed. If the command is still in progress, choose the refresh icon in the top right corner of the console.

16. When the **Status** column shows **Success** or **Failed**, choose the **Output** tab.

17. Choose **View Output**. The command output page shows the results of your command execution.

### Sending Commands to a Fleet

You can send commands to tens, hundreds, or thousands of instances by using the **targets** parameter (the **Select Targets by Specifying a Tag** option in the Amazon EC2 console). The **targets** parameter accepts a **Key,Value** combination based on Amazon EC2 tags that you specified for your instances. When you execute the command, the system locates and attempts to run the command on all instances that match the specified tags. For more information about Amazon EC2 tags, see Tagging Your Amazon EC2 Resources in the Amazon EC2 User Guide (content applies to Windows and Linux instances).

    **Note**
    You can also use the **targets** parameter to target a list of specific instance IDs, as described in the next section.

To control command execution across hundreds or thousands of instances, Run Command also includes parameters for restricting how many instances can simultaneously process a request and how many errors can be thrown by a command before the command is terminated.

### Contents

- Targeting Multiple Instances (p. 178)
- Using Concurrency Controls (p. 179)
- Using Error Controls (p. 179)
Targeting Multiple Instances

You can execute a command and target instances by specifying either Amazon EC2 tags or instance IDs. The targets parameter uses the following syntax in the AWS CLI:

Example 1: Targeting Tags

/aws ssm send-command --document-name name --targets Key=tag:tag_name,Values=tag_value [...]

Note
Example commands in this section are truncated using [...].

Example 2: Targeting Instance IDs

/aws ssm send-command --document-name name --targets Key=instanceids,Values=ID1, ID2, ID3 [...]

If you tagged instances for different environments using a Key named Environment and Values of Development, Test, Pre-production and Production, then you could send a command to all of the instances in one of these environments by using the targets parameter with the following syntax:

/aws ssm send-command --document-name name --targets Key=tag:Environment,Values=Development [...]

You could target additional instances in other environments by adding to the Values list. Separate items using commas.

/aws ssm send-command --document-name name --targets Key=tag:Environment,Values=Development,Test,Pre-production [...]

Example: Refining your targets using multiple Key criteria

You can refine the number of targets for your command by including multiple Key criteria. If you include more than one Key criteria, the system targets instances that meet all of the criteria. The following command targets all instances tagged for the Finance Department and tagged for the database server role.

/aws ssm send-command --document-name name --targets Key=tag:Department,Values=Finance Key=tag:ServerRole,Values=Database [...]

Example: Using multiple Key and Value criteria

Expanding on the previous example, you can target multiple departments and multiple server roles by including additional items in the Values criteria.

/aws ssm send-command --document-name name --targets Key=tag:Department,Values=Finance,Marketing Key=tag:ServerRole,Values=WebServer,Database [...]

Example: Targeting tagged instances using multiple Values criteria

If you tagged instances for different environments using a Key named Department and Values of Sales and Finance, then you could send a command to all of the instances in these environments by using the targets parameter with the following syntax:
Executing Commands

aws ssm send-command --document-name name --targets Key=tag:Department,Values=Sales,Finance

**Note**
You can specify a maximum of 5 keys, and 5 values for each key.

If either a tag key (the tag name) or a tag value includes spaces, then you must enclose the tag key or the value in quotation marks, as show in the following examples.

Example 1: Spaces in Value tag.

aws ssm send-command --document-name name --targets Key=tag:OS,Values="Windows Server 2016 Nano"

Example 2: Spaces in tag key and Value.

aws ssm send-command --document-name name --targets Key="tag:Operating System",Values="Windows Server 2016 Nano"

Example 3: Spaces in one item in a list of Values.

aws ssm send-command --document-name name --targets Key=tag:Department,Values="Sales","Finance","Systems Mgmt"

**Using Concurrency Controls**

You can control how many servers execute the command at the same time by using the `max-concurrency` parameter (the **Execute on** field in the Amazon EC2 console). You can specify either an absolute number of instances, for example 10, or a percentage of the target set, for example 10%. The queueing system delivers the command to a single instance and waits until the initial invocation completes before sending the command to two more instances. The system exponentially sends commands to more instances until the value of `max-concurrency` is met. The default for value `max-concurrency` is 50. The following examples show you how to specify values for the `max-concurrency` parameter:

aws ssm send-command --document-name name --max-concurrency 10 --targets Key=tag:Environment,Values=Development

aws ssm send-command --document-name name --max-concurrency 10% --targets Key=tag:Department,Values=Finance,Marketing Key=tag:ServerRole,Values=WebServer,Database

**Using Error Controls**

You can also control the execution of a command to hundreds or thousands of instances by setting an error limit using the `max-errors` parameters (the **Stop after __ errors** field in the Amazon EC2 console). The parameter specifies how many errors are allowed before the system stops sending the command to additional instances. You can specify either an absolute number of errors, for example 10, or a percentage of the target set, for example 10%. If you specify 3, for example, the system stops sending the command when the fourth error is received. If you specify 0, then the system stops sending the command to additional instances after the first error result is returned. If you send a command to 50 instances and set `max-errors` to 10%, then the system stops sending the command to additional instances when the sixth error is received.

Invocations that are already running a command when `max-errors` is reached are allowed to complete, but some of these invocations may fail as well. If you need to ensure that there won’t be more than max-
errors failed invocations, set max-concurrency to 1 so the invocations proceed one at a time. The default for max-concurrency is 50. The following examples show you how to specify values for the max-errors parameter:

```
aws ssm send-command --document-name name --max-errors 10 --targets Key=tag:Database,Values=Development [...]
```

```
aws ssm send-command --document-name name --max-errors 10% --targets Key=tag:Environment,Values=Development [...]
```

```
aws ssm send-command --document-name name --max-concurrency 1 --max-errors 1 --targets Key=tag:Environment,Values=Production [...]
```

### Canceling a Command

You can attempt to cancel a command as long as the service shows that it is in either a Pending or Executing state. However, even if a command is still in one of these states, we cannot guarantee that the command will be terminated and the underlying process stopped.

**Note**

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

**To cancel a command using the console**

1. In the navigation pane, choose Run Command.
2. Select the command invocation that you want to cancel.
3. Choose Actions, Cancel Command.

**To cancel a command using the AWS CLI**

Use the following command.

```
aws ssm cancel-command --command-id "command ID" --instance-ids "instance ID"
```

For information about the status of a cancelled command, see Setting Up Events and Notifications (p. 164).

### Run Command Walkthroughs

The walkthroughs in this section show you how to execute commands with Run Command using either the AWS Command Line Interface or AWS Tools for Windows PowerShell.

**Contents**

- Systems Manager Run Command Walkthrough Using the AWS CLI (p. 181)
- Systems Manager Run Command Walkthrough Using the AWS Tools for Windows PowerShell (p. 183)

You can also view sample commands in the following references.

- Systems Manager AWS CLI Reference
Systems Manager Run Command Walkthrough Using the AWS CLI

The following sample walkthrough shows you how to use the AWS CLI to view information about commands and command parameters, how to execute commands, and how to view the status of those commands.

Important
Only trusted administrators should be allowed to use Systems Manager pre-configured documents shown in this topic. The commands or scripts specified in Systems Manager documents run with administrative privilege on your instances. If a user has permission to execute any of the pre-defined Systems Manager documents (any document that begins with AWS), then that user also has administrator access to the instance. For all other users, you should create restrictive documents and share them with specific users. For more information about restricting access to Run Command, see Configuring Access to Systems Manager (p. 8).

Step 1: Getting Started

You must either have administrator privileges on the instances you want to configure or you must have been granted the appropriate permission in IAM. Also note, this example uses the us-east-1 region. Run Command is currently available in the following Systems Manager regions. For more information, see Systems Manager Prerequisites (p. 4).

To execute commands using the AWS CLI

1. Run the following command to specify your credentials and the region.

   ```
   aws configure
   ```

2. The system prompts you to specify the following.

   ```
   AWS Access Key ID [None]: key_name
   AWS Secret Access Key [None]: key_name
   Default region name [None]: us-east-1
   Default output format [None]: ENTER
   ```

3. List all available documents

   This command lists all of the documents available for your account based on IAM permissions. The command returns a list of Linux and Windows documents.

   ```
   aws ssm list-documents
   ```

4. Verify that an instance is ready to receive commands

   The output of the following command shows if instances are online.

   ```
   aws ssm describe-instance-information --output text --query "InstanceInformationList[*]"
   ```

5. Use the following command to view details about a particular instance.

   Note
   To execute the commands in this walkthrough, you must replace the instance and command IDs. The command ID is returned as a response of the send-command. The instance ID is available from the Amazon EC2 console.
Step 2: Running Shell Scripts

Using Run Command and the AWS-RunShellScript document, you can execute any command or script on an EC2 instance as if you were logged on locally.

To view the description and available parameters

- Use the following command to view a description of the Systems Manager JSON document.

```bash
aws ssm describe-document --name "AWS-RunShellScript" --query 
"[Document.Name,Document.Description]"
```

- Use the following command to view the available parameters and details about those parameters.

```bash
aws ssm describe-document --name "AWS-RunShellScript" --query "Document.Parameters[*]"
```

Step 3: Send a Command Using the AWS-RunShellScript document - Example 1

Use the following command to get IP information for an instance.

```bash
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" you will need to execute this command again to see the response data.

```bash
aws ssm list-command-invocations --command-id "command ID" --details
```

Step 4: Send a Command Using the AWS-RunShellScript document - Example 2

The following command displays the default user account running the commands.

```bash
sh_command_id=$(aws ssm send-command --instance-ids "instance ID" --document-name "AWS-RunShellScript" --comment "Demo run shell script on Linux Instance" --parameters commands=whoami --output text --query "Command.CommandId")
```

Get command status

The following command uses the Command ID to get the status of the command execution on the instance. This example uses the Command ID that was returned in the previous command.

```bash
aws ssm list-commands --command-id $sh_command_id
```

Get command details

The following command uses the Command ID from the previous command to get the status of the command execution on a per instance basis.
aws ssm list-command-invocations --command-id $sh_command_id --details

**Get command information with response data for a specific instance**

The following command returns the output of the original aws ssm send-command for a specific instance.

aws ssm list-command-invocations --instance-id instance ID --command-id $sh_command_id --details

### Step 5: Additional Examples

The following command returns the version of Python running on an instance.

```
sh_command_id=$(aws ssm send-command --instance-ids instance ID --document-name "AWS-RunShellScript" --comment "Demo run shell script on Linux Instances" --parameters commands='python' --version --output text --query "Command.CommandId")
```

The following command executes a Python script using Run Command.

```
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"
```

**Systems Manager Run Command Walkthrough Using the AWS Tools for Windows PowerShell**

The following examples show how to use the Tools for Windows PowerShell to view information about commands and command parameters, how to execute commands, and how to view the status of those commands. This walkthrough includes an example for each of the pre-defined Systems Manager documents.

**Important**

Only trusted administrators should be allowed to use Systems Manager pre-configured documents shown in this topic. The commands or scripts specified in Systems Manager documents run with administrative privilege on your instances. If a user has permission to execute any of the pre-defined Systems Manager documents (any document that begins with AWS), then that user also has administrator access to the instance. For all other users, you should create restrictive documents and share them with specific users. For more information about restricting access to Run Command, see Configuring Access to Systems Manager (p. 8).

**Configure AWS Tools for Windows PowerShell Session Settings**

Open AWS Tools for Windows PowerShell on your local computer and execute the following command to specify your credentials. You must either have administrator privileges on the instances you want to configure or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

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

Execute the following command to set the region for your PowerShell session. The example uses the us-east-1 region. Run Command is currently available in the following Systems Manager regions.

```
Set-DefaultAWSRegion -Region us-east-1
```
List all Available Documents

This command lists all of the documents available for your account:

```
Get-SSMDocumentList
```

Run PowerShell Commands or Scripts

Using Run Command and the AWS-RunPowerShell document, you can execute any command or script on an EC2 instance as if you were logged onto the instance using Remote Desktop. You can issue commands or type in a path to a local script to execute the command.

View the description and available parameters

```
Get-SSMDocumentDescription -Name "AWS-RunPowerShellScript"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-RunPowerShellScript" | select -ExpandProperty Parameters
```

Send a command using the AWS-RunPowerShellScript document

The following command shows the contents of the C:\Users directory and the contents of the C:\ directory on two instances.

```
$runPSCommand=Send-SSMCommand -InstanceId @('Instance-ID', 'Instance-ID') -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 Command ID to get the status of the command execution on both instances. This example uses the Command ID that was returned in the previous command.

```
Get-SSMCommand -CommandId $runPSCommand.CommandId
```

The status of the command in this example can be Success, Pending, or InProgress.

Get command information per instance

The following command uses the command ID from the previous command to get the status of the command execution on a per instance basis.

```
Get-SSMCommandInvocation -CommandId $runPSCommand.CommandId
```

Get command information with response data for a specific instance

The following command returns the output of the original Send-SSMCommand for a specific instance.

```
Get-SSMCommandInvocation -CommandId $runPSCommand.CommandId -Details $true -InstanceId Instance-ID | select -ExpandProperty CommandPlugins
```

Cancel a command

The following command cancels the Send-SSMCommand for the AWS-RunPowerShellScript document.
Check the command status

The following command checks the status of the Cancel command:

```powershell
Get-SSMCommand -CommandId $cancelCommandResponse.CommandId
```

Install an Application Using the AWS-InstallApplication Document

Using Run Command and the AWS-InstallApplication document, you can install, repair, or uninstall applications on instances. The command requires the path or address to an MSI.

View the description and available parameters:

```powershell
Get-SSMDocumentDescription -Name "AWS-InstallApplication"
```

View more information about parameters:

```powershell
Get-SSMDocumentDescription -Name "AWS-InstallApplication" | select -ExpandProperty Parameters
```

Send a command using the AWS-InstallApplication document:

The following command installs a version of Python on your instance in unattended mode, and logs the output to a local text file on your C: drive.

```powershell
$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 instance:

The following command uses the Command ID to get the status of the command execution.

```powershell
Get-SSMCommandInvocation -CommandId $installAppCommand.CommandId -Details $true
```

Get command information with response data for a specific instance:

The following command returns the results of the Python installation.

```powershell
Get-SSMCommandInvocation -CommandId $installAppCommand.CommandId -Details $true -InstanceId Instance-ID | select -ExpandProperty CommandPlugins
```

Install a PowerShell Module Using the AWS-InstallPowerShellModule JSON Document

You can use Run Command to install PowerShell modules on an EC2 instance. For more information about PowerShell modules, see [Windows PowerShell Modules](#).

View the description and available parameters:
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 Amazon S3 bucket named demo-ssm-output-bucket.

```powershell
$installPSCommand = Send-SSMCommand -InstanceId Instance-ID -DocumentName AWS-InstallPowerShellModule -Parameter @{'source'='https://gallery.technet.microsoft.com/EZOut-33ae0fb7/file/110351/1/EZOut.zip';'commands'=@('Add-WindowsFeature -name XPS-Viewer -restart')} -OutputS3BucketName demo-ssm-output-bucket
```

Get command information per instance

The following command uses the Command ID to get the status of the command execution.

```powershell
Get-SSMCommandInvocation -CommandId $installPSCommand.CommandId -Details $true
```

Get command information with response data for the instance

The following command returns the output of the original Send-SSMCommand for the specific command ID.

```powershell
Get-SSMCommandInvocation -CommandId $installPSCommand.CommandId -Details $true | select -ExpandProperty CommandPlugins
```

Join an Instance to a Domain Using the AWS-JoinDirectoryServiceDomain JSON Document

Using Run Command, you can quickly join an instance to an AWS Directory Service domain. Before executing this command you must create a directory. We also recommend that you learn more about the AWS Directory Service. For more information, see What Is AWS Directory Service?.

Currently you can only join an instance to a domain. You cannot remove an instance from a domain.

Join an instance to a domain

The following command joins the instance to the given AWS Directory Service domain and uploads any generated output to the Amazon S3 bucket.

```powershell
$domainJoinCommand = Send-SSMCommand -InstanceId Instance-ID -DocumentName AWS-JoinDirectoryServiceDomain -Parameter @{directoryId='d-9067386b64';
```
Get command information per instance
The following command uses the Command ID to get the status of the command execution.

```powershell
Get-SSMCommandInvocation -CommandId $domainJoinCommand.CommandId -Details $true
```

Get command information with response data for the instance
This command returns the output of the original Send-SSMCommand for the specific command ID.

```powershell
Get-SSMCommandInvocation -CommandId $domainJoinCommand.CommandId -Details $true | select -ExpandProperty CommandPlugins
```

Send Windows Metrics to Amazon CloudWatch using the AWS-ConfigureCloudWatch document
You can send Windows Server messages in the application, system, security, and Event Tracing for Windows (ETW) logs to Amazon CloudWatch Logs. When you enable logging for the first time, Systems Manager sends all logs generated within 1 minute from the time that you start uploading logs for the application, system, security, and ETW logs. Logs that occurred before this time are not included. If you disable logging and then later re-enable logging, Systems Manager sends logs from the time it left off. For any custom log files and Internet Information Services (IIS) logs, Systems Manager reads the log files from the beginning. In addition, Systems Manager can also send performance counter data to Amazon CloudWatch.

If you previously enabled CloudWatch integration in EC2Config, the Systems Manager settings override any settings stored locally on the instance in the C:\Program Files\Amazon\EC2ConfigService\Settings\AWS.EC2.Windows.CloudWatch.json file. For more information about using EC2Config to manage performance counters and logs on single instance, see Sending Performance Counters to CloudWatch and Logs to CloudWatch Logs.

View the description and available parameters

```powershell
Get-SSMDocumentDescription -Name "AWS-ConfigureCloudWatch"
```

View more information about parameters

```powershell
Get-SSMDocumentDescription -Name "AWS-ConfigureCloudWatch" | select -ExpandProperty Parameters
```

Send Application Logs to CloudWatch
The following command configures the instance and moves Windows Applications logs to CloudWatch.

```powershell
```

187
Get command information per instance

The following command uses the Command ID to get the status of the command execution.

```powershell
Get-SSMCommandInvocation -CommandId $cloudWatchCommand.CommandId -Details $true
```

Get command information with response data for a specific instance

The following command returns the results of the Amazon CloudWatch configuration.

```powershell
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 Documentation.

```powershell
```

Enable/Disable Windows Automatic Update Using the AWS-ConfigureWindowsUpdate document

Using Run Command and the AWS-ConfigureWindowsUpdate document, you can enable or disable automatic Windows updates on your Windows instances. This command configures the Windows update agent to download and install Windows updates on the day and hour that you specify. If an update requires a reboot, the computer reboots automatically 15 minutes after updates have been installed.

With this command you can also configure Windows update to check for updates but not install them.


View the description and available parameters

```powershell
Get-SSMDocumentDescription -Name "AWS-ConfigureWindowsUpdate"
```

View more information about parameters

```powershell
Get-SSMDocumentDescription -Name "AWS-ConfigureWindowsUpdate" | select -ExpandProperty Parameters
```

Enable Windows automatic update

The following command configures Windows Update to automatically download and install updates daily at 10:00 pm.

```powershell
#configureWindowsUpdateCommand = Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-ConfigureWindowsUpdate' -Parameters @('updateLevel'='InstallUpdatesAutomatically'; 'scheduledInstallDay'='Daily'; 'scheduledInstallTime'='22:00')
```
View command status for enabling Windows automatic update

The following command uses the Command ID to get the status of the command execution for enabling Windows Automatic Update.

```
Get-SSMCommandInvocation -Details $true -CommandId $configureWindowsUpdateCommand.CommandId | select -ExpandProperty CommandPlugins
```

Disable Windows automatic update

The following command lowers the Windows Update notification level so the system checks for updates but does not automatically update the instance.

```
$configureWindowsUpdateCommand = Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-ConfigureWindowsUpdate' -Parameters @{$'updateLevel'='NeverCheckForUpdates'}
```

View command status for disabling Windows automatic update

The following command uses the Command ID to get the status of the command execution for disabling Windows automatic update.

```
Get-SSMCommandInvocation -Details $true -CommandId $configureWindowsUpdateCommand.CommandId | select -ExpandProperty CommandPlugins
```

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 instances. This command can update the EC2Config service to the latest version or a version you specify.

View the description and available parameters

```
Get-SSMDocumentDescription -Name "AWS-UpdateEC2Config"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-UpdateEC2Config" | select -ExpandProperty Parameters
```

Update EC2Config to the latest version

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName "AWS-UpdateEC2Config"
```

Get command information with response data for the instance

This command returns the output of the specified command from the previous Send-SSMCommand:

```
Get-SSMCommandInvocation -CommandId ID -Details $true -InstanceId Instance-ID | select -ExpandProperty CommandPlugins
```

Update EC2Config to a specific version

The following command will downgrade EC2Config to an older version:

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName "AWS-UpdateEC2Config" -Parameter @{$'version'='3.8.354'; 'allowDowngrade'='true'}
```
Manage Windows Updates Using Run Command

Run Command includes three documents to help you manage updates for Amazon EC2 Windows instances.

- **AWS-FindWindowsUpdates** — Scans an instance and determines which updates are missing.
- **AWS-InstallMissingWindowsUpdates** — Installs missing updates on your EC2 instance.
- **AWS-InstallSpecificUpdates** — Installs a specific update.

The following examples demonstrate how to perform the specified Windows Update management tasks.

**Search for all missing Windows updates**

```powershell
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-FindWindowsUpdates' -Parameters @{'UpdateLevel'='All'}
```

**Install specific Windows updates**

```powershell
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-InstallSpecificUpdates' -Parameters @{'KbArticleIds'='123456,KB567890,987654'}
```

**Install important missing Windows updates**

```powershell
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-InstallMissingWindowsUpdates' -Parameters @{'UpdateLevel'='Important'}
```

**Install missing Windows updates with specific exclusions**

```powershell
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-InstallMissingWindowsUpdates' -Parameters @{'UpdateLevel'='All';'ExcludeKbArticleIds'='KB567890,987654'}
```

Troubleshooting Systems Manager Run Command

Use the following information to help troubleshoot problems with Run Command.

**Where Are My Instances?**

If you do not see the expected list of instances when you choose **Select Target instances** then verify the following.

- You installed the latest version of the SSM Agent on your instance. Amazon EC2 Windows Amazon Machine Images (AMIs) are pre-configured with the SSM Agent. Linux AMIs are not. For information about installing the SSM Agent on an instance, see *Installing and Configuring SSM Agent on Linux Instances (p. 15)* (for Linux) or *Installing and Configuring SSM Agent on Windows Instances (p. 13)* (for Windows).
- Your instance is configured with an AWS Identity and Access Management (IAM) role that enables the instance to communicate with the Systems Manager API. Also verify that your user account has an IAM user trust policy that enables your account to communicate with the Systems Manager API. For more information, see *Configuring Access to Systems Manager (p. 8)*.

**Check Instance Status Using the Health API**

You can use the Amazon EC2 Health API to quickly determine the following information about Amazon EC2 instances:
• The status of one or more instances
• The last time the instance sent a heartbeat value
• The version of the SSM Agent
• The operating system
• The version of the EC2Config service (Windows)
• The status of the EC2Config service (Windows)

Getting Status Information on Windows Instances

Use the following command to get status details about one or more instances:

```powershell
Get-SSMInstanceInformation -InstanceInformationFilterList @{Key="InstanceIds";ValueSet="instance-ID","instance-ID"}
```

Use the following command with no filters to see all instances registered to your account that are currently reporting an online status. Substitute the ValueSet="Online" with "ConnectionLost" or "Inactive" to view those statuses:

```powershell
Get-SSMInstanceInformation -InstanceInformationFilterList @{Key="PingStatus";ValueSet="Online"}
```

Use the following command to see which instances are running the latest version of the EC2Config service. Substitute ValueSet="LATEST" with a specific version (for example, 3.0.54 or 3.10) to view those details:

```powershell
Get-SSMInstanceInformation -InstanceInformationFilterList @{Key="AgentVersion";ValueSet="LATEST"}
```

Getting Status Information on Linux Instances

Use the following command to get status details about one or more instances:

```bash
aws ssm describe-instance-information --instance-information-filter-list key=InstanceIds,valueSet=instance-ID
```

Use the following command with no filters to see all instances registered to your account that are currently reporting an online status. Substitute the ValueSet="Online" with "ConnectionLost" or "Inactive" to view those statuses:

```bash
aws ssm describe-instance-information --instance-information-filter-list key=PingStatus,valueSet=Online
```

Use the following command to see which instances are running the latest version of the SSM Agent. Substitute ValueSet="LATEST" with a specific version (for example, 1.0.145 or 1.0) to view those details:

```bash
aws ssm describe-instance-information --instance-information-filter-list key=AgentVersion,valueSet=LATEST
```

If the describe-instance-information API operation returns an AgentStatus of Online, then your instance is ready to be managed using Run Command. If the status is Inactive, the instance has one or more of the following problems.

191
Troubleshooting the SSM Agent

If you experience problems executing commands using Run Command, there might be a problem with the SSM Agent. Use the following information to help you troubleshoot the agent.

View Agent Logs

The SSM Agent logs information in the following files. The information in these files can help you troubleshoot problems.

**On Windows**

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

**Note**

The default filename of the seelog is seelog.xml.template. If you modify a seelog, you must rename the file to seelog.xml.

**On Linux**

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

On Linux, you can enable extended logging by updating the seelog.xml file. By default, the configuration file is located here: /etc/amazon/ssm/seelog.xml.

For more information about cihub/seelog configuration, go to the cihub/seelog Wiki. For examples of cihub/seelog configurations, go to cihub/seelog examples.

AWS Systems Manager Patch Manager

AWS Systems Manager Patch Manager automates the process of patching managed instances. You can scan instances to see a report of missing patches, or you can scan and automatically install all missing patches. Patch Manager uses patch baselines that include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. You can install patches on a regular basis by scheduling patching to run as a Systems Manager Maintenance Window task. You can also install patches individually or to large groups of instances by using Amazon EC2 tags.

Patch Manager integrates with AWS Identity and Access Management (IAM), AWS CloudTrail, and Amazon CloudWatch Events to provide a secure patching experience that includes event notifications and the ability to audit usage.

Supported Operating Systems

Patch Manager can patch the following operating systems:
Important
AWS does not test patches for Windows or Linux before making them available in Patch Manager.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>64-Bit and 32-Bit Systems</td>
</tr>
<tr>
<td></td>
<td>• Amazon Linux 2012.03 - 2017.03</td>
</tr>
<tr>
<td></td>
<td>• Ubuntu Server 14.04 LTS, or 16.04 LTS</td>
</tr>
<tr>
<td></td>
<td>• Red Hat Enterprise Linux (RHEL) 6.5 - 6.9</td>
</tr>
<tr>
<td></td>
<td>64-Bit Systems Only</td>
</tr>
<tr>
<td></td>
<td>• Amazon Linux 2015.03 - 2017.09</td>
</tr>
<tr>
<td></td>
<td>• Red Hat Enterprise Linux (RHEL) 7.0 - 7.4</td>
</tr>
<tr>
<td>Windows</td>
<td>Windows Server 2008 through Windows Server 2016, including R2 versions. Patch Manager provides all patches for supported operating systems within hours of their being made available by Microsoft.</td>
</tr>
</tbody>
</table>

Getting Started with Patch Manager

To get started with Patch Manager, complete the tasks described in the following table.

<table>
<thead>
<tr>
<th>Task</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify Systems Manager prerequisites</td>
<td>Systems Manager Prerequisites (p. 4)</td>
</tr>
<tr>
<td>Learn how to set up and configure patching</td>
<td>Working with Patch Manager (p. 193)</td>
</tr>
<tr>
<td>Configure permissions for Maintenance Windows (Required if you intend to use this feature when patching.)</td>
<td>Controlling Access to Maintenance Windows (p. 219)</td>
</tr>
<tr>
<td>Create patch baselines, patch groups, and a Maintenance Window to execute patching in a test environment</td>
<td>Systems Manager Patch Manager Walkthroughs (p. 198)</td>
</tr>
</tbody>
</table>

Working with Patch Manager

To use Patch Manager, complete the following tasks. These tasks are described in more detail in this section.

1. Verify that the default patch baselines meet your needs, or create patch baselines that define a standard set of patches for your instances.
2. Organize instances into patch groups by using Amazon EC2 tags (optional, but recommended).
3. Schedule patching by using a Maintenance Window that defines which instances to patch and when to patch them.
4. Monitor patching to verify compliance and investigate failures.

Topics

- Verify Default Patch Baselines or Create Your Own (p. 194)
- Organize Instances into Patch Groups (p. 197)
- Schedule Patch Updates Using a Maintenance Window (p. 197)

Related Content

- To view an example of how to create a patch baseline, patch groups, and a Maintenance Window, see Systems Manager Patch Manager Walkthroughs (p. 198).
- For more information about Maintenance Windows, see AWS Systems Manager Maintenance Windows (p. 218).
- For information about monitoring patch compliance, see About Patch Compliance (p. 87).

Verify Default Patch Baselines or Create Your Own

A patch baseline defines which patches are approved for installation on your instances. You can specify approved or rejected patches one by one. You can also create auto-approval rules to specify that certain types of updates (for example, critical updates) should be automatically approved. The rejected list overrides both the rules and the approve list.

To use a list of approved patches to install specific packages, you first remove all auto-approval rules. If you explicitly identify a patch as rejected, it will not be approved or installed, even if it matches all of the criteria in an auto-approval rule. Also, a patch is installed on an instance only if it applies to the software on the instance, even if the patch has otherwise been approved for the instance.

Topics

- Pre-Defined vs. Custom Baselines (p. 194)
- Important Differences Between Windows and Linux Patching (p. 196)

Pre-Defined vs. Custom Baselines

Patch Manager provides pre-defined patch baselines for each of the operating systems supported by Patch Manager. You can use these baselines as they are currently configured (you can't customize them) or you can create your own patch baselines if you want greater control over which patches are approved or rejected for your environment.

Topics

- Pre-Defined Baselines (p. 194)
- Custom Baselines (p. 195)

Pre-Defined Baselines

The following table describes the pre-defined patch baselines provided with Patch Manager.

<table>
<thead>
<tr>
<th>Name</th>
<th>Supported Products</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS-DefaultPatchBaseline</td>
<td>Windows (Windows Server 2008 – 2016)</td>
<td>Approves all operating system patches with a classification of CriticalUpdates or SecurityUpdates and an MSRC</td>
</tr>
</tbody>
</table>
### Custom Baselines

If you create your own patch baseline, you can choose which patches to auto-approve by using the following categories.

- Operating system: Windows, Amazon Linux, Ubuntu, etc.
- Product name: For example, RHEL 6.5, Amazon Linux 2014.09, Windows Server 2012, Windows Server 2012 R2, etc.
- Classification: For example, critical updates, security updates, etc.
- Severity: For example, critical, important, etc.

For each auto-approval rule that you create, you can specify an auto-approval delay. This delay is the number of days to wait after the patch was released, before the patch is automatically approved for patching. For example, if you create a rule using the Critical Updates classification and configure it for seven days auto-approval delay, then a new critical patch released on January 7 will automatically be approved on January 14.

**Note**  
If a Linux repository doesn't provide release date information for packages, Systems Manager treats the Auto Approval Delay as having a value of zero.

You can also specify a compliance severity level. If an approved patch is reported as missing, the **Compliance Level** is the severity of the compliance violation.

By using multiple patch baselines with different auto-approval delays, you can deploy patches at different rates to different instances. For example, you can create separate patch baselines and auto-approval delays for development and production environments. This enables you to test patches in your development environment before they get deployed in your production environment.
Keep the following in mind when you create a patch baseline:

- Patch Manager provides a default patch baseline for each supported operating system. You can instead create your own patch baseline and designate that as the default patch baseline for the corresponding operating system.
- For on-premises or non-Amazon EC2 instances, Patch Manager attempts to use your custom default patch baseline. If no custom default patch baseline exists, the system uses the pre-defined patch baseline for the corresponding operating system.
- If a patch is listed as both approved and rejected in the same patch baseline, the patch is rejected.
- An instance can have only one patch baseline defined for it.
- When you create patch baselines for Amazon Linux and RHEL, if you specify Approved Patches, be aware that Systems Manager supports Bugzilla ID, CVE ID, Advisory ID, and package-name wildcards. If you specify Rejected Patches, Systems Manager only supports package-name wildcards. For Ubuntu baselines, only full package names are supported in both the Approved and Rejected Patches fields.

To view an example of how to create a patch baseline by using the Systems Manager console or the AWS CLI, see Systems Manager Patch Manager Walkthroughs (p. 198).

**Important Differences Between Windows and Linux Patching**

The following table describes important differences between Windows and Linux patching.

<table>
<thead>
<tr>
<th>Difference</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patch evaluation</td>
<td>Patch Manager uses a different process to evaluate which patches should be present on Windows managed instances versus Linux managed instance. 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). For Linux patching, Systems Manager evaluates patch baseline rules and the list of approved and rejected patches on each managed instance. Systems Manager must evaluate patching on each instance because the service retrieves the list of known patches and updates from the repositories that are configured on the instance.</td>
</tr>
<tr>
<td>Not Applicable patches</td>
<td>Due to the large number of available packages for Linux operating systems, Systems Manager does not report details about patches in the Not Applicable state. A Not Applicable patch is, for example, a patch for Apache software when the instance does not have Apache installed. Systems Manager does report the number of Not Applicable patches in the summary, but if you call the DescribeInstancePatches API for an instance, the returned data does not include patches with a state of Not Applicable. This behavior is different from Windows.</td>
</tr>
</tbody>
</table>
Note
Also note the following changes and requirements for Linux patching:

- 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 the SSM Agent in Executing Commands from the EC2 Console (p. 174).
- The AWS-ApplyPatchBaseline SSM document is being replaced by the AWS-RunPatchBaseline document.

Organize Instances into Patch Groups

A *patch group* is an optional means of organizing instances for patching. For example, you can create patch groups for different operating systems (Linux or Windows), different environments (Development, Test, and Production), or different server functions (web servers, file servers, databases). Patch groups can help you avoid deploying patches to the wrong set of instances. They can also help you avoid deploying patches before they have been adequately tested.

You create a patch group by using Amazon EC2 tags. Unlike other tagging scenarios across Systems Manager, a patch group must be defined with the tag key: **Patch Group**. Note that the key is case sensitive. You can specify any value, for example "web servers," but the key must be **Patch Group**.

Note
An instance can only be in one patch group.

After you create a patch group and tag instances, you can register the patch group with a patch baseline. By registering the patch group with a patch baseline, you ensure that the correct patches are installed during the patching execution.

When the system executes the task to apply a patch baseline to an instance, the service checks to see if a patch group is defined for the instance. If the instance is assigned to a patch group, the system then checks to see which patch baseline is registered to that group. If a patch baseline is found for that group, the system applies the patch baseline. If an instance isn't configured for a patch group, the system automatically uses the currently configured default patch baseline.

For example, let's say an instance is tagged with key=Patch Group and value=Front-End Servers. When Patch Manager executes the AWS-RunPatchBaseline task on that instance, the service checks to see which patch baseline is registered with Front-End Servers. If a patch baseline is found, the system uses that baseline. If no patch baseline is registered for Front-End Servers, the system uses the default patch baseline.

To view an example of creating a patch baseline and patch groups by using the AWS CLI, see Walkthrough: Patch a Server Environment (AWS CLI) (p. 203). For more information about Amazon EC2 tags, see Tagging Your Amazon EC2 Resources in the Amazon EC2 User Guide.

Schedule Patch Updates Using a Maintenance Window

After you configure a patch baseline (and optionally a patch group), you can apply patches to your instance by using a Maintenance Window. A Maintenance Window can reduce the impact on server availability by letting you specify a time to perform the patching process that doesn't interrupt business operations. A Maintenance Window works like this:

1. Create a Maintenance Window with a schedule for your patching operations.
2. Choose the targets for the Maintenance Window by specifying the **Patch Group** tag for the tag name, and any value for which you have defined Amazon EC2 tags, for example, "production servers".
3. Create a new Maintenance Window task, and specify the AWS-RunPatchBaseline document.
When you configure the task, you can choose to either scan instances or scan and patch instances. If you choose to scan instances, Patch Manager scans each instance and generates a list of missing patches for you to review.

If you choose to scan and patch, Patch Manager scans each instance and compares the list of installed patches against the list of approved patches in the baseline. Patch Manager identifies missing patches, and then downloads and installs all missing and approved patches.

If you want to perform a one-time scan or install to fix an issue, you can use Run Command to call the AWS-RunPatchBaseline document directly.

Important
After installing patches, Systems Manager reboots each instance. The reboot is required to make sure that patches are installed correctly and to ensure that the system did not leave the instance in a potentially bad state.

Systems Manager Patch Manager Walkthroughs

The following walkthroughs show how to use the Systems Manager console and the AWS CLI to create patch baselines, patch groups, and Maintenance Windows to execute patching.

Contents
- Walkthrough: Patch a Server Environment (Console) (p. 198)
- Walkthrough: Patch a Server Environment (AWS CLI) (p. 203)

Walkthrough: Patch a Server Environment (Console)

The following walkthrough describes how to patch a server environment in the Systems Manager console by using a default patch baseline, patch groups, and a Maintenance Window. To learn more about the processes described in this walkthrough, see Working with Patch Manager (p. 193).

Before You Begin

Install or update the SSM Agent on your instances. To patch Linux instances, your instances must be running SSM Agent version 2.0.834.0 or later. For information about updating the agent, see the section titled Example: Update the SSM Agent in Executing Commands from the EC2 Console (p. 174).

In addition, the following walkthrough executes patching during a Maintenance Window. You must configure roles and permissions for Maintenance Windows before you begin. For more information, see Controlling Access to Maintenance Windows (p. 219).

Topics
- Create a Default Patch Baseline (p. 198)
- Add Instances to a Patch Group (p. 200)
- Create a Maintenance Window for Patching (p. 200)

Create a Default Patch Baseline

Patch Manager includes a default patch baseline for each operating system supported by Patch Manager. You can use these default patch baselines (you can't customize them), or you can create your own. The following procedure describes how to view the default patch baselines to see if they meet or you needs. The procedure also describes how to create your own default patch baseline. To learn more about patch baselines, see Verify Default Patch Baselines or Create Your Own (p. 194).
Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

**To create a default patch baseline (AWS Systems Manager)**

2. In the navigation pane, choose Patch Manager.
   -or-
   If the AWS Systems Manager home page opens, scroll down and choose Explore Patch Manager.
3. In the patch baselines list, choose the name of a patch baseline for the operating system you want to patch.
4. Choose the Approval rules tab.
   If the auto-approval rules are acceptable for your instances, then you can skip to the next procedure, Add Instances to a Patch Group (p. 200).
   -or-
   To create your own default patch baseline, in the navigation pane, choose Patch Manager, and then choose Create patch baseline.
5. In the Name field, type a name for your new patch baseline, for example, RHEL-Default.
6. (Optional) Type a description for this patch baseline.
7. In the Operating system list, choose an operating system, for example, Red Hat Enterprise Linux.
8. In the Approval rules section, use the fields to create one or more auto-approval rules.
   
   **Note**
   
   If an approved patch is reported as missing, the option you choose in Compliance level, such as Critical or Medium, determines the severity of the compliance violation.
9. In the Patch exceptions section, enter comma-separated lists of patches you want to explicitly approve and reject for the baseline. For approved patches, choose a corresponding compliance severity level.
10. Choose Create patch baseline.

**To create a default patch baseline (Amazon EC2 Systems Manager)**

1. Open the Amazon EC2 console, expand Systems Manager Services in the navigation pane, and then choose Patch Baselines.
2. In the patch baselines list, choose a patch baseline for the operating system you want to patch.

   **Note**
   
   If the Welcome to EC2 Systems Manager - Patch Baselines page appears, choose Create Patch Baseline. When the Create patch baseline page appears, choose the back button in your browser to view the list of patch baselines.
3. With a default baseline selected, choose the Approval Rules tab. If the auto-approval rules are acceptable for your instances, then you can skip to the next procedure, Add Instances to a Patch Group (p. 200).
4. To create your own default patch baseline, choose Create Patch Baseline.
5. In the Name field, type a name for your new patch baseline, for example, RHEL-Default.
6. (Optional) Type a description for this patch baseline.
7. In the Operating System field, choose an operating system, for example, RedhatEnterpriseLinux.
8. In the **Approval Rules** section, use the fields to create one or more auto-approval rules.

   **Note**
   If an approved patch is reported as missing, the option you choose in **Compliance level**, such as **Critical** or **Medium**, determines the severity of the compliance violation.

9. In the **Patch Exceptions** section, enter comma-separated lists of patches you want to explicitly approve and reject for the baseline. For approved patches, choose a corresponding compliance severity level.

10. Choose **Create Patch Baseline**.

**Add Instances to a Patch Group**

To help you organize your patching efforts, we recommend that you add instances to patch groups by using Amazon EC2 tags. Patch groups require use of the tag key **Patch Group**. You can specify any value, but the tag key must be **Patch Group**. For more information about patch groups, see Organize Instances into Patch Groups (p. 197).

**To add instances to a patch group**

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. From the **Actions** menu, choose **Instance Settings, Add/Edit Tags**.
4. If the instance already has one or more tags applied, choose **Create Tag**.
5. In the **Key** field, type **Patch Group**.
6. In the **Value** field, type a value that helps you understand which instances will be patched.
7. Choose **Save**.
8. Repeat this procedure to add other instances to the same patch group.

**Create a Maintenance Window for Patching**

To minimize the impact on your server availability, we recommend that you configure a Maintenance Window to execute patching during times that won't interrupt your business operations. For more information about Maintenance Windows, see AWS Systems Manager Maintenance Windows (p. 218).

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

**To create a Maintenance Window for patching (AWS Systems Manager)**

2. In the navigation pane, choose **Maintenance Windows**.
   - or -
   If the AWS Systems Manager home page opens, scroll down and choose **Explore Maintenance Windows**.
3. Choose **Create maintenance window**.
4. In the **Name** field, type a name that designates this as a Maintenance Window for patching critical and important updates.
5. In the top of the **Schedule** section, choose the schedule options you want.
6. In the **Duration** field, type the number of hours you want the Maintenance Window to be active.
7. In the **Stop initiating tasks** field, type the number of hours before the Maintenance Window duration ends that you want the system to stop initiating new tasks.

8. Choose **Create maintenance window**.

9. In the Maintenance Windows list, choose the Maintenance Window you just created, and then choose **Actions, Register targets**.

10. (Optional) In the **Maintenance window target details** section, provide a name, a description, and owner information (your name or alias) for this target.

11. In the **Targets** section, choose **Specifying tags**.

12. Under Tag, enter a tag key and a tag value to identify the instances to register with the Maintenance Window.

13. Choose **Register target**. The system creates a Maintenance Window target.

14. In the details page of the Maintenance Window you created, choose **Actions, Register run command task**.

15. (Optional) In the **Maintenance window task details** section, provide a name and description for this task.

16. In the **Command document** list, choose **AWS-RunPatchBaseline**.

17. In the **Task priority** list, choose a priority. One is the highest priority.

18. In the **Targets** section, under **Target by**, choose the Maintenance Window target you created earlier in this procedure.

19. (Optional) In **Rate control**:

   - In **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.

   **Note**
   If you selected targets by choosing Amazon EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document at the same time by specifying a percentage.

   - In **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify 3 errors, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.

20. In the **Role** section, enter the ARN of a IAM role to which the **AmazonSSMMaintenanceWindowRole** is attached. For more information, see [Controlling Access to Maintenance Windows](p. 219).

21. In the **Output options** section, if you want to save the command output to a file, select the **Write command output to an Amazon S3 bucket**. Type the bucket and prefix names in the boxes.

22. 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 [Configuring Amazon SNS Notifications for Run Command](p. 169).

23. In the **Parameters** section:

   - In the **Operation** list, choose **Scan** to scan for missing patches, or choose **Install** to scan for and install missing patches.

   **Note**
   The **Install** operation causes the instance to reboot (if patches are installed). The **Scan** operations does not cause a reboot.

   - You don't need to specify anything in the **Snapshot Id** field. This system automatically generates and provides this parameter.

   - (Optional) In the **Comment** box, enter a tracking note or reminder about this command.

   - In the **Timeout (seconds)** box, enter the number of seconds the system should wait for the operation to finish before it is considered unsuccessful.
24. Choose **Register run command task**.

**To create a Maintenance Window for patching (Amazon EC2 Systems Manager)**

1. Open the Amazon EC2 console.
2. In the navigation pane, choose **Maintenance Windows**, and then choose **Create maintenance window**.
3. In the **Name** field, type a name that designates this as a Maintenance Window for patching critical and important updates.
4. In the **Specify schedule** area, choose the schedule options you want.
5. In the **Duration** field, type the number of hours you want the Maintenance Window to be active.
6. In the **Stop initiating tasks** field, type the number of hours before the Maintenance Window duration ends that you want the system to stop initiating new tasks.
7. Choose **Create maintenance window**.
8. In the Maintenance Window list, choose the Maintenance Window you just created, and then choose **Actions, Register targets**.
9. (Optional) Near the top of the page, specify a name, description, and owner information (your name or alias) for this target.
10. Next to **Select targets by**, choose **Specifying Tags**.
11. Next to **Tag**, use the lists to choose a tag key and a tag value.
12. Choose **Register targets**. The system creates a Maintenance Window target.
13. In the Maintenance Window list, choose the Maintenance Window you created with the procedure, and then choose **Actions, Register run command task**.
14. In the **Command Document** section of the **Register run command task** page, choose **AWS-RunPatchBaseline**.
15. In the **Task Priority** section, specify a priority. One is the highest priority.
16. In the **Targets** section, choose **Select**, and then choose the Maintenance Window target you created earlier in this procedure.
17. In the **Role** field, enter the ARN of a role which has the **AmazonSSMMaintenanceWindowRole** policy attached to it. For more information, see **Controlling Access to Maintenance Windows** (p. 219).
18. In the **Execute on** field, choose either **Targets** or **Percent** to limit the number of instances where the system can simultaneously perform patching operations.
19. In the **Stop after** field, specify the number of allowed errors before the system stops sending the patching task to other instances.
20. In the **Operation** list, choose **Scan** to scan for missing patches, or choose **Install** to scan for and install missing patches.

**Note**

The **Install** operation causes the instance to reboot (if patches are installed). The **Scan** operations does not cause a reboot.

21. You don’t need to specify anything in the **Snapshot Id** field. This system automatically generates and provides this parameter.
22. In the **Advanced** section:

- If you want to write command output and results to an Amazon S3 bucket, choose **Write to S3**. Type the bucket and prefix names in the boxes.
- 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 **Configuring Amazon SNS Notifications for Run Command** (p. 169).
23. Choose **Register task**.
After the Maintenance Window task completes, you can view patch compliance details in the Amazon EC2 console on the Managed Instances page. In the filter bar, use the AWS:PatchSummary and AWS:ComplianceItem filters.

**Note**
You can save your query by bookmarking the URL after you specify the filters.

You can also drill down on a specific instance by choosing the instance in the Managed Instances page, and then choose the Patch tab. You can also use the DescribePatchGroupState and DescribeInstancePatchStatesForPatchGroup APIs to view compliance details. For information about patch compliance data, see About Patch Compliance (p. 87).

### Walkthrough: Patch a Server Environment (AWS CLI)

The following procedure illustrates how a user might patch a server environment by using a custom patch baseline, patch groups, and a Maintenance Window.

For a sample of other AWS CLI commands you might use for your Patch Manager configuration tasks, see Additional Patch Manager CLI Commands (p. 208).

### Before You Begin

Install or update the SSM Agent on your instances. To patch Linux instances, your instances must be running SSM Agent version 2.0.834.0 or later. For information about updating the agent, see the section titled *Example: Update the SSM Agent* in Executing Commands from the EC2 Console (p. 174).

In addition, the following walkthrough executes patching during a Maintenance Window. You must configure roles and permissions for Maintenance Windows before you begin. For more information, see Controlling Access to Maintenance Windows (p. 219).

### To configure Patch Manager and patch instances by using the AWS CLI

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 a Region. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM).

   ```bash
   aws configure
   ```

   The system prompts you to specify the following.

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

3. (Windows) Execute the following command to create a patch baseline named "Production-Baseline" that approves patches for a production environment seven days after they are released.

   ```bash
   aws ssm create-patch-baseline --name "Production-Baseline" --operating-system "WINDOWS" --product "WindowsServer2012R2" --approval-rules "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,Important]}, {Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]]},ApproveAfterDays=7}]},ApproveAfterDays=7],ApproveAfterDays=7]
   --description "Baseline containing all updates approved for production systems"
   ```

   (Linux) Execute the following command to create a patch baseline named "Production-Baseline" that approves patches for a production environment seven days after they are released.
aws ssm create-patch-baseline --name "Production-Baseline" --operating-system "AMAZON_LINUX" --approval-rules "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=PRODUCT,Values=[AmazonLinux2016.03,AmazonLinux2016.09,AmazonLinux2017.03,AmazonLinux2017.09]},{Key=SEVERITY,Values=[Critical,Important]}, {Key=CLASSIFICATION,Values=[Security]}]},ApproveAfterDays=7}]" --description "Baseline containing all updates approved for production systems"

The system returns information like the following.

```
{
"BaselineId":"pb-034c8a5a84f030362"
}
```

4. Execute the following commands to register the "Production-Baseline" patch baseline for three patch groups named "Production," "Database Servers," and "Front-End Patch Group."

```
aws ssm register-patch-baseline-for-patch-group --baseline-id pb-034c8a5a84f030362 --patch-group "Production"
```

The system returns information like the following.

```
{
"PatchGroup":"Production",
"BaselineId":"pb-034c8a5a84f030362"
}
```

```
aws ssm register-patch-baseline-for-patch-group --baseline-id pb-034c8a5a84f030362 --patch-group "Database Servers"
```

The system returns information like the following.

```
{
"PatchGroup":"Database Servers",
"BaselineId":"pb-034c8a5a84f030362"
}
```

5. Execute the following commands to create two Maintenance Windows for the production servers. The first window run every Tuesday at 10 PM. The second window runs every Saturday at 10 PM.

```
aws ssm create-maintenance-window --name "Production-Tuesdays" --schedule "cron(0 0 22 ? * TUE *)" --duration 1 --cutoff 0 --no-allow-unassociated-targets
```

The system returns information like the following.

```
{
"WindowId":"mw-0c66948c711a3b5bd"
}
```

```
aws ssm create-maintenance-window --name "Production-Saturdays" --schedule "cron(0 0 22 ? * SAT *)" --duration 2 --cutoff 0 --no-allow-unassociated-targets
```

The system returns information like the following.

```
{
```

204
6. Execute the following commands to register the Production servers with the two production Maintenance Windows.

```bash
aws ssm register-target-with-maintenance-window --window-id mw-0c66948c711a3b5bd --targets "Key=tag:Patch Group,Values=Production" --owner-information "Production servers" --resource-type "INSTANCE"
```

The system returns information like the following.

```json
{
  "WindowTargetId": "557e7b3a-bc2f-48dd-ae05-e282b5b20760"
}
```

```bash
aws ssm register-target-with-maintenance-window --window-id mw-0c66948c711a3b5bd --targets "Key=tag:Patch Group,Values=Database Servers" --owner-information "Database servers" --resource-type "INSTANCE"
```

The system returns information like the following.

```json
{
  "WindowTargetId": "767b6508-f4ac-445e-b6fe-758cc912e55c"
}
```

7. Execute the following commands to register a patch task that only scans the production servers for missing updates in the first production Maintenance Window.

```bash
aws ssm register-task-with-maintenance-window --window-id mw-0c66948c711a3b5bd --targets "Key=WindowTargetIds,Values=557e7b3a-bc2f-48dd-ae05-e282b5b20760" --task-arn "AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role" --task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 1 --task-parameters '{"Operation":{"Values":[]}}'
```

The system returns information like the following.

```json
{
  "WindowTargetId": "faa01c41-1d57-496c-ba77-ff9cadba4b7d"
}
```
The system returns information like the following.

```json
{
  "WindowTaskId": "968e3b17-8591-4fb2-932a-b62389d6f635"
}
```

Execute the following commands to register a patch task that installs missing updates on the productions servers in the second Maintenance Window.

```bash
aws ssm register-task-with-maintenance-window --window-id mw-09e2a75baadd84e85 --targets "Key=WindowTargetIds,Values=767b6508-f4ac-445e-b6fe-758cc912e55c" --task-arn "AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role" --task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 1 --task-parameters '{"Operation":{"Values":["Install"]}}'
```

The system returns information like the following.

```json
{
  "WindowTaskId": "09f2e873-a3a7-443f-ba0a-05cf4de5a1c7"
}
```

8. Execute the following commands to register a patch task that installs missing updates on the productions servers in the second Maintenance Window.

```bash
aws ssm register-task-with-maintenance-window --window-id mw-09e2a75baadd84e85 --targets "Key=WindowTargetIds,Values=767b6508-f4ac-445e-b6fe-758cc912e55c" --task-arn "AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role" --task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 5 --task-parameters '{"Operation":{"Values":["Scan"]}}'
```

The system returns information like the following.

```json
{
  "WindowTaskId": "968e3b17-8591-4fb2-932a-b62389d6f635"
}
```

9. Execute the following command to get the high-level patch compliance summary for a patch group. The high-level patch compliance summary gives you the number of instances with patches in the following states for a patch group: "NotApplicable," "Missing," "Failed," "InstalledOther," and "Installed."

```bash
aws ssm describe-patch-group-state --patch-group "Production"
```

The system returns information like the following.
10. Execute the following command to get patch summary states per-instance for a patch group. The per-instance summary gives you a number of patches in the following states per instance for a patch group: "NotApplicable," "Missing," "Failed," "InstalledOther," and "Installed."

```
aws ssm describe-instance-patch-states-for-patch-group --patch-group "Production"
```

The system returns information like the following.

```
{
  "InstancePatchStates": [
    {
      "OperationStartTime": 1481259600.0,
      "FailedCount": 0,
      "InstanceId": "i-08ee91c0b17045407",
      "OwnerInformation": "",
      "NotApplicableCount": 2077,
      "OperationEndTime": 1481259757.0,
      "PatchGroup": "Production",
      "InstalledOtherCount": 186,
      "MissingCount": 7,
      "SnapshotId": "b0e65479-79be-4288-9f88-81c96bc3ed5e",
      "Operation": "Scan",
      "InstalledCount": 72
    },
    {
      "OperationStartTime": 1481259602.0,
      "FailedCount": 0,
      "InstanceId": "i-0fff3aab684d01b23",
      "OwnerInformation": "",
      "NotApplicableCount": 2692,
      "OperationEndTime": 1481259613.0,
      "PatchGroup": "Production",
      "InstalledOtherCount": 3,
      "MissingCount": 1,
      "SnapshotId": "b0e65479-79be-4288-9f88-81c96bc3ed5e",
      "Operation": "Scan",
      "InstalledCount": 1
    },
    {
      "OperationStartTime": 1481259547.0,
      "FailedCount": 0,
      "InstanceId": "i-0a00def7fafa94f1dc",
      "OwnerInformation": "",
      "NotApplicableCount": 1859,
      "OperationEndTime": 1481259592.0,
      "PatchGroup": "Production",
      "InstalledOtherCount": 116,
      "MissingCount": 1,
      "SnapshotId": "b0e65479-79be-4288-9f88-81c96bc3ed5e",
      "Operation": "Scan",
      "InstalledCount": 110
    },
    {
      "OperationStartTime": 1481259547.0,
      "FailedCount": 0,
      "InstanceId": "i-0a00def7fafa94f1dc",
      "OwnerInformation": "",
      "NotApplicableCount": 1859,
      "OperationEndTime": 1481259592.0,
      "PatchGroup": "Production",
      "InstalledOtherCount": 116,
      "MissingCount": 1,
      "SnapshotId": "b0e65479-79be-4288-9f88-81c96bc3ed5e",
      "Operation": "Scan",
      "InstalledCount": 110
    }
  ]
}
```
Additional Patch Manager CLI Commands

The section includes additional examples of CLI commands that you can use to perform Patch Manager configuration tasks.

Create a patch baseline

The following command creates a patch baseline that approves all critical and important security updates for Windows Server 2012 R2 five days after they are released.

```bash
aws ssm create-patch-baseline --name "Windows-Server-2012R2" --approval-rules "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Important,Critical]}, {Key=CLASSIFICATION,Values=SecurityUpdates}, {Key=PRODUCT,Values=WindowsServer2012R2}],ApproveAfterDays=5}]]" --description "Windows Server 2012 R2, Important and Critical security updates"
```

The system returns information like the following.

```json
{
  "BaselineId":"pb-00dbb759999aa2bc3"
}
```

Update a patch baseline

The following command adds two patches as rejected and one patch as approved to an existing patch baseline.

```bash
aws ssm update-patch-baseline --baseline-id pb-00dbb759999aa2bc3 --rejected-patches "KB2032276" "MS10-048" --approved-patches "KB2124261"
```

The system returns information like the following.

```json
{
  "BaselineId":"pb-00dbb759999aa2bc3",
  "Name":"Windows-Server-2012R2",
  "RejectedPatches": [
    "KB2032276",
    "MS10-048"
  ],
  "GlobalFilters": [
    "PatchFilters": [
    
  ]
}
rename a patch baseline

aws ssm update-patch-baseline --baseline-id pb-00dbb759999aa2bc3 --name "windows-server-2012-r2-important-and-critical-security-updates"

the system returns information like the following.

{ "baselineid":"pb-00dbb759999aa2bc3", "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" } ] }, "approvafterdays": 5 ] }, "modifieddate": 1481001494.035, "createddate": 1480997823.81, "approvedpatches": [ "kb2124261" ], "description": "windows server 2012 r2, important and critical security updates" }
Delete a patch baseline

```bash
aws ssm delete-patch-baseline --baseline-id "pb-0a34d8c0f03c1e529"
```

The system returns information like the following.

```
{
   "BaselineId":"pb-0a34d8c0f03c1e529"
}
```

List all patch baselines

```bash
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-west-2:755505623295:patchbaseline/pb-04f1fedd7c0c5339"
   },
   {
      "BaselineName":"Windows-Server-2012R2",
      "DefaultBaseline":false,
```
Here is another command that lists all patch baselines in a Region.

```
aws ssm describe-patch-baselines --region us-west-1 --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-west-2:755505623295:patchbaseline/pb-04f1feddd7c0c5339"
        },
        {
            "BaselineName": "Windows-Server-2012R2",
            "DefaultBaseline": false,
            "BaselineDescription": "Windows Server 2012 R2, Important and Critical security updates",
            "BaselineId": "pb-00dbb759999aa2bc3"
        }
    ]
}
```

### List all AWS provided patch baselines

```
aws ssm describe-patch-baselines --region us-west-1 --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-west-2:755505623295:patchbaseline/pb-04f1feddd7c0c5339"
        }
    ]
}
```

### List my patch baselines

```
aws ssm describe-patch-baselines --region us-west-1 --filters "Key=OWNER,Values=[Self]"
```

The system returns information like the following.

```
{
    "BaselineIdentities": [
```

211
Display a patch baseline

aws ssm get-patch-baseline --baseline-id pb-00dbb759999aa2bc3

The system returns information like the following.

```json
{
  "BaselineId":"pb-00dbb759999aa2bc3",
  "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":[]
}
```
Get the default patch baseline

```
aws ssm get-default-patch-baseline --region us-west-1
```

The system returns information like the following.

```
{
  "BaselineId": "arn:aws:ssm:us-west-1:075727635805:patchbaseline/pb-0ca44a362f8af725"
}
```

Set the default patch baseline

```
aws ssm register-default-patch-baseline --region us-west-1 --baseline-id "pb-08b654cf9b9681f04"
```

```
{
  "BaselineId": "pb-08b654cf9b9681f04"
}
```

Register a patch group "Web Servers" with a patch baseline

```
aws ssm register-patch-baseline-for-patch-group --baseline-id "pb-00dbb75999aa2bc3" --patch-group "Web Servers"
```

The system returns information like the following.

```
{
  "PatchGroup": "Web Servers",
  "BaselineId": "pb-00dbb759999aa2bc3"
}
```

Register a patch group "Backend" with the AWS-provided patch baseline

```
aws ssm register-patch-baseline-for-patch-group --region us-west-1 --baseline-id "arn:aws:ssm:us-west-1:075727635805:patchbaseline/pb-0ca44a362f8af725" --patch-group "Backend"
```

The system returns information like the following.

```
{
  "PatchGroup": "Backend",
  "BaselineId": "arn:aws:ssm:us-west-1:075727635805:patchbaseline/pb-0ca44a362f8af725"
}
```

Display patch group registrations

```
aws ssm describe-patch-groups --region us-west-1
```

The system returns information like the following.
Deregister a patch group from a patch baseline

```bash
aws ssm deregister-patch-baseline-for-patch-group --region us-west-1 --patch-group "Production" --baseline-id "arn:aws:ssm:us-west-1:075727635805:patchbaseline/pb-0ca44a362f8afc725"
```

The system returns information like the following.

```json
{
    "PatchGroup":"Production",
    "BaselineId":"arn:aws:ssm:us-west-1:075727635805:patchbaseline/pb-0ca44a362f8afc725"
}
```

Get all patches defined by a patch baseline

```bash
aws ssm describe-effective-patches-for-patch-baseline --region us-west-1 --baseline-id "pb-08b654cf9b9681f04"
```

The system returns information like the following.

```json
{
    "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.

```
Get all patches for Windows Server 2012 that have a MSRC severity of Critical

aws ssm describe-available-patches --region us-west-1 --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,
    "MsrsClassification": "Critical",
    "Language": "All",
    "KbNumber": "KB2727528",
    "MsrsNumber": "MS14-018",
    "Id": "845bac0-bf53-4fbd-915d-499de08c338b"
    }

```
"KbNumber": "KB2727528",
"MsrvNumber": "MS12-072",
"Id": "1eb507-be-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,
"MsrvClassification": "Critical",
"Language": "All",
"KbNumber": "KB2729462",
"MsrvNumber": "MS12-074",
"Id": "af873760-c97c-4088-ab7e-5219e120eab4"
}

---output truncated---

Get all available patches

aws ssm describe-available-patches --region us-west-1

The system returns information like the following.

{
  "NextToken": "--token string truncated--",
  "Patches": [
    {
      "ContentUrl": "https://support.microsoft.com/en-us/kb/2032276",
      "ProductFamily": "Windows",
      "Product": "WindowsServer2008R2",
      "Vendor": "Microsoft",
      "Description": "A security issue has been identified that could allow an unauthenticated remote attacker to compromise your system and gain control over it. You can help protect your system by installing this update from Microsoft. After you install this update, you may have to restart your system.",
      "Classification": "SecurityUpdates",
      "Title": "Security Update for Windows Server 2008 R2 x64 Edition (KB2032276)",
      "ReleaseDate": 1279040400.0,
      "MsrvClassification": "Important",
      "Language": "All",
      "KbNumber": "KB2032276",
      "MsrvNumber": "MS10-043",
      "Id": "8692029b-a3a2-4a87-a73b-8ea881b4b4d6"
    },
    {
      "ContentUrl": "https://support.microsoft.com/en-us/kb/2124261",
      "ProductFamily": "Windows",
      "Product": "Windows7",
      "Vendor": "Microsoft",
      "Description": "A security issue has been identified that could allow an unauthenticated remote attacker to compromise your system and gain control over it. You can help protect your system by installing this update from Microsoft. After you install this update, you may have to restart your system.",
      "Classification": "SecurityUpdates",
      "Title": "Security Update for Windows 7 Service Pack 1 (KB2124261)",
      "ReleaseDate": 1338520600.0,
      "MsrvClassification": "Critical",
      "Language": "All",
      "KbNumber": "KB2124261",
      "MsrvNumber": "MS12-073",
      "Id": "634a1be-9e5b-4872-8f86-8373e5f7d208"
    }
  ]
}
Tag a patch baseline

```bash
aws ssm add-tags-to-resource --resource-type "PatchBaseline" --resource-id "pb-0869b5cf84fa07081" --tags "Key=Project,Value=Testing"
```

List the tags for a patch baseline

```bash
aws ssm list-tags-for-resource --resource-type "PatchBaseline" --resource-id "pb-0869b5cf84fa07081"
```

Remove a tag from a patch baseline

```bash
aws ssm remove-tags-from-resource --resource-type "PatchBaseline" --resource-id "pb-0869b5cf84fa07081" --tag-keys "Project"
```

Get patch summary states per-instance

The per-instance summary gives you a number of patches in the following states per instance: "NotApplicable", "Missing", "Failed", "InstalledOther" and "Installed".

```bash
aws ssm describe-instance-patch-states --instance-ids i-08ee91c0b17045407 i-09a618aec652973a9 i-0a00def7faa94f1c i-0fff3aab684d01b23
```

The system returns information like the following.

```
{
    "InstancePatchStates": [
        {
            "OperationStartTime": "2016-12-09T05:00:00Z",
            "FailedCount": 0,
            "InstanceId": "i-08ee91c0b17045407",
            "OwnerInformation": "",
            "NotApplicableCount": 2077,
            "OperationEndTime": "2016-12-09T05:02:37Z",
            "PatchGroup": "Production",
            "MissingCount": 7,
            "SnapshotId": "b0e65479-79be-4288-9f88-81c96bc3ed5e",
            "Operation": "Scan",
            "InstalledCount": 72
        },
        {
            "OperationStartTime": "2016-12-09T04:59:09Z",
            "FailedCount": 0,
            "InstanceId": "i-09a618aec652973a9",
            "OwnerInformation": "",
            "NotApplicableCount": 1637,
            "OperationEndTime": "2016-12-09T05:03:57Z",
            "PatchGroup": "Production",
```
"InstalledOtherCount":388,
"MissingCount":2,
"SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
"Operation":"Scan",
"InstalledCount":141
}

Get patch compliance details for an instance

aws ssm describe-instance-patches --instance-id i-08ee91c0b17045407

The system returns information like the following.

{

"NextToken":"--token string truncated--",
"Patches":
{

  "KBId":"KB2919355",
  "Severity":"Critical",
  "Classification":"SecurityUpdates",
  "Title":"Windows 8.1 Update for x64-based Systems (KB2919355)",
  "State":"Installed",
  "InstalledTime":"2014-03-18T12:00:00Z"
},
{
  "KBId":"KB2977765",
  "Severity":"Important",
  "Classification":"SecurityUpdates",
  "Title":"Security Update for Microsoft .NET Framework 4.5.1 and 4.5.2 on Windows 8.1 and Windows Server 2012 R2 x64-based Systems (KB2977765)",
  "State":"Installed",
  "InstalledTime":"2014-10-15T12:00:00Z"
},
{
  "KBId":"KB2978126",
  "Severity":"Important",
  "Classification":"SecurityUpdates",
  "Title":"Security Update for Microsoft .NET Framework 4.5.1 and 4.5.2 on Windows 8.1 (KB2978126)",
  "State":"Installed",
  "InstalledTime":"2014-11-18T12:00:00Z"
},
---output truncated---

AWS Systems Manager Maintenance Windows

AWS Systems Manager Maintenance Windows let you define a schedule for when to perform potentially disruptive actions on your instances such as patching an operating system (OS), updating drivers, or installing software. Each Maintenance Window has a schedule, a duration, a set of registered targets, and a set of registered tasks. With Maintenance Windows, you can perform tasks like the following:

- Installing applications, updating patches, installing or updating SSM Agent, or executing PowerShell commands and Linux shell scripts by using a Systems Manager Run Command task.
- Building Amazon Machine Images (AMIs), boot-strapping software, and configuring instances by using Systems Manager Automation.
- Executing AWS Lambda functions that trigger additional actions such as scanning your instances for patch updates.
• Running AWS Step Function state machines to perform tasks such as removing an instance from an Elastic Load Balancing environment, patching the instance, and then adding the instance back to the Elastic Load Balancing environment.

Contents
• Controlling Access to Maintenance Windows (p. 219)
• Working with Maintenance Windows (p. 225)
• Systems Manager Maintenance Window Walkthroughs (p. 231)

Controlling Access to Maintenance Windows

Use one of the following methods to control access to Maintenance Windows by configuring security roles and permissions.

Topics
• Controlling Access to Maintenance Windows (AWS Console) (p. 219)
• Controlling Access to Maintenance Windows (AWS CLI) (p. 221)
• Controlling Access to Maintenance Windows (Tools for Windows PowerShell) (p. 223)

Controlling Access to Maintenance Windows (AWS Console)

The following procedures describe how to create the required roles and permissions for Maintenance Windows by using the AWS console.

Create an IAM Role for Systems Manager

Use the following procedure to create a role so that Systems Manager can execute tasks in Maintenance Windows on your behalf.

To create an IAM role for Maintenance Windows

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. Under Choose the service that will use this role, choose EC2. Under Select your use case, choose EC2, and then choose Next: Permissions.
4. In the list of policies, select the box next to AmazonSSMMaintenanceWindowRole, and then choose Next: Review.
5. In Role name, enter a name that identifies this role as a Maintenance Windows role.
6. Choose Create role. The system returns you to the Roles page.
7. Choose the name of the role you just created.
8. Choose the Trust relationships tab, and then choose Edit trust relationship.
9. Delete the current policy, and then copy and paste the following policy into the Policy Document field:

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"",
            "Effect":"Allow",
```
"Principal":{
  "Service":[
    "ec2.amazonaws.com",
    "ssm.amazonaws.com",
    "sns.amazonaws.com"
  ],
  "Action":"sts:AssumeRole"
}

Note
"sns.amazonaws.com" is required only if you will use Amazon SNS to send notifications related to Maintenance Window tasks executed through Run Command. See step 11 below for more information.

10. Choose Update Trust Policy, and then copy or make a note of the role name and the Role ARN value on the Summary page. You will specify this information when you create your Maintenance Window.

11. If you will configure a Maintenance Window to send notifications about command statuses using Amazon SNS, when executed through a Run Command command task, do the following:

1. Choose the Permissions tab.
2. Choose Add inline policy, and then choose the JSON tab.
3. In Policy Document, paste the following:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "sns-access-role-arn"
    }
  ]
}
```

`sns-access-role-arn` represents the ARN of the IAM role to be for sending SNS notifications related to the Maintenance Window, in the format of `arn:aws:iam::account-id:role/role-name`. For example: `arn:aws:iam::1112233444:role/SNS-Access-role`.

Note
In the Systems Manager console, this ARN is selected in the IAM Role list on the Register run command task page. For information, see Assign Tasks to a Maintenance Window (p. 227). In the Systems Manager API, this ARN is entered as the value of ServiceRoleArn in the SendCommand request.

5. In the Name box, type a name to identify this as a policy to allow sending Amazon SNS notifications.
6. Choose Create policy.

Assign the IAM PassRole Policy to an IAM User Account

When you register a task with a Maintenance Window, you specify the role you created in the previous procedure. This is the role that the service will assume when it runs tasks on your behalf. In order to register the task, you must assign the IAM PassRole policy to your IAM user account. The policy in the
The following procedure provides the minimum permissions required to register tasks with a Maintenance Window.

**To assign the IAM PassRole policy to an IAM user account**

1. In the IAM console navigation pane, choose **Users**, and then choose the user account you want to update.
2. In the policies list, verify that either the AmazonSSMFullAccess policy is listed or there is a comparable policy that gives the IAM user permission to call the Systems Manager API.
3. Choose **Add inline policy**.
4. On the **Set Permissions** page, choose **Policy Generator**, and then choose **Select**.
5. Verify that **Effect** is set to **Allow**.
6. From **AWS Services** choose **AWS Identity and Access Management**.
7. From **Actions**, choose **PassRole**.
8. In the **Amazon Resource Name (ARN)** field, paste the role ARN you created in the previous procedure.
9. Choose **Add Statement**, and then choose **Next Step**.
10. On the **Review Policy** page, choose **Apply Policy**.

**Controlling Access to Maintenance Windows (AWS CLI)**

Use the following procedure to create an IAM role for Maintenance Windows using the AWS CLI.

**To create an IAM role for Maintenance Windows**

1. Copy and paste the following trust policy into a text file. Save the file with the following name and file extension: mw-role-trust-policy.json.

```json
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect":"Allow",
            "Principal":{
                "Service": [
                    "ssm.amazonaws.com",
                    "ec2.amazonaws.com"
                ],
            "Action": "sts:AssumeRole"
            }
        }
    ]
}
```

2. Open the AWS CLI and execute the following command to create a Maintenance Window role called mw-task-role. The command assigns the policy you created in the previous step to this role.

   ```
   aws iam create-role --role-name mw-task-role --assume-role-policy-document file://mw-role-trust-policy.json
   ```

   The system returns information like the following.

   ```json
   {
      "Role":{
         "AssumeRolePolicyDocument":{
            "Version":"2012-10-17",
```
Controlling Access

"Statement": [
  {
    "Action": "sts:AssumeRole",
    "Effect": "Allow",
    "Principal": {
      "Service": [
        "ssm.amazonaws.com",
        "ec2.amazonaws.com"
      ]
    }
  }
],
"RoleId": "AROAIIZKPBKS2LEXAMPLE",
"CreateDate": "2017-04-04T03:40:17.373Z",
"RoleName": "mw-task-role",
"Path": "/",
"Arn": "arn:aws:iam::123456789012:role/mw-task-role"
}

Note

Make a note of the RoleName and the Arn. You will specify these when you create a Maintenance Window.

3. Execute the following command to attach the AmazonSSMResetMaintenanceWindowRole managed policy to the role you created in step 2.

```
aws iam attach-role-policy --role-name mw-task-role --policy-arn arn:aws:iam::aws:policy/service-role/AmazonSSMResetMaintenanceWindowRole
```

Assign the IAM PassRole Policy to an IAM User Account (AWS CLI)

When you register a task with a Maintenance Window, you specify the role you created in the previous procedure. This is the role that the service will assume when it runs tasks on your behalf. In order to register the task, you must assign the IAM PassRole policy to your IAM user account. The policy in the following procedure provides the minimum permissions required to register tasks with a Maintenance Window.

To assign the IAM PassRole policy to an IAM user account

1. Copy and paste the following IAM policy into a text editor and save it with the .json file extension.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1491345526000",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole",
        "ssm:RegisterTaskWithMaintenanceWindow"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```
2. Open the AWS CLI.
3. Execute the following command. For user-name, specify the IAM user who will assign tasks to Maintenance Windows. For policy-document, specify the path to the file you saved in step 1.

```
aws iam put-user-policy --user-name name of user --policy-name a name for the policy --policy-document path to document, for example: file://C:\Temp\passrole.json
```

**Note**
If you plan to register tasks for Maintenance Windows using the AWS Systems Manager console, you must also assign the AmazonSSMReadOnlyAccess policy to your user account. Execute the following command to assign this policy to your account.

```
aws iam attach-user-policy --policy-arn arn:aws:iam::aws:policy/AmazonSSMReadOnlyAccess --user-name IAM account name
```

4. Execute the following command to verify that the policy has been assigned to the user.

```
aws iam list-user-policies --user-name name of user
```

### Controlling Access to Maintenance Windows (Tools for Windows PowerShell)

Use the following procedure to create an IAM role for Maintenance Windows using the Tools for Windows PowerShell.

**To create an IAM role for Maintenance Windows**

1. Copy and paste the following trust policy into a text file. Save the file with the following name and file extension: mw-role-trust-policy.json.

```json
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "Service":[
          "ssm.amazonaws.com",
          "ec2.amazonaws.com"
        ]
      },
      "Action":"sts:AssumeRole"
    }
  ]
}
```

2. Open Tools for Windows PowerShell and execute the following command to create a role called mw-task-role. The role uses the policy that you created in the previous step.

```
New-IAMRole -RoleName "mw-task-role" -AssumeRolePolicyDocument (Get-Content -raw .\mw-role-trust-policy.json)
```

The systems returns information like the following.

```
Arn : arn:aws:iam::123456789012:role/mw-task-role
AssumeRolePolicyDocument : ExampleDoc12345678
```
3. Execute the following command to attach the AmazonSSMMaintenanceWindowRole managed policy to the role you created in the previous step.

   Register-IAMRolePolicy -RoleName mw-task-role -PolicyArn
   arn:aws:iam::aws:policy/service-role/AmazonSSMMaintenanceWindowRole

**Assign the IAM PassRole Policy to an IAM User Account (Tools for Windows PowerShell)**

When you register a task with a Maintenance Window, you specify the role you created in the previous procedure. This is the role that the service will assume when it runs tasks on your behalf. In order to register the task, you must assign the IAM PassRole policy to your IAM user account. The policy in the following procedure provides the minimum permissions required to register tasks with a Maintenance Window.

**To assign the IAM PassRole policy to an IAM user account**

1. Copy and paste the following IAM policy into a text editor and save it with the .json file extension.

   ```json
   {
     "Version":"2012-10-17",
     "Statement":[
       {
         "Sid":"Stmt1491345526000",
         "Effect":"Allow",
         "Action":[
           "iam:GetRole",
           "iam:PassRole",
           "ssm:RegisterTaskWithMaintenanceWindow"
         ],
         "Resource":[]
       }
     ]
   }
   ```


3. Execute the following command. For `user-name`, specify the IAM user who will assign tasks to Maintenance Windows. For `policy-document`, specify the path to the file you saved in step 1.

   ```powershell
   Write-IAMUserPolicy -UserName name of IAM user -PolicyDocument (Get-Content -raw path
to document, for example: C:\temp\passrole-policy.json) -PolicyName a name for the policy
   ```

**Note**

If you plan to register tasks for Maintenance Windows using the AWS Systems Manager console, you must also assign the AmazonSSMReadOnlyAccess policy to your user account. Execute the following command to assign this policy to your account.

   ```powershell
   Register-IAMUserPolicy -UserName IAM account name -PolicyArn
   arn:aws:iam::aws:policy/AmazonSSMReadOnlyAccess
   ```
4. Execute the following command to verify that the policy has been assigned to the user.

```
Get-IAMUserPolicies -UserName name of user
```

**Working with Maintenance Windows**

This section describes how to create, configure, and update or delete a Maintenance Window. This section also describes how to perform these same tasks for the targets and tasks of a Maintenance Window.

**Important**

We recommend that you initially create and configure Maintenance Windows in a test environment.

**Before You Begin**

Before you create a Maintenance Window, you must configure access to Maintenance Windows. For more information, see Controlling Access to Maintenance Windows (p. 219).

**Topics**

- Create a Maintenance Window (p. 225)
- Assign Targets to a Maintenance Window (p. 226)
- Assign Tasks to a Maintenance Window (p. 227)
- Update or Delete a Maintenance Window (p. 228)

**Create a Maintenance Window**

To create a Maintenance Window, you must complete the following tasks:

- Create the window and define its schedule and duration.
- Assign targets for the window.
- Assign tasks to execute during the window.

After you complete these tasks, the Maintenance Window runs according to the schedule you defined and executes the tasks on the targets you specified. After a task is finished, Systems Manager logs the details of the execution.

**Note**

You can execute the following types of tasks on targets:

- Commands by using Systems Manager Run Command
- Automation workflows by using Systems Manager Automation
- Functions by using AWS Lambda
- State machines by using AWS Step Functions

Lambda functions and Step Functions are currently not supported in the AWS Systems Manager console. To register these types of tasks, you must use the AWS CLI. For examples of how to create, configure, and update a Maintenance Window by using the AWS CLI, see the Systems Manager Maintenance Window Walkthroughs (p. 231).

**To create a Maintenance Window**

Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

Note
If you are using the Amazon EC2 console, some field names and locations may differ slightly.

2. In the navigation pane, choose **Maintenance Windows**.
3. Choose **Create a Maintenance Window**.
4. In the **Name** field, type a descriptive name to help you identify this Maintenance Window as a test Maintenance Window.
5. In the **Description** field, enter a description.
6. Choose **Allow unregistered targets** if you want to allow a Maintenance Window task to execute on managed instances, even if you have not registered those instances as targets. If you choose this option, then you can choose the unregistered instances (by instance ID) when you register a task with the Maintenance Window.
   
   If you don't choose this option, then you must choose previously-registered targets when you register a task with the Maintenance Window.
7. Specify a schedule for the Maintenance Window by using one of the scheduling options.
8. In the **Duration** field, type the number of hours the Maintenance Window should run.
9. In the **Stop initiating tasks** field, type the number of hours before the end of the Maintenance Window that the system should stop scheduling new tasks to run.
10. 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

After you create a Maintenance Window, you assign targets where the tasks will run.

**To assign targets to a Maintenance Window**

1. In the Maintenance Window list, choose the Maintenance Window you just created.
2. Choose **Actions**, and then choose **Register targets**.
3. In the **Target Name** field, type a name for the targets.
4. In the **Description** field, type a description.
5. In the **Owner information** field, specify your name or work alias. Owner information is included in any CloudWatch Events raised while running tasks for these targets in this Maintenance Window.
6. In the **Select targets by** section, choose **Specifying Tags** to target instances by using Amazon EC2 tags that you previously assigned to the instances. Choose **Manually Selecting Instances** to choose individual instances according to their instance ID.
   
   **Note**
   If you don't see the instances that you'd like to target, verify that those instances are configured for Systems Manager. For more information, see **Setting Up AWS Systems Manager** (p. 4).
7. Choose **Register targets**.

If you want to assign more targets to this window, choose the **Targets** tab, and then choose **Register new targets**. With this option, you can choose a different means of targeting. For example, if you
previously targeted instances by instance ID, you can register new targets and target instances by specifying Amazon EC2 tags.

**Assign Tasks to a Maintenance Window**

After you assign targets, you assign tasks to perform during the window.

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

**To assign tasks to a Maintenance Window (AWS Systems Manager)**

1. In the Maintenance Window list, choose the Maintenance Window you just created.
2. Choose **Actions** and then, choose either **Register run command task** to execute your choice of commands on targets by using an SSM document, or **Register automation task** to execute your choice of an Automation workflow on targets by using an SSM Automation document. For examples of how to create Lambda and Step Functions tasks by using the AWS CLI, see the [Systems Manager Maintenance Window Walkthroughs](p. 231).
3. In the **Name** field, type a name for the task.
4. In the **Description** field, type a description.
5. From the **Document** list, choose the SSM Command or Automation document that defines the task(s) to run.
6. In the **Document version** list (for Automation tasks), choose the document version to use.
7. In the **Task priority** list, specify a priority for this task. 1 is the highest priority. Tasks in a Maintenance Window are scheduled in priority order with tasks that have the same priority scheduled in parallel.
8. In the **Targets** section, identify the instances to which you want to download and run the script by specifying tags or selecting instances manually.
9. (Optional) In **Rate control**:
   - In **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.
   - **Note**
     If you selected targets by choosing Amazon EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document at the same time by specifying a percentage.
   - In **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify 3 errors, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.
10. In the **IAM Role** field, specify the Maintenance Windows ARN. For more information about creating a Maintenance Windows ARN, see [Controlling Access to Maintenance Windows](p. 219).
11. In the **Input Parameters** section, specify parameters for the document. For Automation documents, the system auto-populates some of the values. You can keep or replace these values.
12. Complete the wizard.

**To assign tasks to a Maintenance Window (Amazon EC2 Systems Manager)**

1. In the Maintenance Window list, choose the Maintenance Window you just created.
2. Choose **Actions** and then, choose either **Register run command task** to execute your choice of commands on targets by using an SSM document, or **Register automation task** to execute your choice of an Automation workflow on targets by using an SSM Automation document. For examples
of how to create Lambda and Step Functions tasks by using the AWS CLI, see the Systems Manager Maintenance Window Walkthroughs (p. 231).

3. In the Task Name field, type a name for the task.
4. In the Description field, type a description.
5. From the Document list, choose the SSM Command or Automation document that defines the task(s) to run.
7. In the Task Priority field, specify a priority for this task. 1 is the highest priority. Tasks in a Maintenance Window are scheduled in priority order with tasks that have the same priority scheduled in parallel.
8. In the Target by section, choose either Selecting registered target groups or Selecting unregistered targets, and then choose the targets.
9. In the Parameters section, specify parameters for the document. For Automation documents, the system auto-populates some of the values. You can keep or replace these values.

   In the Role field, specify the Maintenance Windows ARN. For more information about creating a Maintenance Windows ARN, see Controlling Access to Maintenance Windows (p. 219).

   The Execute on field lets you specify either a number of targets where the Maintenance Window tasks can run concurrently or a percentage of the total number of targets. This field is relevant when you target a large number of instances using tags.

   The Stop after field lets you specify the number of allowed errors before the system stops sending the task to new instances.

10. Complete the wizard.

Update or Delete a Maintenance Window

You can update or delete a Maintenance Window. You can also update or delete the targets or tasks of a Maintenance Window. If you edit the details of a Maintenance Window, you can change the schedule, targets, and tasks. You can also specify names and descriptions for windows, targets, and tasks, which helps you better understand their purpose, and makes it easier to manage your queue of windows.

This section describes how to update or delete a Maintenance Window, targets, and tasks by using the AWS Systems Manager console. For examples of how to do this by using the AWS CLI, see Updating a Maintenance Window (p. 238).

Updating or Deleting a Maintenance Window

You can update a Maintenance Window to change the name, description, and schedule of the window, and whether the window should allow unregistered targets.

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

To update or delete a Maintenance Window (AWS Systems Manager)

2. In the navigation pane, choose Maintenance Windows.
3. Choose the Maintenance Window that you want to update or delete, and then do one of the following:
   - Choose Delete. The system prompts you to confirm your actions.
• Choose **Edit**. On the **Edit maintenance window** page, change the values and options that you want, and then choose **Edit maintenance window**.

**To update or delete a Maintenance Window (Amazon EC2 Systems Manager)**

1. Open the [Amazon EC2 console](https://console.aws.amazon.com/ec2/), expand **Systems Manager Shared Resources** in the navigation pane, and then choose **Maintenance Windows**.
2. Choose the Maintenance Window that you want to update or delete.
3. Choose **Actions**, and then choose either **Delete Maintenance Window** or **Edit maintenance window**. If you chose to delete a Maintenance Window the system prompts you to confirm your actions. If you chose to edit a Maintenance Window, the **Edit maintenance window** page appears.
4. Change the values and options that you want, and then choose **Edit maintenance window**. The system returns you to the Maintenance Window page.

**Updating or Deleting the Targets of a Maintenance Window**

You can update or delete the targets of a Maintenance Window. If you choose to update a Maintenance Window target you can specify a new target name, description, and owner. You can also choose different targets.

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

**To update or delete the targets of a Maintenance Window (AWS Systems Manager)**

1. Open the [AWS Systems Manager console](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, and then do one of the following:
   - To update targets, choose **Edit**.
   - To delete targets, choose **Deregister targets**, and then choose the **Targets** tab.
   
   Choose the target to delete, and then choose **Deregister target**. In the **Deregister maintenance windows target** window, leave the **Safely deregister target** option selected if you want the system to check if the target is referenced by any tasks before deleting it. If the target is referenced by a task, the system returns an error and doesn't delete the target. Clear the **Safely deregister target** option if you want the system to delete the target even if it is referenced by a task.
   
   Choose **Deregister**.

**To update or delete the targets of a Maintenance Window (Amazon EC2 Systems Manager)**

1. Open the [Amazon EC2 console](https://console.aws.amazon.com/ec2/), expand **Systems Manager Shared Resources** in the navigation pane, and then choose **Maintenance Windows**.
2. Choose the Maintenance Window that you want to update.
3. Choose the **Targets** tab.
4. If you want to delete a target, choose the small X beside **Edit**. In the **Deregister target** window, leave the **Safely Deregister Target** option selected if you want the system to check if the target is referenced by any tasks before deleting it. If the target is referenced by a task, the system returns an error and doesn't delete the target. Clear the **Safely Deregister Target** option if you want the system to delete the target even if it is referenced by a task.
If you want to edit the targets, choose **Edit**.

5. Change the values and options that you want, and then choose **Edit Target**. The system returns you to the Maintenance Window page.

**Updating or Deleting the Tasks of a Maintenance Window**

You can update or delete the tasks of a Maintenance Window. If you choose to update, you can specify a new task name, description, and owner. For Run Command and Automation tasks, you can choose a different SSM document for the tasks. You can't, however, edit a task to change its type. For example, if you created an Automation task, you can't edit that task and change it to a Run Command task.

**Note**

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

**To update or delete the tasks of a Maintenance Window**

1. Open the Amazon EC2 console, expand **Systems Manager Shared Resources** in the navigation pane, and then choose **Maintenance Windows**.
2. Choose the Maintenance Window that you want to update.
3. Choose the **Tasks** tab.
4. If you want to delete a task, choose the small X beside **Edit**. If you want to edit the task, choose **Edit**.
5. Change the values and options that you want, and then choose **Edit Task**. The system returns you to the Maintenance Window page.

**Systems Manager Maintenance Window Walkthroughs**

Use the following walkthroughs to create, configure, and update Maintenance Window by using the AWS CLI. Before you attempt these walkthroughs, you must configure Maintenance Window roles and permissions. For more information, see *Controlling Access to Maintenance Windows* (p. 219).

**Topics**
- Creating and Configuring a Maintenance Window (p. 231)
- Updating a Maintenance Window (p. 238)
- Listing Information About Maintenance Windows (p. 243)

**Creating and Configuring a Maintenance Window**

The following walkthrough describes how to create and configure a Maintenance Window, targets, and tasks by using the AWS CLI.

**To create and configure a Maintenance Window Using the AWS CLI**

1. Download the AWS CLI to your local machine.
2. Open the AWS CLI and execute the following command to create a Maintenance Window that runs at 4 PM on every Tuesday for 4 hours, with a 1 hour cutoff, and that allows unassociated targets. For more information about creating cron expressions for the `schedule` parameter, see *Working with Cron and Rate Expressions for Systems Manager* (p. 33).

   ```bash
   aws ssm create-maintenance-window --name "My-First-Maintenance-Window" --schedule "cron(0 16 ? * TUE *)" --duration 4 --cutoff 1 --allow-unassociated-targets
   ```

   The system returns information like the following.

   ```json
   {
       "WindowId":"mw-ab12cd34ef56gh78"
   }
   ```

3. Execute the following command to list all Maintenance Windows in your AWS account.

   ```bash
   aws ssm describe-maintenance-windows
   ```

   The system returns information like the following.

   ```json
   {
       "WindowIdentities":[
           {
               "Duration":4,
               "Cutoff":1,
               "WindowId":"mw-ab12cd34ef56gh78",
               "Disabled":false,
               "Name":"My-First-Maintenance-Window"
           }
       ]
   }
   ```
4. Execute the following command to register an instance as a target for this Maintenance Window. The system returns a Maintenance Window target ID. You will use this ID in a later step to register a task for this Maintenance Window.

```
aws ssm register-target-with-maintenance-window --window-id "mw-ab12cd34ef56gh78" --target "Key=InstanceIds,Values=ID" --owner-information "Single instance" --resource-type "INSTANCE"
```

The system returns information like the following.

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

You could register multiple instances using the following command.

```
aws ssm register-target-with-maintenance-window --window-id "mw-ab12cd34ef56gh78" --targets "Key=InstanceIds,Values=ID 1, ID 2" --owner-information "Two instances in a list" --resource-type "INSTANCE"
```

You could also register instances using EC2 tags.

```
aws ssm register-target-with-maintenance-window --window-id "mw-ab12cd34ef56gh78" --targets "Key=tag:Environment,Values=Prod" "Key=Role,Values=Web" --owner-information "Production Web Servers" --resource-type "INSTANCE"
```

5. Use the following command to display the targets for a Maintenance Window.

```
aws ssm describe-maintenance-window-targets --window-id "mw-ob12cd34ef56gh78"
```

The system returns information like the following.

```
{
    "Targets": [
        {
            "ResourceId": "InstanceIds",
            "OwnerInformation": "Single instance",
            "WindowId": "mw-ab12cd34ef56gh78",
            "Targets": [
                {
                    "Values": ["i-12a23b4c5d6e7f8g9h"],
                    "Key": "InstanceIds"
                }
            ],
            "WindowTargetId": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4"
        },
        {
            "ResourceId": "InstanceIds",
            "OwnerInformation": "Two instances in a list",
            "WindowId": "mw-ab12cd34ef56gh78",
            "Targets": [
                {
                    "Values": ["i-12a23b4c5d6e7f8g9h"]
                }
            ],
            "WindowTargetId": "b1c2d3e4-b1c2-b1c2-b1c2-b1c2d3e4"
        }
    ]
}
```
Execute the following command to register a task for the Maintenance Window. The task in first example uses Systems Manager Run Command to execute the `df` command using the AWS-RunShellScript document. You can also specify tasks that use Systems Manager Automation, AWS Lambda, and AWS Step Functions, as shown in the additional examples. You can specify the following parameters when registering a task:

- **targets**: Specify either Key=WindowTargetIds,Values=IDs to specify a target that is already registered with the Maintenance Window. Or, specify Key=InstanceIds,Values=IDs to target individual instances that may or may not be registered with the Maintenance Window.
- **task-arn**: The resource that the task uses during execution. For RUN_COMMAND and AUTOMATION task types, TaskArn is the SSM document name or ARN. For LAMBDA tasks, it's the function name or ARN. For STEP_FUNCTION tasks, it's the state machine ARN.
- **window-id**: The ID of the target Maintenance Window.
- **task-type**: The type of task. The type can be one of the following: RUN_COMMAND, AUTOMATION, LAMBDA, or STEP_FUNCTION.
- **task-invocation-parameters**: Required and optional parameters. Some of the common task-invocation-parameters parameters are described in the next list.
  - **max-concurrency**: (Optional) The maximum number of instances that are allowed to execute the command at the same time. You can specify a number such as 10 or a percentage such as 10%.
  - **max-errors**: (Optional) The maximum number of errors allowed without the command failing. When the command fails one more time beyond the value of MaxErrors, the systems stops sending the command to additional targets. You can specify a number such as 10 or a percentage such as 10%.
  - **priority**: The priority of the task in the Maintenance Window. The lower the number the higher the priority (for example, 1 is highest priority). Tasks in a Maintenance Window are scheduled in priority order. Tasks that have the same priority are scheduled in parallel.

**Common parameters for task-invocation-parameters**
The following list describes some of the common parameters that you can specify when using `task-invocation-parameters`. You specify these parameters by using the `{ { PARAMETER_NAME } }` syntax, as shown in the examples in this section.

- **TARGET_ID**: The ID of the target. If the target type is INSTANCE (currently the only supported type), then the target ID is the instance ID.
- **TARGET_TYPE**: The type of target. Currently only INSTANCE is supported.
- **WINDOW_ID**: The ID of the target Maintenance Window.
- **WINDOW_TASK_ID**: The ID of the window task that is executing.
- **WINDOW_TARGET_ID**: The ID of the window target that includes the target (target ID).
- **LOGGING_S3_BUCKET_NAME**: The Amazon S3 bucket name, if configured by using the `logging-info` parameter.
- **LOGGING_S3_KEY_PREFIX**: The Amazon S3 key prefix, if configured by using the `logging-info` parameter.
- **LOGGING_S3_REGION**: The Amazon S3 Region, if configured by using the `logging-info` parameter.
- **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.

```bash
aws ssm register-task-with-maintenance-window --window-id mw-ab12cd34ef56gh78 --task-arn "AWS-RunShellScript" --targets "Key=InstanceIds,Values=Instance ID" --service-role-arn "arn:aws:iam::1122334455:role/MW-Role" --task-type "RUN_COMMAND" --task-invocation-parameters "{"commands":{"Values":["df"]}}" --max-concurrency 1 --max-errors 1 --priority 10
```

The system returns information like the following.

```json
{
  "WindowTaskId":"44444444-5555-6666-7777-88888888"
}
```

You can also register a task using a Maintenance Window target ID. The Maintenance Window target ID was returned from an earlier command.

```bash
aws ssm register-task-with-maintenance-window --targets "Key=WindowTargetIds,Values=Window Target ID" --task-arn "AWS-RunShellScript" --service-role-arn "arn:aws:iam::1122334455:role/MW-Role" --window-id "mw-ab12cd34ef56gh78" --task-type "RUN_COMMAND" --task-invocation-parameters "{"commands":{"Values":["df"]}}" --max-concurrency 1 --max-errors 1 --priority 10
```

The system returns information like the following.

```json
{
  "WindowTaskId":"44444444-5555-6666-7777-88888888"
}
```

Here are additional examples that include other task types.

**Important**

The IAM policy for Maintenance Windows requires that you prefix Lambda function (or alias) names and Step Functions state machine names with SSM, as shown in the first two
examples below. Before you proceed to register these types of tasks, you must update their names in AWS Lambda and AWS Step Functions to include SSM.

### Lambda

```bash
aws ssm register-task-with-maintenance-window --window-id mw-0290d787d641f11f3 --targets Key=WindowTargetIds,Values=31547414-69c3-49f8-95b8-ed2dcf045faa --task-arn arn:aws:lambda:us-east-1:711106535523:function:SSMTestFunction --service-role-arn arn:aws:iam::711106535523:role/MaintenanceWindows --task-type LAMBDA --task-invocation-parameters "\"\"\"\"" --priority 0 --max-concurrency 10 --max-errors 5 --name Name --description Description
```

### Step Functions

```bash
aws ssm register-task-with-maintenance-window --window-id mw-0290d787d641f11f3 --targets Key=WindowTargetIds,Values=31547414-69c3-49f8-95b8-ed2dcf045faa --task-arn arn:aws:states:us-east-1:711106535523:stateMachine:SSMTestStateMachine --service-role-arn arn:aws:iam::711106535523:role/MaintenanceWindows --task-type STEP_FUNCTIONS --task-invocation-parameters "\"\"\"\"" --priority 0 --max-concurrency 10 --max-errors 5 --name Name --description Description
```

### Automation

```bash
aws ssm register-task-with-maintenance-window --window-id mw-0290d787d641f11f3 --targets Key=WindowTargetIds,Values=31547414-69c3-49f8-95b8-ed2dcf045faa --task-arn AutomationDocumentName --service-role-arn arn:aws:iam::711106535523:role/MaintenanceWindows --task-type AUTOMATION --task-invocation-parameters "Automation\"DocumentVersion=5,Parameters=\{"instanceId':\"{{TARGET_ID}}\"\}" --priority 0 --max-concurrency 10 --max-errors 5 --name Name --description Description
```

### Run Command

```bash
aws ssm register-task-with-maintenance-window --window-id mw-0290d787d641f11f3 --targets Key=WindowTargetIds,Values=31547414-69c3-49f8-95b8-ed2dcf045faa --task-arn AWS-RunPowerShellScript --service-role-arn arn:aws:iam::711106535523:role/MaintenanceWindows --task-type RUN_COMMAND --task-invocation-parameters "RunCommand\"Comment=SomeComment,DocumentHashType=Sha256,DocumentHash=b9d0966408047ebcafee82de4d24467e8974d2f8ad271106535523:RunCommandTopic,NotificationEvents=[\"Success\",\"Failed\"],NotificationType=Invocation,RunCommand,TimeoutSeconds=30,Parameters=\{"commands':\"ipconfig\"\}" --priority 0 --max-concurrency 10 --max-errors 5 --name Name --description Description
```

7. Execute the following command to list all registered tasks for a Maintenance Window.

```bash
aws ssm describe-maintenance-window-tasks --window-id "mw-ob12cd34ef56gh78"
```

The system returns information like the following.

```json
{
  "Tasks": {
    "ServiceRoleArn": "arn:aws:iam::11111111:role/MW-Role",
    "MaxErrors": "1",
    "TaskArn": "AWS-RunPowerShellScript",
    "MaxConcurreny": "1",
    "WindowTaskId": "3333-3333-3333-333333",
```

```
8. Execute the following command to view a list of task executions for a specific Maintenance Window.

```bash
aws ssm describe-maintenance-window-executions --window-id "mw-ab12cd34ef56gh78"
```

The system returns information like the following.

```json
{
  "WindowExecutions": [
    {
      "Status":"SUCCESS",
      "WindowExecutionId":"1111-1111-1111-11111",
      "StartTime":1478230495.469
    },
    {
      "Status":"SUCCESS",
      "WindowExecutionId":"2222-2-2-22222222-22",
      "StartTime":1478231395.677
    },
    # ... omitting a number of entries in the interest of space...
  ]
}
```
9. Execute the following command to get information about a Maintenance Window task execution.

```bash
aws ssm get-maintenance-window-execution --window-execution-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
```

The system returns information like the following.

```json
{
  "Status":"SUCCESS",
  "TaskIds"[:
    "333-33-3333-333333",
  ],
  "StartTime":1478230495.472,
  "EndTime":1478230516.425,
  "WindowExecutionId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

10. Execute the following command to list the tasks executed as part of a Maintenance Window execution.

```bash
aws ssm describe-maintenance-window-execution-tasks --window-execution-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
```

The system returns information like the following.

```json
{
  "WindowExecutionTaskIdentities"[:
    {
      "Status":"SUCCESS",
      "EndTime":1478230516.425,
      "StartTime":1478230495.782,
      "TaskId":"33333-333-333333"
    }
  ]
}
```

11. Execute the following command to get the details of a task execution.

```bash
aws ssm get-maintenance-window-execution-task --window-execution-id "55555-555-55-55555" --task-id "4444-4444-4444-4444"
```

The system returns information like the following.

```json
{
  "Status":"SUCCESS",
  "MaxErrors":"1",
  "TaskArn":"AWS-RunPowerShellScript",
```
12. Execute the following command to get the specific task invocations performed for a task execution.

```bash
aws ssm describe-maintenance-window-execution-task-invocations --window-execution-id "555555-555-55-555555" --task-id "4444-4444-4444-444444"
```

The system returns information like the following.

```json
{
    "WindowExecutionTaskInvocationIdentities": [
        {
            "Status": "SUCCESS",
            "Parameters": "{ "documentName": "AWS-RunPowerShellScript", "instanceIds": [ "i-1a2b3c4d5e6f7g8h9", "i-0a0def7fa94f1dc" ], "parameters": { "commands": [ "ipconfig.exe" ] }, "maxConcurrency": "1", "maxErrors": "1" },
            "ExecutionId": "555555-555-55-555555",
            "InvocationId": "3333-3333-3333-33333",
            "StartTime": 1478230495.842,
            "EndTime": 1478230516.291
        }
    ]
}
```

13. If you want, execute the following command to delete the Maintenance Window you created.

```bash
aws ssm delete-maintenance-window --window-id "mw-1a2b3c4d5e6f7g8h9"
```

The system returns information like the following:

```json
{
    "WindowId": "mw-1a2b3c4d5e6f7g8h9"
}
```

### Updating a Maintenance Window

This section describes how to update a Maintenance Window by using the AWS CLI. This section also includes information updating different task types, including Systems Manager Run Command, Systems Manager Automation, AWS Lambda, and AWS Step Functions tasks. For more information about updating a Maintenance Window, see [Update or Delete a Maintenance Window](p. 228).

The examples in this section use the following Systems Manager actions for updating a Maintenance Window.

- UpdateMaintenanceWindow
- UpdateMaintenanceWindowTarget
To update a Maintenance Window

1. Execute the following command to update a target to include a name and a description.

```bash
aws ssm update-maintenance-window-target --window-id "mw-12345678910" --window-target-id "a1b2c3d4-e5f6-g7h8i9" --name "NewTargetName" --description "NewTargetName description"
```

The system returns information like the following.

```
{
    "WindowId": "mw-12345678910",
    "WindowTargetId": "a1b2c3d4-e5f6-g7h8i9",
    "Targets": [
        {
            "Key": "InstanceIds",
            "Values": [
                "i-aabbccddeeff"
            ]
        },
        {
            "Name": "NewTargetName",
            "Description": "NewTargetName description"
        }
    ]
}
```

2. Execute the following command to use the `replace` option to remove the description field and add an additional target. The description field is removed, because the update does not include the field (a null value).

```bash
aws ssm update-maintenance-window-target --window-id "mw-12345678910" --window-target-id "a1b2c3d4-e5f6-g7h8i9" --targets "Key=InstanceIds,Values=i-aabbccddeeff,i-223344556677" --name "NewTargetName" --replace
```

The system returns information like the following.

```
{
    "WindowId": "mw-12345678910",
    "WindowTargetId": "a1b2c3d4-e5f6-g7h8i9",
    "Targets": [
        {
            "Key": "InstanceIds",
            "Values": [
                "i-aabbccddeeff",
                "i-223344556677"
            ]
        }
    ],
    "Name": "NewTargetName"
}
```

3. Execute the following command to update a Run Command task.

```bash
aws ssm update-maintenance-window-task --window-id "mw-12345678910" --window-task-id "1111-2222-3333-4444-5555" --targets "Key=WindowTargetIds,Values=a1b2c3d4-e5f6-g7h8i9" --task-arn "AWS-RunPowerShellScript" --service-role-arn "arn:aws:iam::abcdefghijk:role/MaintenanceWindowsRole" --task-invocation-parameters
```

239
"RunCommand={Comment=A_Comment,Parameters={commands=ipconfig}}" --priority 1 --max-concurrency 10 --max-errors 4 --name "RC_Name" --description "RC_Name description extra"

The system returns information like the following.

```
{
    "WindowId": "mw-12345678916",
    "WindowTaskId": "aaa-bbb-ccc-ddd",
    "Targets": [
        {
            "Key": "WindowTargetIds",
            "Values": [
                "alb2c3d4-e5f6-g7h819c"
            ]
        }
    ],
    "TaskArn": "AWS-RunPowerShellScript",
    "ServiceRoleArn": "arn:aws:iam::abcdefhijk:role/MaintenanceWindowsRole",
    "TaskParameters": {},
    "TaskInvocationParameters": {
        "RunCommand": {
            "Comment": "SomeComment",
            "Parameters": {
                "commands": [
                    "ipconfig -tail"
                ]
            }
        }
    },
    "Priority": 1,
    "MaxConcurrency": "10",
    "MaxErrors": "4",
    "Name": "RC_Name",
    "Description": "RC_Name description extra"
}
```

4. Execute the following command to add a name and a description to a Lambda task.

```
aws ssm update-maintenance-window-task --window-id "mw-1234567891" --window-task-id "la2b3c4d-5e6f-7g8h90" --targets "Key=WindowTargetIds,Values=alb2c3d4-e5f6-g7h819c,444-55555-66666-7777d" --task-arn "arn:aws:lambda:us-east-1:131313131:function:SSMTestLambda" --service-role-arn "arn:aws:iam::abcdefhijk:role/MaintenanceWindowsRole" --task-invocation-parameters "{{"Lambda":{"Payload"::\"{{"targetId":"\"{{TARGET_ID}}\",\"targetType\"{{TARGET_TYPE}}\"})\"}}","Priority":0,"MaxConcurrency":10,"MaxErrors":5,"Name":"TestLambda_Name","Description":"TestLambda_Name description"}
```

The system returns information like the following.

```
{
    "WindowId": "mw-1234567891",
    "WindowTaskId": "la2b3c4d-5e6f-7g8h90",
    "Targets": [
        {
            "Key": "WindowTargetIds",
            "Values": [
                "alb2c3d4-e5f6-g7h819c",
                "444-55555-66666-7777d"
            ]
        }
    ],
}
```
5. Execute the following command to update an AWS Step Functions task to update task-invocation-parameters.

```bash
aws ssm update-maintenance-window-task --window-id "mw-1234567891" --window-task-id
"1a2b3c4d-5e6f-7g8h9i" --targets "Key=WindowTargetIds,Values=alb2c3d4-e5f6-g7h8i9c" 
--service-role-arn "arn:aws:iam::abcdefghijk:role/MaintenanceWindowsRole" --
task-invocation-parameters "{{StepFunctions}}:\"\"{{instanceId}}\"\"}}" --priority 0 --max-concurrency 10 --max-errors 5 --name
"TestStepFunction_Task" --description "TestStepFunction_Task description"
```

The system returns information like the following.

```json
{
   "WindowId": "mw-1234567891",
   "WindowTaskId": "1a2b3c4d-5e6f-7g8h9i",
   "Targets": [
      {
         "Key": "WindowTargetIds",
         "Values": [
            "alb2c3d4-e5f6-g7h8i9c"
      }
   ],
   "ServiceRoleArn": "arn:aws:iam::abcdefghijk:role/MaintenanceWindowsRole",
   "TaskParameters": {},
   "TaskInvocationParameters": {
      "StepFunctions": {
         "Input": "{\"instanceId\"":\"{{ TARGET_ID }}\""
      }
   },
   "Priority": 0,
   "MaxConcurrency": "10",
   "MaxErrors": "5",
   "Name": "TestStepFunction_Task",
   "Description": "TestStepFunction_Task description"
}
```

6. Execute 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.

```bash
aws ssm deregister-target-from-maintenance-window --window-id "mw-1234567891b" --
window-target-id "aaaa-bbbb-cccc-dddd" --safe
```

The system returns information like the following.
An error occurred (TargetInUseException) when calling the DeregisterTargetFromMaintenanceWindow operation: This Target cannot be deregistered because it is still referenced in Task: a11b22c33d44e55f66

7. Execute 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.

```
aws ssm deregister-target-from-maintenance-window --window-id "mw-1234567891b" --window-target-id "aaaa-bbbb-cccc-dddd" --no-safe
```

The system returns information like the following.

```
{
    "WindowId": "mw-1234567891b",
    "WindowTargetId": "aaaa-bbbb-cccc-ddd"
}
```

8. Execute 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}}'

```
aws ssm update-maintenance-window-task --window-id "mw-1234567891b" --window-task-id "777-8888-9999-0000" --targets "Key=InstanceIds,Values=i-yyyyzzzzxxx111222" --task-invocation-parameters "RunCommand={Comment=SomeComments,Parameters={UpdateLevel='{{ssm:UpdateLevel}}'}}"
```

The system returns information like the following.

```
{
    "WindowId": "mw-1234567891b",
    "WindowTaskId": "777-8888-9999-0000",
    "Targets": [
        {
            "Key": "InstanceIds",
            "Values": [
                "i-yyyyzzzzxxx1112223"
            ]
        }
    ],
    "TaskArn": "AWS-InstallMissingWindowsUpdates",
    "ServiceRoleArn": "arn:aws:iam::abcdefghijk:role/MaintenanceWindows",
    "TaskParameters": {},
    "TaskInvocationParameters": {
        "RunCommand": {
            "Comment": "SomeComments",
            "Parameters": {
                "UpdateLevel": [
                    "{{ssm:UpdateLevel}}"
                ]
            }
        }
    },
    "Priority": 0,
    "MaxConcurrency": "10",
    "MaxErrors": "5",
    "Name": "TracyMWTest_RunCommand2",
    "Description": "Test_RunCommandandParameterStore description"
}
```
9. Execute the following command to update an Automation task to specify WINDOW_ID and WINDOW_TASK_ID parameters for the task-invocation-parameters parameter.

```
```

The system returns information like the following.

```
{  
  "WindowId": "mw-0a097ccb2abd5775b",  
  "WindowTaskId": "777-8888-9999-0000",  
  "Targets": [  
    {  
      "Key": "WindowTargetIds",  
      "Values": [  
        "999-aaa-888-bbb-777"
    ]  
  ],  
  "TaskArn": "AutoTestDoc",  
  "ServiceRoleArn": "arn:aws:iam::abcdefghijk:role/MaintenanceWindows",  
  "TaskParameters": {},  
  "TaskInvocationParameters": {  
    "Automation": {  
      "Parameters": {  
        "multi": [  
          "{{WINDOW_TASK_ID}}"
        ],  
        "single": [  
          "{{WINDOW_ID}}"
        ]  
      }  
    }  
  },  
  "Priority": 0,  
  "MaxConcurrency": "10",  
  "MaxErrors": "5",  
  "Name": "TestAutomation_Task", 
  "Description": "TestAutomation_Task description"
}
```

Listing Information About Maintenance Windows

This section includes commands to help you update or get information about your Maintenance Windows, tasks, executions, and invocations.

List All Maintenance Windows in Your AWS Account

Execute the command as shown here.

```
aws ssm describe-maintenance-windows
```

The system returns information like the following.

```
{
```
"WindowIdentities": [  {   "Duration": 2,   "Cutoff": 0,   "WindowId": "mw-ab12cd34ef56gh78",   "Enabled": true,   "Name": "IAD-Every-15-Minutes"  },  {   "Duration": 4,   "Cutoff": 1,   "WindowId": "mw-1a2b3c4d5e6f7g8h9",   "Enabled": true,   "Name": "My-First-Maintenance-Window"  },  {   "Duration": 8,   "Cutoff": 2,   "WindowId": "mw-123abc456def789",   "Enabled": false,   "Name": "Every-Day"  } ]

List all enabled Maintenance Windows

Execute the command as shown here.

```
aws ssm describe-maintenance-windows --filters "Key=Enabled,Values=true"
```

The system returns information like the following.

```
{   "WindowIdentities": [  {   "Duration": 2,   "Cutoff": 0,   "WindowId": "mw-ab12cd34ef56gh78",   "Enabled": true,   "Name": "IAD-Every-15-Minutes"  },  {   "Duration": 4,   "Cutoff": 1,   "WindowId": "mw-1a2b3c4d5e6f7g8h9",   "Enabled": true,   "Name": "My-First-Maintenance-Window"  } ]
```

List all Disabled Maintenance Windows

Execute the command as shown here.

```
aws ssm describe-maintenance-windows --filters "Key=Enabled,Values=false"
```

The system returns information like the following.

```
{   "WindowIdentities": [  {   "Duration": 4,   "Cutoff": 1,   "WindowId": "mw-1a2b3c4d5e6f7g8h9",   "Enabled": true,   "Name": "My-First-Maintenance-Window"  } ]
```
"WindowIdentities": [
  {
    "Duration": 8,
    "Cutoff": 2,
    "WindowId": "mw-1a2b3c4d5e6f7g8h9",
    "Enabled": false,
    "Name": "Every-Day"
  }
]
}

**Filter by Name**

In this example, the command returns all Maintenance Windows with a name starting with 'My'.

```
aws ssm describe-maintenance-windows --filters "Key=Name,Values=My"
```

The system returns information like the following.

```
{
  "WindowIdentities": [
    {
      "Duration": 4,
      "Cutoff": 1,
      "WindowId": "mw-1a2b3c4d5e6f7g8h9",
      "Enabled": true,
      "Name": "My-First-Maintenance-Window"
    }
  ]
}
```

**Display the Targets for a Maintenance Window Matching a Specific Owner Information Value**

Execute the command as shown here.

```
aws ssm describe-maintenance-window-targets --window-id "mw-ab12cd34ef56gh78" --filters "Key=OwnerInformation,Values=Single instance"
```

The system returns information like the following.

```
{
  "Targets": [
    {
      "TargetType": "INSTANCE",
      "TagFilters": [],
      "TargetIds": [
        "i-1a2b3c4d5e6f7g8h9"
      ],
      "WindowTargetId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2",
      "OwnerInformation": "Single instance"
    }
  ]
}
```

**Show All Registered Tasks that Invoke the AWS-RunPowerShellScript Run Command**

Execute the command as shown here.
aws ssm describe-maintenance-window-tasks --window-id "mw-ab12cd34ef56gh78" --filters "Key=TaskArn,Values=AWS-RunPowerShellScript"

The system returns information like the following.

```
{
  "Tasks": [
    {
      "ServiceRoleArn": "arn:aws:iam::444444444444:role/MW-Role",
      "MaxErrors": "1",
      "TaskArn": "AWS-RunPowerShellScript",
      "MaxConcurrency": "1",
      "WindowTaskId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c",
      "TaskParameters": {
        "commands": {
          "Values": [
            "driverquery.exe"
          ]
        },
        "Priority": 3,
        "Type": "RUN_COMMAND",
        "Targets": [
          {
            "TaskTargetId": "i-1a2b3c4d5e6f7g8h9",
            "TaskTargetType": "INSTANCE"
          }
        ]
      },
      "Priority": 3,
      "Type": "RUN_COMMAND",
      "Targets": [
        {
          "TaskTargetId": "i-1a2b3c4d5e6f7g8h9",
          "TaskTargetType": "INSTANCE"
        }
      ]
    },
    {
      "ServiceRoleArn": "arn:aws:iam::333333333333:role/MW-Role",
      "MaxErrors": "1",
      "TaskArn": "AWS-RunPowerShellScript",
      "MaxConcurrency": "1",
      "WindowTaskId": "33333-33333-333-33333",
      "TaskParameters": {
        "commands": {
          "Values": [
            "ipconfig.exe"
          ]
        },
        "Priority": 1,
        "Type": "RUN_COMMAND",
        "Targets": [
          {
            "TaskTargetId": "44444-444-4444-44444",
            "TaskTargetType": "WINDOW_TARGET"
          }
        ]
      }
    }
  ]
}
```

Show All Registered Tasks that Have a Priority of 3

Execute the command as shown here.

```
aws ssm describe-maintenance-window-tasks --window-id "mw-ab12cd34ef56gh78" --filters "Key=Priority,Values=3"
```

The system returns information like the following.
Show All Registered Tasks that Have a Priority of 1 and Use Run Command

Execute the command as shown here.

```
aws ssm describe-maintenance-window-tasks --window-id "mw-ab12cd34ef56gh78" --filters "Key=Priority,Values=1" "Key=TaskType,Values=RUN_COMMAND"
```

The system returns information like the following.

```
{
  "Tasks": [
    {
      "ServiceRoleArn": "arn:aws:iam::333333333:role/MW-Role",
      "MaxErrors": "1",
      "TaskArn": "AWS-RunPowerShellScript",
      "MaxConcurrency": "1",
      "WindowTaskId": "66666-555-66-555-6666",
      "TaskParameters": {
        "commands": {
          "Values": [
            "ipconfig.exe"
          ]
        }
      },
      "Priority": 1,
      "Type": "RUN_COMMAND",
      "Targets": [
        {
          "TaskTargetId": "i-1a2b3c4d5e6f7g8h9",
          "TaskTargetType": "INSTANCE"
        }
      ]
    }
  ]
}
```
List All Tasks Executed Before a Date

Execute the command as shown here.

```
aws ssm describe-maintenance-window-executions --window-id "mw-ab12cd34ef56gh78" --filters "Key=ExecutedBefore,Values=2016-11-04T05:00:00Z"
```

The system returns information like the following.

```
{
  "WindowExecutions": [
    {
      "Status": "SUCCESS",
      "EndTime": 1478229594.666,
      "WindowExecutionId": "",
      "StartTime": 1478229594.666
    },
    {
      "Status": "SUCCESS",
      "WindowExecutionId": "06dc5f8a-9ef0-4ae9-a466-ada2d4ce2d22",
      "StartTime": 1478230495.469
    },
    {
      "Status": "SUCCESS",
      "WindowExecutionId": "57ad6419-023e-44b0-a831-6687334390b2",
      "StartTime": 1478231395.677
    },
    {
      "Status": "SUCCESS",
      "WindowExecutionId": "ed1372b7-866b-4d64-bc2a-bbfd5195f4ae",
      "StartTime": 1478232295.529
    },
    {
      "Status": "SUCCESS",
      "WindowExecutionId": "154eb2fa-6390-4cb7-8c9e-55686b88c7b3",
      "StartTime": 1478233195.687
    },
    {
      "Status": "SUCCESS",
      "WindowExecutionId": "1c4de752-eff6-4778-b477-1681c6c03cf1",
      "StartTime": 1478234095.553
    },
    {
      "Status": "SUCCESS",
      "WindowExecutionId": "56062f75-e4d8-483f-b5c2-906d613409a4",
      "StartTime": 1478234995.12
  ]
}
```

List All Tasks Executed After a Date

Execute the command as shown here.

```
aws ssm describe-maintenance-window-executions --window-id "mw-ab12cd34ef56gh78" --filters "Key=ExecutedAfter,Values=2016-11-04T17:00:00Z"
```

The system returns information like the following.

```
{
  "WindowExecutions": [
```
AWS Systems Manager State Manager

AWS Systems Manager State Manager is a secure and scalable configuration management service that ensures your Amazon EC2 and hybrid infrastructure is in a desired or consistent state, which you define. Some examples of these configurations include:

- Bootstrap instances with specific software at start-up
- Regularly download and update agents, including the SSM Agent.
- Configure network settings
- Join instances to a Windows domain
- Patch instances with software updates throughout their lifecycle
- Execute scripts on Linux and Windows managed instances throughout their lifecycle

State Manager integrates with AWS CloudTrail to provide a record of all executions that you can audit, and Amazon CloudWatch Events to track state changes. You can also view detailed output in Amazon S3.

Getting Started with State Manager

To get started with State Manager, complete the following tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update the SSM Agent on your managed instances to the latest version.</td>
<td>Installing and Configuring SSM Agent (p. 13)</td>
</tr>
<tr>
<td>Verify Systems Manager prerequisites.</td>
<td>Systems Manager Prerequisites (p. 4)</td>
</tr>
<tr>
<td>Either choose a predefined AWS Command or Policy type document and specify parameters at runtime. Or create a document that defines the actions to perform on your instances.</td>
<td>Creating Systems Manager Documents (p. 267)</td>
</tr>
<tr>
<td>Create and apply the association to your instances.</td>
<td>Creating an Association (p. 252)</td>
</tr>
</tbody>
</table>

Related Content
See the following blog posts for additional examples of how to use State Manager:

- Combating Configuration Drift Using Amazon EC2 Systems Manager and Windows PowerShell DSC
- Running Ansible Playbooks using Amazon EC2 Systems Manager Run Command and State Manager
- Configure Amazon EC2 Instances in an Auto Scaling Group Using State Manager

**About State Manager**

Systems Manager State Manager is a secure and scalable configuration management service that ensures your Amazon EC2 and hybrid infrastructure is in a desired or consistent state, which you define. State Manager works as follows.

1. You start by determining the state you want to apply to your managed instances. For example, determine the applications to bootstrap or the network settings to configure. You can specify the details of the state as parameters at runtime by using an AWS preconfigured document. Or, you can create your own document and specify the state directly in the document, or as parameters at runtime. These documents are written in JSON or YAML, and are called SSM documents. An SSM document can include multiple actions or steps (for example, multiple commands to execute). State Manager uses either Command or Policy type documents. For more information about SSM documents, see [AWS Systems Manager Documents](p. 258).

2. You specify a schedule for when or how often to apply the state. You can specify a cron or rate expression.

3. You specify the targets for the state. You can target managed instances (Amazon EC2 instances, or machines in your hybrid environment that are configured for Systems Manager.) You can target instances by specifying one or more instance IDs, or you target large groups of managed instances by targeting EC2 tags. Using the AWS CLI, AWS Tools for Windows PowerShell, or the SSM SDK, you can target by specifying multiple tags.

4. You then bind this information (schedule, targets, documents, parameters) to the managed instances. The binding of this information to the targets is called an association. You can create an association by using the Amazon EC2 console, the AWS CLI, AWS Tools for Windows PowerShell, or AWS SDKs.

5. After you send the request to create the association, the status of the association shows "Pending". The system attempts to reach all targets and immediately apply the state specified in the association.

   **Note**
   If you create a new association that is scheduled to run while an earlier association is still running, the earlier association is timed out and the new association executes.

6. Systems Manager reports the status of the request for each instance targeted by the request. You can view status details in the EC2 console, by using the `DescribeInstanceAssociationsStatus` API action, or in Amazon S3, if you specified an S3 bucket when you created the association.

7. After you create the association, State Manager reapplies the state according to the schedule defined in the association. You can update your association documents and reapply them as necessary. You can also create association versions.

**Sample State Manager Documents**

This section includes sample State Manager documents. The samples in this section are purposefully simple for the sake of demonstration. For more information about how to create your own documents, see [AWS Systems Manager Documents](p. 258).

*Document 1:* This document includes one step that invokes the `aws:runPowerShellScript` plugin to return the instance host name. This document can execute on Windows and Linux instances. For more information about Systems Manager plugins, see [Top-level Elements](p. 285).
Document 2: This document includes one step that invokes the aws:runShellScript plugin to return the instance host name. This document can execute on Linux instances.

```json
{
    "schemaVersion": "2.2",
    "description": "Sample document",
    "mainSteps": [
        {
            "action": "aws:runShellScript",
            "name": "runShellScript",
            "inputs": {
                "runCommand": ["hostname"]
            }
        }
    ]
}
```

Document 3: This document includes one step that invokes the aws:cloudWatch plugin to enable Amazon CloudWatch Logs. This document can execute on Windows instances.

```json
{
    "schemaVersion": "2.2",
    "description": "Sample document",
    "mainSteps": [
        {
            "action": "aws:cloudWatch",
            "name": "cloudWatch",
            "settings": {
                "startType": "Enabled"
            }
        }
    ]
}
```

Document 4: This document includes two steps that invoke the aws:runPowerShellScript and aws:runShellScript plugins to return the instance host name. This document can execute on Linux instances.

```json
{
    "schemaVersion": "2.2",
    "description": "Sample document",
    "mainSteps": [
        {
            "action": "aws:runPowerShellScript",
            "name": "runPowerShellScript",
            "inputs": {
                "runCommand": ["hostname"]
            }
        },
        {
            "action": "aws:runShellScript",
            "name": "runShellScript",
            "inputs": {
                "runCommand": ["hostname"]
            }
        }
    ]
}
```
Creating an Association

This section describes how to create a State Manager association by using the Amazon EC2 console. The example in this section shows you how to create an association based on a custom SSM document. If this is your first time creating an association, then we recommend that you perform this procedure in a test environment. For an example of how to create an association by using the AWS CLI, see Systems Manager State Manager Walkthroughs (p. 254).

Before You Begin

Before you complete the following procedure, verify that you have at least one instance running that is configured for Systems Manager. For more information, see Systems Manager Prerequisites (p. 4).

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

To create a State Manager association

1. Open the Amazon EC2 console, expand Systems Manager Shared Resources in the navigation pane, and then choose Documents.
2. Choose Create Document.
3. In the Name field, type a descriptive name that identifies this document as a test document for State Manager.
4. In the Document type list, choose Command.
5. Delete the pre-populated brackets {} in the Content field, and then copy and paste the following sample document into the Content field.

This document includes one step that invokes the aws:runPowerShellScript plugin to return the instance host name. This document can execute on Windows instances.

```
{
    "schemaVersion":"2.0",
    "description":"Sample document",
    "mainSteps":[
    {
        "action": "aws:runPowerShellScript",
        "name": "runPowerShellScript",
        "inputs":{
            "runCommand": [
                "hostname"
            ]
        }
    }
]}
```
6. Choose **Create document**. After the system creates the document, choose **Close**.

7. In the EC2 console navigation pane, expand **Systems Manager Services**, and then choose **State Manager**.

8. Choose **Create Association**.

9. In the **Association Name**, specify a descriptive name for this association. For example, specify TestHostnameAssociation.

10. In the **Select Document** section, choose the document you just created.

11. In the **Document Version** list, leave the default value.

12. In the **Select Targets by** section, choose an option.

13. In the **Schedule** section, choose an option.

14. Disregard the **Parameters** section, as the test document does not take parameters. Also, disregard the **Write to S3** option, because using that option is described in the next procedure.

15. Choose **Create Association**. The system attempts to create the association on the instance and immediately apply the state. In this case, after creating the association, the system attempts to return the host name. The association status shows **Pending**.

16. In the right corner of the Association page, choose the refresh button. The status changes to **Success**.

You can't view the output from this procedure (the host name), because you did not write the output to Amazon S3. To edit the association, write the output to S3, and view the host name, complete the next procedure.

### Editing an Association (Versioning)

You can edit an association to specify a new name, schedule, or targets. You can also choose the option to **Write to S3** if you didn't do this previously. After you edit an association, Systems Manager creates a new version. You can view different versions after editing, as described in the following procedure.

**Note**

This procedure require an S3 bucket. If you have not used S3 before, be aware that you will incur charges for using S3. For information about how to create a bucket, see [Create a Bucket](#).

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

#### To edit a State Manager association

1. Open the Amazon EC2 console, expand **Systems Manager Services** in the navigation pane, and then choose **State Manager**.

2. Choose the association you created in the previous procedure.

3. From the **Actions** menu, choose **Edit Association**.

4. In the **Name** field, type a new name. For example, type TestHostnameAssociation2.

5. In the **Schedule** section, choose a new option. For example, choose **Cron schedule builder**, and then choose **Every 1 hour**.

6. In the **Advanced** section, choose **Write to S3**.

7. Disregard the **S3Region** field. This field is deprecated. Specify the name of your bucket in the **S3Bucket Name** field. If want to write output to a sub-folder, specify the sub folder name in the **S3Key Prefix** field.
8. Choose **Edit Association**, and then choose **Close**.
9. In the State Manager page, choose the association you just edited, and then choose the **Versions** tab. The system lists each version of the association you created and edited.
10. Open the Amazon S3 console at [https://console.aws.amazon.com/s3/](https://console.aws.amazon.com/s3/).
11. Choose your bucket name, and then choose an instance ID that ran the association.
12. Drill down several levels, through the awsrunPowerShell folder, to the **stdout** file.
13. Choose the file, and then choose the link in the **Link** section. The host name is listed in the **HostId** tags.

---

### Creating Associations Using the Targets Parameter

You can create associations on tens, hundreds, or thousands of instances by using the `targets` parameter. The `targets` parameter accepts a `Key,Value` combination based on Amazon EC2 tags that you specified for your instances. When you execute the request to create the association, the system locates and attempts to create the association on all instances that match the specified criteria. For more information about the `targets` parameter, see, [Sending Commands to a Fleet](p. 177). For more information about Amazon EC2 tags, see [Tagging Your Amazon EC2 Resources](p. 177) in the *Amazon EC2 User Guide*.

The following AWS CLI examples show you how to use the `targets` parameter when creating associations.

```shell
aws ssm create-association --targets Key=tag:TagKey,Values=TagValue --name AWS-UpdateSSMAgent --schedule "cron(0 0 2 ? * SUN *)"
```

Create an association for a managed instance named "ws-0123456789012345"

```shell
aws ssm create-association --name value --targets "Key=Instance Ids,Values=ws-0123456789" --name AWS-UpdateSSMAgent --schedule "cron(0 0 2 ? * SUN *)"
```

**Note**

If you remove an instance from a tagged group that's associated with a document, then the instance will be dissociated from the document.

---

### Systems Manager State Manager Walkthroughs

The following walkthroughs can help you understand how to create and configure State Manager associations by using the Amazon EC2 console or the AWS CLI. These walkthrough can also help you understand how to automatically perform common administrative tasks by using State Manager.

**Contents**

- **Walkthrough: Automatically Update the SSM Agent** (p. 254)
- **Walkthrough: Automatically Update PV Drivers on EC2 Windows Instances** (p. 256)

---

### Walkthrough: Automatically Update the SSM Agent

The following procedure walks you through the process of creating a State Manager association using the AWS Command Line Interface (AWS CLI). The association automatically updates the SSM Agent according to a schedule that you specify. For more information about the SSM Agent, see [Installing and Configuring SSM Agent](p. 13).

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

Before you complete the following procedure, verify that you have at least one running Amazon EC2 instance (Linux or Windows) that is configured for Systems Manager. For more information, see Systems Manager Prerequisites (p. 4).

To create an association for automatically updating the SSM Agent

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 a Region. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM).

   ```bash
   aws configure
   ```

   The system prompts you to specify the following.

<table>
<thead>
<tr>
<th>AWS Access Key ID [None]:</th>
<th>key_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Secret Access Key [None]:</td>
<td>key_name</td>
</tr>
<tr>
<td>Default region name [None]:</td>
<td>region</td>
</tr>
<tr>
<td>Default output format [None]:</td>
<td>ENTER</td>
</tr>
</tbody>
</table>

3. Execute the following command to create an association by targeting instances using Amazon EC2 tags. The Schedule parameter sets a schedule to run the association every Sunday morning at 2:00 a.m. (UTC).

   ```bash
   aws ssm create-association --targets Key=tag:TagKey,Values=TagValue --name AWS-UpdateSSM Agent --schedule-expression "cron(0 0 2 ? * SUN *)"
   ```

   **Note**

   State Manager associations do not support all cron and rate expressions. For more information about creating cron and rate expressions for associations, see Working with Cron and Rate Expressions for Systems Manager (p. 33).

   If you want, you can also target multiple instances by specifying instances IDs in a comma-separated list.

   ```bash
   aws ssm create-association --targets Key=instanceids,Values=InstanceID,InstanceID,InstanceID --name your document name --schedule-expression "cron(0 0 2 ? * SUN *)"
   ```

   The system returns information like the following.

   ```json
   {
       "AssociationDescription": {
           "ScheduleExpression": "cron(0 0 2 ? * SUN *)",
           "Name": "AWS-UpdateSSM Agent",
           "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 immediately apply the state. The association status shows Pending.

4. Execute the following command to view an updated status of the association you just created.

```bash
aws ssm list-associations
```

**Note**

If your instances are currently running the most recent version of the SSM Agent, the status shows Failed. This is expected behavior. When a new version of the SSM Agent is published, the association automatically installs the new agent, and the status shows Success.

## Walkthrough: Automatically Update PV Drivers on EC2 Windows Instances

Amazon Windows AMIs contain a set of drivers to permit access to virtualized hardware. These drivers are used by Amazon EC2 to map instance store and Amazon EBS volumes to their devices. We recommend that you install the latest drivers to improve stability and performance of your EC2 Windows instances. For more information about PV drivers, see [AWS PV Drivers](#).

The following walkthrough shows you how to configure a State Manager association to automatically download and install new AWS PV drivers when the drivers become available.

### Before You Begin

Before you complete the following procedure, verify that you have at least one Amazon EC2 Windows instance running that is configured for Systems Manager. For more information, see Systems Manager Prerequisites (p. 4).

The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

### To create a State Manager association that automatically updates PV drivers

1. Open the Amazon EC2 console, expand Systems Manager Services in the navigation pane, and then choose State Manager.
2. Choose Create Association.
3. In the Association Name field, type a descriptive name.
4. In the Select Document list, choose AWS-ConfigureAWSPackage.
5. In the Select Targets by section, choose an option.

**Note**

If you choose to target instances by using tags, and you specify tags that map to Linux instances, the association succeeds on the Windows instance, but fails on the Linux instances. The overall status of the association shows Failed.
6. In the **Schedule** section, choose an option. Updated PV drivers are only released a few times a year, so you can schedule the association to run once a month, if you want.

7. In the **Parameters** section, choose **Install** from the **Action** list.

8. From the **Name** list, choose **AWSPVDriver**. You can leave the **Version** field empty.

9. In the **Advanced** section, choose **Write to S3** if you want to write association details to an Amazon S3 bucket.

10. Disregard the **S3Region** field. This field is deprecated. Specify the name of your bucket in the **S3Bucket Name** field. If you want to write output to a sub-folder, specify the sub-folder name in the **S3Key Prefix** field.

11. Choose **Create Association**, and then choose **Close**. The system attempts to create the association on the instance(s) and immediately apply the state. The association status shows **Pending**.

12. In the right corner of the **Association** page, choose the refresh button. If you created the association on one or more EC2 Windows instances, the status changes to **Success**. If your instances are not properly configured for Systems Manager, or if you inadvertently targeted Linux instances, the status shows **Failed**.

13. If the status is **Failed**, choose the **Instances** tab and verify that the association was successfully created on your EC2 Windows instances. If Windows instances show a status of **Failed**, verify that the SSM Agent is running on the instance, and verify that the instance is configured with an IAM role for Systems Manager. For more information, see **Systems Manager Prerequisites (p. 4)**.
AWS Systems Manager Managed Instances

A managed instance is any machine configured for AWS Systems Manager. You can configure Amazon EC2 instances or on-premises machines in a hybrid environment as managed instances. Systems Manager supports various distributions of Linux, including Raspberry Pi devices, and Microsoft Windows.

In the console, any machine prefixed with "mi-" is an on-premises server or virtual machine (VM) managed instance.

For information about Systems Manager prerequisites, see Systems Manager Prerequisites (p. 4). For information about configuring on-premises servers and VMs as managed instances, see Setting Up AWS Systems Manager in Hybrid Environments (p. 27).

AWS Systems Manager Activations

To set up servers and virtual machines (VMs) in your hybrid environment as managed instances, you create a managed-instance activation. After you complete the activation, you receive an activation code and ID. This code/ID combination functions like an Amazon EC2 access ID and secret key to provide secure access to the Systems Manager service from your managed instances. For information about configuring on-premises servers and VMs as managed instances, see Setting Up AWS Systems Manager in Hybrid Environments (p. 27).

AWS Systems Manager Documents

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

Types of SSM Documents

AWS Systems Manager Shared Resources

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

Topics
  • AWS Systems Manager Managed Instances (p. 258)
  • AWS Systems Manager Activations (p. 258)
  • AWS Systems Manager Documents (p. 258)
  • AWS Systems Manager Parameter Store (p. 330)
The following table describes the different types of SSM documents.

<table>
<thead>
<tr>
<th>Type</th>
<th>Use with</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command document</td>
<td>Run Command (p. 160)</td>
<td>Run Command uses command documents to execute commands. State Manager uses command documents to apply a configuration. These actions can be run on one or more targets at any point during the lifecycle of an instance,</td>
</tr>
<tr>
<td></td>
<td>State Manager (p. 249)</td>
<td></td>
</tr>
<tr>
<td>Policy document</td>
<td>State Manager (p. 249)</td>
<td>Policy documents enforce a policy on your targets. If the policy document is removed, the policy action (for example, collecting inventory) no longer happens.</td>
</tr>
<tr>
<td>Automation document</td>
<td>Automation (p. 92)</td>
<td>Use automation documents when performing common maintenance and deployment tasks such as creating or updating an Amazon Machine Image (AMI).</td>
</tr>
</tbody>
</table>

SSM Document Versions and Execution

You can create and save different versions of documents. You can then specify a default version for each document. The default version of a document can be updated to a newer version or reverted to an older version of the document. When you change the content of a document, Systems Manager automatically increments the version of the document. You can retrieve and use previous versions of a document.

Customizing a Document

If you want to customize the steps and actions in a document, you can create your own. The first time you use a document to perform an action on an instance, the system stores the document with your AWS account. For more information about how to create a Systems Manager document, see Creating Systems Manager Documents (p. 267).

Tagging a Document

You can tag your documents to help you quickly identify one or more documents based on the tags you've assigned to them. For example, you can tag documents for specific environments, departments, users, groups, or periods. You can also restrict access to documents by creating an IAM policy that specifies the tags that a user or group can access. For more information, see Tagging Systems Manager Documents (p. 270).

Sharing a Document

You can make your documents public or share them with specific AWS accounts. For more information, see Sharing Systems Manager Documents (p. 273).

SSM Document Limits

For information about SSM document limits, see AWS Systems Manager Limits.
Systems Manager Pre-Defined Documents

To help you get started quickly, Systems Manager provides pre-defined documents. To view these documents in the AWS Systems Manager console, in the left navigation, choose Documents. After you choose a document, choose View details to view information about the document you selected.

To view these documents in the Amazon EC2 console, expand Systems Manager Shared Resources, and then choose Documents. After you choose a document, use the tabs in the lower pane to view information about the document you selected.

You can also use the AWS CLI and Tools for Windows PowerShell commands to view a list of documents and get descriptions about those documents.

To view information about documents using the AWS CLI, run the following commands:

```
aws ssm list-documents

aws ssm describe-document --name "document_name"
```

To view information about documents using the Tools for Windows PowerShell, run the following commands:

```
Get-SSMDocumentList

Get-SSMDocumentDescription -Name "document_name"
```

SSM Document Schemas and Features

Systems Manager documents currently use the following schema versions.

- Documents of type Command can use schema version 1.2, 2.0, and 2.2. If you are currently using schema 1.2 documents, we recommend that you create documents that use schema version 2.2.
- Documents of type Policy must use schema version 2.0 or later.
- Documents of type Automation must use schema version 0.3.
- You can create documents in JSON or YAML.
By using the latest schema version for Command and Policy documents, you can take advantage of the following features.

**Schema Version 2.2 Document Features**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document editing</td>
<td>Documents can now be updated. With version 1.2, any update to a document required that you save it with a different name.</td>
</tr>
<tr>
<td>Automatic versioning</td>
<td>Any update to a document creates a new version. This is not a schema version, but a version of the document.</td>
</tr>
<tr>
<td>Default version</td>
<td>If you have multiple versions of a document, you can specify which version is the default document.</td>
</tr>
<tr>
<td>Sequencing</td>
<td>Plugins or steps in a document execute in the order that you specified.</td>
</tr>
<tr>
<td>Cross-platform support</td>
<td>Cross-platform support enables you to specify different operating systems for different plugins within the same SSM document. Cross-platform support uses the precondition parameter within a step.</td>
</tr>
</tbody>
</table>

**Note**
You must keep the SSM Agent on your instances updated with the latest version to use new Systems Manager features and SSM document features. For more information, see Example: Update the SSM Agent (p. 176).

The following table lists the differences between major schema versions.

<table>
<thead>
<tr>
<th>Version 1.2</th>
<th>Version 2.0</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>runtimeConfig</td>
<td>mainSteps</td>
<td>In version 2.0 or later, the mainSteps section replaces runtimeConfig. The mainSteps section enables Systems Manager to execute steps in sequence.</td>
</tr>
<tr>
<td>properties</td>
<td>inputs</td>
<td>In version 2.0 or later, the inputs section replaces the properties section. The inputs section accepts parameters for steps.</td>
</tr>
<tr>
<td>commands</td>
<td>runCommand</td>
<td>In version 2.0 or later, the inputs section takes the runCommand parameter instead of the commands parameter.</td>
</tr>
<tr>
<td>id</td>
<td>action</td>
<td>In version 2.0 or later, Action replaces ID. This is just a name change.</td>
</tr>
</tbody>
</table>
Using the Precondition Parameter

With schema version 2.2 or higher, you can use the precondition parameter to specify the target operating system for each plugin. The precondition parameter supports platformType and a value of either Windows or Linux.

For documents that use schema version 2.2 or higher, if precondition is not specified, each plugin is either executed or skipped based on the plugin's compatibility with the operating system. For documents that use schema 2.0 or earlier, incompatible plugins throw an error.

For example, in a schema version 2.2 document, if precondition is not specified and the `aws:runShellScript` plugin is listed, then the step executes on Linux instances, but the system skips it on Windows instances because the `aws:runShellScript` is not compatible with Windows instances. However, for a schema version 2.0 document, if you specify the `aws:runShellScript` plugin, and then run the document on a Windows instances, the execution fails. You can see an example of the the precondition parameter in an SSM document later in this section.

SSM Document Syntax

The syntax of your document is defined by the schema version used to create it. We recommended that you use schema version 2.2 or higher. Documents that use this schema version include the following top-level elements. For information about the properties that you can specify in these elements, see Top-level Elements (p. 285).

- **schemaVersion**: The schema version to use.
- **Description**: Information you provide to describe the purpose of the document.
- **Parameters**: The parameters the document accepts. For parameters that you reference often, we recommend that you store those parameters in Systems Manager Parameter Store and then reference them. You can reference String and StringList Systems Manager parameters in this section of a document. You can't reference Secure String Systems Manager parameters in this section of a document. For more information, see AWS Systems Manager Parameter Store (p. 330).
- **mainSteps**: An object that can include multiple steps (plugins). Steps include one or more actions, an optional precondition, a unique name of the action, and inputs (parameters) for those actions. For a list of supported plugins and plugin properties, see SSM Document Plugin Reference (p. 284).

**Important**

The name of the action can't include a space. If a name includes a space, you will receive an InvalidDocumentContent error.

Schema Version 2.0

The following example shows the top-level elements of a schema version 2.0 or higher document in JSON.

```json
{
  "schemaVersion":"2.0",
  "description":"A description of the document.",
  "parameters":{
```
SSM Document Syntax

YAML Schema Version 2.2 Example

You can use the following YAML document with Run Command to return the hostname of one or more instances.

```yaml
---
schemaVersion: '2.2'
description: Sample document
mainSteps:
  - action: aws:runPowerShellScript
    name: runPowerShellScript
    inputs:
      runCommand:
        - hostname
```

Schema Version 2.2 Precondition Parameter Example

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.

```json
{
  "schemaVersion":"2.2",
  "description":"cross-platform sample",
  "mainSteps": [
    {
      "action":"aws:runPowerShellScript",
      "name":"PatchWindows",
      "precondition":{
        "StringEquals":{
          "platformType", "Windows"
        }
      },
      "inputs":{
        "runCommand":[
          "cmds"
        ]
      }
    }
  ]
}
```
You can use the following YAML document with State Manager to download and install the ClamAV antivirus software. State Manager enforces a specific configuration, which means that each time the State Manager association is run, the system checks to see if the ClamAV software is installed. If not, State Manager reruns this document.

```yaml
---
schemaVersion: '2.0'
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
```
networkConfig:
  type: String
  default: Enabled
  description: "(Optional) Collect data for Network configurations."
  allowedValues:
  - Enabled
  - Disabled
windowsUpdates:
  type: String
  default: Enabled
  description: "(Optional) Collect data for all Windows Updates."
  allowedValues:
  - Enabled
  - Disabled
instanceDetailedInformation:
  type: String
  default: Enabled
  description: "(Optional) Collect additional information about the instance, including
the CPU model, speed, and the number of cores, to name a few."
  allowedValues:
  - Enabled
  - Disabled
customInventory:
  type: String
  default: Enabled
  description: "(Optional) Collect data for custom inventory."
  allowedValues:
  - Enabled
  - Disabled

mainSteps:
- action: aws:softwareInventory
  name: collectSoftwareInventoryItems
  inputs:
    applications: "{{ applications }}"
    awsComponents: "{{ awsComponents }}"
    networkConfig: "{{ networkConfig }}"
    windowsUpdates: "{{ windowsUpdates }}"
    instanceDetailedInformation: "{{ instanceDetailedInformation }}"
    customInventory: "{{ customInventory }}"

Schema Version 2.0 AWS-ConfigureAWSPackage Example

The following example shows the AWS-ConfigureAWSPackage document. The mainSteps section includes the aws:configurePackage plugin in the action step.

```json
{
  "schemaVersion": "2.0",
  "description": "Install or uninstall the latest version or specified version of an AWS
package.",
  "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",
      "allowedValues": ["AWSPVDriver"]
    }
  }
}```
SSM Document Syntax

Note
The aws:applications, aws:psModule, aws:runShellScript, and aws:runPowerShellScript plugins allow you to specify an array of properties.

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 Systems Manager document.",
  "parameters":{
    "parameter 1":{
      "one or more parameter properties"
    },
    "parameter 2":{
      "one or more parameter properties"
    },
    "parameter 3":{
      "one or more parameter properties"
    }
  },
  "runtimeConfig":{
    "plugin 1":{
      "properties":[
        { "one or more plugin properties"
      ]
    }
  }
}
```

Schema Version 1.2 Example

The following example shows the AWS-RunShellScript Systems Manager document. The `runtimeConfig` section includes the `aws:runShellScript` plugin.

```
{
  "schemaVersion":"1.2",
```
Creating Systems Manager Documents

If the Systems Manager public documents limit the actions you want to perform on your managed instances, you can create your own documents. When creating a new document, we recommend that you use schema version 2.2 or later.

Before You Begin

Before you create an SSM document, we recommend that you read about the different schemas, features, and syntax available for SSM documents. For more information, see AWS Systems Manager Documents (p. 258).

Note

If you plan to create an SSM document for State Manager, be aware of the following details:

- You can assign multiple documents to a target by creating different State Manager associations that use different documents.
- You can use a shared document with State Manager, as long as you have permission, but you can't associate a shared document to an instance. If you want to use or share a document that is associated with one or more targets, you must create a copy of the document and then use or share it.
- If you create a document with conflicting plugins (e.g., domain join and remove from domain), the last plugin executed will be the final state. State Manager does not validate the logical sequence or rationality of the commands or plugins in your document.
Creating Systems Manager Documents

- When processing documents, instance associations are applied first, and next tagged group associations are applied. If an instance is part of multiple tagged groups, then the documents that are part of the tagged group will not be executed in any particular order. If an instance is directly targeted through multiple documents by its instance ID, there is no particular order of execution.

- If you change the default version of an State Manager document, any association that uses the document will start using the new default version the next time Systems Manager applies the association to the instance.

If you create an SSM document for State Manager, you must associate the document with your managed instances after you add it the system. For more information, see Creating an Association (p. 252).

Topics
- Copy a Document (p. 268)
- Add a Systems Manager Document (Console) (p. 269)
- Create an SSM Document (AWS CLI) (p. 269)
- Create an SSM Document (Tools for Windows PowerShell) (p. 270)

Copy a Document

When you create a document, you specify the contents of the document in JSON or YAML. The easiest way to get started creating SSM documents is to copy an existing sample from one of the Systems Manager public documents. The following example shows you how to copy a JSON sample.

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

**To copy a Systems Manager document (AWS Systems Manager)**

2. In the navigation pane, choose **Documents**.
   -or-
   If the AWS Systems Manager home page opens, scroll down and choose any **Explore...** button, and then choose **Documents** in the navigation pane.
3. Choose a document.
4. Choose **View details**.
5. Choose the **Content** tab.
6. Copy the JSON to a text editor and specify the details for your custom document.
7. Save the file with a .json file extension.

**To copy a Systems Manager document (Amazon EC2 Systems Manager)**

1. In the Amazon EC2 console, expand **Systems Manager Shared Resources**, and then choose **Documents**.
2. Choose a document.
3. In the lower pane, choose the **Content** tab.
4. Copy the JSON to a text editor and specify the details for your custom document.
5. Save the file with a `.json` file extension.

After you author the content of the document, you can add it SSM using any one of the following procedures.

**Add a Systems Manager Document (Console)**

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

**Add a Systems Manager Document (AWS Systems Manager)**

2. In the navigation pane, choose **Documents**.
   - or -
   If the AWS Systems Manager home page opens, scroll down and choose any **Explore...** button, and then choose **Documents** in the navigation pane.
3. Choose **Create document**.
4. Type a descriptive name for the document
5. In the **Document type** list, choose the type of document you want to create.
6. Delete the brackets in the **Content** field, and then paste the document you created earlier.
7. Choose **Create document** to save the document.

**Add a Systems Manager Document (Amazon EC2 Systems Manager)**

1. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).
2. In the navigation pane, choose **Documents**.
3. Choose **Create Document**.
4. Type a descriptive name for the document
5. In the **Document Type** list, choose the type of document you want to create.
6. Delete the brackets in the **Content** field, and then paste the document you created earlier.
7. Choose **Create Document** to save the document.

**Create an SSM Document (AWS CLI)**

1. Copy and customize an existing document, as described in Copy a Document (p. 268).
2. Add the document using the AWS CLI.

```bash
aws ssm create-document --content file://path to your file\your file --name "document name" --document-type "Command"
```

**Windows example**

```bash
aws ssm create-document --content file://c:\temp\PowershellScript.json --name "PowershellScript" --document-type "Command"
```

**Linux example**
Create an SSM Document (Tools for Windows PowerShell)

1. Copy and customize an existing document, as described in Copy a Document (p. 268).

```bash
$json = Get-Content C:\your file | Out-String
New-SSMDocument -DocumentType Command -Name document name -Content $json
```

Tagging Systems Manager Documents

You can use the Systems Manager console, the AWS CLI, the AWS Tools for Windows, or the AddTagsToResource API to add tags to Systems Manager resources, including documents, managed instances, Maintenance Windows, Parameter Store parameters, and patch baselines.

Tagging is useful when you have many resources of the same type — you can quickly identify a specific resource based on the tags you've assigned to it. Each tag consists of a key and an optional value, both of which you define.

For example, you can tag documents for specific environments, departments, users, groups, or periods. After you tag a document, you can restrict access to it by creating an IAM policy that specifies the tags that a user can access. For more information about restricting access to documents by using tags, see Controlling Access to Documents Using Tags (p. 271).

Topics

- Tag a Document (AWS CLI) (p. 270)
- Tag a Document (AWS Tools for Windows) (p. 271)
- Tag a Document (Console) (p. 271)
- Controlling Access to Documents Using Tags (p. 271)

Tag a Document (AWS CLI)

1. At a terminal (Linux, macOS, or Unix) or command prompt (Windows), run the list-documents command to list the documents that you can tag.

```bash
aws ssm list-documents
```

Note the name of a document that you want to tag.

2. Run the following command to tag a document.

```bash
aws ssm add-tags-to-resource --resource-type "Document" --resource-id "document-name" --tags "Key=key,Value=value"
```

`document-name` the name of the Systems Manager document you want to tag.

`key` is the name of a custom key you supply. For example, `region` or `quarter`. 
Tagging Documents

- `value` is the custom content for the value you want to supply for that key. For example, `west` or `Q318`.

If successful, the command has no output.

3. Execute the following command to verify the document tags.

```
aws ssm list-tags-for-resource --resource-type "Document" --resource-id "document-name"
```

Tag a Document (AWS Tools for Windows)

1. Open AWS Tools for Windows PowerShell and run the following command to list documents that you can tag:

```
Get-SSMDocumentList
```

2. Run the following commands one at a time to tag a document:

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "key"
$tag1.Value = "value"
Add-SSMResourceTag -ResourceType "Document" -ResourceId "document-name" -Tag $tag1
```

- `document-name` is the name of the Systems Manager document you want to tag.
- `key` is the name of a custom key you supply. For example, `region` or `quarter`.
- `value` is the custom content for the value you want to supply for that key. For example, `west` or `Q318`.

If successful, the command has no output.

3. Run the following command to verify the document tags:

```
Get-SSMResourceTag -ResourceType "Document" -ResourceId "document-name"
```

Tag a Document (Console)

2. In the left navigation, choose Documents.
3. Choose the name of an existing document, and then choose the Tags tab.
4. In the first box, enter a key for the tag, such as Region.
5. In the second box, enter a value for the tag, such as West.
6. Choose Save.

Controlling Access to Documents Using Tags

After you tag a document, you can restrict access to it by creating an IAM policy that specifies the tags the user can access. When a user attempts to use a document, the system checks the IAM policy and the tags specified for the document. If the user does not have access to the tags assigned to the document, the user receives an access denied error. Use the following procedure to create an IAM policy that restricts access to documents by using tags.
Before You Begin

Create and tag documents. For more information, see Tagging Systems Manager Documents (p. 270).

To restrict a user's access to documents by using tags

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.
4. Copy the following sample policy and paste it into the text field, replacing the sample text. Replace tag_key and tag_value with the key-value pair for your tag.

   ```json
   {  
     "Version": "2012-10-17",
     "Statement": [  
       {  
         "Effect": "Allow",
         "Action": [  
           "ssm:GetDocument"
         ],
         "Resource": "*",
         "Condition": {  
           "StringLike": {  
             "ssm:resourceTag/tag_key": [  
               "tag_value"
             ]
           }
         }
       }
     ]
   }
   
   You can specify multiple keys in the policy by using the following Condition format. Specifying multiple keys creates an AND relationship for the keys.

   ```json
   "Condition": {  
     "StringLike": {  
       "ssm:resourceTag/tag_key1": [  
         "tag_value1"
       ],
       "ssm:resourceTag/tag_key2": [  
         "tag_value2"
       ]
     }
   }
   
   You can specify multiple values in the policy by using the following Condition format. ForAnyValue establishes an OR relationship for the values. You can also specify ForAllValues to establish an AND relationship.

   ```json
   "Condition": {  
     "ForAnyValue:StringLike": {  
       "ssm:resourceTag/tag_key1": [  
         "tag_value1",
         "tag_value2"
       ]
     }
   }
   
   5. Choose Review policy.
6. In the Name field, specify a name that identifies this as a user policy for tagged documents.
7. Enter a description.
8. Verify details of the policy in the Summary section.
9. Choose Create policy.
10. Assign the policy to IAM users or groups. For more information, see Changing Permissions for an IAM User and Attaching a Policy to an IAM Group.

After you attach the policy to the IAM user or group account, if a user tries to use a document and the user's policy does not allow the user to access a tag for the document (call the GetDocument API), the system returns an error. The error is similar to the following:

"User: user_name is not authorized to perform: ssm:GetDocument on resource: document-name with the following command."

If a document has multiple tags, the user will still receive the Access Denied error if the user does not have permission to access any one of those tags.

Sharing Systems Manager Documents

You can share Systems Manager documents privately or publicly. To privately share a document, you modify the document permissions and allow specific individuals to access it according to their Amazon Web Services (AWS) ID. To publicly share a Systems Manager document, you modify the document permissions and specify All.

Warning
Use shared Systems Manager 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 Guidelines for Sharing and Using Shared Systems Manager Documents (p. 274).

Limitations

As you begin working with Systems Manager 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 Document (p. 277).
• You can share a document with a maximum of 1000 AWS accounts. To increase this limit, go to AWS Support Center and submit a limit increase request form.
• You can publicly share a maximum of five Systems Manager documents. To increase this limit, go to AWS Support Center and submit a limit increase request form.

For more information about Systems Manager limits, see AWS Systems Manager Limits.

Contents
• Guidelines for Sharing and Using Shared Systems Manager Documents (p. 274)
• Share a Systems Manager Document (p. 274)
• Modify Permissions for a Shared Document (p. 277)
• Use a Shared Systems Manager Document (p. 278)
Guidelines for Sharing and Using Shared Systems Manager Documents

Review the following guidelines before you share or use a shared document.

Remove Sensitive Information

Review your Systems Manager document carefully and remove any sensitive information. For example, verify that the document does not include your AWS credentials. If you share a document with specific individuals, those users can view the information in the document. If you share a document publicly, anyone can view the information in the document.

Limit Run Command Actions Using an IAM User Trust Policy

Create a restrictive AWS Identity and Access Management (IAM) user policy for users who will have access to the document. The IAM policy determines which Systems Manager documents a user can see in either the Amazon EC2 console or by calling ListDocuments using the AWS CLI or AWS Tools for Windows PowerShell. The policy also limits the actions the user can perform with Systems Manager document. You can create a restrictive policy so that a user can only use specific documents. For more information, see Configuring Access to Systems Manager (p. 8).

Review the Contents of a Shared Document Before Using It

Review the contents of every document that is shared with you, especially public documents, to understand the commands that will be executed on your instances. A document could intentionally or unintentionally have negative repercussions after it is run. If the document references an external network, review the external source before you use the document.

Send Commands Using the Document Hash

When you share a document, the system creates a Sha-256 hash and assigns it to the document. The system also saves a snapshot of the document content. When you send a command using a shared document, you can specify the hash in your command to ensure that the following conditions are true:

- You are executing a command from the correct Systems Manager document
- The content of the document has not changed since it was shared with you.

If the hash does not match the specified document or if the content of the shared document has changed, the command returns an InvalidDocument exception. Note: The hash cannot verify document content from external locations.

Share a Systems Manager Document

You can share Systems Manager document by using the Amazon EC2 console, the AWS Systems Manager console, or by programmatically calling the ModifyDocumentPermission API operation using the AWS CLI, AWS Tools for Windows PowerShell, or the AWS SDK. Before you share a document, get the AWS account IDs of the people with whom you want to share. You will specify these account IDs when you share the document.

Topics

- Share a Document (Console) (p. 275)
- Share a Document (AWS CLI) (p. 275)
- Share a Document (AWS Tools for Windows PowerShell) (p. 276)
Share a Document (Console)

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

Share a document (AWS Systems Manager)

2. In the navigation pane, choose Documents.
   -or-
   If the AWS Systems Manager home page opens, scroll down and choose any Explore... button, and then choose Documents in the navigation pane.
3. In the documents list, choose the document you want to share, and then choose View details. On the Permissions tab, verify that you are the document owner. Only a document owner can share a document.
4. Choose Edit.
5. To share the command publicly, choose Public and then choose Save. To share the command privately, choose Private, enter the AWS account ID, choose Add permission, and then choose Save.

Share a document (Amazon EC2 Systems Manager)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Documents.
3. In the documents list, choose the document you want to share. Choose the Permissions tab and verify that you are the document owner. Only a document owner can share a document.
4. Choose Edit.
5. To share the command publicly, choose Public and then choose Save. To share the command privately, choose Private, enter the AWS account ID, choose Add Permission, and then choose Save.

Share a Document (AWS CLI)

The following procedure requires that you specify a region for your CLI session. Run Command is currently available in the following Systems Manager regions.

1. Open the AWS CLI on your local computer and execute the following command to specify your credentials.

   ```bash
   aws config
   
   AWS Access Key ID: [your key]
   AWS Secret Access Key: [your key]
   Default region name: [us-east-1]
   Default output format [None]:
   ```

2. Use the following command to list all of the Systems Manager documents that are available for you. The list includes documents that you created and documents that were shared with you.

   ```bash
   aws ssm list-documents --document-filter-list key=Owner,value=all
   ```

3. Use the following command to get a specific document.

   ```bash
   aws ssm get-document --name document name
   ```
4. Use the following command to get a description of the document.

```bash
aws ssm describe-document --name document name
```

5. Use the following command to view the permissions for the document.

```bash
aws ssm describe-document-permission --name document name --permission-type Share
```

6. Use the following command to modify the permissions for the document and share it. You must be the owner of the document to edit the permissions. This command privately shares the document with a specific individual, based on that person's AWS account ID.

```bash
aws ssm modify-document-permission --name document name --permission-type Share --account-ids-to-add AWS account ID
```

Use the following command to share a document publicly.

```bash
aws ssm modify-document-permission --name document name --permission-type Share --account-ids-to-add 'all'
```

**Share a Document (AWS Tools for Windows PowerShell)**

The following procedure requires that you specify a region for your PowerShell session. Run Command is currently available in the following Systems Manager regions.

1. Open **AWS Tools for Windows PowerShell** on your local computer and execute the following command to specify your credentials.

```powershell
Set-AWSCredentials -AccessKey your key -SecretKey your key
```

2. Use the following command to set the region for your PowerShell session. The example uses the us-west-2 region.

```powershell
Set-DefaultAWSRegion -Region us-west-2
```

3. Use the following command to list all of the Systems Manager documents available for you. The list includes documents that you created and documents that were shared with you.

```powershell
Get-SSMDocumentList -DocumentFilterList (@{"key"="Owner";"value"="All"})
```

4. Use the following command to get a specific document.

```powershell
Get-SSMDocument -Name document name
```

5. Use the following command to get a description of the document.

```powershell
Get-SSMDocumentDescription -Name document name
```

6. Use the following command to view the permissions of the document.

```powershell
Get-SSMDocumentPermission -Name document name -PermissionType Share
```

7. Use the following command to modify the permissions for the document and share it. You must be the owner of the document to edit the permissions. This command privately shares the document with a specific individual, based on that person's AWS account ID.
Use the following command to share a document publicly.

```
Edit-SSMDocumentPermission -Name document name -PermissionType Share -AccountIdsToAdd AWS account ID
```

Modify Permissions for a Shared Document

If you share a command, users can view and use that command until you either remove access to the Systems Manager document or delete the Systems Manager document. However, you cannot delete a document as long as it is shared. You must stop sharing it first and then delete it.

Topics
- Stop Sharing a Document (Console) (p. 277)
- Stop Sharing a Document (AWS CLI) (p. 278)
- Stop Sharing a Document (AWS Tools for Windows PowerShell) (p. 278)

Stop Sharing a Document (Console)

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

Stop sharing a document (AWS Systems Manager)

2. In the navigation pane, choose Documents.
   -or-
   If the AWS Systems Manager home page opens, scroll down and choose any Explore... button, and then choose Documents in the navigation pane.
3. In the documents list, choose the document you want to stop sharing, and then choose View details. On the Permissions tab, verify that you are the document owner. Only a document owner can stop sharing a document.
4. Choose Edit.
5. Choose X to delete the AWS account ID that should no longer have access to the command, and then choose Save.

Stop sharing a document (Amazon EC2 Systems Manager)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Documents.
3. In the documents list, choose the document you want to stop sharing. Choose the Permissions tab and verify that you are the document owner. Only a document owner can stop sharing a document.
4. Choose Edit.
5. Delete the AWS account ID that should no longer have access to the command, and then choose Save.
Stop Sharing a Document (AWS CLI)

Open the AWS CLI on your local computer and execute the following command to stop sharing a command.

```bash
aws ssm modify-document-permission --name document name --permission-type Share --account-ids-to-remove 'AWS account ID'
```

Stop Sharing a Document (AWS Tools for Windows PowerShell)

Open AWS Tools for Windows PowerShell on your local computer and execute the following command to stop sharing a command.

```powershell
Edit-SSMDocumentPermission -Name document name -AccountIdsToRemove AWS account ID -PermissionType Share
```

Use a Shared Systems Manager Document

When you share a Systems Manager document, the system generates an Amazon Resource Name (ARN) and assigns it to the command. If you select and execute a shared document from the Amazon EC2 console, you do not see the ARN. However, if you want to execute a shared Systems Manager document from a command line application, you must specify a full ARN. You are shown the full ARN for a Systems Manager document when you execute the command to list documents.

**Note**

You are not required to specify ARNs for AWS public documents (documents that begin with AWS-* or commands that you own.

Topics

- Use a Shared Systems Manager Document (AWS CLI) (p. 278)
- Use a Shared Systems Manager Document (AWS Tools for Windows PowerShell) (p. 279)

Use a Shared Systems Manager Document (AWS CLI)

**To list all public Systems Manager documents**

```bash
aws ssm list-documents --document-filter-list key=Owner,value=Public
```

**To list private Systems Manager documents that have been shared with you**

```bash
aws ssm list-documents --document-filter-list key=Owner,value=Private
```

**To list all Systems Manager documents available to you**

```bash
aws ssm list-documents --document-filter-list key=Owner,value=All
```

**Execute a command from a shared Systems Manager document using a full ARN**

```bash
aws ssm send-command --document-name FullARN/name
```

For example:

```bash
```
Use a Shared Systems Manager Document (AWS Tools for Windows PowerShell)

To list all public Systems Manager documents

```powershell
Get-SSMDocumentList -DocumentFilterList @(
```

To list private Systems Manager documents that have been shared with you

```powershell
Get-SSMDocumentList -DocumentFilterList @(
```

To get information about a Systems Manager document that has been shared with you

```powershell
Get-SSMDocument -Name FullARN/name
```

For example:

```powershell
```

To get a description of a Systems Manager document that has been shared with you

```powershell
Get-SSMDocumentDescription -Name FullARN/name
```

For example:

```powershell
```

To execute a command from a shared Systems Manager document using a full ARN

```powershell
Send-SSMCommand -DocumentName FullARN/name -InstanceId IDS
```

For example:

```powershell
Send-SSMCommand -DocumentName arn:aws:ssm:us-east-1:555450671542:document/highAvailabilityServerSetup -InstanceId @("i-273d4e9e")
```

Creating Composite Documents

A composite SSM document is a custom document that performs a series of actions by running one or more secondary SSM documents. Composite documents promote *infrastructure as code* by enabling you to create a standard set of SSM documents for common tasks such as boot-strapping software or domain-joining instances. You can then share these documents across AWS accounts to reduce SSM document maintenance and ensure consistency.

For example, you can create a composite document that performs the following actions:

1. Updates SSM Agent to the latest version.
2. Installs all whitelisted patches.
3. Installs antivirus software.
4. Downloads scripts from GitHub and runs them.
In this example, your custom SSM document includes the following plugins to perform these actions:

1. The `aws:runDocument` plugin to run the `AWS-UpdateSSMAgent` document, which updates SSM Agent to the latest version.
2. The `aws:runDocument` plugin to run the `AWS-ApplyPatchBaseline` document, which installs all whitelisted patches.
3. The `aws:runDocument` plugin to run the `AWS-InstallApplication` document, which installs the antivirus software.
4. The `aws:downloadContent` plugin to download scripts from GitHub and run them.

Composite and secondary documents can be stored in Systems Manager, GitHub (public and private repositories), or Amazon S3. Composite documents and secondary documents can be created in JSON or YAML.

**Note**

Composite documents can only run to a maximum depth of three documents. This means that a composite document can call a child document; and that child document can call one last document.

### Create a Composite Document

To create a composite document, add the `aws:runDocument` (p. 300) plugin in a custom SSM document and specify the required inputs. The following is an example of a composite document that performs the following actions:

1. Runs the `aws:downloadContent` (p. 295) plugin to download an SSM document from a GitHub public repository to a local directory called `bootstrap`. The SSM document is called `StateManagerBootstrap.yml` (a YAML document).
2. Runs the `aws:runDocument` plugin to run the `StateManagerBootstrap.yml` document. No parameters are specified.
3. Runs the `aws:runDocument` plugin to run the `AWS-ConfigureDocker` pre-defined SSM document. The specified parameters install Docker on the instance.

```json
{
    "schemaVersion": "2.2",
    "description": "My composite document for bootstrapping software and installing Docker."
    "parameters": {
    },
    "mainSteps": [
        {
            "action": "aws:downloadContent",
            "name": "downloadContent",
            "inputs": {
                "sourceType": "GitHub",
                "sourceInfo": "{"owner":"TestUser1", "repository":"TestPublic", "path":"documents/bootstrap/StateManagerBootstrap.yml"}",
                "destinationPath": "bootstrap"
            }
    },
        {
            "action": "aws:runDocument",
            "name": "runDocument",
            "inputs": {
                "documentType": "LocalPath",
                "documentPath": "bootstrap",
                "documentParameters": "{}"
            }
        }
    ]
}`
```
For more information about creating an SSM document, see Creating Systems Manager Documents (p. 267). For more information about the plugins you can add to a custom SSM document, see SSM Document Plugin Reference (p. 284). If you simply want to run a document from a remote location (without creating a composite document), see Running Documents from Remote Locations (p. 281).

Running Documents from Remote Locations

You can run SSM documents from remote locations by using the AWS-RunDocument pre-defined SSM document. This document currently supports the following remote locations:

- GitHub repositories (public and private)
- Amazon S3
- Documents saved in Systems Manager

The following procedure describes how to run remote SSM documents by using the console. This procedure shows how to run the remote document by using Run Command, but you can also run remote documents by using State Manager or Automation.

Before You Begin

Before you run a remote document, you must complete the following tasks.

- Create an SSM document and save it in a remote location. For more information, see Creating Systems Manager Documents (p. 267)
- If you plan to run a remote document that is stored in a private GitHub repository, then you must create a Systems Manager SecureString parameter for your GitHub security access token. You can't access a remote document in a private GitHub repository by manually passing your token over SSH. The access token must be passed as a Systems Manager SecureString parameter. For more information about creating a SecureString parameter, see Creating Systems Manager Parameters (p. 337).

Topics

- Run a Remote Document (Console) (p. 281)

Run a Remote Document (Console)

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

To run a remote document (AWS Systems Manager)

2. In the navigation pane, choose Run Command.

For more information about creating an SSM document, see Creating Systems Manager Documents (p. 267). For more information about the plugins you can add to a custom SSM document, see SSM Document Plugin Reference (p. 284). If you simply want to run a document from a remote location (without creating a composite document), see Running Documents from Remote Locations (p. 281).
-or-

If the AWS Systems Manager home page opens, scroll down and choose Explore Run Command.

3. Choose Run command.


5. In the Targets section, identify the instances to which you want to download and run the script by specifying tags or selecting instances manually.

6. (Optional) In Rate control:

- In Concurrency, specify either a number or a percentage of instances on which to run the command at the same time.

  Note
  If you selected targets by choosing Amazon EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document at the same time by specifying a percentage.

- In Error threshold, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify 3 errors, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.

7. In the Source Type list, choose an option.

- If you choose GitHub, specify Source Info information in the following format:

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

  For example:

  ```json
  {"owner":"TestUser1", "repository": "SSMTestDocsRepo", "path": "SSMDocs/mySSMdoc.yml", "tokenInfo":{"ssm-secure:myAccessTokenParam}}
  ```

- If you choose S3, specify Source Info information in the following format:

  ```json
  {"path":"URL_to_document_in_S3"}
  ```

  For example:

  ```json
  {"path":"https://s3.amazonaws.com/aws-executecommand-test/scripts/ruby/mySSMdoc.json"
  ```

- If you choose SSMDocument, specify Source Info information in the following format:

  ```json
  {"name": "document_name"}
  ```

  For example:

  ```json
  {"name": "mySSMdoc"
  ```

8. In the Document Parameters field, type parameters for the remote SSM document. For example, if you run the AWS-RunPowerShell document, you could specify:

  ```json
  {"commands": ["date", "echo \"Hello World\""]}
  ```
If you run the AWS-ConfigureAWSPack document, you could specify:

```json
{
    "action":"Install",
    "name":"AWSPVDriver"
}
```

9. In **Other parameters**:
   - In the **Comment** box, type information about this command.
   - In **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

10. (Optional) In **Rate control**:
    - In **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.
      
      **Note**
      If you selected targets by choosing Amazon EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document at the same time by specifying a percentage.
    
    - In **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify 3 errors, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.

11. In the **Output options** section, if you want to save the command output to a file, select the **Write command output to an Amazon S3 bucket**. Type the bucket and prefix names in the boxes.

12. 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 Configuring Amazon SNS Notifications for Run Command (p. 169).

13. Choose **Run**.

### To run a remote document (Amazon EC2 Systems Manager)

1. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).
2. In the navigation pane, choose **Run Command**, and then choose **Run a command**.
3. In the **Document** list, choose **AWS-RunDocument**.
4. In the **Select Targets by** section, choose an option and select the instances where you want the document to run.
5. (Optional) In the **Execute on** field, specify a number of **Targets** that can run the document concurrently (for example, 10). Or, specify a percentage of the number of targets that can run the document concurrently (for example, 10%).

    **Note**
    If you selected targets by choosing EC2 tags, and you are not certain how many instances use the selected tags, then limit the number of instances that can run the document by specifying a percentage.

6. (Optional) In the **Stop after** field, specify the maximum number of errors allowed before the system stops sending the command to other instances. For example, if you specify 3, then Systems Manager stops sending the command when the 4th error is received. Instances still processing the command might also send errors.

7. In the **Source Type** list, choose an option.
• If you choose GitHub, specify Source information in the following format:

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

For example:

```json
{"owner":"TestUser1", "repository": "SSMTestDocsRepo", "path": "SSMDocs/mySSMdoc.yml", "tokenInfo":"{{ssm-secure:myAccessTokenParam}}" }
```

• If you choose S3, specify Source information in the following format:

```json
{"path": "URL_to_document_in_S3"}
```

For example:

```json
{"path": "https://s3.amazonaws.com/aws-executecommand-test/scripts/ruby/mySSMdoc.json"
```

• If you choose SSMDocument, specify Source information in the following format:

```json
{"name": "document_name"}
```

For example:

```json
{"name": "mySSMdoc"}
```

8. In the Document parameters field, type parameters for the remote SSM document. For example, if you run the AWS-RunPowerShell document, you could specify:

```json
{"commands": ["date", "echo \"Hello World\""]
```

If you run the AWS-ConfigureAWSPack document, you could specify:

```json
{
  "action": "Install",
  "name": "AWSPVDriver"
}
```

9. In the Comments field, type information about this command.

10. In the Advanced Options section, choose Write to S3 to store command output in an Amazon S3 bucket. Type the bucket and prefix names in the text boxes.

11. Choose Enable SNS notifications to receive notifications and status about the command execution. For more information about configuring SNS notifications for Run Command, see Configuring Amazon SNS Notifications for Run Command (p. 169).


**SSM Document Plugin Reference**

This reference describes the actions, or plugins, that you can specify in an AWS Systems Manager (SSM) document. This reference does not include information about AWS Systems Manager Automation.
document plugins. For information about Automation document plugins, see Systems Manager Automation Document Reference (p. 305).

Systems Manager determines the actions to perform on a managed instance by reading the contents of a Systems Manager document. Each document includes a code-execution section. Depending on the schema version of your document, this code-execution section can include one or more plugins or steps. For the purpose of this Help topic, plugins and steps are called plugins. This section includes information about each of the Systems Manager plugins. For more information about documents, including information about creating documents and the differences between schema versions, see AWS Systems Manager Documents (p. 258).

Note
Some of the plugins described here run only on either Windows instances or Linux instances. Platform dependencies are noted for each plugin.

Contents
- Top-level Elements (p. 285)
- aws:applications (p. 286)
- aws:cloudWatch (p. 287)
- aws:configureDocker (p. 293)
- aws:configurePackage (p. 294)
- aws:domainJoin (p. 294)
- aws:downloadContent (p. 295)
- aws:psModule (p. 297)
- aws:refreshAssociation (p. 298)
- aws:runDockerAction (p. 298)
- aws:runDocument (p. 300)
- aws:runPowerShellScript (p. 301)
- aws:runShellScript (p. 302)
- aws:softwareInventory (p. 302)
- aws:updateAgent (p. 303)
- aws:updateSSMAgent (p. 304)

Top-level Elements

The top-level elements are common for all Systems Manager documents. Top-level elements provide the structure of the Systems Manager document.

Properties

schemaVersion

The version of the schema.

Type: Version

Required: Yes

description

A description of the configuration.

Type: String

Required: No
parameters

parameters is a structure that contains one or more parameters to execute when processing the
document. You can specify parameters at runtime, in a document, or by using Systems Manager
Parameter Store. For more information, see AWS Systems Manager Parameter Store (p. 330).

Type: Structure

The parameters structure accepts the following fields and values:
- type: (Required) Allowed values include the following: String, StringList, Boolean,
  Integer, MapList, and StringMap
- 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) Allowed values for the parameter.
- allowedPattern: (Optional) The regular expression the parameter must match.
- displayType: (Optional) Used to display either a textfield or a textarea in the AWS 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.

runtimeConfig

(Schema version 1.2 only) The configuration for the instance as applied by one or more Systems
Manager plugins. Plugins are not guaranteed to run in sequence.

Type: Dictionary<string,PluginConfiguration>

Required: No

mainSteps

(Schema version 0.3, 2.0, and 2.2 only) The configuration for the instance as applied by one or more
Systems Manager plugins. Plugins are organized as actions within steps. Steps execute in sequential
order as listed in the document.

Type: Dictionary<string,PluginConfiguration>

Required: No

aws:applications

Install, repair, or uninstall applications on an EC2 instance. This plugin only runs on Microsoft Windows
operating systems. For more information, see AWS Systems Manager Documents (p. 258).

Syntax

```
"runtimeConfig":{
  "aws:applications":{
    "properties":{
      "id":"0.aws:applications",
      "action":"{{ action }}",
      "parameters":"{{ parameters }}",
    }
  }
}
```
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 using Amazon CloudWatch metrics. This plugin only runs on Microsoft Windows operating systems. For more information about configuring CloudWatch integration with Amazon EC2, see Sending Logs, Events, and Performance Counters to Amazon CloudWatch. For more information about documents, see AWS Systems Manager Documents (p. 258).

You can export and monitor the following data types:

ApplicationEventLog
Sends application event log data to CloudWatch Logs.

CustomLogs
Sends any text-based log file to CloudWatch Logs. The CloudWatch plugin creates a fingerprint for log files. The system then associates a data offset with each fingerprint. The plugin uploads files when there are changes, records the offset, and associates the offset with a fingerprint. This method
is used to avoid a situation where a user enables the plugin, associates the service with a directory that contains a large number of files, and the system uploads all of the files.

**Warning**

Be aware that if your application truncates or attempts to clean logs during polling, any logs specified for `LogDirectoryPath` can lose entries. If, for example, you want to limit log file size, create a new log file when that limit is reached, and then continue writing data to the new file.

**ETW**

Sends Event Tracing for Windows (ETW) data to CloudWatch Logs. Microsoft Windows Server 2003 is not supported.

**IIS**

Sends IIS log data to CloudWatch Logs.

**PerformanceCounter**

Sends Windows performance counters to CloudWatch. You can select different categories to upload to CloudWatch as metrics. For each performance counter that you want to upload, create a `PerformanceCounter` section with a unique ID (for example, "PerformanceCounter2", "PerformanceCounter3", and so on) and configure its properties.

**Note**

If the SSM Agent or the CloudWatch plugin is stopped, performance counter data is not logged in CloudWatch. This behavior is different than custom logs or Windows Event logs. Custom logs and Windows Event logs preserve performance counter data and upload it to CloudWatch after the 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 additional 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 additional 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**

```json
"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 cannot be used with SSM.
Type: String
Required: No

CategoryName
The performance counter category from Performance Monitor.
Type: String
Required: Yes

CounterName
The name of the performance counter from Performance Monitor.
Type: String
Required: Yes

CultureName
The locale where the timestamp is logged. If CultureName is blank, it defaults to the same locale currently used by your Windows instance.
Type: String
Valid values: For a list of supported values, see National Language Support (NLS) on the Microsoft website. Note that the div, div-MV, hu, and hu-HU values are not supported.
Required: No

DimensionName
A dimension for your Amazon CloudWatch metric. If you specify DimensionName, you must specify DimensionValue. These parameters provide another view when listing metrics. You can also use the same dimension for multiple metrics so that you can view all metrics belonging to a specific dimension.
Type: String
Required: No

DimensionValue
A dimension value for your Amazon CloudWatch metric.
Type: String
Required: No

Encoding
The file encoding to use (for example, UTF-8). Use the encoding name, not the display name.
Type: String
Valid values: For a list of supported values, see Encoding Class in the MSDN Library.
Required: Yes
Filter

The prefix of log names. Leave this parameter blank to monitor all files.

Type: String

Valid values: For a list of supported values, see the FileSystemWatcherFilter Property in the MSDN Library.

Required: No

Flows

Each data type to upload, along with the destination for the data (CloudWatch or CloudWatch Logs). For example, to send a performance counter defined under "Id": "PerformanceCounter" to the CloudWatch destination defined under "Id": "CloudWatch", enter "PerformanceCounter,CloudWatch". Similarly, to send the custom log, ETW log, and system log to the CloudWatch Logs destination defined under "Id": "ETW", enter "ETW,CloudWatchLogs". In addition, you can send the same performance counter or log file to more than one destination. For example, to send the application log to two different destinations that you defined under "Id": "CloudWatchLogs" and "Id": "CloudWatchLogs2", enter "ApplicationEventLog,(CloudWatchLogs, CloudWatchLogs2)".

Type: String

Valid values (source): ApplicationEventLog | CustomLogs | ETW | PerformanceCounter | SystemEventLog | SecurityEventLog

Valid values (destination): CloudWatch | CloudWatchLogs | CloudWatchLogs2 | CloudWatchLogs3

Required: Yes

FullName

The full name of the component.

Type: String

Required: Yes

Id

Identifies the data source or destination. This identifier must be unique within the configuration file.

Type: String

Required: Yes

InstanceName

The name of the performance counter instance. Do not use an asterisk (*) to indicate all instances because each performance counter component only supports one metric. You can, however use _Total.

Type: String

Required: Yes

Levels

The types of messages to send to Amazon CloudWatch.

Type: String

Valid values:
1 - Only error messages uploaded.
2 - Only warning messages uploaded.
4 - Only information messages uploaded.

Note that you can add values together to include more than one type of message. For example, 3 means that error messages (1) and warning messages (2) are included. A value of 7 means that error messages (1), warning messages (2), and informational messages (4) are included.

Required: Yes

Note
Windows Security Logs should set Levels to 7.

LineCount
The number of lines in the header to identify the log file. For example, IIS log files have virtually identical headers. You could enter 3, which would read the first three lines of the log file's header to identify it. In IIS log files, the third line is the date and time stamp, which is different between log files.

Type: Integer
Required: No

LogDirectoryPath
For CustomLogs, the path where logs are stored on your Amazon EC2 instance. For IIS logs, the folder where IIS logs are stored for an individual site (for example, C:\inetpub\logs\LogFiles\W3SVCn). For IIS logs, only W3C log format is supported. IIS, NCSA, and Custom formats are not supported.

Type: String
Required: Yes

LogGroup
The name for your log group. This name is displayed on the Log Groups screen in the CloudWatch console.

Type: String
Required: Yes

LogName
The name of the log file.
1. To find the name of the log, in Event Viewer, in the navigation pane, click Applications and Services Logs.
2. In the list of logs, right-click the log you want to upload (for example, Microsoft-Windows-Backup-Operational), and then click Create Custom View.
3. In the Create Custom View dialog box, click the XML tab. The LogName is in the <Select Path=> tag (for example, Microsoft-Windows-Backup). Copy this text into the LogName parameter.

Type: String
Valid values: Application | Security | System | Microsoft-Windows-WinINet/Analytic

Required: Yes

LogStream
The destination log stream. If you use {instance_id}, the default, the instance ID of this instance is used as the log stream name.
Type: String

Valid values: `{instance_id} | {hostname} | {ip_address} <log_stream_name>`

If you enter a log stream name that doesn't already exist, CloudWatch Logs automatically creates it for you. You can use a literal string or predefined variables `{instance_id}`, `{hostname}`, `{ip_address}`, or a combination of all three to define a log stream name.

The log stream name specified in this parameter appears on the Log Groups > Streams for `<YourLogStream>` screen in the CloudWatch console.

Required: Yes

MetricName

The CloudWatch metric that you want performance data to appear under.

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 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: us-east-1 | us-west-2 | eu-west-1 | eu-central-1 | ap-southeast-1 | ap-southeast-2 | ap-northeast-1

Required: Yes

SecretKey

Your secret access key. This property is required unless you launched your instance using an IAM role.

Type: String

Required: No

startType

Enable or disable CloudWatch on the instance.

Type: String

Valid values: Enabled | Disabled
**Required:** Yes

**TimestampFormat**

The timestamp format you want to use. For a list of supported values, see Custom Date and Time Format Strings in the MSDN Library.

Type: String

**Required:** Yes

**TimeZonKind**

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 runs only on Microsoft Windows operating systems. For more information, see AWS Systems Manager Documents (p. 258).

**Syntax**

```
"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 package. This plugin runs on Linux and Microsoft Windows operating systems. For more information, see [AWS Systems Manager Documents](p. 258).

**Syntax**

```
"mainSteps": [
  {
    "action": "aws:configurePackage",
    "name": "configurePackage",
    "inputs": {
      "name": "{{ name }}",
      "action": "{{ action }}",
      "version": "{{ version }}"
    }
  }
]
```

**Inputs**

**name**

The name of the AWS package to install or uninstall.

Type: String
Required: Yes

**action**

Install or uninstall a package.

Type: Enum
Valid values: Install | Uninstall
Required: Yes

**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 Amazon EC2 instance to a domain. This plugin only runs on Microsoft Windows operating systems. For more information, see [AWS Systems Manager Documents](p. 258).
**Syntax**

```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: Array  
Required: No  
Example: "dnsIpAddresses": ["198.51.100.1","198.51.100.2"]

**Examples**

For examples, see Joining a Windows Instance to an AWS Directory Service Domain in the Amazon EC2 User Guide for Windows Instances.

**aws:downloadContent**

(Schema version 2.0 or later) Download SSM documents and scripts from remote locations. This plugin is supported on Linux and Windows operating systems.
Syntax

```
"mainSteps": [
    {
      "action":"aws:downloadContent",
      "name":"downloadContent",
      "inputs":{
        "sourceType":"{{ sourceType }}",
        "sourceInfo":"{{ sourceInfo }}",
        "destinationPath":"{{ destinationPath }}"
      }
    }
]
```

Inputs

sourceType

The download source. Systems Manager currently supports the following source types for downloading scripts and SSM documents: GitHub, S3, and SSMDocument.

Type: String

Required: Yes

sourceInfo

The information required to retrieve the content from the required source.

Type: StringMap

Required: Yes

For sourceType GitHub, specify the following:

- owner: The repository owner.
- repository: The name of the repository.
- path: The path to the file or directory you want to download.
- getOptions: Extra options to retrieve content from a different branch or a different commit. This parameter uses the following format:
  - branch:branch_name
    The default is master.
  - commitID:commitID
    The default is head.
- tokenInfo: The Systems Manager parameter (a SecureString parameter) where you store your access token information.

Example syntax:

```
{
  "owner":"TestUser",
  "repository":"GitHubTest",
  "path": "scripts/python/test-script",
  "getOptions": "branch:master",
  "tokenInfo": "{{ssm-secure:secure-string-token}}"
}
```

For sourceType S3, specify the following:

- path: The URL to the file or directory you want to download from Amazon S3.
Example syntax:
{
  "path": "https://s3.amazonaws.com/aws-executecommand-test/powershell/helloPowershell.ps1"
}

**For sourceType SSMDocument, specify one of the following:**

- **name**: The name and version of the document in the following format: name:version. Version is optional.

Example syntax:
{
  "name": "Example-RunPowerShellScript:3"
}

- **name**: The ARN for the document in the following format:


Example syntax:
{
  "name":"arn:aws:ssm:us-east-1:i-1234567890:document/MySharedDoc"
}

**aws:psModule**

Install PowerShell modules on an EC2 instance. This plugin only runs on Microsoft Windows operating systems. For more information, see [AWS Systems Manager Documents](p. 258).

**Syntax**

```json
"runtimeConfig":{
  "aws:psModule":{
    "properties":[
      {
        "id":"0.aws:psModule",
        "runCommand":"{{ commands }}",
        "source": "{{ source }}",
        "sourceHash": "{{ sourceHash }}",
        "workingDirectory": "{{ workingDirectory }}",
        "timeoutSeconds": "{{ executionTimeout }}"
      }
    ]
  }
}
```

**Properties**

- **runCommand**

  The PowerShell command to run after the module is installed.

  Type: String or Array

  Required: No

- **source**

  The URL or local path on the instance to the application .zip file.

  Type: String
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 is considered to have failed.
Type: String
Required: No

workingDirectory
The path to the working directory on your instance.
Type: String
Required: No

aws:refreshAssociation
(Schema version 2.0 or later) Refresh (force apply) an association on demand. This action will change the system state based on what is defined in the selected association or all associations bound to the targets. This plugin runs on Linux and Microsoft Windows operating systems. For more information, see AWS Systems Manager Documents (p. 258).

Syntax
```
"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 operating systems. For more information, see AWS Systems Manager Documents (p. 258).

Syntax
```
"mainSteps": [
```
{
    "action": "aws:runDockerAction",
    "name": "RunDockerAction",
    "inputs": {
        "action": "{{ action }}",
        "container": "{{ container }}",
        "image": "{{ image }}",
        "memory": "{{ memory }}",
        "cpuShares": "{{ cpuShares }}",
        "volume": "{{ volume }}",
        "cmd": "{{ cmd }}",
        "env": "{{ env }}",
        "user": "{{ user }}",
        "publish": "{{ publish }}"
    }
}

**Inputs**

**action**

The type of action to perform.

Type: String

Required: Yes

**container**

The Docker container ID.

Type: String

Required: No

**image**

The Docker image name.

Type: String

Required: No

**cmd**

The container command.

Type: String

Required: No

**memory**

The container memory limit.

Type: String

Required: No

**cpuShares**

The container CPU shares (relative weight).

Type: String

Required: No
volume
The container volume mounts.
Type: StringList
Required: No

env
The container environment variables.
Type: String
Required: No

user
The container user name.
Type: String
Required: No

publish
The container published ports.
Type: String
Required: No

aws:runDocument
(Schema version 2.0 or later) Executes SSM documents stored in Systems Manager or on a local share. You can use this plugin with the aws:downloadContent (p. 295) 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 operating systems.

Syntax
```json
"mainSteps": [  
  {    
    "action":"aws:runDocument",    
    "name":"runDocument",    
    "inputs":{    
      "documentType":"{{ documentType }}",    
      "documentPath":"{{ documentPath }}",    
      "documentParameters":"{{ documentParameters }}"    
    }  
  }  
]  
```

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 only runs on Microsoft Windows operating systems. For more information, see AWS Systems Manager Documents (p. 258).

Syntax

```json
"runtimeConfig":{
  "aws:runPowerShellScript":{
    "properties": [
      {
        "id": "0.aws:runPowerShellScript",
        "runCommand": "{{ commands }}",
        "workingDirectory": "{{ workingDirectory }}",
        "timeoutSeconds": "{{ executionTimeout }}"
      }
    ]
  }
}
```

Properties

runCommand

Specify the command(s) to run or the path to an existing script on the instance.

Type: String or Array
Required: Yes
timeoutSeconds

The time in seconds for a command to be completed before it is considered to have failed.

Type: String
Required: No
workingDirectory

The path to the working directory on your instance.

Type: String
Required: No
aws:runShellScript

Run Linux shell scripts or specify the path to a script to run. This plugin only runs on Linux operating systems. For more information, see AWS Systems Manager Documents (p. 258).

Syntax

```
"runtimeConfig":{
   "aws:runShellScript":{
      "properties":[
         {
            "id":"0.aws:runShellScript",
            "runCommand":"{{ commands }}",
            "workingDirectory":"{{ workingDirectory }}",
            "timeoutSeconds":"{{ executionTimeout }}"
         }
      ]
   }
}
```

Properties

**runCommand**

Specify the command(s) to run or the path to an existing script on the instance.

Type: String or Array

Required: Yes

**timeoutSeconds**

The time in seconds for a command to be completed before it is considered to have failed.

Type: String

Required: No

**workingDirectory**

The path to the working directory on your instance.

Type: String

Required: No

aws:softwareInventory

(Schema version 2.0 or later) Gather an inventory of applications, AWS components, network configuration, Windows Updates, and custom inventory from an instance. This plugin runs on Linux and Microsoft Windows operating systems. For more information, see AWS Systems Manager Documents (p. 258).

Syntax

```
"mainSteps": [
   {
      "action": "aws:softwareInventory",
      "name": "collectSoftwareInventoryItems",
      "inputs": {
```
"applications": "{{ applications }}",
"awsComponents": "{{ awsComponents }}",
"networkConfig": "{{ networkConfig }}",
"windowsUpdates": "{{ windowsUpdates }}",
"customInventory": "{{ customInventory }}"
}

**Inputs**

**applications**

Collect data for installed applications.

Type: String

Required: No

**awsComponents**

Collect data for AWS components like amazon-ssm-agent.

Type: String

Required: No

**networkConfig**

Collect data for network configuration.

Type: String

Required: No

**windowsUpdates**

Collect data for all Windows updates.

Type: String

Required: No

**customInventory**

Collect data for 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 operating systems. For more information about the EC2Config service, see Configuring a Windows Instance Using the EC2Config Service. For more information about documents, see AWS Systems Manager Documents (p. 258).

**Syntax**

```
"runtimeConfig": {
    "aws:updateAgent": {
        "properties": {
```
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 operating systems. For more information, see Installing and Configuring SSM Agent (p. 13). For more information about documents, see AWS Systems Manager Documents (p. 258).

Syntax

```
"runtimeConfig": {
  "aws:updateSsmAgent": {
    "properties": [
      {
        "agentName": "amazon-ssm-agent",
        "source": "https://s3.Region.amazonaws.com/aws-ssm-Region/manifest.json",
        "allowDowngrade": "{{ allowDowngrade }}",
        "targetVersion": "{{ version }}"
      }
    ]
  }
}
```
Properties

**agentName**
amazon-ssm-agent. This is the name of the Systems Manager agent that processes requests and executes 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: No

**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 the SSM Agent to install. If not specified, the agent will be updated to the latest version.

Type: String
Required: No

Systems Manager Automation Document Reference

This reference describes the actions (or plugins) that you can specify in an AWS Systems Manager Automation document. For information about plugins for other types of SSM documents, see SSM Document Plugin Reference (p. 284).

Systems Manager Automation executes steps defined in Automation documents. Each step is associated with a particular action. The action determines the inputs, behavior, and outputs of the step. Steps are defined in the `mainSteps` section of your Automation document. Automation supports the following actions.

- **aws:approve**: 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.
- **aws:changeInstanceState**: Changes the state of an instance.
- **aws:copyImage**: Copies an AMI from any region into the current region. This action can also encrypt the new AMI.
• **aws:createImage**: Creates an AMI from a running instance.
• **aws:createStack**: Creates a new AWS CloudFormation stack from a template.
• **aws:createTags**: Creates new tags for Amazon EC2 instances or Systems Manager managed instances.
• **aws:deleteImage**: Deletes an AMI.
• **aws:deleteStack**: Deletes an AWS CloudFormation stack.
• **aws:executeAutomation**: Executes a secondary Automation workflow by calling a secondary Automation document.
• **aws:invokeLambdaFunction**: Enables you to run external worker functions in your automation workflow.
• **aws:runCommand**: Executes a remote command.
• **aws:runInstances**: Launches one or more instances.
• **aws:sleep**: Delays Automation execution for a specified amount of time.

You don't need to specify the outputs of an action or step. The outputs are predetermined by the action associated with the step. When you specify step inputs in your Automation documents, you can reference one or more outputs from an earlier step. For example, you can make the output of **aws:runInstances** available for a subsequent **aws:runCommand** action. You can also reference outputs from earlier steps in the Output section of the Automation document.

## Common Properties In All Actions

The following properties are common to all actions:

```
"mainSteps": [
    {
        "name": "name",
        "action": "action",
        "maxAttempts": value,
        "timeoutSeconds": value,
        "onFailure": "Abort",
        "inputs": {
            ...
        }
    },
    {
        "name": "name",
        "action": "action",
        "maxAttempts": value,
        "timeoutSeconds": value,
        "onFailure": "Abort",
        "inputs": {
            ...
        }
    }
]
```

### name

An identifier that must be unique across all step names in the document.

**Type:** String

**Required:** Yes

### action

The name of the action the step is to execute.
maxAttempts

The number of times the step should be retried in case of failure. If the value is greater than 1, the step is not considered to have failed until all retry attempts have failed. The default value is 1.

Type: Integer
Required: No

timeoutSeconds

The execution timeout value for the step. If the timeout is reached and the value of maxAttempts is greater than 1, then the step is not considered to have timed out until all retries have been attempted. There is no default value for this field.

Type: Integer
Required: No

onFailure

Indicates whether the workflow should continue on failure. The default is to abort on failure.

Type: String
Valid values: Abort | Continue
Required: No

inputs

The properties specific to the action.

Type: Map
Required: Yes

aws:approve

Temporarily pauses an Automation execution until designated principals either approve or reject the action. After the required number of approvals is reached, the Automation execution resumes. You can insert the approval step any place in the mainSteps section of your Automation document.

In the following example, the aws:approve action temporarily pauses the Automation workflow until one approver either accepts or rejects the workflow. Upon approval, the document executes a simple PowerShell command.

```json
{
    "description": "RunInstancesDemo1",
    "schemaVersion": "0.3",
    "assumeRole": "{{ assumeRole }}",
    "parameters": {
        "assumeRole": {
            "type": "String"
        },
        "message": {
            "type": "String"
        }
    }
}
```
You can approve or deny Automations that are waiting for approval in the Amazon EC2 console.

**Note**
The following procedure describes steps that you perform in the Amazon EC2 console. You can also perform these steps in the new AWS Systems Manager console. The steps in the new console will differ from the steps below.

**To approve or deny waiting Automations**

1. Open the Amazon EC2 console, expand Systems Manager Services in the navigation pane, and then choose Automations.
2. Choose an Automation with a status of Waiting, choose Actions, and then choose Approve/Deny this request.
3. Review the details of the Automation in the **Approve/Deny this request** page.

![Approve/Deny this request](image)

4. Choose either **Approve** or **Reject**, type an optional comment, and then choose **Submit**.

**Input**

```json
{
    "Message":"Please approve this step of the Automation.",
    "MinRequiredApprovals":3,
    "Approvers":[
        "IamUser1",
        "IamUser2",
        "arn:aws:iam::12345678901:user/IamUser3",
        "arn:aws:iam::12345678901:role/IamRole"
    ]
}
```

**NotificationArn**

The ARN of an Amazon SNS topic for Automation approvals. When you specify an `aws:approve` step in an Automation document, Automation sends a message to this topic letting principals know that they must either approve or reject an Automation step. The title of the Amazon SNS topic must be prefixed with “Automation”.

Type: String

Required: No

**Message**

The information you want to include in the SNS topic when the approval request is sent. The maximum message length is 4096 characters.
MinRequiredApprovals

The minimum number of approvals required to resume the Automation execution. If you don't specify a value, the system defaults to one. The value for this parameter must be a positive number. The value for this parameter can't exceed the number of approvers defined by the Approvers parameter.

Type: Integer
Required: No

Approvers

A list of AWS authenticated principals who are able to either approve or reject the action. The maximum number of approvers is 10. You can specify principals by using any of the following formats:
- An AWS Identity and Access Management (IAM) user name
- An IAM user ARN
- An IAM role ARN
- An IAM assume role user ARN

Type: StringList
Required: Yes

Output

ApprovalStatus

The approval status of the step. The status can be one of the following: Approved, Rejected, or Waiting. Waiting means that Automation is waiting for input from approvers.

Type: String

ApproverDecisions

A JSON map that includes the approval decision of each approver.

Type: MapList

aws:changeInstanceState

Changes or asserts the state of the instance.

This action can be used in assert mode (do not execute the API to change the state but verify the instance is in the desired state.) To use assert mode, set the CheckStateOnly parameter to true. This mode is useful when running the Sysprep command on Windows, which is an asynchronous command that can run in the background for a long time. You can ensure that the instance is stopped before you create an AMI.

Input

```json
{
    "name": "stopMyInstance",
    "action": "aws:changeInstanceState",
}
```
"maxAttempts": 3,
"timeoutSeconds": 3600,
"onFailure": "Abort",
"inputs": {
    "InstanceIds": ["i-1234567890abcdef0"],
    "CheckStateOnly": true,
    "DesiredState": "stopped"
}
}

InstanceId
The IDs of the instances.
Type: String
Required: Yes

CheckStateOnly
If false, sets the instance state to the desired state. If true, asserts the desired state using polling.
Type: Boolean
Required: No

DesiredState
The desired state.
Type: String
Valid values: running | stopped | terminated
Required: Yes

Force
If set, forces the instances to stop. The instances do not have an opportunity to flush file system caches or file system metadata. If you use this option, you must perform file system check and repair procedures. This option is not recommended for Windows instances.
Type: Boolean
Required: No

AdditionalInfo
Reserved.
Type: String
Required: No

Output
None

aws:copyImage
Copies an AMI from any region into the current region. This action can also encrypt the new AMI.

Input
This action supports most CopyImage parameters. For more information, see CopyImage.

The following example creates a copy of an AMI in the Seoul region (SourceImageID: ami-0fe10819, SourceRegion: ap-northeast-2). The new AMI is copied to the region where you initiated the Automation action. The copied AMI will be encrypted because the optional Encrypted flag is set to true.

```json
{
    "name": "createEncryptedCopy",
    "action": "aws:copyImage",
    "maxAttempts": 3,
    "onFailure": "Abort",
    "inputs": {
        "SourceImageId": "ami-0fe10819",
        "SourceRegion": "ap-northeast-2",
        "ImageName": "Encrypted Copy of LAMP base AMI in ap-northeast-2",
        "Encrypted": true
    }
}
```

SourceRegion

The region where the source AMI currently exists.
Type: String
Required: Yes

SourceImageId

The AMI ID to copy from the source region.
Type: String
Required: Yes

ImageName

The name for the new image.
Type: String
Required: Yes

ImageDescription

A description for the target image.
Type: String
Required: No

Encrypted

Encrypt the target AMI.
Type: Boolean
Required: No

KmsKeyId

The full Amazon Resource Name (ARN) of the AWS Key Management Service CMK to use when encrypting the snapshots of an image during a copy operation. For more information, see CopyImage.
Type: String
Required: No

ClientToken

A unique, case-sensitive identifier that you provide to ensure request idempotency. For more information, see CopyImage.

Type: String
Required: No

Output

ImageId
The ID of the copied image.

ImageState
The state of the copied image.

Valid values: available | pending | failed

**aws:createImage**

Creates a new AMI from a stopped instance.

**Important**

This action does not stop the instance implicitly. You must use the aws:changeInstanceState action to stop the instance. If this action is used on a running instance, the resultant AMI might be defective.

**Input**

This action supports most CreateImage parameters. For more information, see CreateImage.

```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 of the image.
Type: String  
Required: Yes

ImageDescription

A description of the image.
Type: String  
Required: No

NoReboot

A boolean literal.
Type: Boolean  
Required: No

BlockDeviceMappings

The block devices for the instance.
Type: Map  
Required: No

Output

ImageId

The ID of the newly created image.

ImageState

An execution script provided as a string literal value. If a literal value is entered, then it must be Base64-encoded.
Required: No

`aws:createStack`

Creates a new AWS CloudFormation stack from a template.

Input

```json
{
    "name": "makeStack",
    "action": "aws:createStack",
    "maxAttempts": 1,
    "onFailure": "Abort",
    "inputs": {
        "Capabilities": [
            "CAPABILITY_IAM"
        ],
        "StackName": "myStack",
        "TemplateURL": "http://s3.amazonaws.com/mybucket/myStackTemplate",
        "TimeoutInMinutes": 5
    }
}
```
Capabilities

A list of values that you specify before AWS CloudFormation can create certain stacks. Some stack templates include resources that can affect permissions in your AWS account. For example, creating new AWS Identity and Access Management (IAM) users can affect permissions in your account. For those stacks, you must explicitly acknowledge their capabilities by specifying this parameter.

The only valid values are `CAPABILITY_IAM` and `CAPABILITY_NAMED_IAM`. The following resources require you to specify this parameter.

- 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.

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.

For more information, see Acknowledging IAM Resources in AWS CloudFormation Templates.

Type: array of Strings

Valid Values: `CAPABILITY_IAM` | `CAPABILITY_NAMED_IAM`

Required: No

DisableRollback

Set to `true` to disable rollback of the stack if stack creation failed.

Conditional: You can specify either the DisableRollback parameter or the OnFailure parameter, but not both.

Default: false

Type: Boolean

Required: No

NotificationARNs

The Amazon SNS topic ARNs for publishing stack-related events. You can find SNS topic ARNs using the Amazon SNS console, https://console.aws.amazon.com/sns/v2/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 service:

  `AWS::service_name::resource_logical_ID`

If the list of resource types doesn't include a resource that you're creating, the stack creation fails. By default, AWS CloudFormation grants permissions to all resource types. IAM uses this parameter for AWS CloudFormation-specific condition keys in IAM policies. For more information, see Controlling Access with AWS Identity and Access Management.

Type: array of Strings

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

RoleARN

The Amazon Resource Name (ARN) of an IAM role that AWS CloudFormation assumes to create the stack. AWS CloudFormation uses the role's credentials to make calls on your behalf. AWS
CloudFormation always uses this role for all future operations on the stack. As long as users have permission to operate on the stack, AWS CloudFormation uses this role even if the users don't have permission to pass it. Ensure that the role grants the least amount of privileges.

If you don't specify a value, AWS CloudFormation uses the role that was previously associated with the stack. If no role is available, AWS CloudFormation uses a temporary session that is generated from your user credentials.

Type: String
Required: No

StackName
The name that is associated with the stack. The name must be unique in the region in which you are creating the stack.

Note
A stack name can contain only alphanumeric characters (case sensitive) and hyphens. It must start with an alphabetic character and cannot be longer than 128 characters.

Type: String
Required: Yes

StackPolicyBody
Structure containing the stack policy body. For more information, see Prevent Updates to Stack Resources.

Conditional: You can specify either the StackPolicyBody parameter or the StackPolicyURL parameter, but not both.

Type: String
Required: No

StackPolicyURL
Location of a file containing the stack policy. The URL must point to a policy located in an Amazon S3 bucket in the same region as the stack. The maximum file size allowed for the stack policy is 16 KB.

Conditional: You can specify either the StackPolicyBody parameter or the StackPolicyURL parameter, but not both.

Type: String
Required: No

Tags
Key-value pairs to associate with this stack. AWS CloudFormation also propagates these tags to the resources created in the stack. You can specify a maximum number of 10 tags.

Type: array of Tag objects
Required: No
TemplateBody

Structure containing the template body with a minimum length of 1 byte and a maximum length of 51,200 bytes. For more information, see Template Anatomy.

Conditional: You can specify either the TemplateBody parameter or the TemplateURL parameter, but not both.

Type: String

Length Constraints: Minimum length of 1.

Required: No

TemplateURL

Location of a file containing the template body. The URL must point to a template that is located in an Amazon S3 bucket. The maximum size allowed for the template is 460,800 bytes. For more information, see Template Anatomy.

Conditional: You can specify either the TemplateBody parameter or the TemplateURL parameter, but not both.

Type: String


Required: No

TimeoutInMinutes

The amount of time that can pass before the stack status becomes CREATE_FAILED. If DisableRollback is not set or is set to false, the stack will be rolled back.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

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 | REVIEW_IN_PROGRESS

Required: Yes
StackStatusReason

Success or failure message associated with the stack status.

Type: String

Required: No

For more information, see CreateStack.

Security Considerations

Before you can use the aws:createStack action, you must assign the following policy to the IAM Automation assume role. For more information about the assume role, see Task 1: Create a Service Role for Automation (p. 99).

```json
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": [ 
            "sqs:*",
            "cloudformation:CreateStack",
            "cloudformation:DescribeStacks"
         ],
         "Resource": "*"
      }
   ]
}
```

**aws:createTags**

Create new tags for Amazon EC2 instances or Systems Manager managed instances.

**Input**

This action supports most EC2 CreateTags and SSM AddTagsToResource parameters. For more information, see CreateTags and AddTagsToResource.

The following example shows how to tag an AMI and an instance as being production resources for a particular department.

```json
{
   "name": "createTags",
   "action": "aws:createTags",
   "maxAttempts": 3,
   "onFailure": "Abort",
   "inputs": { 
      "ResourceType": "EC2",
      "ResourceIds": [ 
         "ami-9a3768fa",
         "i-02951acd511a8169"
      ],
      "Tags": [ 
         { 
            "Key": "production",
            "Value": "" 
         },
         { 
            "Key": "department",
```
Resourcelds

The IDs of the resource(s) to be tagged. If resource type is not “EC2”, this field can contain only a single item.

Type: String List

Required: Yes

Tags

The tags to associate with the resource(s).

Type: List of Maps

Required: Yes

ResourceType

The type of resource(s) to be tagged. If not supplied, the default value of “EC2” is used.

Type: String

Required: No

Valid Values: EC2 | ManagedInstance | MaintenanceWindow | Parameter

Output

None

**aws:deleteImage**

Deletes the specified image and all related snapshots.

Input

This action supports only one parameter. For more information, see the documentation for DeregisterImage and DeleteSnapshot.

```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**

Deletes an AWS CloudFormation stack.

**Input**

```json
{
    "name":"deleteStack",
    "action":"aws:deleteStack",
    "maxAttempts":1,
    "onFailure":"Abort",
    "inputs":{
        "StackName": "{{stackName}}"
    }
}
```

**ClientRequestToken**

A unique identifier for this `DeleteStack` request. Specify this token if you plan to retry requests so that AWS CloudFormation knows that you're not attempting to delete a stack with the same name. You can retry `DeleteStack` requests to verify that AWS CloudFormation received them.

Type: String


Pattern: `[a-zA-Z][-a-zA-Z0-9]*`

Required: No

**RetainResources.member.N**

This input applies only to stacks that are in a `DELETE_FAILED` state. A list of logical resource IDs for the resources you want to retain. During deletion, AWS CloudFormation deletes the stack, but does not delete the retained resources.

Retaining resources is useful when you can't delete a resource, such as a non-empty Amazon S3 bucket, but you want to delete the stack.

Type: array of strings

Required: No

**RoleARN**

The Amazon Resource Name (ARN) of an IAM role that AWS CloudFormation assumes to create the stack. AWS CloudFormation uses the role's credentials to make calls on your behalf. AWS CloudFormation always uses this role for all future operations on the stack. As long as users have permission to operate on the stack, AWS CloudFormation uses this role even if the users don't have permission to pass it. Ensure that the role grants the least amount of privileges.

If you don't specify a value, AWS CloudFormation uses the role that was previously associated with the stack. If no role is available, AWS CloudFormation uses a temporary session that is generated from your user credentials.

Type: String
Required: No

StackName
The name or the unique stack ID that is associated with the stack.
Type: String
Required: Yes

Security Considerations
Before you can use the `aws:deleteStack` action, you must assign the following policy to the IAM Automation assume role. For more information about the assume role, see Task 1: Create a Service Role for Automation (p. 99).

```json
{
   "Version":"2012-10-17",
   "Statement":[
   {
      "Effect":"Allow",
      "Action":[
      "sqs:*",
      "cloudformation:DeleteStack",
      "cloudformation:DescribeStacks"
      ],
      "Resource":"*"
   }
   ]
}
```

**aws:executeAutomation**
Executes a secondary Automation workflow by calling a secondary Automation document. With this action, you can create Automation documents for your most common workflows, and reference those documents during an Automation execution. This action can simplify your Automation documents by removing the need to duplicate steps across similar documents.

The secondary Automation runs in the context of the user who initiated the primary Automation. This means that the secondary Automation uses the same IAM role or user account as the user who started the first Automation.

**Important**
If you specify parameters in a secondary Automation that use an assume role (a role that uses the `iam:passRole` policy), then the user or role that initiated the primary Automation must have permission to pass the assume role specified in the secondary Automation. For more information about setting up an assume role for Automation, see Method 2: Using IAM to Configure Roles for Automation (p. 98).

**Input**

```json
{
   "name":"Secondary_Automation_Workflow",
   "action":"aws:executeAutomation",
   "maxAttempts":3,
   "timeoutSeconds":3600,
   "onFailure":"Abort",
   "inputs":{
      "DocumentName":"secondaryWorkflow",
```
"RuntimeParameters":{
   "instanceIds":[
      "i-1234567890abcdef0"
   ]
}
}
}

DocumentName

The name of the secondary Automation document to execute during the step. The document must belong to the same AWS account as the primary Automation document.

Type: String
Required: Yes

DocumentVersion

The version of the secondary Automation document to execute. If not specified, Automation runs the default document version.

Type: String
Required: Yes

RuntimeParameters

Required parameters for the secondary document execution. The mapping uses the following format: "parameter1": ["value1"], "parameter2": ["value2"]

Type: Map
Required: No

Output

Output

The output generated by the secondary execution. You can reference the output by using the following format: Secondary_Automation_Step_Name.Output

Type: StringList

ExecutionId

The execution ID of the secondary execution.

Type: String

Status

The status of the secondary execution.

Type: String

aws:invokeLambdaFunction

Invokes the specified Lambda function.

Input

This action supports most invoke parameters for the Lambda service. For more information, see Invoke.
```json
{
    "name": "invokeMyLambdaFunction",
    "action": "aws:invokeLambdaFunction",
    "maxAttempts": 3,
    "timeoutSeconds": 120,
    "onFailure": "Abort",
    "inputs": {
        "FunctionName": "MyLambdaFunction"
    }
}
```

**FunctionName**

The name of the Lambda function. This function must exist.

Type: String

Required: Yes

**Qualifier**

The function version or alias name.

Type: String

Required: No

**InvocationType**

The invocation type. The default is RequestResponse.

Type: String

Valid values: Event | RequestResponse | DryRun

Required: No

**LogType**

If Tail, the invocation type must be RequestResponse. AWS Lambda returns the last 4 KB of log data produced by your Lambda function, base64-encoded.

Type: String

Valid values: None | Tail

Required: No

**ClientContext**

The client-specific information.

Required: No

**Payload**

The JSON input for your Lambda function.

Required: No

**Output**

**StatusCode**

The function execution status code.
FunctionError

Indicates whether an error occurred while executing the Lambda function. If an error occurred, this field will show either Handled or Unhandled. Handled errors are reported by the function. Unhandled errors are detected and reported by AWS Lambda.

LogResult

The base64-encoded logs for the Lambda function invocation. Logs are present only if the invocation type is RequestResponse, and the logs were requested.

Payload

The JSON representation of the object returned by the Lambda function. Payload is present only if the invocation type is RequestResponse.

aws:runCommand

Runs the specified commands.

Note

Automation only supports output of one Run Command action. A document can include multiple Run Command actions and plugins, but output is supported for only one action and plugin at a time.

Input

This action supports most send command parameters. For more information, see SendCommand.

{  
  "name": "installPowerShellModule",  
  "action": "aws:runCommand",  
  "inputs": {  
    "DocumentName": "AWS-InstallPowerShellModule",  
    "InstanceIds": ["i-1234567890abcdef0"],  
    "Parameters": {  
      "source": "https://my-s3-url.com/MyModule.zip ",  
      "sourceHash": "ASDFWER12321WRW"  
    }  
  }  
}

DocumentName

The name of the Run Command document.

Type: String

Required: Yes

InstanceIds

The IDs of the instances.

Type: String

Required: Yes

Parameters

The required and optional parameters specified in the document.

Type: Map
Required: No

Comment
User-defined information about the command.
Type: String
Required: No

DocumentHash
The hash for the document.
Type: String
Required: No

DocumentHashType
The type of the hash.
Type: String
Valid values: Sha256 | Sha1
Required: No

NotificationConfig
The configurations for sending notifications.
Required: No

OutputS3BucketName
The name of the S3 bucket for command execution responses.
Type: String
Required: No

OutputS3KeyPrefix
The prefix.
Type: String
Required: No

ServiceRoleArn
The ARN of the IAM role.
Type: String
Required: No

TimeoutSeconds
The run-command timeout value, in seconds.
Type: Integer
Required: No

**aws:runInstances**
Launches a new instance.
Input

The action supports most API parameters. For more information, see the RunInstances API documentation.

```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"
  }
}
```

**ImageId**

The ID of the Amazon Machine Image (AMI).

Required: Yes

**InstanceType**

The instance type.

Required: No

**MinInstanceCount**

The minimum number of instances to be launched.

Required: No

**MaxInstanceCount**

The maximum number of instances to be launched.

Required: No

**AdditionalInfo**

Reserved.

Required: No

**BlockDeviceMappings**

The block devices for the instance.

Required: No

**ClientToken**

The identifier to ensure idempotency of the request.

Required: No

**DisableApiTermination**

Enables or disables instance API termination.

Required: No
EbsOptimized
  Enables or disabled EBS optimization.
  Required: No

IamInstanceProfileArn
  The ARN of the IAM instance profile for the instance.
  Required: No

IamInstanceProfileName
  The name of the IAM instance profile for the instance.
  Required: No

InstanceInitiatedShutdownBehavior
  Indicates whether the instance stops or terminates on system shutdown.
  Required: No

KernelId
  The ID of the kernel.
  Required: No

KeyName
  The name of the key pair.
  Required: No

Monitoring
  Enables or disables detailed monitoring.
  Required: No

NetworkInterfaces
  The network interfaces.
  Required: No

Placement
  The placement for the instance.
  Required: No

PrivateIpAddress
  The primary IPv4 address.
  Required: No

RamdiskId
  The ID of the RAM disk.
  Required: No

SecurityGroupIds
  The IDs of the security groups for the instance.
  Required: No
SecurityGroups

The names of the security groups for the instance.
Required: No

SubnetId

The subnet ID.
Required: No

UserData

An execution script provided as a string literal value. If a literal value is entered, then it must be Base64-encoded.
Required: No

Output

InstanceIds

The IDs of the instances.

/aws:sleep

Delays Automation execution for a specified amount of time. This action uses the International Organization for Standardization (ISO) 8601 date and time format. For more information about this date and time format, see ISO 8601.

Input

You can delay execution for a specified duration.

```
{
  "name": "sleep",
  "action": "aws:sleep",
  "inputs": {
    "Duration": "PT10M"
  }
}
```

You can also delay execution until a specified date and time. If the specified date and time has passed, the action proceeds immediately.

```
{
  "name": "sleep",
  "action": "aws:sleep",
  "inputs": {
    "Timestamp": "2020-01-01T01:00:00Z"
  }
}
```

Note

Automation currently supports a maximum delay of 604800 seconds (7 days).

Duration

An ISO 8601 duration. You can't specify a negative duration.
AWS Systems Manager Parameter Store

AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, 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 that you specified when you created the parameter. Highly scalable, available, and durable, Parameter Store is backed by the AWS Cloud. Parameter Store is offered at no additional charge.

Parameter Store offers the following benefits and features.

- Use a secure, scalable, hosted secrets management service (No servers to manage).
- Improve your security posture by separating your data from your code.
- Store configuration data and secure strings in hierarchies and track versions.
- Control and audit access at granular levels.
- Configure change notifications and trigger automated actions.
- Tag parameters individually, and then secure access from different levels, including operational, parameter, EC2 tag, or path levels.
- Reference parameters across AWS services such as Amazon EC2, Amazon Elastic Container Service, AWS Lambda, AWS CloudFormation, AWS CodeBuild, AWS CodeDeploy, and other Systems Manager capabilities.
- Configure integration with AWS KMS, Amazon SNS, Amazon CloudWatch, and AWS CloudTrail for encryption, notification, monitoring, and audit capabilities.

Getting Started with Systems Manager Parameters

To get started with Systems Manager Parameters, complete the following tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learn how to use different types of Systems Manager parameters.</td>
<td>About Systems Manager Parameters (p. 331)</td>
</tr>
<tr>
<td>Learn how to organize, create, and tag parameters.</td>
<td>Working with Systems Manager Parameters (p. 334)</td>
</tr>
</tbody>
</table>
About Systems Manager Parameters

You can reference Systems Manager parameters in your scripts, commands, and configuration and automation workflows. Parameters work with Systems Manager capabilities such as Run Command, State Manager, and Automation. You can also reference parameters in other AWS services such as Amazon Elastic Container Service and AWS Lambda.

With Systems Manager capabilities, you can reference Systems Manager parameters in your AWS CLI or AWS Tools for Windows PowerShell commands or scripts. You can also reference parameters in SSM documents. For more information about SSM documents, see AWS Systems Manager Documents (p. 258).

The following is an example of a Systems Manager parameter in an AWS CLI command for Run Command. Systems Manager Parameters are always prefixed with `ssm:`.

```
aws ssm send-command --instance-ids i-1a2b3c4d5e6f7g8 --document-name AWS-RunPowerShellScript --parameter '{"commands":["echo {{ssm:parameter name}}"]}'
```

You can also reference Systems Manager parameters in the `Parameters` section of an SSM document, as shown in the following example.

```json
{
  "schemaVersion":"2.0",
  "description":"Sample version 2.0 document v2",
  "parameters":{
    "commands" : {
      "type": "StringList",
      "default": ["{{ssm:parameter name}}"]
    },
  },
  "mainSteps":[
    {"action":"aws:runShellScript",
      "name":"runShellScript",
      "inputs":{
        "runCommand": "{{commands}}"
      }
    }
  ]
}
```
Note
The runtimeConfig section of SSM documents use similar syntax for local parameters. A local parameter is not 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 }}"
      }
    ]
  }
}

SSM documents currently don't support references to Secure String parameters. This means that to use Secure String 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:

**AWS CLI**

```
#value=aws ssm get-parameters --names the parameter name --with-decryption

aws ssm send-command -name AWS-JoinDomain -parameters password=$value -instance-id the instance ID
```

**Tools for Windows PowerShell**

```
$secure = (Get-SSMParameterValue -Names the parameter name -WithDecryption $True).Parameters[0].Value | ConvertTo-SecureString -AsPlainText -Force

$cred = New-Object System.Management.Automation.PSCredential -argumentlist user name, $secure
```

**Using Secure String Parameters**

A Secure String 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 clear text, such as passwords or license keys, then create those parameters using the Secure String data type. We recommend using Secure String parameters for the following scenarios.

- You want to use data/parameters across AWS services without exposing the values as clear text in commands, functions, agent logs, or AWS CloudTrail logs.
- You want to control who has access to sensitive data.
- You want to be able to audit when sensitive data is accessed (AWS CloudTrail).
- You want AWS-level encryption for your sensitive data and you want to bring your own encryption keys to manage access.

If you choose the Secure String data type when you create your parameter, then AWS KMS encrypts the parameter value. For more information, see [How AWS Systems Manager Parameter Store Uses AWS KMS](https://docs.aws.amazon.com/systems-manager/user-guide/using-parameter-store-aws-kms.html) in the [AWS Key Management Service Developer Guide](https://docs.aws.amazon.com/kms/latest/developerguide/).
Important
Only the value of the secure string parameter is encrypted. The name of the parameter, description, and other properties are not encrypted. For this reason, consider creating a naming system that avoids the word "password" in parameter names.

Create a Secure String Parameter Using the Default KMS CMK

Systems Manager includes a default AWS KMS customer master key (CMK). You can view this key by executing the following command from the AWS CLI:

```
aws kms describe-key --key-id alias/aws/ssm
```

If you create a Secure String parameter using the default KMS CMK, then you don’t have to provide a value for the `--key-id` parameter. The following CLI example shows the command to create a new Secure String parameter in Parameter Store without the `--key-id` parameter:

```
aws ssm put-parameter --name a_name --value "a value" --type SecureString
```

Create a Secure String Parameter Using a Custom KMS CMK

If you want to use a custom KMS CMK instead of the default CMK assigned to your account, then you must specify the custom KMS CMK by using the `--key-id` parameter. The parameter supports the following AWS KMS parameter formats.

- Key ARN example:
  
  `arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-123456789012`

- Alias ARN example:
  
  `arn:aws:kms:us-east-1:123456789012:alias/MyAliasName`

- Globally Unique Key ID example:
  
  `12345678-1234-1234-1234-123456789012`

- Alias Name example:
  
  `alias/MyAliasName`

You can create a custom AWS KMS CMK from the AWS CLI by using the following commands:

```
aws kms create-key
```

Use the following command to create a Secure String parameter using the key you just created.

```
aws ssm put-parameter --name a_name --value "a value" --type SecureString --key-id arn:aws:kms:us-east-1:123456789012:key/1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e
```

Note
You can manually create a parameter with an encrypted value. In this case, because the value is already encrypted, you don’t have to choose the Secure String data type. If you do choose Secure String, your parameter will be doubly encrypted.

By default, all Secure String values are displayed as cipher text. To decrypt a Secure String value, a user must have permission to call the KMS Decrypt API action. For information about configuring KMS...
access control, see Authentication and Access Control for AWS KMS in the AWS Key Management Service Developer Guide.

Using Secure String Parameters With Other AWS Services

You can also use Secure String parameters with other AWS services. In the following example, the Lambda function retrieves a Secure String parameter by using the GetParameters API.

```python
from __future__ import print_function
import json
import boto3

ssm = boto3.client('ssm', 'us-east-1')

def get_parameters():
    response = ssm.get_parameters(
        Names=['LambdaSecureString'], WithDecryption=True
    )
    for parameter in response['Parameters']:
        return parameter['Value']

def lambda_handler(event, context):
    value = get_parameters()
    print("value1 = " + value)
    return value  # Echo back the first key value
```

Related topics

For an example of how to create and use a Secure String parameter, see Walkthrough: Create a Secure String Parameter and Join an Instance to a Domain (PowerShell) (p. 355). For more information about using Systems Manager parameters with other AWS services, see the following blogpost.

- Managing Secrets for Amazon ECS Applications Using Parameter Store and IAM Roles for Tasks
- Use Parameter Store to Securely Access Secrets and Config Data in AWS CodeDeploy
- Interesting Articles on Amazon EC2 Systems Manager Parameter Store

Working with Systems Manager Parameters

This section describes how to organize, create, and tag parameters, and create different versions of parameters.

Topics

- Organizing Parameters into Hierarchies (p. 334)
- Creating Systems Manager Parameters (p. 337)
- Tagging Systems Manager Parameters (p. 342)
- Working with Parameter Versions (p. 344)

Organizing Parameters into Hierarchies

Managing dozens or hundreds of parameters as a flat list is time consuming and prone to errors. It can also be difficult to identify the correct parameter for a task. This means you might accidentally use the wrong parameter, or you might create multiple parameters that use the same configuration data.

You can use parameter hierarchies to help you organize and manage parameters. A hierarchy is a parameter name that includes a path that you define by using forward slashes. Here is an example that uses three hierarchy levels in the name to identify the following:
You can create a hierarchy with a maximum of five 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/git-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/db_password

You could then create a unique password for your production environment, as shown in the following example:

/prod/MyApp/database/db_password

You are not required to specify a parameter hierarchy. You can create parameters at level one. These are called root parameters. For backward compatibility, all parameters created in Parameter Store before hierarchies were released are root parameters. The system treats both of the following parameters as root parameters.

/parameter-name

parameter-name
For an example of how to work with parameter hierarchies, see Walkthrough: Manage Parameters Using Hierarchies (AWS CLI) (p. 357).

Querying Parameters in a Hierarchy

Another benefit of using hierarchies is the ability to query for all parameters within a hierarchy by using the GetParametersByPath API action. For example, if you execute the following command from the AWS CLI, the system returns all parameters in the IIS level.

```bash
aws ssm get-parameters-by-path --path /Dev/Web/IIS
```

To view decrypted SecureString parameters in a hierarchy, you specify the path and the `--with-decryption` parameter, as shown in the following example.

```bash
aws ssm get-parameters-by-path --path /Prod/ERP/SAP --with-decryption
```

Restricting IAM Permissions Using Hierarchies

Using hierarchies and AWS Identity and Access Management (IAM) policies for Parameter Store API actions, you can provide or restrict access to all parameters in one level of a hierarchy. The following example policy allows all Parameter Store operations on all parameters for the AWS account 123456789012 in the us-east-1 Region. The user can't create parameters because the PutParameter action is explicitly denied. This policy also forbids the user from calling the GetParametersByPath action.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ssm:*"],
        },
        {
            "Effect": "Deny",
            "Action": ["ssm:GetParametersByPath"],
            "Condition": {
                "StringEquals": {
                    "ssm:Recursive": ["true"]
                }
            },
            "Resource": "arn:aws:ssm:us-east-1:123456789012:parameter/Prod/ERP/Oracle/"
        },
        {
            "Effect": "Deny",
            "Action": ["ssm:PutParameter"],
            "Condition": {
                "StringEquals": {
                    "ssm:Overwrite": ["false"]
                }
            },
            "Resource": "arn:aws:ssm:us-east-1:123456789012:parameter/"
        }
    
```
Creating Systems Manager Parameters

You can create parameters by using the AWS CLI, AWS Tools for Windows PowerShell, the Amazon EC2 console, or the AWS Systems Manager console.

Topics

• About Creating Systems Manager Parameters (p. 337)
• Create a Systems Manager Parameter (AWS CLI) (p. 338)
• Create a Systems Manager Parameter (Tools for Windows PowerShell) (p. 340)
• Create a Systems Manager Parameter (Console) (p. 341)

About Creating Systems Manager Parameters

When you create a parameter, you specify the following information:

• **Name** (Required) Specify a name to identify your parameter. Be aware of the following requirements and restrictions for Systems Manager parameter names:
  • Parameter names are case sensitive.
  • A parameter name must be unique within your AWS account. For example, Systems Manager treats the following as separate parameters:
    • /CMH/TestParam1
    • /TestParam1
  The following examples are also unique:
    • /CMH/TestParam1/Logpath1
    • /CMH/TestParam1
  The following examples are not unique:
    • /TestParam1
    • TestParam1
  A parameter name *can’t* be prefixed with "aws" or "ssm" (case-insensitive). For example, awsTestParameter, SSM-testparameter, or /aws/testparam1 will fail with an exception.
  • Parameter names can only include the following symbols and letters:
    a-zA-Z0-9_-./
  • A parameter name can’t include spaces.
  • 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 valid:
    /Level-1/Level-2/Level-3/Level-4/Level-5/parameter-name
    /Level-1/parameter-name
  Attempting to create the following parameter would fail with a HierarchyLevelLimitExceededException exception:
    /Level-1/Level-2/Level-3/Level-4/Level-5/Level-6/parameter-name
• **Data Type** (Required) Specify a data type to define how the system uses a parameter. Parameter Store currently supports the following data types: String, StringList, and SecureString.
**Note**
Items in a StringList must be separated by a comma (,). You can’t use other punctuation or special character to escape items in the list. If you have a parameter value that requires a comma, then use the String data type.

- **Description** (Optional, but recommended): Type a description to help you identify parameters and their intended use.
- **Value** (Required) Your parameter value.
- **Key ID**: Key ID applies only to parameters that use the SecureString data type. Key ID can either be the default AWS Key Management Service (AWS KMS) key automatically assigned to your AWS account or a custom key. Note the following:
  - To use your default AWS KMS key, choose the SecureString data type, and do not specify the Key ID when you create the parameter. The system automatically populates Key ID with your default KMS key.
  - To use a custom KMS key, choose the SecureString data type with the Key ID parameter.

After you create a parameter, you can specify it in your SSM documents, commands, or scripts using the following syntax (no space between brackets):

```bash
{{ssm:parameter_name}} or {{ ssm:parameter_name }}
```

**Note**
The name of a Systems Manager parameter can’t be prefixed with "ssm" or "aws," but when you specify the parameter in an SSM document or a command, the syntax includes "ssm", as shown in the following examples.

Valid: `{{ssm:addUsers}}`
Invalid: `{{ssm:ssmAddUsers}}`

**Create a Systems Manager Parameter (AWS CLI)**

You can use the AWS CLI to create a parameter that uses the String, StringList, or SecureString data type.

For more information about using the AWS CLI to create parameters, see Walkthrough: Create and Use a Parameter in a Command (AWS CLI) (p. 352).

**Note**
Parameters are only available in the Region where they were created.

**Topics**
- Create a String or StringList Parameter (AWS CLI) (p. 338)
- Create a SecureString Parameter (AWS CLI) (p. 339)

**Create a String or StringList Parameter (AWS CLI)**

1. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM).

```bash
aws configure
```

The system prompts you to specify the following.

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
```
2. Execute the following command to create a parameter.

```
aws ssm put-parameter --name "a_name" --value "a value, or a comma-separated list of values" --type String or StringList
```

If successful, the command has no output.

Here is an example that uses the StringList data type.

```
aws ssm put-parameter --name /IAD/ERP/Oracle/addUsers --value "Milana,Mariana,Mark,Miguel" --type StringList
```

**Note**

Items in a StringList must be separated by a comma (,). You can’t use other punctuation or special character to escape items in the list. If you have a parameter value that requires a comma, then use the String data type.

3. Execute the following command to verify the details of the parameter.

```
aws ssm get-parameters --name "the name you specified"
```

Here is an example that uses the name specified in the earlier example.

```
aws ssm get-parameters --name "/IAD/ERP/Oracle/addUsers"
```

### Create a SecureString Parameter (AWS CLI)

Before you create a SecureString parameter, read about the requirements for this type of parameter. For more information, see Using Secure String Parameters (p. 332).

1. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon Elastic Compute Cloud (Amazon EC2), or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

```
aws configure
```

The system prompts you to specify the following.

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

2. Execute the following command to create a parameter.

```
aws ssm put-parameter --name "a_name" --value "a value" --type SecureString --key-id "a KMS CMK ID, a KMS CMK ARN, an alias name, or an alias ARN"
```

**Note**

To use the default AWS KMS CMK assigned to your account, remove the key-id parameter from the command.
Here is an example that uses an obfuscated name (elixir3131) for a password and a custom AWS KMS key.

```bash
aws ssm put-parameter --name /Finance/Payroll/elixir3131 --value "P@$SwW)rd" --type SecureString --key-id arn:aws:kms:us-east-1:123456789012:key/1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e
```

3. Execute the following command to verify the details of the parameter.

```bash
aws ssm get-parameters --name "the name you specified" --with-decryption
```

**Note**
If you don't specify the `with-decryption` parameter, or if you specify the `no-with-decryption` parameter, the command returns an encrypted GUID.

---

### Create a Systems Manager Parameter (Tools for Windows PowerShell)

You can use Tools for Windows PowerShell to create a parameter that uses the String, StringList, or SecureString data type.

**Note**
Parameters are only available in the Region where they were created.

**Topics**
- Create a String or StringList parameter (Tools for Windows PowerShell) (p. 340)
- Create a SecureString parameter (Tools for Windows PowerShell) (p. 341)

#### Create a String or StringList parameter (Tools for Windows PowerShell)

1. Open AWS Tools for Windows PowerShell and execute the following command to specify your credentials. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

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

2. Execute the following command to set the Region for your PowerShell session. The example uses the us-east-2 Region.

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

3. Execute the following command to create a parameter.

```powershell
Write-SSMParameter -Name "a name" -Value "a value, or a comma-separated list of values" -Type "String or StringList"
```

If successful, the command has no output.

**Note**
Items in a StringList must be separated by a comma (,). You can't use other punctuation or special character to escape items in the list. If you have a parameter value that requires a comma, then use the String data type.

Here is an example that uses a String data type.
4. Execute the following command to verify the details of the parameter.

```
(Get-SSMParameterValue -Name "the name you specified").Parameters
```

Create a SecureString parameter (Tools for Windows PowerShell)

Before you create a SecureString parameter, read about the requirements for this type of parameter. For more information, see Using Secure String Parameters (p. 332).

1. Open AWS Tools for Windows PowerShell and execute the following command to specify your credentials. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

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

2. Execute the following command to set the Region for your PowerShell session. The example uses the us-east-2 region.

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

3. Execute the following command to create a parameter.

```
Write-SSMParameter -Name "a name" -Value "a value" -Type "SecureString" -KeyId "a KMS CMK ID, a KMS CMK ARN, an alias name, or an alias ARN"
```

If successful, the command has no output.

**Note**
To use the default AWS KMS CMK assigned to your account, remove the -KeyId parameter from the command.

Here is an example that uses an obfuscated name (elixir3131) for a password and the user's default KMS CMK.

```
Write-SSMParameter -Name "/Finance/Payroll/elixir3131" -Value "P@sSwW)rd" -Type "SecureString"
```

4. Execute the following command to verify the details of the parameter.

```
(Get-SSMParameterValue -Name "the name you specified" –WithDecryption $true).Parameters
```

Create a Systems Manager Parameter (Console)

You can use the Amazon EC2 console or AWS Systems Manager console to create a Systems Manager parameter.

**Note**
Parameters are only available in the Region where they were created.

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:
To create a parameter (AWS Systems Manager)

2. In the navigation pane, choose Parameter Store.
   -or-
   If the AWS Systems Manager home page opens, scroll down and choose Explore Parameter Store.
3. Choose Create parameter.
4. For Name, type a hierarchy and a parameter name. For example, type /Test/helloWorld.
5. In the Description box, type a description that identifies this parameter as a test parameter.
6. For Type, choose String, StringList, or SecureString.
   **Note**
   If you choose SecureString, the KMS Key ID field appears. If you don’t provide a KMS CMK ID, a KMS CMK ARN, an alias name, or an alias ARN, then the system uses alias/aws/ssm (Default), which is the default KMS CMK for Systems Manager. If you don’t want to use this key, then you can choose a custom key. For more information, see Using Secure String Parameters (p. 332).
7. In the Value box, type a value. For example, type MyFirstParameter. If you chose Secure String, the value is masked as you type.
8. Choose Create parameter.
9. 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.

To create a parameter (Amazon EC2 console)

1. Open the Amazon EC2 console, expand Systems Manager Shared Resources in the navigation pane, and then choose Parameter Store.
2. Choose Create Parameter.
3. For Name, type a hierarchy and a parameter name. For example, type /Test/helloWorld.
4. In the Description box, type a description that identifies this parameter as a test parameter.
5. For Type, choose String, String List, or Secure String.
   **Note**
   If you choose SecureString, the KMS Key ID field appears. If you don’t provide a KMS CMK ID, a KMS CMK ARN, an alias name, or an alias ARN, then the system uses alias/aws/ssm (Default), which is the default KMS CMK for Systems Manager. If you don’t want to use this key, then you can choose a custom key. For more information, see Using Secure String Parameters (p. 332).
6. In the Value box, type a value. For example, type MyFirstParameter. If you chose Secure String, the value is masked as you type.
7. Choose Create Parameter. After the system creates the parameter, choose Close.
8. In the parameters list, choose the parameter you just created. Verify the details on the Description tab. If you created a SecureString parameter, choose Show to view the unencrypted value.

Tagging Systems Manager Parameters

You can use the Systems Manager console, the AWS CLI, the AWS Tools for Windows, or the AddTagsToResource API to add tags to Systems Manager resources, including documents, managed instances, Maintenance Windows, Parameter Store parameters, and patch baselines.
Tags are used to organize parameters. For example, you can tag parameters for specific environments, departments, or users and groups. After you tag a parameter, you can restrict access to it by creating an IAM policy that specifies the tags that the user can access. For more information about restricting access to parameters by using tags, see Controlling Access to Parameters Using Tags (p. 349).

For information about the Regions where Systems Manager is available, see regions.

Topics
- Tag a Parameter (Console) (p. 343)
- Tag a Parameter (AWS CLI) (p. 343)
- Tag a Parameter (AWS Tools for Windows) (p. 344)

Tag a Parameter (Console)
2. In the left navigation, choose Parameter Store.
3. Choose the name of a parameter you have already created, and then choose the Tags tab.
4. In the first box, enter a key for the tag, such as Environment.
5. In the second box, enter a value for the tag, such as Test.
6. Choose Save.

Tag a Parameter (AWS CLI)
1. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2 or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

```
aws configure
```

The system prompts you to specify the following.

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

2. Execute the following command to list parameters that you can tag.

```
aws ssm describe-parameters
```

Note the name of a parameter that you want to tag.

3. Execute the following command to tag a parameter.

```
aws ssm add-tags-to-resource --resource-type "Parameter" --resource-id "the parameter name" --tags "Key=a key, for example Environment,Value=a value, for example TEST"
```

If successful, the command has no output.

4. Execute the following command to verify the parameter tags.

```
aws ssm list-tags-for-resource --resource-type "Parameter" --resource-id "the parameter name"
```
Tag a Parameter (AWS Tools for Windows)

1. Open AWS Tools for Windows PowerShell and execute the following command to specify your credentials. You must either have administrator privileges in Amazon EC2 or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

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

2. Execute the following command to set the Region for your PowerShell session. The example uses the us-east-2 Region. Systems Manager is currently available in the following regions.

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

3. Execute the following command to list parameters that you can tag.

   ```powershell
   Get-SSMParameterList
   ```

4. Execute the following commands to tag a parameter.

   ```powershell
   $tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
   $tag1.Key = "Environment"
   $tag1.Value = "TEST"
   Add-SSMResourceTag -ResourceType "Parameter" -ResourceId "the parameter name" -Tag $tag1
   ```

   If successful, the command has no output.

5. Execute the following command to verify the parameter tags.

   ```powershell
   Get-SSMResourceTag -ResourceType "Parameter" -ResourceId "the parameter name"
   ```

Working with Parameter Versions

When you initially create a parameter, Parameter Store assigns version 1 to that parameter. When you edit a parameter, Parameter Store 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 details, including the values, of all versions. You can also use parameter versions to see how many times a parameter changed over a period of time.

You can reference specific parameter versions, including older versions, in commands, API calls, and SSM documents by using the following format: {{ssm:parameter_name:version}}. Here is an example for a parameter named RunCommand specified in an SSM document:

```json
{
   "schemaVersion": "1.2",
   "description": "Run a shell script or specify the commands to run.",
   "parameters": {
      "commands": {
         "type": "StringList",
         "description": "(Required) Specify a shell script or a command to run.",
         "minItems": 1,
         "displayType": "textarea",
         "default": "{{ssm:RunCommand:2}}"
      }
   }
}
```
The following procedures show you how to edit a parameter and then verify that a new version was created.

**Topics**
- Create a New Parameter Version (Console) (p. 345)
- Create a New Parameter Version (AWS CLI) (p. 346)
- Create a New Parameter Version (Windows PowerShell) (p. 347)

**Create a New Parameter Version (Console)**

You can use the Amazon EC2 console or AWS Systems Manager console to create a new version of a parameter.

Depending on the service you are using, AWS Systems Manager or Amazon EC2 Systems Manager, use one of the following procedures:

**To create a new parameter version (AWS Systems Manager)**

2. In the navigation pane, choose **Parameter Store**.
   - **or-**
   - If the AWS Systems Manager home page opens, scroll down and choose **Explore Parameter Store**.
3. Choose the name of a parameter that you created earlier. For information about creating a new parameter, see **Creating Systems Manager Parameters** (p. 337).

   **Note**
   Parameters are only available in the Region where they were created. If you don’t see a parameter that you want to update, then verify your Region.

4. Choose **Edit**.
5. In the **Value** box, type a new value, and then choose **Save changes**.
6. In the parameters list, choose the name of the parameter you just updated, and then view the **History** tab. On the **Overview** tab, verify that the version number incremented by 1, and verify the new value.
To create a new parameter version (Amazon EC2 console)

1. Open the Amazon EC2 console, expand Systems Manager Shared Resources in the navigation pane, and then choose Parameter Store.
2. Choose a parameter that you created earlier. For information about creating a new parameter, see Creating Systems Manager Parameters (p. 337).
   
   **Note**
   Parameters are only available in the Region where they were created. If you don’t see a parameter that you want to update, then verify your Region.
3. Choose Actions, Edit parameter.
4. In the Value box, type a new value, and then choose Save parameter.
5. In the parameters list, choose the parameter you just updated, and then choose the History tab. Verify that the version number incremented by 1, and verify the new value.

Create a New Parameter Version (AWS CLI)

Use the following procedure to create a new version of a parameter by using the AWS CLI.

1. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2 or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

   ```bash
   aws configure
   ```

   The system prompts you to specify the following:

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

2. Execute the following command to list parameters that you can update.

   **Note**
   Parameters are only available in the Region where they were created. If you don’t see a parameter that you want to update, then verify your Region.

   ```bash
   aws ssm describe-parameters
   ```

   Note the name of a parameter that you want to update.

3. Execute the following command to update the parameter value.

   ```bash
   aws ssm put-parameter --name "the_parameter_name" --type the_parameter_type --value "the_new_value" --overwrite
   ```

4. Execute the following command to view all versions of the parameter.

   ```bash
   aws ssm get-parameter-history --name "the_parameter_name"
   ```

5. Execute the following command to retrieve information about a parameter by version number.

   ```bash
   aws ssm get-parameters --names "the_parameter_name:the_version_number"
   ```

   Here is an example.
Create a New Parameter Version (Windows PowerShell)

Use the following procedure to create a new version of a parameter by using the AWS Tools for Windows PowerShell.

1. Open AWS Tools for Windows PowerShell and execute the following command to specify your credentials. You must either have administrator privileges in Amazon EC2 or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager Prerequisites (p. 4).

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

2. Execute the following command to set the Region for your PowerShell session. The example uses the us-east-2 Region. Systems Manager is currently available in the following Regions.

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

3. Execute the following command to list parameters that you can update.

   ```powershell
   Get-SSMParameterList
   ```

   Note
   Parameters are only available in the Region where they were created. If you don't see a parameter that you want to update, then verify your Region.

4. Execute the following command to update the parameter value.

   ```powershell
   Write-SSMParameter -Name "the_parameter_name" -Value "the_new_value" -Type "the_parameter_type" -Overwrite $true
   ```

5. Execute the following command to view all versions of the parameter.

   ```powershell
   Get-SSMParameterHistory -Name "the_parameter_name"
   ```

6. Execute the following command to retrieve information about a parameter by version number.

   ```powershell
   (Get-SSMParameterValue -Names "the_parameter_name").Parameters | fl
   ```

Setting Up Systems Manager Parameters

To set up Systems Manager Parameters, you configure AWS Identity and Access Management (IAM) roles so that Systems Manager has permission to perform the actions you specify for the service. This section includes information about how to manually configure these roles using the IAM console. This section also include information about how to create Amazon CloudWatch Events Events to receive notifications about Systems Manager Parameters actions.

Contents

- Controlling Access to Systems Manager Parameters (p. 348)
- Setting Up Notifications and Events for Systems Manager Parameters (p. 351)
Controlling Access to Systems Manager Parameters

You control access to Systems Manager Parameters by using AWS Identity and Access Management (IAM). More specifically, you create IAM policies that restrict access to the following API operations:

- DeleteParameter
- DeleteParameters (to delete parameters by using the Amazon EC2 console)
- DescribeParameters
- GetParameter
- GetParameters
- PutParameter

We recommend that you control access to Systems Manager parameters by creating restrictive IAM policies. For example, the following policy allows you to call the DescribeParameters and GetParameters API operations for a specific resource. This means that you can get information about and use all parameters that begin with prod-•.

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

For trusted administrators, you could provide full access to all Systems Manager parameter API operations by using a policy like the following example. This policy gives the user full access to all production parameters that begin with dbserver-prod-•.

```
{
    "Version": "2012-10-17",
    "Effect": "Allow",
    "Action": [
        "ssm:DescribeParameters",
        "ssm:PutParameter",
        "ssm:GetParameters",
        "ssm:DeleteParameter"
    ],
    "Resource": [
        "arn:aws:ssm:region:account id:parameter/dbserver-prod-*"
    ]
}
```

Topics
- Allowing Only Specific Parameters to Run on Instances (p. 349)
- Controlling Access to Parameters Using Tags (p. 349)
Allowing Only Specific Parameters to Run on Instances

You can also control access so that instances can only run specific parameters. The following example enables instances to get a parameter value only for parameters that begin with "prod-" If the parameter is a secure string, then the instance decrypts the string using AWS KMS.

**Note**

If you choose the Secure String data type when you create your parameter, then AWS KMS encrypts the parameter value. For more information about AWS KMS, see [AWS Key Management Service Developer Guide](https://docs.aws.amazon.com/kms/latest/developerguide/). Each AWS account is assigned a default AWS KMS key. You can view your key by executing the following command from the AWS CLI:

```
aws kms describe-key --key-id alias/aws/ssm
```

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ssm:GetParameters"
         ],
         "Resource": [
            "arn:aws:ssm:region:account-id:parameter/prod-*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "kms:Decrypt"
         ],
         "Resource": [
            "arn:aws:kms:region:account-id:key/CMK"
         ]
      }
   ]
}
```

**Note**

Instance policies, like in the previous 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 [Configuring Access to Systems Manager (p. 8)](https://docs.aws.amazon.com/systemsmanager/latest/userguide/smsErotate.html).

Controlling Access to Parameters Using Tags

After you tag a parameter, you can restrict access to it by creating an IAM policy that specifies the tags the user can access. When a user attempts to use a parameter, the system checks the IAM policy and the tags specified for the parameter. If the user does not have access to the tags assigned to the parameter, the user receives an access denied error. Use the following procedure to create an IAM policy that restricts access to parameters by using tags.

**Before You Begin**

Create and tag parameters. For more information, see [Setting Up Systems Manager Parameters (p. 347)](https://docs.aws.amazon.com/systemsmanager/latest/userguide/sms-basics.html).

**To restrict a user's access to parameters by using tags**

2. In the navigation pane, choose **Policies**, and then choose **Create policy**.
3. Choose the **JSON** tab.

4. Copy the following sample policy and paste it into the text field, replacing the sample text. Replace `tag_key` and `tag_value` with the key-value pair for your tag.

```json
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":[
                "ssm:GetParameters"
            ],
            "Resource": "*",
            "Condition":{
                "StringLike":{
                    "ssm:resourceTag/tag_key": ["tag_value"]
                }
            }
        }
    ]
}
```

You can specify multiple keys in the policy by using the following **Condition** format. Specifying multiple keys creates an **AND** relationship for the keys.

```json
"Condition":{
    "StringLike":{
        "ssm:resourceTag/tag_key1": ["tag_value1"],
        "ssm:resourceTag/tag_key2": ["tag_value2"]
    }
}
```

You can specify multiple values in the policy by using the following **Condition** format. **ForAnyValue** establishes an **OR** relationship for the values. You can also specify **ForAllValues** to establish an **AND** relationship.

```json
"Condition":{
    "ForAnyValue: StringLike":{
        "ssm:resourceTag/tag_key1": ["tag_value1", "tag_value2"]
    }
}
```

5. Choose **Review policy**.

6. In the **Name** field, specify a name that identifies this as a user policy for tagged parameters.

7. Enter a description.

8. Verify details of the policy in the **Summary** section.

9. Choose **Create policy**.

10. Assign the policy to IAM users or groups. For more information, see [Changing Permissions for an IAM User](#) and [Attaching a Policy to an IAM Group](#).
After you attach the policy to the IAM user or group account, if a user tries to use a parameter and the user's policy does not allow the user to access a tag for the parameter (call the GetParameters API), the system returns an error. The error is similar to the following:

User: user_name is not authorized to perform: ssm:GetParameters on resource: parameter ARN with the following command.

If a parameter has multiple tags, the user will still receive the access denied error if the user does not have permission to access any one of those tags.

Setting Up Notifications and Events for Systems Manager Parameters

You can use Amazon CloudWatch Events and Amazon SNS to notify you about changes to Systems Manager Parameters. You can be notified when a parameter is accessed, updated, or deleted. You can also be notified when a parameter has not been updated or accessed for a period of time.

You can also use CloudWatch to perform an action on a target for specific parameter events. This means, for example, that you can execute an AWS Lambda function to recreate a parameter when it is deleted. You can also set up a notification to trigger a Lambda function when your database password is updated. The Lambda function can force your database connections to reset or reconnect with the new password.

Before You Begin

Create an Amazon SNS topic. For more information, see Getting Started with Amazon SNS in the Amazon Simple Notification Service Developer Guide.

To configure CloudWatch Events for Systems Manager Parameters

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose Events, and then choose Create rule.
3. Under Event Source, verify that Event Pattern is selected.
4. In the Service Name field, choose EC2 Simple Systems Manager (SSM)
5. In the Event Type field, choose Parameter Store.
6. Choose the detail types and statuses for which you want to receive notifications, and then choose Add targets.
7. In the Targets list, choose a target type. For example, choose Lambda function or choose SNS topic. For information about the different types of targets, see the corresponding AWS Help documentation.
8. Scroll down on the page, and then choose Configure details.
9. Specify the rule details, and then choose Create rule.

Systems Manager Parameter Store Walkthroughs

The following walkthroughs show you how to create, store, and execute parameters with Parameter Store in a test environment. These walkthroughs show you how to use Parameter Store with other Systems Manager capabilities. You can also use Parameter Store with other AWS services. For more information, see Using Secure String Parameters With Other AWS Services (p. 334).

Contents

- Walkthrough: Create and Use a Parameter in a Command (Console) (p. 352)
- Walkthrough: Create and Use a Parameter in a Command (AWS CLI) (p. 352)
Walkthrough: Create a Secure String Parameter and Join an Instance to a Domain (PowerShell) (p. 355)

Walkthrough: Manage Parameters Using Hierarchies (AWS CLI) (p. 357)

Walkthrough: Create and Use a Parameter in a Command (Console)

The following procedure walks you through the process of creating a parameter in Parameter Store and then executing a Run Command command that uses this parameter.

To create a parameter using Parameter Store

   -or-
   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   **Note**
   If you are using the Amazon EC2 console, some field names and locations may differ slightly.

2. In the navigation pane, choose Parameter Store.
3. Choose Create parameter.
4. In the Name box, type a hierarchy and a name. For example, type /Test/helloWorld.
   For more information about parameter hierarchies, see Organizing Parameters into Hierarchies (p. 334).
5. In the Description field, type a description that identifies this parameter as a test parameter.
6. For Type, choose String.
7. In the Value field, type a string. For example, type My1stParameter.
8. Choose Create parameter.
9. In the navigation pane, choose Run Command.
10. Choose Run command.
11. In the Command document list, choose AWS-RunPowershellScript (Windows) or AWS-RunShellScript (Linux).
12. Under Target instances, choose an instance you created earlier.
13. In the Commands field, type echo {{ssm:parameter name}}, for example, echo {{ssm:/Test/helloWorld}}.
15. Scroll to the bottom of the Command ID page, select the radio button next to an instance ID, and then choose View output.

Walkthrough: Create and Use a Parameter in a Command (AWS CLI)

The following procedure walks you through the process of creating and storing a parameter using the AWS CLI.

To create a String parameter using Parameter Store

1. **Download** the AWS CLI to your local machine.
2. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM).

```bash
aws configure
```

The system prompts you to specify the following.

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. Execute the following command to create a parameter that uses the String data type. The `--name` parameter uses a hierarchy. For more information about hierarchies, see Organizing Parameters into Hierarchies (p. 334).

```bash
aws ssm put-parameter --name "a_name" --value "a_value" --type String
```

Here is an example that uses a parameter hierarchy in the name. For more information about parameter hierarchies, see Organizing Parameters into Hierarchies (p. 334).

```bash
aws ssm put-parameter --name "/Test/IAD/helloWorld" --value "My1stParameter" --type String
```

The command has no output.

4. Execute the following command to view the parameter metadata.

```bash
aws ssm describe-parameters --filters "Key=Name,Values=/Test/IAD/helloWorld"
```

**Note**

Name must be capitalized.

The system returns information like the following.

```json
{
 "Parameters": [
  {
   "LastModifiedUser": "arn:aws:iam::123456789:user/User's name",
   "LastModifiedDate": 1494529763.156,
   "Type": "String",
   "Name": "helloworld"
  }
 ]
}
```

5. Execute the following command to change the parameter value.

```bash
aws ssm put-parameter --name "/Test/IAD/helloWorld" --value "good day sunshine" --type String --overwrite
```

The command has no output.

6. Execute the following command to view the latest parameter value.

```bash
aws ssm get-parameters --names "/Test/IAD/helloWorld"
```
The system returns information like the following.

```json
{
  "InvalidParameters": [],
  "Parameters": [
    {
      "Type": "String",
      "Name": "/Test/IAD/helloWorld",
      "Value": "good day sunshine"
    }
  ]
}
```

7. Execute the following command to view the parameter value history.

```bash
aws ssm get-parameter-history --name "/Test/IAD/helloWorld"
```

8. Execute the following command to use this parameter in a Run Command command.

```bash
aws ssm send-command --document-name "AWS-RunShellScript" --parameters "commands=["echo {{ssm:/Test/IAD/helloWorld}}"]" --targets "Key=instance-ids,Values=the ID of an instance configured for Systems Manager"
```

Use the following procedure to create a Secure String parameter. For more information about Secure String parameters, see Using Secure String Parameters (p. 332).

**To create a Secure String parameter using the AWS CLI**

1. Execute one of the following commands to create a parameter that uses the Secure String data type.

   **Create a Secure String parameter that uses your default KMS key**
   ```bash
   aws ssm put-parameter --name "a_name" --value "a value, for example P@ssW%rd#1" --type "SecureString"
   ```

   **Create a Secure String parameter that uses a custom AWS KMS key**
   ```bash
   aws ssm put-parameter --name "a_name" --value "a value" --type "SecureString" --key-id "your AWS user account ID/the custom AWS KMS key"
   ```

   Here is an example that uses a custom AWS KMS key.
   ```bash
   aws ssm put-parameter --name "db-password" --value "P@ssW%rd#1" --type "SecureString" --key-id "arn:aws:kms:us-east-1:123456789012:key/1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e"
   ```

   **Important**
   Only the value of the secure string parameter is encrypted. The name of the parameter, description, and other properties are not encrypted. For this reason, consider creating a naming system that avoids the word "password" in parameter names.

2. Execute the following command to view the parameter metadata.

   ```bash
   aws ssm describe-parameters --filters "Key=Name,Values=the name that you specified"
   ```

3. Execute the following command to change the parameter value.
aws ssm put-parameter --name "the name that you specified" --value "new value" --type "SecureString" --overwrite

Updating a Secure String parameter that uses your default KMS key

aws ssm put-parameter --name "the name that you specified" --value "new value" --type "SecureString" --key-id "the AWS KMS key ID" --overwrite

Updating a Secure String parameter that uses a custom KMS key

aws ssm put-parameter --name "the name that you specified" --value "new value" --type "SecureString" --key-id "your AWS user account alias/the custom KMS key" --overwrite

4. Execute the following command to view the latest parameter value.

aws ssm get-parameters --names "the name that you specified" --with-decryption

5. Execute the following command to view the parameter value history.

aws ssm get-parameter-history --name "the name that you specified"

Important
Only the value of the secure string parameter is encrypted. The name of the parameter, description, and other properties are not encrypted. For this reason, consider creating a naming system that avoids the word "password" in parameter names.

Walkthrough: Create a Secure String Parameter and Join an Instance to a Domain (PowerShell)

This walkthrough shows you how to join a Windows instance to a domain using Systems Manager Secure String parameters and Run Command. The walkthrough uses typical domain parameters, such as the DNS address, the domain name, and a domain user name. These values are passed as unencrypted string values. The domain password is encrypted using a AWS KMS master key and passed as a Secure String.

To create a Secure String Parameter and Join an Instance to a Domain

1. Enter parameters into the system using AWS Tools for Windows PowerShell.

Write-SSMParameter -Name DNS-IP -Value a DNS IP address -Type String
Write-SSMParameter -Name domainName -Value the domain name -Type String
Write-SSMParameter -Name domainJoinUserName -Value a user name -Type String
Write-SSMParameter -Name domainJoinPassword -Value a password -Type SecureString

Important
Only the value of the secure string parameter is encrypted. The name of the parameter, description, and other properties are not encrypted. For this reason, consider creating a naming system that avoids the word "password" in parameter names.

2. Attach the AmazonEC2RoleforSSM managed policy to the IAM role permissions for your instance. For information, see Managed Policies and Inline Policies.

3. Edit the IAM role attached to the instance and add the following policy. This policy gives the instance permissions to call the kms:Decrypt API.
4. Copy and paste the following json sample into a simple text editor and save the file as JoinInstanceToDomain.json in the following location: c:\temp\JoinInstanceToDomain.json.

```json
{
  "schemaVersion":"2.0",
  "description":"Run a PowerShell script to securely domain-join a Windows instance",
  "mainSteps":[
    {
      "action":"aws:runPowerShellScript",
      "name":"runPowerShellWithSecureString",
      "inputs":{
        "runCommand":
        "$ipdns = (Get-SSMParameterValue -Name dns).Parameters[0].Value
        ",
        "$domain = (Get-SSMParameterValue -Name domainName).Parameters[0].Value
        
        ",
        "$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")"
        
        
        "Set-DnsClientServerAddress \"Ethernet 2\" -ServerAddresses $ipdns"
        
        
        "Add-Computer -DomainName $domain -Credential $credential"
        
        "Restart-Computer -force"
      }
    }
  ]
}
```

5. Execute the following command in AWS Tools for Windows PowerShell to create a new SSM document.

```powershell
$json = Get-Content C:\temp\JoinInstanceToDomain | Out-String
New-SSMDocument -Name JoinInstanceToDomain -Content $json -DocumentType Command
```

6. Execute the following command in AWS Tools for Windows PowerShell to join the instance to the domain.

```powershell
Send-SSMCommand -InstanceId Instance-ID -DocumentName JoinInstanceToDomain
```
Walkthrough: Manage Parameters Using Hierarchies (AWS CLI)

This walkthrough shows you how to work with parameters and parameter hierarchies by using the AWS CLI. For more information about parameter hierarchies, see Organizing Parameters into Hierarchies (p. 334).

To manage parameters using hierarchies

1. Download the AWS CLI to your local machine.
2. Open the AWS CLI and run the following command to specify your credentials and a Region. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in AWS Identity and Access Management (IAM).

```bash
aws configure
```

The system prompts you to specify the following.

- 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. Execute the following command to create a parameter that uses the allowedPattern parameter and the String data type. The allowed pattern in this example means the value for the parameter must be between 1 and 4 digits long.

```bash
aws ssm put-parameter --name "/MyService/Test/MaxConnections" --value 100 --allowed-pattern \d{1,4} --type String
```

The command has no output.

4. Execute the following command to attempt to overwrite the parameter you just created with a new value.

```bash
aws ssm put-parameter --name "/MyService/Test/MaxConnections" --value 10,000 --type String --overwrite
```

The system throws the following error because the new value does not meet the requirements of the allowed pattern you specified in the previous step.

An error occurred (ParameterPatternMismatchException) when calling the PutParameter operation: Parameter value, cannot be validated against allowedPattern: \d{1,4}

5. Execute the following command to create a Secure String parameter that uses your default AWS KMS key. The allowed pattern in this example means the user can specify any character, and the value must be between 8 and 20 characters.

```bash
aws ssm put-parameter --name "/MyService/Test/DBpassword" --value "p#sW*r33" --allowed-pattern \.{8,20} --type SecureString
```

6. Execute the following commands to create more parameters that use the hierarchy structure from the previous step.

```bash
aws ssm put-parameter --name "/MyService/Test/DBname" --value "SQLDevDb" --type String
```
aws ssm put-parameter --name "/MyService/Test/user" --value "SA" --type String

aws ssm put-parameter --name "/MyService/Test/userType" --value "SQLuser" --type String

7. Execute the following command to get the value of two parameters.

aws ssm get-parameters --names "/MyService/Test/user" "/MyService/Test/userType"

8. Execute the following command to query for all parameters within a single level.

aws ssm describe-parameters --filters Key=Name,Values="/MyService/Test"

9. Execute the following command to delete two parameters

aws ssm delete-parameters --name "/IADRegion/Dev/user" "/IADRegion/Dev/userType"
Monitoring Instances with AWS Systems Manager

SSM Agent writes information about executions, scheduled actions, errors, and health statuses to log files on each instance. Manually connecting to an instance to view log files and troubleshoot an issue with SSM Agent is time-consuming. For more efficient instance monitoring, you can configure the either the SSM Agent itself or the CloudWatch Agent to send this log data to Amazon CloudWatch Logs. Using CloudWatch Logs, you can monitor log data in real-time, search and filter log data by creating one or more metric filters, and archive and retrieve historical data when you need it. For more information about CloudWatch Logs, see the Amazon CloudWatch Logs User Guide.

Configuring an agent to send log data to Amazon CloudWatch Logs provides the following benefits:

- Centralized log file storage for all of your SSM Agent log files.
- Quicker access to files to investigate errors.
- Indefinite log file retention (configurable).
- Logs can be maintained and accessed regardless of the status of the instance.
- Access to other CloudWatch features such as metrics and alarms.

Topics

- Send Logs to CloudWatch Logs (SSM Agent) (p. 359)
- Send Logs to CloudWatch Logs (CloudWatch Agent) (p. 360)

Send Logs to CloudWatch Logs (SSM Agent)

AWS Systems Manager Agent is Amazon software that runs on your Amazon EC2 instances and your hybrid instances that are configured for Systems Manager (hybrid instances). 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 Installing and Configuring SSM Agent (p. 13).

In addition, following the steps below, you can configure SSM Agent to send log data to Amazon CloudWatch Logs.

Before You Begin

Create a log group in Amazon CloudWatch Logs. For more information, see Create a Log Group in CloudWatch Logs in the Amazon CloudWatch Logs User Guide.

To configure SSM Agent to send logs to CloudWatch

1. Log into an instance and locate the following file:
   - On Windows: %PROGRAMFILES%\Amazon\SSM\seelog.xml.template
   - On Linux: /etc/amazon/ssm/seelog.xml.template
2. Change the file name from seelog.xml.template to seelog.xml.
3. Open the seelog.xml file in a text editor, and locate the following section:

```xml
<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 the following `custom name` element after the closing `</filter>` tag, as shown in the following example.

```xml
<seelog minlevel="info" critmsgcount="500" maxinterval="100000000"
  mininterval="2000000" type="adaptive">
  <exceptions>
    <exception minlevel="error" filepattern="test*"/>
  </exceptions>
  <outputs formatid="fmtinfo">
    <console formatid="fmtinfo"/>
    <rollingfile type="size" maxrolls="5" maxsize="30000000"
      filename="{{LOCALAPPDATA}}\Amazon\SSM\Logs\amazon-ssm-agent.log"/>
    <filter formatid="fmterror" levels="error,critical">
      <rollingfile type="size" maxrolls="5" maxsize="10000000"
        filename="{{LOCALAPPDATA}}\Amazon\SSM\Logs\errors.log"/>
    </filter>
    <custom name="cloudwatch_receiver" formatid="fmtdebug" data-log-group="Your CloudWatch Log Group Name"/>
  </outputs>
```

5. Save your changes.


7. Choose Logs, and then choose your log group. The log stream for SSM Agent log file data is organized by instance ID.

---

**Send Logs to CloudWatch Logs (CloudWatch Agent)**

You can configure and use the Amazon CloudWatch Agent to collect metrics and logs from your instances instead of using the SSM Agent for these tasks. The CloudWatch Agent enables you to gather more metrics on Amazon EC2 instances than are available using the SSM Agent. In addition, you can gather metrics from on-premises servers using the CloudWatch Agent.

For complete information about using the CloudWatch Agent, see Collect Metrics from Amazon Elastic Compute Cloud Instances and On-Premises Servers with the CloudWatch Agent in the Amazon CloudWatch User Guide.

**Migrating to the CloudWatch Agent**

On 64-bit versions of Windows only, Systems Manager supports migrating from the SSM Agent to the CloudWatch Agent for collecting metrics and logs. You can also store agent configuration settings in the Systems Manager Parameter Store for use with the CloudWatch Agent.

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.
Migrate Windows Server Instance Log Collection to the CloudWatch Agent

If you are currently using the SSM Agent on supported Windows Server instances to send SSM Agent log files to Amazon CloudWatch Logs, you can use Systems Manager to migrate from the SSM Agent to the CloudWatch Agent as your log collection tool, as well as migrate your configuration settings.

The CloudWatch Agent is not supported on 32-bit versions of Windows Server.

For 64-bit Amazon EC2 Windows instances, you can perform the migration to the CloudWatch Agent automatically or manually. For on-premises instances, the process must be performed manually.

**Note**
During the migration process, the data sent to CloudWatch may be interrupted or duplicated. Your metrics and log data will be recorded accurately again in CloudWatch after the migration is completed.

We recommend testing the migration on a limited number of instances before migrating an entire fleet to the CloudWatch Agent. After migration, if you prefer log collection with the 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 the SSM Agent specifies multiple Regions.
- The existing configuration for the SSM Agent specifies multiple sets of access/secret key credentials.

In these cases, it will be necessary to disable log collection in the SSM Agent and install the CloudWatch Agent without a migration process. For more information, see the following topics:

- Install the CloudWatch Agent on an Amazon EC2 Instance
- Install the CloudWatch Agent on an On-Premises Server

**Before You Begin**
Before you begin a migration to the CloudWatch Agent for log collection, ensure that the instances on which you will perform the migration meet these requirements:

- The OS is a 64-bit version of Windows Server.
- SSM Agent 2.2.93.0 or later is installed on the instance.
- The SSM Agent is configured for monitoring on the instance.

**Topics**
- Automatically Migrate to the CloudWatch Agent (p. 362)
- Manually Migrate to the CloudWatch Agent (p. 362)
Automatically Migrate to the CloudWatch Agent

For Amazon EC2 Windows instances only, you can use the AWS Systems Manager console, the Amazon EC2 console, or the AWS CLI to automatically migrate to the CloudWatch Agent as you log collection tool.

After the migration succeeds, check your results in CloudWatch to ensure you are receiving the metrics, logs, or Windows event logs you expect. If you are satisfied with the results, you can optionally Store CloudWatch Agent Configuration Settings in Parameter Store (p. 364). If the migration is not successful or the results are not as expected, you can Roll Back to Log Collection with the SSM Agent (p. 365).

To automatically migrate to the CloudWatch Agent (console)

   - or -
   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   
   **Note**
   If you are using the Amazon EC2 console, some field names and locations may differ slightly.

2. In the navigation pane, choose Run Command, and then choose Run command.

   **Note**
   AWS Systems Manager only: If the AWS Systems Manager home page opens, scroll down and choose Explore Run Command.

3. In the Command document list, choose AmazonCloudWatch-MigrateCloudWatchAgent.

4. In the Targets section, choose an option and select the instances to update.

5. Choose Run.

To automatically migrate to the CloudWatch Agent (AWS CLI)

- Run the following command:

  ```
  aws ssm send-command --document-name AmazonCloudWatch-MigrateCloudWatchAgent --targets Key=instanceids,Values=ID1, ID2, ID3
  ```

  ID1, ID2, and ID3 represent the IDs of instances you want to update, such as i-1234567890abcdef0.

Manually Migrate to the CloudWatch Agent

For on-premises Windows instances or Amazon EC2 Windows instances, follow these steps to manually migrate log collection to the Amazon CloudWatch Agent.

One: Install the CloudWatch Agent

   - or -
   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

   **Note**
   If you are using the Amazon EC2 console, some field names and locations may differ slightly.
2. In the navigation pane, choose Run Command, and then choose Run command.
   
   **Note**
   AWS Systems Manager only: If the AWS Systems Manager home page opens, scroll down and choose Explore Run Command.

3. In the Command document list, choose AWS-ConfigureAWSPackage.

4. In the Targets section, choose an option and select the instances to update.

5. In the Action list, choose Install.

6. In Name, type AmazonCloudWatchAgent.

7. In Version, type latest if it is not already provided by default.


**Two: Update Config Data JSON Format**

- To update the JSON formatting of the existing config settings for the CloudWatch Agent, use AWS Systems Manager Run Command or log into the instance directly with an RDP connection to run the following Windows PowerShell commands on the instance, one at a time:

  ```powershell
cd $({Env:ProgramFiles}\Amazon\AmazonCloudWatchAgent)\amazon-cloudwatch-agent-config-wizard.exe --isNonInteractiveWindowsMigration
  ```

  `{Env:ProgramFiles}` represents the location where the Amazon folder containing the CloudWatch Agent can be found, typically `C:\Program Files`.

**Three: Configure and Start the CloudWatch Agent**

   
   -or-

   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   
   **Note**
   If you are using the Amazon EC2 console, some field names and locations may differ slightly.

2. In the navigation pane, choose Run Command, and then choose Run command.
   
   **Note**
   AWS Systems Manager only: If the AWS Systems Manager home page opens, scroll down and choose Explore Run Command.

3. In the Command document list, choose AWS-RunPowerShellScript.

4. In the Targets section, choose an option and select the instances to update.

5. In the Commands box, enter the following two commands:

  ```powershell
cd $({Env:ProgramFiles}\Amazon\AmazonCloudWatchAgent)\amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m ec2 -c file:config.json -s
  ```

  `{Env:ProgramFiles}` represents the location where the Amazon folder containing the CloudWatch Agent can be found, typically `C:\Program Files`.

Four: Disable Log Collection in the SSM Agent

   -or-
   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

   **Note**
   If you are using the Amazon EC2 console, some field names and locations may differ slightly.

2. In the navigation pane, choose **Run Command**, and then choose **Run command**.

   **Note**
   AWS Systems Manager only: If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.

3. In the **Command document** list, choose **AWS-ConfigurecloudWatch**.

4. In the **Targets** section, choose an option and select the instances to update.

5. In the **Status** list, choose **Disabled**.

6. Choose **Run**.

   After completing these steps, check your logs in CloudWatch to ensure you are receiving the metrics, logs, or Windows event logs you expect. If you are satisfied with the results, you can optionally **Store CloudWatch Agent Configuration Settings in Parameter Store** (p. 364). If the migration is not successful or the results are not as expected, you can **Roll Back to Log Collection with the SSM Agent** (p. 365).

Store CloudWatch Agent Configuration Settings in Parameter Store

You can store the contents of an Amazon CloudWatch Agent configuration file in Parameter Store. By maintaining this configuration data in a parameter, multiple instances can derive their configuration settings from it, and you avoid having to create or manually update configuration files on your instances. For example, you can use Run Command to write the contents of the parameter to configuration files on multiple instances, or use State Manager to help avoid configuration drift in the CloudWatch Agent configuration settings across a fleet of instances.

When you run the CloudWatch Agent configuration wizard, you can choose to let the wizard save your configuration settings as a new parameter in Parameter Store. For information about running the CloudWatch Agent configuration wizard, see Create the CloudWatch Agent Configuration File with the Wizard.

If you ran the wizard but did not choose the option to save the settings as a parameter, or you created the CloudWatch Agent configuration file manually, you can retrieve the data to save as a parameter on your instance in the following file:

```
${Env:ProgramFiles}\Amazon\AmazonCloudWatchAgent\config.json
```

`{Env:ProgramFiles}` represents the location where the Amazon folder containing the CloudWatch Agent can be found, typically `C:\Program Files`.

We recommend keeping a backup of the JSON in this file on a location other than the instance itself.

For information about creating a parameter, see **Creating Systems Manager Parameters** (p. 337).
For more information about the CloudWatch Agent, see Collect Metrics from Amazon Elastic Compute Cloud Instances and On-Premises Servers with the CloudWatch Agent in the Amazon CloudWatch User Guide.

Roll Back to Log Collection with the SSM Agent

If you want to return to using the SSM Agent for log collection, follow these steps.

One: Retrieve Config Data from the SSM Agent

1. On the instance where you want to return to collecting logs with the SSM Agent, locate the contents of the SSM Agent config file. This JSON file is typically found in the following location:

   ```
   ${Env:ProgramFiles}\Amazon\SSM\Plugins\awsCloudWatch\AWS.EC2.Windows.CloudWatch.json
   ```

   `{Env:ProgramFiles}` represents the location where the Amazon folder can be found, typically C:\Program Files.

2. Copy this data into a text file for use in a later step.

   We recommend storing a backup of the JSON on a location other than the instance itself.

Two: Uninstall the CloudWatch Agent

   -or-

   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

   Note
   If you are using the Amazon EC2 console, some field names and locations may differ slightly.

2. In the navigation pane, choose Run Command, and then choose Run command.

   Note
   AWS Systems Manager only: If the AWS Systems Manager home page opens, scroll down and choose Explore Run Command.

3. In the Command document list, choose AWS-ConfigureAWSPackage.

4. In the Targets section, choose an option and select the instances to update.

5. In the Action list, choose Uninstall.

6. In Name, type AmazonCloudWatchAgent.

7. Choose Run.

Three: Reenable Log Collection in the SSM Agent

   -or-

   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

   Note
   If you are using the Amazon EC2 console, some field names and locations may differ slightly.
2. In the navigation pane, choose **Run Command**, and then choose **Run command**.

   **Note**
   AWS Systems Manager only: If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.

3. In the **Command document** list, choose **AWS-ConfigureCloudWatch**.

4. In the **Targets** section, choose an option and select the instances to update.

5. In the **Status** list, choose **Enabled**.

6. In the **Properties** box (AWS Systems Manager console) or **Parameters** box (Amazon EC2 console), paste the contents of the old config data you saved to the text file.

7. Choose **Run**.
Authentication and Access Control for AWS Systems Manager

Access to AWS Systems Manager requires credentials. Those credentials must have permissions to access AWS resources for tasks such as creating or updating documents and registering tasks and targets with Maintenance Windows. The following sections provide details on how you can use AWS Identity and Access Management (IAM) and Systems Manager to help secure access to your resources:

- Authentication (p. 367)
- Access Control (p. 368)

**Note**
For more information about configuring access to AWS Systems Manager, see Configuring Access to Systems Manager (p. 8).

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. These are your root credentials and they provide complete access to all of your AWS resources.

  **Important**
  For security reasons, we recommend that you use the root credentials only to create an administrator user, which is an IAM user with full permissions to your AWS account. Then, you can use this administrator user to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating an Admin User and Group in the IAM User Guide.

- **IAM user** – An IAM user is simply an identity within your AWS account that has specific custom permissions (for example, permissions to send event data to a target in Systems Manager). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

In addition to a user name and password, you can also generate access keys for each user. Users can use these keys when accessing AWS services programmatically, either through one of the several SDKs or by using the AWS Command Line Interface (AWS CLI). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don’t use the AWS tools, you must sign the request yourself. Systems Manager supports Signature Version 4, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the AWS General Reference.

- **IAM role** – An IAM role is another IAM identity that you can create in your account and which has specific permissions. It is similar to an IAM user, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:
Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access Systems Manager resources. For example, you must have permissions to create, view, or delete activations, associations, documents, and Maintenance Windows; to register or deregister instances and patch baselines; and so on.

The following sections describe how to manage permissions for Systems Manager. We recommend that you read the overview first.

- Overview of Managing Access Permissions to Your AWS Systems Manager Resources (p. 368)
- Using Identity-based Policies (IAM Policies) for AWS Systems Manager (p. 374)
- AWS Systems Manager Permissions Reference (p. 377)

Overview of Managing Access Permissions to Your AWS Systems Manager Resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles). Some services—such as AWS Lambda, Amazon Simple Notification Service (Amazon SNS), and Amazon Simple Storage Service (Amazon S3)—also support attaching permissions policies to resources.
Note
An account administrator (or administrator user) is a user with administrator privileges. For more information, see IAM Best Practices in the IAM User Guide.

When granting permissions, the account administrator decides who gets the permissions, the resources that they get permissions for, and the specific actions that you want to allow on those resources.

Topics
- AWS Systems Manager Resources and Operations (p. 369)
- Understanding Resource Ownership (p. 371)
- Managing Access to Resources (p. 371)
- Specifying Policy Elements: Resources, Actions, Effects, and Principals (p. 373)
- Specifying Conditions in a Policy (p. 373)

AWS Systems Manager Resources and Operations

Systems Manager includes several primary resources:

- Automation definition
- Automation execution
- Document
- Maintenance Window
- Managed instance
- Managed instance inventory
- Parameter
- Patch baseline
- Resource data sync

For automation definitions, Systems Manager supports a second-level resource, version ID. In AWS, these second-level resources are, known as subresources. Specifying a version subresource for an automation definition resource lets you provide access to certain versions of an automation definition. For example, you might want to ensure that only the latest version of an automation definition is used in your instance management.

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 five levels. We suggest that you create hierarchies that reflect an existing hierarchical structure in your environment. For more information, see Creating Systems Manager Parameters (p. 337).

Each resource has a unique Amazon Resource Names (ARNs). In a policy, you identify the resource that a policy applies to by using its ARN. For more information about ARNs, see Amazon Resource Names (ARN) and AWS Service Namespaces in the Amazon Web Services General Reference.

The following table shows the structure of the ARN format for each resource type in Systems Manager:

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>ARN Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Type</td>
<td>ARN Format</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
</tr>
<tr>
<td>Managed instance inventory</td>
<td>arn:aws:ssm:region:account-id:managed-instance-inventory/managed-instance-id</td>
</tr>
</tbody>
</table>
| Parameter | A one-level parameter:
  • arn:aws:ssm:region:account-id:parameter/parameter-name/
A parameter named with a hierarchical construction:
| Patch baseline | arn:aws:ssm:region:account-id:patchbaseline/patch-baseline-ID |
| All Systems Manager resources | arn:aws:ssm:* |
| All Systems Manager resources owned by the specified account in the specified region | arn:aws:ssm:region:account-id:* |

**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.

For example, you can indicate a specific document (`myDocument`) in your statement using its ARN as follows:


You can specify all documents that belong to a specific account by using the wildcard character (*) as follows:


For Parameter Store API actions, you can provide or restrict access to all parameters in one level of a hierarchy by using hierarchical names and AWS Identity and Access Management (IAM) policies as follows:

To specify all resources, or when a specific API action does not support ARNs, use the wildcard character (*) in the Resource element as follows:

"Resource": "*"

Some Systems Manager API actions accept multiple resources. To specify multiple resources in a single statement, separate their ARNs with commas as follows:

"Resource": ["arn1", "arn2"]

For a list of Systems Manager operations that work with these resource types, see AWS Systems Manager Permissions Reference (p. 377).

Understanding Resource Ownership

A resource owner is the AWS account that created the resource, regardless of who in the account created the resources. Specifically, the resource owner is the AWS account of the principal entity (the root account, an IAM user, or an IAM role) that authenticates the resource creation request. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create a rule, your AWS account is the owner of the Systems Manager resource.
- If you create an IAM user in your AWS account and grant permissions to create Systems Manager resources to that user, the user can create Systems Manager resources. However, your AWS account, to which the user belongs, owns the Systems Manager resources.
- If you create an IAM role in your AWS account with permissions to create Systems Manager resources, anyone who can assume the role can create Systems Manager resources. Your AWS account, to which the role belongs, owns the Systems Manager resources.

Managing Access to Resources

A permissions policy describes who has access to what. The following section explains the available options for creating permissions policies.

Note
This section discusses using IAM in the context of Systems Manager. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What Is IAM? in the IAM User Guide. For information about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the IAM User Guide.

Policies attached to an IAM identity are referred to as identity-based policies (IAM policies). Policies attached to a resource are referred to as resource-based policies. Systems Manager supports only identity-based policies.

Topics
- Identity-Based Policies (IAM Policies) (p. 371)
- Resource-Based Policies (p. 373)

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. By creating identity-based IAM policies, you can restrict the calls and resources that users in your account have access to, and then attach those policies to IAM users. For more information about how to create IAM roles and to explore example IAM policy statements
for Systems Manager, see Overview of Managing Access Permissions to Your AWS Systems Manager Resources (p. 368). For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to view applications, deployment groups, and other Systems Manager resources in the AWS Systems Manager console, you can attach a permissions policy to a user or a group that the user belongs to.

- **Attach a permissions policy to a role (grant cross-account permissions)** – To grant cross-account permissions, you can attach an identity-based permissions policy to an IAM role. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:

  1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.

  2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.

  3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. If you want to grant an AWS service permissions to assume the role, the principal in the trust policy can also be an AWS service principal.

For more information about using IAM to delegate permissions, see Access Management in the IAM User Guide.

The types of actions that you can control access to with resource-based policies vary depending on the resource type, as outlined in the following table:

<table>
<thead>
<tr>
<th>Resource types</th>
<th>Action types</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>View and list details about resources</td>
</tr>
<tr>
<td>Automation definition</td>
<td>Start</td>
</tr>
<tr>
<td></td>
<td>Stop</td>
</tr>
<tr>
<td>Document</td>
<td>Create</td>
</tr>
<tr>
<td>Maintenance Window</td>
<td>Delete</td>
</tr>
<tr>
<td>Parameter</td>
<td>Update</td>
</tr>
<tr>
<td>Managed instance</td>
<td>Deregister</td>
</tr>
<tr>
<td></td>
<td>Register</td>
</tr>
<tr>
<td>Managed instance inventory</td>
<td>Create</td>
</tr>
<tr>
<td></td>
<td>Update</td>
</tr>
<tr>
<td>Patch baseline</td>
<td>Create</td>
</tr>
<tr>
<td></td>
<td>Delete</td>
</tr>
</tbody>
</table>
Specifying Policy Elements: Resources, Actions, Effects, and Principals

<table>
<thead>
<tr>
<th>Resource types</th>
<th>Action types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deregister</td>
<td></td>
</tr>
<tr>
<td>Register</td>
<td></td>
</tr>
<tr>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>Resource data sync</td>
<td>Create</td>
</tr>
<tr>
<td></td>
<td>Delete</td>
</tr>
</tbody>
</table>

Resource-Based Policies

Other AWS services, such as Amazon Simple Storage Service, also 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.

Specifying Policy Elements: Resources, Actions, Effects, and Principals

For each Systems Manager resource, Systems Manager defines a set of applicable API operations. To allow you to grant permissions for these API operations, Systems Manager defines a set of actions that you can specify in a policy. Some API operations can require permissions for more than one action. For more information about resources and API operations, see AWS Systems Manager Resources and Operations (p. 369) and AWS Systems Manager Permissions Reference (p. 377). For a list of actions, see AWS Systems Manager Resources and Operations (p. 369) Actions.

The following are the basic policy elements:

- **Resource** – You use an Amazon Resource Name (ARN) to identify the resource that the policy applies to. For Systems Manager resources, you can, use the wildcard character (*) in IAM policies. For more information, see AWS Systems Manager Resources and Operations (p. 369).

- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, the `ssm:GetDocument` permission allows the user permissions to perform the `GetDocument` operation.

- **Effect** – You specify the effect that occurs when the user requests the specific action, either allow or deny. If you don’t explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.

- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions. Systems Manager supports only identity-based policies.

To learn more about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the IAM User Guide.

For a table showing all of the Systems Manager API actions and the resources that they apply to, see AWS Systems Manager Permissions Reference (p. 377).

Specifying Conditions in a Policy

When you grant permissions, you can use the language in the access policy to specify the conditions under which a policy should take effect. For example, you might want a policy to be applied only after a
specific date. For more information about specifying conditions in a policy language, see Condition in the
IAM User Guide.

To express conditions, you use predefined condition keys. There are no condition keys specific to AWS
Systems Manager. However, there are AWS-wide condition keys that you can use as appropriate. For a
complete list of AWS-wide keys, see Available Keys for Conditions in the IAM User Guide.

Using Identity-based Policies (IAM Policies) for
AWS Systems Manager

The following examples of identity-based policies demonstrate how an account administrator can attach
permissions policies to IAM identities (that is, users, groups, and roles) and thereby grant permissions to
perform operations on Systems Manager resources.

Important
We recommend that you first review the introductory topics that explain the basic concepts
and options available to manage access to your Systems Manager resources. For more
information, see Overview of Managing Access Permissions to Your AWS Systems Manager
Resources (p. 368).

Topics
• Permissions Required to Use the AWS Systems Manager Console (p. 374)
• AWS Managed (Predefined) Policies for AWS Systems Manager (p. 375)
• Customer Managed Policy Examples (p. 376)

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-west-2 region:

```json
{
   "Version": "2012-10-17",
   "Statement" : [
      {
         "Effect" : "Allow",
         "Action" : ["ssm:DeleteDocument"],
      }
   ]
}
```

Permissions Required to Use the AWS Systems
Manager Console

To use the AWS Systems Manager console, a user must have a minimum set of permissions that allows
the user to describe other AWS resources for their AWS account. To fully use Systems Manager in the
Systems Manager console, you must have permissions from the following services:

• AWS Systems Manager
• Amazon Elastic Compute Cloud (Amazon EC2)
• AWS Identity and Access Management (IAM)
You can grant the required permissions with the following policy statement:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:*",
                "ec2:describeInstances",
                "iam:PassRole",
                "iam:ListRoles"
            ],
            "Resource": "*"
        }
    ]
}
```

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended for users with that IAM policy. To ensure that those users can use the Systems Manager console, also attach the AmazonSSMReadOnlyAccess managed policy to the user, as described in AWS Managed (Predefined) Policies for AWS Systems Manager (p. 375).

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the Systems Manager API.

### AWS Managed (Predefined) Policies for AWS Systems Manager

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so you can avoid having to investigate which permissions are needed. For more information, see AWS Managed Policies in the IAM User Guide.

The following AWS managed policies, which you can attach to users in your account, are specific to AWS Systems Manager:

- **AmazonSSMFullAccess** – User trust policy that grants full access to the Systems Manager API and documents.

- **AmazonSSMAutomationRole** – Service role that provides permissions for the AWS Systems Manager automation service to execute activities defined within automation documents. Assign this policy to administrators and trusted power users.

- **AmazonSSMReadOnlyAccess** – User trust policy that grants access to Systems Manager read-only API actions, such as `Get*` and `List*`.

- **AmazonSSMMaintenanceWindowRole** – Service role for Systems Manager Maintenance Windows.

- **AmazonEC2RoleforSSM** – Instance trust policy that enables an instance to communicate with the Systems Manager API.

You can also create your own custom IAM policies to allow permissions for Systems Manager actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.
Note
In a hybrid environment, you need an additional IAM role that allows servers and VMs to communicate with the Systems Manager service. This is the IAM service role for Systems Manager. This role grants AWS Security Token Service (AWS STS) AssumeRole trust to the Systems Manager service. The AssumeRole action returns a set of temporary security credentials (consisting of an access key ID, a secret access key, and a security token). You use these temporary credentials to access AWS resources that you might not normally have access to. For more information, see Create an IAM Service Role (p. 28) and AssumeRole in AWS Security Token Service API Reference.

Customer Managed Policy Examples

You can create standalone policies that you administer in your own AWS account. We refer to these as customer managed policies. You can attach these policies to multiple principal entities in your AWS account. When you attach a policy to a principal entity, you give the entity the permissions that are defined in the policy. For more information, see Customer Managed Policies in IAM User Guide.

The following examples of user policies grant permissions for various AWS Systems Manager actions. Use them to limit the Systems Manager access for your IAM users and roles. These policies work when performing actions in the Systems Manager API, AWS SDKs, or the AWS CLI. For users who use the console, you need to grant additional permissions specific to the console. For more information, see Permissions Required to Use the AWS Systems Manager Console (p. 374).

Note
All examples use the US West (Oregon) Region (us-west-2) and contain fictitious account IDs.

Examples

- Example 1: Allow a User to Perform Systems Manager Operations in a Single Region (p. 376)
- Example 2: Allow a User to List Documents for a Single Region (p. 376)

Example 1: Allow a User to Perform Systems Manager Operations in a Single Region

The following example grants permissions to perform AWS Systems Manager operations only in the us-west-2 Region:

```json
{
   "Version": "2012-10-17",
   "Statement" : [
      {
         "Effect" : "Allow",
         "Action" : [
            "arn:aws:ssm:*"
         ],
         "Resource" : [
            "arn:aws:ssm:us-west-2:11122233444:*"
          ]
      }
   ]
}
```

Example 2: Allow a User to List Documents for a Single Region

The following example grants permissions to list all document names that begin with Update in the us-west-2 Region:

```json
```
AWS Systems Manager Permissions Reference

The following table lists the AWS Systems Manager API operations and their corresponding actions for which you can grant permissions. Use this table as a reference when setting up Access Control (p. 368) and writing permissions policies to attach to an IAM identity (identity-based policies). You specify the actions in the policy’s Action field. To specify an action, use the ssm: prefix followed by the API operation name (for example, ssm:GetDocument and ssm:CreateDocument). To specify multiple actions in a single statement, separate them with commas (for example, "Action": ["ssm:action1", "ssm:action2"]). For the resource value in the policy’s Resource field, you specify an ARN. To specify multiple actions or resources, use a wildcard character (*) in your ARN. For example, ssm:* specifies all of the Systems Manager actions, and ssm:Get* specifies all of the Systems Manager actions that begin with the word Get. The following example grants access to all documents with names that begin with West:

```
```

For more information about wildcards, see IAM Identifiers in IAM User Guide. For a list of Systems Manager resources with the ARN format, see AWS Systems Manager Resources and Operations (p. 369).

To express conditions, use AWS-wide condition keys in your Systems Manager policies. For a complete list of AWS-wide keys, see Available Keys in the IAM User Guide.

**Specifying multiple actions or resources**

**Systems Manager API Operations and Required Permissions for Actions**

**AddTagsToResource**

**Action(s):** ssm:AddTagsToResource

Required to add or overwrite tags for a specified resource.

**CancelCommand**

**Action(s):** ssm:CancelCommand

Required to attempt to cancel the command with the specified command ID.

**CreateActivation**

**Action(s):** ssm:CreateActivation

Required to register an on-premises server or virtual machine with Amazon EC2 so that it can be managed using ’
CreateAssociation

**Action(s):** ssm:CreateAssociation

Required to associate a Systems Manager document with the specified instances or targets.

CreateAssociationBatch

**Action(s):** ssm:CreateAssociationBatch

Required to associate multiple Systems Manager documents with the specified instances or targets.

CreateDocument

**Action(s):** ssm:CreateDocument

Required to create a Systems Manager document.

CreateMaintenanceWindow

**Action(s):** ssm:CreateMaintenanceWindow

Required to create a Maintenance Window.

CreatePatchBaseline

**Action(s):** ssm:CreatePatchBaseline

Required to create a patch baseline.

CreateResourceDataSync

**Action(s):** ssm:CreateResourceDataSync

Required to create a resource data sync configuration for a single Amazon S3 bucket.

DeleteActivation

**Action(s):** ssm:DeleteActivation

Required to delete an activation.

DeleteAssociation

**Action(s):** ssm:DeleteAssociation

Required to disassociate the specified Systems Manager document from the specified instance.

DeleteDocument

**Action(s):** ssm:DeleteDocument

Required to delete a Systems Manager document and all instance associations to the document.

DeleteMaintenanceWindow

**Action(s):** ssm:DeleteMaintenanceWindow

Required to delete a Maintenance Window.

DeleteParameter

**Action(s):** ssm:DeleteParameter

Required to delete a parameter from the system.

DeleteParameters

**Action(s):** ssm:DeleteParameters
Required to delete one or more parameters from the system.

**DeletePatchBaseline**

**Action(s):** `ssm:DeletePatchBaseline`  
Required to delete a patch baseline.

**DeleteResourceDataSync**

**Action(s):** `ssm:DeleteResourceDataSync`  
Required to delete a resource data sync configuration.

**DeregisterManagedInstance**

**Action(s):** `ssm:DeregisterManagedInstance`  
Required to remove a server or virtual machine from the list of registered servers.

**DeregisterPatchBaselineForPatchGroup**

**Action(s):** `ssm:DeregisterPatchBaselineForPatchGroup`  
Required to remove a patch group from a patch baseline.

**DeregisterTargetFromMaintenanceWindow**

**Action(s):** `ssm:DeregisterTargetFromMaintenanceWindow`  
Required to remove a target from a Maintenance Window.

**DeregisterTaskFromMaintenanceWindow**

**Action(s):** `ssm:DeregisterTaskFromMaintenanceWindow`  
Required to remove a task from a Maintenance Window.

**DescribeActivations**

**Action(s):** `ssm:DescribeActivations`  
Required to view details about an activation, such as the date and time the activation was created, the expiration date, and the IAM role assigned to the instances in the activation.

**DescribeAssociation**

**Action(s):** `ssm:DescribeAssociation`  
Required to view the associations for the specified Systems Manager document or instance.

**DescribeAutomationExecutions**

**Action(s):** `ssm:DescribeAutomationExecutions`  
Required to view information about all active and terminated Automation executions.

**DescribeAutomationStepExecutions**

**Action(s):** `ssm:DescribeAutomationStepExecutions`  
Required to view information about all active and terminated step executions in an Automation workflow.

**DescribeAvailablePatches**

**Action(s):** `ssm:DescribeAvailablePatches`  
Required to view information about patches that could be included in a patch baseline.
DescribeDocument

**Action(s):** `ssm:DescribeDocument`

Required to view information about the specified Systems Manager document.

DescribeDocumentPermission

**Action(s):** `ssm:DescribeDocumentPermission`

Required to view the permissions for a Systems Manager document.

DescribeEffectiveInstanceAssociations

**Action(s):** `ssm:DescribeEffectiveInstanceAssociations`

Required to view information about associations for one or more instances.

DescribeEffectivePatchesForPatchBaseline

**Action(s):** `ssm:DescribeEffectivePatchesForPatchBaseline`

Required to view information about the current effective patches (the patch and the approval state) for the specified patch baseline. Applies only to Windows Server patch baselines.

DescribeInstanceAssociationsStatus

**Action(s):** `ssm:DescribeInstanceAssociationsStatus`

Required to view the status of the associations for one or more instances.

DescribeInstanceInformation

**Action(s):** `ssm:DescribeInstanceInformation`

Required to view information about one or more instances.

DescribeInstancePatches

**Action(s):** `ssm:DescribeInstancePatches`

Required to view information about the patches on a specified instance and their state relative to the patch baseline being used for the instance.

DescribeInstancePatchStates

**Action(s):** `ssm:DescribeInstancePatchStates`

Required to view information about the high-level patch state of one or more instances.

DescribeInstancePatchStatesForPatchGroup

**Action(s):** `ssm:DescribeInstancePatchStatesForPatchGroup`

Required to view the high-level patch state for the instances in a specified patch group.

DescribeMaintenanceWindowExecutions

**Action(s):** `ssm:DescribeMaintenanceWindowExecutions`

Required to view information about the execution of a Maintenance Window. This includes details about when the Maintenance Window was scheduled to be active and information about tasks registered and run with the maintenance window.

DescribeMaintenanceWindowExecutionTaskInvocations

**Action(s):** `ssm:DescribeMaintenanceWindowExecutionTaskInvocations`
Required to retrieve information about the individual task executions (one per target) for a particular task executed as part of a Maintenance Window execution.

**DescribeMaintenanceWindowExecutionTasks**

**Action(s):** `ssm:DescribeMaintenanceWindowExecutionTasks`

Required to view information about the tasks that have been run for a specified Maintenance Window execution.

**DescribeMaintenanceWindows**

**Action(s):** `ssm:DescribeMaintenanceWindows`

Required to view information about the Maintenance Windows created in an AWS account.

**DescribeMaintenanceWindowTargets**

**Action(s):** `ssm:DescribeMaintenanceWindowTargets`

Required to view information about the targets registered with a specified Maintenance Window.

**DescribeMaintenanceWindowTasks**

**Action(s):** `ssm:DescribeMaintenanceWindowTasks`

Required to view information about the tasks in a specified maintenance window.

**DescribeParameters**

**Action(s):** `ssm:DescribeParameters`

Required to view information about one or more parameters.

**DescribePatchBaselines**

**Action(s):** `ssm:DescribePatchBaselines`

Required to view information about the patch baselines in an AWS account.

**DescribePatchGroups**

**Action(s):** `ssm:DescribePatchGroups`

Required to view information about all patch groups that have been registered with patch baselines.

**DescribePatchGroupState**

**Action(s):** `ssm:DescribePatchGroupState`

Required to view information about the high-level aggregated patch compliance state for a patch group.

**GetAutomationExecution**

**Action(s):** `ssm:GetAutomationExecution`

Required to view detailed information about a particular Automation execution.

**GetCommandInvocation**

**Action(s):** `ssm:GetCommandInvocation`

Required to view detailed information about command execution for an invocation or plugin.

**GetDefaultPatchBaseline**

**Action(s):** `ssm:GetDefaultPatchBaseline`
Required to view information about the default patch baseline.

**GetDeployablePatchSnapshotForInstance**

**Action(s):** ssm:GetDeployablePatchSnapshotForInstance

Required to view the current snapshot for the patch baseline used by the instance. Used primarily by the AWS-RunPatchBaseline Systems Manager document.

**GetDocument**

**Action(s):** ssm:GetDocument

Required to view the contents of a specified Systems Manager document.

**GetInventory**

**Action(s):** ssm:GetInventory

Required to view information about inventory items.

**GetInventorySchema**

**Action(s):** ssm:GetInventorySchema

Required to view inventory type names for the account, or to return a list of attribute names for a specific inventory item type.

**GetMaintenanceWindow**

**Action(s):** ssm:GetMaintenanceWindow

Required to view information about a specified Maintenance Window.

**GetMaintenanceWindowExecution**

**Action(s):** ssm:GetMaintenanceWindowExecution

Required to view information about a specific task run as part of a Maintenance Window execution.

**GetMaintenanceWindowExecutionTask**

**Action(s):** ssm:GetMaintenanceWindowExecutionTask

Required to view information about a specific task run as part of a Maintenance Window execution.

**GetMaintenanceWindowExecutionTaskInvocation**

**Action(s):** ssm:GetMaintenanceWindowExecutionTaskInvocation

Required to retrieve a task invocation, which is a specific task executing on a specific target.

**GetMaintenanceWindowTask**

**Action(s):** ssm:GetMaintenanceWindowTask

Required to list the tasks in a Maintenance Window.

**GetParameter**

**Action(s):** ssm:GetParameter

Required to view information about a specified parameter, including the parameter name, type, and value.

**GetParameterHistory**

**Action(s):** ssm:GetParameterHistory
Required to view historical information about a specified parameter. In addition to parameter name, type, and value, returns the parameter description, query key ID, last modified date, and ARN of the AWS user who last modified the parameter.

GetParameters

Action(s): ssm:GetParameters

Required to view information about parameters.

GetParametersByPath

Action(s): ssm:GetParametersByPath

Required to view information about parameters in a hierarchical structure.

GetPatchBaseline

Action(s): ssm:GetPatchBaseline

Required to view information about a patch baseline.

GetPatchBaselineForPatchGroup

Action(s): ssm:GetPatchBaselineForPatchGroup

Required to view information about the patch baseline that should be used for a specified patch group.

ListAssociations

Action(s): ssm:ListAssociations

Required to view the associations for the specified Systems Manager document or instance.

ListAssociationVersions

Action(s): ssm:ListAssociationVersions

Required to retrieve all versions of an association for a specific association ID.

ListCommandInvocations

Action(s): ssm:ListCommandInvocations

Required to view a list of invocations, or copies of commands sent to a specific instance.

ListCommands

Action(s): ssm:ListCommands

Required to view a list of commands requested by users of the AWS account.

ListComplianceItems

Action(s): ssm:ListComplianceItems

Required to retrieve a list of compliance statuses for different resource types for a specific resource ID.

ListComplianceSummaries

Action(s): ssm:ListComplianceSummaries

Required to retrieve a summary count of compliant and non-compliant resources for a compliance type.

ListDocuments

Action(s): ssm:ListDocuments
Required to view a list of Systems Manager documents.

**ListDocumentVersions**

**Action(s):** `ssm:ListDocumentVersions`

Required to view information about the versions of a document.

**ListInventoryEntries**

**Action(s):** `ssm:ListInventoryEntries`

Required to view information about inventory items on an instance.

**ListResourceComplianceSummaries**

**Action(s):** `ssm:ListResourceComplianceSummaries`

Required to retrieve a resource-level summary count, including information about compliant and non-compliant statuses.

**ListResourceDataSync**

**Action(s):** `ssm:ListResourceDataSync`

Required to view information about resource data sync configurations, including when a sync last attempted to start, the last sync status, and the last time a sync completed successfully.

**ListTagsForResource**

**Action(s):** `ssm:ListTagsForResource`

Required to view a list of tags assigned to a specified resource.

**ModifyDocumentPermission**

**Action(s):** `ssm:ModifyDocumentPermission`

Required to shared a Systems Manager document publicly or privately.

**PutComplianceItems**

**Action(s):** `ssm:PutComplianceItems`

Required to register a compliance type and other compliance details on a designated resource.

**PutInventory**

**Action(s):** `ssm:PutInventory`

Required to add or update custom inventory items on one or more instances.

**PutParameter**

**Action(s):** `ssm:PutParameter`

Required to add one or more parameters to the system.

**RegisterDefaultPatchBaseline**

**Action(s):** `ssm:RegisterDefaultPatchBaseline`

Required to define the default patch baseline.

**RegisterPatchBaselineForPatchGroup**

**Action(s):** `ssm:RegisterPatchBaselineForPatchGroup`

Required to register a patch baseline for a patch group.
RegisterTargetWithMaintenanceWindow

Action(s): ssm:RegisterTargetWithMaintenanceWindow

Required to register a target with a Maintenance Window.

RegisterTaskWithMaintenanceWindow

Action(s): ssm:RegisterTaskWithMaintenanceWindow

Required to register a task with a Maintenance Window.

RemoveTagsFromResource

Action(s): ssm:RemoveTagsFromResource

Required to remove tags from a specified resource.

SendAutomationSignal

Action(s): ssm:SendAutomationSignal

Required to send a signal to an Automation execution to change the current behavior or status of the execution.

SendCommand

Action(s): ssm:SendCommand

Required to run commands on one or more managed instances.

StartAutomationExecution

Action(s): ssm:StartAutomationExecution

Required to start running an Automation document.

StopAutomationExecution

Action(s): ssm:StopAutomationExecution

Required to start running an Automation document.

UpdateAssociation

Action(s): ssm:UpdateAssociation

Required to update an association. Updates can be made only to the document version, schedule, parameters, and Amazon S3 output of an association.

UpdateAssociationStatus

Action(s): ssm:UpdateAssociationStatus

Required to update the status of the Systems Manager document associated with a specified instance.

UpdateDocument

Action(s): ssm:UpdateDocument

Required to update the content, version, or name of a document.

UpdateDocumentDefaultVersion

Action(s): ssm:UpdateDocumentDefaultVersion

Required to set the default version of a document.
**UpdateMaintenanceWindow**

**Action(s):** `ssm:UpdateMaintenanceWindow`

Required to update one or more parameters in a Maintenance Window.

**UpdateMaintenanceWindowTarget**

**Action(s):** `ssm:UpdateMaintenanceWindowTarget`

Required to modify the target of an existing Maintenance Window.

**UpdateMaintenanceWindowTask**

**Action(s):** `ssm:UpdateMaintenanceWindowTask`

Required to modify the task assigned to a Maintenance Window.

**UpdateManagedInstanceRole**

**Action(s):** `ssm:UpdateManagedInstanceRole`

Required to assign an Amazon Identity and Access Management (IAM) role to a managed instance, or to change the assigned IAM role.

**UpdatePatchBaseline**

**Action(s):** `ssm:UpdatePatchBaseline`

Required to update one or more fields in an existing patch baseline.
Use Cases and Best Practices

This topic lists common use cases and best practices for AWS Systems Manager capabilities. If available, this topic also includes links to relevant blog posts and technical documentation.

Note
The title of each section here is an active link to the corresponding section in the technical documentation.

Automation (p. 92)
- Create self-service runbooks for infrastructure as Automation documents.
- Use Automation to simplify creating AMIs from the AWS Marketplace or custom AMIs, using public SSM documents or by authoring your own workflows.
- Use the AWS-UpdateLinuxAmi and AWS-UpdateWindowsAmi Automation documents (or create custom Automation documents) to build and maintain AMIs. (p. 103)

Inventory (p. 61)
- Use Systems Manager Inventory with AWS Config to audit your application configurations over time.

Maintenance Windows (p. 218)
- Define a schedule to perform potentially disruptive actions on your instances such as OS patching, driver updates, or software installations.

Parameter Store (p. 330)
- Use Parameter Store to centrally manage global configuration settings.
- Use Parameter Store to encrypt and manage secrets by using AWS KMS (p. 351).
- Use Parameter Store with ECS task definitions to store secrets.

Patch Manager (p. 192)
- Use patch manager to rollout patches at scale and increase fleet compliance visibility across your instances.

Run Command (p. 160)
- Manage Instances at Scale without SSH Access Using EC2 Run Command.
- Audit all API calls made by on or on behalf of Run Command using AWS CloudTrail.
- Use the rate control feature in Run Command to perform a staged command execution (p. 177).
- Use fine-grained access permissions for Run Command (and all Systems Manager capabilities) by using AWS Identity and Access Management (IAM) policies (p. 10).

State Manager (p. 249)
- Update the SSM Agent at least once a month using the pre-configured AWS-UpdateSSMAgent document.
• **Bootstrap EC2 Instances on launch using EC2Config for Windows**

  (Windows) Upload the PowerShell or DSC module to Amazon S3, and use AWS-InstallPowerShellModule.

• Use Amazon EC2 tags to create application groups for your instances. And then target instances using the `Targets` parameter instead of specifying individual instance IDs.

• **Automatically remediate findings generated by Amazon Inspector by using Systems Manager.**

• **Use a centralized configuration repository for all of your SSM documents, and share documents across your organization (p. 273).**
## Systems Manager Documentation Update History

The following table describes important changes to the documentation since the preceding release of AWS Systems Manager.

- API version: 2014-11-06
- Last update: December 14, 2017

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retitled and moved the Systems Manager Automation Actions Reference</td>
<td>Based on customer feedback, the Automation Actions Reference is now called the Systems Manager Automation Document Reference. Furthermore, we moved the reference into the Shared Resources &gt; Documents node so it is closer to the SSM Document Plugin Reference (p. 284). For more information, see Systems Manager Automation Document Reference (p. 305).</td>
<td>December 20, 2017</td>
</tr>
<tr>
<td>New Monitoring chapter and content</td>
<td>A new chapter, Monitoring Instances with AWS Systems Manager (p. 359), provides instructions for sending metrics and log data to Amazon CloudWatch Logs. A new topic, Send Logs to CloudWatch Logs (CloudWatch Agent) (p. 360), provides instructions for migrating on-instance monitoring tasks, on 64-bit Windows Server instances only, from SSM Agent to the CloudWatch Agent.</td>
<td>December 14, 2017</td>
</tr>
<tr>
<td>New chapter</td>
<td>A new chapter, Authentication and Access Control for AWS Systems Manager (p. 367), provides comprehensive information about using AWS Identity and Access Management (IAM) and AWS Systems Manager to help secure access to your resources through the use of credentials. These credentials provide the permissions required to access AWS resources, such as accessing data stored in Amazon S3 buckets and sending commands to and reading the tags on Amazon EC2 instances.</td>
<td>December 11, 2017</td>
</tr>
<tr>
<td>Changes to the left navigation</td>
<td>We changed the headings in the left navigation of this user guide to match the headings in the new AWS Systems Manager console.</td>
<td>December 8, 2017</td>
</tr>
</tbody>
</table>
| 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 easily centralize operational data and automate tasks across your AWS resources. You can access the new AWS Systems Manager console [here](https://aws.amazon.com/systemsmanager/). For more information, see What Is AWS Systems Manager? (p. 1)
- **YAML Support**: You can create SSM documents in YAML. For more information, see AWS Systems Manager Documents (p. 258). | November 29, 2017 |
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using Run Command to Take VSS-Enabled Snapshots of EBS Volumes</td>
<td>Using Run Command, you can take application-consistent snapshots of all Amazon Elastic Block Store (Amazon EBS) volumes attached to your Amazon EC2 Windows instances. The snapshot process uses the Windows Volume Shadow Copy Service (VSS) to take image-level backups of VSS-aware applications, including data from pending transactions between these applications and the disk. Furthermore, you don’t need to shut down your instances or disconnect them when you need to back up all attached volumes. For more information, see Using Run Command to Take VSS-Enabled Snapshots of EBS Volumes (p. 50).</td>
<td>November 20, 2017</td>
</tr>
<tr>
<td>Enhanced Systems Manager Security Available By Using VPC Endpoints</td>
<td>You can improve the security posture of your managed instances (including managed instances in your hybrid environment) by configuring Systems Manager to use an interface VPC endpoint. Interface endpoints are powered by PrivateLink, a technology that enables you to privately access Amazon EC2 and Systems Manager APIs by using private IP addresses. PrivateLink restricts all network traffic between your managed instances, Systems Manager, and EC2 to the Amazon network (managed instances don't have access to the Internet). Also, you don't need an Internet gateway, a NAT device, or a virtual private gateway. For more information, see Setting Up VPC Endpoints for Systems Manager (p. 12).</td>
<td>November 7, 2017</td>
</tr>
</tbody>
</table>
| Inventory Support for Files, Services, Windows Roles, and the Windows Registry | SSM Inventory now supports gathering the following information from your managed instances.  
- **Files**: Name, size, version, installed date, modification and last accessed times, etc.  
- **Services**: Name, display name, status, dependent services, service type, start type, etc.  
- **Windows Registry**: Registry key path, value name, value type, and value.  
- **Windows roles**: Name, display name, path, feature type, installed state, etc.  

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 the SSM Agent (p. 254).  

For more information Inventory, see About Systems Manager Inventory (p. 61). | November 6, 2017 |
<p>| 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 (p. 96). | October 31, 2017 |</p>
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 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 Amazon EC2 instances and on-premises managed instances in your hybrid environment. For more information, see [Partner and Product Integration](p. 38).  

**Create composite SSM documents**: Systems Manager now supports running one or more secondary SSM documents from a primary SSM document. These primary documents that run other documents are called *composite* documents. Composite documents enable you to create and share a standard set of secondary SSM documents across AWS accounts for common tasks such as boot-strapping anti-virus software or domain-joining instances. You can run composite and secondary documents stored in Systems Manager, GitHub, or Amazon S3. After you create a composite document, you can run it by using the `AWS-RunDocument` pre-defined SSM document. For more information, see [Creating Composite Documents](p. 279) and [Running Documents from Remote Locations](p. 281).  

**SSM document plugin reference**: For easier access, we moved the SSM Plugin Reference for SSM documents out of the Systems Manager API Reference and into the User Guide. For more information, see [SSM Document Plugin Reference](p. 284). | October 26, 2017 |
| Support for Parameter Versions in Parameter Store | When you edit a parameter, Parameter Store now automatically iterates the version number by 1. You can specify a parameter name and a specific version number in API calls and SSM documents. If you don't specify a version number, the system automatically uses the latest version.  

Parameter versions provide a layer of protection in the event that a parameter is accidentally changed. You can view the values of all versions, and reference older versions if necessary. You can also use parameter versions to see how many times a parameter changed over a period of time. For more information, see [Working with Parameter Versions](p. 344). | October 24, 2017 |
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Tagging Systems Manager Documents</td>
<td>You can now use the AddTagsToResource API, the AWS CLI, or the AWS Tools for Windows to tag Systems Manager documents with key-value pairs. Tagging helps you quickly identify specific resources based on the tags you've assigned to them. This is in addition to existing tagging support for managed instances, Maintenance Windows, Parameter Store parameters, and patch baselines. New topics include Tagging Systems Manager Documents (p. 270) and Controlling Access to Documents Using Tags (p. 271).</td>
<td>October 3, 2017</td>
</tr>
</tbody>
</table>
| Various Documentation Updates to Fix Errors or Update Content Based on Feedback | • Updated Setting Up AWS Systems Manager in Hybrid Environments (p. 27) with information for Raspbian Linux.  
• Updated Setting Up AWS Systems Manager (p. 4) with new requirement for Windows instances. SSM Agent requires Windows PowerShell 3.0 or later to execute certain SSM Documents on Windows instances (for example, the AWS-ApplyPatchBaseline document). Verify that your Windows 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 (p. 119). | September 29, 2017 |
| SSM Agent Installed By Default on Amazon Linux | SSM Agent is installed, by default, on Amazon Linux AMIs dated 2017.09 and later. You must manually install SSM Agent on other versions of Linux, as described in Installing and Configuring SSM Agent on Linux Instances (p. 15). | September 27, 2017 |
| Run Command Enhancements                  | Run Command includes the following enhancements.  
• You can restrict command execution to specific instances by creating an IAM user policy that includes a condition that the user can only execute commands on instances that are tagged with specific Amazon EC2 tags. For more information, see Restricting Run Command Access Based on Instance Tags (p. 162).  
• 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 Sending Commands to a Fleet (p. 177). | September 12, 2017 |
<p>| Systems Manager Supported on Raspbian     | Systems Manager can now run on Raspbian Jessie and Raspbian Stretch devices, including Raspberry Pi (32-Bit). For more information, see Raspbian (p. 24). | September 7, 2017 |</p>
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Automatically Send SSM Agent Logs to Amazon CloudWatch Logs</strong></td>
<td>You can now make a simple configuration change on your instances to have SSM Agent send log files to CloudWatch. For more information, see [Send Logs to CloudWatch Logs (SSM Agent)](p. 359).</td>
<td>September 7, 2017</td>
</tr>
<tr>
<td><strong>Encrypt Resource Data Sync</strong></td>
<td>Systems Manager Resource Data Sync lets you aggregate Inventory data collected on dozens or hundreds of managed instance in a central Amazon S3 bucket. You can now encrypt Resource Data Sync by using an AWS Key Management Service key. For more information, see [Using Resource Data Sync to Aggregate Inventory Data](p. 78).</td>
<td>September 1, 2017</td>
</tr>
</tbody>
</table>
| **New State Manager Walkthroughs** | Added two new walkthroughs to the State Manager documentation:  
  - *Walkthrough: Automatically Update the SSM Agent* (p. 254)  
  - *Walkthrough: Automatically Update PV Drivers on EC2 Windows Instances* (p. 256) | August 31, 2017 |
<p>| <strong>Systems Manager Configuration Compliance</strong> | Use Configuration Compliance to scan your fleet of managed instances for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren’t compliant. By default, Configuration Compliance displays compliance data about Patch Manager patching and State Manager associations. You can also customize the service and create your own compliance types based on your IT or business requirements. For more information, see [AWS Systems Manager Configuration Compliance](p. 84). | August 28, 2017 |
| <strong>New Automation Action: aws:executeAutomation</strong> | Executes a secondary Automation workflow by calling a secondary Automation document. With this action, you can create Automation documents for your most common workflows, and reference those documents during an Automation execution. This action can simplify your Automation documents by removing the need to duplicate steps across similar documents. For more information, see [aws:executeAutomation](p. 322). | August 22, 2017 |
| <strong>Automation as the Target of a CloudWatch Event</strong> | You can start an Automation workflow by specifying an Automation document as the target of an Amazon CloudWatch event. You can start workflows according to a schedule, or when a specific AWS system event occurs. For more information, see [Configuring Automation as a CloudWatch Events Target (Optional)](p. 102). | August 21, 2017 |
| <strong>State Manager Association Versioning and General Updates</strong> | You can now create different State Manager association versions. There is a limit of 1,000 versions for each association. You can also specify names for your associations. Also, the State Manager documentation has been updated to address outdated information and inconsistencies. For more information, see [AWS Systems Manager State Manager](p. 249). | August 21, 2017 |</p>
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changes to Maintenance Windows</td>
<td>Maintenance Windows include the following changes or enhancements:</td>
<td>August 16, 2017</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• You can edit the targets of a Maintenance Window, specify a target name, description, and owner.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• You can edit tasks in a Maintenance Window, including specifying a new SSM document for Run Command and Automation tasks.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• All Run Command parameters are now supported, including DocumentHash, DocumentHashType, TimeoutSeconds, Comment, and NotificationConfig.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• You can now use a <strong>safe</strong> flag when you attempt to deregister a target. If enabled, the system returns an error if the target is referenced by any task.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see [AWS Systems Manager Maintenance Windows](p. 218).</td>
<td></td>
</tr>
<tr>
<td>New Automation Action: aws:approve</td>
<td>This new action for Automation documents temporarily pauses an Automation execution until designated principals either approve or reject the action. After the required number of approvals is reached, the Automation execution resumes.</td>
<td>August 10, 2017</td>
</tr>
<tr>
<td></td>
<td>For more information, see [Systems Manager Automation Document Reference](p. 305).</td>
<td></td>
</tr>
<tr>
<td>Automation Assume Role No Longer Required</td>
<td>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.</td>
<td>August 3, 2017</td>
</tr>
<tr>
<td></td>
<td>However, the following situations still require that you specify a service role for Automation:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When you want to restrict a user’s privileges on a resource, but you want the user to execute an Automation workflow that requires higher privileges. In this scenario, you can create a service role with higher privileges and allow the user to execute the workflow.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Operations that you expect to run longer than 12 hours require a service role.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see [Setting Up Automation](p. 96).</td>
<td></td>
</tr>
</tbody>
</table>
### Change Description

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Compliance</td>
<td>Use Amazon EC2 Systems Manager Configuration Compliance to scan your fleet of managed instances for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren’t compliant. For more information, see [AWS Systems Manager Configuration Compliance](p. 84).</td>
<td>August 8, 2017</td>
</tr>
</tbody>
</table>
| SSM Document Enhancements     | SSM Command and Policy documents now offer cross-platform support. This means that a single SSM document can process plugins for Windows and Linux operating systems. Cross-platform support enables you to consolidate the number of documents you manage. Cross-platform support is offered in SSM documents that use schema version 2.2 or later.  
SSM Command documents that use schema version 2.0 or later can now include multiple plugins of the same type. For example, you can create a Command document that calls the aws:runRunShellScript plugin multiple times.  
For more information about schema version 2.2 changes, see [AWS Systems Manager Documents](p. 258). For more information about SSM plugins, see [Systems Manager Plugins](p. 65). | July 12, 2017 |
| Linux Patching                | Patch Manager can now patch the following Linux distributions:  
**64-Bit and 32-Bit Systems**  
• Amazon Linux 2014.03, 2014.09, or later  
• Ubuntu Server 16.04 LTS, 14.04 LTS, or 12.04 LTS  
• Red Hat Enterprise Linux (RHEL) 6.5 or later  
**64-Bit Systems Only**  
• Amazon Linux 2015.03, 2015.09, or later  
• Red Hat Enterprise Linux (RHEL) 7.x or later  
For more information, see [AWS Systems Manager Patch Manager](p. 192).  
**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 the SSM Agent* in [Executing Commands from the EC2 Console](p. 174).  
• The AWS-ApplyPatchBaseline SSM document is being replaced by the AWS-RunPatchBaseline document. | July 6, 2017   |
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Data Sync</td>
<td>You can use Systems Manager Resource Data Sync to send Inventory data collected from all of your managed instances to a single Amazon S3 bucket. Resource Data Sync then automatically updates the centralized data when new Inventory data is collected. With all Inventory data stored in a target Amazon S3 bucket, you can use services like Amazon Athena and Amazon QuickSight to query and analyze the aggregated data. For more information, see Configuring Resource Data Sync for Inventory (p. 71). For an example of how to work with Resource Data Sync, see Using Resource Data Sync to Aggregate Inventory Data (p. 78).</td>
<td>June 29, 2017</td>
</tr>
<tr>
<td>Systems Manager Parameter Hierarchies</td>
<td>Managing dozens or hundreds of Systems Manager parameters as a flat list is time-consuming and prone to errors. You can use parameter hierarchies to help you organize and manage Systems Manager parameters. A hierarchy is a parameter name that includes a path that you define by using forward slashes. Here is an example that uses three hierarchy levels in the name to identify the following: /Environment/Type of computer/Application/Data /Dev/DBServer/MySQL/db-string13 For more information, see Organizing Parameters into Hierarchies (p. 334). For an example of how to work with parameter hierarchies, see Walkthrough: Manage Parameters Using Hierarchies (AWS CLI) (p. 357).</td>
<td>June 22, 2017</td>
</tr>
<tr>
<td>SSM Agent Support for SUSE Linux Enterprise Server</td>
<td>You can install the SSM Agent on 64-bit SUSE Linux Enterprise Server (SLES). For more information, see Installing and Configuring SSM Agent on Linux Instances (p. 15).</td>
<td>June 14, 2017</td>
</tr>
</tbody>
</table>
AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the AWS General Reference.