Amazon EC2 Systems Manager

User Guide

Amazon EC2 Systems Manager: User Guide

Table of Contents

What Is Amazon EC2 Systems Manager?

Amazon EC2 Systems Manager is a collection of capabilities that helps you automate management tasks such as collecting system inventory, applying operating system patches, automating the creation of Amazon Machine Images (AMIs), and configuring operating systems and applications at scale. Systems Manager 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.

Features

Tasks	Details
Run Command (p. 169)	Run Command helps you remotely and securely manage the configuration of your managed instances at scale. Use Run Command to perform ad hoc changes like updating applications or running Linux shell scripts and Windows PowerShell commands on a target set of dozens or hundreds of instances.
Inventory (p. 126)	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.
State Management (p. 198)	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.

Tasks	Details
Automation (p. 68)	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.
Patch Management (p. 135)	Patch Manager automates the process of patching Windows 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 EC2 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. 43)	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.
Parameter Store (p. 159)	Parameter Store centralizes the management of configuration data. You can use Parameter Store to store passwords, license keys, or database connection strings that you commonly reference in scripts, commands, or other automation and configuration workflows.
Systems Manager Documents (p. 29)	A Systems Manager 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) and include steps and parameters that you specify. Steps execute in sequential order.

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 Systems Manager (p. 4).

Optionally, you can also complete the Systems Manager walkthroughs in a test environment. These walkthroughs describe how to configure roles and permissions and use Systems Manager capabilities on an Amazon EC2 instance.

- Maintenance Window Walkthroughs (p. 52)
- Parameter Store Walkthroughs (p. 166)
- Run Command Tutorial

- Inventory Manager Walkthrough (p. 130)
- State Manager Walkthroughs (p. 200)
- Automation Walkthroughs (p. 76)
- Patch Manager Walkthrough (p. 140)

Accessing Systems Manager

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

- **AWS Management 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 Amazon EC2 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 Reference
- Systems Manager AWS CLI Reference
- AWS SDKs
- Amazon EC2 Systems Manager Limits

Setting Up Systems Manager

To get started with Amazon EC2 Systems Manager, verify prerequisites, configure AWS Identity and Access Management (IAM) roles, and install the SSM Agent on managed instances.

Systems Manager Prerequisites

Amazon EC2 Systems Manager includes the following prerequisites.

Requirement	Description
Supported Operating System (Windows)	Instances must run a supported version of Windows Server: Windows Server 2003 through Windows Server 2016, including R2 versions.
Supported Operating System (Linux)	Instances must run a supported version of Linux. **64-Bit and 32-Bit Systems** • Amazon Linux 2014.09, 2014.03 or later • Ubuntu Server 16.04 LTS, 14.04 LTS, or 12.04 LTS • Red Hat Enterprise Linux (RHEL) 6.5 or later • CentOS 6.3 or later **64-Bit Systems Only** • Amazon Linux 2015.09, 2015.03 or later • Red Hat Enterprise Linux (RHEL) 7.x or later • CentOS 7.1 or later
Supported Regions	Systems Manager is available in these regions. For servers and VMs in your hybrid environment, we recommend that you choose the region closest to your data center or computing environment.

Requirement	Description
Roles for Systems Manager	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 Security Roles for Systems Manager (p. 6). 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. 24).
SSM Agent (EC2 Linux instances)	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 SSM Agent on Linux (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.
SSM Agent (EC2 Windows instances)	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 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 SSM Agent on Windows (p. 13).

Requirement	Description
SSM Agent (hybrid environment)	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. 26).
Internet Access	Verify that your EC2 instances have outbound Internet access. Inbound Internet access is not required.
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. 171).
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 Amazon EC2 Systems Manager Limits. To increase limits, go to AWS Support Center and submit a limit increase request form.

Configuring Security Roles for Systems Manager

Amazon EC2 Systems Manager requires an IAM role for EC2 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 as described in this section.

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. 24).

Contents

Configuring Access Using Systems Manager Managed Policies

IAM managed policies for Systems Manager can help you quickly configure access and permissions for Systems Manager users and instances. Managed policies perform the following functions:

- **AmazonEC2RoleforSSM (instance trust policy)**: Enables an instance to communicate with the Systems Manager API.
- **AmazonSSMAutomationRole (service role)**: Provides permissions for EC2 Automation service to execute activities defined within Automation documents.
- **AmazonSSMFullAccess (user trust policy)**: Grants the user access to the Systems Manager API and documents. Assign this policy to administrators and trusted power users.
- **AmazonSSMMaintenanceWindowRole (service role)**: Service role for EC2 Maintenance Windows.
- **AmazonSSMReadOnlyAccess (user trust policy)**: Grants the user access to Systems Manager read-only API actions, such as Get and List.

If you want to create your own custom roles and policies, see Configuring Access Using Custom Roles and Polices (p. 8).

Tasks

Task 1: Create a User Account for Systems Manager

If your IAM user account has administrator access in your VPC, then you have permission to call the Systems Manager API on an instance. If you like, you can create a unique user account specifically for managing instances with Systems Manager. Use the following procedure to create a new user that uses an IAM managed policy for Systems Manager.

To create a user account for Systems Manager

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, *SystemsManagerUserFullAccess* or *SystemsManagerUserReadOnly*).
3. In the **Select AWS access type** section, choose one or both access options. If you choose **AWS Management Console access**, you must also choose passwords options.
4. Choose **Next:Permissions**.
5. In the **Set permissions for** section, choose **Attach existing policies directly**.
6. In the filter field, type AmazonSSM.
7. Choose either the checkbox beside **AmazonSSMFullAccess** or **AmazonSSMReadOnlyAccess**, and then choose **Next:Review**.
8. Verify the details, and then choose **Create**.

Important
If you specified password information for the user, review the password information carefully after the user account is created.

Task 2: Create a Role for Systems Manager Managed Instances

Use the following procedure to create an instance role that enables an instance to communicate with the Systems Manager API. After you create the role, you can assign it to instances as described in Task 3.

To create 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 New Role**.
3. In **Step 1: Set Role Name**, enter a name that identifies this role as a Systems Manager role for managed instances.
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 the **AmazonEC2RoleforSSM** managed policy.
6. In **Step 5: Review**, make a note of the role name. You will specify this role name when you create new instances that you want to manage using Systems Manager.
7. Choose **Create Role**. The system returns you to the **Roles** page.

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 Systems Manager instance role for each instance.

Configuring Access Using Custom Roles and Polices

If you choose not to use Systems Manager managed policies, then use the following procedures to create and configure a custom instance role and user account for Systems Manager.

Important
If you want to use an existing instance role and user account, you must attach the policies shown in this section to the role and the user account. You must also verify that ec2.amazonaws.com is listed in the trust policy for the instance role. For more information, see Task 4: Verify the Trust Policy (p. 12).

Tasks

Task 1: Create a Custom IAM Policy for Systems Manager Managed Instances

The following IAM policy enables managed instances to communicate with the Systems Manager API. You will create the role and attach this policy to that role later in this topic.

To create an IAM policy for Systems Manager managed instances

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**.
3. In the filter field, type AmazonEC2RoleforSSM.
4. Choose the AmazonEC2RoleforSSM policy. The system displays the **Policy Document** for the policy.
5. Copy the contents of the policy document.

   Note
   You can't alter the content of the policy document in the IAM console because it is a managed policy, but you can copy it.
6. In the navigation pane, choose **Policies**.
7. Choose **Create Policy**.
8. Beside **Create Your Own Policy**, choose **Select**.
9. Type a policy name (for example, *SystemsManagerInstance*) and description, and then paste the policy you copied earlier into the **Policy Document** field
10. Change the policy as you want.

    Important
    In the last section of this IAM policy, you can restrict access to the Amazon S3 bucket by specifying an Amazon Resource Name (ARN). For example, you can change the last `"Resource": "*"` item to "Resource": "arn:aws:s3:::*AnS3Bucket*/*
11. Choose **Validate Policy**. Verify that the policy is valid. If you receive an error, verify that you included the opening and closing brackets { }. After the policy is validated, choose **Create Policy**.

Task 2: Create a Custom IAM User Policy

The IAM user policy determines which Systems Manager documents a user can see in the **Document** list. Users can see this list 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 a Systems Manager Document.

Note
You will create a user account and attach this policy to that account later on.

The IAM policy in the following procedure enables the user to perform any Systems Manager action on the instance. Assign this policy only to trusted administrators. For all other users, create a restrictive IAM policy, as described in this section, or use the AmazonSSMReadOnlyAccess policy.

To create an IAM user policy for Systems Manager

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**.
3. In the filter field, type AmazonSSMFullAccess.
4. Choose the AmazonSSMFullAccess policy. The system displays the **Policy Document** for the policy.
5. Copy the contents of the policy document.

Note
You can't alter the content of the policy document in the IAM console because it is a managed policy, but you can copy it.

6. In the navigation pane, choose **Policies**.
7. Choose **Create Policy**.
8. Beside **Create Your Own Policy**, choose **Select**.
9. Type a policy name (for example, *SystemsManagerUserFull*) and description, and then paste the policy you copied earlier into the **Policy Document** field
10. Change the policy as you want.
11. Choose **Validate Policy**. Verify that the policy is valid. If you receive an error, verify that you included the opening and closing brackets { }. After the policy is validated, choose **Create Policy**.

Create a Restrictive IAM User Policy

Create restrictive IAM user policies to further delegate access to Systems Manager. 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:

  ```
  arn:aws:ssm:us-east-1:*:document/name_of_restrictive_document
  ```
- 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-
```

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

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "ssm:ListDocuments",
            "ssm:ListDocumentsVersions",
            "ssm:DescribeDocument",
            "ssm:GetDocument",
            "ssm:DescribeInstanceInformation",
            "ssm:DescribeDocumentParameters",
   "ssm:DescribeInstanceProperties"
         ],
         "Effect":"Allow",
         "Resource":"*"
      },
      {
         "Action":"ssm:SendCommand",
         "Effect":"Allow",
         "Resource": [
                     "arn:aws:ec2:us-east-1:*:instance/i-1234567890abcdef0",
                     "arn:aws:ec2:us-east-1:*:instance/i-0598c7d356eba48d7",
                     "arn:aws:ec2:us-east-1:*:instance/i-345678abcdef12345",
                     "arn:aws:s3:::bucket_name",
                     "arn:aws:ssm:us-east-1:*:document/name_of_restrictive_document"
         ]
      },
      {
         "Action":[
            "ssm:CancelCommand",
            "ssm:ListCommands",
            "ssm:ListCommandInvocations"
         ],
         "Effect":"Allow",
         "Resource":"*"
      },
      {
         "Action":"ec2:DescribeInstanceStatus",
         "Effect":"Allow",
         "Resource":"*"
      },
      {
         "Action":"ssm:StartAutomationExecution",
         "Effect":"Allow",
         "Resource":[
            "arn:aws:ssm:::automation-definition/"
         ]
      },
      {
         "Action":"ssm:DescribeAutomationExecutions ",
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      },
      {
         "Action":[
            "ssm:StopAutomationExecution",
            "ssm:GetAutomationExecution"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:ssm:::automation-execution/"
         ]
      }
   ]
}
```

For more information about creating IAM user policies, see Managed Policies and Inline Policies.

Task 3: Create a Role for Systems Manager Managed Instances

The instance role enables the instance to communicate with the Systems Manager API. The role uses the instance policy you created earlier.

To create a role for Systems Manager managed instances

1. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create New Role**.
2. On the **Set Role Name** page, enter a name for the role that designates it as the instance role, for example, *SystemsManagerInstance*. Choose **Next Step**.
3. On the **Select Role Type** page, choose **Select** next to **Amazon EC2**.
4. On the **Attach Policy** page, select the instance policy you created in Task 1. Choose **Next Step**.
5. Review the role information and then choose **Create Role**.

Task 4: Verify the Trust Policy

If you want to use an existing EC2 instance role, you must verify that ec2.amazonaws.com is listed in the trust policy for the role. If you created a new role, you must add ec2.amazonaws.com as a trusted entity.

To verify the trust policy

1. In the navigation pane of the IAM console, choose **Roles**, and then choose the server role you just created.
2. Choose **Trust Relationships**.
3. Under **Trusted Entities**, choose **Edit Trust Relationship**.
4. Copy and paste the following policy into the **Policy Document** field and create the policy:

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

Task 5: Create the User Account

The user account enables a user to call the Systems Manager API on an instance. This account uses the IAM user policy you created earlier.

To create a user account for Systems Manager

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, *SystemsManagerUserFullAccess*).
3. In the **Select AWS access type** section, choose one or both access options. If you choose **AWS Management Console access**, you must also choose passwords options.
4. Choose **Next:Permissions**.
5. In the **Set permissions for** section, choose **Attach existing policies directly**.
6. In the filter field, type the name of the user policy you created earlier.
7. Choose the checkbox beside the policy, and then choose **Next:Review**.
8. Verify the details, and then choose **Create**.

Create an Amazon EC2 instance that uses the custom instance role you created. For more information, see Task 3: Create an Amazon EC2 Instance that Uses the Systems Manager Role (p. 8).

Installing SSM Agent

The Amazon EC2 Systems Manager (SSM) agent processes Systems Manager requests and configures your machine as specified in the request.

SSM communicates with the SSM Agent on your instance by using the EC2 Messaging service. If you monitor traffic, you will see your instances communicating with ec2messages.* endpoints.

Use the following procedures to install, configure, or uninstall the SSM agent.

Contents

Installing SSM Agent on Windows

An agent running on your instances processes Systems Manager requests and configures your instances as specified in the request. The 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 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).

Note

The SSM Agent download and installation process for managed 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. 26).

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

Installing SSM Agent

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 remotely update EC2Config and install SSM Agent on one or more instances by using Run Command. For more information, see Executing Commands from the EC2 Console (p. 181).

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 the SSM Agent to your instance.

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

   This URL lets you download the SSM agent from any AWS region. If you want to download the agent from a specific region, use a region-specific URL instead:

   ```
   https://amazon-ssm-region.s3.amazonaws.com/latest/windows_amd64/AmazonSSMAgentSetup.exe
   ```
3. Start or restart the SSM agent (AmazonSSMAgent.exe) using the Windows Services control panel or by sending the following command in PowerShell:

   ```
   Restart-Service AmazonSSMAgent
   ```

SSM Agent Logs

On Windows, SSM Agent stores log files in the following locations.

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

Configuring SSM Agent to Use a Proxy

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. If you connected using Remote Desktop, then launch PowerShell as an administrator.
3. Run the following command block in PowerShell:

   ```
   $serviceKey = "HKLM:\SYSTEM\CurrentControlSet\Services\AmazonSSMAgent"
   $proxyVariables = @("http_proxy=hostname:port", "no_proxy=169.254.169.254")
   New-ItemProperty -Path $serviceKey -Name Environment -Value $proxyVariables -
   PropertyType MultiString
   Restart-Service AmazonSSMAgent
   ```

To reset the 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:

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

Installing SSM Agent on Linux

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 the SSM agent.

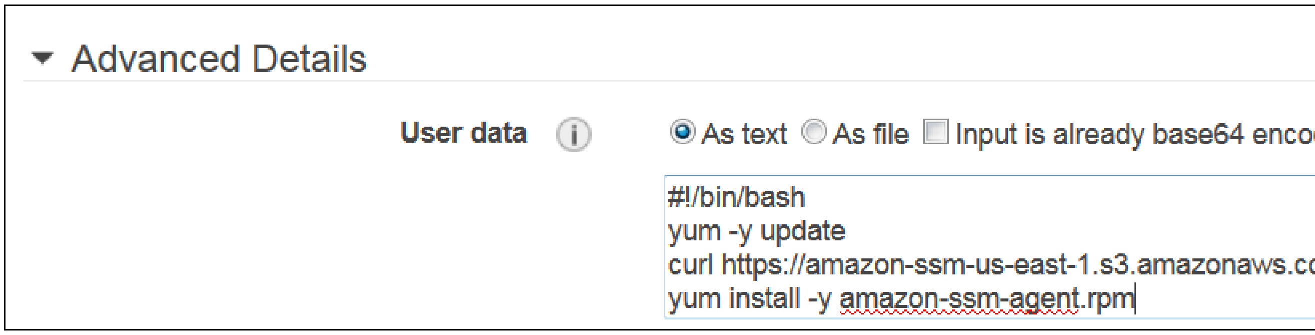
The source code for the SSM agent is available on GitHub so that you can adapt the agent to meet your needs. We encourage you to submit pull requests for changes that you would like to have included. However, AWS does not currently provide support for running modified copies of this software.

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

Contents

Install SSM Agent on EC2 Instances at Launch

You can install the SSM agent when you launch an instance for the first time by using 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:



```
#!/bin/bash
yum -y update
curl https://amazon-ssm-us-east-1.s3.amazonaws.co
yum install -y amazon-ssm-agent.rpm
```

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, replace the URL below with one of the following region-specific URLs. You must specify a region where Systems Manager is available:
Amazon Linux, RHEL, and CentOS 64-bit:

https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/linux_amd64/amazon-ssm-agent.rpm
Amazon Linux, RHEL, and CentOS 32-bit:
https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/linux_386/amazon-ssm-agent.rpm
Ubuntu Server 64-bit:
https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/debian_amd64/amazon-ssm-agent.deb
Ubuntu Server 32-bit:
https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/debian_386/amazon-ssm-agent.deb

Amazon Linux, RHEL, and CentOS 64-bit

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

Amazon Linux, RHEL, and CentOS 32-bit

```
#!/bin/bash
cd /tmp
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/
linux_386/amazon-ssm-agent.rpm
```

Ubuntu Server 16 64-bit

```
#!/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

```
#!/bin/bash
cd /tmp
wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_386/amazon-ssm-
agent.deb
sudo dpkg -i amazon-ssm-agent.deb
sudo systemctl enable amazon-ssm-agent
```

Ubuntu Server 14 64-bit

```
#!/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

```
#!/bin/bash
cd /tmp
wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_386/amazon-ssm-
agent.deb
```

```
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 the 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. 180).

Manually Install the SSM Agent on EC2 Instances

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

- Amazon Linux (p. 17)
- Ubuntu (p. 18)
- Red Hat Enterprise Linux (p. 19)
- CentOS (p. 20)

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, replace the URL below with one of the following region-specific URLs. You must specify a region where Systems Manager is available:
Amazon Linux, RHEL, and CentOS 64-bit:
https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/linux_amd64/amazon-ssm-agent.rpm
Amazon Linux, RHEL, and CentOS 32-bit:
https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/linux_386/amazon-ssm-agent.rpm
Ubuntu Server 64-bit:
https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/debian_amd64/amazon-ssm-agent.deb
Ubuntu Server 32-bit:
https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/debian_386/amazon-ssm-agent.deb

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.

To install the 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

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

   32-Bit

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

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

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

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

   a. Start the service.

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

   b. Check the status of the agent.

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

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 the SSM agent on Ubuntu

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

```
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

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

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

   Ubuntu Server 14

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

   Ubuntu Server 16

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

4. Execute the following commands if the previous command returned "amazon-ssm-agent is stopped," "inactive," or "disabled.

   a. Start the service.

      Ubuntu Server 14

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

      Ubuntu Server 16

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

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

   b. Check the status of the agent.

      Ubuntu Server 14

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

      Ubuntu Server 16

      ```
      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 the SSM agent. Perform these steps on each instance that will execute commands using Systems Manager.

To install the SSM agent on Red Hat Enterprise Linux

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

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

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

   RHEL 7.x

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

RHEL 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.

      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 the 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

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

3. Run one of the following commands to determine if the 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
      ```

Configure the SSM Agent to Use a Proxy

You can configure the 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 the SSM agent to use a proxy (Upstart)

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

   /etc/init/amazon-ssm-agent.conf
3. Add the `http_proxy` setting to the file in the following format:

   ```
   env http_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:

```
env no_proxy=169.254.169.254
```

5. Save your changes and close the editor.
6. Restart the SSM agent using the following command:

```
sudo restart amazon-ssm-agent
```

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

```
description "Amazon SSM Agent"
author "Amazon.com"

start on (runlevel [345] and started network)
stop on (runlevel [!345] or stopping network)

respawn

env http_proxy=http://i-1234567890abcdef0:443
env no_proxy=169.254.169.254

chdir /usr/bin/
exec ./amazon-ssm-agent
```

To configure the SSM agent to use a proxy (systemd)

1. Connect to the instance where you installed the 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 `http_proxy` setting to the file in the following format:

```
Environment="HTTP_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 the 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
```

SSM Agent Logs

The SSM agent logs information in the following files. The information in these files can help you troubleshoot problems.

- /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: /opt/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 the SSM Agent on Linux

Use the following commands to uninstall the SSM agent.

To uninstall the SSM agent on Amazon Linux, RHEL, or Cent OS

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

To uninstall the SSM agent on Ubuntu

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

Setting Up Systems Manager in Hybrid Environments

Amazon EC2 Systems Manager lets you remotely and securely manage on-premises servers and virtual machines (VMs) and VMs from other cloud providers. By setting up Systems Manager in this way, you do the following.

- Create a consistent and secure way to remotely manage your on-premises and cloud 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.

After you set up your hybrid machines for Systems Manager, they are listed in the EC2 console and called *managed instances*, like other EC2 instances.

To get started using Systems Manager in hybrid environments

1. **Create IAM service and user roles**: The IAM *service* role enables your servers and VMs in your hybrid environment to communicate with the Systems Manager SSM service. The IAM *user* role

enables users to communicate with the SSM API to execute commands from either the Amazon EC2 console or by directly calling the API.

2. **Verify prerequisites**: Verify that your servers and VMs in your hybrid environment meet the minimum requirements for Systems Manager. For more information, see Systems Manager Prerequisites (p. 4).
3. **Create a managed-instance activation**: A managed-instance activation registers one or more servers and VMs in your hybrid environment with Systems Manager.
4. **Deploy SSM Agent**: SSM Agent processes Systems Manager requests and configures your machine as specified in the request. You must download and install the SSM Agent on servers and VMs in your hybrid environment.

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.

```
{
  "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.

```
{
  "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.

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

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

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

You must now create IAM roles that enable users to communicate with the SSM API. For more information, see Configuring Security Roles for Systems Manager (p. 6).

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.

To create a managed-instance activation using the console

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 the 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.

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

   For example:

   ```
   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.

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

   For example:

   ```
   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 instance, see Installing SSM Agent on Windows (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.

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

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

  https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/linux_amd64/amazon-ssm-agent.rpm
- **Amazon Linux, RHEL, and CentOS 32-bit**

  https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/linux_386/amazon-ssm-agent.rpm
- **Ubuntu Server 64-bit**

  https://s3.*region*.amazonaws.com/amazon-ssm-*region*/latest/debian_amd64/amazon-ssm-agent.deb
- **Ubuntu Server 32-bit**

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

And then replace *region* with a region where SSM is available.

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

   On RHEL 7.x and CentOS 7.x

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

   On Ubuntu

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

3. Press Enter.

The command downloads and installs the SSM agent onto the server or VM in your hybrid environment. The command stops the SSM agent and then 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.

For information about how to execute commands on managed instances using Run Command, see Executing Commands Using Systems Manager Run Command (p. 180).

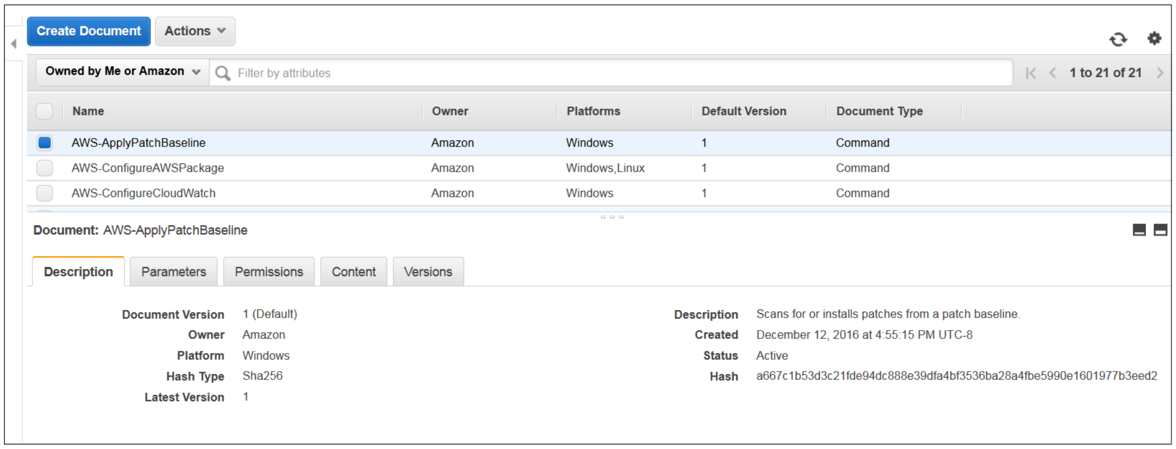
Systems Manager Documents

An Amazon EC2 Systems Manager 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), and they include steps and parameters that you specify. Steps execute in sequential order.

Type	Use with	Details
Command document	Run Command State Manager	Run Command uses command documents to execute commands. State Manager uses command documents to apply a policy. These actions can be run on one or more targets at any point during the lifecycle of an instance,
Policy document	State Manager	Policy documents enforce a policy on your targets. If the policy document is removed, the policy (for example, collecting inventory) no longer happens.
Automation document	Automation	Use automation documents when performing common maintenance and deployment tasks such as creating or updating an Amazon Machine Image (AMI).

Systems Manager Pre-Defined Documents

To help you get started quickly, Systems Manager provides pre-defined documents. You can view these documents in the Amazon EC2 console. In the 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, as shown in the following image.



Name	Owner	Platforms	Default Version	Document Type
AWS-ApplyPatchBaseline	Amazon	Windows	1	Command
AWS-ConfigureAWSPackage	Amazon	Windows,Linux	1	Command
AWS-ConfigureCloudWatch	Amazon	Windows	1	Command

Document: AWS-ApplyPatchBaseline

Document Version	1 (Default)	Description	Scans for or installs patches from a patch baseline.
Owner	Amazon	Created	December 12, 2016 at 4:55:15 PM UTC-8
Platform	Windows	Status	Active
Hash Type	Sha256	Hash	a667c1b53d3c21fde94dc888e39dfa4bf3536ba28a4fbe5990e1601977b3eed2
Latest Version	1		

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.

AWS CLI

```
aws ssm list-documents
```

```
aws ssm describe-document --name "document_name"
```

Tools for Windows PowerShell

```
Get-SSMDocumentList
```

```
Get-SSMDocumentDescription -Name "document_name"
```

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. 35).

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. 37).

Document Schemas and Features

Systems Manager documents currently use the following schema versions.

- Documents of type `Command` can use schema version 1.2 or 2.0. If you are currently using schema 1.2 documents, we recommend that you create documents that use schema version 2.0.

- Documents of type `Policy` must use schema version 2.0.
- Documents of type `Automation` must use schema version 0.3.

By using the latest schema versions for each document type, you can take advantage of the following features.

Schema Version 2.0 Document Features

Feature	Details
Document editing	Documents can now be updated. With version 1.2, any update to a document requires that you save it with a different name.
Automatic versioning	Any update to a document creates a new version. This is not a schema version, but a version of the document.
Default version	If you have multiple versions of a document, you can specify which version is the default document.
Sequencing	Steps in the document execute in the order that you specified.
Document types	Systems Manager supports `Command`, `Policy`, and `Automation` document types.

Document Examples by Schema Version

The following example shows a document that uses schema version 1.2. In this example, the document includes the `aws:runShellScript` plugin for executing `ifconfig` with Run Command.

Schema 1.2 example

```
{
  "schemaVersion": "1.2",
  "description": "Check ip configuration of a Linux instance.",
  "parameters": {

  },
  "runtimeConfig": {
    "aws:runShellScript": {
      "properties": [
        {
          "id": "0.aws:runShellScript",
          "runCommand": ["ifconfig"]
        }
      ]
    }
  }
}
```

The following example shows a document that uses schema version 2.0. In this example, the document includes the `aws:runShellScript` and `aws:runPowerShellScript` plugins for executing commands with Run Command.

Schema 2.0 example

```
{
   "schemaVersion":"2.0",
   "description":"Run a script",
   "parameters":{
      "commands":{
         "type":"StringList",
         "description":"(Required) Specify a shell script or a command to run.",
         "minItems":1,
         "displayType":"textarea"
      }
   },
   "mainSteps":[
      {
         "action":"aws:runShellScript",
         "name":"runShellScript",
         "inputs":{
            "runCommand":"{{ commands }}"
         }
      },
      {
         "action":"aws:runPowerShellScript",
         "name":"runPowerShellScript",
         "inputs":{
            "runCommand":"{{ commands }}"
         }
      }
   ]
}
```

The following sample shows a basic policy document that uses the `aws:runPowerShellScript` plugin to get information about a process. A policy document can have multiple steps.

Schema 2.0 example

```
{
   "schemaVersion": "2.0",
   "description": "Sample version 2.0 document v2",
   "parameters": {

   },
   "mainSteps": [
      {
         "action": "aws:runPowerShellScript",
         "name": "runPowerShellScript",
         "inputs": {
            "runCommand": [
               "Get-Process"
            ]
         }
      }
   ]
}
```

The following sample includes multiple actions.

Schema 2.0 example

```
{
 "schemaVersion": "2.0",
 "description": "Sample version 2.0 document v2 to install application and run a command",
 "parameters": {
  "action": {
   "type": "String",
   "default": "Install",
   "description": "(Optional) The type of action to perform. Valid values: Install | Repair
 | Uninstall",
   "allowedValues": [
    "Install",
    "Repair",
    "Uninstall"
   ]
  },
  "parameters": {
   "type": "String",
   "default": "",
   "description": "(Optional) The parameters for the installer."
  },
  "source": {
   "type": "String",
   "description": "(Required) The URL or local path on the instance to the application .msi
 file."
  },
  "sourceHash": {
   "type": "String",
   "default": "",
   "description": "(Optional) The SHA256 hash of the .msi file."
  },
  "commands": {
   "type": "StringList",
   "description": "(Required) Specify a shell script or a command to run.",
   "minItems": 1,
   "displayType": "textarea"
  },
  "workingDirectory": {
   "type": "String",
   "default": "",
   "description": "(Optional) The path to the working directory on your instance.",
   "maxChars": 4096
  },
  "executionTimeout": {
   "type": "String",
   "default": "3600",
   "description": "(Optional) The time in seconds for a command to complete before it is
 considered to have failed. Default is 3600 (1 hour). Maximum is 28800 (8 hours).",
   "allowedPattern": "([1-9][0-9]{0,3})|(1[0-9]{1,4})|(2[0-7][0-9]{1,3})|(28[0-7][0-9]
{1,2})|(28800)"
  }
 },
 "mainSteps": [
  {
   "action": "aws:applications",
   "name": "installApplication",
   "inputs": {
    "action": "{{ action }}",
    "parameters": "{{ parameters }}",
    "source": "{{ source }}",
    "sourceHash": "{{ sourceHash }}"
   }
  },
  {
   "action": "aws:runPowerShellScript",
   "name": "runPowerShellScript",
   "inputs": {
    "runCommand": "{{ commands }}",
    "workingDirectory": "{{ workingDirectory }}",
    "timeoutSeconds": "{{ executionTimeout }}"
   }
  }
 ]
}
```

The following table lists the differences between versions.

Version 1.2	Version 2.0	Details
runtimeConfig	mainSteps	In version 2.0, the mainSteps section replaces runtimeConfig. The mainSteps section enables Systems Manager to execute steps in sequence.
properties	inputs	In version 2.0, the inputs section replaces the properties section. The inputs section accepts parameters for steps.
commands	runCommand	In version 2.0, the inputs section takes the `runCommand` parameter instead of the `commands` parameter.
id	action	Action replaces ID in version 2.0. This is just a name change.
not applicable	name	Name is any user-defined name for a step.

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. If you change the default version of a State Manager Policy or Command 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.

When you change the JSON content of a document, Systems Manager automatically increments the version of the document. You can retrieve and view previous versions of the document. State Manager Policy or Command documents can be associated with either instances or tagged groups.

Also note the following details about policy documents.

- You can assign multiple documents to a target by creating different associations that use different policy documents.
- If you associate multiple documents to a target, you can use the AWS CLI or SDK to view a consolidated list of plugins that will be executed across all associated documents.
- The order in which steps are specified in a document is the order in which they will be executed.
- 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.

- 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.

Limitations

As you begin working with Systems Manager documents, be aware of the following limitations.

- By default, you can create a maximum of 200 documents per AWS account per region.
- Systems Manager documents that you create are only available in the region where you created them. To add a document in another region, copy the content and recreate it in the new region.
- Each document can store up to 1,000 versions.

For information about Systems Manager limits, see Amazon EC2 Systems Manager Limits. To increase limits, go to AWS Support Center and submit a limit increase request form.

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 2.0 or later. A Systems Manager document contains the following sections.

- schemaVersion: the schema version to use.
- Description: Information you provide to describe the purpose of the document.
- Parameters: The parameters the document accepts, for example command. To easily reference parameters you use often, use Parameter Store parameters in the following format: {{ssm:parameter_name}}. For more information, see Systems Manager Parameter Store (p. 159).
- mainSteps: An object that can include multiple steps (plugins). Steps include one or more actions, a unique name of the action, and inputs (parameters) for those actions. For more information, see SSM Plugins in the *Amazon EC2 Systems Manager API Reference*.

The following example shows the Systems Manager required sections.

```
{
   "schemaVersion":"2.0",
   "description":"Run a script",
   "parameters":{
      "commands":{
         "type":"StringList",
         "description":"(Required) Specify a shell script or a command to run.",
         "minItems":1,
         "displayType":"textarea"
      }
   },
   "mainSteps":[
      {
         "action":"aws:runShellScript",
         "name":"runShellScript",
         "inputs":{
            "runCommand":"{{ commands }}"
         }
      },
      {
         "action":"aws:runPowerShellScript",
         "name":"runPowerShellScript",
         "inputs":{
            "runCommand":"{{ commands }}"
         }
      }
   ]
}
```

Note
You currently can't use the same plugin twice in a policy document.

When you create a document, you specify the contents of the document in JSON. The easiest way to get started with the JSON is to copy an existing sample from one of the Systems Manager public documents.

To copy a Systems Manager document

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 to Systems Manager using any one of the following procedures.

Note
If you author a policy document, you must associate the document with your managed instances after you add it the system. For more information, see State Manager Associations (p. 199).

Add a Systems Manager Document Using the Amazon EC2 Console

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. 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 Using Windows PowerShell

1. Copy and customize an existing document, as described earlier.
2. Add the document using the AWS Tools for Windows PowerShell.

```
$json = Get-Content C:\your file | Out-String
New-SSMDocument -Name document name -Content $json
```

Add a Systems Manager Document Using the AWS CLI

1. Copy and customize an existing document, as described earlier.
2. Add the document using the AWS CLI.

```
aws ssm create-document --content file://path to your file\your file --name "document
 name" --document-type "Command"
```

Windows example

```
aws ssm create-document --content file://c:\temp\PowershellScript.json --name
 "PowerShellScript" --document-type "Command"
```

Linux example

```
aws ssm create-document --content file:///home/ec2-user/RunShellScript.json  --name
 "RunShellScript" --document-type "Command"
```

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. 38).

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 How to Modify Permissions for a Shared Document (p. 40).
- 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 Amazon EC2 Systems Manager Limits.

Contents

- How to Use a Shared Systems Manager Document (p. 41)

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 Security Roles for Systems Manager (p. 6).

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.

How to Share a Systems Manager Document

You can share Systems Manager document by using the Amazon EC2 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.

Share a Document Using the Amazon EC2 Console

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 Using the 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.

```
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.

```
aws ssm list-documents --document-filter-list key=Owner,value=all
```

3. Use the following command to get a specific document.

```
aws ssm get-document --name document name
```

4. Use the following command to get a description of the document.

```
aws ssm describe-document --name document name
```

5. Use the following command to view the permissions for the document.

```
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.

```
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.

```
aws ssm modify-document-permission --name document name --permission-type Share --
account-ids-to-add 'all'
```

Share a Document Using 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.

```
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.

```
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.

```
Get-SSMDocumentList -DocumentFilterList (@{"key"="Owner";"value"="All"})
```

4. Use the following command to get a specific document.

```
Get-SSMDocument –Name document name
```

5. Use the following command to get a description of the document.

```
Get-SSMDocumentDescription –Name document name
```

6. Use the following command to view the permissions of the document.

```
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.

```
Edit-SSMDocumentPermission –Name document name -PermissionType Share -
AccountIdsToAdd AWS account ID
```

   Use the following command to share a document publicly.

```
Edit-SSMDocumentPermission -Name document name -AccountIdsToAdd ('all') -PermissionType
 Share
```

How to 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.

Stop Sharing a Document Using the Amazon EC2 Console

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 Using the AWS CLI

Open the AWS CLI on your local computer and execute the following command to stop sharing a command.

```
aws ssm modify-document-permission --name document name --permission-type Share --account-
ids-to-remove 'AWS account ID'
```

Stop Sharing a Document Using 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.

```
Edit-SSMDocumentPermission -Name document name –AccountIdsToRemove AWS account ID -
PermissionType Share
```

How to 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.

This section includes examples of how to view and execute shared Systems Manager documents from the AWS CLI and AWS Tools for Windows PowerShell.

Using a Shared Systems Manager Document from the AWS CLI

To list all public Systems Manager documents

```
aws ssm list-documents --document-filter-list key=Owner,value=Public
```

To list private Systems Manager documents that have been shared with you

```
aws ssm list-documents --document-filter-list key=Owner,value=Private
```

To list all Systems Manager documents available to you

```
aws ssm list-documents --document-filter-list key=Owner,value=All
```

Execute a command from a shared Systems Manager document using a full ARN

```
aws ssm send-command --document-name FullARN/name
```

For example:

```
aws ssm send-command --document-name arn:aws:ssm:us-east-1:12345678912:document/
highAvailabilityServerSetup --instance-ids i-12121212
```

Using a Shared Systems Manager Document from the AWS Tools for Windows PowerShell

To list all public Systems Manager documents

```
Get-SSMDocumentList -DocumentFilterList @(New-Object
 Amazon.SimpleSystemsManagement.Model.DocumentFilter("Owner", "Public"))
```

To list private Systems Manager documents that have been shared with you

```
Get-SSMDocumentList -DocumentFilterList @(New-Object
 Amazon.SimpleSystemsManagement.Model.DocumentFilter("Owner", "Shared"))
```

To get information about a Systems Manager document that has been shared with you

```
Get-SSMDocument –Name FullARN/name
```

For example:

```
Get-SSMDocument –Name arn:aws:ssm:us-east-1:12345678912:document/
highAvailabilityServerSetup
```

To get a description of a Systems Manager document that has been shared with you

```
Get-SSMDocumentDescription –Name FullARN/name
```

For example:

```
Get-SSMDocumentDescription –Name arn:aws:ssm:us-east-1:12345678912:document/
highAvailabilityServerSetup
```

To execute a command from a shared Systems Manager document using a full ARN

```
Send-SSMCommand –DocumentName FullARN/name –InstanceId IDs
```

For example:

```
Send-SSMCommand –DocumentName arn:aws:ssm:us-east-1:555450671542:document/
highAvailabilityServerSetup –InstanceId @{"i-273d4e9e"}
```

Systems Manager Maintenance Windows

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. Each task is defined to run for a subset of the registered targets. Currently, you can perform tasks like installing applications, installing or updating SSM Agent, executing commands with Run Command, or installing patches (Windows).

Contents

Creating a Maintenance Window

Creating a Maintenance Window requires that you complete the following tasks:

- Create one or more SSM command documents that define the tasks to perform on your instances during the Maintenance Window. For information about how to create an SSM command document, see Creating Systems Manager Documents (p. 35).
- Create the Maintenance Window and define its schedule.
- Register targets for the Maintenance Window. Targets can either be instance IDs or EC2 tags.
- Register one or more tasks (SSM command documents) with the Maintenance Window.

After you complete these tasks, the Maintenance Window runs according to the schedule you defined and executes the tasks in your SSM documents on the targets you specified. After a task completes, Systems Manager logs the details of the execution.

Before you create a Maintenance Window, you must configure a Maintenance Window role with an Amazon Resource Name (ARN). For more information, see Configuring Roles and Permissions for Maintenance Windows (p. 46).

You can create a Maintenance Window using the **Maintenance Window** page in the Amazon EC2 console, the AWS CLI, the Systems Manager API, or the AWS SDKs. For examples of how to create a Maintenance Window, see the Systems Manager Maintenance Window Walkthroughs (p. 52).

Cron Schedules for Systems Manager

When you create a Systems Manager Maintenance Window or an association using Systems Manager State Manager, you specify a schedule for when the window/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*.

Example: This cron expression runs the Maintenance Window or the association at 4 PM (16:00) every Tuesday: `cron(0 16 ? * TUE *)`

In the AWS CLI, specify this expression using the --schedule parameter as follows:

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

Example: This rate expression runs the Maintenance Window or the association every other day: `rate(2 days)`

In the AWS CLI, specify this expression using the --schedule parameter as follows:

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

Cron expressions have six required fields. Fields are separated by white space.

Minutes	Hours	Day of month	Month	Day of week	Year	Meaning
0	10	*	*	?	*	Run at 10:00 am (UTC) every day
15	12	*	*	?	*	Run at 12:15 PM (UTC) every day
0	18	?	*	MON-FRI	*	Run at 6:00 PM (UTC) every Monday through Friday
0	8	1	*	?	*	Run at 8:00 AM (UTC) every 1st day of the month
0/15	*	*	*	?	*	Run every 15 minutes
0/10	*	?	*	MON-FRI	*	Run every 10 minutes

Minutes	Hours	Day of month	Month	Day of week	Year	Meaning
						Monday through Friday
0/5	8-17	?	*	MON-FRI	*	Run every 5 minutes Monday through Friday between 8:00 AM and 5:55 PM (UTC)

The following table shows more examples of cron expressions:

Cron Expression Example	Runs At
0 0 2 ? 1/1 THU#3 *	02:00 AM the third Thursday of every month
0 15 10 ? * *	10:15 AM every day
0 0 0 21 1/1 ? *	midnight on the 21st of each month
0 15 10 ? * MON-FRI	10:15 AM every Monday, Tuesday, Wednesday, Thursday and Friday
0 0 2 L * ?	02:00 AM on the last day of every month
0 15 10 ? * 6L	10:15 AM on the last Friday of every month

The following table shows supported values for required cron entries:

Field	Values	Wildcards
Minutes	0-59	, - * /
Hours	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /
Day-of-week	1-7 or SUN-SAT	, - * ? / L
Year	1970-2199	, - * /

Note
You 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*.

Rate Expressions

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

Field	Values
Value	positive number
Unit	minute(s) OR hour(s) OR day(s)

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.

Configuring Roles and Permissions for Maintenance Windows

Use one of the following methods to configure security roles and permissions for Maintenance Windows. After you configure roles and permissions, you can perform a test run with Maintenance Windows as described in Systems Manager Maintenance Window Walkthroughs (p. 52).

Topics

Controlling Access to Maintenance Windows Using the AWS Console

The following procedures describe how to create the required roles and permissions for Maintenance Windows 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 New Role**.
3. In **Step 1: Select Role Type**, choose **Amazon EC2**. The system skips **Step 2: Establish Trust** because this is a managed policy.
4. In **Step 3: Attach Policy**, choose **AmazonSSMMaintenanceWindowRole**, and then choose **Next Step**.
5. In **Step 4: Set role name and review**, 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. Locate the role you just created and double-click it.
8. Choose the **Trust Relationships** tab, and then choose **Edit Trust Relationship**.
9. Add a comma after "ec2.amazonaws.com"**,** and then add "Service": "ssm.amazonaws.com" to the existing policy as the following code snippet illustrates:

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

10. Choose **Update Trust Policy**.
11. Copy or make a note of the role name and the **Role ARN**. You will specify this information when you create your Maintenance Window.

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 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.
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 Using the 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.

```
{
   "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.

```
{
   "Role":{
      "AssumeRolePolicyDocument":{
         "Version":"2012-10-17",
         "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 `AmazonSSMMaintenanceWindowRole` 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/AmazonSSMMaintenanceWindowRole
```

Assign the IAM PassRole Policy to an IAM User Account Using the 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.

```
{
   "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 Amazon EC2 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 Using 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.

```
{
   "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
CreateDate : 4/4/2017 11:24:43
Path : /
RoleId : AROAIIZKPBKS2LEXAMPLE
RoleName : mw-task-role
```

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 Using 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.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"Stmt1491345526000",
         "Effect":"Allow",
         "Action":[
            "iam:GetRole",
            "iam:PassRole",
            "ssm:RegisterTaskWithMaintenanceWindow"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

2. Open Tools for Windows PowerShell.
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.

```
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 Amazon EC2 console, you must also assign the AmazonSSMReadOnlyAccess policy to your user account. Execute the following command to assign this policy to your account.

```
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
```

Systems Manager Maintenance Window Walkthroughs

Use the following walkthroughs to create and run a Maintenance Window in a test environment. Before you use these walkthroughs, you must configure Maintenance Window roles and permissions. For more information, see Configuring Roles and Permissions for Maintenance Windows (p. 46).

Contents

Launch a New Instance

Use the following procedure to create a test instance with the required AWS Identity and Access Management (IAM) role. The role enables the instance to communicate with the Systems Manager API.

To create an instance that uses a Systems Manager-supported 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 Amazon Machine Image (AMI).
4. Choose your instance type and then choose **Next: Configure Instance Details**.
5. In **Auto-assign Public IP**, choose **Enable**.
6. Beside **IAM role** choose **Create new IAM role**. The IAM console opens in a new tab.

   a. Choose **Create New Role**.
   b. In **Step 1: Set Role Name**, enter a name that identifies this role as a Systems Manager role.
   c. In **Step 2: Select Role Type**, choose **Amazon EC2 Role for Simple Systems Manager**. The system skips **Step 3: Establish Trust** because this is a managed policy.
   d. In **Step 4: Attach Policy**, choose **AmazonEC2RoleforSSM**.
   e. Choose **Next Step**, and then choose **Create Role**.
   f. Close the tab with the IAM console.
7. In the Amazon EC2 console, choose the **Refresh** button beside **Create New IAM role**.
8. From **IAM role**, choose the role you just created.
9. Complete the wizard to launch the new instance. Make a note of the instance ID. You will need to specify this ID later in this walkthrough.

Important
On Linux instances, you must install the SSM Agent on the instance you just created. For more information, see Installing SSM Agent on Linux (p. 15).

To assign the role to one of your existing instances, see Attaching an IAM Role to an Instance in the *Amazon EC2 User Guide*.

Maintenance Window Console Walkthrough

The following walkthrough introduces you to Maintenance Windows concepts and walks you through the process of creating and configuring a maintenance window using the Amazon EC2 console. You'll configure the Maintenance Window to run on a test instance that is configured for Systems Manager. After you finish the walkthrough, you can delete the test instance.

To create 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 **Create a Maintenance Window**.
3. For **Name**, type a descriptive name to help you identify this Maintenance Window as a test Maintenance Window.
4. **Allow unregistered targets**: This option is not selected by default, which means any managed instance can execute a Maintenance Window task as long as the instance is targeted using its instance ID. Targets defined by tags must be registered.
5. Specify a schedule for the Maintenance Window using either the schedule builder or by specifying a schedule in cron format. For more information about cron format, see Cron Schedules for Systems Manager (p. 44).
6. In the **Duration** field, type the number of hours the Maintenance Window should run.
7. 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.
8. Choose **Create maintenance window**. The system returns you to the Maintenance Window page. The state of the Maintenance Window you just created is **Enabled**.

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. From the **Actions** list, choose **Register targets**.
3. In the **Owner information** field, specify your name or work alias.
4. In the **Select targets by** section, choose **Specifying instances**.
5. Choose the instance you created at the start of this walkthrough.
6. Choose **Register targets**.

The tasks you specified run on the targets you selected according to the Maintenance Window you defined when you created the window.

After you assign targets, you assign tasks to perform during the window.

To assign tasks to a Maintenance Window

1. In the Maintenance Window list, choose the Maintenance Window you just created.
2. From the **Actions**list, choose **Register task**.

3. From the **Document** list, choose the SSM command document that defines the task(s) to run. For more information about creating SSM command documents, see Creating Systems Manager Documents (p. 35).
4. 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.
5. In the **Target by** section, choose **Selecting unregistered targets**, and then choose the instance you created at the start of this walkthrough.
6. In the **Parameters** section, specify parameters for the SSM command document.
7. In the **Role** field, specify the Maintenance Windows ARN. For more information about creating a Maintenance Windows ARN, see Configuring Roles and Permissions for Maintenance Windows (p. 46).
8. 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. For the purposes of this walkthrough, specify 1.
9. In the **Stop after** field, specify the number of allowed errors before the system stops sending the task to new instances.
10. Choose **Register task**.

Maintenance Window CLI Walkthrough

The following walkthrough introduces you to Maintenance Windows concepts and walks you through the process of creating and configuring a Maintenance Window using the AWS CLI. You'll perform the walkthrough on a test instance that is configured for Systems Manager. After you finish the walkthrough, you can delete the test instance.

Creating and Configuring a Maintenance Window Using the 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 Cron Schedules for Systems Manager (p. 44).

```
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.

```
{
   "WindowId":"mw-ab12cd34ef56gh78"
}
```

3. Execute the following command to list all Maintenance Windows in your AWS account.

```
aws ssm describe-maintenance-windows
```

   The system returns information like the following.

```
{
   "WindowIdentities":[
      {
         "Duration":4,
         "Cutoff":1,
         "WindowId":"mw-ab12cd34ef56gh78",
         "Enabled":true,
         "Name":"My-First-Maintenance-Window"
      }
   ]
}
```

4. Execute the following command to register the instance you created earlier as a target for this Maintenance Windows. 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-ab12cd34ef56gh78"
```

The system returns information like the following.

```
{
   "Targets":[
      {
         "ResourceType":"INSTANCE",
         "OwnerInformation":"Single instance",
         "WindowId":"mw-ab12cd34ef56gh78",
         "Targets":[
            {
               "Values":[
                  "i-11aa22bb33cc44dd5"
               ],
               "Key":"InstanceIds"
            }
         ],
         "WindowTargetId":"a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4"
      },
      {
         "ResourceType":"INSTANCE",
         "OwnerInformation":"Two instances in a list",
         "WindowId":"mw-ab12cd34ef56gh78",
         "Targets":[
            {
               "Values":[
                  "i-1a2b3c4d5e6f7g8h9",
                  "i-aa11bb22cc33dd44e "
               ],
               "Key":"InstanceIds"
            }
         ],
         "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
      },
      {
         "ResourceType":"INSTANCE",
         "OwnerInformation":"Production Web Servers",
         "WindowId":"mw-ab12cd34ef56gh78",
         "Targets":[
            {
               "Values":[
                  "Prod"
               ],
               "Key":"tag:Environment"
            },
            {
               "Values":[
                  "Web"
               ],
               "Key":"tag:Role"
            }
         ],
         "WindowTargetId":"1111aaa-2222-3333-4444-1111aaa "
      }
   ]
}
```

6. Execute the following command to register a task on the instance you created earlier. This task uses Systems Manager Run Command to execute the `df` command using the AWS-RunShellScript document. This command uses the following parameters:

   - `targets`: Specify either `Key=`*`WindowTargetIds`*`,Values=`*`Window Target ID`* to specify a target registered with the Maintenance Window or `Key=`*`InstanceIds`*`,Values=`*`Instance ID`* to specify individual instances registered with the Maintenance Window.
   - `task-arn`: Specify the name of a Systems Manager Run Command document. For example: AWS-RunShellScript, AWS-RunPowerShellScript, or arn:aws:ssm:us-east-1:123456789:document/Restart_Apache (for a shared document).
   - `window-id`: Specify the ID of the target Maintenance Window.
   - `task-type`: Specify `RUN_COMMAND`. Currently only Run Command tasks are supported.
   - `task-parameters`: Specify required and optional parameters for the Run Command document.
   - `max-concurrency`: (Optional) Specify 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) Specify 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`: Specify 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.

```
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-
parameters "{\"commands\":{\"Values\":[\"df\"]}}" --max-concurrency 1 --max-errors 1 --
priority 10
```

The system returns information like the following.

```
{
   "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.

```
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-parameters  "{\"commands\":{\"Values
\":[\"df\"]}}" --max-concurrency 1 --max-errors 1 --priority 10
```

The system returns information like the following.

```
{
   "WindowTaskId":"44444444-5555-6666-7777-88888888"
}
```

7. Execute the following command to list all registered tasks for a Maintenance Window.

```
aws ssm describe-maintenance-window-tasks --window-id "mw-ab12cd34ef56gh78"
```

The system returns information like the following.

```
{
   "Tasks":[
      {
         "ServiceRoleArn":"arn:aws:iam::11111111:role/MW-Role",
         "MaxErrors":"1",
         "TaskArn":"AWS-RunPowerShellScript",
         "MaxConcurrency":"1",
         "WindowTaskId":"3333-3333-3333-333333",
         "TaskParameters":{
            "commands":{
               "Values":[
                  "driverquery.exe"
               ]
            }
         },
         "Priority":3,
         "Type":"RUN_COMMAND",
         "Targets":[
            {
               "Values":[
                  "i-1a2b3c4d5e6f7g8h9"
               ],
               "Key":"InstanceIds"
            }
         ]
      },
      {
         "ServiceRoleArn":"arn:aws:iam::2222222222:role/MW-Role",
         "MaxErrors":"1",
         "TaskArn":"AWS-RunPowerShellScript",
         "MaxConcurrency":"1",
         "WindowTaskId":"44444-44-44-444444",
         "TaskParameters":{
            "commands":{
               "Values":[
                  "ipconfig.exe"
               ]
            }
         },
         "Priority":1,
         "Type":"RUN_COMMAND",
         "Targets":[
            {
               "Values":[
                  "555555-55555-555-5555555"
               ],
               "Key":"WindowTargetIds"
            }
         ]
      }
   ]
}
```

8. Execute the following command to view a list of task executions for a specific Maintenance Window.

```
aws ssm describe-maintenance-window-executions --window-id "mw-ab12cd34ef56gh78"
```

The system returns information like the following.

```
{
   "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...      {
         "Status":"SUCCESS",
         "WindowExecutionId":"33333-333-333-3333333",
         "StartTime":1478272795.021
      },
      {
         "Status":"SUCCESS",
         "WindowExecutionId":"4444-44-44-44444444",
         "StartTime":1478273694.932
      }
   ],
   "NextToken":111111   ..."
}
```

9. Execute the following command to get information about a Maintenance Window task execution.

```
aws ssm get-maintenance-window-execution --window-execution-id
 "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
```

The system returns information like the following.

```
{
   "Status":"SUCCESS",
   "TaskIds":[
      "333-33-3333-333333"
   ],
   "StartTime":1478230495.472,
   "EndTime":1478230516.505,
   "WindowExecutionId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

10. Execute the following command to list the tasks executed as part of a Maintenance Window execution.

```
aws ssm describe-maintenance-window-execution-tasks --window-execution-id
 "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
```

The system returns information like the following.

```
{
   "WindowExecutionTaskIdentities":[
      {
         "Status":"SUCCESS",
         "EndTime":1478230516.425,
         "StartTime":1478230495.782,
         "TaskId":"33333-333-333-3333333"
      }
   ]
}
```

11. Execute the following command to get the details of a task execution.

```
aws ssm get-maintenance-window-execution-task --window-execution-id
 "555555-555-55-555555" --task-id "4444-4444-4444-444444"
```

The system returns information like the following.

```
{
   "Status":"SUCCESS",
   "MaxErrors":"1",
   "TaskArn":"AWS-RunPowerShellScript",
   "MaxConcurrency":"1",
   "ServiceRole":"arn:aws:iam::333333333:role/MW-Role",
   "WindowExecutionId":"555555-555-55-555555",
   "Priority":0,
   "StartTime":1478230495.782,
   "EndTime":1478230516.425,
   "Type":"RUN_COMMAND",
   "TaskParameters":[

   ],
   "TaskExecutionId":"4444-4444-4444-444444"
}
```

12. Execute the following command to get the specific task invocations performed for a task execution.

```
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.

```
{
   "WindowExecutionTaskInvocationIdentities":[
      {
         "Status":"SUCCESS",
         "Parameters":"{\" documentName \" : \" AWS-RunPowerShellScript \" , \"
 instanceIds \" :[ \" i-1a2b3c4d5e6f7g8h9 \" , \" i-0a
00def7faa94f1dc \" ], \" parameters \" :{ \" commands \" :[ \" ipconfig.exe \" ]}, \"
 maxConcurrency \" : \" 1 \" , \" maxErrors \" : \" 1 \" }",
         "ExecutionId":"555555-555-55-555555",
         "InvocationId":"3333-33333-3333-33333",
         "StartTime":1478230495.842,
         "EndTime":1478230516.291
      }
   ]
}
```

Additional Maintenance Window Configuration Commands

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

```
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

```
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

```
aws ssm describe-maintenance-windows --filters "Key=Enabled,Values=false"
```

The system returns information like the following.

```
{
   "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"
        }
    ]
}
```

Modify a Maintenance Window

You can modify the following parameters: Name, Schedule, Duration, Cutoff, AllowUnassociatedTargets, and Enabled. The following example modifies the `name` value.

```
aws ssm update-maintenance-window --window-id "mw-1a2b3c4d5e6f7g8h9" --name "My-Renamed-MW"
```

The system returns information like the following.

```
{
    "Cutoff": 1,
    "Name": "My-Renamed-MW",
    "Schedule": "cron(0 16 ? * TUE *)",
    "Enabled": true,
    "AllowUnassociatedTargets": true,
    "WindowId": "mw-1a2b3c4d5e6f7g8h9",
    "Duration": 4
}
```

Modifying the unassociated targets parameter

```
aws ssm update-maintenance-window --window-id "mw-1a2b3c4d5e6f7g8h9" --no-allow-
unassociated-targets
```

The system returns information like the following.

```
{
    "Cutoff": 2,
    "Name": "Every-Tuesday-4pm",
    "Schedule": "cron(0 16 ? * TUE *)",
    "Enabled": true,
    "AllowUnassociatedTargets": false,
    "WindowId": "mw-1a2b3c4d5e6f7g8h9",
    "Duration": 8
}
```

```
aws ssm update-maintenance-window --window-id "mw-1a2b3c4d5e6f7g8h9" --allow-unassociated-
targets --no-enabled
```

The system returns information like the following.

```
{
    "Cutoff": 2,
    "Name": "Every-Tuesday-4pm",
    "Schedule": "cron(0 16 ? * TUE *)",
    "Enabled": false,
    "AllowUnassociatedTargets": true,
    "WindowId": "mw-1a2b3c4d5e6f7g8h9",
    "Duration": 8
}
```

Display the Targets for a Maintenance Window Matching a Specific Owner Information Value

```
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

```
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"
            }
         ]
      },
      {
         "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":"44444-444-4444-444444",
               "TaskTargetType":"WINDOW_TARGET"
            }
         ]
      }
   ]
}
```

Show All Registered Tasks that Have a Priority of 3

```
aws ssm describe-maintenance-window-tasks --window-id "mw-ab12cd34ef56gh78" --filters
 "Key=Priority,Values=3"
```

The system returns information like the following.

```
{
   "Tasks":[
      {
         "ServiceRoleArn":"arn:aws:iam::222222222:role/MW-Role",
         "MaxErrors":"1",
         "TaskArn":"AWS-RunPowerShellScript",
         "MaxConcurrency":"1",
         "WindowTaskId":"333333-333-33333-33333",
         "TaskParameters":{
            "commands":{
               "Values":[
                  "driverquery.exe"
               ]
            }
         },
         "Priority":3,
         "Type":"RUN_COMMAND",
         "Targets":[
            {
               "TaskTargetId":"i-1a2b3c4d5e6f7g8h9",
               "TaskTargetType":"INSTANCE"
            }
         ]
      }
   ]
}
```

Show All Registered Tasks that Have a Priority of 1 and Use Run Command

```
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":"777-77-777-7777777",
               "TaskTargetType":"WINDOW_TARGET"
            }
         ]
      }
   ]
}
```

List All Tasks Executed Before a Date

```
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

```
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":[
      {
         "Status":"SUCCESS",
         "WindowExecutionId":"33333-4444-444-5555555",
         "StartTime":1478279095.042
      },
      {
         "Status":"SUCCESS",
         "WindowExecutionId":"55555-6666-6666-777777",
         "StartTime":1478279994.958
      },
      {
         "Status":"SUCCESS",
         "WindowExecutionId":"8888-888-888-888888",
         "StartTime":1478280895.149
      }
   ]
}
```

Remove a Target from a Maintenance Window

```
aws ssm deregister-target-from-maintenance-window --region an SSM region --window-id "mw-
ab12cd34ef56gh78" --window-target-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
```

The system returns information like the following.

```
{
   "WindowId":"mw-ab12cd34ef56gh78",
   "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Remove a Task from a Maintenance Window

```
aws ssm deregister-task-from-maintenance-window --window-id "mw-ab12cd34ef56gh78" --window-
task-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c"
```

The system returns information like the following.

```
{
   "WindowTaskId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c",
   "WindowId":"mw-ab12cd34ef56gh78"
}
```

Delete a Maintenance Window

```
aws ssm delete-maintenance-window --window-id "mw-1a2b3c4d5e6f7g8h9"
```

The system returns information like the following:

```
{
   "WindowId":"mw-1a2b3c4d5e6f7g8h9"
}
```

Systems Manager Automation

Amazon EC2 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 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 EC2 console.

Contents

Setting Up Automation

To set up Automation, 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 configure these roles using either an AWS CloudFormation template or manually in the IAM console. This section also include information about how to create CloudWatch Events to receive notifications about Automation actions.

Note
After you complete one of the following methods for configuring access, you can further delegate access to Automation by using a more restrictive IAM user policy. For more information, see Create a Restrictive IAM User Policy (p. 10).

Choose one of the following methods to configure IAM roles. And then, optionally, configure CloudWatch Events.

Topics

Method 1: Using AWS CloudFormation to Configure Roles for Automation

Automation requires an IAM instance profile role and a service role. The instance profile role gives Automation permission to perform actions on your instances, such as executing commands or starting and stopping services. 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 Amazon Machine Image (AMI) when executing the `aws:createImage` action in an Automation document. You can create an IAM instance profile role and a service role for Systems Manager Automation from an AWS CloudFormation template, as described in this section.

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. 8).

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. 78).

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. On your local computer, open a text editor such as Notepad. Copy and paste the following AWS CloudFormation template into the text editor and save the file with a .yaml file extension (for example, automationsetup.yaml).

   Important
   Preserve the indentations of this sample template when you paste it into the text editor. YAML uses the indentations to distinguish between data layers.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS CloudFormation template IAM Roles for Systems Manager | Automation
 Service
```

```
Resources:
  ManagedInstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - ssm.amazonaws.com
            - ec2.amazonaws.com
          Action: sts:AssumeRole
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonEC2RoleforSSM
      Path: "/"

  ManagedInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: "/"
      Roles:
      - !Ref ManagedInstanceRole

  AutomationServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - ssm.amazonaws.com
            - ec2.amazonaws.com
          Action: sts:AssumeRole
      ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonSSMAutomationRole
      Path: "/"
      Policies:
      - PolicyName: passrole
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
          - Effect: Allow
            Action:
            - iam:PassRole
            Resource:
            - !GetAtt ManagedInstanceRole.Arn
```

2. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
3. Choose **Create Stack**.
4. On the **Create Stack** page, under **Choose a template**, choose **Upload a template to Amazon S3**.
5. Choose **Browse**, and then choose the file you created in Step 1.
6. Choose **Next**.
7. On the **Specify Details** page, in the **Stack Name** field, type Automation, and then choose **Next**.
8. On the **Options** page, you don’t need to make any selections. Choose **Next**.
9. On the **Review** page, scroll down and choose the **I acknowledge that AWS CloudFormation might create IAM resources** option.
10. Choose **Create**.

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

Copying Role Information for Automation

Use the following procedure to copy the instance profile role and Automation service role from the AWS CloudFormation console. You must specify these roles when you when you run an Automation document, such as the AWS-UpdateLinuxAmi document.

To copy the role names

1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
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. 76) and Automation CLI Walkthrough: Patch a Linux AMI (p. 78).

Method 2: Using IAM to Configure Roles for Automation

Automation requires an IAM instance profile role and a service role. The instance profile role gives Automation permission to perform actions on your instances, such as executing commands or starting and stopping services. 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 Amazon Machine Image (AMI) when executing the `aws:createImage` action in an Automation document. You can create an IAM instance profile role and a service role for Systems Manager Automation by using the IAM console, as described in this section.

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. 8).

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. 78).

To configure access to Automation, you must perform the following tasks. If you do not configure roles and permissions correctly, Automation returns errors when executing.

Tasks

Task 1: Create an Instance Profile Role for Systems Manager Managed Instances

Managed instances require an IAM role that gives Systems Manager permission to perform actions on your instances. You can also specify this role in your Automation documents, such as the AWS-UpdateLinuxAmi document, so that Automation can perform actions on your managed instances or launch new instances.

Use the following procedure to create an instance profile role for Systems Manager that uses the **AmazonEC2RoleforSSM** managed policy. This policy enables the instance to communicate with the Systems Manager API for a limited set of management tasks.

To create an instance profile role for 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 New Role**.
3. In **Step 1: Set Role Name**, enter a name that identifies this role as a Systems Manager role for managed instances.
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 the **AmazonEC2RoleforSSM** managed policy.
6. In **Step 5: Review**, make a note of the role name. You will specify this role name when you create new instances that you want to manage using Automation and in Automation documents.
7. Choose **Create Role**. The system returns you to the **Roles** page.

You can assign the instance profile role to new instances when you create the instance, or you can attach it to an existing instance. For more information, see Working with IAM Roles in the *Amazon EC2 User Guide*.

Task 2: Add a Trust Relationship for Systems Manager

Use the following procedure to configure the role policy to trust Systems Manager.

To add a trust relationship for Systems Manager

1. Locate the role you just created and double-click it.
2. Choose the **Trust Relationships** tab, and then choose **Edit Trust Relationship**.
3. Add a comma after "ec2.amazonaws.com"**,** and then add "Service": "ssm.amazonaws.com" to the existing policy as the following code snippet illustrates:

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

4. Choose **Update Trust Policy**.

Task 3: Create an IAM Role for Automation

Systems Manager Automation needs to have permission to perform the actions that you specify for the service on your behalf. It obtains these permissions by assuming your IAM role. Use the following procedures to:

- Create a role so that Automation can perform tasks on your behalf while processing Automation documents.
- Establish a trust relationship between the Automation role and Systems Manager
- Assign permissions to the role so that you can reference IAM roles within an Automation document.

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 New Role**.
3. In **Step 1: Set Role Name**, enter a name that identifies this role as an Automation role.
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 the **AmazonSSMAutomationRole** managed policy. They provide the same access permissions.
6. In **Step 5: Review**, 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 will also specify the role name and the ARN in EC2 Automation documents.
7. Choose **Create Role**. The system returns you to the **Roles** page.

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 4: Add a Trust Relationship for Automation

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

To add a trust relationship for Automation

1. In the IAM console, locate the role you just created and double-click it.
2. Choose the **Trust Relationships** tab, and then choose **Edit Trust Relationship**.
3. Using the following code snippet as an example, add a comma after "ec2.amazonaws.com"**,** and then add "Service": "ssm.amazonaws.com" to the existing policy:

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

4. Choose **Update Trust Policy**.
5. Copy or make a note of the **Role ARN**. You will specify this ARN in your automation document.

Task 5: Attach the iam:PassRole Policy to Your Automation Role

Use the following procedure to attach the iam:PassRole policy to your Automation role. This enables the Automation service to pass the roles you created earlier during execution.

To attach the iam:PassRole policy to your Automation role

1. In the IAM console, copy the ARNs of the roles created in Tasks 1 and 3.
2. Locate the Automation role you created in Task 3 and double-click it.
3. Choose the **Permissions** tab.
4. In the **Inline Policies** section, choose **Create User Policy**. If you don't see this button, choose the down arrow beside **Inline Policies**, and then choose **click here**.
5. On the **Set Permissions** page, choose **Policy Generator**, and then choose **Select**.
6. Verify that **Effect** is set to **Allow**.
7. From **AWS Services**, choose **AWS Identity and Access Management**.
8. From **Actions**, choose **PassRole**.
9. In the **Amazon Resource Name (ARN)** field, paste the Automation role ARN that you created in Task 1.
10. Choose **Add Statement**.
11. From **AWS Services**, choose **AWS Identity and Access Management**.
12. From **Actions**, choose **PassRole**.
13. In the **Amazon Resource Name (ARN)** field, paste the Automation role ARN that you created in Task 3.
14. Choose **Add Statement**, and then choose **Next Step**.
15. On the **Review Policy** page, choose **Apply Policy**.

Task 6: Configure User Access to Automation

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*.

Use the following procedure to add the iam:PassRole policy you created in Task 5 to the user account. This enables the user account to pass the role to Automation. In this procedure, you will also configure the account to use the **AmazonSSMFullAccess policy** so the account can communicate with the Systems Manager API.

To attach the iam:PassRole policy to a user account

1. In the IAM navigation pane, choose **Users**, and then double-click the user account you want to configure.
2. In the **Managed Policies** section, verify that either the `AmazonSSMFullAccess` policy is listed or there is a comparable policy that gives the account permissions for the SSM API.
3. In the **Inline Policies** section, choose **Create User Policy**. If you don't see this button, choose the down arrow beside **Inline Policies**, and then choose **click here**.
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 role you created in Task 3.
9. Choose **Add Statement**, and then choose **Next Step**.
10. On the **Review Policy** page, choose **Apply Policy**.

You can further delegate access to Automation by using a more restrictive IAM user policy. For more information, see Create a Restrictive IAM User Policy (p. 10).

Configuring CloudWatch Events for Systems Manager Automation

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

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. 68). For information about creating a custom Automation document, see Create an Automation Document (p. 83).

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.

Walkthroughts

- Automation Console Walkthrough: Patch a Linux AMI (p. 76)
- Automation CLI Walkthrough: Patch a Linux AMI (p. 78)

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 Systems Manager Documents (p. 29). For information about actions you can add to a document, see Systems Manager Automation Actions (p. 100).

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 Amazon EC2 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.

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.
   - An AWS Identity and Access Management (IAM) instance profile role that gives Systems Manager permission to perform actions on your instances. For more information, see Method 2: Using IAM to Configure Roles for Automation (p. 71).
   - An IAM service role for Automation (assume role) that Automation uses to perform actions on your behalf. For more information, see Setting Up Automation (p. 68).
   - (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.
2. Open the Amazon EC2 console, expand **Systems Manager Services** in the navigation pane, and then choose **Automations**.
3. Choose **Run automation**.
4. In the **Document name** list, choose **AWS-UpdateLinuxAmi**.
5. In the **Version** list, choose **1**.
6. In the **Input parameters** section, enter the information you collected in Step 1.
7. Choose **Run automation**. The system displays an automation execution ID. Choose **OK**.
8. 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 Amazon EC2 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.

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.
   - An AWS Identity and Access Management (IAM) instance profile role that gives Automation permission to perform actions on your instances. For more information, see Method 2: Using IAM to Configure Roles for Automation (p. 71).
   - An IAM service role for Automation (assume role) that Automation uses to perform actions on your behalf. For more information, see Setting Up Automation (p. 68).
2. Download the AWS CLI to your local machine.
3. 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 role.

```
aws ssm start-automation-execution \
    --document-name "AWS-UpdateLinuxAmi" \
    --parameters \
    "AutomationAssumeRole=arn:aws:iam::1234561213:role/MyAutomationRole,
     SourceAmiId=ami-e6d5d2f1,
     InstanceIamRole=MyEc2InstanceProfileRole"
```

The command returns an execution ID. Copy this ID to the clipboard. You will use this ID to view the status of the workflow.

```
{
    "AutomationExecutionId": "ID"
}
```

4. To view the workflow execution using the CLI, execute the following command:

```
aws ssm describe-automation-executions
```

5. To view details about the execution progress, execute the following command:

```
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 Amazon EC2 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

An Amazon EC2 Systems Manager Automation document defines the actions that Systems Manager performs on your managed instances and AWS resources. Documents use JavaScript Object Notation (JSON), 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.

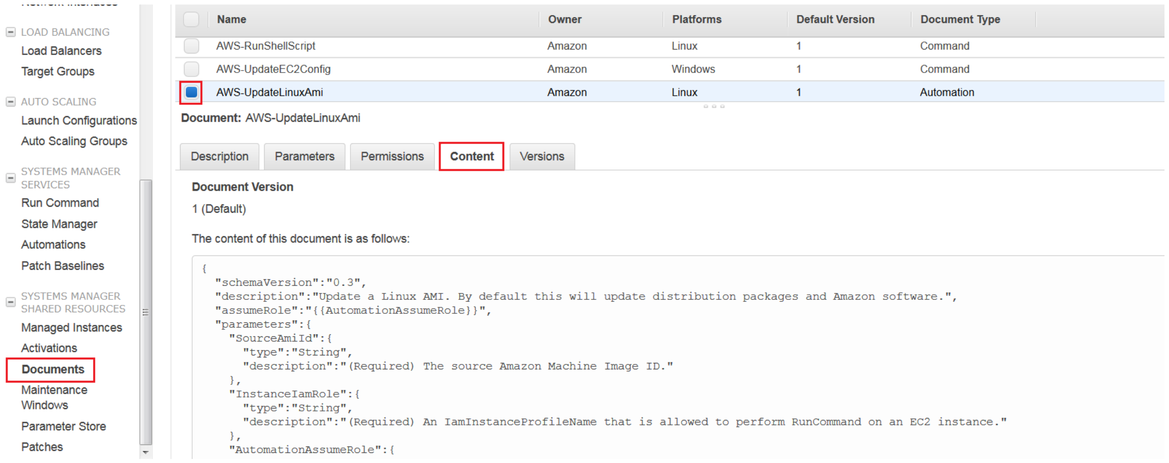
Contents

Working with Predefined Automation Documents

To help you get started quickly, Systems Manager provides a pre-defined Automation document that is maintained by Amazon Web Services. The document is named AWS-UpdateLinuxAmi. This document enables you to automate image-maintenance tasks without having to author the workflow in JavaScript Object Notation (JSON). 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 Cent OS 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 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 and other information about the document, as shown in the following image.



	Name	Owner	Platforms	Default Version	Document Type
	AWS-RunShellScript	Amazon	Linux	1	Command
	AWS-UpdateEC2Config	Amazon	Windows	1	Command
✓	AWS-UpdateLinuxAmi	Amazon	Linux	1	Automation

Document: AWS-UpdateLinuxAmi

Document Version

1 (Default)

The content of this document is as follows:

```
{
  "schemaVersion":"0.3",
  "description":"Update a Linux AMI. By default this will update distribution packages and Amazon software.",
  "assumeRole":"{{AutomationAssumeRole}}",
  "parameters":{
    "SourceAmiId":{
      "type":"String",
      "description":"(Required) The source Amazon Machine Image ID."
    },
    "InstanceIamRole":{
      "type":"String",
      "description":"(Required) An IamInstanceProfileName that is allowed to perform RunCommand on an EC2 instance."
    },
    "AutomationAssumeRole":{
```

The AWS-UpdateLinuxAmi document accepts the following input parameters.

Parameter	Type	Description
SourceAmiId	String	(Required) The source AMI ID.
InstanceIamRole	String	(Required) The name of the AWS Identity and Access Management (IAM) instance profile role you created in Setting Up Automation (p. 68). 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.
AutomationAssumeRole	String	(Required) The name of the IAM service role you created in Setting Up Automation (p. 68). The service role (also called an assume role) gives Automation permission to assume your IAM

Parameter	Type	Description
		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.
TargetAmiName	String	(Optional) The name of the new AMI after it is created. The default name is a system-generated string that includes the source AMI ID, and the creation time and date.
InstanceType	String	(Optional) The type of instance to launch as the workspace host. Instance types vary by region. The default type is t2.micro.
PreUpdateScript	String	(Optional) URL of a script to run before updates are applied. Default (\"none\") is to not run a script.
PostUpdateScript	String	(Optional) URL of a script to run after package updates are applied. Default (\"none\") is to not run a script.
IncludePackages	String	(Optional) Only update these named packages. By default (\"all\"), all available updates are applied.
ExcludePackages	String	(Optional) Names of packages to hold back from updates, under all conditions. By default (\"none\"), no package is excluded.

Automation Steps

The AWS-UpdateLinuxAmi 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 Actions (p. 100). For information about how to use Automation documents, see Systems Manager Automation Walkthroughs (p. 76)

Create 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. Automation documents include the actions to be performed during workflow execution. For more information about Systems Manager documents, see Systems Manager Documents (p. 29). For information about actions you can add to a document, see Systems Manager Automation Actions (p. 100)

The following list shows the supported actions:

- **aws:runInstance**: Launches one or more instances for a given AMI ID.

- **aws:runCommand**: Remote command execution. Executes an SSM Run Command document.
- **aws:invokeLambdaFunction**: Enables you to run external worker functions in your automation workflow.
- **aws:changeInstanceState**: Changes an instance state to `stopped`, `terminated` or `running`.
- **aws:createImage**: Creates an AMI from a running instance.
- **aws:deleteImage**: Deletes an AMI.
- **aws:createTags**: Tags EC2 and Systems Manager resources.

You can view these actions in the sample Automation document in the following procedure.

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.
   - An AWS Identity and Access Management (IAM) instance role that gives Systems Manager permission to perform actions on your instances. For more information, see Method 2: Using IAM to Configure Roles for Automation (p. 71).
   - An IAM role for Automation that Systems Manager uses to perform actions on your behalf. This is called the *assume role*. For more information, see Method 2: Using IAM to Configure Roles for Automation (p. 71).
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.

```
{
   "description":"Systems Manager Automation Demo - Patch and Create a New AMI",
   "schemaVersion":"0.3",
   "assumeRole":"the role ARN you created",
   "parameters":{
      "sourceAMIid":{
         "type":"String",
         "description":"AMI to patch"
      },
      "targetAMIname":{
         "type":"String",
         "description":"Name of new AMI",
         "default":"patchedAMI-{{global:DATE_TIME}}"
      }
   },
   "mainSteps":[
      {
         "name":"startInstances",
         "action":"aws:runInstances",
         "timeoutSeconds":1200,
         "maxAttempts":1,
         "onFailure":"Abort",
         "inputs":{
            "ImageId":"{{ sourceAMIid }}",
            "InstanceType":"m3.large",
            "MinInstanceCount":1,
            "MaxInstanceCount":1,
            "IamInstanceProfileName":"the name of the IAM role you created"
         }
      },
      {
         "name":"installMissingWindowsUpdates",
         "action":"aws:runCommand",
         "maxAttempts":1,
         "onFailure":"Continue",
         "inputs":{
            "DocumentName":"AWS-InstallMissingWindowsUpdates",
            "InstanceIds":[
               "{{ startInstances.InstanceIds }}"
            ],
            "Parameters":{
               "UpdateLevel":"Important"
            }
         }
      },
      {
         "name":"stopInstance",
         "action":"aws:changeInstanceState",
         "maxAttempts":1,
         "onFailure":"Continue",
         "inputs":{
            "InstanceIds":[
               "{{ startInstances.InstanceIds }}"
            ],
            "DesiredState":"stopped"
         }
      },
      {
         "name":"createImage",
         "action":"aws:createImage",
         "maxAttempts":1,
         "onFailure":"Continue",
         "inputs":{
            "InstanceId":"{{ startInstances.InstanceIds }}",
            "ImageName":"{{ targetAMIname }}",
            "NoReboot":true,
            "ImageDescription":"AMI created by EC2 Automation"
         }
      },
      {
         "name":"createTags",
         "action":"aws:createTags",
         "maxAttempts":1,
         "onFailure":"Continue",
         "inputs":{
            "ResourceType":"EC2",
            "ResourceIds":[
               "{{createImage.ImageId}}"
            ],
            "Tags":[
               {
                  "Key": "Generated By Automation",
                  "Value": "{{automation:EXECUTION_ID}}"
               },
               {
                  "Key": "From Source AMI",
                  "Value": "{{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 file on your local machine. Execute the command to create the required Automation document.

```
aws ssm create-document --name "patchWindowsAmi" --content file:///Users/test-user/
Documents/patchWindowsAmi.json --document-type Automation
```

   The system returns information about the command progress.

```
{
    "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": "809632081692",
        "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.

```
aws ssm list-documents --document-filter-list key=Owner,value=Self
```

   The system returns information like the following:

```
{
   "DocumentIdentifiers":[
      {
         "Name":" patchWindowsAmi",
         "PlatformTypes": [

         ],
         "DocumentVersion": "5",
         "DocumentType": "Automation",
         "Owner": "12345678901",
         "SchemaVersion": "0.3"
      }
   ]
}
```

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": "99d5b2e33571a6bb52c629283bca0a164026cd201876adf0a76de16766fb98ac",
      "Name": "patchWindowsAmi",
      "Parameters": [
         {
            "DefaultValue": "ami-3f0c4628",
            "Type": "String",
            "Name": "sourceAMIid",
            "Description": "AMI to patch"
         },
         {
            "DefaultValue": "patchedAMI-{{global:DATE_TIME}}",
            "Type": "String",
            "Name": "targetAMIname",
            "Description": "Name of new AMI"
         }
      ],
      "DocumentType": "Automation",
      "PlatformTypes": [

      ],
      "DocumentVersion": "5",
      "HashType": "Sha256",
      "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.

```
{
    "AutomationExecutionId": "ID"
}
```

You can monitor the status of the workflow in the EC2 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:

```
aws ssm describe-automation-executions
```

9. To view details about the execution progress, execute the following command.

```
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. 88).

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 Create an Automation Document (p. 83).

Examples

Simplify AMI Patching Using Automation, Lambda, and Parameter Store

The following example expands on how to update a Windows AMI, as described in Create an Automation Document (p. 83). 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. 68).

Contents

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.

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**.
3. On the **Select blueprint** page, choose **Blank 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.

```
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 Descrition field, update it PLUS the Value
    if 'Description' in response['Parameters'][0]:
        description = response['Parameters'][0]['Description']

        response = client.put_parameter(
          Name=event['parameterName'],
          Value=event['parameterValue'],
          Description=description,
          Type='String',
          Overwrite=True
        )
    #otherwise just update Value
    else:
        response = client.put_parameter(
          Name=event['parameterName'],
          Value=event['parameterValue'],
          Type='String',
          Overwrite=True
        )

    reponseString = 'Updated parameter %s with value %s.' % (event['parameterName'],
 event['parameterValue'])

    return reponseString
```

8. 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.

```
{
   "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.

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.

   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. 68).

```
{
   "description":"Systems Manager Automation Demo – Patch AMI and Update SSM Param",
   "schemaVersion":"0.3",
   "assumeRole":"the role ARN you created",
   "parameters":{
      "sourceAMIid":{
         "type":"String",
         "description":"AMI to patch",
         "default":"{{ssm:latestAmi}}"
      },
      "targetAMIname":{
         "type":"String",
         "description":"Name of new AMI",
         "default":"patchedAMI-{{global:DATE_TIME}}"
      }
   },
   "mainSteps":[
      {
         "name":"startInstances",
         "action":"aws:runInstances",
         "timeoutSeconds":1200,
         "maxAttempts":1,
         "onFailure":"Abort",
         "inputs":{
            "ImageId":"{{ sourceAMIid }}",
            "InstanceType":"m3.large",
            "MinInstanceCount":1,
            "MaxInstanceCount":1,
            "IamInstanceProfileName":"the name of the IAM role you created"
         }
      },
      {
         "name":"installMissingWindowsUpdates",
         "action":"aws:runCommand",
         "maxAttempts":1,
         "onFailure":"Continue",
         "inputs":{
            "DocumentName":"AWS-InstallMissingWindowsUpdates",
            "InstanceIds":[
               "{{ startInstances.InstanceIds }}"
            ],
            "Parameters":{
               "UpdateLevel":"Important"
            }
         }
      },
      {
         "name":"stopInstance",
         "action":"aws:changeInstanceState",
         "maxAttempts":1,
         "onFailure":"Continue",
         "inputs":{
            "InstanceIds":[
               "{{ startInstances.InstanceIds }}"
            ],
            "DesiredState":"stopped"
         }
      },
      {
         "name":"createImage",
         "action":"aws:createImage",
         "maxAttempts":1,
         "onFailure":"Continue",
         "inputs":{
            "InstanceId":"{{ startInstances.InstanceIds }}",
            "ImageName":"{{ targetAMIname }}",
            "NoReboot":true,
            "ImageDescription":"AMI created by EC2 Automation"
         }
      },
      {
         "name":"terminateInstance",
         "action":"aws:changeInstanceState",
         "maxAttempts":1,
         "onFailure":"Continue",
         "inputs":{
            "InstanceIds":[
               "{{ startInstances.InstanceIds }}"
            ],
            "DesiredState":"terminated"
         }
      },
      {
         "name":"updateSsmParam",
         "action":"aws:invokeLambdaFunction",
         "timeoutSeconds":1200,
         "maxAttempts":1,
         "onFailure":"Abort",
         "inputs":{
            "FunctionName":"Automation-UpdateSsmParam",
            "Payload":"{\"parameterName\":\"latestAmi\", \"parameterValue\":
\"{{createImage.ImageId}}\"}"
         }
      }
   ],
   "outputs":[
      "createImage.ImageId"
   ]
}
```

7. Choose **Create Document** to save the document.
8. Expand **Systems Manager Services** in the navigation pane, choose **Automations**, and then choose **Run automation**.
9. In the **Document name** list, choose **UpdateMyLatestWindowsAmi**.
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. 88) 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. 68).
- 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.

```
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

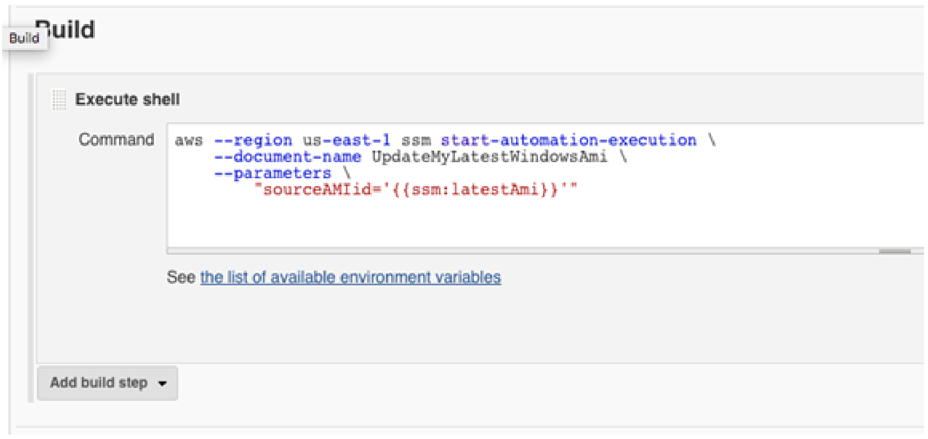
1. Open the Jenkins console in a web browser.
2. Choose the project that you want to configure with Automation, and then choose **Configure**.
3. On the **Build** tab, choose **Add Build Step**.
4. Choose **Execute shell** or **Execute Windows batch command** (depending on your operating system).
5. In the **Command** box, execute an AWS CLI command like the following:

```
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. 88):

```
aws --region us-east-1 ssm start-automation-execution \
    --document-name UpdateMyLatestWindowsAmi \
    --parameters \
        "sourceAMIid='{{ssm:latestAmi}}'"
```

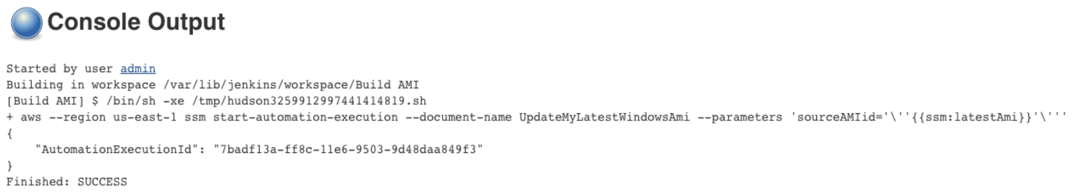
In Jenkins, the command looks like the example in the following screenshot.



Command
```
aws --region us-east-1 ssm start-automation-execution \
    --document-name UpdateMyLatestWindowsAmi \
    --parameters \
        "sourceAMIid='{{ssm:latestAmi}}'"
```

See the list of available environment variables

6. In the Jenkins project, choose **Build Now**. Jenkins returns output similar to the following example.



```
Started by user admin
Building in workspace /var/lib/jenkins/workspace/Build AMI
[Build AMI] $ /bin/sh -xe /tmp/hudson3259912997441414819.sh
+ aws --region us-east-1 ssm start-automation-execution --document-name UpdateMyLatestWindowsAmi --parameters 'sourceAMIid='\''{{ssm:latestAmi}}'\'''
{
    "AutomationExecutionId": "7badf13a-ff8c-11e6-9503-9d48daa849f3"
}
Finished: SUCCESS
```

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. 88) 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. 88) 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. 68).

Task 1: Create an IAM Role for AWS Lambda

Use the following procedure to create an IAM service role for AWS Lambda. This role includes the **AWSLambdaExecute** and **AutoScalingFullAccess** managed policies. These policies give Lambda permission to create a new Auto Scaling group with the latest, patched AMI using a Lambda function.

To create an IAM service role for Lambda

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**, and then choose **Create 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/.
2. Choose **Create a Lambda function**.
3. On the **Select blueprint** page, choose **Blank 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.

```
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.

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 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. 68).

```
{
   "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":{
            "InstanceIds":[
               "{{ startInstances.InstanceIds }}"
            ],
            "DesiredState":"stopped"
         }
      },
      {
         "name":"createImage",
         "action":"aws:createImage",
         "maxAttempts":1,
         "onFailure":"Continue",
         "inputs":{
            "InstanceId":"{{ startInstances.InstanceIds }}",
            "ImageName":"{{ targetAMIname }}",
            "NoReboot":true,
            "ImageDescription":"AMI created by EC2 Automation"
         }
      },
      {
         "name":"terminateInstance",
         "action":"aws:changeInstanceState",
         "maxAttempts":1
         "onFailure":"Continue",
         "inputs":{
            "InstanceIds":[
               "{{ startInstances.InstanceIds }}"
            ],
            "DesiredState":"terminated"
         }
      },
      {
         "name":"updateASG",
         "action":"aws:invokeLambdaFunction",
         "timeoutSeconds":1200,
         "maxAttempts":1,
         "onFailure":"Abort",
         "inputs": {
            "FunctionName": "Automation-UpdateAsg",
            "Payload": "{\"targetASG\":\"{{targetASG}}\", \"newAmiID\":
\"{{createImage.ImageId}}\"}"
         }
      }
   ],
   "outputs":[
      "createImage.ImageId"
   ]
}
```

7. Choose **Create Document** to save the document.
8. Expand **Systems Manager Services** in the navigation pane, choose **Automations**, and then choose **Run automation**.
9. In the **Document name** list, choose **PatchAmiandUpdateAsg**.
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.

Systems Manager Automation Actions

Systems Manager Automation performs tasks defined in Automation documents. To define a task, you specify one or more of the following actions in any order in the `mainSteps` section of your Automation document.

- **aws:runInstances**: Launches one or more instances.
- **aws:runCommand**: Executes a remote command.
- **aws:invokeLambdaFunction**: Enables you to run external worker functions in your automation workflow.
- **aws:changeInstanceState**: Changes the state of an instance.
- **aws:createImage**: Creates an AMI from a running instance.
- **aws:createTag**: Creates new tags for Amazon EC2 instances or Systems Manager managed instances.
- **aws:copyImage**: Copies an AMI from any region into the current region. This action can also encrypt the new AMI.
- **aws:deleteImage**: Deletes an AMI.

The output of an action is not supposed to be specified in the document. Output is available for you to link steps or add to the output section of the document. For example, you can make the output of aws:runInstances available for a subsequent aws:runCommand action.

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.

Type: String

Required: Yes

maxAttempts

The number of times the step should be retried in case of failure. If the value is greater than 1, the step 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:runInstances Action

Launches a new instance.

Input

The action supports most API parameters. For more information, see the RunInstances API documentation.

```
{
    "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.

Required: No

Output

InstanceIds

The IDs of the instances.

aws:runCommand Action

Runs the specified commands.

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

Output

CommandId

The ID of the command.

Output

The truncated output of the command.

ResponseCode

The command status code.

Status

The status of the command.

aws:invokeLambdaFunction Action

Invokes the specified Lambda function.

Input

This action supports most invoke parameters for the Lambda service. For more information, see Invoke.

```
{
    "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:changeInstanceState Action

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

```
{
    "name":"stopMyInstance",
    "action": "aws:changeInstanceState",
    "maxAttempts": 3,
    "timeoutSeconds": 3600,
    "onFailure": "Abort",
    "inputs": {
        "InstanceIds": ["i-1234567890abcdef0"],
        "CheckStateOnly": true,
        "DesiredState": "stopped"
    }
}
```

InstanceIds

  The IDs of the instances.

  Type: 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:createImage Action

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.

```
{
    "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

The state of the newly created image.

aws:createTags Action

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.

```
{
        "name": "createTags",
        "action": "aws:createTags",
        "maxAttempts": 3,
        "onFailure": "Abort",
        "inputs": {
            "ResourceType": "EC2",
            "ResourceIds": [
                "ami-9a3768fa",
                "i-02951acd5111a8169"
            ],
            "Tags": [
                {
                    "Key": "production",
                    "Value": ""
                },
                {
                    "Key": "department",
                    "Value": "devops"
                }
            ]
        }
    }
```

ResourceIds

The IDs of the resource(s) to be tagged. If resource type 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:copyImage Action

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`.

```
{
    "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:deleteImage Action

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.

```
{
    "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

Automation System Variables

The following variables are used in Systems Manager Automation documents.

System Variables

Automation documents currently support the following system variables.

Variable	Details
global:DATE	The date (at execution time) in the format yyyy-MM-dd.
global:DATE_TIME	The date and time (at execution time) in the format yyyy-MM-dd_HH.mm.ss.
global:REGION	The region which the document is executed in. For example, us-east-1.

Automation Variables

Automation documents currently support the following automation variables.

Variable	Details
automation:EXECUTION_ID	The unique identifier assigned to the current automation execution. For example 1a2b3c-1a2b3c-1a2b3c-1a2b3c1a2b3c1a2b3c.

Terminology

The following terms describe how variables and parameters are resolved.

Term	Definition	Example
Constant ARN	A valid ARN without variables	arn:aws:iam::123456789012:role/roleName
Document Parameter	A parameter defined at the document level for an Automation document (for example, instanceId). The parameter is used in a basic string replace. Its value is supplied at Start Execution time.	`{ "description": "Create Image Demo", "version": "0.3", "assumeRole": "Your_Automation_Assume_Role_ARN", "parameters":{ "instanceId": { "type": "STRING", "description": "Instance to create image from" } }`
System variable	A general variable substituted into the document when any part of the document is evaluated.	`"activities": [{ "id": "copyImage", "activityType": "AWS-CopyImage", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "ImageName": "{{imageName}}", "SourceImageId": "{{sourceImageId}}", "SourceRegion": "{{sourceRegion}}", "Encrypted": true, "ImageDescription": "Test CopyImage Description created on {{global:DATE}}" } }]`
Automation variable	A variable relating to the automation execution substituted into the document when any part of the document is evaluated.	`{ "name": "runFixedCmds", "action": "aws:runCommand", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "DocumentName": "AWS-RunPowerShellScript", "InstanceIds": [ "{{LaunchInstance.InstanceIds}}" ],`

Term	Definition	Example
		```
		      "Parameters": {
		         "commands": [
		            "dir",
		            "date",
		            "echo {Hello
		 {{ssm:administratorName}}}",

		 "“{{outputFormat}}” -f
		 “left”,”right”,”{{global:DATE}}”,”{{auto
		         ]
		      }
		   }
		}
		```

Term	Definition	Example
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}}","{{auto
               ]
            }
         }
}
```

Supported Scenarios

Scenario	Comments	Example
Constant ARN assumeRole at create	An authorization check will be performed to check the calling user is permitted to pass the given assume role.	`{ "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" }, ...` Inputs to Start Automation Execution include : {"amiId" : ["ami-12345678"] }
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": { "commands" : [`

Scenario	Comments	Example

		`"echo {Hello`
		` {{ssm:administratorName}}}"`
		
		`]`
		
		`       }`
		`      }`
		`    },`
		
		`...`

System variable referenced within step definition	A system variable is substituted into the document at execution time. The value injected into the document is relative to when the substitution occurs. 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.	`...`
		`  "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}}}"`
		
		`]`
		
		`       }`
		`      }`
		`    }, ...`

Scenario	Comments	Example

Automation variable referenced within step definition.	Automation variables do not need to be set in the parameter list of the document. The only supported Automation variable is **automation:EXECUTION_ID**.	`...`
		`"mainSteps": [`
		`    {`
		`      "name":`
		` "invokeLambdaFunction",`
		`      "action":`
		` "aws:invokeLambdaFunction",`
		`      "maxAttempts": 1,`
		`      "onFailure":`
		` "Continue",`
		`      "inputs": {`
		`        "FunctionName":`
		` "Hello-World-`
		`LambdaFunction",`
		
		`"Payload" :`
		` "{ "executionId" :`
		` "{{automation:EXECUTION_ID}}" }"`
		`      }`
		`    }`
		`...`

Refer to output from previous step within next step definition.	This is parameter redirection. The output of a previous step is referenced using the syntax {{stepName.OutputName}}. This syntax cannot be used by the customer for Document Parameters. This is resolved at the time of execution for the referring step. The parameter is not listed in the parameters of the document.	`...`
		`"mainSteps": [`
		`    {`
		`      "name":`
		` "LaunchInstance",`
		`      "action":`
		` "aws:runInstances",`
		`      "maxAttempts": 1,`
		`      "onFailure":`
		` "Continue",`
		`      "inputs": {`
		`        "ImageId":`
		` "{{amiId}}",`
		`        "MinInstanceCount":`
		` 1,`
		`        "MaxInstanceCount":`
		` 2`
		`      }`
		`    },`
		`    {`
		`      "name":"changeState",`
		`      "action":`
		` "aws:changeInstanceState",`
		`      "maxAttempts": 1,`
		`      "onFailure":`
		` "Continue",`
		`      "inputs": {`
		`        "InstanceIds":`
		` ["{{LaunchInstance.InstanceIds}}"],`
		`        "DesiredState":`
		` "terminated"`
		`      }`
		`    }`
		
		`...`

Unsupported Scenarios

Scenario	Comment	Example

SSM Parameter supplied for assumeRole at create	Not supported.	`...`
		
		`{`
		`  "description": "Test all Automation resolvable parameters",`
		`  "schemaVersion": "0.3",`
		`  "assumeRole": "{{ssm:administratorRoleARN}}",`
		`  "parameters": {`
		
		`...`

SSM Parameter supplied for Document Parameter at start	The user supplies an input parameter at start time which is an SSM parameter	`...`
		
		`"parameters": {`
		`    "amiId": {`
		`      "type": "STRING",`
		`      "default": "ami-7f2e6015",`
		`      "description": "list of commands to execute as part of first step"`
		`    },`
		
		`...`
		
		`User supplies input : { "amiId" : "{{ssm:goldenAMIId}}" }`

Variable step definition	The definition of a step in the document is constructed by variables.	`...`
		
		`"mainSteps": [`
		`    {`
		`      "name": "LaunchInstance",`
		`      "action": "aws:runInstances",`
		`      "{{attemptModel}}": 1,`
		`      "onFailure": "Continue",`
		`      "inputs": {`
		`        "ImageId": "ami-12345678",`
		`        "MinInstanceCount": 1,`
		`        "MaxInstanceCount": 2`
		`      }`
		
		`...`

Scenario	Comment	Example
		User supplies input : { "attemptModel" : "**minAttempts**" }

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}}"
		    },
		
		...

Scenario	Comment	Example
Multi-level expansion	The document defines a variable which evaluates to the name of a variable. This sits within the variable delimeters (that is *{{ }}*) and is expanded to the value of that variable/parameter.	`... "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}} }}"] } ...` **`{{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

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

Error message: `User: user arn is not authorized to perform: ssm:StartAutomationExecution on resource: document arn (Service: AWSSimpleSystemsManagement; Status Code: 400; Error Code: AccessDeniedException; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)`

- 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 6: Configure User Access to Automation (p. 75).
- 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 6: Configure User Access to Automation (p. 75).

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 5: Attach the iam:PassRole Policy to Your Automation Role (p. 74).

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. 68). Specific details for creating this role are described in the following topic, Task 3: Create an IAM Role for Automation (p. 73).
- 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 4: Add a Trust Relationship for Automation (p. 73).

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:runInstance` action failed. Each example explores a different type of error.

Missing Image

Error message: `Automation Step Execution fails when it is launching the instance(s). Get Exception from RunInstances API of ec2 Service. Exception Message from RunInstances API: [The image id '[ami id]' does not exist (Service: AmazonEC2; Status Code: 400; Error Code: InvalidAMIID.NotFound; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)]. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.`

The `aws:runInstances` action received input for an `ImageId` that doesn't exist. To resolve this problem, update the automation document or parameter values with the correct AMI ID.

Assume Role Policy Doesn't Have Sufficient Permissions

Error message: `Automation Step Execution fails when it is launching the instance(s). Get Exception from RunInstances API of ec2 Service. Exception Message from RunInstances API: [You are not authorized to perform this operation. Encoded authorization failure message: xxxxxxx (Service: AmazonEC2; Status Code: 403; Error Code: UnauthorizedOperation; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)]. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.`

The assume role doesn't have sufficient permission to invoke the `RunInstances` API on EC2 instances. To resolve this problem, attach an IAM policy to the assume role that has permission to invoke the `RunInstances` API. For more information, see the Method 2: Using IAM to Configure Roles for Automation (p. 71).

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.

Action	AWS Service(s) Invoked by This Action	For Information About This Service	Troubleshooting Content
aws:runInstances	Amazon EC2	Amazon EC2 User Guide	Troubleshooting EC2 Instances
aws:changeInstanceState	Amazon EC2	Amazon EC2 User Guide	Troubleshooting EC2 Instances
aws:runCommand	Systems Manager	Systems Manager Run Command	Troubleshooting Run Command
aws:createImage	Amazon EC2	Amazon Machine Images	
aws:deleteImage	Amazon EC2	Amazon Machine Images	
aws:copyImage	Amazon EC2	Amazon Machine Images	
aws:createTag	Amazon EC2, Systems Manager	EC2 Resource and Tags	
aws:invokeLambdaFunction	AWS Lambda	AWS Lambda Developer Guide	Troublshooting Lambda

Automation Service Internal Error

Error message: `Internal Server Error. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.`

A problem with the Automation service is preventing the specified 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.

Systems Manager Inventory Management

You can use 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). 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

To get started with Inventory, complete the following tasks.

Task	For More Information
Update the SSM Agent on your managed instances to the latest version.	Installing SSM Agent (p. 13)
Configure your on-premises servers and VMs for Systems Manager. After you configure them, they are described as *managed instances*.	Setting Up Systems Manager in Hybrid Environments (p. 23)
Verify Systems Manager prerequisites.	Systems Manager Prerequisites (p. 4)

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 parts of inventory collection in more detail.

Part	Details
Type of information to collect	• Instance details, including system name, OS name, OS version, last boot, DNS, domain, workgroup, OS architecture, etc. • Network configuration details, including IP address, MAC address, DNS, gateway, and subnet mask. • Application details, including application names, publishers, and versions • AWS component details, including EC2 driver, agents, and versions. • Windows Server Update history. • Custom inventory details. Custom inventory is described in more detail later in this section.
Instances to collect information from	You can individually select instances or target groups of instances using EC2 tag.
When to collect information	You can specify a collection interval in terms of minutes, hours, days, and weeks. The shortest collection interval is every 30 minutes.

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. 129). To view samples of how to set up inventory collection using the Amazon EC2 console and the AWS CLI, see Systems Manager Inventory Manager Walkthrough (p. 130).

Custom Inventory

You can use custom inventory to attach any metadata you want to your instances. 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. You store information about server location in the racks in a spreadsheet. With custom inventory, you could assign rack location metadata to each instance. When you perform inventory tasks with Systems Manager, the metadata would be combined with other inventory metadata to help you understand the contents of your data center.

To record custom inventory, you can either use the Systems Manager PutInventory API action, or use the SSM Agent to upload custom inventory directly from the instance. For more information about the PutInventory API action, see the Amazon EC2 Systems Manager API Reference.

To upload custom data using SSM Agent, you must create custom inventory JSON files, as shown in the following example.

Note
You must save the file with the following extension: `.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.

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

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

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

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

The Content section includes attributes and *data*. These items 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.

The following table shows the location where custom inventory JSON files must be stored on the instance:

Operating System	Path
Windows	%SystemDrive%\ProgramData\Amazon\SSM \InstanceData\<instance-id>\inventory\custom

Operating System	Path
Linux	/var/lib/amazon/ssm/<instance-id>/inventory/custom

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

Use the following procedure to configure inventory collection on a managed instance using the Amazon EC2 console. For an example of how to configure inventory collection using the AWS CLI, see Systems Manager Inventory Manager Walkthrough (p. 130).

Before you begin

Before you configure inventory collection, complete the following tasks.

- Verify that your instances meet Systems Manager prerequisites. For more information, see Systems Manager Prerequisites (p. 4).
- Update the SSM Agent if you plan to collect inventory from an existing instance. For more information, see Installing SSM Agent (p. 13).

To configure inventory collection on a managed instance

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 **Specify Parameters** section, use the lists to enable or disable different types of inventory collection.
6. In the **Specify Output Location** 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 inventory collection is processed. If the status showed **Failed**, verify that you have the latest version of the SSM Agent installed on your instances.
9. After the collection timeframe 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.

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.

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 Walkthrough

Use the following walkthrough to collect and manage inventory in a test environment.

Contents

Launch a New Instance

Instances require an AWS Identity and Access Management (IAM) role that enables the instance to communicate with Amazon EC2 Systems Manager (SSM). You can attach an IAM role when you create a new instance, or you can attach it to an existing instance.

To create an SSM-supported IAM role

1. Open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane, choose **Roles**, **Create New Role**.
3. In **Step 1: Set Role Name**, enter a name that identifies this role as a Run Command role.
4. In **Step 2: Select Role Type**, choose **Amazon EC2 Role for Simple Systems Manager**. The system skips **Step 3: Establish Trust** because this is a managed policy.
5. In **Step 4: Attach Policy**, choose **AmazonEC2RoleforSSM**.
6. Choose **Next Step**, and then choose **Create Role**.

The following procedure describes how to attach the role you've created to a new instance. For more information about attaching a role to an existing instance, see Attaching an IAM Role to an Instance in the *Amazon EC2 User Guide*.

To create an instance that uses an SSM-supported 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 Amazon Machine Image (AMI).
4. Choose your instance type and then choose **Next: Configure Instance Details**.
5. In **Auto-assign Public IP**, choose **Enable**.
6. From **IAM role**, choose the role you created earlier.
7. Complete the wizard to launch the new instance. Make a note of the instance ID. You will need to specify this ID later in this tutorial.

   Important
   On Linux instances, you must install the SSM Agent on the instance you just created. For more information, see Installing SSM Agent on Linux (p. 15).

Grant Your User Account Access to SSM

Your user account must be configured to communicate with the SSM API. Use the following procedure to attach a managed IAM policy to your user account that grants you full access to SSM API actions.

To create the IAM policy for your user account

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**. (If this is your first time using IAM, choose **Get Started**, and then choose **Create Policy**.)
3. In the **Filter** field, type **`AmazonSSMFullAccess`** and press Enter.
4. Select the check box next to **AmazonSSMFullAccess** and then choose **Policy Actions**, **Attach**.
5. On the **Attach Policy** page, choose your user account and then choose **Attach Policy**.

Inventory Manager CLI Walkthrough

The following procedure walks you through the process of using Inventory to collect metadata from the test instance you created earlier.

To gather inventory from an instance

1. Execute the following command to create a State Manager association that runs Inventory on the instance you created earlier. This command configures the service to run every six hours and to collect

network configuration, Windows Update, and application metadata on the test instance you created earlier.

```
aws ssm create-association --name "AWS-GatherSoftwareInventory" --
targets "Key=instanceids,Values=ID of the instance you created earlier"
 --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.

```
{
    "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.

```
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"
```

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

```
aws ssm describe-instance-associations-status --instance-id ID of the instance you
 created earlier
```

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"
        }
]
}
```

The following procedure walks you through the process of using the PutInventory API to assign custom metadata to the test instance you created earlier. This example assigns rack location information to a managed instance.

To assign custom metadata to an instance for Inventory

1. Execute the following command to assign rack location information to the test instance you created earlier.

```
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"}]'
```

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

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

The system responds with information like the following.

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

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

```
aws ssm get-inventory
```

The system responds with information like the following.

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

Systems Manager Patch Management

Patch Manager automates the process of patching Windows managed instances. Use this feature of Amazon EC2 Systems Manager to scan instances for missing patches, or scan and install missing patches. You can install patches individually or to large groups of instances by using EC2 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.

Patch Manager can patch Windows Server operating systems, versions 2008 through 2016 (including all R2 versions). Patch Manager provides all patches for supported operating systems within hours of their being made available by Microsoft.

Important
AWS currently does not test the patches released by Microsoft before making them available in Patch Manager.

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.

Getting Started with Patch Manager

To get started with Patch Manager, complete the following tasks.

Task	For More Information
Update the SSM Agent on your managed instances to the latest version.	Executing Commands from the EC2 Console (p. 181)

Task	For More Information
Configure your on-premises servers and VMs for Systems Manager. After you configure them, they are described as *managed instances*.	Setting Up Systems Manager in Hybrid Environments (p. 23)
Verify Systems Manager prerequisites.	Systems Manager Prerequisites (p. 4)

Working with Patch Manager

The process of using Patch Manager involves the following steps. These steps are described in more detail in this section.

1. Verify that the AWS-DefaultPatchBaseline meets your needs, or create a patch baseline that defines 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.

Step 1: Verifying the Default Patch Baseline, or Creating a Patch Baseline

A patch baseline defines which patches should and shouldn't be installed on your instances. You can individually specify approved or rejected patches, or you can use auto-approval rules to specify that certain types of updates (for example, critical updates), should automatically be approved for patching.

Patch Manager has a pre-defined patch baseline that approves all patches classified as critical updates or security updates with a severity of Critical or Important. These patches are automatically approved by this baseline 7 days after they are released by Microsoft. You can use this baseline as it is currently configured (you can't customize it) or you can create your own patch baseline if you want greater control over which patches are approved for deployment.

If you create your own patch baseline, you can choose which patches to auto-approve by using the following categories.

- Product name: For example, 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.

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.

When you create a patch baseline keep the following information in mind:

- By default, the pre-defined patch baseline that ships with Patch Manager is designated as the *default* patch baseline. However, you can specify your own patch baseline as the default.
- For on-premises or non-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 that ships with Patch Manager.
- If a patch is listed as both approved and rejected in the same patch baseline, the patch is rejected.
- Only one patch baseline can be used for an instance.

To view an example of how to create a patch baseline by using the AWS CLI, see Systems Manager Patch Manager Walkthrough (p. 140).

Step 2: Organizing 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 environments such as development, test, and production. You can also create patch groups based on server function, for example web servers and databases. Patch groups can help you avoid deploying patches to the wrong set of instances. They can also help you avoid deploying patches too early (before they have been adequately tested).

You create a patch group by using 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 patch baseline is installed during patching.

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-ApplyPatchBaseline 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 how to create a patch baseline and patch groups by using the AWS CLI, see Systems Manager Patch Manager Walkthrough (p. 140). For more information about Amazon EC2 tags, see Tagging Your Amazon EC2 Resources in the *Amazon EC2 User Guide*.

Step 3: Scheduling 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. The process 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 EC2 tags, for example, "production servers".
3. Create a new Maintenance Window task, and specify the AWS-ApplyPatchBaselines document.

When you configure the task, you can choose to either scan instances or scan and patch instances. If you choose to scan instances, then Patch Manager scans each instance and generates a list of missing patches for you to review.

If you choose to scan and patch, then 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-ApplyPatchBaselines 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.

To view an example of how to create a patch baseline, patch groups, and a Maintenance Window task that uses the AWS-ApplyPatchBaselines document by using the AWS CLI, see Systems Manager Patch Manager Walkthrough (p. 140). For more information about Maintenance Windows, see Systems Manager Maintenance Windows (p. 43).

Step 4: Monitoring Patch Compliance

After the Maintenance Window task is complete, you can view results and patch compliance details in either the Amazon EC2 console or by using the Systems Manager API. You can view an aggregate of compliance details per instance. This aggregate view includes details such as the overall compliance state, the date of the last scan, the number of patches installed, and the number of missing patches. You can review this information on a per-instance basis to view details about specific patches. The specific patches show one of the following states.

- **Installed**: Either the patch was already installed, or Patch Manager installed it when the AWS-ApplyPatchBaseline 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 baseline, but it's not installed on instance. If you configure the AWS-ApplyPatchBaseline 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 the Internet Information Services (IIS) web server role would show Not_Applicable if it was approved in the baseline, but IIS 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.

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: PatchCompliance** filters. You can also review a specific instance by choosing the instance in the **Managed Instances** page, and then choosing the **Patch** tab. You can also use the DescribePatchGroupState and DescribeInstancePatchStatesForPatchGroup APIs to view compliance details.

`DescribePatchGroupState` returns high-level aggregated patch compliance information for a patch group, as shown in the following example.

```
{
   "InstancesWithNotApplicablePatches":0,
   "InstancesWithMissingPatches":0,
   "InstancesWithFailedPatches":1,
   "InstancesWithInstalledOtherPatches":4,
   "Instances":4,
   "InstancesWithInstalledPatches":3
}
```

`DescribeInstancePatchStatesForPatchGroup` returns the high-level patch state for the instances in the specified patch group, as shown in the following example.

```
{
   "InstancePatchStates":[
      {
         "OperationStartTime":1481259600.0,
         "FailedCount":0,
         "InstanceId":"i-08ee91c0b17045407",
         "OwnerInformation":"",
         "NotApplicableCount":2077,
         "OperationEndTime":1481259757.0,
         "PatchGroup":"Production",
         "InstalledOtherCount":186,
         "MissingCount":7,
         "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
         "Operation":"Scan",
         "InstalledCount":72
      },
      {
         "OperationStartTime":1481259602.0,
         "FailedCount":0,
         "InstanceId":"i-0fff3aab684d01b23",
         "OwnerInformation":"",
         "NotApplicableCount":2692,
         "OperationEndTime":1481259613.0,
         "PatchGroup":"Production",
         "InstalledOtherCount":3,
         "MissingCount":1,
         "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
         "Operation":"Scan",
         "InstalledCount":1
      },
      {
         "OperationStartTime":1481259547.0,
         "FailedCount":0,
         "InstanceId":"i-0a00def7faa94f1dc",
         "OwnerInformation":"",
         "NotApplicableCount":1859,
         "OperationEndTime":1481259592.0,
         "PatchGroup":"Production",
         "InstalledOtherCount":116,
         "MissingCount":1,
         "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
         "Operation":"Scan",
         "InstalledCount":110
      },
      {
         "OperationStartTime":1481259549.0,
         "FailedCount":0,
         "InstanceId":"i-09a618aec652973a9",
         "OwnerInformation":"",
         "NotApplicableCount":1637,
         "OperationEndTime":1481259837.0,
         "PatchGroup":"Production",
         "InstalledOtherCount":388,
         "MissingCount":2,
         "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
         "Operation":"Scan",
         "InstalledCount":141
      }
   ]
}
```

To view an example of how to review patch compliance details by using the AWS CLI, see Systems Manager Patch Manager Walkthrough (p. 140).

Systems Manager Patch Manager Walkthrough

The following walkthroughs show you how to use either the Amazon EC2 console or the AWS CLI to create patch baselines, patch groups, and Maintenance Windows to execute patching.

Contents

Configure Your Instances

The walkthroughs are designed to illustrate how to use Patch Manager. If you want to perform the steps in the walkthroughs on your instances, then you must configure your instances with an AWS Identity and Access Management (IAM) role for Systems Manager, assign Amazon EC2 tags to your instances, and verify that the latest version of the SSM Agent is installed on your instances.

- **Assign an IAM role**: You can use the **Amazon EC2 Role for Systems Manager** with the **AmazonEC2RoleforSSM** managed policy. You can associate the IAM role when you create a new instance, or you can attach it to an existing instance. For more information, see Working with IAM Roles in the *Amazon EC2 User Guide*.
- **Assign EC2 tags**: The walkthrough uses patch groups, which are Amazon EC2 tags. Assign tags to your instances. The key for a patch group tag must be **Patch Group**. Note that the key is case sensitive. The value can be anything you want to specify, but the key must be **Patch Group**. For more information about Amazon EC2 tags, see Tagging Your Amazon EC2 Resources in the *Amazon EC2 User Guide*.
- **Update the SSM Agent**: For more information, see, Executing Commands from the EC2 Console (p. 181)

Grant Your User Account Access to the Systems Manager API

Your user account must be configured to communicate with the Systems Manager API. Use the following procedure to attach a managed IAM policy to your user account that grants you full access to Systems Manager API actions.

To create the IAM policy for your user account

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**. (If this is your first time using IAM, choose **Get Started**, and then choose **Create Policy**.)
3. In the **Filter** field, type `AmazonSSMFullAccess` and press Enter.
4. Select the check box next to **AmazonSSMFullAccess** and then choose **Policy Actions**, **Attach**.
5. On the **Attach Policy** page, choose your user account and then choose **Attach Policy**.

Configure PassRole Permissions for a Maintenance Window

The walkthroughs perform the patch operation by using a Maintenance Window task. Systems Manager must assume your role so that it has permission to perform the actions you specify for your Maintenance Window. Use the following procedure to attach the iam:PassRole policy to your existing 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. 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. In the IAM console navigation pane, choose **Users** and then double-click your user account.
2. In the **Managed Policies** section, verify that either the `AmazonSSMFullAccess` policy is listed or there is a comparable policy that gives you permission to the Systems Manager API.
3. In the **Inline Policies** section, choose **Create User Policy**. If you don't see this button, choose the down arrow beside **Inline Policies**, and then choose **click here**.
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**.

Patch Manager Walkthrough Using the EC2 Console

The following procedures illustrate how to patch a server environment by using the AWS-DefaultPatchBaseline, patch groups, and a Maintenance Windows.

To verify the AWS-DefaultPatchBaseline

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 **AWS-DefaultPatchBaseline**.

   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 patch baselines list.

3. With the AWS-DefaultPatchBaseline select, choose the **Approval Rules** tab. Verify that auto-approving all critical and security updates with a severity of Critical or Important seven days after they are released by Microsoft is acceptable for your instances.

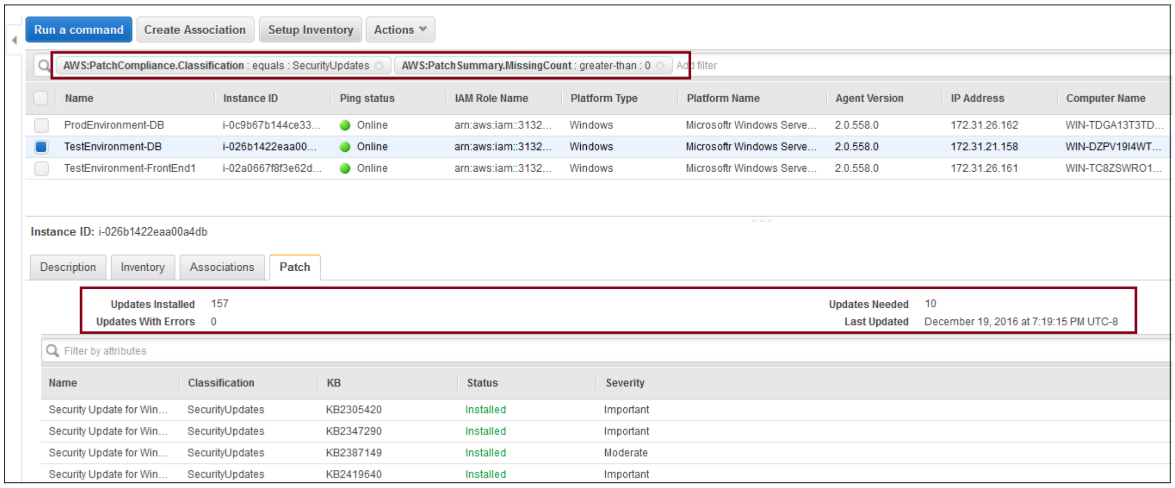
To create a Maintenance Window for patching

1. In the Amazon EC2 console navigation pane, choose **Maintenance Windows**, and then choose **Create maintenance window**.
2. In the **Name** field, type a name that designates this as a maintenance window for patching critical and important updates.
3. In the **Specify schedule** area, choose the schedule options you want.
4. In the **Duration** field, type the number of hours you want the Maintenance Window to be active.
5. 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.
6. Choose **Create maintenance window**.
7. In the Maintenance Window list, choose the Maintenance Window you just created, and then choose **Actions**, **Register targets**.
8. In the **Owner information** field, type your name or alias.
9. In the **Select targets by** area, choose **Specifying tags**.
10. In the **Tag Filters** section, in the **Tag Name** list, choose **Patch Group**.

    Note
    If you don't see this tag name in the list, then you might not have tagged your instances with the EC2 tags required for Patch Manager.
11. In the **Tag Value** list, choose the value you want, and then choose **Register targets**. The system creates a Maintenance Window target.
12. In the Maintenance Window list, choose the Maintenance Window you created with the procedure, and then choose **Actions**, **Register task**.
13. In the **Documents** section of the **Register task** page, choose **AWS-ApplyPatchBaseline**.
14. In the **Task Priority** section, specify a priority. One is the highest priority.
15. In the **Targets** section, choose **Select**, and then choose the Maintenance Window target you created earlier in this procedure.
16. In the **Operation** list, choose **Scan** to scan for missing patches, or choose **Install** to scan for and install missing patches.

    Note
    Installing missing patches will reboot the instance. Scanning does not cause a reboot.
17. You don't need to specify anything in the **Snapshot Id** field. This system automatically generates and provides this parameter.
18. In the **Role** field, enter the ARN of a role which has the **AmazonSSMMaintenanceWindowRole** policy attached to it. For more information, see Configuring Roles and Permissions for Maintenance Windows (p. 46).
19. In the **Execute on** field, choose either **Targets** or **Percent** to limit the number of instances where the system can simultaneously perform patching operations.
20. In the **Stop after** field, specify the number of allowed errors before the system stops sending the patching task to other instances.
21. In the **Advanced** section, choose **Write to S3** if you want to write command output and results to an Amazon S3 bucket.
22. 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: PatchCompliance** filters.



AWS:PatchCompliance.Classification : equals : SecurityUpdates
AWS:PatchSummary.MissingCount : greater-than : 0

Name	Instance ID	Ping status	IAM Role Name	Platform Type	Platform Name	Agent Version	IP Address	Computer Name
ProdEnvironment-DB	i-0c9b67b144ce33...	● Online	arn:aws:iam::3132...	Windows	Microsoftr Windows Serve...	2.0.558.0	172.31.26.162	WIN-TDGA13T3TD...
TestEnvironment-DB	i-026b1422eaa00...	● Online	arn:aws:iam::3132...	Windows	Microsoftr Windows Serve...	2.0.558.0	172.31.21.158	WIN-DZPV19I4WT...
TestEnvironment-FrontEnd1	i-02a0667f8f3e62d...	● Online	arn:aws:iam::3132...	Windows	Microsoftr Windows Serve...	2.0.558.0	172.31.26.161	WIN-TC8ZSWRO1...

Instance ID: i-026b1422eaa00a4db

Updates Installed	157	**Updates Needed**	10
Updates With Errors	0	**Last Updated**	December 19, 2016 at 7:19:15 PM UTC-8

Name	Classification	KB	Status	Severity
Security Update for Win...	SecurityUpdates	KB2305420	Installed	Important
Security Update for Win...	SecurityUpdates	KB2347290	Installed	Important
Security Update for Win...	SecurityUpdates	KB2387149	Installed	Moderate
Security Update for Win...	SecurityUpdates	KB2419640	Installed	Important

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 more information, see the Amazon EC2 Systems Manager API Reference.

Patch Manager Walkthrough Use the 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.

To configure Patch Manager and patch instances by 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 patch baseline named "Production-Baseline" that approves patches for a production environment seven days after they are released by Microsoft.

```
aws ssm create-patch-baseline --name "Production-Baseline" --approval-rules
 "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,Important,Modera
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]}]},ApproveAfterD
 --description "Baseline containing all updates approved for production systems"
```

The system returns information like the following.

```
{
   "BaselineId":"pb-034cba5a84f030362"
}
```

3. 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-034cba5a84f030362 --
patch-group "Production"
```

The system returns information like the following.

```
{
   "PatchGroup":"Production",
   "BaselineId":"pb-034cba5a84f030362"
}
```

```
aws ssm register-patch-baseline-for-patch-group --baseline-id pb-034cba5a84f030362 --
patch-group "Database Servers"
```

The system returns information like the following.

```
{
   "PatchGroup":"Database Servers",
   "BaselineId":"pb-034cba5a84f030362"
}
```

4. 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.

```
{
   "WindowId":"mw-09e2a75baadd84e85"
}
```

5. Execute the following commands to register the Production servers with the two production Maintenance Windows.

```
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.

```
{
   "WindowTargetId":"557e7b3a-bc2f-48dd-ae05-e282b5b20760"
}
```

```
aws ssm register-target-with-maintenance-window --window-id mw-0c66948c711a3b5bd --
targets "Key=tag:Patch Group,Values=Database Servers" --owner-information "Database
 servers" --resource-type "INSTANCE"
```

The system returns information like the following.

```
{
   "WindowTargetId":"767b6508-f4ac-445e-b6fe-758cc912e55c"
}
```

```
aws ssm register-target-with-maintenance-window --window-id mw-09e2a75baadd84e85
 --targets "Key=tag:Patch Group,Values=Production" --owner-information "Production
 servers" --resource-type "INSTANCE"
```

The system returns information like the following.

```
{
   "WindowTargetId":"faa01c41-1d57-496c-ba77-ff9cadba4b7d"
}
```

```
aws ssm register-target-with-maintenance-window --window-id mw-09e2a75baadd84e85 --
targets "Key=tag:Patch Group,Values=Database Servers" --owner-information "Database
 servers" --resource-type "INSTANCE"
```

The system returns information like the following.

```
{
   "WindowTargetId":"673b5840-58a4-42ab-8b80-95749677cb2e"
}
```

6. Execute the following commands to register a patch task that only scans the production servers for missing updates in the first production Maintenance Window.

```
aws ssm register-task-with-maintenance-window --window-id mw-0c66948c711a3b5bd --
targets "Key=WindowTargetIds,Values=557e7b3a-bc2f-48dd-ae05-e282b5b20760" --task-arn
 "AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role"
 --task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 1 --task-
parameters '{\"Operation\":{\"Values\":[\"Scan\"]}}'
```

The system returns information like the following.

```
{
   "WindowTaskId":"968e3b17-8591-4fb2-932a-b62389d6f635"
}
```

```
aws ssm register-task-with-maintenance-window --window-id mw-0c66948c711a3b5bd --
targets "Key=WindowTargetIds,Values=767b6508-f4ac-445e-b6fe-758cc912e55c" --task-arn
 "AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role"
 --task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 5 --task-
parameters '{\"Operation\":{\"Values\":[\"Scan\"]}}'
```

The system returns information like the following.

```
{
   "WindowTaskId":"09f2e873-a3a7-443f-ba0a-05cf4de5a1c7"
}
```

7. Execute the following commands to register a patch task that installs missing updates on the productions servers in the second Maintenance Window.

```
aws ssm register-task-with-maintenance-window --window-id mw-09e2a75baadd84e85 --
targets "Key=WindowTargetIds,Values=557e7b3a-bc2f-48dd-ae05-e282b5b20760" --task-arn
 "AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role"
 --task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 1 --task-
parameters '{\"Operation\":{\"Values\":[\"Install\"]}}'
```

   The system returns information like the following.

```
{
   "WindowTaskId":"968e3b17-8591-4fb2-932a-b62389d6f635"
}
```

```
aws ssm register-task-with-maintenance-window --window-id mw-09e2a75baadd84e85 --
targets "Key=WindowTargetIds,Values=767b6508-f4ac-445e-b6fe-758cc912e55c" --task-arn
 "AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role"
 --task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 5 --task-
parameters '{\"Operation\":{\"Values\":[\"Install\"]}}'
```

   The system returns information like the following.

```
{
   "WindowTaskId":"09f2e873-a3a7-443f-ba0a-05cf4de5a1c7"
}
```

8. 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."

```
aws ssm describe-patch-group-state --patch-group "Production"
```

   The system returns information like the following.

```
{
   "InstancesWithNotApplicablePatches":0,
   "InstancesWithMissingPatches":0,
   "InstancesWithFailedPatches":1,
   "InstancesWithInstalledOtherPatches":4,
   "Instances":4,
   "InstancesWithInstalledPatches":3
}
```

9. 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-0a00def7faa94f1dc",
         "OwnerInformation":"",
         "NotApplicableCount":1859,
         "OperationEndTime":1481259592.0,
         "PatchGroup":"Production",
         "InstalledOtherCount":116,
         "MissingCount":1,
         "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
         "Operation":"Scan",
         "InstalledCount":110
      },
      {
         "OperationStartTime":1481259549.0,
         "FailedCount":0,
         "InstanceId":"i-09a618aec652973a9",
         "OwnerInformation":"",
         "NotApplicableCount":1637,
         "OperationEndTime":1481259837.0,
         "PatchGroup":"Production",
         "InstalledOtherCount":388,
         "MissingCount":2,
         "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
         "Operation":"Scan",
         "InstalledCount":141
      }
   ]
}
```

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.

```
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.

```
{
   "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.

```
aws ssm update-patch-baseline --baseline-id pb-00dbb759999aa2bc3 --rejected-patches
 "KB2032276" "MS10-048" --approved-patches "KB2124261"
```

The system returns information like the following.

```
{
   "BaselineId":"pb-00dbb759999aa2bc3",
   "Name":"Windows-Server-2012R2",
   "RejectedPatches":[
      "KB2032276",
      "MS10-048"
   ],
   "GlobalFilters":{
      "PatchFilters":[

      ]
   },
   "ApprovalRules":{
      "PatchRules":[
         {
            "PatchFilterGroup":{
               "PatchFilters":[
                  {
                     "Values":[
                        "Important",
                        "Critical"
                     ],
                     "Key":"MSRC_SEVERITY"
                  },
                  {
                     "Values":[
                        "SecurityUpdates"
                     ],
                     "Key":"CLASSIFICATION"
                  },
                  {
                     "Values":[
                        "WindowsServer2012R2"
                     ],
                     "Key":"PRODUCT"
                  }
               ]
            },
            "ApproveAfterDays":5
         }
      ]
   },
   "ModifiedDate":1481001494.035,
   "CreatedDate":1480997823.81,
   "ApprovedPatches":[
      "KB2124261"
   ],
   "Description":"Windows Server 2012 R2, Important and Critical security updates"
}
```

Rename a patch baseline

```
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"
                  }
               ]
            },
            "ApproveAfterDays":5
         }
      ]
   },
   "ModifiedDate":1481001795.287,
   "CreatedDate":1480997823.81,
   "ApprovedPatches":[
      "KB2124261"
   ],
   "Description":"Windows Server 2012 R2, Important and Critical security updates"
}
```

Delete a patch baseline

```
aws ssm delete-patch-baseline --baseline-id "pb-0a34d8c0f03c1e529"
```

The system returns information like the following.

```
{
   "BaselineId":"pb-0a34d8c0f03c1e529"
}
```

List all patch baselines

```
aws ssm describe-patch-baselines
```

The system returns information like the following.

```
{
   "BaselineIdentities":[
      {
         "BaselineName":"AWS-DefaultPatchBaseline",
         "DefaultBaseline":true,
         "BaselineDescription":"Default Patch Baseline Provided by AWS.",
         "BaselineId":"arn:aws:ssm:us-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"
      }
   ]
}
```

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":[
      {
         "BaselineName":"Windows-Server-2012R2",
         "DefaultBaseline":false,
         "BaselineDescription":"Windows Server 2012 R2, Important and Critical security
 updates",
         "BaselineId":"pb-00dbb759999aa2bc3"
      }
   ]
}
```

Display a patch baseline

```
aws ssm get-patch-baseline --baseline-id pb-00dbb759999aa2bc3
```

The system returns information like the following.

```
{
   "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":[

   ],
   "Description":"Windows Server 2012 R2, Important and Critical security updates"
}
```

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-0ca44a362f8afc725"
}
```

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-00dbb759999aa2bc3" --
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-0ca44a362f8afc725" --patch-group
 "Backend"
```

The system returns information like the following.

```
{
   "PatchGroup":"Backend",
   "BaselineId":"arn:aws:ssm:us-west-1:075727635805:patchbaseline/pb-0ca44a362f8afc725"
}
```

Display patch group registrations

```
aws ssm describe-patch-groups --region us-west-1
```

The system returns information like the following.

```
{
   "PatchGroupPatchBaselineMappings":[
      {
         "PatchGroup":"Backend",
         "BaselineIdentity":{
            "BaselineName":"AWS-DefaultPatchBaseline",
            "DefaultBaseline":false,
            "BaselineDescription":"Default Patch Baseline Provided by AWS.",
            "BaselineId":"arn:aws:ssm:us-west-1:075727635805:patchbaseline/
pb-0ca44a362f8afc725"
         }
      },
      {
         "PatchGroup":"Web Servers",
         "BaselineIdentity":{
            "BaselineName":"Windows-Server-2012R2",
            "DefaultBaseline":true,
            "BaselineDescription":"Windows Server 2012 R2, Important and Critical updates",
            "BaselineId":"pb-08b654cf9b9681f04"
         }
      }
   ]
}
```

Deregister a patch group from a patch baseline

```
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.

```
{
   "PatchGroup":"Production",
   "BaselineId":"arn:aws:ssm:us-west-1:075727635805:patchbaseline/pb-0ca44a362f8afc725"
}
```

Get all patches defined by a patch baseline

```
aws ssm describe-effective-patches-for-patch-baseline --region us-west-1 --baseline-id
 "pb-08b654cf9b9681f04"
```

The system returns information like the following.

```
{
   "NextToken":"--token string truncated--",
   "EffectivePatches":[
      {
         "PatchStatus":{
            "ApprovalDate":1384711200.0,
            "DeploymentStatus":"APPROVED"
         },
         "Patch":{
            "ContentUrl":"https://support.microsoft.com/en-us/kb/2876331",
            "ProductFamily":"Windows",
            "Product":"WindowsServer2012R2",
            "Vendor":"Microsoft",
            "Description":"A security issue has been identified in a Microsoft software
 product that could affect your system. You can help protect your system by installing
 this update from Microsoft. For a complete listing of the issues that are included in
 this update, see the associated Microsoft Knowledge Base article. After you install this
 update, you may have to restart your system.",
            "Classification":"SecurityUpdates",
            "Title":"Security Update for Windows Server 2012 R2 Preview (KB2876331)",
            "ReleaseDate":1384279200.0,
            "MsrcClassification":"Critical",
            "Language":"All",
            "KbNumber":"KB2876331",
            "MsrcNumber":"MS13-089",
            "Id":"e74ccc76-85f0-4881-a738-59e9fc9a336d"
         }
      },
      {
         "PatchStatus":{
            "ApprovalDate":1428858000.0,
            "DeploymentStatus":"APPROVED"
         },
         "Patch":{
            "ContentUrl":"https://support.microsoft.com/en-us/kb/2919355",
            "ProductFamily":"Windows",
            "Product":"WindowsServer2012R2",
            "Vendor":"Microsoft",
            "Description":"Windows Server 2012 R2 Update is a cumulative set of security
 updates, critical updates and updates. You must install Windows Server 2012 R2 Update
 to ensure that your computer can continue to receive future Windows Updates, including
 security updates. For a complete listing of the issues that are included in this update,
 see the associated Microsoft Knowledge Base article for more information. After you
 install this item, you may have to restart your computer.",
            "Classification":"SecurityUpdates",
            "Title":"Windows Server 2012 R2 Update (KB2919355)",
            "ReleaseDate":1428426000.0,
            "MsrcClassification":"Critical",
            "Language":"All",
            "KbNumber":"KB2919355",
            "MsrcNumber":"MS14-018",
            "Id":"8452bac0-bf53-4fbd-915d-499de08c338b"
         }
      }
     ---output truncated---
```

Get all patches for 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,
         "MsrcClassification":"Critical",
         "Language":"All",
         "KbNumber":"KB2727528",
         "MsrcNumber":"MS12-072",
         "Id":"1eb507be-2040-4eeb-803d-abc55700b715"
      },
      {
         "ContentUrl":"https://support.microsoft.com/en-us/kb/2729462",
         "ProductFamily":"Windows",
         "Product":"WindowsServer2012",
         "Vendor":"Microsoft",
         "Description":"A security issue has been identified that could allow an
 unauthenticated remote attacker to compromise your system and gain control over it. You
 can help protect your system by installing this update from Microsoft. After you install
 this update, you may have to restart your system.",
         "Classification":"SecurityUpdates",
         "Title":"Security Update for Microsoft .NET Framework 3.5 on Windows 8 and Windows
 Server 2012 for x64-based Systems (KB2729462)",
         "ReleaseDate":1352829600.0,
         "MsrcClassification":"Critical",
         "Language":"All",
         "KbNumber":"KB2729462",
         "MsrcNumber":"MS12-074",
         "Id":"af873760-c97c-4088-ab7e-5219e120eab4"
      }

---output truncated---
```

Get all available patches

```
aws ssm describe-available-patches --region us-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,
         "MsrcClassification":"Important",
         "Language":"All",
         "KbNumber":"KB2032276",
         "MsrcNumber":"MS10-043",
         "Id":"8692029b-a3a2-4a87-a73b-8ea881b4b4d6"
      },
      {
         "ContentUrl":"https://support.microsoft.com/en-us/kb/2124261",
         "ProductFamily":"Windows",
         "Product":"Windows7",
         "Vendor":"Microsoft",
         "Description":"A security issue has been identified that could allow an
 unauthenticated remote attacker to compromise your system and gain control over it. You
 can help protect your system by installing this update from Microsoft. After you install
 this update, you may have to restart your system.",
         "Classification":"SecurityUpdates",
         "Title":"Security Update for Windows 7 (KB2124261)",
         "ReleaseDate":1284483600.0,
         "MsrcClassification":"Important",
         "Language":"All",
         "KbNumber":"KB2124261",
         "MsrcNumber":"MS10-065",
         "Id":"12ef1bed-0dd2-4633-b3ac-60888aa8ba33"
      }
      ---output truncated---
```

Tag a patch baseline

```
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

```
aws ssm list-tags-for-resource --resource-type "PatchBaseline" --resource-id
 "pb-0869b5cf84fa07081"
```

Remove a tag from a patch baseline

```
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".

```
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",
         "InstalledOtherCount":186,
         "MissingCount":7,
         "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
         "Operation":"Scan",
         "InstalledCount":72
      },
      {
         "OperationStartTime":"2016-12-09T04:59:09Z",
         "FailedCount":0,
         "InstanceId":"i-09a618aec652973a9",
         "OwnerInformation":"",
         "NotApplicableCount":1637,
         "OperationEndTime":"2016-12-09T05:03:57Z",
         "PatchGroup":"Production",
         "InstalledOtherCount":388,
         "MissingCount":2,
         "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
         "Operation":"Scan",
         "InstalledCount":141
      }
     ---output truncated---
```

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---
```

Systems Manager Parameter Store

Amazon EC2 Systems Manager Parameter Store provides secure storage for configuration data such as passwords, database strings, and license codes. You can store parameters as plain text or as encrypted objects. You can then reference these parameters in your scripts and commands without having to type parameters in plain text. Additionally, you can reference parameters across your AWS configuration and automation workflows. This improves your overall security posture.

Parameter Store also simplifies the process of managing configuration data by storing the data in one, secure location instead of in configuration files across your fleet. You can reference parameters across AWS services such as Amazon EC2 Container Service and AWS Lambda. You can also reference parameters in other Systems Manager capabilities such as Run Command, State Manager, and Automation.

Parameter Store integrates with AWS Identity and Access Management (IAM) to control parameter access. You can specify which users have access to parameters and on which resources those parameters can be used. Parameter Store integrates with AWS Key Management Service so that you can encrypt your sensitive information and protect the security of your keys. Additionally, AWS CloudTrail records all calls to Parameter Store so that you can audit usage.

Getting Started with Parameter Store

To get started with Parameter Store, complete the following tasks.

Task	For More Information
Learn about Systems Manager Parameters	About Parameter Store Parameters (p. 160)
Learn about how to use different types of Systems Manager parameters	Using Parameters (p. 160) Using Secure String Parameters (p. 162)
Configure access to Systems Manager parameters	Control Access to Systems Manager Parameters (p. 165)
Create a parameter using either the Amazon EC2 console or the AWS CLI	Systems Manager Parameter Store Walkthroughs (p. 166)

Related Content

The following blog posts provide additional information about Parameter Store and how to use this capability with other AWS services.

- 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 EC2 Systems Manager Parameter Store

About Parameter Store Parameters

A Systems Manager parameter is a key-value pair that you create by specifying 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:
  - A parameter name must be unique within your AWS account.
  - Parameter names are case-sensitive.
  - A parameter name *can't* be prefixed with "aws" or "ssm" (case-insensitive). For example, awsTestParameter or SSM-testparameter will fail with an exception.
  - Parameter names can only include the following symbols and letters:

    a-zA-Z0-9_.-
- **Data Type**: (Required) Specify a data type to define how the system uses a parameter. Parameter Store currently supports the following data types: String, String List, and Secure String.
- **Description** (Optional): Type a description to help you identify your parameters and their intended use.
- **Value**: (Required) Your parameter value.
- **Key ID** (for Secure String): Either the default AWS KMS key automatically assigned to your AWS account or a custom key.

Note
You can use a period "." or an underscore "_" to group similar parameters. For example, you could group parameters as follows: prod.db.string and prod.domain.password.

After you create a parameter, you can specify it in your SSM documents, commands, or scripts using the following syntax (no space between brackets):

{{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 name must be prefixed with "ssm:". Valid: {{ssm:addUsers}}. Invalid: {{ssm:ssmAddUsers}}.

For information about using different types of parameters, see Using Parameters (p. 160) and Using Secure String Parameters (p. 162). For information about creating a Systems Manager parameter, see Systems Manager Parameter Store Walkthroughs (p. 166).

Using 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 EC2 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 Systems Manager Documents (p. 29).

The following is an example of a Systems Manager parameter in an AWS CLI command for Run Command.

```
aws ssm send-command --instance-ids i-1a2b3c4d5e6f7g8 --document-name AWS-
RunPowerShellScript --parameter '{"commands":["echo {{ssm:addUsers}}"]}'
```

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 }}"
```

You can reference Systems Manager parameters in the *Parameters* section of an SSM document, as shown in the following example.

```
{
   "schemaVersion":"2.0",
   "description":"Sample version 2.0 document v2",
   "parameters":{
      "commands" : {
        "type": "StringList",
        "default": ["{{ssm:commands}}"]
      }
    },
    "mainSteps":[
       {
          "action":"aws:runShellScript",
          "name":"runShellScript",
          "inputs":{
             "commands": "{{commands}}"
          }
       }
    ]
}
```

Predefined SSM documents (all documents that begin with "AWS") currently don't support Secure Strings or references to Secure String type parameters. This means that to use Secure String parameters with Run Command, you have to retrieve the parameter value before passing it to Run Command, as shown in the following examples:

Linux

```
$value=aws ssm get-parameters --names secureparam --with-decryption
```

```
aws ssm send-command –name AWS-JoinDomain –parameters password=$value –instance-
id instance_ID
```

Windows

```
$secure = (Get-SSMParameterValue -Names SecureParam -WithDecryption
 $True).Parameters[0].Value | ConvertTo-SecureString -AsPlainText -Force
```

```
$cred = New-Object System.Management.Automation.PSCredential -argumentlist username,$secure
```

Using Secure String Parameters

A secure string 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 domain join passwords or license keys, then create those parameters using the Secure String data type. You should use secure strings when:

- 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 about AWS KMS, see AWS Key Management Service Developer Guide.

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
```

Create a Secure String Parameter Using the Default KMS Key

If you create a Secure String parameter using the default KMS key, 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 secure_string1_default_key --value "a_secure_string_value" --
type SecureString
```

Create a Secure String Parameter Using Your KMS Customer Master Key (CMK)

If you want to use a custom KMS key instead of the default key assigned to your account, then you must specify the ARN using the `--key-id` parameter. The parameter supports all AWS KMS parameter formats. For example:

- 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 key 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 secure_string1_custom_key --value
 "a_secure_string_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 in the Amazon EC2 console and the AWS CLI. To decrypt a Secure String value, a user must have KMS decryption permissions, as described in the next section.

Secure String Parameter Walkthrough

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 a Domain to an Instance

1. Enter parameters into the system using AWS Tools for Windows PowerShell.

   ```
   Write-SSMParameter -Name dns -Type String -Value DNS_IP_Address
   Write-SSMParameter -Name domainName -Type String -Value Domain_Name
   Write-SSMParameter -Name domainJoinUserName -Type String -Value DomainJoinUserName
   Write-SSMParameter -Name domainJoinPassword -Type SecureString -
   Value DomainJoinPassword
   ```
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.

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

```
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:kms:region:account_id:key/key_id"
         ]
      }
   ]
}
```

4. Copy and paste the following json sample into a simple text editor and save the file as JoinInstanceToDomain.json in the following location: `c:\temp\JoinInstanceToDomain.json`.

```
{
   "schemaVersion":"2.0",
   "description":"Run a PowerShell script to securely domain-join a Windows instance",
   "mainSteps":[
      {
         "action":"aws:runPowerShellScript",
         "name":"runPowerShellWithSecureString",
         "inputs":{
            "runCommand":[
               "$ipdns = (Get-SSMParameterValue -Name dns).Parameters[0].Value\n",
               "$domain = (Get-SSMParameterValue -Name domainName).Parameters[0].Value
\n",
               "$username = (Get-SSMParameterValue -Name
 domainJoinUserName).Parameters[0].Value\n",
               "$password = (Get-SSMParameterValue -Name domainJoinPassword -
WithDecryption $True).Parameters[0].Value | ConvertTo-SecureString -asPlainText -Force
\n",
               "$credential = New-Object
 System.Management.Automation.PSCredential($username,$password)\n",
               "Set-DnsClientServerAddress \"Ethernet 2\" -ServerAddresses $ipdns\n",
               "Add-Computer -DomainName $domain -Credential $credential\n",
               "Restart-Computer -force"
            ]
         }
      }
   ]
}
```

5. Execute the following command in AWS Tools for Windows PowerShell to create a new SSM document.

```
$json = Get-Content C:\temp\JoinInstanceToDomain | Out-String
New-SSMDocument -Name JoinInstanceToDomain -Content $json
```

6. Execute the following command in AWS Tools for Windows PowerShell to join the instance to the domain

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName JoinInstanceToDomain
```

Using Secure String Parameters With Other AWS Services

You can also use secure string parameters with other AWS services. Rather than referencing variables or secrets in plain text, you can store them in Parameter Store and reference them with a single API call. In the following example, the Lambda function retrieves a secure string parameter by using the GetParameters API.

```
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
```

Control Access to Systems Manager Parameters

We recommend the you restrict user access to Systems Manager parameters by creating restrictive AWS Identity and Access Management (IAM) user policies. For example, the following policy gives the user read-only permission (GetParameters and DescribeParameters) to all production parameters (parameters that begin with prod.*).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeParameters"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": "arn:aws:ssm:us-east-1:123456123:parameter/prod.*"
        }
    ]
}
```

If you want to provide a user with full access to all Systems Manager Parameter API operations, use 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.*"
   ]
}
```

You can also delegate access so that instances can only run specific parameters. For secure strings, you have to provide KMS decrypt permissions so that secure string parameters can be decrypted by the instance. The following example enable 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 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.
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
```

```
{
   "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 Security Roles for Systems Manager (p. 6).

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, as described in the following blog post, Use Parameter Store to Securely Access Secrets and Config Data in AWS CodeDeploy.

Contents

Systems Manager Parameter Store Console Walkthrough

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

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 helloWorld.
4. In the **Description** field, type a description that identifies this parameter as a test parameter.
5. For **Type**, choose **String**.
6. In the **Value** field, echo a word.
7. Choose **Create Parameter** and then choose **OK** after the system creates the parameter.
8. In the EC2 console navigation pane, expand **Commands** and then choose **Run Command**.
9. Choose **Run a command**.
10. In the **Command Document** list, choose AWS-RunPowershellScript (Windows) or AWS-RunShellScript (Linux).
11. Under **Target instances**, choose the instance you created earlier.
12. In the **Commands** field, type echo {{ssm:helloWorld}} and then choose **Run**.
13. In the command history list, choose the command you just ran, choose the **Output** tab, and then choose **View Output.**. They output is the name of the parameter you created earlier, for example, {{ssm:helloWorld}}.

Systems Manager Parameter Store CLI Walkthrough

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. Execute the following command to create a parameter that uses the String data type.

```
aws ssm put-parameter --name a name --type String —-value "a value, for example
 "helloWorld""
```

3. Execute the following command to view the parameter metadata.

```
aws ssm describe-parameters --filters "Key=Name,Values=helloWorld"
```

4. Execute the following command to change the parameter value.

```
aws ssm put-parameter --name "helloWorld" --type String --value ""good day sunshine""
 --overwrite
```

5. Execute the following command to view the latest parameter value.

```
aws ssm get-parameters --name "helloWorld"
```

6. Execute the following command to view the parameter value history.

```
aws ssm get-parameter-history --name "helloWorld"
```

7. Execute the following command to use this parameter in a Run Command command.

```
aws ssm send-command --name "AWS-RunPowerShellScript" --parameters "commands=["echo
 {{ssm:helloWorld}}"]" --targets "Key=instanceids,Values=the ID of the instance you
 created earlier"
```

To create a Secure String parameter using Parameter Store

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

```
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 KMS key

```
aws ssm put-parameter --name "a name" --value "a value, for example P@ssW%rd#1" --type
 "SecureString" --key-id "your AWS user account alias/the custom KMS key"
```

2. Execute the following command to view the parameter metadata.

```
aws ssm describe-parameters --filters "Key=Name,Values=the name that you specified"
```

3. Execute the following command to change the parameter value.

   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" --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"
```

Systems Manager Run Command

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 EC2 console, the AWS Command Line Interface, 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.

Topic	Details
Tutorial: Remotely Manage Your Amazon EC2 Instances (Amazon EC2 User Guide)	(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).
Systems Manager Prerequisites (p. 4)	(Required) Verify that your instances meet the minimum requirements for Run Command, configure required roles, and install the SSM Agent.
Setting Up Systems Manager in Hybrid Environments (p. 23)	(Optional) Register on-premises servers and VMs with AWS so that you can manage them using Run Command.
Executing Commands Using Systems Manager Run Command (p. 180)	Learn how to execute a command from the EC2 console and how to execute commands to a fleet of managed instances.
Run Command Walkthroughs (p. 184)	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.

Component/Concept	Details
Systems Manager Documents	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 Systems Manager Documents (p. 29).
Commands	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.
SSM Agent	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 SSM Agent on Linux (p. 15) (Linux) and Installing SSM Agent on Windows (p. 13) (Windows).

For information about Systems Manager limits, see Amazon EC2 Systems Manager Limits. To increase limits, go to AWS Support Center and submit a limit increase request form.

Contents

Related Content

- Configuring Security Roles for Systems Manager (p. 6)
- Installing SSM Agent (p. 13)
- Configure Run Command as a CloudWatch Events Target (p. 174)

- Amazon EC2 Systems Manager API Reference

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

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 Systems Manager Plugins in the *Amazon EC2 Systems Manager API Reference*.

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. 183).

Detailed Status for Command Plugins and Invocations

Status	Details
Pending	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 **Timeout (seconds)** parameter is reached, then the status changes to `Delivery Timed Out`.

Status	Details
In Progress	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.
Delayed	The system attempted to send the command to the instance but was not successful. The system will retry again.
Success	The command or plugin execution was successfully completed. This is a terminal state.
Delivery Timed Out	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.
Execution Timed Out	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.
Failed	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.
Canceled	The command was terminated before it was completed. This is a terminal state.
Undeliverable	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.
Terminated	The parent command exceeded its `max-errors` limit and subsequent command invocations were canceled by the system. This is a terminal state.

Detailed Status for a Command

Status	Details
Pending	The command was not yet received by an agent on any instances.
In Progress	The command has been sent to at least one instance but has not reached a final state on all instances.
Delayed	The system attempted to send the command to the instance but was not successful. The system will retry again.
Success	The command attempted to execute on all specified or targeted instances, all command invocations have reached a terminal state, and the value of `max-errors` was not reached. This is a terminal state.
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.
Rate Exceeded	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.

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

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 Systems Manager Documents (p. 29).
6. In the **Target key** field, specify either "InstanceIds" or "tag:*EC2_tag_name*" (retain quotation marks for both options). 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. 180) 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 Security Roles for Systems Manager (p. 6).
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 in the *Amazon Simple Notification Service Developer Guide*.

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. 174).

To set up Amazon SNS notifications when a command changes status, you must complete the following tasks.

1. Configure Account Permissions (p. 177)
2. Create an IAM Role for Notifications (p. 178)
3. Configure Amazon SNS (p. 179)
4. Send a Command that Returns Status Notifications (p. 179)

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. 171).

- 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:

Field	Type	Description
EventTime	String	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
DocumentName	String	The name of the SSM document used to execute this command.
CommandId	String	The ID generated by Run Command after the command was sent.
ExpiresAfter	Date	If this time is reached and the command has not already started executing, it will not execute.
OutputS3BucketName	String	The Amazon Simple Storage Service (Amazon S3) bucket where the responses to the command execution should be stored.
OutputS3KeyPrefix	String	The Amazon S3 directory path inside the bucket where the responses to the command execution should be stored.
RequestedDateTime	String	The time and date the request was sent to this specific instance.
InstanceId	String	The instance targeted by the command.

Field	Type	Description
Status	String	Command status for the command.

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:

Field	Type	Description
EventTime	String	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
DocumentName	String	The name of the Systems Manager document used to execute this command.
RequestedDateTime	String	The time and date the request was sent to this specific instance.
CommandId	String	The ID generated by Run Command after the command was sent.
InstanceId	String	The instance targeted by the command.
Status	String	Command status for this invocation.

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.

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. 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:

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

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).

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**.
4. For **Command document**, choose a Systems Manager document.
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.
11. Choose **Run**.
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. 187) 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. 185) or the examples in the SSM CLI Reference.

Warning
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

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. 182) on an instance by using Run Command.

Before You Begin

Before you send a command using Run Command, verify that you have completed the following tasks.

- Verify that your instances meet Systems Manager requirements (p. 4).
- Determine which SSM document to use. SSM documents define the actions you can perform on your instances. You can use a pre-defined document, or create your own.

The following high-level procedure describes how to execute a command from the EC2 console.

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 under **Systems Manager Services**, choose **Run Command**.
3. Choose **Run a command**.
4. For **Command document**, choose a document.
5. For **Target instances**, choose one or more 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. Configure parameters or options for your SSM document.
7. For **Comment**, we recommend providing information that will help you identify this command in your list of commands.
8. 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.
9. (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.
10. (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. 175).

    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.
11. Choose **Run**, and then choose **View results**.
12. 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.
13. When the **Status** column shows **Success** or **Failed**, choose the **Output** tab.

14. 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. 184).

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.

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. For **Target instances**, choose one or more 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. (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.
7. (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.
8. For **Comment**, we recommend providing information that will help you identify this command in your list of commands.
9. 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.
10. (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.
11. (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. 175).

    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.
12. Choose **Run**, and then choose **View results**.
13. 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.
14. When the **Status** column shows **Success** or **Failed**, choose the **Output** tab.

15. 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, which is currently supported when executing commands from the AWS CLI. 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 criteria. For more information about Amazon EC2 tags, see Tagging Your Amazon EC2 Resources in the *Amazon EC2 User Guide*.

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

The `targets` parameter uses the following syntax:

```
aws ssm send-command --document-name name --targets "Key=tag:tag name,Values=tag
 value" [...]
```

Note
Example commands in this section are truncated using [...].

For example, 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" [...]
```

Using Concurrency Controls

You can control how many servers execute the command at the same time by using the `max-concurrency` parameter. 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 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 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 after the fifth error.

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" [...]
```

```
--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.

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. 171).

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. 185)
- Systems Manager Run Command Walkthrough Using the AWS Tools for Windows PowerShell (p. 187)

You can also view sample commands in the following references.

- Systems Manager AWS CLI Reference
- Systems Manager AWS Tools for Windows PowerShell 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 Security Roles for Systems Manager (p. 6).

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.

```
aws ssm describe-instance-information --instance-information-filter-list
 key=InstanceIds,valueSet=instance ID
```

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.

```
aws ssm describe-document --name "AWS-RunShellScript" --query
 "[Document.Name,Document.Description]"
```

- Use the following command to view the available parameters and details about those parameters.

```
aws ssm describe-document --name "AWS-RunShellScript" --query "Document.Parameters[*]"
```

Step 3: Send a Command Using the AWS-RunShellScript document - Example 1

Use the following command to get IP information for an instance.

```
aws ssm send-command --instance-ids "instance ID" --document-name "AWS-RunShellScript" --
comment "IP config" --parameters commands=ifconfig --output text
```

Get command information with response data

The following command uses the Command ID that was returned from the previous command to get the details and response data of the command execution. The system returns the response data if the command completed. If the command execution shows "Pending" you will need to execute this command again to see the response data.

```
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.

```
sh_command_id=$(aws ssm send-command --instance-ids "instance ID" --document-name
 "AWS-RunShellScript" --comment "Demo run shell script on Linux Instance" --parameters
 commands=whoami --output text --query "Command.CommandId")
```

Get command status

The following command uses the Command ID to get the status of the command execution on the instance. This example uses the Command ID that was returned in the previous command.

```
aws ssm list-commands  --command-id $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 Security Roles for Systems Manager (p. 6).

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-SSMComand for the AWS-RunPowerShellScript document.

```
$cancelCommandResponse=Send-SSMCommand -InstanceId @('Instance-ID','Instance-ID')
 -DocumentName AWS-RunPowerShellScript -Comment 'Demo AWS-RunPowerShellScript with
 two instances' -Parameter @{'commands'='Start-Sleep –Seconds 120; dir C:\'} Stop-
SSMCommand -CommandId $cancelCommandResponse.CommandId Get-SSMCommand -CommandId
 $cancelCommandResponse.CommandId
```

Check the command status

The following command checks the status of the Cancel command

```
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

```
Get-SSMDocumentDescription -Name "AWS-InstallApplication"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-InstallApplication" | select -ExpandProperty
 Parameters
```

Send a command using the AWS-InstallApplication document

The following command installs a version of Python on your instance in unattended mode, and logs the output to a local text file on your C: drive.

```
$installAppCommand=Send-SSMCommand -InstanceId Instance-ID -DocumentName AWS-
InstallApplication -Parameter @{'source'='https://www.python.org/ftp/python/2.7.9/
python-2.7.9.msi'; 'parameters'='/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

```
Get-SSMCommandInvocation -CommandId $installAppCommand.CommandId -Details $true
```

Get command information with response data for a specific instance

The following command returns the results of the Python installation.

```
Get-SSMCommandInvocation -CommandId $installAppCommand.CommandId -Details $true -
InstanceId Instance-ID | select -ExpandProperty CommandPlugins
```

Install a PowerShell Module Using the AWS-InstallPowerShellModule JSON Document

You can use Run Command to install PowerShell modules on an EC2 instance. For more information about PowerShell modules, see Windows PowerShell Modules.

View the description and available parameters

```
Get-SSMDocumentDescription -Name "AWS-InstallPowerShellModule"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-InstallPowerShellModule" | select -ExpandProperty
 Parameters
```

Install a PowerShell module

The following command downloads the EZOut.zip file, installs it, and then runs an additional command to install XPS viewer. Lastly, the output of this command is uploaded to an Amazon S3 bucket named demo-ssm-output-bucket.

```
$installPSCommand=Send-SSMCommand -InstanceId Instance-ID -DocumentName AWS-
InstallPowerShellModule -Parameter @{'source'='https://gallery.technet.microsoft.com/
EZOut-33ae0fb7/file/110351/1/EZOut.zip';'commands'=@('Add-WindowsFeature -name XPS-Viewer -
restart')} -OutputS3BucketName demo-ssm-output-bucket
```

Get command information per instance

The following command uses the Command ID to get the status of the command execution.

```
Get-SSMCommandInvocation -CommandId $installPSCommand.CommandId -Details $true
```

Get command information with response data for the instance

The following command returns the output of the original Send-SSMCommand for the specific command ID.

```
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.

View the description and available parameters

```
Get-SSMDocumentDescription -Name "AWS-JoinDirectoryServiceDomain"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-JoinDirectoryServiceDomain" | select -ExpandProperty
 Parameters
```

Join an instance to a domain

The following command joins the instance to the given AWS Directory Service domain and uploads any generated output to the Amazon S3 bucket.

```
$domainJoinCommand=Send-SSMCommand -InstanceId Instance-ID -DocumentName
 AWS-JoinDirectoryServiceDomain -Parameter @{'directoryId'='d-9067386b64';
 'directoryName'='ssm.test.amazon.com'; 'dnsIpAddresses'=@('172.31.38.48',
 '172.31.55.243')} -OutputS3BucketName demo-ssm-output-bucket
```

Get command information per instance

The following command uses the Command ID to get the status of the command execution.

```
Get-SSMCommandInvocation -CommandId $domainJoinCommand.CommandId -Details $true
```

Get command information with response data for the instance

This command returns the output of the original Send-SSMCommand for the specific command ID.

```
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

```
Get-SSMDocumentDescription -Name "AWS-ConfigureCloudWatch"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-ConfigureCloudWatch" | select -ExpandProperty
 Parameters
```

Send Application Logs to CloudWatch

The following command configures the instance and moves Windows Applications logs to CloudWatch.

```
$cloudWatchCommand=Send-SSMCommand -InstanceID Instance-ID -DocumentName
 'AWS-ConfigureCloudWatch' -Parameter @{'properties'='{"engineConfiguration":
 {"PollInterval":"00:00:15", "Components":[{"Id":"ApplicationEventLog",
 "FullName":"AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
 "Parameters":{"LogName":"Application", "Levels":"7"}},{"Id":"CloudWatch",
 "FullName":"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
 "Parameters":{"Region":"us-east-1", "LogGroup":"My-Log-Group",
 "LogStream":"i-1234567890abcdef0"}}], "Flows":{"Flows":
["ApplicationEventLog,CloudWatch"]}}}'}
```

Get command information per instance

The following command uses the Command ID to get the status of the command execution.

```
Get-SSMCommandInvocation -CommandId $cloudWatchCommand.CommandId -Details $true
```

Get command information with response data for a specific instance

The following command returns the results of the Amazon CloudWatch configuration.

```
Get-SSMCommandInvocation -CommandId $cloudWatchCommand.CommandId -Details $true -
InstanceId Instance-ID | select -ExpandProperty CommandPlugins
```

Send Performance Counters to CloudWatch Using the AWS-ConfigureCloudWatch document

The following demonstration command uploads performance counters to CloudWatch. For more information, see the Amazon CloudWatch Documentation.

```
$cloudWatchMetricsCommand=Send-SSMCommand -InstanceID Instance-ID -DocumentName
 'AWS-ConfigureCloudWatch' -Parameter @{'properties'='{"engineConfiguration":
 {"PollInterval":"00:00:15", "Components":[{"Id":"PerformanceCounter",
 "FullName":"AWS.EC2.Windows.CloudWatch.PerformanceCounterComponent.PerformanceCounterInputComponent,AW
 "Parameters":{"CategoryName":"Memory", "CounterName":"Available
 MBytes", "InstanceName":"", "MetricName":"AvailableMemory",
 "Unit":"Megabytes","DimensionName":"", "DimensionValue":""}},{"Id":"CloudWatch",
 "FullName":"AWS.EC2.Windows.CloudWatch.CloudWatch.CloudWatchOutputComponent,AWS.EC2.Windows.CloudWatch
 "Parameters":{"AccessKey":"", "SecretKey":"","Region":"us-east-1", "NameSpace":"Windows-
Default"}}], "Flows":{"Flows":["PerformanceCounter,CloudWatch"]}}}'}
```

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. The AWS-ConfigureWindowsUpdate document is compatible with Windows Server 2008, 2008 R2, 2012, 2012 R2.

View the description and available parameters

```
Get-SSMDocumentDescription –Name "AWS-ConfigureWindowsUpdate"
```

View more information about parameters

```
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.

```
$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

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-FindWindowsUpdates' -Parameters
 @{'UpdateLevel'='All'}
```

Install specific Windows updates

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-InstallSpecificWindowsUpdates' -
Parameters @{'KbArticleIds'='123456,KB567890,987654'}
```

Install important missing Windows updates

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-InstallMissingWindowsUpdates' -
Parameters @{'UpdateLevel'='Important'}
```

Install missing Windows updates with specific exclusions

```
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 SSM Agent on Linux (p. 15) (for Linux) or Installing SSM Agent on Windows (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 Security Roles for Systems Manager (p. 6).

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)

Using the Health API on Windows Instances

Use the following command to get status details about one or more instances:

```
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:

```
Get-SSMInstanceInformation -InstanceInformationFilterList
 @{Key="PingStatus";ValueSet="Online"}
```

Use the following command to see which instances are running the latest version of the EC2Config service. Substitute ValueSet="LATEST" with a specific version (for example, 3.0.54 or 3.10) to view those details:

```
Get-SSMInstanceInformation -InstanceInformationFilterList
 @{Key="AgentVersion";ValueSet="LATEST"}
```

Using the Health API on Linux Instances

Use the following command to get status details about one or more instances:

```
aws ssm describe-instance-information --instance-information-filter-list
 key=InstanceIds,valueSet=instance-ID
```

Use the following command with no filters to see all instances registered to your account that are currently reporting an online status. Substitute the ValueSet="Online" with "ConnectionLost" or "Inactive" to view those statuses:

```
aws ssm describe-instance-information --instance-information-filter-list
 key=PingStatus,valueSet=Online
```

Use the following command to see which instances are running the latest version of the SSM agent. Substitute ValueSet="LATEST" with a specific version (for example, 1.0.145 or 1.0) to view those details:

```
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.

- The SSM agent is not installed.
- The instance does not have outbound internet connectivity.
- The instance was not launched with an IAM role that enables it to communicate with the SSM API, or the permissions for the IAM role are not correct for Run Command. For more information, see Configuring Security Roles for Systems Manager (p. 6).

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

On Linux

- /var/log/amazon/ssm/amazon-ssm-agent.log
- /var/log/amazon/ssm/error.log

On Linux, you can enable extended logging by updating the seelog.xml file. By default, the configuration file is located here: /opt/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.

Systems Manager State Management

Systems Manager State Manager is a secure and scalable service that automates the process of keeping your Amazon EC2 and hybrid infrastructure in a state that you define. You can use State Manager to ensure that your instances are bootstrapped with specific software at startup, configured according to your security policy, joined to a Windows domain, or patched with specific software updates throughout their lifecycle. You can also use State Manager to execute Linux shell scripts or Windows PowerShell scripts at different times during the lifecycle of an instance.

State Manager integrates with AWS CloudTrail to keep an audit trail of all association executions.

How It Works

You start by specifying the state you want to apply to your managed instances (for example, applications to bootstrap or network settings to configure) in a Systems Manager command or policy document. These documents are written in JSON and are called simply *documents*. Next, you bind the document to targets by using the AWS CLI or the Amazon EC2 console. You can target instance IDs or EC2 tags. The binding of the document to a target is called an association. After you associate your instance with a specific policy document, the instance remains in the state that you want because State Manager reapplies the state defined in the associated document according to the schedule that you define.

Getting Started with State Manager

To get started with State Manager, complete the following tasks.

Task	For More Information
Update the SSM Agent on your managed instances to the latest version.	Installing SSM Agent (p. 13)
Configure your on-premises servers and VMs for Systems Manager. After you configure them, they are described as *managed instances*.	Setting Up Systems Manager in Hybrid Environments (p. 23)
Verify Systems Manager prerequisites.	Systems Manager Prerequisites (p. 4)

Task	For More Information
Create a policy document that defines the actions to perform on your instances.	Creating Systems Manager Documents (p. 35)
Create and apply the association to your instances.	State Manager Associations (p. 199)

State Manager Associations

After you define the actions to perform on your instances in a policy document, you create an association. An association binds a policy document and one or more targets. Any actions defined in the document will be applied to instances when *you create* the association and when the association runs during scheduled times. You can create an association using the Amazon EC2 console, the AWS CLI, AWS Tools for Windows PowerShell, or the AWS SDKs. For examples of how to create and use associations using the Amazon EC2 console and the AWS CLI, see Systems Manager State Manager Walkthroughs (p. 200).

When you create an association, specify the following items.

- A policy document to use.
- The instances that should be associated with the policy document. You choose instances by manually selecting them, or by using the Targets option, which locates instances using EC2 tags.
- A schedule, which specifies how often the association should run.
- Parameters to execute when applying the association.
- An Amazon S3 bucket where the output should be written.

Scheduling and Running Associations

State Manager processes association tasks when you create the association. You can also run tasks on demand or set a schedule when the association should be reapplied. If you set a schedule, you can still run the association on demand.

Note
If a new association is scheduled to run while an earlier association is still running, the earlier association will be timed out and the new association will execute.

Your instances are accessible while associations are running.

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. 183). For more information about Amazon EC2 tags, see Tagging Your Amazon EC2 Resources in the *Amazon EC2 User Guide*.

The following AWS CLI examples show you how to use the `targets` parameter when creating associations. The example commands have been truncated using [...].

Create an association for all the database servers (hosts with a tag named "Database" regardless of tag value).

```
aws ssm create-association --name value --targets "Key=tag:Database"[...]
```

Create an association for a managed instance named "ws-0123456789012345"

```
aws ssm create-association --name value --targets "Key=Instance Ids,Values=ws-0123456789"}
 [...]
```

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

Use the following walkthroughs to manage the state of an EC2 instance in a test environment.

Contents

Launch a New Instance

Instances require an AWS Identity and Access Management (IAM) role that enables the instance to communicate with State Manager (SSM). The following procedure creates an instance with the required SSM-supported role.

To create an instance that uses an SSM-supported 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 Amazon Machine Image (AMI).
4. Choose your instance type and then choose **Next: Configure Instance Details**.
5. In **Auto-assign Public IP**, choose **Enable**.
6. Beside **IAM role** choose **Create new IAM role**. The IAM console opens in a new tab.
   a. Choose **Create New Role**.
   b. In **Step 1: Set Role Name**, enter a name that identifies this role as a Systems Manager role.
   c. In **Step 2: Select Role Type**, choose **Amazon EC2 Role for Simple Systems Manager**. The system skips **Step 3: Establish Trust** because this is a managed policy.
   d. In **Step 4: Attach Policy**, choose **AmazonEC2RoleforSSM**.
   e. Choose **Next Step**, and then choose **Create Role**.
   f. Close the tab with the IAM console.
7. In the Amazon EC2 console, choose the **Refresh** button beside **Create New IAM role**.
8. From **IAM role**, choose the role you just created.
9. Complete the wizard to launch the new instance. Make a note of the instance ID. You will need to specify this ID later in this tutorial.

Important
On Linux instance, you must install the SSM Agent on the instance you just created. For more information, see Installing SSM Agent on Linux (p. 15).

To assign the role to one of your existing instances, see Attaching an IAM Role to an Instance in the *Amazon EC2 User Guide*.

Grant Your User Account Access to SSM

Your user account must be configured to communicate with the SSM API. Use the following procedure to attach a managed IAM policy to your user account that grants you full access to SSM API actions.

To create the IAM policy for your user account

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**. (If this is your first time using IAM, choose **Get Started**, and then choose **Create Policy**.)
3. In the **Filter** field, type `AmazonSSMFullAccess` and press Enter.
4. Select the check box next to **AmazonSSMFullAccess** and then choose **Policy Actions**, **Attach**.
5. On the **Attach Policy** page, choose your user account and then choose **Attach Policy**.

Systems Manager State Manager Console Walkthrough

The following procedure walks you through the process of creating an association using the EC2 console.

To create an association using State Manager

1. Open the Amazon EC2 console and choose **Systems Manager Shared Resources** in the navigation pane.
2. Choose **Documents** and then choose **Create Document**.
3. For **Name**, type a descriptive name that identifies this document as a test policy document.
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 in the **Content** field.

   The following is a sample of a basic policy document that defines the schema to use and a main step that uses the aws:runShellScript plugin to get network adapter information. A policy document can have multiple steps.

```
{
   "schemaVersion": "2.0",
   "description": "Sample version 2.0 document v2",
   "parameters": {

   },
   "mainSteps": [
      {
         "action": "aws:runShellScript",
         "name": "runShellScript",
         "inputs": {
            "runCommand": [
               "ifconfig"
            ]
         }
      }
   ]
}
```

6. Choose **Create document**, and then choose **OK** after the system creates the policy document.
7. In the EC2 console navigation pane, expand **Systems Manager Services**, and then choose **State Manager**.
8. Choose **Create Association**.
9. In the **Document name** list, choose the document you just created.
10. In the **Select Targets by** section, choose **Manually Selecting Instances**, and then choose the instance you created at the beginning of this walkthrough.
11. In the **Schedule** section, choose an option.
12. Disregard the **Specify Parameters** section, as the test policy document does not take parameters.
13. Choose **Create Association**.

Systems Manager State Manager CLI Walkthrough

The following procedure walks you through the process of creating an association using the AWS Command Line Interface (AWS CLI).

1. Copy one of the following sample policy documents and paste it into a simple text editor like Notepad.

   Linux

```
{
   "schemaVersion": "2.0",
   "description": "Sample version 2.0 document v2",
   "parameters": {

   },
   "mainSteps": [
      {
         "action": "aws:runShellScript",
         "name": "runShellScript",
         "inputs": {
            "runCommand": [
               "ifconfig"
            ]
         }
      }
   ]
}
```

   Windows

```
{
   "schemaVersion": "2.0",
   "description": "Sample version 2.0 document v2",
   "parameters": {

   },
   "mainSteps": [
      {
         "action": "aws:runPowerShellScript",
         "name": "runShellScript",
         "inputs": {
            "runCommand": [
               "ipconfig"
            ]
         }
      },
      {
         "action": "aws:applications",
         "name": "installapp",
         "inputs": {
            "action": "Install",
            "source": "http://dev.mysql.com/get/Downloads/MySQLInstaller/mysql-
installer-community-5.6.22.0.msi"
         }
      }
   ]
}
```

2. Save the document with a descriptive name and a `.json` file extension.
3. Execute the following command to create the document and save it with your AWS user account using the AWS CLI.

```
aws ssm create-document --content file://c:\temp\your file --name "document name"
```

4. Execute the following command to create an association with the instance you created at the start of this walkthrough. The `Schedule` parameter sets a schedule to run the association every 30 minutes.

```
aws ssm create-association --targets Key=instanceids,Values=Instance ID --document your
 document name --schedule "cron(0 0/30 * 1/1 * ? *)"
```

5. Execute the following command to view the associations for the instance. Copy the association ID returned by the command. You'll specify this ID in the next step.

```
aws ssm  list-instance-associations --instance-id=Instance ID
```