AWS Service Catalog

Administrator Guide

AWS Service Catalog: Administrator Guide

Table of Contents

What Is AWS Service Catalog? ... 1
- Overview ... 1
  - Users ... 1
  - Products ... 2
  - Provisioned Products ... 2
  - Portfolios ... 2
  - Versioning ... 2
  - Permissions ... 2
  - Constraints ... 3
  - Initial Administrator Workflow ... 3
  - Initial End User Workflow ... 3
- Quotas ... 4
  - App registry ... 4
  - AWS Organizations ... 4
  - Constraint quotas ... 5
  - Portfolio quotas ... 5
  - Product quotas ... 5
  - Provisioned product quotas ... 5
  - Regional quotas ... 5
  - Service action quotas ... 5
  - TagOptions quotas ... 5

Setting Up ... 6
- Sign Up for Amazon Web Services ... 6
- Grant Permissions to Administrators and End Users ... 6
- Grant Permissions to Administrators ... 6
- Grant Permissions to End Users ... 8

Getting Started ... 9
- Step 1: Download the Template ... 9
  - Template Download ... 9
  - Template Overview ... 9
- Step 2: Create a Key Pair ... 12
- Step 3: Create a Portfolio ... 12
- Step 4: Create a Product ... 13
- Step 5: Add a Template Constraint ... 13
- Step 6: Add a Launch Constraint ... 14
- Step 7: Grant End Users Access to the Portfolio ... 15
- Step 8: Test the End User Experience ... 15

Getting Started Library ... 17
- Prerequisites ... 17
- Reference Architectures ... 17
- High Reliability Architectures ... 17
- Learn More ... 18

Security ... 19
- Data Protection ... 19
  - Protecting Data with Encryption ... 20
- Identity and Access Management ... 20
  - Audience ... 21
  - Predefined AWS Managed Policies ... 21
  - Identity-based policy examples for AWS Service Catalog ... 22
  - Troubleshooting AWS Service Catalog identity and access ... 26
  - Controlling Access ... 28
  - Using Service-Linked Roles for AWS Service Catalog AppRegistry ... 28
- Logging and Monitoring ... 31
- Compliance Validation ... 31

Resilience 31
Infrastructure Security 32
Security Best Practices 32
Managing Catalogs 33
Managing Portfolios 33
Creating, Viewing, and Deleting Portfolios 33
Viewing Portfolio Details 34
Creating and Deleting Portfolios 34
Adding Products 34
Adding Constraints 36
Granting Access to Users 37
Sharing a Portfolio 37
Sharing and Importing Portfolios 40
Managing Products 42
Viewing the Products Page 43
Creating Products 43
Adding Products to Portfolios 44
Updating Products 44
Deleting Products 45
Managing Versions 45
Using Constraints 46
Launch Constraints 46
Notification Constraints 49
Tag Update Constraints 49
Stack Set Constraints 50
Template Constraints 50
Using Service Actions 53
Prerequisites 53
Step 1: Configure end user permissions 53
Step 2: Create a service action 54
Step 3: Associate the service action with a product version 55
Step 4: Test the end user experience 55
Step 5: Troubleshooting 55
Adding AWS Marketplace Products to Your Portfolio 57
Managing AWS Marketplace Products Using AWS Service Catalog 57
Managing and Adding AWS Marketplace Products Manually 57
Using AWS CloudFormation StackSets 62
Stack sets vs. stack instances 62
Stack set constraints 62
Managing Budgets 62
Prerequisites 62
Creating a Budget 64
Associating a Budget 64
Viewing a Budget 65
Disassociating a Budget 65
Managing Provisioned Products 66
Managing All Provisioned Products as Administrator 66
Changing Provisioned Product Owner 66
See Also 67
Tutorial: Identifying User Resource Allocation 67
Managing Tags 71
AutoTags 71
TagOption Library 72
Launching a Product with TagOptions 72
Managing TagOptions 75
Monitoring 77
Monitoring Tools 77

Automated Tools ... 77
CloudWatch Metrics ... 77
Enabling CloudWatch Metrics ... 78
Available Metrics and Dimensions ... 78
Viewing AWS Service Catalog Metrics ... 79
CloudTrail logs ... 79
AWS Service Catalog information in CloudTrail ... 79
Understanding AWS Service Catalog log file entries ... 80
Using AppRegistry ... 4
Creating applications ... 82
Accessing application details ... 83
Editing applications ... 83
Deleting applications ... 84
Associating application resources ... 84
Associating attribute groups ... 85
Creating attribute groups ... 85
Accessing attribute group details ... 85
Associating attribute groups ... 86
Adding tags ... 88
Product and Service Integrations ... 90
Connector for ServiceNow ... 90
Background ... 91
Getting started ... 91
Release notes ... 92
Baseline permissions ... 93
Configuring AWS Service Catalog ... 98
Configuring AWS Config ... 99
Configuring AWS Security Hub ... 99
Configuring AWS Systems Manager OpsCenter ... 100
Configuring ServiceNow ... 100
Configuring AWS Config integration in ServiceNow ... 111
Configuring AWS Systems Manager OpsCenter integration in ServiceNow ... 117
Validating configurations ... 119
ServiceNow additional features ... 124
Version 2.3.4 release transition instructions ... 127
Connector for Jira Service Management ... 128
Background ... 129
Jira Service Management Supported Versions and Releases ... 129
Getting Started ... 130
Release Notes ... 131
Baseline Permissions ... 132
Configuring AWS Service Catalog ... 137
Configuring Jira Service Management ... 138
IT Lifecycle Management Setup and Use Case ... 144
Validating Configurations ... 149
Jira Additional Administrator Features ... 152
Document History ... 153

What Is AWS Service Catalog?

AWS Service Catalog enables organizations to create and manage catalogs of IT services that are approved for AWS. These IT services can include everything from virtual machine images, servers, software, databases, and more to complete multi-tier application architectures.

AWS Service Catalog allows organizations to centrally manage commonly deployed IT services, and helps organizations achieve consistent governance and meet compliance requirements. End users can quickly deploy only the approved IT services they need, following the constraints set by your organization.

AWS Service Catalog provides the following benefits:

- **Standardization**

  Administer and manage approved assets by restricting where the product can be launched, the type of instance that can be used, and many other configuration options. The result is a standardized landscape for product provisioning for your entire organization.
- **Self-service discovery and launch**

  Users browse listings of products (services or applications) that they have access to, locate the product that they want to use, and launch it all on their own as a provisioned product.
- **Fine-grain access control**

  Administrators assemble portfolios of products from their catalog, add constraints and resource tags to be used at provisioning, and then grant access to the portfolio through Amazon Identity and Access Management (IAM) users and groups.
- **Extensibility and version control**

  Administrators can add a product to any number of portfolios and restrict it without creating another copy. Updating the product to a new version propagates the update to all products in every portfolio that references it.

For more information, see the AWS Service Catalog detail page.

The AWS Service Catalog API provides programmatic control over all end-user actions as an alternative to using the AWS Management Console. For more information, see AWS Service Catalog Developer Guide.

Overview of AWS Service Catalog

As you get started with AWS Service Catalog, you'll benefit from understanding its components and the initial workflows for administrators and end users.

Users

AWS Service Catalog supports the following types of users:

- **Catalog administrators (administrators)** – Manage a catalog of products (applications and services), organizing them into portfolios and granting access to end users. Catalog administrators prepare AWS CloudFormation templates, configure constraints, and manage IAM roles that are assigned to products to provide for advanced resource management.

- **End users** – Receive AWS credentials from their IT department or manager and use the AWS Management Console to launch products to which they have been granted access. Sometimes referred to as simply *users*, end users may be granted different permissions depending on your operational requirements. For example, a user may have the maximum permission level (to launch and manage all of the resources required by the products they use) or only permission to use particular service features.

Products

A *product* is an IT service that you want to make available for deployment on AWS. A product consists of one or more AWS resources, such as EC2 instances, storage volumes, databases, monitoring configurations, and networking components, or packaged AWS Marketplace products. A product can be a single compute instance running AWS Linux, a fully configured multi-tier web application running in its own environment, or anything in between. You create a product by importing an AWS CloudFormation template. AWS CloudFormation templates define the AWS resources required for the product, the relationships between resources, and the parameters that end users can plug in when they launch the product to configure security groups, create key pairs, and perform other customizations.

Provisioned Products

AWS CloudFormation stacks make it easier to manage the lifecycle of your product by enabling you to provision, tag, update, and terminate your product instance as a single unit. An AWS CloudFormation stack includes an AWS CloudFormation template, written in either JSON or YAML format, and its associated collection of resources. A *provisioned product* is a stack. When an end user launches a product, the instance of the product that is provisioned by AWS Service Catalog is a stack with the resources necessary to run the product. For more information, see AWS CloudFormation User Guide.

Portfolios

A *portfolio* is a collection of *products*, together with configuration information. Portfolios help manage who can use specific products and how they can use them. With AWS Service Catalog, you can create a customized portfolio for each type of user in your organization and selectively grant access to the appropriate portfolio. When you add a new *version* of a product to a portfolio, that version is automatically available to all current users. You also can share your portfolios with other AWS accounts and allow the administrator of those accounts to distribute your portfolios with additional *constraints*, such as limiting which EC2 instances a user can create. Through the use of portfolios, permissions, sharing, and constraints, you can ensure that users are launching products that are configured properly for the organization's needs and standards.

Versioning

AWS Service Catalog allows you to manage multiple versions of the products in your catalog. This allows you to add new versions of templates and associated resources based on software updates or configuration changes. When you create a new version of a product, the update is automatically distributed to all users who have access to the product, allowing the user to select which version of the product to use. Users can update running instances of the product to the new version quickly and easily.

Permissions

Granting a user access to a portfolio enables that user to browse the portfolio and launch the products in it. You apply AWS Identity and Access Management (IAM) permissions to control who can view and modify your catalog. IAM permissions can be assigned to IAM users, groups, and roles. When a user launches a product that has an IAM role assigned to it, AWS Service Catalog uses the role to launch the

product's cloud resources using AWS CloudFormation. By assigning an IAM role to each product, you can avoid giving users permissions to perform unapproved operations and enable them to provision resources using the catalog.

Constraints

Constraints control the ways that specific Amazon Web Services resources can be deployed for a product. You can use them to apply limits to products for governance or cost control. There are different types of Amazon Service Catalog constraints: *launch constraints*, *notification constraints*, and *template constraints*.

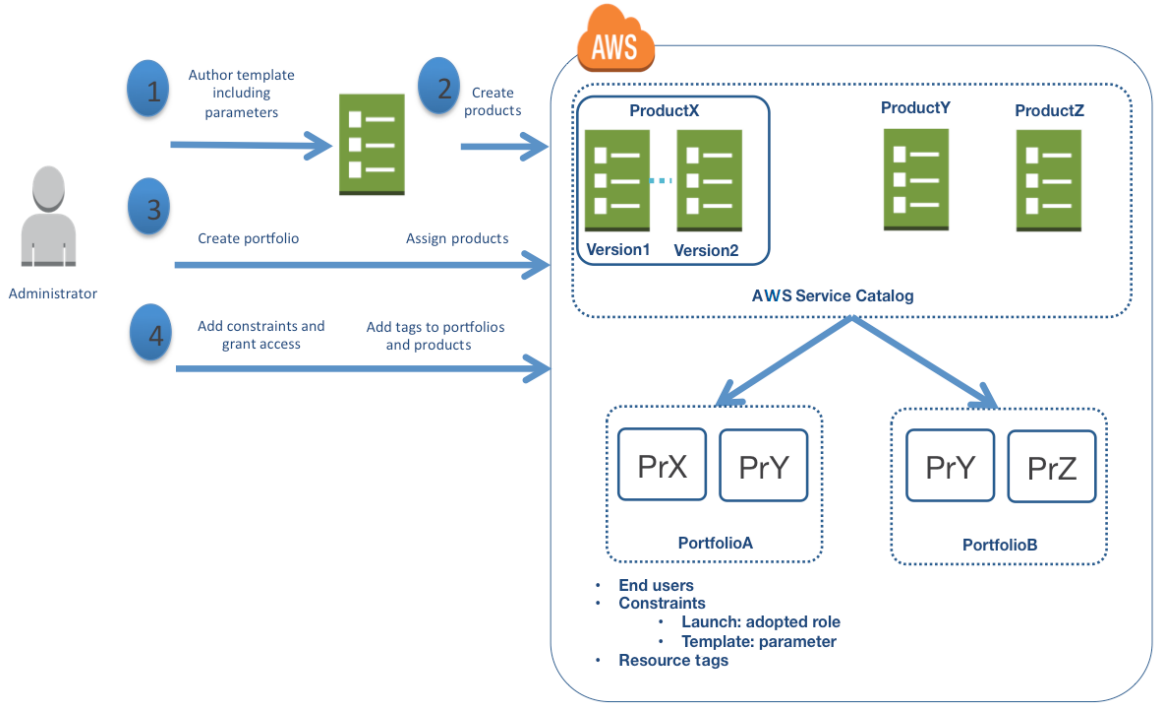
With launch constraints, you specify a role for a product in a portfolio. This role is used to provision the resources at launch, so you can restrict user permissions without impacting users' ability to provision products from the catalog.

Notification constraints enable you to get notifications about stack events using an Amazon SNS topic.

Template constraints restrict the configuration parameters that are available for the user when launching the product (for example, EC2 instance types or IP address ranges). With template constraints, you reuse generic AWS CloudFormation templates for products and apply restrictions to the templates on a per-product or per-portfolio basis.

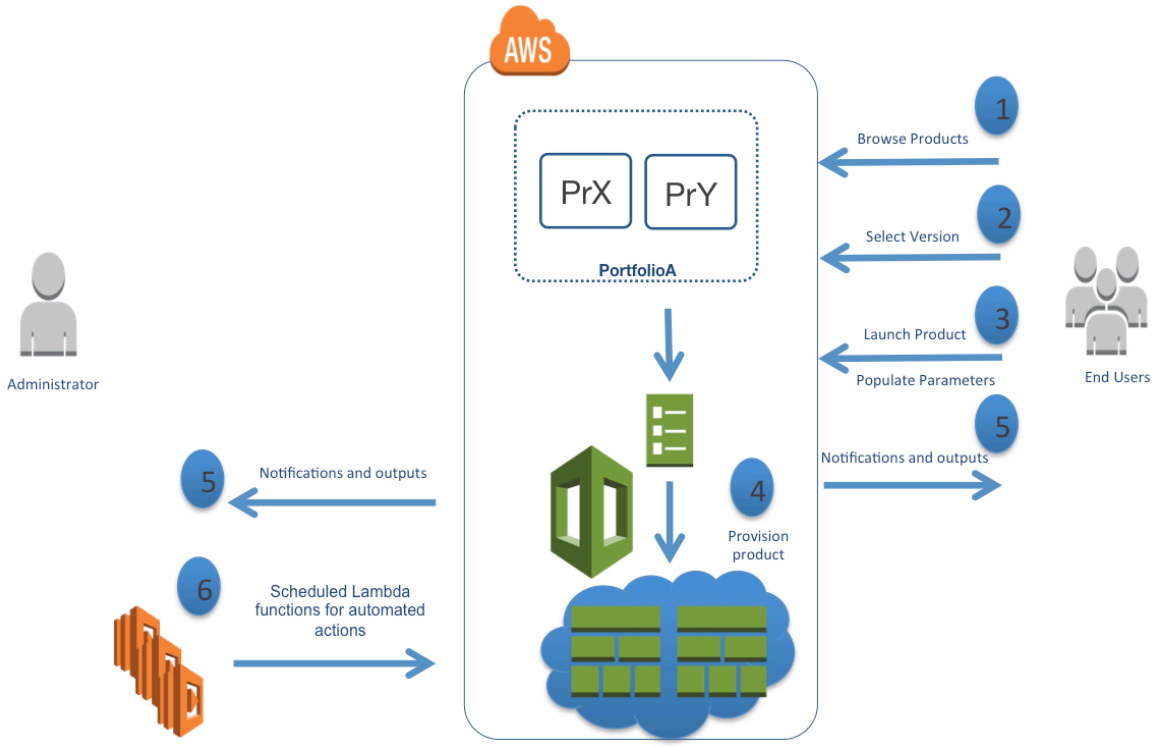
Initial Administrator Workflow

The following diagram shows the initial workflow for an administrator when creating a catalog.



Initial End User Workflow

Using the state of the administrator workflow as a starting point, the following diagram shows the initial workflow for an end user. This example shows the end user product view and provisioning tasks, on the right, as well as the administrator's tasks, on the left. The tasks are numbered in order.



AWS Service Catalog default service quotas

Your AWS account has the following default quotas related to AWS Service Catalog for App registry, AWS Organizations, constraint, portfolio, product, provisioned product, regional, service action, and TagOptions.

You can use AWS Service Quotas to manage your quotas or to request a quota increase. For more information about Service Quotas, see What Is Service Quotas? in the *Service Quotas User Guide*. To learn how to request a quota increase, see Requesting a Quota Increase.

App registry

- Applications per account and region: 100
- Attribute groups per account and region: 100
- Associated resources per application: 200
- Associated attribute groups per application: 100
- Size of attribute group: 400 KB

AWS Organizations

- AWS Service Catalog delegated administrators per organization: 50

Constraint quotas

- Constraints per product per portfolio: 100

Portfolio quotas

- Users, groups, and roles per portfolio: 100
- Products per portfolio: 150
- Tags per portfolio: 20
- Shared accounts per portfolio: 5000
- Tag values per tag key: 25

Product quotas

- Users, groups, and roles per product: 200
- Product versions per product: 100
- Tags per product: 20
- Tag values per tag key: 25

Provisioned product quotas

- Tags per provisioned product: 50

Regional quotas

- Portfolios: 100
- Products: 350

Service action quotas

- Service actions per region: 200
- Service action associations per product version: 25

TagOptions quotas

- TagOptions per resource: 25
- Values per TagOption: 25

Setting Up AWS Service Catalog

Before you get started with AWS Service Catalog, complete the following tasks.

Sign Up for Amazon Web Services

To use Amazon Web Services (AWS), you will need to sign up for an AWS account.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

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

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

Grant Permissions to Administrators and End Users

Catalog administrators and end users require different IAM permissions to use AWS Service Catalog. As a catalog administrator, you must have IAM permissions that allow you to access the AWS Service Catalog administrator console, create products, and manage products. Before your end users can use your products, you must grant them permissions that allow them to access the AWS Service Catalog end user console, launch products, and manage launched products as provisioned products.

AWS Service Catalog provides many of these permissions using managed policies. AWS maintains these policies and provides them in the AWS Identity and Access Management (IAM) service. You can use these policies by attaching them to the IAM users, groups, or roles that you and your end users use.

- Identity and Access Management in AWS Service Catalog (p. 20)
- Grant Permissions to AWS Service Catalog Administrators (p. 6)
- Grant Permissions to AWS Service Catalog End Users (p. 8)

Grant Permissions to AWS Service Catalog Administrators

As a catalog administrator, you require access to the AWS Service Catalog administrator console view and IAM permissions that allow you to perform tasks such as the following:

- Creating and managing portfolios

- Creating and managing products
- Adding template constraints to control the options that are available to end users when launching a product
- Adding launch constraints to define the IAM roles that AWS Service Catalog assumes when end users launch products
- Granting end users access to your products

You, or an administrator who manages your IAM permissions, must attach policies to your IAM user, group, or role that are required to complete this tutorial.

To grant permissions to a catalog administrator

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users**. If you have already created an IAM user that you would like to use as the catalog administrator, choose the user name and choose **Add permissions**. Otherwise, create a user as follows:
   a. Choose **Add user**.
   b. For **User name**, type **`ServiceCatalogAdmin`**.
   c. Select **Programmatic access** and **AWS Management Console access**.
   d. Choose **Next: Permissions**.
3. Choose **Attach existing policies directly**.
4. Choose **Create policy** and do the following:
   a. Choose the **JSON** tab.
   b. Copy the following example policy and paste it in **Policy Document**:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateKeyPair",
                "iam:AddRoleToInstanceProfile",
                "iam:AddUserToGroup",
                "iam:AttachGroupPolicy",
                "iam:CreateAccessKey",
                "iam:CreateGroup",
                "iam:CreateInstanceProfile",
                "iam:CreateLoginProfile",
                "iam:CreateRole",
                "iam:CreateUser",
                "iam:Get*",
                "iam:List*",
                "iam:PutRolePolicy",
                "iam:UpdateAssumeRolePolicy"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

   c. Choose **Review policy**.
   d. For **Policy Name**, type **`ServiceCatalogAdmin-AdditionalPermissions`**.

e. You must grant administrators permissions for Amazon S3 so they can access templates stored by AWS Service Catalog in Amazon S3. For more information, see User Policy Examples in the *Amazon Simple Storage Service Developer Guide*
f. Choose **Create Policy**.

5. Return to the browser window with the permissions page and choose **Refresh**.
6. In the search field, type **`ServiceCatalog`** to filter the policy list.
7. Select the checkboxes for the **AWSServiceCatalogAdminFullAccess** and **ServiceCatalogAdmin-AdditionalPermissions** policies, and then choose **Next: Review**.
8. If you are updating a user, choose **Add permissions**.

   If you are creating a user, choose **Create user**. You can download or copy the credentials and then choose **Close**.
9. To sign in as the catalog administrator, use your account-specific URL. To find this URL, choose **Dashboard** in the navigation pane and choose **Copy Link**. Paste the link in your browser, and use the name and password of the IAM user you created or updated in this procedure.

Grant Permissions to AWS Service Catalog End Users

Before the end user can use AWS Service Catalog, you must grant access to the AWS Service Catalog end user console view. To grant access, you attach policies to the IAM user, group, or role that is used by the end user. In the following procedure, we attach the **AWSServiceCatalogEndUserFullAccess** policy to an IAM group. For more information, see Predefined AWS Managed Policies (p. 21).

To grant permissions to an end user group

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Groups**.
3. Choose **Create New Group** and do the following:

   a. For **Group Name**, type **`Endusers`**, and then choose **Next Step**.
   b. In the search field, type **`AWSServiceCatalog`** to filter the policy list.
   c. Select the checkbox for the **AWSServiceCatalogEndUserFullAccess** policy, and then choose **Next Step**. You also have the option to choose **AWSServiceCatalogEndUserReadOnlyAccess** instead.
   d. On the **Review page**, choose **Create Group**.
4. In the navigation pane, choose **Users**.
5. Choose **Add user** and do the following:

   a. For **User name**, type a name for the user.
   b. Select **AWS Management Console access**.
   c. Choose **Next: Permissions**.
   d. Choose **Add user to group**.
   e. Select the checkbox for the **Endusers** group and choose **Next: Tags** and then **Next: Review**.
   f. On the **Review** page, choose **Create user**. Download or copy the credentials and then choose **Close**.

Getting Started

This tutorial introduces you to the key tasks that you do as a catalog administrator. You create a product that is based on an AWS CloudFormation template, which defines the AWS resources used by the product. The product, Linux Desktop, is a cloud development environment that runs on Amazon Linux. You add the product to a portfolio and distribute it to the end user. Finally, you log in as the end user to test the product.

Before You Begin

Complete the tasks described in Setting Up AWS Service Catalog (p. 6).

Tasks

- Step 1: Download the AWS CloudFormation Template (p. 9)
- Step 2: Create a Key Pair (p. 12)
- Step 3: Create an AWS Service Catalog Portfolio (p. 12)
- Step 4: Create an AWS Service Catalog Product (p. 13)
- Step 5: Add a Template Constraint to Limit Instance Size (p. 13)
- Step 6: Add a Launch Constraint to Assign an IAM Role (p. 14)
- Step 7: Grant End Users Access to the Portfolio (p. 15)
- Step 8: Test the End User Experience (p. 15)

Step 1: Download the AWS CloudFormation Template

To provision and configure portfolios and products, you use AWS CloudFormation templates, which are JSON– or YAML-formatted text files. For more information, see Template Formats in the *AWS CloudFormation User Guide*. These templates describe the resources that you want to provision. You can use the AWS CloudFormation editor or any text editor to create and save templates. For this tutorial, we've provided a simple template to get you started. This template launches a single Linux instance configured for SSH access.

Template Download

The sample template provided for this tutorial, `development-environment.template`, is available at https://awsdocs.s3.amazonaws.com/servicecatalog/development-environment.template.

Template Overview

The text of the sample template follows:

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS Service Catalog sample template. Creates an Amazon EC2 instance
                   running the Amazon Linux AMI. The AMI is chosen based on the region
                   in which the stack is run. This example creates an EC2 security
                   group for the instance to give you SSH access. **WARNING** This
                   template creates an Amazon EC2 instance. You will be billed for the
```

```
                    AWS resources used if you create a stack from this template.",

 "Parameters" : {
   "KeyName": {
     "Description" : "Name of an existing EC2 key pair for SSH access to the EC2
instance.",
     "Type": "AWS::EC2::KeyPair::KeyName"
   },

   "InstanceType" : {
     "Description" : "EC2 instance type.",
     "Type" : "String",
     "Default" : "t2.micro",
     "AllowedValues" : [ "t2.micro", "t2.small", "t2.medium", "m3.medium", "m3.large",
       "m3.xlarge", "m3.2xlarge" ]
   },

   "SSHLocation" : {
     "Description" : "The IP address range that can SSH to the EC2 instance.",
     "Type": "String",
     "MinLength": "9",
     "MaxLength": "18",
     "Default": "0.0.0.0/0",
     "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
     "ConstraintDescription": "Must be a valid IP CIDR range of the form x.x.x.x/x."
  }
 },

 "Metadata" : {
   "AWS::CloudFormation::Interface" : {
     "ParameterGroups" : [{
       "Label" : {"default": "Instance configuration"},
       "Parameters" : ["InstanceType"]
     },{
       "Label" : {"default": "Security configuration"},
       "Parameters" : ["KeyName", "SSHLocation"]
     }],
     "ParameterLabels" : {
       "InstanceType": {"default": "Server size:"},
       "KeyName": {"default": "Key pair:"},
       "SSHLocation": {"default": "CIDR range:"}
     }
   }
 },

 "Mappings" : {
   "AWSRegionArch2AMI" : {
     "us-east-1"      : { "HVM64" : "ami-08842d60" },
     "us-west-2"      : { "HVM64" : "ami-8786c6b7" },
     "us-west-1"      : { "HVM64" : "ami-cfa8a18a" },
     "eu-west-1"      : { "HVM64" : "ami-748e2903" },
     "ap-southeast-1" : { "HVM64" : "ami-d6e1c584" },
     "ap-northeast-1" : { "HVM64" : "ami-35072834" },
     "ap-southeast-2" : { "HVM64" : "ami-fd4724c7" },
     "sa-east-1"      : { "HVM64" : "ami-956cc688" },
     "cn-north-1"     : { "HVM64" : "ami-ac57c595" },
     "eu-central-1"   : { "HVM64" : "ami-b43503a9" }
   }

 },

 "Resources" : {
   "EC2Instance" : {
     "Type" : "AWS::EC2::Instance",
     "Properties" : {
       "InstanceType" : { "Ref" : "InstanceType" },
```

```
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "KeyName" : { "Ref" : "KeyName" },
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region" },
"HVM64" ] }
      }
    },

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : { "Ref" : "SSHLocation"}
        } ]
      }
    }
  },

  "Outputs" : {
    "PublicDNSName" : {
      "Description" : "Public DNS name of the new EC2 instance",
      "Value" : { "Fn::GetAtt" : [ "EC2Instance", "PublicDnsName" ] }
    },
    "PublicIPAddress" : {
      "Description" : "Public IP address of the new EC2 instance",
      "Value" : { "Fn::GetAtt" : [ "EC2Instance", "PublicIp" ] }
    }
  }
}
```

Template Resources

The template declares resources to be created when the product is launched. It consists of the following sections:

- **AWSTemplateFormatVersion** – The version of the AWS Template Format used to create this template.
- **Description** – A description of the template.
- **Parameters** – The parameters that your user must specify to launch the product. For each parameter, the template includes a description and constraints that must be met by the value typed. For more information about constraints, see Using AWS Service Catalog Constraints (p. 46).

  The `KeyName` parameter allows you to specify an Amazon Elastic Compute Cloud (Amazon EC2) key pair name that end users must provide when they use AWS Service Catalog to launch your product. You will create the key pair in the next step.
- **Metadata** – An optional section that defines details about the template. The AWS::CloudFormation::Interface key defines how the end user console view displays parameters. The `ParameterGroups` property defines how parameters are grouped and headings for those groups. The `ParameterLabels` property defines friendly parameter names. When a user is specifying parameters to launch a product that is based on this template, the end user console view displays the parameter labeled `Server size:` under the heading `Instance configuration`, and it displays the parameters labeled `Key pair:` and `CIDR range:` under the heading `Security configuration`.
- **Mappings** – A list of regions and the Amazon Machine Image (AMI) that corresponds to each. AWS Service Catalog uses the mapping to determine which AMI to use based on the region that the user selects in the AWS Management Console.
- **Resources** – An EC2 instance running Amazon Linux and a security group that allows SSH access to the instance. The `Properties` section of the EC2 instance resource uses the information that the user types to configure the instance type and a key name for SSH access.

AWS CloudFormation uses the current region to select the AMI ID from the mappings defined earlier and assigns a security group to it. The security group is configured to allow inbound access on port 22 from the CIDR IP address range that the user specifies.

- **Outputs** – Text that tells the user when the product launch is complete. The provided template gets the public DNS name of the launched instance and displays it to the user. The user needs the DNS name to connect to the instance using SSH.

Step 2: Create a Key Pair

To enable your end users to launch the product that is based on the sample template for this tutorial, you must create an Amazon EC2 key pair. A key pair is a combination of a public key that is used to encrypt data and a private key that is used to decrypt data. For more information about key pairs, see Amazon EC2 Key Pairs in the *Amazon EC2 User Guide for Linux Instances*.

The AWS CloudFormation template for this tutorial, `development-environment.template`, includes the `KeyName` parameter:

```
. . .
  "Parameters" : {
    "KeyName": {
      "Description" : "Name of an existing EC2 key pair for SSH access to the EC2
 instance.",
      "Type": "AWS::EC2::KeyPair::KeyName"
    },
. . .
```

End users must specify the name of a key pair when they use AWS Service Catalog to launch the product that is based on the template.

If you already have a key pair in your account that you would prefer to use, you can skip ahead to Step 3: Create an AWS Service Catalog Portfolio (p. 12). Otherwise, complete the following steps.

To create a key pair

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Network & Security**, choose **Key Pairs**.
3. On the **Key Pairs** page, choose **Create Key Pair**.
4. For **Key pair name**, type a name that is easy for you to remember, and then choose **Create**.
5. When the console prompts you to save the private key file, save it in a safe place.

   Important
   This is the only chance for you to save the private key file.

Step 3: Create an AWS Service Catalog Portfolio

To provide users with products, begin by creating a portfolio for those products.

To create a portfolio

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. If you are using the AWS Service Catalog administrator console for the first time, choose **Launch solutions with the Getting Started library** to start the wizard for configuring a portfolio. Otherwise, choose **Create portfolio**.

3. Type the following values:
   - **Portfolio name** – **`Engineering Tools`**
   - **Description** – **`Sample portfolio that contains a single product.`**
   - **Owner** – **`IT (it@example.com)`**
4. Choose **Create**.

Step 4: Create an AWS Service Catalog Product

After you have created a portfolio, you're ready to add a product. For this tutorial, you will create a product called Linux Desktop, a cloud development environment that runs on Amazon Linux.

To create a product

1. If you've just completed the previous step, the **Portfolios** page is already displayed. Otherwise, open https://console.aws.amazon.com/servicecatalog/.
2. Choose and open a portfolio. Next choose a product and then choose **Upload new product**.
3. On the **Enter product details** page, type the following and then choose **Next**:
   - **Product name** – **`Linux Desktop`**
   - **Description** – **`Cloud development environment configured for engineering staff. Runs AWS Linux.`**
   - **Owner** – **`IT`**
   - **Vendor** – *(blank)*
4. On the **Version details** page, choose **Specify an Amazon S3 template URL**, type the following, and then choose **Next**:
   - **Select template** – **`https://awsdocs.s3.amazonaws.com/servicecatalog/development-environment.template`**
   - **Version title** – **`v1.0`**
   - **Description** – **`Base Version`**
5. On the **Enter support details** page, type the following and then choose **Next**:
   - **Email contact** – **`ITSupport@example.com`**
   - **Support link** – **`https://wiki.example.com/IT/support`**
   - **Support description** – **`Contact the IT department for issues deploying or connecting to this product.`**
6. On the **Review** page, choose **Create product**.

Step 5: Add a Template Constraint to Limit Instance Size

Constraints add another layer of control over products at the portfolio level. Constraints can control the launch context of a product (launch constraints), or add rules to the AWS CloudFormation template (template constraints). For more information, see Using AWS Service Catalog Constraints (p. 46).

Now add a template constraint to the Linux Desktop product that prevents users from selecting large instance types at launch time. The development-environment template allows the user to select from six instance types; this constraint limits valid instance types to the two smallest types, `t2.micro` and `t2.small`. For more information, see T2 Instances in the *Amazon EC2 User Guide for Linux Instances*.

To add a template constraint to the Linux Desktop product

1. On the portfolio details page, expand the **Constraints** section, and choose **Add constraints**.
2. In the **Select product and type** window, for **Product**, choose **Linux Desktop**. Then, for **Constraint type**, choose **Template**.
3. Choose **Continue**.
4. For **Description**, type **`Small instance sizes`**.
5. Paste the following into the **Template constraint** text box:

```
{
  "Rules": {
    "Rule1": {
      "Assertions": [
        {
          "Assert" : {"Fn::Contains": [["t2.micro", "t2.small"], {"Ref":
 "InstanceType"}]},
          "AssertDescription": "Instance type should be t2.micro or t2.small"
        }
      ]
    }
  }
}
```

6. Choose **Submit**.

Step 6: Add a Launch Constraint to Assign an IAM Role

A launch constraint designates an IAM role that AWS Service Catalog assumes when an end user launches a product.

For this step, you add a launch constraint to the Linux Desktop product so that AWS Service Catalog can use the Amazon Web Services resources that are part of the product's Amazon CloudFormation template. This launch constraint enables the end user to launch the product and, after launch, manage it as a provisioned product. For more information, see AWS Service Catalog Launch Constraints (p. 46).

Without a launch constraint, you would need to grant additional IAM permissions to your end users before they could use the Linux Desktop product. For example, the `ServiceCatalogEndUserAccess` policy grants the minimum IAM permissions required to access the AWS Service Catalog end user console view.

By using a launch constraint, you can keep your end users' IAM permissions to a minimum, which is an IAM best practice. For more information, see Grant least privilege in the *IAM User Guide*.

To add a launch constraint

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**. Choose **Create policy** and do the following:

   a. On the **Create policy** page, choose the **JSON** tab.

   b. Copy the following example policy and paste it in **Policy Document**, replacing the placeholder JSON in the text field:

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

```
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "s3:ExistingObjectTag/servicecatalog:provisioning":"true"
            }
         }
      }
   ]
}
```

   c. Choose **Review policy**.
   d. For **Policy Name**, type **`linuxDesktopPolicy`**.
   e. Choose **Create policy**.
3. In the **Template constraint** window, choose **Text editor**. Choose **Create role** and do the following:
   a. For **Select type of trusted entity**, choose **AWS service** and then choose **Service Catalog**. Select the **Service Catalog** use case and then choose **Next: Permissions**.
   b. Search for the **linuxDesktopPolicy** policy and then select the checkbox.
   c. Choose **Next: Tags**, and then **Next: Review**.
   d. For **Role name**, type **`linuxDesktopLaunchRole`**.
   e. Choose **Create role**.
4. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
5. Choose the **Engineering Tools** portfolio.
6. On the portfolio details page, choose the **Constraints** tab, and then choose **Create constraint**.
7. For **Product**, choose **Linux Desktop**, and for **Constraint type**, choose **Launch**.
8. Choose **Select IAM role**. Next choose **linuxDesktopLaunchRole**, and then choose **Create**.

Step 7: Grant End Users Access to the Portfolio

Now that you have created a portfolio and added a product, you are ready to grant access to end users.

Prerequisites

If you haven't created an IAM group for the endusers, see Grant Permissions to AWS Service Catalog End Users (p. 8).

To provide access to the portfolio

1. On the portfolio details page, choose the **Groups, roles, and users** tab.
2. Choose **Add groups, roles, users**.
3. On the **Groups** tab, select the checkbox for the IAM group for the end users.
4. Choose **Add Access**.

Step 8: Test the End User Experience

To verify the end user can successfully access the end user console view and launch your product, sign in to AWS as the end user and perform those tasks.

To verify that the end user can access the end user console

1. To sign in as the IAM user, use the account-specific URL. To find this URL, open the IAM console, choose **Dashboard** in the navigation pane, and choose **Copy to clipboard.** Paste the link in your browser, and use the name and password of the IAM user.
2. In the menu bar, choose the region in which you created the `Engineering Tools` portfolio.
3. Choose Service Catalog from the recently used services to see:

   - **Products** – The products that the user can use.
   - **Provisioned products** – The provisioned products that the user has launched.

To verify the end user can launch the Linux Desktop product

1. In the **Products** section of the console, choose **Linux Desktop**.
2. Choose **Launch product** to start the wizard that configures your product.
3. On the **Launch: Linux Desktop** page, enter **`Linux-Desktop`** for the provisioned product name.
4. On the **Parameters** page, enter the following and choose **Next**:

   - **Server size** – Choose **`t2.micro`**.
   - **Key pair** – Select the key pair that you created in Step 2: Create a Key Pair (p. 12).
   - **CIDR range** – Enter a valid CIDR range for the IP address to connect to the instance. You can use the default value (0.0.0.0/0) to allow access from any IP address, then your IP address, followed by **`/32`** to restrict access to your IP address only, or something in between.
5. Choose **Launch** to launch the stack. The console displays the stack details page for the Linux-Desktop stack. The initial status of the product is **Launching**. It takes several minutes for AWS Service Catalog to launch the product. To see the current status, refresh your browser. After the product launches, the status is **Available**.

Getting Started Library

AWS Service Catalog provides a Getting Started Library of well-architected product templates so you can get started quickly. You can copy any of the products in our Getting Started Library portfolios to your own account, then customize them to suit your needs.

Topics

- Prerequisites (p. 17)
- Reference Architectures (p. 17)
- High Reliability Architectures (p. 17)
- Learn More (p. 18)

Prerequisites

Before you use the templates in our Getting Started Library, make sure you have the following:

- The required permissions to use AWS CloudFormation templates. For more information, see Controlling Access with AWS Identity and Access Management.
- The required administrator permissions to manage AWS Service Catalog. For more information, see the section called "Identity and Access Management" (p. 20).

Reference Architectures

Our **Reference Architectures** portfolio is a general repository available to all AWS Service Catalog administrators. It contains well-architected, best practice templates for common AWS services, including:

- **Compute** - *with Amazon EC2*
- **Storage** - *with Amazon S3*
- **Networking** - *with Amazon VPC*
- **Database** - *with Amazon RDS*

To view the Reference Architectures portfolio in the administrator console

1. In the AWS Service Catalog console, choose **Portfolios**.
2. On the **Portfolios** page, choose the **Getting Started library** tab.
3. Choose the **Reference Architectures** portfolio.
4. You can browse the list of available product templates, copy them to your own portfolio, and customize them.

You can view the repository of AWS Service Catalog Reference Architectures on GitHub here: Sample AWS CloudFormation templates and architecture for AWS Service Catalog.

High Reliability Architectures

Our **High Reliability Architectures** portfolio is a repository of well-architected, multi-region blueprints. Each blueprint provides prescriptive implementation guidance for AWS services commonly used to

build multi-region workloads. Examples include patterns for managing infrastructure changes and data storage backup and recovery for user identity, key-value, and object data across multiple regions.

Learn More

- For more information about the well-architected framework, see AWS Well-Architected.

Security in AWS Service Catalog

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS Service Catalog, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS Service Catalog. The following topics show you how to configure AWS Service Catalog to meet your security and compliance objectives. You also will be introduced to other AWS services that help you to monitor and secure your AWS Service Catalog resources.

Topics

- Data Protection in AWS Service Catalog (p. 19)
- Identity and Access Management in AWS Service Catalog (p. 20)
- Logging and Monitoring in AWS Service Catalog (p. 31)
- Compliance Validation for AWS Service Catalog (p. 31)
- Resilience in AWS Service Catalog (p. 31)
- Infrastructure Security in AWS Service Catalog (p. 32)
- Security Best Practices for AWS Service Catalog (p. 32)

Data Protection in AWS Service Catalog

The AWS shared responsibility model applies to data protection in AWS Service Catalog. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.

- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with AWS Service Catalog or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into AWS Service Catalog or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

Protecting Data with Encryption

Encryption at rest

AWS Service Catalog uses Amazon S3 buckets and Amazon DynamoDB databases that are encrypted at rest using Amazon-managed keys. To learn more, refer to information about encryption at rest provided by Amazon S3 and Amazon DynamoDB.

Encryption in transit

AWS Service Catalog uses Transport Layer Security (TLS) and client-side encryption of information in transit between the caller and AWS.

You can privately access AWS Service Catalog APIs from your Amazon Virtual Private Cloud (Amazon VPC) by creating VPC endpoints. With VPC endpoints, the routing between the VPC and AWS Service Catalog is handled by the AWS network without the need for an internet gateway, NAT gateway, or VPN connection.

The latest generation of VPC endpoints used by AWS Service Catalog is powered by AWS PrivateLink, an AWS technology enabling the private connectivity between AWS services using Elastic Network Interfaces with private IPs in your VPCs.

Identity and Access Management in AWS Service Catalog

Access to AWS Service Catalog requires credentials. Those credentials must have permission to access AWS resources, such as an AWS Service Catalog portfolio or product. AWS Service Catalog integrates with AWS Identity and Access Management (IAM) to enable you to grant AWS Service Catalog administrators the permissions they need to create and manage products, and to grant AWS Service Catalog end users the permissions they need to launch products and manage provisioned products. These policies are either created and managed by AWS or individually by administrators and end users. To control access, you attach these policies to the IAM users, groups, and roles that you use with AWS Service Catalog.

Topics

- Audience (p. 21)

- Predefined AWS Managed Policies (p. 21)
- Identity-based policy examples for AWS Service Catalog (p. 22)
- Troubleshooting AWS Service Catalog identity and access (p. 26)
- Controlling Access (p. 28)
- Using Service-Linked Roles for AWS Service Catalog AppRegistry (p. 28)

Audience

The permissions you have with AWS Identity and Access Management (IAM) may depend on the role you play in AWS Service Catalog.

Administrator - As an AWS Service Catalog administrator, you need full access to the administrator console and IAM permissions that allow you to perform tasks such as creating and managing portfolios and products, managing constraints, and granting access to end users.

End user - Before your end users can use your products, you need to grant them permissions that give them access to the AWS Service Catalog end user console. They can also have permissions to launch products and manage provisioned products.

IAM administrator - If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS Service Catalog. To view example AWS Service Catalog identity-based policies that you can use in IAM, see the section called “Predefined AWS Managed Policies” (p. 21).

Predefined AWS Managed Policies

The managed policies created by AWS grant the required permissions for common use cases. You can attach these policies to your IAM users and roles. For more information, see AWS Managed Policies in the *IAM User Guide*.

The following are the AWS managed policies for AWS Service Catalog.

Administrators

- **AWSServiceCatalogAdminFullAccess** — Grants full access to the administrator console view and permission to create and manage products and portfolios.
- **AWSServiceCatalogAdminReadOnlyAccess** — Grants full access to the administrator console view. Does not grant access to create or manage products and portfolios.

End users

- **AWSServiceCatalogEndUserFullAccess** — Grants full access to the end user console view. Grants permission to launch products and manage provisioned products.
- **AWSServiceCatalogEndUserReadOnlyAccess** — Grants read-only access to the end user console view. Does not grant permission to launch products or manage provisioned products.

To attach a policy to an IAM user

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users**.
3. Choose the name (not the check box) of the IAM user.
4. On the **Permissions** tab, choose **Add permissions**.
5. On the **Add permissions** page, choose **Attach existing policies directly**.

6. Select the check box next to the managed policy for AWS Service Catalog, and then choose **Next: Review**.
7. On the **Permissions summary** page, choose **Add permissions**.
8. (Optional) You must grant administrators additional permissions for Amazon S3 if they need to use a private CloudFormation template. For more information, see User Policy Examples in the *Amazon Simple Storage Service Developer Guide.*

Deprecated policies

The following managed policies are deprecated:

- **ServiceCatalogAdminFullAccess** — Use **AWSServiceCatalogAdminFullAccess** instead.
- **ServiceCatalogAdminReadOnlyAccess** — Use **AWSServiceCatalogAdminReadOnlyAccess** instead.
- **ServiceCatalogEndUserFullAccess** — Use **AWSServiceCatalogEndUserFullAccess** instead.
- **ServiceCatalogEndUserAccess** — Use **AWSServiceCatalogEndUserReadOnlyAccess** instead.

Use the following procedure to ensure that your administrators and end users are granted permissions using the current policies.

To migrate from the deprecated policies to the current policies

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**.
3. In the search field, type **`ServiceCatalog`** to filter the policy list. Choose the name (not the check box) for **ServiceCatalogAdminFullAccess**.
4. For each attached entity (user, group, or role), do the following:
   a. Open the summary page for the entity.
   b. Add one of the current policies described in the procedure linkend="attach-managed-policy"

      Add one of the current policies described in th procedure:To attach a policy to an IAM user (p. 21).
   c. On the **Permissions** tab, next to **ServiceCatalogAdminFullAccess**, choose **Detach Policy**. When prompted for confirmation, choose **Detach**.
5. Repeat the process for **ServiceCatalogEndUserFullAccess**.

Identity-based policy examples for AWS Service Catalog

Topics

- Console access for end users (p. 22)
- Product access for end users (p. 23)
- Example policies for managing provisioned products (p. 23)

Console access for end users

The **AWSServiceCatalogEndUserFullAccess** and **AWSServiceCatalogEndUserReadOnlyAccess** policies grant access to the AWS Service Catalog end user console view. When a user who has either of these

policies chooses AWS Service Catalog in the AWS Management Console, the end user console view displays the products they have permission to launch.

Before end users can successfully launch a product from AWS Service Catalog to which you give access, you must provide them additional IAM permissions to allow them to use each of the underlying AWS resources in a product's AWS CloudFormation template. For example, if a product template includes Amazon Relational Database Service (Amazon RDS), you must grant the users Amazon RDS permissions to launch the product.

To learn about how to enable end users to launch products while enforcing least-access permissions to AWS resources, see the section called “Using Constraints” (p. 46).

If you apply the **AWSServiceCatalogEndUserReadOnlyAccess** policy, your users have access to the end user console, but they won't have the permissions that they need to launch products and manage provisioned products. You can grant these permissions directly to an end user using IAM, but if you want to limit the access that end users have to AWS resources, you should attach the policy to a launch role. You then use AWS Service Catalog to apply the launch role to a launch constraint for the product. For more information about applying a launch role, launch role limitations, and a sample launch role, see AWS Service Catalog Launch Constraints (p. 46).

> **Note**
> If you grant users IAM permissions for AWS Service Catalog administrators, the administrator console view displays instead. Don't grant end users these permissions unless you want them to have access to the administrator console view.

Product access for end users

Before end users can use a product to which you give access, you must provide them additional IAM permissions to allow them to use each of the underlying AWS resources in a product's AWS CloudFormation template. For example, if a product template includes Amazon Relational Database Service (Amazon RDS), you must grant the users Amazon RDS permissions to launch the product.

If you apply the **ServiceCatalogEndUserAccess** policy, your users have access to the end user console view, but they won't have the permissions that they need to launch products and manage provisioned products. You can grant these permissions directly to an end user in IAM, but if you want to limit the access that end users have to AWS resources, you should attach the policy to a launch role. You then use AWS Service Catalog to apply the launch role to a launch constraint for the product. For more information about applying a launch role, launch role limitations, and a sample launch role, see AWS Service Catalog Launch Constraints (p. 46).

Example policies for managing provisioned products

You can create custom policies to help meet the security requirements of your organization. The following examples describe how to customize the access level for each action with support for user, role, and account levels. You can grant users access to view, update, terminate, and manage provisioned products created only by that user or created by others also under their role or the account to which they are logged in. This access is hierarchical — granting account level access also grants role level access and user level access, while adding role level access also grants user level access but not account level access. You can specify these in the policy JSON using a `Condition` block as `accountLevel`, `roleLevel`, or `userLevel`.

These examples also apply to access levels for AWS Service Catalog API write operations: `UpdateProvisionedProduct` and `TerminateProvisionedProduct`, and read operations: `DescribeRecord`, `ScanProvisionedProducts`, and `ListRecordHistory`. The `ScanProvisionedProducts` and `ListRecordHistory` API operations use `AccessLevelFilterKey` as input, and that key's values correspond to the `Condition` block levels discussed here (`accountLevel` is equivalent to an `AccessLevelFilterKey` value of "Account", `roleLevel` to "Role", and `userLevel` to "User"). For more information, see the AWS Service Catalog Developer Guide.

Examples

- Example: Full admin access to provisioned products (p. 24)
- Example: End-user access to provisioned products (p. 24)
- Example: Partial admin access to provisioned products (p. 25)

Example: Full admin access to provisioned products

The following policy allows full read and write access to provisioned products and records within the catalog at the account level.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "servicecatalog:*"
         ],
         "Resource":"*",
         "Condition": {
            "StringEquals": {
               "servicecatalog:accountLevel": "self"
            }
         }
      }
   ]
}
```

This policy is functionally equivalent to the following policy:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "servicecatalog:*"
         ],
         "Resource":"*"
      }
   ]
}
```

In other words, not specifying a `Condition` block in any policy for AWS Service Catalog is treated as the same as specifying `"servicecatalog:accountLevel"` access. Note that `accountLevel` access includes `roleLevel` and `userLevel` access.

Example: End-user access to provisioned products

The following policy restricts access to read and write operations to only the provisioned products or associated records that the current user created.

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

```
            "Action": [
                "servicecatalog:DescribeProduct",
                "servicecatalog:DescribeProductView",
                "servicecatalog:DescribeProvisioningParameters",
                "servicecatalog:DescribeRecord",
                "servicecatalog:ListLaunchPaths",
                "servicecatalog:ListRecordHistory",
                "servicecatalog:ProvisionProduct",
                "servicecatalog:ScanProvisionedProducts",
                "servicecatalog:SearchProducts",
                "servicecatalog:TerminateProvisionedProduct",
                "servicecatalog:UpdateProvisionedProduct"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "servicecatalog:userLevel": "self"
                }
            }
        }
    ]
}
```

Example: Partial admin access to provisioned products

The two policies below, if both applied to the same user, allow what might be called a type of "partial admin access" by providing full read-only access and limited write access. This means the user can see any provisioned product or associated record within the catalog's account but cannot perform any actions on any provisioned products or records that aren't owned by that user.

The first policy allows the user access to write operations on the provisioned products that the current user created, but no provisioned products created by others. The second policy adds full access to read operations on provisioned products created by all (user, role, or account).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "servicecatalog:DescribeProduct",
                "servicecatalog:DescribeProductView",
                "servicecatalog:DescribeProvisioningParameters",
                "servicecatalog:ListLaunchPaths",
                "servicecatalog:ProvisionProduct",
                "servicecatalog:SearchProducts",
                "servicecatalog:TerminateProvisionedProduct",
                "servicecatalog:UpdateProvisionedProduct"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "servicecatalog:userLevel": "self"
                }
            }
        }
    ]
}
```

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

```
        {
            "Effect": "Allow",
            "Action": [
                "servicecatalog:DescribeRecord",
                "servicecatalog:ListRecordHistory",
                "servicecatalog:ScanProvisionedProducts"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "servicecatalog:accountLevel": "self"
                }
            }
        }
    ]
}
```

Troubleshooting AWS Service Catalog identity and access

Use the following information to help you diagnose and fix common issues you might encounter when working with AWS Service Catalog and AWS Identity and Access Management.

Topics

- I am not authorized to perform an action in AWS Service Catalog (p. 26)
- I am not authorized to perform iam:PassRole (p. 26)
- I want to view my access keys (p. 27)
- I'm an administrator and want to allow others to access AWS Service Catalog (p. 27)
- I want to allow people outside of my AWS account to access my AWS Service Catalog resources (p. 27)

I am not authorized to perform an action in AWS Service Catalog

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. The following example error occurs when the mateojackson IAM user tries to use the console to view details about a fictional my-example-widget resource but does not have the fictional awes:GetWidget permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the my-example-widget resource using the awes:GetWidget action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the iam:PassRole action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to AWS Service Catalog.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named marymajor tries to use the console to perform an action in AWS Service Catalog. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the iam:PassRole action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, AKIAIOSFODNN7EXAMPLE) and a secret access key (for example, wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY).

Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

Do not provide your access keys to a third party, even to help find your canonical user ID in the *AWS General Reference guide.* By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing access keys in the *IAM User Guide*.

I'm an administrator and want to allow others to access AWS Service Catalog

To allow others to access AWS Service Catalog, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in AWS Service Catalog.

To get started right away, see Creating your first IAM delegated user and group in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my AWS Service Catalog resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS Service Catalog supports these features, see Identity and Access Management in AWS Service Catalog in the *AWS Service Catalog Administrator Guide*.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.

- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Controlling Access

An AWS Service Catalog portfolio gives your administrators a level of access control for your groups of end users. When you add users to a portfolio, they can browse and launch any of the products in the portfolio. For more information, see the section called “Managing Portfolios” (p. 33).

Constraints

Constraints control which rules are applied to your end users when launching a product from a specific portfolio. You use them to apply limits to products for governance or cost control. For more information about constraints, see the section called “Using Constraints” (p. 46).

AWS Service Catalog launch constraints give you more control over permissions needed by an end user. When your administrator creates a launch constraint for a product in a portfolio, the launch constraint associates a role ARN that is used when your end users launch the product from that portfolio. Using this pattern, you can control access to AWS resource creation. For more information, see the section called “Launch Constraints” (p. 46).

Using Service-Linked Roles for AWS Service Catalog AppRegistry

This section describes how AWS Service Catalog AppRegistry uses the service-linked role, **AWSServiceCatalogAppRegistryServiceRolePolicy**, to create, update, and delete AWS Resource Groups in your accounts. AWS Resource Groups provide customers with visibility and operation of all resources in an application across CloudFormation stacks.

AppRegistry uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that links directly to AppRegistry. AppRegistry predefines service-linked roles and includes all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up AppRegistry easier because you don’t have to manually add the necessary permissions. AppRegistry defines the permissions of its service-linked roles, and unless defined otherwise, only AppRegistry can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This action protects your AppRegistry resources because you can't inadvertently remove permission to access the resources.

Service-Linked Role Permissions for AWS Service Catalog AppRegistry

AppRegistry uses the service-linked role named **AWSServiceRoleForAWSServiceCatalogAppRegistry** to enable AppRegistry to call AWS APIs on your behalf.

The **AWSServiceRoleForServiceCatalogAppRegistry** service-linked role trusts the servicecatalog-appregistry.amazonaws.com service principal to assume the role.

The role permissions policy allows AppRegistry to complete the following actions on the specified resources:

```
          {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "cloudformation:DescribeStacks",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "resource-groups:CreateGroup",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/EnableAWSServiceCatalogAppRegistry": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "resource-groups:DeleteGroup",
                "resource-groups:UpdateGroup",
                "resource-groups:GetGroup",
                "resource-groups:GetTags",
                "resource-groups:Tag",
                "resource-groups:Untag"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/EnableAWSServiceCatalogAppRegistry": "true"
                }
            }
        }
    ]
}
```

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

To allow an IAM entity to create the **AWSServiceRoleForServiceCatalogAppRegistry** service-linked role, add this statement to the permissions policy for the IAM entity that needs to create the service-linked role.

```
          {
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/servicecatalog-
appregistry.amazonaws.com/AWSServiceRoleForServiceCatalogAppRegistry*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "servicecatalog-
appregistry.amazonaws.com"}}
}
```

Creating a service-linked role for AWS Service Catalog AppRegistry

When the customer requests a specific operation, our service automatically creates the role. When you create an application or update and existing application in the AWS Management Console, the AWS CLI, or the AWS API, AppRegistry creates the service-linked role for you.

> **Important**
> This service-linked role can appear in your account if you completed an action in another service that uses the features supported by this role.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create and application or update an existing application, AppRegistry creates the service-linked role for you again. For more information, see the section called "Deleting a Service-Linked Role for AWS Service Catalog AppRegistry" (p. 30).

You can also use the IAM console to create a service-linked role with the **AWSServiceRoleForAWSServiceCatalogAppRegistry** use case. In the AWS CLI or the AWS API, create a service-linked role with the servicecatalog-appregistry.amazonaws.com service name. For more information, see Creating a service-linked role in the *IAM User Guide.* If you delete this service-linked role, you can use this same process to create the role again.

Editing a Service-Linked Role for AWS Service Catalog AppRegistry

After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. But you can use the IAM console, CLI, or API to edit the service-linked role's description.

For more information, see Editing a service-linked role in the *IAM User Guide.*

Deleting a Service-Linked Role for AWS Service Catalog AppRegistry

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean the resources for your service-linked role before you can manually delete it.

You can use AppRegistry to clean resources and then use the IAM console, CLI, or API to perform the deletion.

To clean your service-linked role's resources before manual deletion, you must first disassociate all resources from applications. Next you must disassociate all attribute groups form applications. Then you can delete the applications.

Supported Regions for AWS Service Catalog AppRegistry Service-Linked Roles

AppRegistry supports using service-linked roles in all of the regions where the service is available. For more information, see AWS Regions and Endpoints in the *AWS General Reference guide.*

Logging and Monitoring in AWS Service Catalog

AWS Service Catalog integrates with AWS CloudTrail, a service that captures all of the AWS Service Catalog API calls and delivers the log files to an Amazon S3 bucket that you specify. For more information, see Logging Amazon Service Catalog API Calls with CloudTrail.

You can also use notification constraints to set up Amazon SNS notifications about stack events. For more information, see the section called "Notification Constraints" (p. 49).

Compliance Validation for AWS Service Catalog

Third-party auditors assess the security and compliance of AWS Service Catalog as part of multiple AWS compliance programs, including the following:

- System and Organization Controls (SOC)
- Payment Card Industry Data Security Standard (PCI DSS)
- Federal Risk and Authorization Management Program (FedRAMP)
- Health Insurance Portability and Accountability Act (HIPAA)

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS Service Catalog depends on the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides these resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in AWS Service Catalog

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, AWS Service Catalog offers AWS Service Catalog self-service actions. With self-service actions, customers can reduce administrative maintenance and end-user training while adhering to compliance and security measures. With self-service actions, as the administrator, you can enable end users to perform operational tasks such as backup and restore, troubleshoot issues, run approved commands, and request permissions in AWS Service Catalog. To learn more, see the section called “Using Service Actions” (p. 53).

Infrastructure Security in AWS Service Catalog

As a managed service, AWS Service Catalog is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access AWS Service Catalog through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

With AWS Service Catalog, you can control the Regions in which data is stored. Portfolios and products are only available in the Regions in which you have made them available. You can use the `CopyProduct` API to copy a product to another Region.

Security Best Practices for AWS Service Catalog

AWS Service Catalog provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don’t represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

You can define rules that limit the parameter values that a user enters when launching a product. These rules are called template constraints because they constrain how the AWS CloudFormation template for the product is deployed. You use a simple editor to create template constraints, and you apply them to individual products.

AWS Service Catalog applies constraints when provisioning a new product or updating a product that is already in use. It always applies the most restrictive constraint among all constraints applied to the portfolio and the product. For example, consider a scenario where the product allows all Amazon EC2 instances to be launched and the portfolio has two constraints: one that allows all non-GPU type EC2 instances to be launched and one that allows only t1.micro and m1.small EC2 instances to be launched. For this example, AWS Service Catalog applies the second, more restrictive constraint (t1.micro and m1.small).

Limit access end users have to AWS resources by attaching an IAM policy to a launch role. You then use AWS Service Catalog to create a launch constraint to use the role when launching the product.

Managing Catalogs

AWS Service Catalog provides an interface for managing portfolios, products, and constraints from an administrator console.

> **Note**
> To perform any of the tasks in this section, you must have administrator permissions for AWS Service Catalog. For more information, see Identity and Access Management in AWS Service Catalog (p. 20).

Tasks

- Managing Portfolios (p. 33)
- Managing Products (p. 42)
- Using AWS Service Catalog Constraints (p. 46)
- AWS Service Catalog Service Actions (p. 53)
- Adding AWS Marketplace Products to Your Portfolio (p. 57)
- Using AWS CloudFormation StackSets (p. 62)
- Managing Budgets (p. 62)

Managing Portfolios

You create, view, and update portfolios on the **Portfolios** page in the AWS Service Catalog administrator console.

Tasks

- Creating, Viewing, and Deleting Portfolios (p. 33)
- Viewing Portfolio Details (p. 34)
- Creating and Deleting Portfolios (p. 34)
- Adding Products (p. 34)
- Adding Constraints (p. 36)
- Granting Access to Users (p. 37)
- Sharing a Portfolio (p. 37)
- Sharing and Importing Portfolios (p. 40)

Creating, Viewing, and Deleting Portfolios

The **Portfolios** page displays a list of the portfolios that you have created in the current region. Use this page to create new portfolios, view a portfolio's details, or delete portfolios from your account.

To view the Portfolios page

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Select a different region as necessary.
3. If you are new to AWS Service Catalog, you see the AWS Service Catalog start page. Choose **Get started** to create a portfolio. Follow the instructions to create your first portfolio, and then proceed to the **Portfolios** page.

While using AWS Service Catalog, you can return to the **Portfolios** page at any time; choose **Service Catalog** in the navigation bar and then choose **Portfolios**.

Viewing Portfolio Details

In the AWS Service Catalog administrator console, the **Portfolio details** page lists the settings for a portfolio. Use this page to manage the products in the portfolio, grant users access to products, and apply TagOptions and constraints.

To view the Portfolio details page

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose the portfolio that you want to manage.

Creating and Deleting Portfolios

Use the **Portfolios** page to create and delete portfolios. Deleting a portfolio removes it from your account. Before you can delete a portfolio, you must remove all the products, constraints, and users that it contains.

To create a new portfolio

1. Navigate to the **Portfolios** page.
2. Choose **Create portfolio**.
3. On the **Create portfolio** page, enter the requested information.
4. Choose **Create**. AWS Service Catalog creates the portfolio and displays the portfolio details.

To delete a portfolio

1. Navigate to the **Portfolios** page.
2. To select the portfolio, choose the corresponding radio button or the name of the portfolio.
3. Choose **Actions**.
4. Choose **Delete**.

   A confirmation message appears.
5. Choose **Delete** to confirm.

Adding Products

To add products to a portfolio, you either create a new product or add an existing product from your catalog to the portfolio.

> **Note**
> The AWS CloudFormation template that you upload when you create an AWS Service Catalog product is stored in an Amazon Simple Storage Service (Amazon S3) bucket that starts with `cf-templates-` in your AWS account. Do not delete these files unless you are sure that they are no longer in use.

Adding a New Product

You add new products directly from the portfolio details page. When you create a product from this page, AWS Service Catalog adds it to the currently selected portfolio. You can also add a product to other portfolios.

To add a new product

1. Navigate to the **Portfolios** page, and then choose the name of the portfolio to which you want to add the product.
2. On the portfolio details page, expand the **Products** section, and then choose **Upload new product**.
3. For **Enter product details**, enter the following:

   - **Product name** – The name of the product.
   - **Short description** – The short description. This description appears in search results to help the user choose the correct product.
   - **Description** – The full description. This description is shown in the product listing to help the user choose the correct product.
   - **Provided by** – The name or email address of your IT department or administrator.
   - **Vendor** (optional) – The name of the application's publisher. This field allows users to sort their products list to makes it easier to find the products that they need.

   Choose **Next**.
4. For **Enter support details**, enter the following:

   - **Email contact** (optional) – The email address for reporting issues with the product.
   - **Support link** (optional) – A URL to a site where users can find support information or file tickets. The URL must begin with `http://` or `https://`.
   - **Support description** (optional) – A description of how users should use the **Email contact** and **Support link**.

   Choose **Next**.
5. On the **Version details** page, enter the following:

   - **Select template** – An AWS CloudFormation template from a local drive or a URL that points to a template stored in Amazon S3. If you specify an Amazon S3 URL, it must begin with `https://`. The extension for the template file must be `.template`.
   - **Version title** – the name of the product version (e.g., "v1", "v2beta"). No spaces are allowed.
   - **Description** (optional) – A description of the product version including how this version differs from the previous version.

   Choose **Next**.
6. On the **Review** page, verify that the information is correct, and then choose **Confirm and upload**. After a few seconds, the product appears in your portfolio. You might need to refresh your browser to see the product.

Adding an Existing Product

You can add existing products to a portfolio from three places: the **Portfolios** list, the portfolio details page, or the **Products** page.

To add an existing product to a portfolio

1. Navigate to the **Portfolios** page.
2. Choose a portfolio, and then choose **Add product**.
3. Choose a product, and then choose **Add product to portfolio**.

Removing a Product from a Portfolio

When you no longer want users to use a product, remove it from a portfolio. The product is still available in your catalog from the **Products** page, and you can still add it to other portfolios. You can remove multiple products from a portfolio at one time.

To remove a product from a portfolio

1. Navigate to the **Portfolios** page, and then choose the portfolio that contains the product. The portfolio details page opens.
2. Expand the **Products** section.
3. Choose one or more products, and then choose **Remove product**.
4. Choose **Continue**.

Adding Constraints

To control how users are able to use products, add constraints. For more information about the types of constraints that AWS Service Catalog supports, see Using AWS Service Catalog Constraints (p. 46).

You add constraints to products after they have been placed in a portfolio.

To add a constraint to a product

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose **Portfolios** and select a portfolio.
3. In the portfolio details page, expand the **Constraints** section and choose **Add constraints**.
4. For **Product**, select the product to which to apply the constraint.
5. For **Constraint type**, choose one of the following options:

   Launch – The IAM role that AWS Service Catalog uses to launch and manage the product. For more information, see AWS Service Catalog Launch Constraints (p. 46).

   Notification – The Amazon SNS topic specified to receive notifications. For more information, see AWS Service Catalog Notification Constraints (p. 49).

   Template – A JSON–formatted text file that contains one or more rules. Rules are added to the AWS CloudFormation template used by the product. For more information, see Template Constraint Rules (p. 51).

   Stack Set – Uses AWS CloudFormation StackSets to specify multiple accounts and regions for the AWS Service Catalog product launch. For more information, see AWS Service Catalog Stack Set Constraints (p. 50).

   Tag Update – Allows you to update tags after the product has been provisioned. For more information, see Amazon Service Catalog Tag Update Constraints.
6. Choose **Continue**.

To edit a constraint

1. Sign in to the AWS Management Console and open the AWS Service Catalog administrator console at https://console.aws.amazon.com/catalog/.
2. Choose **Portfolios** and select a portfolio.
3. In the portfolio details page, expand the **Constraints** section and select the constraint to edit.
4. Choose **Edit constraints**.

5. Edit the constraint as needed, and choose **Submit**.

Granting Access to Users

Give users access to portfolios by using IAM users, groups, and roles. The best way to provide portfolio access for many users is to put the users in an IAM group and grant access to that group. That way you can simply add and remove users from the group to manage portfolio access. For more information, see IAM users and groups in the *IAM User Guide*.

In addition to access to a portfolio, IAM users must also have access to the AWS Service Catalog end user console. You grant access to the console by applying permissions in IAM. For more information, see Identity and Access Management in AWS Service Catalog (p. 20).

To grant portfolio access to users or groups

1. In the portfolio details page, expand **Users, groups and roles**, and then choose **Add user, group or role**.
2. Choose the **Groups**, **Users**, or **Roles** tab to add groups, users, or roles, respectively.
3. Choose one or more users, groups, or roles, and then choose **Add Access** to grant them access to the current portfolio.

   Tip
   To grant access to a combination of groups, users, and roles, you can switch between the tabs without losing your selection.

To remove access to a portfolio

1. On the portfolio details page, choose the checkbox for the user or group.
2. Choose **Remove user, group or role**.

Sharing a Portfolio

To enable an AWS Service Catalog administrator for another AWS account to distribute your products to end users, share your AWS Service Catalog portfolio with them using either account-to-account sharing or AWS Organizations.

When you share a portfolio using account-to-account sharing or Organizations, you are sharing a *reference* of that portfolio. The products and constraints in the imported portfolio stay in sync with changes that you make to the *shared portfolio*, the original portfolio that you shared.

The recipient cannot change the products or constraints, but can add AWS Identity and Access Management (IAM) access for end users.

Note
You can't share a shared resource. This includes portfolios that contain a shared product.

Account-to-account sharing

To complete these steps, you must obtain the account ID of the target AWS account. You can find the ID on the **My Account** page in the AWS Management Console of the target account.

To share a portfolio with an AWS account

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. In the left navigation menu, choose **Portfolios**, select the portfolio you want to share, then choose **Actions**, and **Share**.

3. Select the portfolio you want to share. Then choose **Actions**, then **Share**.
4. In **Enter AWS account ID**, enter the account ID of the AWS account that you are sharing with. Then, choose **Share**.
5. Send the URL to the AWS Service Catalog administrator of the target account. The URL opens the **Import Portfolio** page with the ARN of the shared portfolio automatically provided.

Importing a Portfolio

If an AWS Service Catalog administrator for another AWS account shares a portfolio with you, import that portfolio into your account so that you can distribute its products to your end users.

To import the portfolio, you must get a URL to import the portfolio from the administrator.

Open the URL. In **Import Portfolio** , choose **Import**. The **Portfolios** page appears, and the portfolio displays in the **Imported Portfolios** table.

You don't need to import a portfolio if the portfolio was shared through AWS Organizations.

Sharing with AWS Organizations

You can share AWS Service Catalog portfolios using AWS Organizations.

First, you must decide if you're sharing from the management account or from a delegated administrator account. If you don't want to share from your management account, register a delegated admin account and use it for sharing. Next, you must decide who to share to. You can share to the following entities:

- An organization account.
- An organizational unit (OU).
- The organization itself. (This shares with every account in the organization.)

Sharing from a management account

To share a portfolio with an organization

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. On the **Portfolios** page, select the portfolio that you want to share, then choose **Actions**, and **Share**.
3. In the **Enter AWS account ID** window, choose **Organization**.
4. Select the **Node Type** and enter the **Input Value** for the organization you wish to share with.
5. Choose **Share**.

Sharing from a delegated administrator account

The management account of an organization can register and de-register other accounts as delegated administrators for the organization.

A delegated administrator can share AWS Service Catalog resources in their organization the same way a management account can. They are authorized to create, delete, and share portfolios, and more.

To register or de-register a delegated administrator, you must use the API or CLI from the management account. For more information, see RegisterDelegatedAdministrator and DeregisterDelegatedAdministrator in the *AWS Organizations API Reference*.

> **Note**
> Before you can designate a delegated , you administrator must call EnableAWSOrganizationsAccess.

The procedure for sharing a portfolio from a delegated administrator account is the same as sharing from a management account, as seen above in the section called “Sharing from a management account” (p. 38).

If a member is de-registered as a delegated administrator, the following occurs:

- Portfolio shares that were created from that account are removed.
- They can no longer create new portfolio shares.

> **Note**
> If the portfolio and shares created by a delegated administrator do not get removed after the delegated administrator is de-registered, register and de-register the delegated administrator again. This action removes the portfolio and shares created by that account.

Moving accounts within your organization

If you move an account within your AWS Organization, the Service Catalog portfolios shared with the account might change.

Accounts only have access to portfolios shared with their destination organization or organizational unit.

If you create a new child organizational unit after sharing a portfolio, you should unshare and create a new share with that portfolio for accounts in the new OU to maintain access.

Sharing TagOptions when sharing portfolios

As an administrator, you can create a share to include TagOptions. TagOptions are key-value pairs that enables administrators to:

- Define and enforce the taxonomy for tags.
- Define tag options and associate them to products and portfolios.
- Share tag options associated with portfolios and products with other accounts.

When you add or remove tag options in the main account, the change automatically appears in recipient accounts. In recipient accounts, when an end user provisions a product with TagOptions, they must choose values for tags that become tags on the provisioned product.

In recipient accounts administrators can associate additional local TagOptions to their imported portfolio to enforce tagging rules that are specific to that account.

> **Note**
> To share a portfolio, you need the the consumer's AWS Account ID. Find the AWS Account ID in **My Account** on the AWS console.

> **Note**
> If a TagOption has a single value, Service Catalog automatically enforces that value during the provisioning process.

To share TagOptions when sharing portfolios

1. In the left navigation menu, choose **Portfolios**.
2. In **Local portfolios**, choose and open a portfolio.
3. Choose **Share** from the list above and then choose the **Share** button.
4. "SEA;IAD">Choose to share with another AWS account or AWS organization.
5. Enter the 12 digit account ID number, select **Enable**, and then choose **Share**.

The account you shared displays in the **Accounts shared with** section. It indicates whether TagOptions were enabled.

You can also update a portfolio share to include TagOptions. All TagOptions that belong to the portfolio and product now share to this account.

To update a portfolio share to include TagOptions

1. In the left navigation menu, choose **Portfolios**.
2. In **Local portfolio**, choose and open a portfolio.
3. Choose **Share** from the list above.
4. In **Accounts shared with**, choose an account ID and then choose **Actions**.
5. Select **Update unshare** or **Unshare**.

   When you select **Update unshare**, choose **Enable** to initiate sharing TagOptions. The account you shared displays in the **Accounts shared with** section.

   When you select **Unshare**, confirm you no longer want to share the account.

Sharing and Importing Portfolios

Topics

- Relationship Between Shared and Imported Portfolios (p. 41)

To make your AWS Service Catalog products available to users who are not in your AWS account, such as users who belong to other organizations or to other AWS accounts in your organization, you share your portfolios with them. You can share in several ways, including account-to-account sharing, organizational sharing, and deploying catalogs using stack sets.

Before you share your products and portfolios to other accounts, you must decide whether you want to share a reference of the catalog or to deploy a copy of the catalog into each recipient account. Note that if you deploy a copy, you must redeploy if there are updates you want to propagate to the recipient accounts.

You can use stack sets to deploy your catalog to many accounts at the same time. If you want to share a reference (an imported version of your portfolio that stays in sync with the original), you can use account-to-account sharing or you can share using AWS Organizations.

If you want to use stack sets to deploy a copy of your catalog, see How to set up a multi-region, multi-account catalog of company standard AWS Service Catalog products.

When you share a portfolio using account-to-account sharing or AWS Organizations, you allow an AWS Service Catalog administrator of another AWS account to import your portfolio into his or her account and distribute the products to end users in that account.

This *imported portfolio* isn't an independent copy. The products and constraints in the imported portfolio stay in sync with changes that you make to the *shared portfolio*, the original portfolio that you shared. The *recipient administrator*, the administrator with whom you share a portfolio, cannot change the products or constraints, but can add AWS Identity and Access Management (IAM) access for end users. For more information, see Granting Access to Users (p. 37).

The recipient administrator can distribute the products to end users who belong to his or her AWS account in the following ways:

- By adding IAM users, groups, and roles to the imported portfolio.

- By adding products from the imported portfolio to a *local portfolio*, a separate portfolio that the recipient administrator creates and that belongs to his or her AWS account. The recipient administrator then adds IAM users, groups, and roles to the local portfolio. The constraints that you applied to the products in the shared portfolio are also present in the local portfolio. The recipient administrator can add additional constraints to the local portfolio, but cannot remove the imported constraints.

When you add products or constraints to the shared portfolio or remove products or constraints from it, the change propagates to all imported instances of the portfolio. For example, if you remove a product from the shared portfolio, that product is also removed from the imported portfolio. It is also removed from all local portfolios that the imported product was added to. If an end user launched a product before you removed it, the end user's provisioned product continues to run, but the product becomes unavailable for future launches.

If you apply a launch constraint to a product in a shared portfolio, it propagates to all imported instances of the product. To override this launch constraint, the recipient administrator adds the product to a local portfolio and then applies a different launch constraint to it. The launch constraint that is in effect sets a launch role for the product.

A *launch role* is an IAM role that AWS Service Catalog uses to provision AWS resources (such as EC2 instances or RDS databases) when an end user launches the product. As an administrator you can choose to designate a specific launch role ARN or a local role name. If you use the role ARN, the role will be used even if the end user belongs to a different AWS account than the one that owns the launch role. If you use a local role name, the IAM role with that name in the end user's account will be used.

For more information about launch constraints and launch roles, see AWS Service Catalog Launch Constraints (p. 46). The AWS account that owns the launch role provisions the AWS resources, and this account incurs the usage charges for those resources. For more information, see AWS Service Catalog Pricing.

This video shows you how to share portfolios across accounts in AWS Service Catalog.

Share (https://www.youtube.com/embed/BVSohYOppjk%22%3EShare) Portfolios Across Accounts in AWS Service Catalog

Note
You cannot re-share products from a portfolio that has been imported or shared.

Note
Portfolio imports must occur in the same region between the management and dependent accounts.

Relationship Between Shared and Imported Portfolios

This table summarizes the relationship between an imported portfolio and a shared portfolio, and the actions that an administrator who imports a portfolio can and can't take with that portfolio and the products in it.

Element of Shared Portfolio	Relationship to Imported Portfolio	Recipient Administrator Can	Recipient Administrator Cannot
Products and product versions	Inherited. If the portfolio creator adds products to or removes products from the shared portfolio, the change propagates to the imported portfolio.	Add imported products to local portfolios. Products stay in sync with shared portfolio.	Upload or add products to the imported portfolio or remove products from the imported portfolio.

Element of Shared Portfolio	Relationship to Imported Portfolio	Recipient Administrator Can	Recipient Administrator Cannot
Launch constraints	Inherited. If the portfolio creator adds launch constraints to or removes launch constraints from a shared product, the change propagates to all imported instances of the product. If the recipient administrator adds an imported product to a local portfolio, the imported launch constraint that is applied to that product is present in the local portfolio.	In a local portfolio, the administrator can override the imported launch constraint by applying a different one to the product.	Add launch constraints to or remove launch constraints from the imported portfolio.
Template constraints	Inherited. If the portfolio creator adds a template constraint to or removes a template constraints from a shared product, the change propagates to all imported instances of the product. If the recipient administrator adds an imported product to a local portfolio, the imported template constraints that are applied to that product are inherited by the local portfolio.	In a local portfolio, the administrator can add template constraints that take effect in addition to the imported constraints.	Remove the imported template constraints.
IAM users, groups, and roles	Not inherited.	Add IAM users, groups, and roles that are in administrator's AWS account.	Not applicable.

Managing Products

You create products by packaging an AWS CloudFormation template with metadata, update products by creating a new version based on an updated template, and group products together into portfolios to distribute them to users.

New versions of products are propagated to all users who have access to the product through a portfolio. When you distribute an update, end users can update existing provisioned products with just a few clicks.

Tasks

- Viewing the Products Page (p. 43)
- Creating Products (p. 43)
- Adding Products to Portfolios (p. 44)
- Updating Products (p. 44)
- Deleting Products (p. 45)
- Managing Versions (p. 45)

Viewing the Products Page

You manage products from the **Products** page in the AWS Service Catalog administrator console.

To view the Products page

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose **Service Catalog** in the navigation bar.
3. Choose **Products**.

Creating Products

You create products from the **Products** page in the AWS Service Catalog administrator console.

To create a new AWS Service Catalog product

1. Navigate to the **Products** page.
2. Choose **Upload new product**.
3. For **Enter product details**, enter the following:

   - **Product name** – The name of the product.
   - **Short description** – The short description. This description appears in search results to help the user choose the correct product.
   - **Description** – The full description. This description is shown in the product listing to help the user choose the correct product.
   - **Provided by** – The name of your IT department or administrator.
   - **Vendor** (optional) – The name of the application's publisher. This field allows users to sort their products list to makes it easier to find the products that they need.

   Choose **Next**.
4. For **Enter support details**, enter the following:

   - **Email contact** (optional) – The email address for reporting issues with the product.
   - **Support link** (optional) – A URL to a site where users can find support information or file tickets. The URL must begin with `http://` or `https://`.
   - **Support description** (optional) – A description of how users should use the **Email contact** and **Support link**.

   Choose **Next**.

5. For **Version details**, enter the following:

   - **Select template** – An AWS CloudFormation template from a local drive or a URL that points to a template stored in Amazon S3. If you specify an Amazon S3 URL, it must begin with `https://`. The extension for the template file must be `.template`.
   - **Version title** – the name of the product version (e.g., "v1", "v2beta"). No spaces are allowed.
   - **Description** (optional) – A description of the product version including how this version differs from the previous version.
   - **Guidance** – By default, product versions don't have any guidance, so end users can use that version to update and launch provisioned products. If you set the guidance to deprecated, users can make updates to a provisioned product but can't launch new provisioned products of that version.
6. Choose **Next**.
7. On the **Review** page, verify that the information is correct, and then choose **Confirm and upload**. After a few seconds, the product appears on the **Products** page. You might need to refresh your browser to see the product.

You can also use CodePipeline to create and configure a pipeline to deploy your product template to AWS Service Catalog and deliver changes you have made in your source repository. For more information, see Tutorial: Create a Pipeline That Deploys to AWS Service Catalog.

You can define parameter properties in your AWS CloudFormation template and enforce those rules during provisioning. These properties have the ability to define the minimum and maximum length, minimum and maximum values, allowed values, and a regular expression for the value. AWS Service Catalog warns users during provisioning if the value they provide does not adhere to the parameter property. To learn more about parameter properties, see Parameters in the *AWS CloudFormation User Guide.*

Adding Products to Portfolios

You can add products in any number of portfolios. When a product is updated, all of the portfolios that contain the product automatically receive the new version, including shared portfolios.

To add a product from your catalog to a portfolio

1. Navigate to the **Products** page.
2. Choose a product, choose **Actions**, and then choose **Add product to portfolio**.
3. Choose a portfolio, and then choose **Add product to portfolio**.

Updating Products

When you need to update a product's AWS CloudFormation template, you create a new version of your product. A new product version is automatically available to all users who have access to a portfolio that contains the product.

Users who are currently running a provisioned product of the previous version of the product can update their provisioned product using the end user console view. When a new version of a product is available, users can use the **Update provisioned product** command on either the **Provisioned product list** or **Provisioned product details** pages.

> **Note**
> Before you create a new version of a product, test your product updates in AWS CloudFormation to ensure that they work.

To create a new product version

1. Navigate to the **Products** page.
2. Choose the product name.
3. On the product details page, expand the **Versions** section, and then choose **Create new version**.
4. For **Version details**, enter the following:

   - **Select template** – An AWS CloudFormation template from a local drive or a URL that points to a template stored in Amazon S3. If you specify an Amazon S3 URL, it must begin with `https://`. The extension for the template file must be `.template` and can be either JSON– or YAML-formatted text files. For more information, see Template Formats in the *AWS CloudFormation User Guide*.
   - **Version title** – the name of the product version (e.g., "v1", "v2beta"). No spaces are allowed.
   - **Description** (optional) – A description of the product version including how this version differs from the previous version.
   - **Guidance** – By default, product versions don't have any guidance, so end users can use that version to update and launch provisioned products. If you set the guidance to deprecated, users can make updates to a provisioned product but can't launch new provisioned products of that version.

   Choose **Save**.

You can also use CodePipeline to create and configure a pipeline to deploy your product template to AWS Service Catalog and deliver your changes in your source repository. For more information, see Tutorial: Create a Pipeline That Deploys to AWS Service Catalog.

Deleting Products

To remove products from your account completely, delete them from your catalog. Deleting a product removes all versions of the product from every portfolio that contains the product. Deleted products cannot be recovered.

If your product has a budget associated to it, you will need to disassociate the budget before you can delete the product. For more information on disassociating a budget, see the section called “Managing Budgets” (p. 62).

To delete a product from your catalog

1. Navigate to the **Products** page.
2. Choose the product, choose **Actions**, and then choose **Delete product**.
3. Verify that you have chosen the product that you want to delete, and then choose **Continue**.

Managing Versions

You assign product versions when you create a product, and you can update product versions any time.

Versions have an AWS CloudFormation template, a title, a description, a status, and guidance.

Version Status

A version can have one of three statuses:

- **Active** - An active version appears in the version list and allows users to launch it.

- **Inactive** - An inactive version is hidden from the version list. Existing provisioned products launched from this version will not be affected.
- **Deleted** - If a version is deleted, it is removed from the version list. Deleting a version can't be undone.

Version Guidance

You can set version guidance to provide information to end users about the product version. Version guidance only affects active product versions.

There are two options for version guidance:

- **None** - By default, product versions don't have any guidance, so end users can use that version to update and launch provisioned products.
- **Deprecated** - With a deprecated version, users can make updates to a provisioned product but can't launch new provisioned products using the deprecated version.

Updating Versions

You assign product versions when creating a product, and you can also update a version any time. For more information about creating a product, see Creating Products (p. 43).

To update a product version

1. In the AWS Service Catalog console, choose **Products**.
2. From the product list, choose the product you want to update the version of.
3. On the **Product details** page, choose the **Versions** tab, then choose the version you want to update.
4. On the **Version details** page, edit the product version, then choose **Save changes**.

Using AWS Service Catalog Constraints

You apply constraints to control the rules that are applied to a product in a specific portfolio when the end users launches it. When the end users launches the product, they will see the rules you have applied using constraints. You can apply constraints to a product once it is put into a portfolio. Constraints are active as soon as you create them, and they're applied to all current versions of a product that have not been launched.

Constraints

- AWS Service Catalog Launch Constraints (p. 46)
- AWS Service Catalog Notification Constraints (p. 49)
- AWS Service Catalog Tag Update Constraints (p. 49)
- AWS Service Catalog Stack Set Constraints (p. 50)
- AWS Service Catalog Template Constraints (p. 50)

AWS Service Catalog Launch Constraints

A launch constraint specifies the AWS Identity and Access Management (IAM) role that AWS Service Catalog assumes when an end user launches a product. An IAM role is a collection of permissions that an IAM user or AWS service can assume temporarily to use AWS services. For an introductory example, see Step 6: Add a Launch Constraint to Assign an IAM Role (p. 14).

Launch constraints apply to products in the portfolio (product-portfolio association). Launch constraints do not apply at the portfolio level or to a product across all portfolios. To associate a launch constraint with all products in a portfolio, you must apply the launch constraint to each product individually.

Without a launch constraint, end users must launch and manage products using their own IAM credentials. To do so, they must have permissions for AWS CloudFormation, AWS services that the products use, and AWS Service Catalog. By using a launch role, you can instead limit the end users' permissions to the minimum they require for that product. For more information about end user permissions, see Identity and Access Management in AWS Service Catalog (p. 20).

To create and assign IAM roles, you must have the following IAM administrative permissions:

- `iam:CreateRole`
- `iam:PutRolePolicy`
- `iam:PassRole`
- `iam:Get*`
- `iam:List*`

Configuring a Launch Role

The IAM role that you assign to a product as a launch constraint must have permissions to use the following:

- Amazon CloudFormation
- Services in the Amazon CloudFormation template for the product
- Read access to the Amazon CloudFormation template in Amazon S3

The IAM role also must have a trust relationship with AWS Service Catalog, which you assign by selecting **AWS Service Catalog** as the role type in the following procedure. The trust relationship allows AWS Service Catalog to assume the role during the launch process to create resources.

> **Note**
> The `servicecatalog:ProvisionProduct`, `servicecatalog:TerminateProduct`, and `servicecatalog:UpdateProduct` permissions cannot be assigned in a launch role. You must use IAM roles, as shown in the inline policy steps in the section Grant Permissions to AWS Service Catalog End Users (p. 8).

To create a launch role

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Roles**.
3. Choose **Create New Role**.
4. Enter a role name and choose **Next Step**.
5. Under **AWS Service Roles** next to **AWS Service Catalog**, choose **Select**.
6. On the **Attach Policy** page, Choose **Next Step**.
7. To create the role, choose **Create Role**.

To attach a policy to the new role

1. Choose the role that you created to view the role details page.
2. Choose the **Permissions** tab, and expand the **Inline Policies** section. Then, choose **click here**.

3. Choose **Custom Policy**, and then choose **Select**.
4. Enter a name for the policy, and then paste the following into the **Policy Document** editor:

```
  "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "s3:ExistingObjectTag/servicecatalog:provisioning":"true"
            }
         }
      }
   ]
}
```

5. Add a line to the policy for each additional service the product uses. For example, to add permission for Amazon Relational Database Service (Amazon RDS), enter a comma at the end of the last line in the `Action` list, and then add the following line:

```
"rds:*"
```

6. Choose **Apply Policy**.

Applying a Launch Constraint

After you configure the launch role, assign the role to the product as a launch constraint. This action tells AWS Service Catalog to assume the role when an end user launches the product.

To assign the role to a product

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose the portfolio that contains the product.
3. Choose the **Constraints** tab and choose **Create constraint**.
4. Choose the product from **Product** and choose **Launch** under **Constraint type**. Choose **Continue**.
5. In the **Launch constraint** section, you can select an IAM role from your account and enter an IAM role ARN, or enter the role name.

   If you specify the role name and if an account uses the launch constraint, the account uses that name for the IAM role. This approach allows launch-role constraints to be account-agnostic so you can create fewer resources per shared account.

   Note
   The given role name must exist in the account that created the launch constraint and the account of the user who launches a product with this launch constraint.
6. After specifying the IAM role, choose **Create**.

Verifying the Launch Constraint

To verify AWS Service Catalog uses the role to launch the product and successfully provisions the product, launch the product from the AWS Service Catalog console. To test a constraint prior to releasing it to users, create a test portfolio that contains the same products and test the constraints with that portfolio.

To launch the product

1. In the menu for the AWS Service Catalog console, choose **Service Catalog**, **End user**.
2. Choose the product to open the **Product details** page. In the **Launch options** table, verify that the Amazon Resource Name (ARN) of the role appears.
3. Choose **Launch product**.
4. Proceed through the launch steps, filling in any required information.
5. Verify that the product starts successfully.

AWS Service Catalog Notification Constraints

A notification constraint specifies an Amazon SNS topic to receive notifications about stack events. The SNS topic specifies the email address to receive the notifications.

Use the following procedure to create an SNS topic and subscribe to it.

To create an SNS topic and a subscription

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. Choose **Create topic**.
3. Type a topic name and then choose **Create topic**.
4. Choose **Create subscription**.
5. For **Protocol**, select **Email**. For **Endpoint**, type an email address that you can use to receive notifications. Choose **Create subscription**.
6. You'll receive a confirmation email with the subject line `AWS Notification - Subscription Confirmation`. Open the email and follow the directions to complete your subscription.

Use the following procedure to apply a notification constraint using the SNS topic that you created using the previous procedure.

To apply a notification constraint to a product

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose the portfolio that contains the product.
3. Expand **Constraints** and choose **Add constraints**.
4. Choose the product from **Product** and set **Constraint type** to **Notification**. Choose **Continue**.
5. Choose **Choose a topic from your account** and select the SNS topic that you created from **Topic Name**.
6. Choose **Submit**.

AWS Service Catalog Tag Update Constraints

With tag update constraints, AWS Service Catalog administrators can allow or disallow end users to update tags on resources associated with an AWS Service Catalog provisioned product. If tag updating is allowed, then new tags associated with the AWS Service Catalog product or portfolio will be applied to provisioned resources during a provisioned product update.

To enable tag updates to a product

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.

2. Choose the portfolio that contains the product you want to update.
3. Choose the **Constraints** tab and choose **Add constraints**.
4. Under **Constraint type**, choose **Tag Update**.
5. Choose the product from **Product**, then choose **Continue**.
6. On the **Tag Updates page**, select **Enable Tag Updates**.
7. Choose **Submit**.

AWS Service Catalog Stack Set Constraints

Note
AutoTags are not currently supported with AWS CloudFormation StackSets.

A stack set constraint allows you to configure product deployment options using AWS CloudFormation StackSets. You can specify multiple accounts and regions for the product launch. End users can manage those accounts and determine where products deploy and the order of deployment.

To apply a stack set constraint to a product

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose the portfolio with the product you want.
3. Choose the **Constraints** tab and then choose **Create constraints**.
4. In **Product**, choose the product. In **Constraint type**, choose **Stack Set**.
5. Configure the accounts, regions, and permissions for your stack set constraints.
   - In **Account settings**, identify the accounts where you want to create products.
   - In **Region settings**, choose the geographic regions to deploy products and the order you want those products to be deployed in those regions.
   - In **Permissions**, choose an IAM StackSet Administrator Role to manage your target accounts. If you don't choose a role, StackSets uses the default ARN. Learn more about setting up stack set permissions.
6. Choose **Create**.

AWS Service Catalog Template Constraints

To limit the options that are available to end users when they launch a product, you apply template constraints. Apply template constraints to ensure that the end users can use products without breaching the compliance requirements of your organization. You apply template constraints to a product in an AWS Service Catalog portfolio. A portfolio must contain one or more products before you can define template constraints.

A template constraint consists of one or more rules that narrow the allowable values for parameters that are defined in the product's underlying AWS CloudFormation template. The parameters in an AWS CloudFormation template define the set of values that users can specify when creating a stack. For example, a parameter might define the various instance types that users can choose from when launching a stack that includes EC2 instances.

If the set of parameter values in a template is too broad for the target audience of your portfolio, you can define template constraints to limit the values that users can choose when launching a product. For example, if the template parameters include EC2 instance types that are too large for users who should use only small instance types (such as `t2.micro` or `t2.small`), then you can add a template constraint to limit the instance types that end users can choose. For more information about AWS CloudFormation template parameters, see Parameters in the *AWS CloudFormation User Guide*.

Template constraints are bound within a portfolio. If you apply template constraints to a product in one portfolio, and if you then include the product in another portfolio, the constraints will not apply to the product in the second portfolio.

If you apply a template constraint to a product that has already been shared with users, the constraint is active immediately for all subsequent product launches and for all versions of the product in the portfolio.

You define template constraint rules by using a rule editor or by writing the rules as JSON text in the AWS Service Catalog administrator console. For more information about rules, including syntax and examples, see Template Constraint Rules (p. 51).

To test a constraint prior to releasing it to users, create a test portfolio that contains the same products and test the constraints with that portfolio.

To apply template constraints to a product

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. On the **Portfolios** page, choose the portfolio that contains the product to which you want to apply a template constraint.
3. Expand the **Constraints** section and choose **Add constraints**.
4. In the **Select product and type** window, for **Product** choose the product for which you want to define the template constraints. Then, for **Constraint type**, choose **Template**. Choose **Continue**.
5. On the **Template constraint builder** page, edit the constraint rules by using the JSON editor or the rule builder interface.

   - To edit the JSON code for the rule, choose the **Constraint Text Editor** tab. Several samples are provided on this tab to help you get started.

     To build the rules by using a rule builder interface, choose the **Rule Builder** tab. On this tab, you can choose any parameter that is specified in the template for the product, and you can specify the allowable values for that parameter. Depending on the type of parameter, you specify the allowable values by choosing items in a checklist, by specifying a number, or by specifying a set of values in a comma-separated list.

     When you have finished building a rule, choose **Add rule**. The rule appears in the table on the **Rule Builder** tab. To review and edit the JSON output, choose the **Constraint Text Editor** tab.
6. When you are done editing the rules for your constraint, choose **Submit**. To see the constraint, go to the portfolio details page and expand **Constraints**.

Template Constraint Rules

The rules that define template constraints in an AWS Service Catalog portfolio describe when end users can use the template and which values they can specify for parameters that are declared in the AWS CloudFormation template used to create the product they are attempting to use. Rules are useful for preventing end users from inadvertently specifying an incorrect value. For example, you can add a rule to verify whether end users specified a valid subnet in a given VPC or used `m1.small` instance types for test environments. AWS CloudFormation uses rules to validate parameter values before it creates the resources for the product.

Each rule consists of two properties: a rule condition (optional) and assertions (required). The rule condition determines when a rule takes effect. The assertions describe what values users can specify for a particular parameter. If you don't define a rule condition, the rule's assertions always take effect. To define a rule condition and assertions, you use *rule-specific intrinsic functions*, which are functions that can only be used in the `Rules` section of a template. You can nest functions, but the final result of a rule condition or assertion must be either true or false.

As an example, assume that you declared a VPC and a subnet parameter in the `Parameters` section. You can create a rule that validates that a given subnet is in a particular VPC. So when a user specifies a VPC, AWS CloudFormation evaluates the assertion to check whether the subnet parameter value is in that VPC before creating or updating the stack. If the parameter value is invalid, AWS CloudFormation immediately fail to create or update the stack. If users don't specify a VPC, AWS CloudFormation doesn't check the subnet parameter value.

Syntax

The `Rules` section of a template consists of the key name `Rules`, followed by a single colon. Braces enclose all rule declarations. If you declare multiple rules, they are delimited by commas. For each rule, you declare a logical name in quotation marks followed by a colon and braces that enclose the rule condition and assertions.

A rule can include a `RuleCondition` property and must include an `Assertions` property. For each rule, you can define only one rule condition; you can define one or more asserts within the `Assertions` property. You define a rule condition and assertions by using rule-specific intrinsic functions, as shown in the following pseudo template:

```
"Rules" : {
  "Rule01" : {
    "RuleCondition" : { Rule-specific intrinsic function },
    "Assertions" : [
      {
        "Assert" : { Rule-specific intrinsic function },
        "AssertDescription" : "Information about this assert"
      },
      {
        "Assert" : { Rule-specific intrinsic function },
        "AssertDescription" : "Information about this assert"
      }
    ]
  },
  "Rule02" : {
    "Assertions" : [
      {
        "Assert" : { Rule-specific intrinsic function },
        "AssertDescription" : "Information about this assert"
      }
    ]
  }
}
```

The pseudo template shows a `Rules` section containing two rules named `Rule01` and `Rule02`. `Rule01` includes a rule condition and two assertions. If the function in the rule condition evaluates to true, both functions in each assert are evaluated and applied. If the rule condition is false, the rule doesn't take effect. `Rule02` always takes effect because it doesn't have a rule condition, which means the one assert is always evaluated and applied.

For information on rule-specific intrinsic functions to define rule conditions and assertions, see AWS Rule Functions in the *AWS CloudFormation User Guide*.

Example: Conditionally Verify a Parameter Value

The following two rules check the value of the `InstanceType` parameter. Depending on the value of the Environment parameter (`test` or `prod`), the user must specify `m1.small` or `m1.large` for the `InstanceType` parameter. The `InstanceType` and `Environment` parameters must be declared in the `Parameters` section of the same template.

```
"Rules" : {
```

```
  "testInstanceType" : {
    "RuleCondition" : {"Fn::Equals":[{"Ref":"Environment"}, "test"]},
    "Assertions" : [
      {
        "Assert" :  { "Fn::Contains" : [ ["m1.small"], {"Ref" : "InstanceType"} ] },
        "AssertDescription" : "For the test environment, the instance type must be
 m1.small"
      }
    ]
  },
  "prodInstanceType" : {
    "RuleCondition" : {"Fn::Equals":[{"Ref":"Environment"}, "prod"]},
    "Assertions" : [
      {
        "Assert" :  { "Fn::Contains" : [ ["m1.large"], {"Ref" : "InstanceType"} ] },
        "AssertDescription" : "For the prod environment, the instance type must be
 m1.large"
      }
    ]
  }
}
```

AWS Service Catalog Service Actions

AWS Service Catalog enables you to reduce administrative maintenance and end user training while adhering to compliance and security measures. With service actions, as the administrator you can enable end users to perform operational tasks, troubleshoot issues, run approved commands, or request permissions in AWS Service Catalog. You use AWS Systems Manager documents to define service actions. The AWS Systems Manager documents provide access to pre-defined actions that implement AWS best practices, such as Amazon EC2 stop and reboot, and you can define custom actions too.

In this tutorial, you provide end users with the ability to restart an Amazon EC2 instance. You add the necessary permissions, define the service action, associate the service action with a product, and test the end user experience using the action with a provisioned product.

Prerequisites

This tutorial assumes that you have full AWS administrator permissions, you are already familiar with AWS Service Catalog, and that you already have a base set of products, portfolios, and users. If you are not familiar with AWS Service Catalog, complete the Setting Up (p. 6) and Getting Started (p. 9) tasks before using this tutorial.

Topics

- Step 1: Configure end user permissions (p. 53)
- Step 2: Create a service action (p. 54)
- Step 3: Associate the service action with a product version (p. 55)
- Step 4: Test the end user experience (p. 55)
- Step 5: Troubleshooting (p. 55)

Step 1: Configure end user permissions

End user accounts must have the necessary permissions to view and perform specific service actions. In this example, the end user needs permission to access the AWS Service Catalog service actions feature and to perform an Amazon EC2 restart.

To update permissions

1. Open the AWS Identity and Access Management (IAM) console at https://console.aws.amazon.com/iam/.
2. From the menu, choose **Groups**.
3. On the **Groups** page, select the groups used by end users to access AWS Service Catalog resources. In this example, we select the end user group. In your own implementation, choose the group that is used by the relevant end users.
4. On the **Permissions** tab of your group's detail page, you either create a new policy or edit an existing policy. In this example, we add permissions to the existing policy by selecting the custom policy created for the group's AWS Service Catalog Provision and Terminate permissions.
5. On the **Policy** page, choose **Edit Policy** to add the necessary permissions. You can use either the visual editor or the JSON editor to edit the policy. In this example, we use the JSON editor to add the permissions. For this tutorial, add the following permissions to the policy:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "Stmt1536341175150",
   "Action": [
    "servicecatalog:ListServiceActionsForProvisioningArtifact",
    "servicecatalog:ExecuteprovisionedProductServiceAction",
    "ssm:DescribeDocument",
    "ssm:GetAutomationExecution",
    "ssm:StartAutomationExecution",
    "ssm:StopAutomationExecution",
    "cloudformation:ListStackResources",
    "ec2:DescribeInstanceStatus",
    "ec2:StartInstances",
    "ec2:StopInstances"
   ],
   "Effect": "Allow",
   "Resource": "*"
  }
 ]
}
```

6. After you edit the policy, review and approve the change to the policy. Users in the end user group now have the necessary permissions to perform the Amazon EC2 restart action in AWS Service Catalog.

Step 2: Create a service action

Next, you create a service action to restart Amazon EC2 instances.

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/sc/.
2. From the menu, choose **Service actions**.
3. On the **service actions** page, choose **Create new action**.
4. On the **Create action** page, choose an AWS Systems Manager document to define the service action. The Amazon EC2 Instance Restart action is defined by an AWS Systems Manager document, so we keep the default option on the drop-down menu, **Amazon documents**.
5. Choose the **AWS-RestartEC2Instance** action.

6. Provide a name and description for the action that make sense for your environment and team. The end user will see this description, so choose something that helps them understand what the action does.
7. Under **Parameter and target configuration**, choose the SSM document parameter that will be the target of the action (for example, the **Instance ID**), and choose the target of the parameter. Choose **Add parameter** to add additional parameters.
8. Under **Permissions**, choose a role. We are using default permissions for this example. Other permission configurations are possible and are defined on this page.
9. After you have reviewed the configuration, choose **Create action**.
10. On the next page, a confirmation appears when the action has been created and is ready to use.

Step 3: Associate the service action with a product version

After you define an action, you must associate a product with that action.

1. On the **Service actions** page, choose **AWS-RestartEC2instance**, and then choose **Associate action**.
2. On the **Associate action** page, choose the product that you want your end users to take the service action on. In this example, we choose **Linux Desktop**.
3. Select a product version. Note that you can use the topmost check box to select all versions.
4. Choose **Associate action**.
5. On the next page, a confirmation message appears.

You have now created the service action in AWS Service Catalog. The next step of this tutorial is to use the service action as an end user.

Step 4: Test the end user experience

End users can perform service actions on provisioned products. For the purposes of this tutorial, the end user must have at least one provisioned product. The provisioned product should be launched from the product version that you associated with the service action in the previous step.

To access the service action as an end user

1. Log in to the AWS Service Catalog console as an end user.
2. On the AWS Service Catalog dashboard, in the navigation pane, choose **Provisioned products list**. The list shows the products that are provisioned for the end user's account.
3. On the **Provisioned products list** page, choose the instance that is provisioned.
4. On the **Provisioned product details** page, choose **Actions** in the upper right side, and then choose the **AWS-RestartEC2instance** action.
5. Confirm that you want to execute the custom action. You receive confirmation that the action has been sent.

Step 5: Troubleshooting

If your service action execution fails, you can find the error message in the **Outputs** section of the service action execution event on the **Provisioned product** page. Below you can see explanations for common error messages you may find.

Note
The exact text of the error message is subject to change, so you should avoid using these in any kind of automated process.

Internal failure

AWS Service Catalog experienced an internal error. Try again later. If the issue persists, contact customer support.

An error occurred (ThrottlingException) when calling the StartAutomationExecution operation

The service action execution was throttled by the backend service, such as SSM.

Access denied while assuming the role

AWS Service Catalog was unable to assume the role specified in the service action definition. Make sure that the *servicecatalog.amazonaws.com* principal, or a regional principal such as *servicecatalog.us-east-1.amazonaws.com*, is allowlisted in the role's trust policy.

An error occurred (AccessDeniedException) when calling the StartAutomationExecution operation: User is not authorized to perform: ssm:StartAutomationExecution on the resource.

The role specified in the service action definition does not have permissions to invoke ssm:StartAutomationExecution. Make sure the role has the appropriate SSM permissions.

Cannot find any resources with type *`TargetType`* in provisioned product

The provisioned product does not contain any resources that match the target type specified in the SSM document, such as AWS::EC2::Instance. Check your provisioned product for these resources or confirm the document is correct.

Document with that name does not exist

The document specified in the service action definition does not exist.

Failed to describe SSM Automation document

AWS Service Catalog encountered an unknown exception from SSM when trying to describe the specified document.

Failed to retrieve credentials for role

AWS Service Catalog encountered an unknown error when assuming the specified role.

Parameter has value "*`InvalidValue`*" not found in *`{ValidValue1}, {ValidValue2}`*

The parameter value passed to SSM is not in the allowed values list for the document. Confirm the parameters provided are valid, and try again.

Parameter type error. The value supplied for *`ParameterName`* is not a valid string.

The value of the parameter passed to SSM is not valid for the type on the document.

Parameter is not defined in service action definition

A parameter was passed to AWS Service Catalog that is not defined in the service action definition. You can only use parameters defined in the service action definition.

Step fails when it is executing/canceling action. *`Error message.`* Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

A step in the SSM automation document failed. See the error in the message to troubleshoot further.

The following values for the parameter are not allowed because they are not in the provisioned product: *InvalidResourceId*

The user requested action on a resource that is not in the provisioned product.

TargetType not defined for SSM Automation document

Service actions require SSM automation documents to have a TargetType defined. Check your SSM automation document.

Adding AWS Marketplace Products to Your Portfolio

You can add AWS Marketplace products to your portfolios to make those products available to your AWS Service Catalog end users.

AWS Marketplace is an online store in which you can find, subscribe to, and immediately start using a large selection of software and services. The types of products in AWS Marketplace include databases, application servers, testing tools, monitoring tools, content management tools, and business intelligence software. AWS Marketplace is available at https://aws.amazon.com/marketplace. Note that you can't add software as a service (SaaS) products in AWS Marketplace.

You distribute an AWS Marketplace product to AWS Service Catalog end users by defining the product in an AWS CloudFormation template and adding the template to a portfolio. Any end user who has access to the portfolio will be able to launch the product from the console.

AWS Marketplace supports AWS Service Catalog directly or subscribe and add products using the manual option. We recommend adding products using the functionality specifically designed for AWS Service Catalog.

Managing AWS Marketplace Products Using AWS Service Catalog

You can add your subscribed AWS Marketplace products directly to AWS Service Catalog using a custom interface. In AWS Marketplace, choose **Service Catalog**. For more information, see Copying Products to AWS Service Catalog in the *AWS Marketplace Help and FAQ*.

Managing and Adding AWS Marketplace Products Manually

Complete the following steps to subscribe to an AWS Marketplace product, define that product in an AWS CloudFormation template, and add the template to an AWS Service Catalog portfolio.

To subscribe to an AWS Marketplace product

1. Go to AWS Marketplace at https://aws.amazon.com/marketplace.
2. Browse the products or search to find the product that you want to add to your AWS Service Catalog portfolio. Choose the product to view the product details page.

3. Choose **Continue** to view the fulfillment page, and then choose the **Manual Launch** tab.

   The information on the fulfillment page includes the supported Amazon Elastic Compute Cloud (Amazon EC2) instance types, the supported AWS regions, and the Amazon Machine Image (AMI) ID that the product uses for each AWS region. Note that some choices will affect cost. You will use this information to customize the AWS CloudFormation template in later steps.

4. Choose **Accept Terms** to subscribe to the product.

   After you subscribe to a product, you can access the information on the product fulfillment page in AWS Marketplace at any time by choosing **Your Software**, and then choosing the product.

To define your AWS Marketplace product in an AWS CloudFormation template

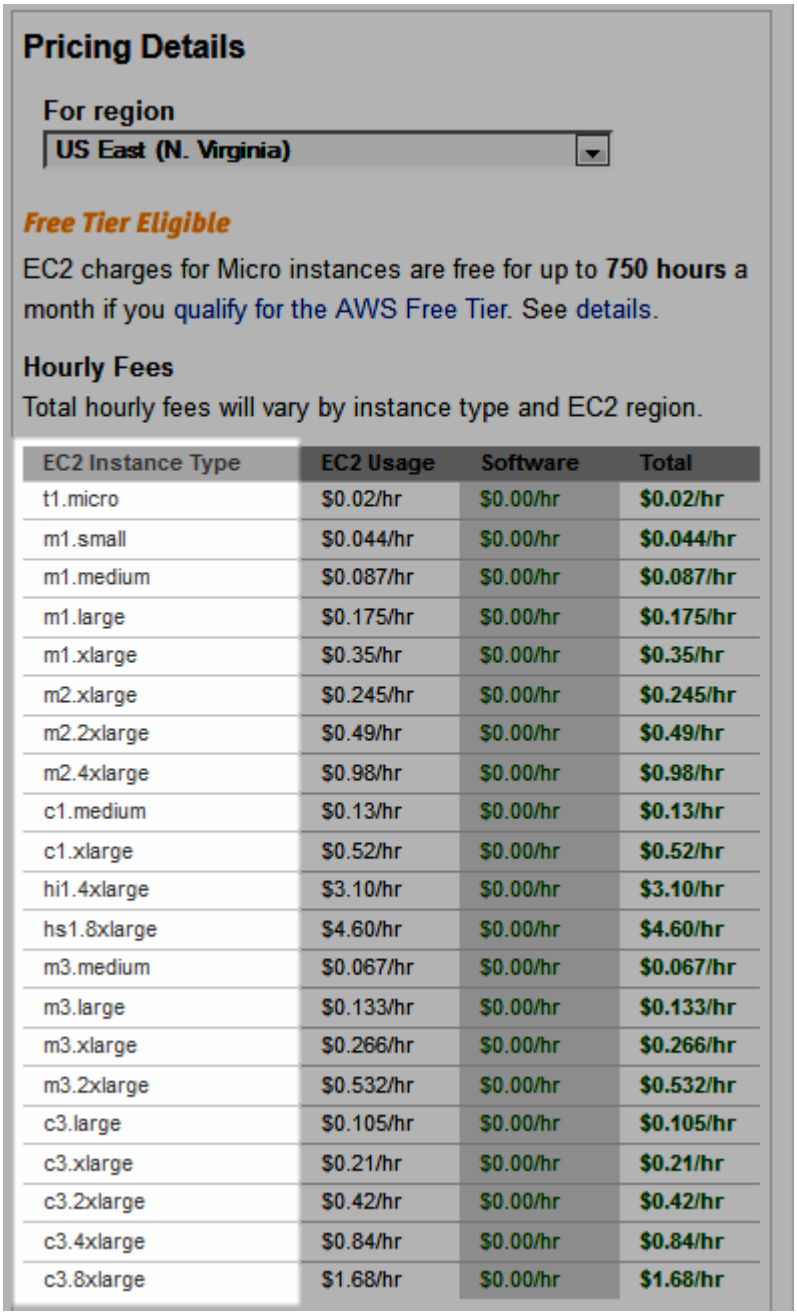
To complete the following steps, you will use one of the AWS CloudFormation sample templates as a starting point, and you will customize the template so that it represents your AWS Marketplace product. To access the sample templates, see Sample Templates in the *AWS CloudFormation User Guide*.

1. On the Sample Templates page in the *AWS CloudFormation User Guide*, choose a region that your product will be used in. The region must be supported by your AWS Marketplace product. You can view the supported regions on the product fulfillment page in AWS Marketplace.
2. To view a list of service sample templates that are appropriate for the region, choose the **Services** link.
3. You can use any of the samples that are appropriate for your needs as a starting point. The steps in this procedure use the **Amazon EC2 instance in a security group** template. To view the sample template, choose **View** , and then save a copy of the template locally so that you can edit it. Your local file must have the `.template` extension.
4. Open your template file in a text editor.
5. Customize the description at the top of the template. Your description might look like the following example:

   ```
   "Description": "Launches a LAMP stack from AWS Marketplace",
   ```

6. Customize the `InstanceType` parameter so that it includes only EC2 instance types that are supported by your product. If your template includes unsupported EC2 instance types, the product will fail to launch for your end users.

   a. On the product fulfillment page in AWS Marketplace, view the supported EC2 instance types in the **Pricing Details** section, as in the following example:



Pricing Details

For region

US East (N. Virginia)

Free Tier Eligible

EC2 charges for Micro instances are free for up to **750 hours** a month if you qualify for the AWS Free Tier. See details.

Hourly Fees

Total hourly fees will vary by instance type and EC2 region.

EC2 Instance Type	EC2 Usage	Software	Total
t1.micro	$0.02/hr	$0.00/hr	$0.02/hr
m1.small	$0.044/hr	$0.00/hr	$0.044/hr
m1.medium	$0.087/hr	$0.00/hr	$0.087/hr
m1.large	$0.175/hr	$0.00/hr	$0.175/hr
m1.xlarge	$0.35/hr	$0.00/hr	$0.35/hr
m2.xlarge	$0.245/hr	$0.00/hr	$0.245/hr
m2.2xlarge	$0.49/hr	$0.00/hr	$0.49/hr
m2.4xlarge	$0.98/hr	$0.00/hr	$0.98/hr
c1.medium	$0.13/hr	$0.00/hr	$0.13/hr
c1.xlarge	$0.52/hr	$0.00/hr	$0.52/hr
hi1.4xlarge	$3.10/hr	$0.00/hr	$3.10/hr
hs1.8xlarge	$4.60/hr	$0.00/hr	$4.60/hr
m3.medium	$0.067/hr	$0.00/hr	$0.067/hr
m3.large	$0.133/hr	$0.00/hr	$0.133/hr
m3.xlarge	$0.266/hr	$0.00/hr	$0.266/hr
m3.2xlarge	$0.532/hr	$0.00/hr	$0.532/hr
c3.large	$0.105/hr	$0.00/hr	$0.105/hr
c3.xlarge	$0.21/hr	$0.00/hr	$0.21/hr
c3.2xlarge	$0.42/hr	$0.00/hr	$0.42/hr
c3.4xlarge	$0.84/hr	$0.00/hr	$0.84/hr
c3.8xlarge	$1.68/hr	$0.00/hr	$1.68/hr

b. In your template, change the default instance type to a supported EC2 instance type of your choice.

c. Edit the `AllowedValues` list so that it includes only EC2 instance types that are supported by your product.

d. Remove any EC2 instance types that you do not want your end users to use when they launch the product from the `AllowedValues`list .

When you are done editing the `InstanceType` parameter, it might look similar to the following example:

```
    "InstanceType" : {
```

```
      "Description" : "EC2 instance type",
      "Type" : "String",
      "Default" : "m1.small",
      "AllowedValues" : [ "t1.micro", "m1.small", "m1.medium", "m1.large", "m1.xlarge",
 "m2.xlarge", "m2.2xlarge", "m2.4xlarge", "c1.medium", "c1.xlarge", "c3.large",
 "c3.large", "c3.xlarge", "c3.xlarge", "c3.4xlarge", "c3.8xlarge" ],
      "ConstraintDescription" : "Must be a valid EC2 instance type."
    },
```

7. In the `Mappings` section of your template, edit the `AWSInstanceType2Arch` mappings so that only supported EC2 instance types and architectures are included.

   a. Edit the list of mappings by removing all EC2 instance types that are not included in the `AllowedValues` list for the `InstanceType` parameter.

   b. Edit the `Arch` value for each EC2 instance type to be the architecture type that is supported by your product. Valid values are `PV64`, `HVM64`, and `HVMG2`. To learn which architecture your product supports, refer to the product details page in AWS Marketplace. To learn which architectures are supported by EC2 instance families, see Amazon Linux AMI Instance Type Matrix.

   When you have finished editing the `AWSInstanceType2Arch` mappings, it might look similar to the following example:

```
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "PV64"  },
      "m1.small"    : { "Arch" : "PV64"  },
      "m1.medium"   : { "Arch" : "PV64"  },
      "m1.large"    : { "Arch" : "PV64"  },
      "m1.xlarge"   : { "Arch" : "PV64"  },
      "m2.xlarge"   : { "Arch" : "PV64"  },
      "m2.2xlarge"  : { "Arch" : "PV64"  },
      "m2.4xlarge"  : { "Arch" : "PV64"  },
      "c1.medium"   : { "Arch" : "PV64"  },
      "c1.xlarge"   : { "Arch" : "PV64"  },
      "c3.large"    : { "Arch" : "PV64"  },
      "c3.xlarge"   : { "Arch" : "PV64"  },
      "c3.2xlarge"  : { "Arch" : "PV64"  },
      "c3.4xlarge"  : { "Arch" : "PV64"  },
      "c3.8xlarge"  : { "Arch" : "PV64"  }
    }
,
```

8. In the `Mappings` section of your template, edit the `AWSRegionArch2AMI` mappings to associate each AWS region with the corresponding architecture and AMI ID for your product.

   a. On the product fulfillment page in AWS Marketplace, view the AMI ID that your product uses for each AWS region, as in the following example:

Region	ID	
US East (N. Virginia)	ami-	Launch with EC2 Console
US West (Oregon)	ami-	Launch with EC2 Console
US West (N. California)	ami-	Launch with EC2 Console
EU (Frankfurt)	ami-	Launch with EC2 Console
EU (Ireland)	ami-	Launch with EC2 Console
Asia Pacific (Singapore)	ami-	Launch with EC2 Console
Asia Pacific (Sydney)	ami-	Launch with EC2 Console
Asia Pacific (Tokyo)	ami-	Launch with EC2 Console
South America (Sao Paulo)	ami-	Launch with EC2 Console

b. In your template, remove the mappings for any regions that you do not support.

c. Edit the mapping for each region to remove the unsupported architectures (`PV64`, `HVM64`, or `HVMG2`) and their associated AMI IDs.

d. For each remaining region and architecture mapping, specify the corresponding AMI ID from the product details page in AWS Marketplace.

When you have finished editing the `AWSRegionArch2AMI` mappings, your code might look similar to the following example:

```
"AWSRegionArch2AMI" : {
  "us-east-1"         : {"PV64" : "ami-nnnnnnnn"},
  "us-west-2"         : {"PV64" : "ami-nnnnnnnn"},
  "us-west-1"         : {"PV64" : "ami-nnnnnnnn"},
  "eu-west-1"         : {"PV64" : "ami-nnnnnnnn"},
  "eu-central-1"      : {"PV64" : "ami-nnnnnnnn"},
  "ap-northeast-1"    : {"PV64" : "ami-nnnnnnnn"},
  "ap-southeast-1"    : {"PV64" : "ami-nnnnnnnn"},
  "ap-southeast-2"    : {"PV64" : "ami-nnnnnnnn"},
  "sa-east-1"         : {"PV64" : "ami-nnnnnnnn"}
}
```

You can now use the template to add the product to an AWS Service Catalog portfolio. If you want to make additional changes, see Working with AWS CloudFormation Templates to learn more about templates.

To add your AWS Marketplace product to an AWS Service Catalog portfolio

1. Sign in to the AWS Management Console and navigate to the AWS Service Catalog administrator console at https://console.aws.amazon.com/servicecatalog/.
2. On the **Portfolios** page, choose the portfolio that you want to add your AWS Marketplace product to.
3. On the portfolio details page, choose **Upload new product**.
4. Type the requested product and support details.
5. On the **Version details** page, choose **Upload a template file**, choose **Browse**, and then choose your template file.
6. Type a version title and description.

7. Choose **Next**.
8. On the **Review** page, verify that the summary is accurate, and then choose **Confirm and upload**. The product is added your portfolio. It is now available to end users who have access to the portfolio.

Using AWS CloudFormation StackSets

Note
This feature is currently in beta mode. AutoTags are not currently supported with AWS CloudFormation StackSets.

You can use AWS CloudFormation StackSets to launch AWS Service Catalog products across multiple regions and accounts. You can specify the order in which products deploy sequentially within regions. Across accounts, products are deployed in parallel. When launching, users can specify failure tolerance and the maximum number of accounts in which to deploy in parallel. For more information, see Working with AWS CloudFormation StackSets.

Stack sets vs. stack instances

A *stack* lets you create stacks in AWS accounts across regions by using a single AWS CloudFormation template.

A *stack instance* refers to a stack in a target account within a region and is associated with only one stack set.

For more information, see StackSets Concepts.

Stack set constraints

In AWS Service Catalog, you can use stack set constraints to configure product deployment options.

For more information, see Amazon Service Catalog Stack Set Constraints.

Managing Budgets

You can use AWS Budgets to track your service costs and usage within AWS Service Catalog. You can associate budgets with AWS Service Catalog products and portfolios.

AWS Budgets gives you the ability to set custom budgets that alert you when your costs or usage exceed (or are forecasted to exceed) your budgeted amount. Information about AWS Budgets is available at https://aws.amazon.com/aws-cost-management/aws-budgets.

Tasks

- Prerequisites (p. 62)
- Creating a Budget (p. 64)
- Associating a Budget (p. 64)
- Viewing a Budget (p. 65)
- Disassociating a Budget (p. 65)

Prerequisites

Before using AWS Budgets, you need to activate cost allocation tags in the AWS Billing and Cost Management console. For more information, see Activating User-Defined Cost Allocation Tags in the *AWS Billing and Cost Management User Guide*.

Note
Tags take up to 24 hours to activate.

You also need to enable user access to the AWS Billing and Cost Management console for any users or groups who will be using the Budgets feature. You can do this by creating a new policy for your users.

To allow IAM users to create budgets, you must also allow users to view billing information. If you want to use Amazon SNS notifications, you can give users the ability to create Amazon SNS notifications, as shown in the policy example below.

To create the budgets policy

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**.
3. In the content pane, choose **Create policy**.
4. Choose the **JSON** tab and copy the text from the following JSON policy document. Paste this text into the **JSON** text box.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1435216493000",
            "Effect": "Allow",
            "Action": [
                "aws-portal:ViewBilling",
                "aws-portal:ModifyBilling",
                "budgets:ViewBudget",
                "budgets:ModifyBudget"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Stmt1435216552000",
            "Effect": "Allow",
            "Action": [
                "sns:*"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1"
            ]
        }
    ]
}
```

5. When you are finished, choose **Review policy**. The Policy Validator reports any syntax errors.
6. On the **Review** page, give your policy a name. Review the policy **Summary** to see the permissions granted by your policy, and then choose **Create policy** to save your work.

   The new policy appears in the list of managed policies and is ready to attach to your users and groups. For more information, see Create and Attach Customer Managed Policy in the *AWS Identity and Access Management User Guide*.

Creating a Budget

In the AWS Service Catalog administrator console, the **Products** and **Portfolios** pages list information about existing products and portfolios and allow you to take actions on them. To create a budget, first decide which product or portfolio you want to associate the budget to.

To create a budget

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose either **Products** or **Portfolios**.
3. Select the product or portfolio you want to add a budget to.
4. Open the **Actions** menu, then choose **Create budget**.
5. On the **Budget creation** page, associate one tag type to your budget.

   There are two types of tags: AutoTags and TagOptions. AutoTags are tags that identify the portfolio, product, and user that launched a product, and are automatically applied by AWS Service Catalog to provisioned resources. A TagOption is an administrator-defined key-value pair managed in AWS Service Catalog.

   In order for spending that occurs on a portfolio or product to reflect on the associated budget, they must have the same tag. Note that a tag key being used for the first time can take 24 hours to activate. For more information, see the section called “Prerequisites” (p. 62).
6. Choose **Continue**.
7. You will be taken to the **Set up your budget** page. Continue your budget setup by following the steps on Creating a Budget.

After you create a budget, you need to associate it to the product or portfolio.

Associating a Budget

Each portfolio or product can have one budget associated to it, but each budget can be associated to multiple products and portfolios.

When you associate a budget to a product or portfolio, you will be able to view information about the budget from that product or portfolio's detail page. In order for spending that occurs on the product or portfolio to be reflected on the budget, you must associate the same tags on both the budget and the product or portfolio.

> **Note**
> If you delete a budget from within AWS Budgets, existing associations with AWS Service Catalog products and portfolios will still exist but AWS Service Catalog will be unable to display any information about the deleted budget.

To associate a budget

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose either **Products** or **Portfolios**.
3. Select the product or portfolio you want to associate a budget to.
4. Open the **Actions** menu, then choose **Associate budget**.
5. On the **Budget association** page, select an existing budget. Then choose **Continue**.
6. The **Portfolios** or **Products** table will now include data for the budget you just added.

Viewing a Budget

If a budget is associated to a product, you can view information about the budget on the **Products** and **Product details** page. If a budget is associated to a portfolio, you can view information about the budget on the **Portfolios** and **Portfolio details** page.

Both the **Portfolios** and **Products** pages display budget information for existing resources. You can see columns displaying **Current vs. budget** and **Forecast vs. budget**.

When you click on a product or portfolio, you are taken to a detail page. These **Portfolio detail** and **Product detail** pages have a section with detailed information about the associated budget. You can see the budgeted amount, current spend, and forecasted spend. You also have the option to view budget details and edit the budget.

Disassociating a Budget

You can disassociate a budget from a portfolio or product.

> **Note**
> If you delete a budget from within AWS Budgets, existing associations with AWS Service Catalog products and portfolios will still exist but AWS Service Catalog will be unable to display any information about the deleted budget.

To disassociate a budget

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose **Products** or **Portfolios**.
3. Select the product or portfolio you want to disassociate a budget from.
4. Open the **Actions** menu, then choose **Disassociate budget**.
5. An alert will appear asking you to confirm that you want to disassociate the budget. Choose **Confirm**.

Managing Provisioned Products

AWS Service Catalog provides an interface for managing provisioned products. You can view, update, and terminate all provisioned products for your catalog based on access level. Refer to the following sections for example procedures.

Topics

- Managing All Provisioned Products as Administrator (p. 66)
- Changing Provisioned Product Owner (p. 66)
- Tutorial: Identifying User Resource Allocation (p. 67)

Managing All Provisioned Products as Administrator

To manage all provisioned products for the account, you will need **AWSServiceCatalogAdminFullAccess** or equivalent access to the provisioned product write operations. For more information, see Identity and Access Management in AWS Service Catalog (p. 20).

To view and manage all provisioned products

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.

   If you are already logged in to the AWS Service Catalog console, choose **Service Catalog**, **End user**.
2. If necessary, scroll down to the **Provisioned products** section.
3. In the **Provisioned products** section, choose the **View:** list and select the level of access you wish to see: **User**, **Role**, or **Account**. This displays all the provisioned products in the catalog.
4. Choose a provisioned product to view, update, or terminate. For more information about the information provided in this view, see Viewing Provisioned Product Information.

Changing Provisioned Product Owner

You can change the owner of a provisioned product anytime. You need to know the ARN of the user or role you want to set as the new owner.

By default, this feature is available to administrators using the **AWSServiceCatalogAdminFullAccess** managed policy. You can enable it for end users by granting them the **servicecatalog:UpdateProvisionedProductProperties** permission in AWS Identity and Access Management (IAM).

To change the owner of a provisioned product

1. In the AWS Service Catalog console, choose **Provisioned products list**.
2. Locate the provisioned product you want to update, then choose the three dots beside it and choose **Change provisioned product owner**. You can also find the **Change owner** option on the provisioned product's detail page, in the **Actions** menu.

3. In the dialog box, enter the ARN of the user or role you want to set as the new owner. An ARN begins with `arn:` and includes other information separated by colons or slashes, for example, `arn:aws:iam::123456789012:user/NewOwner`.
4. Choose **Submit**. You will see a success message when the owner has been updated.

See Also

- UpdateProvisionedProductProperties

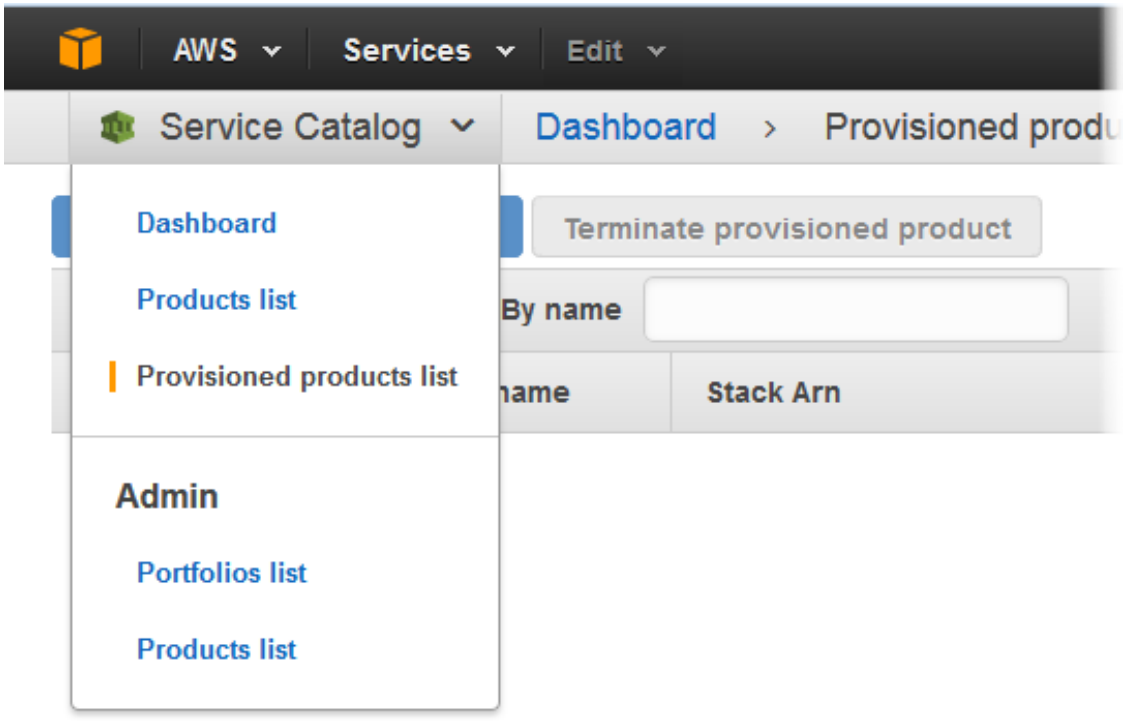
Tutorial: Identifying User Resource Allocation

You can identify the user who provisioned a product and resources associated with the product using the AWS Service Catalog console. This tutorial helps translate this example to your own specific provisioned products.

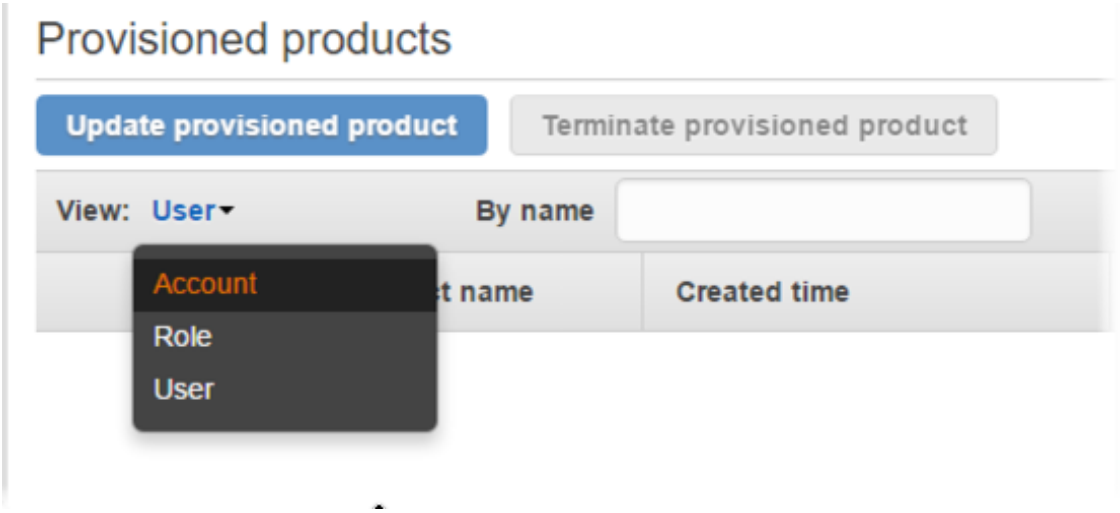
To manage all provisioned products for the account, you need **AWSServiceCatalogAdminFullAccess** or equivalent access to the provisioned product write operations. For more information, see Identity and Access Management in AWS Service Catalog (p. 20).

To identify the user who provisioned a product and the associated resources

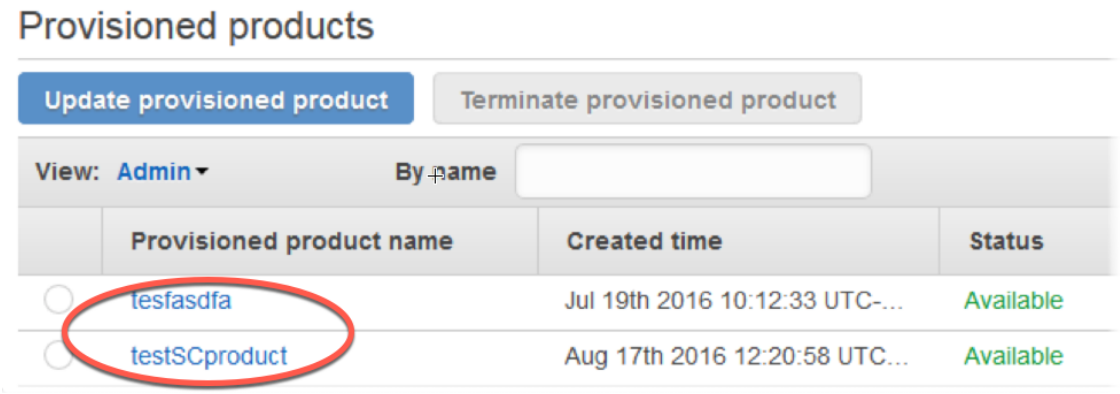
1. Navigate to the provisioned products console in AWS Service Catalog console.



2. In the **Provisioned products** pane, for **View:**, choose **Account**.



3. Identify the provisioned product to investigate, and select the provisioned product.

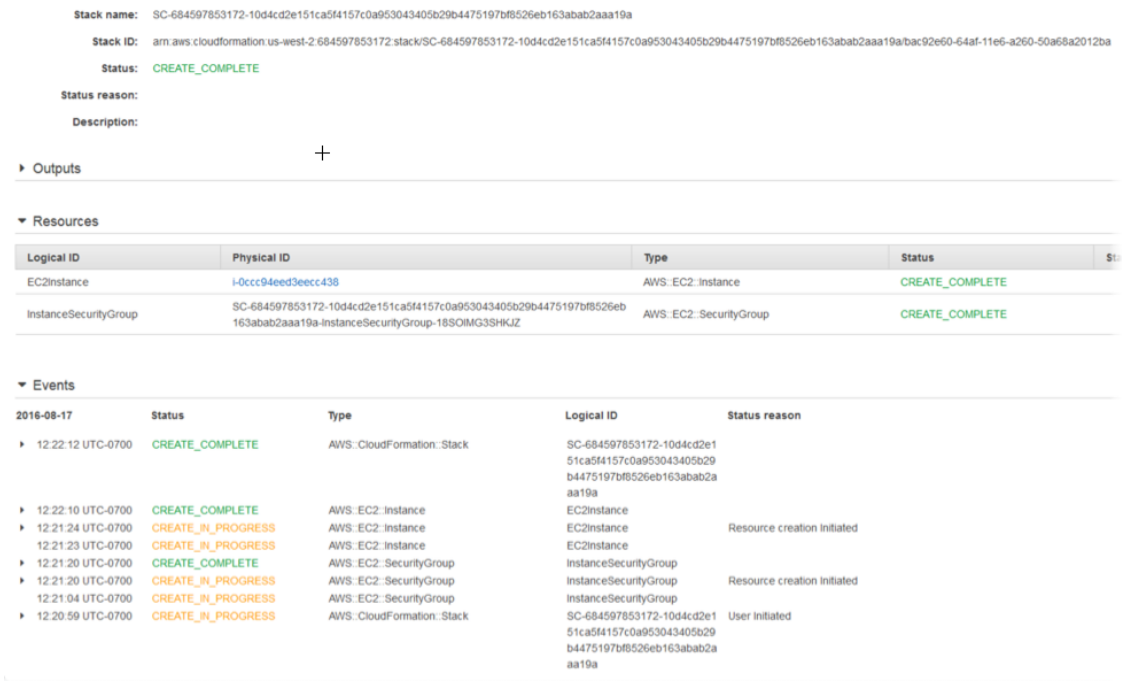


4. Expand the **Events** section and note the **Provisioned product ID** and **CloudformationStackARN** values.
5. Use the provisioned product ID to identify the CloudTrail record that corresponds to this launch and identify the requesting user (typically, this is entered as an email address during federation). In this example, it is "steve".

```
{
  "eventVersion":"1.03","userIdentity":
  {
    "type":"AssumedRole",
    "principalId":"[id]:steve",
    "arn":"arn:aws:sts::[account number]:assumed-role/SC-usertest/steve",
    "accountId":[account number],
    "accessKeyId":[access key],
    "sessionContext":
    {
      "attributes":
      {
        "mfaAuthenticated":[boolean],
        "creationDate":[timestamp]
      },
      "sessionIssuer":
      {
        "type":"Role",
        "principalId":"AROAJEXAMPLELH3QXY",
        "arn":"arn:aws:iam::[account number]:role/[name]",
```

```
        "accountId":[account number],
        "userName":[username]
      }
    }
  },
  "eventTime":"2016-08-17T19:20:58Z","eventSource":"servicecatalog.amazonaws.com",
  "eventName":"ProvisionProduct",
  "awsRegion":"us-west-2",
  "sourceIPAddress":[ip address],
  "userAgent":"Coral/Netty",
  "requestParameters":
  {
    "provisioningArtifactId":[id],
    "productId":[id],
    "provisioningParameters":[Shows all the parameters that the end user entered],
    "provisionToken":[token],
    "pathId":[id],
    "provisionedProductName":[name],
    "tags":[],
    "notificationArns":[]
  },
  "responseElements":
  {
    "recordDetail":
    {
      "provisioningArtifactId":[id],
      "status":"IN_PROGRESS",
      "recordId":[id],
      "createdTime":"Aug 17, 2016 7:20:58 PM",
      "recordTags":[],
      "recordType":"PROVISION_PRODUCT",
      "provisionedProductType":"CFN_STACK",
      "pathId":[id],
      "productId":[id],
      "provisionedProductName":"testSCproduct",
      "recordErrors":[],
      "provisionedProductId":[id]
    }
  },
  "requestID":[id],
  "eventID":[id],
  "eventType":"AwsApiCall",
  "recipientAccountId":[account number]
}
```


6. Use the `CloudformationStackARN` value to identify AWS CloudFormation events to find information about resources created. You can also use the AWS CloudFormation API to obtain this information. For more information, see AWS CloudFormation API Reference.



Note that you can perform steps 1 through 4 using the AWS Service Catalog API or the AWS CLI. For more information, see AWS Service Catalog Developer Guide and AWS Service Catalog Command Line Reference.

Managing Tags in AWS Service Catalog

AWS Service Catalog provides tags so you can categorize your resources. There are two types of tags: AutoTags and TagOptions.

AutoTags are tags that identify information about the origin of a provisioned resource in AWS Service Catalog and are automatically applied by AWS Service Catalog to provisioned resources.

TagOptions are key-value pairs managed in AWS Service Catalog that serve as templates for creating AWS tags.

Topics

- AWS Service Catalog AutoTags (p. 71)
- AWS Service Catalog TagOption Library (p. 72)

AWS Service Catalog AutoTags

AutoTags are tags that identify information about the origin of a provisioned resource in AWS Service Catalog and are automatically applied by AWS Service Catalog to provisioned resources.

AutoTags include tags for the unique identifiers for portfolio, product, user, product version, and provisioned product. This provides a set of tags that reflect the AWS Service Catalog structure that customers have configured in the catalog. AutoTags do not count against the customer's 50-tag limit.

AWS Service Catalog AutoTags can help provide consistent tagging for your resources, which is useful when setting budgets for a portfolio, product, or user. You can also use the AutoTags to identify resources for post-launch operations such as setting AWS Config rules. AutoTags for your provisioned resources can be viewed in the Tags section of the downstream services used for provisioning, such as AWS CloudFormation, Amazon EC2, and Amazon S3.

AutoTag details

- **aws:servicecatalog:portfolioArn** - The ARN of the portfolio from which the provisioned product was launched.
- **aws:servicecatalog:productArn** - The ARN of the product from which the provisioned product was launched.
- **aws:servicecatalog:provisioningPrincipalArn** - The ARN of the provisioning principal (user) who created the provisioned product.
- **aws:servicecatalog:provisionedProductArn** - The provisioned product ARN.
- **aws:servicecatalog:provisioningArtifactIdentifier** - The ID of the original provisioning artifact (product version).

> **Note**
> AWS Service Catalog recently added two new AutoTags, **aws:servicecatalog:provisionedProductArn** and **aws:servicecatalog:provisioningArtifactIdentifier**. These new AutoTags will be automatically backfilled during updates on provisioned products.

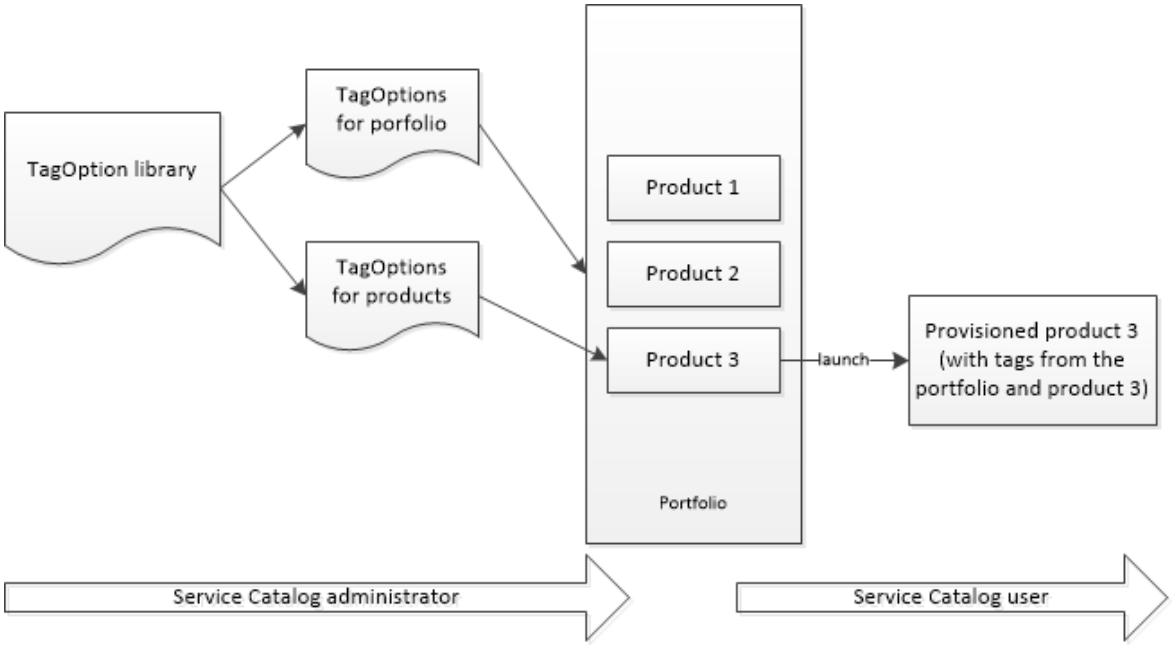
AWS Service Catalog TagOption Library

To allow administrators to easily manage tags on provisioned products, AWS Service Catalog provides a TagOption library. A TagOption is a key-value pair managed in AWS Service Catalog. It is not an AWS tag, but serves as a template for creating an AWS tag based on the TagOption.

The TagOption library makes it easier to enforce the following:

- A consistent taxonomy
- Proper tagging of AWS Service Catalog resources
- Defined, user-selectable options for allowed tags

Administrators can associate TagOptions with portfolios and products. During a product launch (provisioning), AWS Service Catalog aggregates the associated portfolio and product TagOptions, and applies them to the provisioned product, as shown in the following diagram.



With the TagOption library, you can deactivate TagOptions and retain their associations to portfolios or products, and reactivate them when you need them. This approach not only helps maintain library integrity, it also allows you to manage TagOptions that might be used intermittently, or only under special circumstances.

You manage TagOptions with the AWS Service Catalog console or the TagOption library API. For more information, see AWS Service Catalog API Reference.

Contents

- Launching a Product with TagOptions (p. 72)
- Managing TagOptions (p. 75)

Launching a Product with TagOptions

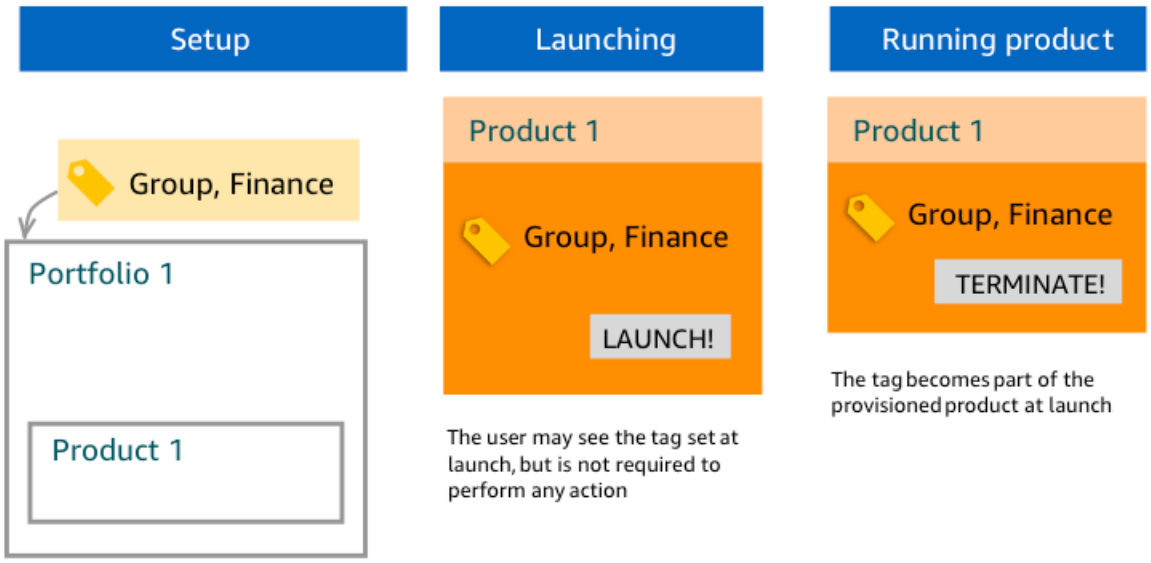
When a user launches a product that has TagOptions, AWS Service Catalog performs the following actions on your behalf:

- Collects all TagOptions for the product and the launching portfolio.
- Ensures that only TagOptions with unique keys are used in a tag on the provisioned product. Users get a multiple-choice value lists for a key. After the user chooses a value, it becomes a tag on the provisioned product.
- Allows users to add non-conflicting tags to the product during provisioning.

The following use cases demonstrate how TagOptions work during launch.

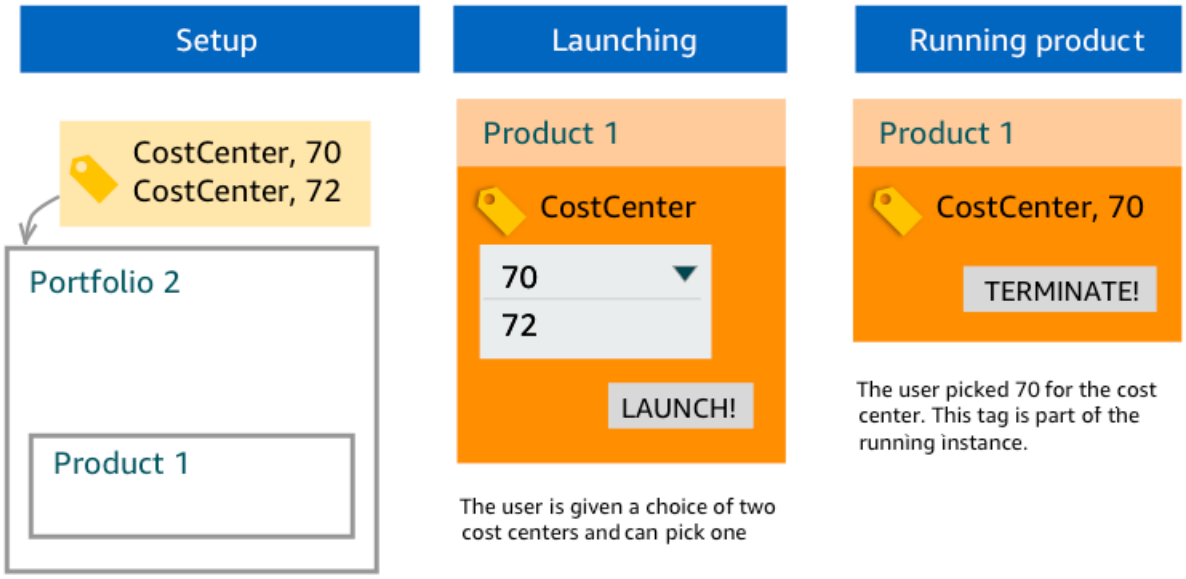
Example 1: A Unique TagOption Key

An administrator creates **TagOption[Group=Finance]** and associates it with **Portfolio1**, which has **Product1** with no TagOptions. When a user launches the provisioned product, the single TagOption becomes **Tag[Group=Finance]**, as follows:



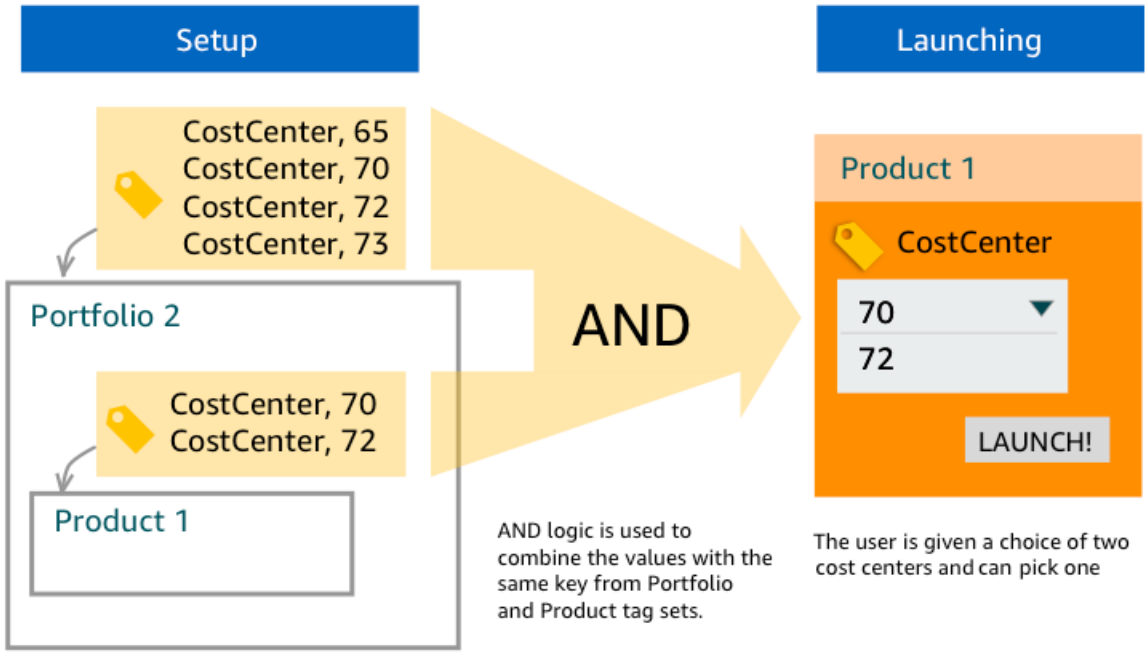
Example 2: A Set of TagOptions with the Same Key on a Portfolio

An administrator has placed two TagOptions with the same key on a portfolio, and there are no TagOptions with the same key on any products within that portfolio. During launch, the user must select one of the two values associated with the key. The provisioned product is then tagged with the key and the user-selected value.



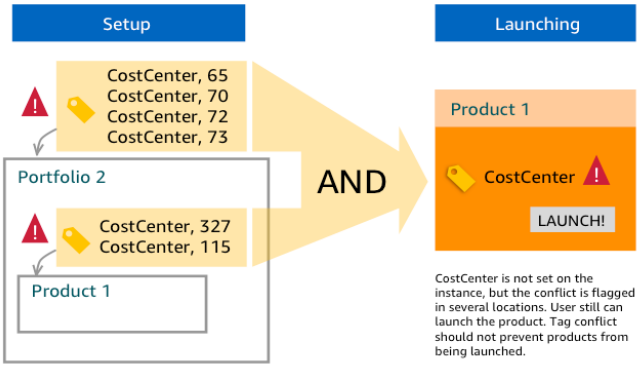
Example 3: A Set of TagOptions with the Same Key on Both the Portfolio and a Product in that Portfolio

An administrator has placed several TagOptions with the same key on a portfolio, and there are also several TagOptions with the same key on the product within that portfolio. AWS Service Catalog creates a set of values from the aggregation (logical AND operation) of the TagOptions. When the user launches the product, he or she sees and selects from this set of values. The provisioned product is tagged with the key and the user-selected value.



Example 4: Multiple TagOptions with the Same Key and Conflicting Values

An administrator has placed several TagOptions with the same key on a portfolio, and there are also several TagOptions with the same key on the product in that portfolio. AWS Service Catalog creates a set of values from the aggregation (logical AND operation) of the TagOptions. If the aggregation doesn't find values for the key, AWS Service Catalog creates a tag with the same key and a value of `sc-tagconflict-`*`portfolioid`*`-`*`productid`*, where *`portfolioid`* and *`productid`* are the ARNs of the portfolio and product. This ensures that the provisioned product is tagged with the correct key and with a value that the administrator can find and correct.



Managing TagOptions

As an administrator, you can perform the following actions to manage TagOptions in the TagOptions library:

- Create and delete
- Activate or deactivate
- Associate or disassociate
- Edit TagOption values

To create TagOptions in the console

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. In the left navigation menu, choose **TagOptions library**.
3. In **Create new TagOption**, enter a key and value, and then choose **Add**.

   After the new TagOption has been created, it's grouped by key-value pair and sorted alphabetically in the **TagOptions** list.

   To create a TagOption using the AWS Service Catalog API, see CreateTagOption.

To delete TagOptions in the console

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. In the left navigation menu, choose **TagOptions library** and then choose **Actions**.
3. Select **Delete** and confirm the deletion.

To activate or deactivate a TagOptions with a portfolio or product in the console

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. In the left navigation menu, choose **TagOptions library** and then choose **Actions**.
3. Select **Activate** or **Deactivate** and confirm the selection.

To associate or disassociate TagOptions in the console

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. In the left navigation menu, choose **TagOptions library** and then choose **Actions**.
3. Select **Associate** or **Disassociate** and confirm the selection.

   To associate TagOptions with a portfolio or product using the AWS Service Catalog API, see AssociateTagOptionWithResource.

   To remove (disassociate) TagOptions using the AWS Service Catalog API, see DisassociateTagOptionFromResource.

To edit values for TagOptions in the console

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. In the left navigation menu, choose **TagOptions library**.
3. Choose a TagOption and open the value. (The value is hyperlinked.) Then choose **Edit**.
4. In the **Value** field, edit the value and choose **Save changes**.

Monitoring in AWS Service Catalog

You can monitor your AWS Service Catalog resources using Amazon CloudWatch, which collects and processes raw data from AWS Service Catalog into readable metrics. These statistics are recorded for a period of two weeks, so that you can access historical information and gain a better perspective on how your service is performing. AWS Service Catalog metric data is automatically sent to CloudWatch in 1-minute periods. For more information about CloudWatch, see the Amazon CloudWatch User Guide.

For a list of available metrics and dimensions, see AWS Service Catalog CloudWatch Metrics (p. 77).

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Service Catalog and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. Before you start monitoring AWS Service Catalog, you should create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

Monitoring Tools

AWS provides various tools that you can use to monitor AWS Service Catalog. You can configure some of these tools to do the monitoring for you, while some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

Automated Monitoring Tools

You can use the following automated monitoring tools to watch AWS Service Catalog and report when something is wrong:

- Amazon CloudWatch alarms – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Amazon EC2 Auto Scaling policy. CloudWatch alarms do not invoke actions simply because they are in a particular state; the state must have changed and been maintained for a specified number of periods. To learn how to create an alarm, see Creating Amazon CloudWatch Alarms. For more information on using Amazon CloudWatch metrics with AWS Service Catalog, see AWS Service Catalog CloudWatch Metrics (p. 77).

AWS Service Catalog CloudWatch Metrics

You can monitor your AWS Service Catalog resources using Amazon CloudWatch, which collects and processes raw data from AWS Service Catalog into readable metrics. These statistics are recorded for a

period of two weeks, so that you can access historical information and gain a better perspective on how your service is performing. AWS Service Catalog metric data is automatically sent to CloudWatch in 1-minute periods. For more information about CloudWatch, see the Amazon CloudWatch User Guide.

Topics

- Enabling CloudWatch Metrics (p. 78)
- Available Metrics and Dimensions (p. 78)
- Viewing AWS Service Catalog Metrics (p. 79)

Enabling CloudWatch Metrics

Amazon CloudWatch metrics are enabled by default.

Available Metrics and Dimensions

The metrics and dimensions that AWS Service Catalog sends to Amazon CloudWatch are listed below.

AWS Service Catalog Metrics

The `AWS/ServiceCatalog` namespace includes the following metrics.

Metric	Description
`ProvisionedProductLaunch`	The number of provisioned products launched for a given product and provisioning artifact in a specified time period. Units: Count Valid statistics: Minimum, Maximum, Sum, Average

Dimensions for AWS Service Catalog Metrics

AWS Service Catalog sends the following dimensions to CloudWatch.

Dimension	Description
`State`	This dimension filters the data you request for all provisioned products launched with this specified state. This helps you categorize your data by the state of launch. Valid State: SUCCEEDED, FAILED
`ProductId`	This dimension filters the data you request for the identified product id only. This helps you to pinpoint an exact product from which to be launched.
`ProvisioningArtifactId`	This dimension filters the data you request for the identified provisioning artifact id only. This helps you to pinpoint an exact version of products from which to be launched.

Viewing AWS Service Catalog Metrics

You can view CloudWatch metrics in the CloudWatch console, which provides a fine-grained and customizable display of your resources, as well as the number of running tasks in a service.

Topics

- Viewing AWS Service Catalog Metrics in the CloudWatch Console (p. 79)

Viewing AWS Service Catalog Metrics in the CloudWatch Console

You can view AWS Service Catalog metrics in the CloudWatch console. The CloudWatch console provides a detailed view of AWS Service Catalog metrics, and you can tailor the views to suit your needs. For more information about CloudWatch, see the Amazon CloudWatch User Guide.

To view metrics in the CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Metrics** section in the left navigation, choose **Service Catalog**.
3. Choose the metrics to view.

Logging AWS Service Catalog API calls using AWS CloudTrail

AWS Service Catalog is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Service Catalog. CloudTrail captures all API calls for AWS Service Catalog as events. The calls captured include calls from the AWS Service Catalog console and code calls to the AWS Service Catalog API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Service Catalog. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history. Using the information collected by CloudTrail, you can determine the request that was made to AWS Service Catalog, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the Amazon CloudTrail User Guide .

AWS Service Catalog information in CloudTrail

CloudTrail is enabled on your AWS account when you create it. When activity occurs in AWS Service Catalog, that activity is recorded in a CloudTrail event along with other AWS service events in Event history. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for AWS Service Catalog, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail

- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All AWS Service Catalog actions are logged by CloudTrail and are documented in the https:// linkto.APIref.topic. For example, calls to the `CreatePortfolio`, `CreateProduct` and `UpdateProvisionedProduct` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Understanding AWS Service Catalog log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order. The following example shows a CloudTrail log entry that demonstrates the `CreateApplication` API.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "account",
        "arn": "arn:aws:iam::12345789012:user/dev-haw",
        "accountId": "12345789012",
        "accessKeyId": "keyId",
        "userName": "dev-haw"
    },
    "eventTime": "2020-09-23T21:07:58Z",
    "eventSource": "servicecatalog-appregistry.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "205.251.233.48",
    "userAgent": "aws-cli/1.18.140 Python/3.6.11
 Linux/4.9.217-0.1.ac.205.84.332.metal1.x86_64 botocore/1.17.63",
    "requestParameters": {
        "name": "hawTestCT",
        "clientToken": "6f36d650-a086-47cf-810a-fbfab2f8ad33"
    },
    "responseElements": {
        "application": {
            "applicationArn": "arn:aws:servicecatalog:us-east-1:12345789012:application/
app-02ocuq2cie2328pv64ya78e22f",
            "applicationId": "app-02ocuq2cie2328pv64ya78e22f",
            "creationTime": 1600895277.775,
            "lastUpdateTime": 1600895277.775,
            "name": "hawTestCT",
            "tags": {}
```

```
        }
    },
    "requestID": "1b6ad353-3b06-421b-bcb4-00075a782762",
    "eventID": "0a2ca224-cdfd-4c4b-a4ed-163218ff5e2d",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "12345789012"
}
```

Using AppRegistry

Use AWS Service Catalog AppRegistry you to create a repository of your applications and associated resources. You can then define and manage your application metadata to understand the context of your applications and resources across your environments.

An AppRegistry application consists of associated resources and attribute groups. An application resource can be either an AWS Service Catalog provisioned products or an AWS CloudFormation stack. You can add or remove resources from your application at any time. You can associate a resource with only one application.

You can also associate new or existing attribute groups to your application. Attribute groups contain the metadata for your application. When you update an attribute group definition, the update applies to every application associated with that attribute group.

You can use tags to assign metadata to your AppRegistry application. Tags can help you manage, identify, search for, and manage your applications.

For every application, AppRegistry creates an application resource group. An application resource group is a collection of the resources in your application. It also creates a stack level resource group for every stack associated with the application.

Topics

- Creating applications (p. 82)
- Associating application resources (p. 84)
- Associating attribute groups (p. 85)
- Adding tags (p. 88)

Creating applications

AppRegistry applications enable you to store your applications and associated resources, and then define and manage your application's metadata. This section describes how to create, edit, and delete AppRegistry applications. It also shows how to access information about your application’s resources, attribute groups, and tags, and associate them to your application.

To create an AppRegistry application

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, chose **Create application**.
3. In the **Create an application** section, enter a name for your AppRegistry application and a description. Your application name must be unique within your account and AWS Region.
4. If you want to:

   - Create your application, choose **Finish**. You can add resources and attributes groups later.
   - Add resources, attributes groups, and tags now, choose **Next**.

     To add resources to your application, see Associating application resources (p. 84).

     To add group attributes, see Associating attribute groups (p. 85).

Topics

- Accessing application details (p. 83)

- Editing applications (p. 83)
- Deleting applications (p. 84)

Accessing application details

You can obtain information about your AppRegistry application from application details. In the application's details, you can see:

- Name
- ID
- Amazon Resource Name (ARN)
- Creation date
- Description
- Resources, attribute groups, and tags

Resources

You can associate additional resources with the application and view the current associated resources. This view lists the resource and its ARN. You can also search for and disassociate resources.

For more information, see Associating application resources (p. 84).

Attribute groups

You can associate attribute groups with the application and view the current associated attribute groups. This view lists the attribute group, its description, and the time of the last update.

For more information, see Associating attribute groups (p. 85).

Tags

You can add tags to the application. This view shows all the tags associated it. You can also search for and delete tags.

For more information, see Adding tags (p. 88).

Editing applications

You can change the name and description of an AppRegistry application.

To edit applications from Applications

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, choose an application.
3. Choose **Actions**, **Edit**.
4. In **Edit application name and description**, change the name and description, and choose **Save changes**.

To edit an application from Applications details

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, open an application to display **Application details**. Then choose **Edit**.
3. In **Edit application name and description**, edit the name and description, and choose **Save changes**.

Deleting applications

You can delete AppRegistry applications from Applications and Application details.

To delete applications from Applications

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, choose an application.
3. Choose **Actions**, **Delete**.
4. Confirm your deletion and choose **Delete application**.

To delete applications from Applications details

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, open an application to display application details.
3. Choose **Delete**.
4. Confirm your deletion and choose **Delete application**.

Associating application resources

AppRegistry enables you to add provisioned products and CloudFormation stacks as resources to associate with your application.

You can only associate a provisioned product and CloudFormation stack with one application. You can add or remove resources from your application at any time.

To associate application resources from Applications

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, choose **Create application**.
3. In **Create an application**, enter a unique name for your AppRegistry application and a brief description. If you want to only create an application without resources, attribute groups, or tags, choose **Finish**. To add resources, choose **Next**.
4. In **Resources**, add one or more provisioned products as resources to associate to this application. You can also enter the ARN of the CloudFormation stack to associate as a resource to this application. To add more ARNs, choose **Add another ARN**.
5. If you want to only associate application resources, choose **Finish**. To associate attribute groups and tags, choose **Next.**

To associate application resources from Application details

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, open an application to display **Application details** and choose **Resources**.
3. Choose **Associate resource**.
4. In **Resources**, add one or more provisioned products as resources to associate to this application. You can also enter the ARN of the CloudFormation stack to associate as a resource to this application. To add more ARNs, choose **Add another ARN**. To add the resources to the application, choose **Submit**.

To disassociate application resources from Application details

You can stop the association of a resource with your AppRegistry application.

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, open an application to display **Application details** and choose **Resources**.
3. Choose **Disassociate resource**. Choose **Ok** to complete the process.

Associating attribute groups

You can associate new or existing attribute groups to your application. An attribute group is an open JSON object where you define the metadata for a resource. When you update an attribute group definition, the update applies to every application associated with that attribute group.

You can associate a new or existing attribute groups from Attribute groups or when you create an AppRegistry application.

Topics

- Creating attribute groups (p. 85)
- Accessing attribute group details (p. 85)
- Associating attribute groups (p. 86)

Creating attribute groups

To create attribute groups

1. In the left navigation menu, choose **AppRegistry**, **Attribute groups**.
2. In **Attribute groups**, choose **Create attribute group**.
3. In **New attribute group**, enter a name and description. Then add an open JSON object schema that captures metadata to filter, sort, and search your applications in AppRegistry.

   Here is an example of an open JSON object schema:

   ```
   {
   "ApplicationResilience":"high",
   "DataSecurity":"high",
   "DataSensitivity":"high"
   }
   ```

4. Choose **Next** to assign this attribute groups to an existing application and add tags.

   To assign attribute groups to an application, choose one or more applications.

   If you want to only assign attribute groups to applications, choose **Finish**.

   To add tags to the attribute group, choose **Next**.

Accessing attribute group details

In Attribute group details, you can see the attribute group's:

- Name
- Description
- ID

- ARN
- Metadata as a JSON open object

You can also see your existing tags and create additional tags. Tags with an *aws* prefix are internal tags. AppRegistry automatically adds them, and you can't remove them.

On the Attribute group details page, you can delete and edit an attribute group. You can access Attribute group details from Applications or Attribute groups.

To access Attribute group details from Applications

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. Open an application to show the **Application details**, and choose **Attribute groups**.

To access Attribute group details from Attribute groups

1. In the left navigation menu, choose **AppRegistry**, then **Attribute groups**.
2. Open an attribute group to show the **Attribute group details**.

Associating attribute groups

You can associate attribute groups from Applications or Application details.

To associate attribute groups from Applications

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, choose **Create application**.
3. In **Create an application**, enter a name and description and choose **Next** to add resources. Choose **Finish** if you do not want to add resources, associate attribute groups, or add tags now.
4. In **Resources** add one or more provisioned products or CloudFormation stack ARNs as resources. Choose **Next** to associate to attribute groups, or choose **Finish** if you do not want to add tags now.
5. You can associate your application to existing attribute groups or create new attribute groups.

   In **Associate existing attribute groups**, choose one or more attribute groups from your attribute library.

   In **New attribute group**, enter a name and description. Then add an open JSON object schema to gather metadata to manage your application in AppRegistry.

   Here is an example of an open JSON object schema:

   ```
   {
      "ApplicationResilience":"high",
      "DataSecurity":"high",
      "DataSensitivity":"high"
   }
   ```

6. If you want to only create or add existing attribute groups to your application, choose **Finish**. To add tags to your application, choose **Next**.

To associate new attribute groups from Attributes groups

1. In the left navigation menu, choose **AppRegistry**, **Attribute groups**.

2. In **Attribute group**, chose **Create attribute group**.
3. In **New attribute group**, enter a name and description.
4. Add an open JSON object schema that gathers the metadata you can use to filter, sort, and search your applications in AppRegistry. Here is an example of the open JSON object schema:

```
{
   "ApplicationResilience":"high",
   "DataSecurity":"high",
   "DataSensitivity":"high"
}
```

5. To associate the attribute group to one or more existing applications, choose **Next**.
6. Associate the attribute group to existing applications. If you want to only associate applications, choose **Finish**. The Attribute group details appear with information about your new attribute group.
7. If you want to add tags, choose **Next**.

To associate or disassociate attribute groups from Application details

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, open an application to display **Application details**.
3. Choose **Attribute groups**.
4. To associate existing attribute groups, choose **Associate attribute group**.
5. Select one or more attribute groups from your attribute library and choose **Save changes**.

   To disassociate existing attribute groups, select an attribute group and choose **Diassociate**. Confirm your deletion and choose **Ok**.

To delete attribute groups from Attribute groups

1. In the left navigation menu, choose **AppRegistry**, **Attribute groups**.
2. Choose an attribute group. Then choose **Actions**, **Delete attribute group**.
3. In **Delete attribute group**, confirm your deletion and choose **Delete attribute group**.

To delete attribute groups from Attribute group details

1. In the left navigation menu, choose **AppRegistry**, **Attribute groups**.
2. Open an attribute group to display **Attribute group details**.
3. Choose **Delete**. Confirm your deletion and choose **Delete attribute group**,

To edit attribute groups from Attribute group details

1. In the left navigation menu, choose **AppRegistry**, **Attribute groups**.
2. Open an attribute group to display **Attribute group details**.
3. Choose **Edit**. In Edit attribute group, enter your changes and choose **Save changes**.

To edit attribute groups from Attribute groups

1. In the left navigation menu, choose **AppRegistry**, **Attribute groups**.
2. Choose an attribute group. Then choose **Actions**, **Edit attribute group**.
3. In **Edit attribute group**, enter your changes and choose **Save changes**.

Adding tags

You can use tags to assign metadata to your AppRegistry applications and attribute groups. You can create a maximum of 50 tags on an application or attribute group to categorize them by purpose, owner, environment, or other criteria.

You can access tags from Applications, Application details, Attribute groups, and Attribute key details.

Tags in AppRegistry with an *aws* prefix indicate internal tags. AppRegistry automatically adds them, and you can't remove them.

To add tags from Applications

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, choose **Create application**.
3. In **Create an application**, enter a unique name for your AppRegistry application and a brief description. If you only want to:
   - Create an application without resources, attribute groups, or tags, choose **Finish**.
   - Add tags and associate resources, choose **Next**. Associate attribute groups and choose **Next** to add tags.
4. Choose **Add a tag** and enter information in the **Key** and **Value** fields. If you want to add more tags, choose **Add another**. Then choose **Finish**.

To add and delete tags from Applications details

1. In the left navigation menu, choose **AppRegistry**, **Applications**.
2. In **Applications**, open an application to display **Application details** and choose **Tags**. You can add, search, or delete tags.
   - To add tags, enter information in the **Key** and **Value** fields, and choose **Add tag**.
   - To delete tags, choose a tag in the **Application specific** tags list. Choose **Delete tag**, confirm your deletion, and then choose **Ok**.

To add tags to a new Attribute group

1. In the left navigation menu, choose **AppRegistry**, **Attribute groups**.
2. In **Attribute groups**, choose **Create attribute group**.
3. In **New attribute group**, enter a name and description. Then add an open JSON object schema to capture metadata to manage your applications.
4. In **Assign attribute group to an application**, choose **Next** to skip this step if you don't want to add tags. If you want to add tags, choose to associate existing applications to this attribute group and proceed to add tags.
5. To add tags, choose **Add tag**. Enter data in the **Key** and **Value** field. To add more tags, choose **Add another**. To remove tags, choose **Remove**. To complete the process, choose **Finish**.

To add tags from Attribute group details

1. In the left navigation menu, choose **AppRegistry**, **Attribute groups**.
2. Open an attribute group to display **Attribute group details**. Then choose **Tags**. You can add, search, or delete tags.
   - To add tags, enter information in the **Key** and **Value** fields, and choose **Add tag**.

- To delete tags, choose a tag in the **Application specific tags** list. Choose **Delete tag**, confirm your deletion, and choose **Ok**.

Product and Service Integrations with AWS Service Catalog

AWS Service Catalog is integrated with a number of AWS services and partner products and services. Use the information in the following sections to help you configure AWS Service Catalog to integrate with the products and services you use.

Topics

- AWS Service Management Connector for ServiceNow (p. 90)
- AWS Service Management Connector for Jira Service Management (p. 128)

AWS Service Management Connector for ServiceNow

The AWS Service Management Connector for ServiceNow (formerly the AWS Service Catalog Connector) enables ServiceNow end users to provision, manage, and operate AWS resources natively through ServiceNow.

ServiceNow administrators can:

- Provide pre-approved, secured, and governed AWS resources to end users through AWS Service Catalog.
- Execute automation playbooks through AWS Systems Manager.
- View and manage operational items as incidents through AWS Systems Manager OpsCenter
- Track resources in the CMDB, powered by AWS Config, seamlessly on ServiceNow with the AWS Service Management Connector.
- Define new resource types based on ServiceNow CMDB tables and synchronize these with AWS Config custom resources.
- Configure syncing AWS Security Hub findings to ServiceNow incidents or problems.

ServiceNow end users can:

- Browse, request, and provision pre-secured AWS solutions.
- View, update and resolve incidents from AWS Systems Manager OpsItems
- View configuration item details.
- Execute workflows in ServiceNow on AWS resources.
- View, update, and resolve ServiceNow incidents or problems through AWS Security Hub findings.

These features simplify AWS product request actions for ServiceNow users, and provide ServiceNow governance and oversight over AWS products.

The AWS-supplied connector is available at no charge in the ServiceNow store. It supports ServiceNow platform releases Quebec (Q), Paris (P) and Orlando (O). These new features are generally available in all AWS Regions where AWS Service Catalog, AWS Config, and AWS Systems Manager services are available.

Topics

- Background (p. 91)
- Getting started (p. 91)
- Release notes (p. 92)
- Baseline permissions (p. 93)
- Configuring AWS Service Catalog (p. 98)
- Configuring AWS Config (p. 99)
- Configuring AWS Security Hub (p. 99)
- Configuring AWS Systems Manager OpsCenter (p. 100)
- Configuring ServiceNow (p. 100)
- Configuring AWS Config integration in ServiceNow (p. 111)
- Configuring AWS Systems Manager OpsCenter integration in ServiceNow (p. 117)
- Validating configurations (p. 119)
- ServiceNow additional features (p. 124)
- Version 2.3.4 release transition instructions (p. 127)

Background

AWS Service Catalog allows you to centrally manage commonly deployed AWS services and provisioned software products. It helps your organization achieve consistent governance and compliance requirements, while enabling users to quickly deploy only the approved AWS services they need.

AWS Config enables you to assess, audit, and evaluate the configurations of your AWS resources. AWS Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations.

AWS Systems Manager gives you visibility and control of your infrastructure on AWS. Systems Manager provides a unified user interface so you can view operational data from multiple AWS services, investigate and resolve operational issues through the OpsCenter, and automate operational tasks across your AWS resources.

AWS Security Hub gives you a comprehensive view of your security alerts and security posture across your AWS accounts. With Security Hub, there is a single place that aggregates, organizes, and prioritizes your security alerts, or findings.

ServiceNow is an enterprise service management platform that places a service-oriented lens on the activities, tasks, and processes that enables day-to-day work life and a modern work environment. ServiceNow Service Catalog is a self-service application that end users can use to order IT services based on request fulfillment approvals and workflows. The ServiceNow CMDB provides resource transparency and relationships for the logical components of a service.

Getting started

Before installing the AWS Service Management Connector for ServiceNow, verify that you have the necessary permissions in your AWS account and ServiceNow instance.

AWS prerequisites

To start, use the following services:

- AWS Service Catalog with the Connector

  You need an AWS account to configure your AWS portfolios and products. For details, see Setting up for AWS Service Catalog.

- AWS Config details

  Configure the service settings to record data for the resource types of interest. We recommend you include provisioned products and AWS CloudFormation stacks, in addition to the major resource types that your team uses. For more information, see Setting up AWS Config with the console. This version of the Connector enables the import of aggregated Config data in a single AWS account from more than one AWS Region or account. To use this feature, you must configure an aggregator in AWS. For more information, see Setting up an Aggregator using the console.
- AWS Systems Manager Automation with the Connector

  This feature requires no AWS-side set up. As standard, AWS provides a number of automation documents (runbooks). If you want additional automation documents (runbook), retrieve them in the Connector. For more information, see Working with Automation Runbooks.
- AWS Systems Manager OpsCenter with the Connector

  You must enable the service in all Regions and accounts where you want to sync OpsItems. For more information, see Getting started with OpsCenter
- AWS Security Hub with the Connector

  You must enable the service in all Regions and accounts where you want to sync Findings. For details see Setting up Security Hub. We recommend you connect ServiceNow with the primary (master) AWS account for AWS Security Hub. For more information, see Managing master and member accounts.

ServiceNow prerequisites

In addition to the AWS account, you need a ServiceNow instance to install the ServiceNow Connector scoped application. The initial installation should occur in either an enterprise sandbox or a ServiceNow Personal Developer Instance (PDI), depending on your organization's technology governance requirements.

The ServiceNow administrator needs the admin role to install the Connector for ServiceNow scoped application.

Release notes

Version 3.7.1 of the AWS Service Management Connector for ServiceNow (formerly the AWS Service Catalog Connector) includes:

AWS ServiceNow Connector core features	AWS Systems Manager OpsCenter integration features	AWS Config integration features
Enables synchronization of key(s) rotated for AWS account credentials opted into the Connector.	Views operational item (OpsItem) from AWS Systems Manager OpsCenter in ServiceNow.	Optimizes memory utilization during sync.
Optimizes AWS API calls from the ServiceNow Connector scoped app.	Creates ServiceNow incident(s) from AWS OpsItem(s).	Identifies and sets install status of stale records when synchronizing with Config Aggregators.
Supports multiple AWS accounts.	Updates and resolves correlated OpsItem(s) and incident(s) in AWS and ServiceNow respectively.	Identifies AWS resources details, such as the Account, Region, ResourceType, and ResourceId (default ServiceNow field Correlation ID).

AWS ServiceNow Connector core features	AWS Systems Manager OpsCenter integration features	AWS Config integration features
Supports FIPS endpoints and usage in the AWS GovCloud West and East Regions.	Views related resource details associated in AWS Systems Manager OpsCenter in ServiceNow	
Supports the latest ServiceNow platform releases for Quebec (Q), Paris (P), and Orlando (O).	Views and executes automation documents (runbook) to resolve OpsItems in the ServiceNow incident and view execution results through ServiceNow Service Catalog.	

This version also includes prior AWS Service Management Connector for ServiceNow feature integrations to AWS services, such as AWS Security Hub, AWS Config, and AWS Systems Manager Automation.

Baseline permissions

This section provides instructions on how to set up baseline AWS users and permissions for the AWS Service Management Connector for ServiceNow.

Available template for baseline permissions

To use an AWS CloudFormation template to set up the AWS configurations of the Connector for ServiceNow, see the AWS configurations for Connector for ServiceNow 3.7.1 - AWS Commercial Regions and AWS GovCloud Regions.

> **Note**
> If you use the Connector for ServiceNow v3.7.1_AWS Configuration template, skip to Configuring AWS Service Catalog (p. 98).

For each AWS account, the Connector for ServiceNow requires two IAM users:

- **AWS Sync User**: An IAM user to sync AWS resources (such as portfolios, products, automation documents (runbook), configuration items, and security findings) to ServiceNow.
- **AWS End User role**: An IAM user who can provision products as an end user, execute requests, and view resources that ServiceNow exposes. This role includes any required roles to provision and execute.

Creating AWS Service Management Connector Sync user

The following section describes how to create the AWS Sync user and associate the appropriate IAM permission. To perform this task, you need IAM permissions to create new users.

To create AWS Service Management Connector sync user

1. Follow the instructions in Creating an IAM user in your AWS account to create a sync user (SMSyncUser). The user needs programmatic and AWS Management Console access to follow the Connector for ServiceNow installation instructions.
2. Set permissions for your sync user (SMSyncUser). Choose **Attach existing policies directly** and select:
   - **AWSServiceCatalogAdminReadOnlyAccess** (AWS managed policy)
   - **AmazonSSMReadOnlyAccess** (AWS managed policy)

- **AWSConfigUserAccess** (AWS managed policy)

3. Create this policy: `ConfigBidirectionalSecurityHubSQSBaseline`. Then follow the instructions in Creating IAM Policies, and add this code in the JSON editor:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "cloudformation:RegisterType",
                "cloudformation:DescribeTypeRegistration",
                "cloudformation:DeregisterType",
          "sqs:ReceiveMessage",
           "sqs:DeleteMessage",
                          "sqs:DeleteMessageBatch",
            "config:PutResourceConfig",
            "securityhub:BatchUpdateFindings"
            ],
            "Resource": "*"
        }
    ]
}
```

The provided AWS Configuration template consists of two policies: `ConfigBiDirectionalPolicy` and `SecurityHubPolicy`.

4. Create this policy: `OpsCenterExecutionPolicy` Then follow the instructions in Creating IAM Policies, and add this code in the JSON editor:

```
 {
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "ssm:GetOpsItem",
               "ssm:UpdateOpsItem",
               "ssm:DescribeOpsItems"
            ],
           "Resource": "*"
       }
```

```
    ]
}
```

5. Add a policy that allows `budgets:ViewBudget` on all resources (*).
6. Review and choose **Create User**.
7. Note the access and secret access information. Download the .csv file that contains the user credential information.

Creating AWS Service Catalog end user

This section describes how to create the AWS Service Management Connector end user and associates the appropriate IAM permission. To perform this task, you need IAM permissions to create new users.

To create AWS Service Management Connector end user

1. Follow the instructions in Creating an IAM user in your AWS account to create a user (SMEndUser). The user needs programmatic and AWS Management Console access to follow the Connector for ServiceNow installation instructions.

   For products using AWS CloudFormation StackSets, you need to create a StackSet inline policy. With AWS CloudFormation StackSets, you are able to create products across multiple accounts and Regions.

   Using an administrator account, you define and manage an AWS Service Catalog product. You also use it to provision stacks into selected target accounts across specified Regions. You need to have the necessary permissions defined in your AWS accounts.

   To set up the necessary permissions, see Granting Permissions for Stack Set Operations. Follow the instructions to create an **AWSCloudFormationStackSetAdministrationRole** and an **AWSCloudFormationStackSetExecutionRole**.

   Create the StackSet inline policy to enable provisioning a product across multiple Regions within one account.

```
    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": [
                    "sts:AssumeRole"
                ],
                "Resource": [
                "arn:aws:iam::123456789123:role/AWSCloudFormationStackSetExecutionRole"
                ],
                "Effect": "Allow"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "iam:GetRole",
                    "iam:PassRole"
                ],
    "Resource":       "arn:aws:iam::123456789123:role/
AWSCloudFormationStackSetAdministrationRole"
            }
        ]
    }
```

Note
Replace **123456789123** with your account information. The Connector for ServiceNow v3.7.1 - AWS Commercial Regions and Connector for ServiceNow v3.7.1 - AWS GovCloud Regions files include the stack set permissions.

2. Add the following permissions (policies) to the user:

   - **AWSServiceCatalogEndUserFullAccess** (AWS managed policy)
   - **StackSet (inline policy)** - For AWS Service Catalog products with stack sets, you need to modify the SMEndUser to include the Read Only permissions for the services you want to provision. For example, to provision an Amazon S3 bucket, include the **AmazonS3ReadOnlyAccess** policy to the SMEndUser.
   - **OpsCenterExecutionPolicy**
   - **AmazonEC2ReadOnlyAccess** (AWS managed policy)
   - **AmazonS3ReadOnlyAccess** (AWS managed policy)

Creating SCConnectLaunch role

The SCConnect Launch role is an IAM role that places baseline AWS service permissions into the AWS Service Catalog launch constraints. Configuring this role enables segregation of duty through provisioning product resources for ServiceNow end users.

The SCConnectLaunch role baseline contains permissions to Amazon EC2 and Amazon S3 services. If your products contain more AWS services, you must either include those services in the SCConnectLaunch role or create new launch roles.

This section describes how to create the **SCConnectLaunch** role. This role places baseline AWS service permissions in the AWS Service Catalog launch constraints. For more information, see AWS Service Catalog Launch Constraints.

To create SCConnectLaunch role

1. Create this policy: `AWSCloudFormationFullAccess` policy. Choose **create policy** and add this code in the JSON editor:

```
                {
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
        "cloudformation:DescribeStackResource",
        "cloudformation:DescribeStackResources",
        "cloudformation:GetTemplate",
        "cloudformation:List*",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStacks",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStacks",
        "cloudformation:GetTemplateSummary",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DescribeChangeSet",
```

```
                "cloudformation:ExecuteChangeSet",
                "cloudformation:DeleteChangeSet",
                "s3:GetObject"
                ],
                "Resource": "*"
            }
        ]
    }
```

Note
AWSCloudFormationFullAccess includes additional permissions for ChangeSets.

2. Create this policy: `ServicecodeCatalogSSMActionsBaseline`. Follow the instructions in Creating IAM Policies, add this code in the JSON editor:

```
                    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1536341175150",
            "Action": [
                "servicecatalog:ListServiceActionsForProvisioningArtifact",
                "servicecatalog:ExecuteprovisionedProductServiceAction",
                "ssm:DescribeDocument",
                "ssm:GetAutomationExecution",
                "ssm:StartAutomationExecution",
                "ssm:StopAutomationExecution",
                "cloudformation:ListStackResources",
                "ec2:DescribeInstanceStatus",
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

3. Create the **SCConnectLaunch** role. Then assign the trust relationship to AWS Service Catalog.

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

4. Attach the relevant policies to the **SCConnectLaunch** role.

   Attach these baseline IAM policies:

   - **AmazonEC2FullAccess** (AWS managed policy)

- **AmazonS3FullAccess** (AWS managed policy)
- **AWSCloudFormationFullAccess** (custom managed policy)
- **ServiceCatalogSSMActionsBaseline** (custom managed policy)

Configuring AWS Service Catalog

After you create two IAM users with baseline permissions in each account, the next step is to configure AWS Service Catalog. This section describes how to configure AWS Service Catalog to have a portfolio with an Amazon S3 bucket product. Use the Amazon S3 template in Creating an Amazon S3 Bucket for Website Hosting for your preliminary product. Copy and save the Amazon S3 template to your device.

To configure AWS Service Catalog

1. Follow the steps in Create an AWS Service Catalog Portfolio (p. 12) to create a portfolio.
2. To add the Amazon S3 bucket product to the portfolio you created in Step 1, go to the AWS Service Catalog console. In the **Upload new product** page, enter the product details.
3. For Select template, choose the Amazon S3 bucket AWS CloudFormation template you saved to your device.
4. Set **Constraint type** to **Launch** for the product that you created now with the SCConnectLaunch role in the baseline permissions. For additional launch constraint instructions, see AWS Service Catalog Launch Constraints (p. 46).

   Note
   The AWS configuration design requires each AWS Service Catalog product to have a launch constraint. Failure to follow this step could result in an “Unable to Retrieve Parameter” message in the ServiceNow Service Catalog.
5. Add the SnowEndUser IAM role to the AWS Service Catalog portfolio. For additional user access instructions, see Granting Access to Users (p. 37).

Note
The AWS configuration design requires each AWS Service Catalog product to have either a launch constraint or a stack set constraint. Failure to follow this step could result in an “Unable to Retrieve Parameter” error in the ServiceNow Service Catalog.

Creating Stack Set constraints

AWS CloudFormation StackSets enable users to create and deploy products across multiple accounts and Regions.

To apply a stack set constraint to an AWS Service Catalog product

1. Go to AWS Service Catalog as a catalog admin.
2. Choose the portfolio that contains the product.
3. Expand **Contraints** and choose **Add constraints**.
4. Choose the product from **Product** and set **Constraint type** to **Stack Set**. Choose **Continue**.
5. On the Stack Set constraint page, enter a description.
6. Choose the account(s) in which you want to create products.
7. Choose the Region(s) in which you want to deploy products. Products deploy in these Regions in the order you specify.
8. Choose **AWSCloudFormationStackSetAdministrationRole** to manage your target accounts.
9. Choose **AWSCloudFormationStackSetExecutionRole** for the role the Administrator will assume.
10. Choose **Submit**.

Note that the Connector for ServiceNow v3.7.1 - AWS Commercial Regions and Connector for ServiceNow v3.7.1 - AWS GovCloud Regions templates create the permissions as well as outputs for stack set constraints.

Example stack set outputs:

```
            SCStackSetAdministratorRoleARN
            arn:aws:iam::123456789123:role/AWSCloudFormationStackSetAdministrationRole
SCIAMStackSetExecutionRoleName
            AWSCloudFormationStackSetExecutionRole
            SCIAMAdminRoleARN
            arn:aws:iam::123456789123:role/AWSCloudFormationStackSetAdministrationRole
```

Note
Replace the **123456789123** with your account information.

Relating budgets to products and portfolios

The Connector for ServiceNow enables ServiceNow administrators to view budgets related to AWS Service Catalog products and portfolios. AWS Service Catalog administrators can create or associate existing budgets to products and portfolios.

For more information on creating and associating budgets, see the section called “Managing Budgets” (p. 62).

Configuring AWS Config

AWS Config integration to ServiceNow CMDB

To allow the connector to synchronize Config data for a given Region, you must enable AWS Config in that Region. For more information, see Setting Up AWS Config with the Console.

The Connector can now synchronize Config data from multiple accounts and Regions using an Aggregator. You must configure the Config Aggregator in AWS before using this feature. For more information, see Setting up an Aggregator in the console.

Note
The Config Aggregator view in AWS displays only current config item resources in AWS Config. Thus, terminated resources are not available in the Config Aggregator view.
To minimize stale config item records rendering in the ServiceNow CMDB from the AWS Config Aggregator, we recommend you remove Config rules associated to terminated resources. For more information, see Managing your AWS Config Rules

Configuring ServiceNow tables as AWS Config custom resources

Version 3.7.1 of the Connector for ServiceNow enables ServiceNow administrators to specify select ServiceNow tables as custom resources within AWS Config.

To set up these resources, use the preconfigured files in the Connector. These required files include the custom resource schema.

Configuring AWS Security Hub

AWS Security Hub enables users to view security findings from AWS services such as Amazon Guard Duty, Amazon Inspector, as well as AWS Partner solutions.

View the following video, *AWS Security Hub - Bidirectional integration with ServiceNow ITSM*, for an overview of the AWS Security Hub integration to the Connector for ServiceNow.

Share (https://www.youtube.com/embed/OYTi0sjEggEShare) AWS Security Hub - Bidirectional integration with ServiceNow ITSM

To configure AWS Security Hub integration features

1. Enable AWS SecurityHub. For more information, see Setting up AWS SecurityHub with the Console.
2. Set up an SQS queue to receive updated Findings. Name the queue `AwsServiceManagementConnectorForSecurityHubQueue` to align with the default name in the ServiceNow System Properties for the AWS Security Hub integration. For more information, see Getting started with Amazon SQS.
3. Set up a CloudWatch rule to detect changes to Findings and push these to the queue. For more information, see Getting started with Amazon CloudWatch.
4. The CloudWatch rule should have this event pattern and point to the SQS queue created in Step 2.

```
"EventPattern": {

       "source": [

        "aws.securityhub"

        ],

        "detail-type": [

        "Security Hub Findings - Imported",

        "Security Hub Findings - Custom Action"

  ]

}
```

Note that the Connector for ServiceNow v3.7.1 - AWS Commercial Regions and Connector for ServiceNow v3.7.1 - AWS GovCloud Regions] templates are available to automate the AWS Config custom resource and AWS Security Hub integration features.

Configuring AWS Systems Manager OpsCenter

To allow the Connector to synchronize AWS Systems Manager - OpsCenter data for a specific Region, you must enable OpsCenter in that account and Region. For more information, see Gettting Started with Ops Center.

Configuring ServiceNow

After completing the IAM and AWS Service Catalog configurations, the next configuration area to set up is ServiceNow. Installation tasks in ServiceNow include:

- Clear the ServiceNow platform cache.
- Clear the web browser cache.
- Activate two ServiceNow plugins.
- Install the ServiceNow Connector scoped application, and upload and commit the ServiceNow Connector Update Set.

- Configure ServiceNow platform system admin components.
- Configure AWS Service Management Connector scoped application, including accounts, scheduled jobs sync, and permissions.
- Validate connectivity to AWS Regions.
- Manually sync scheduled jobs.
- Configure the AWS Service Catalog Product Widget Components and Assignment Group for Closed Change Records.
- Grant access to AWS Service Catalog portfolios.
- Configure AWS Tags for provisioned products.
- Configure synchronization of AWS Config data using an Aggregator into ServiceNow CMDB.
- Configure available ServiceNow tables to sync with AWS Config.
- Configure AWS Security Hub integration.

Clearing the ServiceNow platform cache

Before installing the AWS Service Management scoped app, we recommend you clear the ServiceNow platform cache. To do so, enter this URL: `https://[InsertServiceNowInstanceNameHere]/cache.do`.

> **Note**
> Ensure that you install the update set in a non-production or sandbox environment. Consult a ServiceNow system administrator if you need approval to clear the ServiceNow platform cache.

Clearing the web browser acche

Clear the web browser cache to remove previously rendered product forms.

Activating User Criteria Scoped API and Discovery and Service Mappings Patterns plugins

To activate the User Criteria Scoped API plugin

1. From your ServiceNow dashboard, enter **`plugins`** into the navigation panel in the upper left.
2. When the **System Plugins** page populates, next to the **Name** dropdown, search for **Discovery**.
3. Choose **User Criteria Scoped API** and then choose **Activate**.

To activate the Discovery and Service Mappings Patterns plugin

1. From your ServiceNow dashboard, enter **`plugins`** into the navigation panel in the upper left.
2. When the **System Plugins** page populates, next to the **Name** dropdown, search for **`Discovery`**.
3. Choose **Discovery and Service Mapping Patterns** and then choose **Activate**.

> **Note**
> This plugin is free and aligns to the CMDB tables outside of ServiceNow's family release CMDB updates.

Installing ServiceNow Connector scoped application

The AWS Service Management Connector for ServiceNow release is a conventional ServiceNow scoped application through a ServiceNow Update Set.

ServiceNow update sets are code changes to the out-of-the-box platform and enable developers to move code across ServiceNow instance environments. The Connector for ServiceNow update set is available to download in the ServiceNow store.

We provide the code for Connector for ServiceNow version 3.7.1 for users who install the update set on a ServiceNow Personal Developer Instance (PDI).

You can apply the Connector for ServiceNow version 3.7.1 update set to a “Quebec”, "Paris", or “Orlando" platform release of ServiceNow.

If you do not already have a ServiceNow instance, start with the first step below. If you already have a ServiceNow instance, proceed to the instructions below on how to install the update set.

To obtain a ServiceNow instance

1. Open Obtaining a Personal Developer Instance.
2. Create ServiceNow developer program credentials.
3. Follow the instructions for requesting a ServiceNow instance.
4. Capture your instance details, including URL, administrative ID, and temporary password credentials.

To install the update set

1. From your ServiceNow dashboard, enter **`update sets`** into the navigation panel in the upper left.
2. Choose **Retrieved Update Sets** from the results.
3. Choose **Import Update Set from XML** and upload the release XML file.
4. Choose the **AWS Service Management Connector for ServiceNow** update set.
5. Choose **Preview Update Set**, which makes ServiceNow validate the connector update set.
6. Choose **Update**.
7. Choose **Commit Update Set** to apply the update set and create the application. This procedure should complete 100%.

Platform system administrator components

To enable the AWS Service Management Connector for ServiceNow scoped application named **AWS Service Management**, the system admin must create a discovery source, and configure specific platform tables, forms, and views.

Create a discovery source AWS Service Management Connector entry

To allow AWS to report discovered CIs into your CMDB, you must create a new discovery data source, AWS Service Management Connector. Perform the following steps:

1. Choose **System Definition**. Then select **Choice Lists**.
2. Choose **New**.
3. Create a new entry with these details:

   - **Table:** **`Configuration Item [cmdb_ci]`**
   - **Element:** **`discovery_source`**
   - **Label:** **`AWS Service Management Connector`**
   - **Value:** **`AWS Service Management Connector`**

   Note
   Make sure you are in Global mode in ServiceNow System Settings to modify System Definitions.

Enable permissions on ServiceNow Platform table (Catalog Item Category)

For AWS products to display under AWS portfolios as sub-categories in the ServiceNow Service Catalog, you need to modify the Application Access form for Catalog Item Category tables. This action is necessary because a ServiceNow scoped API is not available for the Catalog Item Category table.

1. Enter **`Tables`** in the Navigator and choose **System Definition**, then choose **Tables**.
2. In the list of tables, search for a table with label **Catalog Item Category** (or with the name `sc_cat_item_category`). The list of tables displays. Choose **Category** to view the form defining the table.
3. Choose the **Application Access** tab on the form and choose the **Can Create**, **Can Update**, and **Can Delete** checkboxes on the form. Then choose **Update**.

ServiceNow permissions for administrators of the Connector scoped app

The AWS Service Management scoped app has two ServiceNow roles that enable access to configure the application. This feature enables system admins to grant one or more user's privileges to administer the application, without having to open full sysadmin access to them. System admins can assign these roles to either individual users or to one administrator user.

To set up Connector application administrator privileges

1. Enter *Users* in the navigator and select **System Security – Users**.
2. Choose a user to grant one or both previous roles (such as admin). You can also create a user.
3. Choose **Edit** on the Roles tab of the form.
4. Filter the collection of roles by the prefix **`x_`**.
5. Choose one or both of the following and add them to the user: **x_126749_aws_sc_account_admin**, **x_126749_aws_sc_portfolio_manager**, **x_126749_ aws_sc.automation_manager**, **x_126749_aws_sc.finding_manager** and **x_126749_aws_sc.opscenter_manager**.
6. Choose **Save**.

To add AWS Service Catalog to ServiceNow Service Catalog categories

1. Choose **Self Service | Service Catalog** and select the **Add content** icon in the upper right.
2. Choose the **AWS Service Catalog Product** entry. To add it to your catalog home page, choose the first **Add Here** link on the second row of the selection panel at the bottom of the page.

To add AWS Systems Manager automation documents (runbook) to ServiceNow Service Catalog categories

1. Choose **Self Service | Service Catalog** and select the **Add content** icon in the upper right.
2. Select the **AWS Systems Manager** entry. To add it to your catalog home page, choose the first **Add Here** link on the second row of the selection panel at the bottom of the page.

> **Note**
> This Connector release displays all AWS Systems Manager documents in the AWS account that has AWS Systems Manager selected.

System administrators can deactivate AWS Systems Manager documents requests. To deactivate requests, choose **AWS Systems Manager**, **Automation Documents**, and deselect the **Active** button. After deactivation of the document, end users no longer see the document in the ServiceNow Service Catalog.

The Connector creates closed change requests on post provision actions (such as update, terminate and self-service) for AWS Service Catalog products visible in ServiceNow.

To achieve a closed change request from post provisioned actions, add a change request type and configure the `sys_id` for the group assigned to the closed change records in the Connector AWS Service Catalog system properties.

To add a change request type

1. If you are upgrading from a previous version of the AWS Service Management scoped app, you must remove the **AWS Product Termination** change request type before you create a new change request type.
2. You must add a new change request type called **AWS Provisioned Product Event** for the scoped application to trigger an automated change request in Change Management. For instructions, see Add a new change request type.
3. Open an existing change request.
4. Open (right-click) the context menu for **Type** and then choose **Show Choice List**.
5. Choose **New** and complete these fields:

   - **Table**: **`Change Request`**
   - **Label**: **`AWS Provisioned Product Event`**
   - **Value**: **`AWSProvisionedProductEvent`**
   - **Sequence**: pick the next unused value
6. Submit the form.

Note
For more information on how to associate the Change Assignment group, see *Configuring the AWS Service Catalog Product Widget Components* and *Assignment Group for Closed Change Records*.

Configuring AWS Service Management Connector scoped application

After installing and configuring the AWS Service Management Connector for ServiceNow, you must configure the scoped application and applicable roles.

To configure the AWS Service Management Connector scoped application permissions

1. In your ServiceNow instance, create a user group called **Order_AWS_Products**. Members of this group can order AWS Service Catalog products. For instructions, see Create a user group.
2. Grant ServiceNow permissions to these users:

   - **System Administrator (admin)**: For simplicity in this example, user **admin** is the administrator of the AWS Service Management scoped application. Grant this user both of the administrative permissions from the adapter: **x_126749_aws_sc_account_admin, x_126749_aws_sc_portfolio_manager**, **x_126749_ aws_sc.automation_manager**, **x_126749_aws_sc.finding_manager** and **x_126749_aws_sc.opscenter_manager**.

     Add System Administrator to the new ServiceNow group **Order_AWS_Products**. In a real scenario, these roles would likely be granted to different users or groups. In a real scenario, these roles would likely be granted to different users or groups.
   - **Abel Tuter**: The user **abel.tuter** is an illustrative end user. Grant Abel the new role **Order_AWS_Products**. This permission allows Abel to order products from AWS.

ServiceNow Permissions Recap

ServiceNow Persona	Scoped App Permissions	ServiceNow Permission Type
Admin	**x_126749_aws_sc_portfolio_manager**	Role (scoped app)
	x_126749_aws_sc_account_admin	Role (scoped app)
	x_126749_aws_sc.automation_manager	Role (scoped app)
	x_126749_aws_sc.finding_manager	Role (scoped app)
	x_126749_aws_sc.opscenter_manager	Role (scoped app)
End User (i.e., Abel Tuter)	**Order_AWS_Products**	Group

Configuring AWS accounts to synchronize in the Connector

1. Log in as the system administrator.
2. Enter **`aws`** in the navigator. Choose the **AWS Service Management** scoped app.
3. In the **AWS Service Management** scoped app Accounts menu, create one entry for every AWS account. You need to use the keys and secret keys from the users you created in AWS.

To create account entry

1. Enter the name as an account entry identifier, such as **Connector_Demo** (for Commercial Region), or **Connector_Demo_GovCloud** (for GovCloud Region).
2. Enter the AWS access key and secret access key from the AWS account sync user IAM configurations.
3. Enter the AWS access key and secret access key from the AWS account end user IAM configurations.
4. Select the AWS service integrations that you want visible for this AWS account. The choices include:
   - Integrate with AWS Service Catalog
   - Integrate with AWS Config
     - Select AWS Config if you plan to integrate AWS Config cloud resources per each AWS account or through the latest AWS Config Aggregator integration feature. Version 3.7.1 of the Connector for ServiceNow includes an AWS Config aggregator feature that enables ServiceNow administrators to align aggregated AWS Config details into one AWS account.
     - If you plan to use the Config Aggregator feature, proceed with configuring the AWS account in this section. For more information on the Config Aggregator steps, see *Configuring synchronization of AWS Config data using an Aggregator into ServiceNow CMDB*.
   - Integrate with AWS Systems Manager Automation.
   - Integrate with AWS Systems Manager OpsCenter.
   - Integrate with AWS Security Hub.

     Note
     For the AWS Systems Manager OpsCenter integration, you should select AWS Systems Manager Automation if you want to execute automation documents (runbook) to remediate incidents from OpsItems.
5. Choose **Account Regions**. Select the **Commercial** or **GovCloud Region**. To see the AWS account Regions, double-click **Insert a new row...**.
6. Repeat the step above to insert additional Regions.

7. Save or update the account entries.
8. Validate AWS account connectivity by following the steps in *Validating Connectivity to AWS Regions*. Note that in this Connector for ServiceNow version 3.7.1, the **Validate Accounts** button only appears once after you submit or update the account entry.

Validating connectivity to AWS Regions

You can now validate connectivity to AWS accounts between the ServiceNow **Connector_Demo** account and the AWS IAM SMSyncUser and SMEndUser.

To validate connectivity to AWS account

1. In the AWS Service Management scoped app, choose **Setup**, then **AWS Accounts**.
2. Choose **Connector_Demo** and select **Validate Account**.
3. A successful connection results in the message, “Successfully validating AWS account in each referenced Region.”

If the AWS IAM access key or secret access key are incorrect, you will receive an error message.

Manually syncing scheduled jobs

During the initial setup, manually execute the sync instead of waiting for **Scheduled Jobs** to run. The default sync schedule is every 31 minutes.

To sync the accounts manually

1. Log in as system administrator.
2. Find **Scheduled Jobs** in the navigator panel.
3. Search for job **Sync all Accounts**, select it, and choose **Execute Now**.

   Note
   If you do not see **Execute Now** in the upper left corner, choose **Configure Job Definition**. **Execute Now** will be visible.

Data is visible in the AWS Service Management scoped app menus after the adapter’s scheduled synchronization job has run.

Configuring the AWS Service Catalog product widget components and assignment group for closed change records

To address the varying personas of end users requesting AWS products, the Connector for ServiceNow includes a scoped app setting to enable or disable components of the AWS product widget. By default, all AWS product components are active.

To modify the AWS product view

1. In the navigator, enter **`System Properties`** and select **AWS Service Catalog**.

   Note
   Make sure you are in the AWS Service Management Connector scoped application mode.
2. Deselect any AWS product component such as:

   - Enable editing of the AWS Service Catalog product name.

- Enable selection of launch options for AWS Service Catalog Products. (This component is only visible if the AWS product has more than one launch path.)
- Enable selection of product versions for AWS Service Catalog. (This component is only visible if the AWS product has more than one product version.)
- Enable tags for AWS Service Catalog products.
- Enable plans (ChangeSet) creation for product. (If set to false the plan section is not visible.)

3. Choose **Save**.

The AWS Service Catalog system properties also include a section that identifies an assignment group. This group associates with closed change records from post provision actions of products (such as terminate, update, or self-service actions).

To associate the assignment group for change records from AWS Service Catalog post provision actions

1. In the navigator, enter **System Properties** and select **AWS Service Catalog**. Make sure you are in the AWS Service Management Connector scoped application mode.
2. Choose the section **Set the 'assignment group' sys_id or name that the connector will use when creating change requests**.
3. Enter the **Assignment group sys_id**.
4. If you need to find the `group sys_id`, enter **System Security** in the left navigator.
5. Select **Groups** module.
6. Search for the **Group** name.
7. Choose the group that you want to associate to close changed records and select **Copy sys_id**. You are now able to paste the copied `sys_id` into the AWS Service Catalog System Properties for the Connector under **Set the 'assignment group' sys_id or name that the connector will use when creating change requests**.

   If the `sys_id` is left blank, the change record sends a message that no assignment group exists for the record, which causes change requests created from the Connector to be in an open state.

Granting access to AWS Service Catalog portfolios

This release of the Connector does not require you to link AWS identities to ServiceNow roles. To grant access to AWS Service Catalog products in ServiceNow, you must establish a link between the AWS Service Catalog portfolios and the ServiceNow group (for example, **Order_AWS_Products** from an earlier installation example).

To grant access to AWS Service Catalog portfolios in ServiceNow

1. In the AWS Service Management scoped app, choose **AWS Service Catalog**, then **Portfolios** module.
2. Select the desired Portfolio ARN. You can double-click the AWS Service Catalog portfolio name.
3. Select the **Allowed Groups** tab.
4. Choose **New** and enter the **Group** named **Order_AWS_Products**.
5. Choose **Submit**.

Configuring AWS Tags for provisioned products

The AWS Service Management Connector v3.7.1 enables ServiceNow administrators to add tags (metadata) to provisioned products globally across the scoped app or granularly at the portfolio level. These tags are not visible to end users.

Three tag types are available in this release:

- Generic tags in which the administrator can enter the key and value.
- ServiceNow Request Item tags in which the admin can enter the syntax for key and value in the table below.
- ServiceNow table(s) values that end users can select as tags for provisioned AWS resources. This release now enables administrators to identify any ServiceNow tables, such as Cost center or Department, and makes values from that table selectable for end users.

  Note
  Generic tags (from administrators) and ServiceNow Request Item tags are not viewable by end users.

Key	Value
Requested Item Number	${REQUEST_NUMBER}
User	${USERNAME}
Requested for	${REQUESTED_FOR}
Opened by	${OPENED_BY}

To add generic AWS tags to AWS Service Catalog provisioned products in ServiceNow

1. In the AWS Service Management scoped app, choose **Setup**, then the **Automated Tags** module.
2. Choose **New**.
3. For Global tags, enter the Key and Value entries and choose **Submit**.
4. For Portfolio tags, deselect **Global check**. The **Portfolio** field appears.

   Select the AWS Service Catalog portfolio, enter the Key and Value entries, and choose **Submit**.

To add in-scope ServiceNow request item AWS tags to AWS Service Catalog provisioned products derived from ServiceNow

1. In the AWS Service Management scoped app, choose **Setup**, then the **Automated Tags** module.
2. Choose **New**.
3. For Global tags, enter the specific Key and Value entries for either User or Request Item Number, and choose **Submit**.
4. For Portfolio tags, deselect **Global check**. The Portfolio field appears. Select the AWS Service Catalog portfolio, enter the Key and Value entries, and choose **Submit**.

To add tags to AWS provisioned products from ServiceNow tables and fields that are selectable by end users

1. In the AWS Service Management scoped app, choose **Setup**, then the **Automated Tags** module.
2. Choose **New**.
3. Choose **Selectable by End User**.
4. Choose a table from the dropdown list **Table Name**.
5. Choose a field from the dropdown list **Table Field**.
6. For Global tags, enter the Key and Value entries and choose **Submit**.

7. For Portfolio tags, deselect **Global check**. The **Portfolio** field appears.

   Select the AWS Service Catalog portfolio, enter the Key and Value entries, and choose **Submit**.

   The ServiceNow Table and field value appear on the AWS Product (ServiceNow catalog item). It is a required value prior to ordering. After product provisioning, you can see in the AWS console that these tags associate with the resource.

Configuring AWS Security Hub integration

To configure AWS SecurityHub synchronization behavior to the Connector in ServiceNow

1. In the navigator, enter **AWS Service Management**.
2. Select **System Properties**, then **AWS Security Hub**.
3. Set these configuration items:

   - Select the types of AWS Security Hub Findings to sync in ServiceNow: CRITICAL, HIGH, MEDIUM, LOW, and INFORMATIONAL.
   - Select an action for a newly synced Finding to the Connector in ServiceNow:
     - **Do Nothing**. This action only imports Security Finding types for the scoped app. Users with scoped app permissions can view and choose to create incident or problem. **Do Nothing** is the default value in the Connector.
     - **Create Incident**. This action automatically creates incidents from Security Findings and syncs updates in ServiceNow to AWS Security Hub.
     - **Create Problem**. This action automatically creates incidents from Security Findings and syncs updates in ServiceNow to AWS Security Hub.
     - **Create Incident and Problem**. This action automatically creates incidents and problems from Security Findings and syncs updates in ServiceNow to AWS Security Hub.
   - Adjust the maximum number of messages to fetch from the SQS queue per sync, account, or Region (default 50). By default, the sync process runs every five minutes.
   - Change the SQS Queue name if you're not using the default that the Connector created. The CloudFormation template supplies the Connector.

     Note
     We recommend you not change the SQS name in the ServiceNow scoped app (`AwsServiceManagementConnectorForSecurityHubQueue`) unless you change the SQS name in the AWS account.
4. Select **Save** after any changes.

   Fields synchronized from AWS Security Hub Findings to the ServiceNow scoped app AWS Security Hub Findings module in ServiceNow

Region	The Region in which the Finding was generated.
Account Id	The account in which the Finding was generated.
Company Name	The company which generated the finding (e.g. AWS).
Compliance	Whether a resource passes the configured compliance criteria. Contains status (PASSED, WARNING, FAILED, NOT_AVAILABLE) and if the resource does not pass will contain some information regarding the reason for this.
Created At	The creation time of the Finding.

Description	A description of the Finding.
Criticality	The level of importance for the resource associated with the Finding.
First Observed At	First observation of when Findings captured any potential security issues.
Last Observed at	The most recent time Findings captured any potential security issues.
Product Name	The name of the product that generates the Finding (such as Security Hub).
Product Arn	The ARN of the product that generates the Finding.
Record State	Either ACTIVE or ARCHIVED.
Severity (normalized)	A value from 0 to 100 that indicates the severity of the problem associated with the Finding.
Status	PASSED, WARNING, FAILED, or NOT AVAILABLE.
Title	The title of the Finding.
Updated At	When the Finding provider last updated the record.
Workflow Status	The workflow status can be: NEW, ASSIGNED, IN PROGRESS, RESOLVED, DEFERRED, or DUPLICATE.
Remediation Text	A description of suggested action to resolve the discovered issue.
Remediation Url	A link to a resource that can resolve the discovered issue.

Adding the My AWS Products widget to the Service Portal view

We recommend ServiceNow administrators add the **My AWS Products** widget to the ServiceNow Portal view. The widget enables users to view their AWS product requests, view outputs, and perform post-operational actions such as update, terminate, and service actions (AWS Systems Manager documents).

To include the My AWS Products widget on the Service Portal view

1. Log in as system administrator in the ServiceNow standard user interface (Fulfiller view).
2. In the navigator panel, find **Service Portal**.
3. Choose **Service Portal Configuration**.
4. Choose **Designer**.
5. Search for **Service Portal** in the filter.
6. Select the **Service Portal** box with a house image and the word **Index** in the lower right corner.
7. In the left panel in **Widgets**, enter **`My AWS Products`** in the Filter Widget.
8. Drag the widget to the Service Portal edit view to your desired location.
9. Preview your changes.

Viewing budgets related to AWS Service Catalog portfolios and products

ServiceNow administrators can view budgets and actual costs related to AWS Service Catalog portfolios and products in the ServiceNow standard user interface.

To view portfolio budgets

1. Log in as system administrator.
2. In the navigator panel, search for **AWS Service Catalog**.
3. Select the **Portfolios** module.
4. Select the AWS Service Catalog portfolio that contains an associated budget.
5. Choose the **Budget** tab.

To view product budgets

1. Log in as system administrator.
2. In the navigator panel, search for **AWS Service Catalog**.
3. Select the **Products** module.
4. Select the AWS Service Catalog product that contains an associated budget.
5. Choose the **Budget** tab.

Syncing updated keys programmatically in ServiceNow

AWS Service Management Connector for ServiceNow allows synchronization of updated keys by any automation or integration, through a new REST endpoint.

You can send requests to sync updated keys for one or more AWS accounts registered in the AWS Service Management Connector, for either the sync or end user.

For more information about syncing updated keys syntax and instructions, see Syncing Updated Keys Programmatically in ServiceNow.

Configuring AWS Config integration in ServiceNow

This version of the Connector enables ServiceNow administrators to configure system properties, Config Aggregators, and AWS Config custom resources from select ServiceNow tables.

To configure the new AWS Config integration System Properties

1. In the navigator, enter **AWS Service Management**.
2. Select **System Properties**, then **AWS Config**.
3. Review the available settings and recommendations in the table below.

Available settings	Description
The name of the S3 bucket from where to get the resource provider ZIP files	The S3 bucket for custom resources from ServiceNow that populates AWS config. Default and hard coded value: `cmdb-resource-providers`

Available settings	Description
	Note We do not recommend changing this setting.
Name of the Discovery Source for synchronization with AWS Config	The setting that correlates the Discovery source in ServiceNow. Default and hard coded value: AWS Service Management Connector **Note** We do not recommend changing this setting.
What field to use for correlation ID	Administrators use this setting to specify which column contains the correlation ID for each AWS Config. The correlation ID disambiguates AWS Config item that might have the same resource ID (such as SQS queues). It consists of the comma separated string of: • Source account number • Source Region • Resource type, such as AWS::EC2::Instance • Resource ID Default: `correlation_id`
What field to use for AWS capture time	Administrators use this setting to specify which column contains the capture time (such as capture time from AWS Config) for each AWS Config item. Default: `last_discovered`
What field to use for last sync time	Administrators use this setting to specify which column contains the last sync time (such as the last time AWS Config integration performed a synchronization for a given item) for each AWS Config item. Default: `checked_in`

Available settings	Description
Enable the creation of a relationship for state sync	Administrators use this setting to enable the creation of a relationship to a special *state sync* configuration item. When enabled, each synchronized item links to a particular state sync, or execution. By enabling this feature, it allows the SMC to identify stale items. **Warning**: This action creates an additional relationship per synchronized item. Depending on the number of items, it might have a performance impact. Default: No
Enable the deletion of the previous relationship for state sync	Administrators use this setting to enable the deletion of previous relationships to a special *state sync* configuration item. When enabled, a successful synchronization to a given AWS Config time deletes the previous relationships to state sync configuration item. **Warning:** This action performs `GlideAggregate` queries for each group of synchronized accounts, Regions, or aggregators. Depending on the number of items, it might have a performance impact. Default: No
What "Install status" to put stale config item into	Administrators use this setting to automatically change the *install_status* of configuration items identified as stale. This action ensures that the status of stale resources correctly updates when using an aggregator. Be aware this feature works only if you set *What field to use for last sync time* and enable *Enable the creation of a relationship for state sync*. Allowed values: • Installed • Retired • Absent • Do nothing Default: Do nothing
Interval in minutes between the execution of full Config synchronization	Administrators use this setting to control the time between full syncs of Config data. The default is 720 minutes or 12 hours.

Addressing stale AWS Config items in the ServiceNow CMDB

In addition to the AWS Config settings, AWS SMC for ServiceNow now exposes a global API to identify stale config items from the AWS Config integration.

Note
This feature requires you to enable the creation relationship to sync the status setting in the AWS Config System Properties in the ServiceNow scoped app.

Stale Config items

Stale Config items are the existing AWS Config items that did not update during the most recent sync for the same source (such as account, Region and aggregator name).

Identifying stale Config items

Note
ServiceNow administrators are the target audience for this section

The script include, `x_126749_aws_sc.AwsSmc`, exposes a public API. You can use this script to access any application scope, including *global* scope. As an example, run this script:

```
   x_126749_aws_sc.AwsSmc.asSyncUser().getStaleConfigItems().forAll(function(object)
{
  gs.info(
       object.accountNumber + '/' + object.region + ' '
       + (object.aggregatorName ? 'aggregator: ' + object.aggregatorName + ' ' :
'')
       + 'ci: ' + object.ci.name
       + ' - ' + object.ci.getDisplayValue('install_status')
  );
});
```

As a background script, it would log the following:

```
Info: 11111111/us-east-1 ci: i-1234567fg6j8 - Installed
Info: 11111111/us-west-1 ci: i-9876541fdgfd - Installed
Info: 22222222/eu-west-1 aggregator: all-dev ci: i-1df5235ftt55 - Installed
```

Each *object* contains the below properties:

Property	Type	Description
accountNumber	String	The account number from which the stale config item originates.
region	String	The Region from which the stale config item originates.
aggregatorName	String	The aggregator name (if applicable) from which the stale config item originates.
lastSynced	`GlideDateTime`	The `GlideDateTime` of the when the last synchronization occurred.

Property	Type	Description
CI	`GlideRecord`	The `GlideRecord` of the stale config item.

Optionally, you can also pass an `options` object as the second argument to the `forAll` method which allows you to customize the search for stale items.

Property	Type	Description
`lowerTimeLimit`	`GlideDateTime`	The threshold `GlideDateTime` from when you should search items. Any stale item last updated prior to that date does not return.
`upperTimeLimit`	`GlideDateTime`	The threshold `GlideDateTime` untilyou should search for items. Any item last updated after that date does not return.
`excludeStatus`	Number	The `install_status` to filter on.

Timestamps of sync resources:

- `LastSyncTimeField`(default `checked_in`): The start of the current sync process.
- `first_discovered` (for new records): The current time. We set the LastDiscoveredField (default `last_discovered`) to the `configurationItemCaptureTime` of the resource, if it exists or undefined otherwise.

Additional notes on stale records

When AWS Service Management Connector reads AWS Config records that refer to other resources, it often creates a relationship to those resources.

In some cases, the related resource does not have an entry in the ServiceNow CMDB. In these cases, the Connector creates a record for that relationship, with an install status of *absent*. When the Connector reads the AWS Config record for the related resource, that record populates.

To see active resources, you should filter ServiceNow records synced from AWS Config by an install status of *not Absent*.

Disclaimer

Because the script compares items linked to state sync records, it is unable to identify stale resources synced before the installation of this SMC version. When switching to sync with an aggregator or switching from aggregator sync to non-aggregator sync, the script also fails to detect items that became stale between the last non-aggregator sync and the first aggregator sync.

Configuring synchronization of AWS Config data using an Aggregator in ServiceNow CMDB

Prerequisite: You need to opt-in and configure the AWS account that contains the aggregated AWS Config resources details prior to performing the steps below. For more information, see the *Configuring AWS Accounts to Synchronize in the Connector*.

To configure the Connector to use an Aggregator to synchronize AWS Config data

1. In the AWS Service Management scoped app, choose the **Setup** module.

2. Select **Aggregators for AWS Config**.
3. Choose **New**.
4. Enter the name of the new Config Aggregator.
5. Select the Region where you created the new Config Aggregator.
6. Choose the AWS account that should use the new Aggregator. Only AWS accounts opted into the Connector for ServiceNow that have **Integrate with AWS Config** are viewable.
7. Select **Submit**.

   If you define an Aggregator for an AWS account and Region, the Aggregator integration becomes the only AWS Config to ServiceNow CMDB synchronization mechanism for that AWS account.

Configuring available ServiceNow tables to sync as AWS Config custom resources

In this Connector for ServiceNow release, you can now sync a set of ServiceNow tables in the CMDB to AWS Config as custom resources.

The ServiceNow tables and AWS Config custom resource mapping are as follows:

ServiceNow CMDB table	AWS custom resource
`cmdb_ci_apache_web_server`	Apache Web Server
`cmdb_ci_app_server`	Application Server
`cmdb_ci_app_server_java`	Java Server
`cmdb_ci_app_server_tomcat`	Tomcat Server
`cmdb_ci_app_server_tomcat_war`	Tomcat Web Application
`cmdb_ci_app_server_websphere`	IBM Websphere Application
`cmdb_ci_app_server_ws_ear`	Websphere Enterprise Archive
`cmdb_ci_appl`	Application
`cmdb_ci_appl_dot_net`	A .Net Application
`cmdb_ci_appl_now_app_comp`	ServiceNow Application Component
`cmdb_ci_appl_sap`	Sap Application
`cmdb_ci_appl_sap_hana_db`	SAP Hana Database
`cmdb_ci_appl_sap_system`	SAP System
`cmdb_ci_appl_sharepoint`	Microsoft Sharepoint Application
`cmdb_ci_application_cluster`	Application Cluster
`cmdb_ci_application_server_resource`	Application Server Resource
`cmdb_ci_application_software`	Application Software
`cmdb_ci_db_mssql_database`	MySql Database
`cmdb_ci_db_mysql_instance`	MySql Instance

ServiceNow CMDB table	AWS custom resource
`cmdb_ci_kubernetes_cluster`	Kubernetes Cluster

To configure select ServiceNow tables as AWS Config custom resources

1. In the navigator, enter **AWS Service Management**.
2. Choose **Setup**, then **Tables Sync to AWS Config**.
3. Choose **New**.
4. Select an in scope ServiceNow table.
5. Select an account and Region for the new resource type. You can select any supported Region, in addition to preconfigured Regions for the account.
6. Click **Submit**.
7. Repeat steps above to include additional ServiceNow tables available to sync as AWS Config custom resources.

   The amount of time to create new AWS Config resources depends on the number of ServiceNow tables you selected. You can see resources in the **Schema version** field upon successful completion. The period synchronization of resources automatically includes the new AWS Config custom resource type. As details in the ServiceNow table update, this information syncs to AWS Config custom resource.

Configuring AWS Systems Manager OpsCenter integration in ServiceNow

To configure the AWS Systems Manager OpsCenter integration System Properties

1. In the navigator, enter **AWS Service Management**.
2. Choose **System Properties**, then **AWS Systems Manager - OpsCenter**.
3. Review the available settings and recommendations in the table below.

Available settings	Description
What action to take when synchronizing a new OpsItem with a severity 1	**Do Nothing**. This action only imports selected OpsItems for the scoped app. Users with scoped app permissions can view and choose to create an incident or problem. **Create Incident**. This action automatically creates incidents from OpsItems and syncs updates in ServiceNow to AWS Systems Manager - OpsCenter Default value: Create Incident
What action to take when synchronizing a new OpsItem with a severity 2	**Do Nothing**. This action only imports selected OpsItems for the scoped app. Users with scoped app permissions can view and choose to create incident or problem. **Create Incident**. This action automatically creates incidents from OpsItems and syncs

Available settings	Description
	updates in ServiceNow to AWS Systems Manager - OpsCenter Default value: Create Incident
What action to take when synchronizing a new OpsItem with a severity 3	**Do Nothing**. This action only imports selected OpsItems for the scoped app. Users with scoped app permissions can view and choose to create incident or problem. **Create Incident**. This action automatically creates incidents from OpsItems and syncs updates in ServiceNow to AWS Systems Manager - OpsCenter Default value: Do Nothing
What action to take when synchronizing a new OpsItem with a severity 4	· **Do Nothing**. This action only imports selected OpsItems for the scoped app. Users with scoped app permissions can view and choose to create incident or problem. · **Create Incident**. This action automatically creates incidents from based on OpsItems and syncs updates in ServiceNow to AWS Systems Manager - OpsCenter **Default value**: Do Nothing
Assignment Group (SYS_ID) for created incidents	ServiceNow incidents from AWS OpsItems need assignment group. To associate the assignment group for ServiceNow incidents from AWS OpsItems 1. Choose the section **`Set the assignment group sys_id`** or name that the Connector uses when creating incidents. 2. Enter the Assignment group sys_id. If you need to find the group sys_id, enter System Security in the left navigator. 3. Select the Groups module. 4. Search for the Group name. 5. Choose the group that you want to associate to ServiceNow incidents generated from AWS OpsItems and select **Copy sys_id**. You can now paste the copied sys_id into the AWS Systems Manager – OpsCenter System Properties.

Validating configurations

You can validate the AWS Service Management Connector for ServiceNow installation procedures.

AWS Service Catalog integration features

To order an AWS Service Catalog product

1. Log in to your ServiceNow instance as the end user (for this example, Abel Tuter).
2. Enter **`Service Catalog`** in the navigation filter and choose **Service Catalog**.
3. Select the **AWS Service Catalog S3 Storage** product to provision.
4. Enter the product request details, including product name, parameters, and tags.
5. Choose **Order Now** to submit the ServiceNow request and provision the AWS Service Catalog product.
6. After approximately one minute, you receive an order status acknowleging the submission.

To view provisioned products

End users can view products in two places on the ServiceNow portal: request items (Requests) or My AWS Products widgets.

To view products in Service Portal Requests

1. Choose **Requests** in the home page navigation bar.
2. Select the request item, which contains the AWS Service Catalog product and request item number.

   Note
   AWS product events and outputs update the request item. When you terminate the AWS product, the ServiceNow request item enters a state of **Closed Complete**.

To view products in the My AWS Products widget Service Portal Requests

1. In the My AWS Products widget and choose the AWS Select Product name that you entered into the request form.
2. View the **Status and Product Events**.
3. If you want to perform post-provisioned operational actions, choose **Request Update**, **Request Self-Service Action**, or **Terminate**.

AWS Config integration features

To see AWS Config details, configure the service settings to record data for the resource types of interest. For more information, see Setting Up AWS Config with the Console.

To view configuration item details from AWS Config in the ServiceNow CMDB

1. Log in to your ServiceNow instance as a user (for example, System Administrator) in the fulfiller view (Standard user interface view).
2. In the navigator, enter **`AWS Service Management`**.
3. Choose **AWS Config**.

Select and view relationships for available AWS resources. This table illustrates the available AWS resources, ServiceNow CMDB label, and table name.

AWS resources (AWS Config)	ServiceNow CMDB/Scoped App Table Label	ServiceNow CMDB/Scoped App Table Name
Accounts	CMDB CI Cloud Service Accounts	`mdb_ci_cloud_service_account`
VPCs	Cloud Networks	`cmdb_ci_network`
Availability Zones	Availability Zone	`cmdb_ci_availability_zone`
EC2 Instances	Virtual Machine Instance	`cmdb_ci_vm_instance`
EBS Volumes	Storage Volume	`cmdb_ci_storage_volume`
Security Groups	Compute Security Group	`cmdb_ci_compute_security_group`
Auto Scaling Group	Auto Scaling Groups	`x_126749_aws_sc_cmdb_ci_autoscaling_`
Network Interfaces	Cloud Mgmt Network Interface	`cmdb_ci_nic`
RDS Instances	Cloud DataBase	`cmdb_ci_cloud_database`
Subnets	Cloud Subnet	`cmdb_ci_cloud_subnet`
Load Balancers (V2)	Load Balancer Service	`cmdb_ci_lb_service`
S3 Buckets	Cloud Object Storages	`cmdb_ci_cloud_object_storage`
CloudFormation Stacks	CloudFormation Stack	`x_126749_aws_sc_cmdb_ci_cloudformati`
CloudFormation Provisioned Products	CloudFormation Provisioned Product	`x_126749_aws_sc_cmdb_ci_config_pp`
Tags	Key Value	`cmdb_key_value`
Lambdas	Cloud Function	`cmdb_ci_cloud_function`
Dynamo DB	DynamoDB Table	`cmdb_ci_dynamodb_table`
OS images	Images	`cmdb_ci_os_template`

Note
AWS resources, in scope for this release, determine configuration items and relationships. Configuration item relationships display AWS Regions. If you have questions or feedback, email `<aws-servicemanagement-connector@amazon.com>`.

AWS Systems Manager Automation Integration features

To request an AWS Systems Manager automation document (runbook) execution

1. Log in to your ServiceNow instance as the end user (for this example, Abel Tuter).
2. In the navigation filter, enter **`AWS Systems Manager`**, then choose **Systems Manager**.
3. Select an AWS Systems Manager document to execute.
4. Enter the request details, including parameters and tags.
5. Choose **Order Now** to submit the ServiceNow request and execute the AWS Systems Manager document.
6. You receive an order status acknowledging your request submission.

To view AWS Systems Manager document executions

1. Log in to your ServiceNow instance as the end user (for example, Abel Tuter).
2. In the navigation filter, enter **`AWS Systems Manager`**, then choose **Automation Executions**.
3. The user interface view displays the latest executions and provides the status.

AWS Systems Manager - OpsCenter Integration features

To view OpsItems from AWS Systems Manager - OpsCenter

1. To view AWS OpsItem, you must have the role `x_126749_aws_sc.opscenter_manager` with the Connector scope app.
2. Log in to your ServiceNow instance as a user (for example, System Administrator) in the fulfiller view (Standard user interface view).
3. In the navigator, enter **`AWS Service Management`**.
4. Choose **AWS Systems Manager - OpsCenter**.
5. Choose **OpsItems** to show a list of all synced Findings.
6. Choose an OpsItems to open the record.

   The Incident and Problem fields show the Incident for the OpsItems, if these exist.
7. Choose the ⓘ icon to the right of the field to preview the Incident.
8. Choose **Open Record** on the preview form to open the Incident.

   If the Connector configuration does not to automatically create a ServiceNow Incident when a new Finding syncs, you can create one manually. To do so, choose the link at the bottom of the form.

To execute an AWS Systems Manager – Automation Document from an AWS OpsItems associated to a ServiceNow Incident

One of the following conditions must be true to view or execute automation documents (runbooks):

- The user has the role Account Manager or Automation Manager.
- The user is assigned to a linked incident.
- The system parameter *Assignment Group (SYS_ID) for created incidents* is set to a valid group, a linked incident whose Assignment group is set to that group, and the user is a member of that group.

> **Note**
> To enable this feature, you must activate AWS Systems Manager Automation in the AWS Account and opt in to the Connector

1. Log in to your ServiceNow instance as a user (for example, System Administrator) in the fulfiller view (standard user interface view).
2. In the navigator, enter **`AWS Service Management`**. Then choose **AWS Systems Manager - OpsCenter**.
3. Choose OpsItems to show a list of all synced Findings. Then choose **Execute Automation Document**.
4. Choose your Automation Document.

   > **Note**
   > You can configure an OpsItem with Automation Documents and mark it as *Associated*.
5. Choose **Order Execution** next to the Automation Document you want to execute. You'll see the ServiceNow catalog item associated with the Automation Document.

6. Enter the necessary AWS parameters and choose **Order Now**.
7. In OpsItems in the scoped app. Choose the OpsItem in the Automation Document where you executed it.
8. In OpsItem Automation Executions, review the success or failure status.
9. Follow your organization's incident management procedures to determine related incident resolution actions.

Fields mapped from OpsCenter OpsItem records to ServiceNow Incident records

This table shows how AWS OpsItems map to ServiceNow incidents.

AWS Ops Center	ServiceNow Incident
Title	short_description
Description	description
CreatedTime	opened_at
Status	incident_state
Severity	impact/urgency
Priority	priority
CreatedBy	Not synced
LastModifiedTime	Not synced
LastModifiedBy	Not synced
Source	Not synced
OpsItemId	Not synced
OperationalData	Not synced
Category	Software

Incident Status is an integer in ServiceNow. We map OpsItem status values to values.

ServiceNow Incident Status	OpsCenter Status
New (primary)	Open
On Hold	Open
In Progress	In Progress
Resolved (primary)	Resolved
Closed	Resolved
Cancelled	Resolved

In this type of subjective mapping, we only change the target value if it is incompatible. An example of subjective mapping would be if *New* and *On Hold* in ServiceNow both map to *Open* in AWS. An example

of an incompatible target would be if the incident is *On Hold*, while we're synchronizing from AWS an OpsItem that is *Open*, and we don't change *On Hold*.

Priority: In Incident, you can't set the Priority field directly. The values of the **Impact** and **Urgency** fields calculate the **Priority** field. When synchronizing from AWS, we set by default the fields shown in the table below:

OpsItem Priority	**ServiceNow Incident**		
	Impact	**Urgency**	**Priority (Calculated)**
1	High	High	Critical (1)
2	Medium	High	High (2)
3	Medium	Medium	Moderate (3)
4	Low	Medium	Low (4)
5	Low	Low	Planning (5)

You can find these mappings in a ServiceNow table *Priority Data Lookup*. While we can use this table to find the required values of **Impact** and **Urgency**, note that you can customize the mappings and also define new priority values. Additionally, you might want a specific priority in AWS to map to an entirely different priority in an Incident or Problem.

AWS Security Hub Integration Features

To view Findings from AWS SecurityHub

1. To view AWS Security Hub Findings, you must have the role **x_126749_aws_sc.finding_manager** from the Connector scope app.
2. Log in to your ServiceNow instance as a user (for example, System Administrator) in the fulfiller view (standard user interface view).
3. In the navigator, enter **AWS Service Management**.
4. Choose **AWS Security Hub**.
5. Choose **Findings** to show a list of all synced Findings.
6. Choose a Finding to open the record.
7. The Incident and Problem fields show the Incident and Problem related to the Finding if these exist.
8. Choose the ⓘ symbol to the right of the field to preview the Incident or Problem.
9. Choose **Open Record** on the preview form to open the Incident or Problem.
10. If the Connector does not automatically create a ServiceNow Incident or Problem when a new Finding syncs, choose the link at the bottom of the form to create one manually.

This table shows how fields map from ServiceNow Findings records to ServiceNow as Incident or Problem records.

Finding	**Incident**	**Problem**
Created at	Opened at	Opened at
Company Name	Company	Company

Finding	Incident	Problem
Description	Description	Description
Criticality	Impact	Impact
Severity	Urgency	Urgency
Hardcoded to *software*	Category	Category
Id of record in *cmdb_ci_service* with name **AWS Security Hub**	Business service	Business service
Description	Short description	Short description
Reference to related Problem if it exists	problem_id	n/a

This table shows how fields synchronize between AWS Security Findings and ServiceNow Incidents or Problems.

AWS Security Hub value	ServiceNow Incident	ServiceNow Problem
Severity Label	Urgency	Urgency
Criticality	Impact	Impact

Fields synchronized between AWS Security Findings, Incidents, and Problems in ServiceNow

- Finding severity label → Problem/Incident urgency
  - INFORMATIONAL or LOW → LOW
  - MEDIUM → MEDIUM
  - HIGH or CRITICAL → HIGH
- Finding criticality → Problem/Incident impact
  - 0 - 29 → LOW
  - 30 - 69 → MEDIUM
  - 70 - 100 → HIGH

Fields synchronized from Findings to AWS Security Hub

- Severity (Label and Normalized)
- WorkflowStatus

ServiceNow additional features

This section provides information about additional features for the AWS Service Management Connector for ServiceNow.

Viewing products in the Standard User Interface (Fulfiller View)

To view provisioned products as an end user

1. Choose **My Assets** in the ServiceNow standard user interface.
2. In **My Asset Requests**, view the requests.
3. To view the product, personalize the list view to show the associated configuration item. To show items, choose the **Settings** icon in the header row of the table of asset requests.
4. Select **Configuration item (configuration_item)** and add it to the view with the **>** icon. Move it to below **Stage** in the list. The configuration item (the ordered product) shows in the list of assets.
5. To view the product, choose the configuration item name.
6. View the **Outputs** for the provisioned product in the **Outputs** tab of the form.
7. View the provisioning history of the product in the Product Events tab of the form.

To view provisioned products from the scoped app as an administrator

1. Log in to your ServiceNow instance as the end user (for example, Abel Tuter).
2. Enter **AWS Service Catalog** in the navigation filter and choose **Provisioned Products**. The user interface view displays the provisioned products.
3. Select a provisioned product to view the current status. You can also select post provisioned actions such as **Request Update**, **Request Termination**, as well as associated service actions.

Ordering AWS Service Catalog products through the ServiceNow Service portal

The Connector for ServiceNow version 3.7.1 supports ordering AWS Service Catalog products through Service Portal by using the Service Catalog and Order Something views. The release also includes pages and widgets you can add to Service Portal that enable users to view their provisioned products.

> **Note**
> The audience for the Service Portal Features section is a ServiceNow administrator or equivalent. The ServiceNow user requires permissions to modify the Service Portal.

Service portal widgets

The Connector for ServiceNow includes widgets that you can add to your Service Portal. It also includes two alternative view Portal Pages for the following:

- **My AWS Products** – Overview of all provisioned products the user owns.
- **AWS Product Details** – Details of a single provisioned product.

To access the new widgets, you need to update the Service Portal Designer.

To update the Service Portal Designer

1. Go to Create and edit a page using the Service Portal Designer.
2. Following the instructions, choose the **Service Portal Index** page.
3. Under the **Order Something** container, add the **My AWS** widget.
4. The new widget appears on your main Service Portal view.

Service portal pages

The following section describes the two new pages available in the Service Portal Beta release of the AWS Service Management Connector, **My AWS Products** and **AWS Product Details**. You can add links to these pages on the Service Portal home page or other pages by using the usual page configuration mechanism in Service Portal.

My AWS Products

An overview of all provisioned products that the user owns. Terminated products display separately from current products in a collapsed panel on initial page load.

The **My AWS Products** page is available using the following format:

```
http://<insertinstancename>.service-now.com/sp?id=aws_sc_pp
```

AWS Product Details

Details of a single provisioned product.

The **AWS Product Details** page is available using the following format:

```
http://<insertinstancename>.service-now.com/sp?id=aws_sc_pp_details&sys_id=<provisioned
 product id>
```

Reference: AWS API calls used in the Connector

- AWSBudgets.describeBudget
- AWSCloudFormation.registerType
- AWSCloudFormation.deregisterType
- AWSCloudFormation.describeTypeRegistration
- AmazonConfig.describeConfigurationRecorders
- AmazonConfig.getResourceConfigHistory
- AmazonConfig.listDiscoveredResources
- AmazonConfig.putResourceConfig
- AmazonConfig.selectResourceConfig
- AmazonConfig.selectAggregateResourceConfig
- AWSSecurityHub.batchUpdateFindings
- AWSSecurityTokenService.getCallerIdentity
- AWSServiceCatalog.createProvisionedProductPlan
- AWSServiceCatalog.deleteProvisionedProductPlan
- AWSServiceCatalog.describePortfolio
- AWSServiceCatalog.describeProduct
- AWSServiceCatalog.describeProductAsAdmin
- AWSServiceCatalog.describeProductView
- AWSServiceCatalog.describeProvisionedProduct
- AWSServiceCatalog.describeProvisionedProductPlan
- AWSServiceCatalog.describeProvisioningParameters
- AWSServiceCatalog.describeRecord
- AWSServiceCatalog.executeProvisionedProductPlan

- AWSServiceCatalog.executeProvisionedProductServiceAction
- AWSServiceCatalog.listBudgetsForResource
- AWSServiceCatalog.listLaunchPaths
- AWSServiceCatalog.listPortfolioAccess
- AWSServiceCatalog.listPortfolios
- AWSServiceCatalog.listProvisionedProductPlans
- AWSServiceCatalog.listServiceActionsForProvisioningArtifact
- AWSServiceCatalog.listStackInstancesForProvisionedProduct
- AWSServiceCatalog.provisionProduct
- AWSServiceCatalog.searchProducts
- AWSServiceCatalog.searchProductsAsAdmin
- AWSServiceCatalog.terminateProvisionedProduct
- AWSServiceCatalog.updateProvisionedProduct
- AWSSimpleQueueService.DeleteMessage
- AWSSimpleQueueService.DeleteMessageBatch
- AWSSimpleQueueService.ReceiveMessage
- AWSSimpleSystemsManagement.describeAutomationExecutions
- AWSSimpleSystemsManagement.describeDocument
- AWSSimpleSystemsManagement.getAutomationExecution
- AWSSimpleSystemsManagement.getDocument
- AWSSimpleSystemsManagement.listDocuments
- AWSSimpleSystemsManagement.startAutomationExecution
- AWSSimpleSystemsManagement.describeOpsItems
- • AWSSimpleSystemsManagement.getOpsItem

Version 2.3.4 release transition instructions

Previous versions of the AWS Service Catalog Connector for ServiceNow are not fully compatible with major release v3.7.1 due to:

- Streamlining the AWS Account configuration in the ServiceNow scoped app
- Removing the dependency of ServiceNow roles mapping to AWS identities.

Because of this, customers with previous versions (v2.3.4 and on) of the AWS Service Catalog Connector in the ServiceNow production environments must plan to transition to version 3.7.1 .

Major changes in version 3.7.1

The AWS Service Management Connector for ServiceNow v3.7.1 includes:

- Adding AWS Config and AWS Systems Manager integration features to the Connector.
- Streamlining the Connector Scoped App to six modules (Getting Started, Setup, AWS Service Catalog, AWS Config, AWS Systems Manager and AWS Security Hub.)

The AWS Service Management Connector for ServiceNow v3.7.1 no longer includes:

- Identities and Role Grants modules in the Connector Scoped app.

- Previously provisioned AWS Service Catalog products details in the Connector scoped app.

  Note
  The provisioned products details are still visible within AWS Service Catalog for the customer's AWS account. These details are no longer available in the ServiceNow Connector scoped app since we mapped these products based on ServiceNow user role grants as opposed to the new mapping to ServiceNow groups.

V2.3.4 Version sunset support and transition to 3.7.1

To provide customers time to plan and transition:

- AWS will support AWS Service Catalog Connector version 2.3.4 until December 31, 2021. Email <aws-sm-connector-issues@amazon.com> if you have any questions.
- Documentation for v2.3.4 is available. You must download and extract the zip file.

Transition recommendations

To transition to AWS Service Management Connector version 3.7.1 from a ServiceNow Production environment:

- Install the AWS Service Management Connector in a ServiceNow sandbox instance.
- Follow the AWS Service Management Connector installation instructions starting at the section called "Baseline permissions" (p. 93).

  Note
  There is a known issue with committing update sets that have a previous version of the Connector installed. Previewing the update set is successful. However, at the conclusion of the committing update, an error appears that states: "Version loading was stopped by DictionaryUpdateLoader…." We consider these errors as false positives after further testing that there is not impact on the update set. AWS is logging a ServiceNow support case and provides a new release if needed.
- Compare the two versions to plan how you move through your ServiceNow Development.
- Determine how you want to address AWS Service Catalog provisioned products in previous releases.
- Create a check list of all your transition action items that include but are not limited to:
  - Transition plan
    - Decision point on AWS Service Catalog provisioned products.
    - Steps to update/install Connector in ServiceNow Development to Production environments.
  - ServiceNow platform admin communications
  - End user communications

AWS Service Management Connector for Jira Service Management

To help customers provision and manage secure, compliant AWS resources into their Jira Service Management portal, AWS created the AWS Service Management Connector for Jira Service Management (formerly the AWS Service Catalog Connector).

The AWS Service Management Connector for Jira Service Management allows Jira Service Management end-users to provision, manage and operate AWS resources natively through Atlassian's Jira Service Management. Jira Service Management administrators can provide pre-approved, secured and governed

AWS resources to end-users through AWS Service Catalog, create and manage operational items through AWS Systems Manager OpsCenter, execute automation playbooks through AWS Systems Manager Automation and track resources in a configuration item view powered by AWS Config seamlessly on the Jira Service Management with the AWS Service Management Connector.

This simplifies AWS product request actions for Jira Service Management users and provides Jira Service Management governance and oversight over AWS products.

The AWS-supplied connector is available at no charge in the Atlassian Marketplace. This new feature is generally available in all AWS Regions where AWS Service Catalog, AWS Config and AWS Systems Manager services are available.

Topics

- Background (p. 129)
- Jira Service Management Supported Versions and Releases (p. 129)
- Getting Started (p. 130)
- Release Notes (p. 131)
- Baseline Permissions (p. 132)
- Configuring AWS Service Catalog (p. 137)
- Configuring Jira Service Management (p. 138)
- IT Lifecycle Management Setup and Use Case (p. 144)
- Validating Configurations (p. 149)
- Jira Additional Administrator Features (p. 152)

Background

AWS Service Catalog allows you to centrally manage commonly deployed AWS services and provisioned software products. It helps your organization meet consistent governance and compliance requirements, while enabling users to quickly deploy only the approved AWS services they need.

AWS Config enables you to assess, audit, and evaluate the configurations of your AWS resources. AWS Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations.

AWS Systems Manager gives you visibility and control of your infrastructure on AWS. Systems Manager provides a unified user interface so you can view operational data from multiple AWS services, investigate and resolve operational issues through OpsCenter, and allows you to automate operational tasks across your AWS resources.

Atlassian Jira Service Management is service desk software for modern IT teams. Jira Service Management request types enable self-service for developers and end users to order IT services based on request fulfillment approvals and workflows.

Jira Service Management Supported Versions and Releases

The AWS Service Management Connector for Jira Service Management supports Jira Service Management Server and Data Center versions. Jira software (Jira Service Management) releases are supported for the current and one previous version in each of the major, minor, and point release streams:

- Jira Server 7.13.17 to 8.15.0

- Jira Data Center 7.13.17 to 8.15.0

A Jira Service Management Cloud Connector is also available in the Atlassian Marketplace. For more information, see AWS Service Catalog for Jira Service Management Cloud.

Getting Started

Before installing the AWS Service Management Connector for Jira Service Management , you need an AWS account and an Atlassian instance with Jira Service Management pre-installed. Verify that you have the necessary permissions in your AWS account and Jira Service Management software.

For a zip file containing Connector add-on code as well as AWS Configuration files, download and extract the AWS Service Management Connector for JSM-Configuration Files.

> **Note**
> The Jira Products on AWS Reference Deployment Quick Start is available to use AWS resources for infrastructure required to install Jira Service Management data center version.

AWS prerequisites

To get started:

- To use AWS Service Catalog with the Connector, you need an AWS account to configure your AWS portfolios and products. For details, see Setting Up AWS Service Catalog (p. 6).
- To see AWS Config details, the service settings need to be configured to record data for the resource types of interest. It is recommended to include provisioned products and AWS CloudFormation stacks in addition to the major resource types used by your team. For details,see Setting Up AWS Config with the Console.
- To use AWS Systems Manager Automation with the Connector, no AWS-side setup is required. A number of automation documents are provided by AWS as standard. If you have additional automation documents you wish to use, they will be available in the Connector. For details, see Working with Automation Documents (Playbooks).
- To use AWS Systems Manager OpsCenter with the Connector, enable OpsCenter in the AWS Systems Manager console. For details see, *Getting Started with OpsCenter*. The Connector also enables viewing resources and automation documents (runbooks) associated to OpsItem. For details to associate resources to OpsItems in AWS OpsCenter, see Working with Related Resources . For details to associate automation documents to OpsItems in AWS OpsCenter, see Remediating OpsItem issues using Systems Manager automation.

For each AWS account, the Connector for Jira Service Management also requires API access with the section called “Baseline permissions” (p. 93) as described below.

Jira Service Management prerequisites

In addition to your AWS account, you need the Jira Service Management software installed on your Atlassian instance before you can install the AWS Service Management Connector add-on. The Jira Service Management administrator needs the admin role to install the AWS Service Management Connector add-on.

Before configuring your AWS connector, ensure that you follow Atlassian recommendations for securing your Jira Service Management instances. For more information, see Preventing Security Attacks.

The Connector for Jira Service Management add-on is available to download in the Atlassian Marketplace.

Release Notes

Version 1.7.1 of the AWS Service management Connector for Jira Service Management (formerly the AWS Service Catalog Connector) is a patch release that fixes a bug in the workflow for end users. Version 1.7.1 also includes recently launched Version 1.7.0 features:

AWS Service Catalog integration features

- The ability for Jira Service Management administrators to assign the default Jira user used by the Jira workflow engine.
- Enable Jira Service Management administrators to configure AWS product request form components available for end users to view.
- Enable Jira Service Management administrators to create AWS Tags across provisioned products.
- The ability for end users to view AWS specific parameters on EC2 resources such as Availability Zones, Image ID, Instance Id, KeyPair, Security Group and VPC.

AWS Systems Manager OpsCenter integration features

- Create and update a Jira Issue when an operational item (OpsItem) is created or updated in AWS Systems Manager OpsCenter.
- Ability to update OpsItems in AWS Systems Manager OpsCenter when the Jira issue is updated in Jira Service Management.
- Ability to view and execute automation runbooks to resolve OpsItems and view execution results from the Jira Issue.

  This version also includes prior AWS Service Management Connector for Jira Service Management features such as:

AWS Service Catalog

- Rendering AWS Service Catalog portfolios and products in the Jira Service Management Customer Portal and Jira Agent views.
- The ability for Jira Service Management administrators to associate Jira Service Management approval groups to AWS Service Catalog portfolios to require approvals for Jira Service Management user product requests.
- The ability for Jira Service Management users to request AWS Service Catalog products through Jira Service Management.
- The ability for administrators to view portfolio and product budgets and actual costs. (Requires budgets to be associated within AWS Service Catalog.)
- Support for AWS Service Catalog self-service actions for Jira Service Management users to update and terminate products.
- Support for AWS CloudFormation StackSets, enabling launch of AWS Service Catalog products across multiple regions and accounts.
- Support for AWS CloudFormation Change Sets, enabling a preview of resource changes prior to a launch or update.

AWS Config

- Rendering of AWS Config configuration item details on provisioned AWS products through Jira Service Management request.
- The ability to view the configuration item relationships in a tree structure.

- The ability to associate AWS Config items details to Jira issues.

AWS Systems Manager Automation

- Rendering of AWS Systems Manager automation documents in the Jira Service Management Customer Portal and Jira Agent views.
- The ability for Jira Service Management administrators to associate AWS Systems Manager automation to Jira projects.
- The ability for Jira Service Management users to request and execute AWS Systems Manager automation documents through Jira Service Management.
- The ability to create Jira issues (incidents) that provide actionable remediation suggestions through a Connector specific AWS Systems Manager automation document.
- Support for multiple AWS accounts.
- Support for FIPS endpoints and usage in the AWS GovCloud East and GovCloud West regions.
- Support for the latest releases of Jira Service Management Server and Data Center versions.

Baseline Permissions

This section provides instructions on how to set up the baseline AWS users and permissions needed for the AWS Service Management Connector for Jira Service Management.

Available template for baseline permissions

To use an AWS CloudFormation template to set up the AWS configurations of the Connector for Jira Service Management, see the AWS configurations for Connector for Jira Service Management for Connector for Jira Service Management v1.7.1 - AWS Commercial Regions and Connector for Jira Service Management v1.7.1 - AWS GovCloud West Region.

> **Note**
> If you use the Connector for Jira Service Management v1.7.1_AWS Configuration template, skip to Configuring AWS Service Catalog.

For each AWS account, the Connector for Jira Service Management requires two sets of an access key identifier and a secret key for API access. These correspond to users in AWS Identity and Access Management (IAM). Specifically, you should set up:

- An IAM user to sync AWS resources to Jira Service Management.
- An IAM user able to perform end user functionality to provision and execute requests exposed through Jira Service Management, including assuming any roles required to perform the provisioning and execution (launch roles are recommended for AWS Service Catalog).

These can be the same user and can be an existing user, but in line with the best practice to give minimal permissions it is recommended these be two new users for Connector.

Full details of the permissions required by these users is included below, with an AWS CloudFormation template to facilitate setting this up. Baseline permissions enable an end user to provision the following AWS services: Amazon Simple Storage Service and Amazon Elastic Compute Cloud. To allow end users to provision AWS services beyond the baseline permissions, you must include the additional AWS service permissions to the appropriate roles.

> **Note**
> To use an AWS CloudFormation template to set up the AWS configurations of the Connector for Jira Service Management, see the two JSON AWS Configurations for Connector for Jira Service

Management v1.7.1 - AWS Commercial Regions and Connector for Jira Service Management v1.7.1 - AWS GovCloud West Region.

Creating AWS Service Management Connector Sync User

The following section describes how to create the AWS Connector sync user and associate the appropriate IAM permissions. To perform this task, you need IAM permissions to create new users.

To create AWS Service Management Connector sync user

1. Go to **Creating IAM Policies**. Following the instructions there, create a policy called **SSMOpsItemActionPolicy** for Jira administrators to create and manage AWS Systems Manager OpsItems. Copy the following policy and paste it into **Policy Document**:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:CreateOpsItem",
                "ssm:GetOpsItem",
                "ssm:UpdateOpsItem",
                "ssm:DescribeOpsItems",
                "ssm:CreateOpsItem"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Go to **Creating an IAM User in Your AWS Account**. Following the instructions there, create a sync user (that is, SCSyncUser). The user needs programmatic and AWS Management Console access to follow the Connector for Jira Service Management installation instructions.

   Set permissions for your sync user (SCSyncUser). Choose **Attach the following policies directly** and select **AWSServiceCatalogAdminReadOnlyAccess**, **AmazonSSMReadOnlyAccess**, and **SSMOpsItemActionPolicy.**
3. Also add a policy allowing **budgets:ViewBudget** on all resources (*).
4. Review and choose **Create User**.
5. Note the access and secret access information. Download the .csv file that contains the user credential information.

Creating AWS Service Management Connector End User

The following section describes how to create the AWS Service Management Connector end user and associate the appropriate IAM permissions. To perform this task, you need IAM permissions to create new users.

To create AWS Service Management Connector end user

1. Go to Creating an IAM User in Your AWS Account. Following the instructions there, create a user (such as SCEndUser). The user needs programmatic and AWS Management Console access to follow the Connector for Jira Service Management installation instructions.
2. For products with AWS CloudFormation StackSets, you need to create a stack set inline policy. With AWS CloudFormation StackSets, you can create products that are deployed across multiple accounts and regions. Using an administrator account, you define and manage an AWS Service Catalog product and use it as the basis for provisioning stacks into selected target accounts across specified regions. You need to have the necessary permissions defined in your AWS accounts.

   To set up the necessary permissions, go to Granting Permissions for Stack Set Operations. Following the instructions there, create an **AWSCloudFormationStackSetAdministrationRole** and an **AWSCloudFormationStackSetExecutionRole**.
3. Create the stack set inline policy to enable provisioning a product across multiple regions in one account, replacing the `arn` number string with your account number.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Action": [
        "sts:AssumeRole"
        ],
        "Resource": [
        "arn:aws:iam::123456789123:role/AWSCloudFormationStackSetExecutionRole"
        ],
        "Effect": "Allow"
        },
        {
        "Effect": "Allow",
        "Action": [
        "iam:GetRole",
        "iam:PassRole"
        ],
        "Resource":       "arn:aws:iam::123456789123:role/
AWSCloudFormationStackSetAdministrationRole"
        }
     ]
}
```

   Note
   The Connector for Jira Service Management v1.7.1 - AWS Commercial Regions and Connector for Jira Service Management v1.7.1 - AWS GovCloud West Region templates include the AWS CloudFormation StackSet permissions.
4. Add the following permissions (policies) to the user **SCEndUser**:

   - **AWSServiceCatalogEndUserFullAccess** - (AWS managed policy)
   - **StackSet** - (inline policy)
   - **AmazonS3ReadOnlyAccess** - (AWS managed policy)

- **AmazonEC2ReadOnlyAccess** - (AWS managed policy)
- **AWSConfigUserAccess** - (AWS managed policy)
- **SSMOpsItemActionPolicy** - (inline policy)

Note
For AWS Service Catalog products with AWS CloudFormation StackSets, you need to include the read only permissions for the services you want to provision. For example, to provision an Amazon S3 bucket, include the **AmazonS3ReadOnlyAccess** policy to the **SCEndUser** role.

5. Also add a policy allowing the following on all resources (*): **ssm:DescribeAutomationExecutions**, **ssm:DescribeDocument**, and ssm:**StartAutomationExecution**.
6. Review and choose **Create User**.
7. Note the access and secret access information. Download the .csv file that contains the user credential information.

Creating SCConnectLaunch Role

The following section describes how to create the **SCConnectLaunch** role. This role is used to place baseline AWS service permissions into the AWS Service Catalog launch constraints. For more information, see AWS Service Catalog Launch Constraints (p. 46).

To create SCConnectLaunch role

1. Create the **AWSCloudFormationFullAccess** policy. Choose **create policy** and then paste the following in the JSON editor.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
            "cloudformation:DescribeStackResource",
            "cloudformation:DescribeStackResources",
            "cloudformation:GetTemplate",
            "cloudformation:List*",
            "cloudformation:DescribeStackEvents",
            "cloudformation:DescribeStacks",
            "cloudformation:CreateStack",
            "cloudformation:DeleteStack",
            "cloudformation:DescribeStackEvents",
            "cloudformation:DescribeStacks",
            "cloudformation:GetTemplateSummary",
            "cloudformation:SetStackPolicy",
            "cloudformation:ValidateTemplate",
            "cloudformation:UpdateStack",
            "cloudformation:CreateChangeSet",
            "cloudformation:DescribeChangeSet",
            "cloudformation:ExecuteChangeSet",
            "cloudformation:DeleteChangeSet",
            "s3:GetObject"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Create a policy called **ServiceCatalogSSMActionsBaseline**. Follow the instructions on Creating IAM Policies, and paste the following into the JSON editor.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Sid": "Stmt1536341175150",
           "Action": [
               "servicecatalog:ListServiceActionsForProvisioningArtifact",
               "servicecatalog:ExecuteprovisionedProductServiceAction",
               "ssm:DescribeDocument",
               "ssm:GetAutomationExecution",
               "ssm:StartAutomationExecution",
               "ssm:StopAutomationExecution",
               "cloudformation:ListStackResources",
               "ec2:DescribeInstanceStatus",
               "ec2:StartInstances",
               "ec2:StopInstances"
           ],
           "Effect": "Allow",
           "Resource": "*"
       }
   ]
}
```

3. Create the **SCConnectLaunch** role. Assign the trust relationship to AWS Service Catalog.

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

4. Attach the relevant policies to the **SCConnectLaunch** role. Attach the following baseline IAM policies:

   - **AmazonEC2FullAccess** (AWS managed policy)
   - **AmazonS3FullAccess** (AWS managed policy)
   - **AWSCloudFormationFullAccess** (custom managed policy)
   - **ServiceCatalogSSMActionsBaseline** (custom managed policy)

   Note
   To use an AWS CloudFormation template to set up the AWS configurations of the Connector for Jira Service Management, see the two JSON AWS Configurations for Connector for Jira Service

Management v1.7.1 - AWS Commercial Regions and Connector for Jira Service Management v1.7.1 - AWS GovCloud West Region.

Configuring AWS Service Catalog

Now that you have created two IAM users with baseline permissions in each account, the next step is to configure AWS Service Catalog. This section describes how to configure AWS Service Catalog to have a portfolio that includes an Amazon S3 bucket product. Use the Amazon S3 template located at Creating an Amazon S3 Bucket for Website Hosting for your preliminary product. Copy and save the Amazon S3 template to your device.

To configure AWS Service Catalog

1. Create a portfolio by following the steps at Step 3: Create an AWS Service Catalog Portfolio (p. 12).
2. To add the Amazon S3 bucket product to the portfolio you just created, in the AWS Service Catalog console, on the **Upload new product** page, enter product details.
3. For **Select template**, choose the Amazon S3 bucket AWS CloudFormation template you saved to your device.
4. Set **Constraint type** to **Launch** for the product that you just created with the SCConnectLaunch role in the baseline permissions. For additional launch constraint instructions, see AWS Service Catalog Launch Constraints (p. 46).

   Note
   The AWS configuration design requires each AWS Service Catalog product to have either a launch or StackSet constraint. Failure to follow this step may result in an *Unable to Retrieve Parameter* message within Jira Service Management Service Catalog.

Creating Stack Set Constraint

AWS CloudFormation StackSets enable users to create products that are deployed across multiple accounts and regions. In AWS Service Catalog, a stack set constraint allows you to configure product deployment options.

To apply a stack set constraint to an AWS Service Catalog product

1. Go to AWS Service Catalog as a catalog administrator.
2. Choose the portfolio that contains the product you want to apply a constraint to.
3. Expand **Constraints** and choose **Add constraints**.
4. Choose the product from **Product** and set **Constraint type** to **Stack Set**. Choose **Continue**.
5. On the **Stack set constraint** page, enter a description.
6. Choose the accounts in which you want to create products.
7. Choose the regions in which you want to deploy products. Products are deployed in these regions in the order that you specify.
8. Choose the **AWSCloudFormationStackSetAdministratorRole** role that will be used to manage your target accounts.
9. Choose the **AWSCloudFormationStackSetExecutionRole** role that the administrator role will assume.
10. Choose **Submit**.

    Note that the Connector for Jira Service Management v1.7.1 - AWS Commercial Regions and Connector for Jira Service Management v1.7.1 - AWS GovCloud West Region templates create the permissions as well as the outputs needed for stack set constraints. Example stack set outputs:

```
SCStackSetAdministratorRoleARN
arn:aws:iam::123456789123:role/AWSCloudFormationStackSetAdministrationRole
 SCIAMStackSetExecutionRoleName
AWSCloudFormationStackSetExecutionRole
SCIAMAdminRoleARN
arn:aws:iam::123456789123:role/AWSCloudFormationStackSetAdministrationRole
```

Note that AWS Service Catalog products can have either a stack set or a launch constraint, but not both.

Configuring Jira Service Management

The AWS Service Management Connector for Jira Service Management is released as a conventional Jira Service Management add-on. Add-ons are code changes to the Jira software that extend its functionality or extend the functionality of Jira Service Management software. The Connector for Jira Service Management add-on is available to download in the Atlassian Marketplace.

After completing the IAM and AWS Service Catalog configurations, you must configure Jira Service Management. Installation tasks within Jira Service Management include:

- Clear your web browser cache.
- Install the Jira Service Management Connector add-on.
- Configure AWS Service Management Connector add-on, including accounts, schedule sync, request and approval permissions, and core operational settings.

  Note
  You'll need to select a dedicated administrator on the Connector settings page to perform operations on Jira tickets, such as status transitions or comments. If you don't select a dedicated administrator, we list the first administrator in the dropdown by default. For more details, see the **Core Operation Settings** under the **Configuring Connector Settings** sub-section.

Clear Web Browser Cache

Clear your web browser cache to remove previously rendered Jira Service Management forms.

Installing Jira Service Management Connector Add-on

1. Log in to your Jira instance as an admin.
2. Open the admin menu and choose **Add-ons**.
3. On the **Manage add-ons** screen, choose **Find new apps** or **Find new add-ons** from the left side of the page.
4. Find AWS Service Management Connector for JSM. The search results should include app versions compatible with your Jira instance.
5. Choose **Install** to download and install your app.
6. Proceed to **Configuring AWS Accounts and Regions**.

Alternatively, you can download the code from the OBR file: AWS Service Management Connector for Jira Service Management v1.7.1 OBR.

1. Go to **Manage apps**.
2. Select **Upload app** and upload the OBR file.
3. Proceed to **Configuring AWS Accounts and Regions**.

The Connector for Jira Service Management version 1.7.1 add-on can be applied to the supported Jira software (Jira Service Management) releases noted above.

Configuring AWS Accounts and Regions

Once the AWS Service Management Connector is installed, configure it by choosing the Jira administration icon in the top right, then choosing **Add-ons**.

1. From the AWS Service Catalog section on the left navigation menu, choose **AWS Accounts**.
2. Choose **Connect new account**.
3. Enter the account alias (used to identify the AWS account in the Connector).
4. Enter the credentials for SC-sync-user. This is the access key identity and credentials for a sync user saved from the AWS configuration. SC-sync-user credentials are used to retrieve portfolios and products to make them available through Jira Service Management. You will have the opportunity to set the groups allowed to access these.
5. Enter the credentials for SC-end-user. This is the access key identity and credentials for the end user saved from the AWS configuration. The SC-end-user credentials are used to provision products on behalf of a Jira user.
6. Add AWS Regions containing AWS Service Catalog products and portfolios that you want available in Jira Service Management.
7. Choose **Test Connectivity**.
8. Upon successful connection status, choose **Connect**.

> **Note**
> We recommend that the sync user and end user be new users in AWS used only with AWS Service Management Connectors. These users should have minimum required privileges. An AWS CloudFormation template with the minimal permissions for AWS Service Management Connectors is available.

Configuring AWS Service Catalog Portfolios within Jira

AWS Product Access

This section describes how to configure AWS Service Catalog portfolios within Jira.

Once your account or accounts are set up and connectivity is successful, use the **AWS Account** page to manage, for each account, which groups are permitted to access each portfolio in each Region. You can expand and collapse each Region and edit and add groups for each portfolio. Only users in the designated groups have access to those products. By default, no groups have access.

> **Note**
> At least one group must be associated to an AWS Service Catalog portfolio for Jira Service Management end users to request AWS products.

To provision products and portfolios

1. Choose **AWS Accounts**.
2. Choose **Manage** for the AWS account on which you want to configure portfolios.

3. Under **Portfolios**, expand the Region associated with the account. Portfolios are displayed under each Region.
4. In the **Permission to request** column, choose **Add groups** for the portfolios that you want to make visible in Jira Service Management. Select the group that you want to be able to see and request AWS Service Catalog products.

   Note
   Because the AWS Service Management Connector for Jira Service Management allows Jira users to provision AWS products in the portfolios their groups have access to, and to control those provisioned products, users should be reminded of the importance of maintaining the security of their Jira accounts.
5. If products in this portfolio do not require approvals, choose **Save**.

Jira Service Management Approvals for Products in AWS Service Catalog Portfolios

The AWS Service Management Connector for Jira Service Management enables administrators to configure approvals for products at the portfolio level. All products within a portfolio that contains approval permissions will require approval, so AWS and Jira administrators may need to collaborate on the AWS Service Catalog portfolio structure.

To configure the approval process

1. Choose **AWS Accounts**.
2. Choose **Manage** on the AWS account for which you want to configure portfolio approvals.
3. In the **Permission to approve** column, choose **Add groups** for the portfolios that require product approvals.
4. Select **Require approval for provisioning**.
5. Under **Permission to approve**, choose **Add group**.
6. Choose **Save**.

Note
If a portfolio only has a group associated with **Permissions to request**, products within the portfolio immediately provision when the product request is submitted.

Viewing Products and Budgets

Two other tabs in the **Admin -> AWS Accounts -> Manage** section let you view information on portfolios, for reference. The **Available Products** tab lists the products in the portfolio and budgetary information on each. The **Budgets** tab gives overall budgetary information on the portfolio.

Note
Note: Additional configurations for the AWS Service Catalog request form and Automated Tags are detailed in the next section Configuring Connector Settings.

Configuring Connector Settings (Jira Project Enablement and Request Type)

In addition to configuring AWS accounts, the AWS Service Management Connector contains selections AWS services, UI settings (AWS Service Catalog), enabling projects and configuring AWS Systems Manager OpsCenter.

Note
There are no per-account settings for AWS Config and AWS Systems Manager Automation through the JSM Connector.

Connector features enabled by default

To configure the default Connector features for specific AWS services

For a new installation of Connector, the default project configuration is for all Connector features (AWS Service Catalog, AWS Config, AWS Systems Manager Automation, and AWS Systems Manager OpsCenter) to be enabled. If you are upgrading an existing installation, for security reasons, new features are initially not enabled.

1. In the left navigation menu, under **AWS Service Management**, select **Connector settings**.
2. At the top, under **Connector features enabled by default**, select each feature depending whether you want projects using the default configuration to be able to use them or not.
3. Choose **Save**.

UI Settings (AWS Service Catalog)

Configure the AWS Service Catalog product widget components to make them viewable to end users.

To address the varying personas of end users requesting AWS products, the Connector for Jira Service Management includes an add-on app setting to enable or disable components of the AWS product widget. By default, all AWS product components are enabled.

To modify the AWS product view

1. In the left navigation menu, under AWS Service Management, **AWS Connector settings**.
2. Go the UI settings (Service Catalog) section and deselect any AWS product component such as:

    1. Allow the product name to be edited. (If unchecked, an autogenerated name will be used which the user will not be able to edit.)
    2. Allow the user to select a launch option. (If unchecked, the default launch option will be selected and the launch option section hidden.)
    3. Allow the user to select a product version. (If unchecked, the default product version will be selected and the product version section hidden.)
    4. Allow the user to add or edit tags. (If unchecked, default values will be chosen for tag options and the tags section will be hidden.)
    5. Allow user to create a plan for creation or update of a provisioned product. (If unchecked, the plans section will be hidden.)
3. Choose **Save**.

Projects enabled for the Connector

The AWS Service Management Connector for Jira Service Management requires the add-on to be associated to one or more Jira projects and, for JSM request types. You can configure which Connector features are enabled for each Jira project.

To configure the Jira projects for AWS Service Catalog, AWS Config, AWS Systems Manager Automation and AWS Systems Manager OpsCenter

1. In the left navigation menu, under **AWS Service Management Connector**, select **Connector settings**.
2. Under **Projects enabled for Connector**, you must enable at least one Jira project. You can create a new Jira Service Management project or add an existing one. Only users with access to the associated project will be able to access the Connector. When this update is applied, the Connector adds the necessary issue types and other Jira items needed for AWS Service Catalog products to be available in those projects. You can return to this screen and add or remove projects at any time.

3. Projects initially take the default configuration in regards which Connector features are enabled. Choose **Edit** in a project row to change the configuration for individual projects. It is permitted for projects to use more features than the default.
4. Choose **Save.**

   Note
   For end-users to be able to request AWS Service Catalog products, one or more projects must be enabled and users must have Jira permissions to create issues in the Jira project and Permission to Request in the Jira settings for the AWS Account for at least one portfolio with products.

   AWS Systems Manager Automation enablement considerations

   It Is not currently supported to have fine-grained permissions in Jira for which users and groups should be allowed to access which AWS Systems Manager automation documents. If a project is enabled for Systems Manager Automation, then any user with permission to create issues in that project will have the ability to run any of the automations. Access can be restricted by limiting which users have access to projects with AWS Systems Manager Automation enabled.

OpsCenter Configuration

Once you've enabled project(s) for the Connector, AWS Systems Manager OpsCenter requires Jira admins to associate Jira project(s) to this integration as well as determine the full sync and delta sync intervals.

To associate the Jira projects enabled for the Connector to the AWS Systems Manager OpsCenter integration features

1. In the left navigation menu, under **AWS Service Management Connector**, select **Connector settings**.
2. Create a new Jira Service Management Project. Under **OpsCenter Configuration**, you must enable at least one Jira project. You can create a new Jira Service Management project or add an existing one. Only users with access to the associated project will be able to access the Connector. When this update is applied, the Connector adds the necessary issue type to associated project(s). You can return to this screen and add or remove projects at any time.
3. Under **OpsCenter Configuration**, in the **Full Sync Interva**l and **Delta Sync Interval** fields, you can change the sync interval if you want. The **Full Sync** and **Delta** interval determines how often Jira Service Management conducts syncs all or changes to OpsItems details with AWS Systems Manager OpsCenter respectively. Increasing this number will reduce the number of API calls to AWS but will mean it takes longer for OpsItems updates to be reflected in the Connector.
4. Choose **Save**.

Core Operational Settings

To configure operational settings for the AWS Service Management Connector for Jira Service Management

1. In the left navigation menu, under **AWS Service Management Connector**, select **Connector settings**.
2. Under **Core operational settings**, in the **Synchronization interval** field, you can change the sync interval if you want. This interval determines how often Jira Service Management syncs with AWS. Increasing this number will reduce the number of API calls to AWS but will mean it takes longer for updates in AWS portfolios and automation documents to be reflected in the Connector. Information on actively provisioning products and ongoing automation executions updates more frequently.
3. Under **Core operational settings**, in the **JIRA Administrator to run as** field, you can change the admin user assigned to perform automated operations within JIRA.

It is important to emphasize the Connector performs many actions within Jira, and needs to do those actions as a Jira user. By default, Connector will choose the Jira Admin user with the lowest ID, which works for many environments.

However, that may be the wrong strategy if the initial admin user has been disabled or if a different admin user is preferred. For clarity within the Connector, it can be a good idea to create a new user called, for example, "AWS Connector Admin", and select that as the default user.

Actions performed automatically by the Connector, such as synchronizing OpsItems from AWS or adding a commenting on detecting a change to an AWS Provisioned Product, will be recorded as being done by this user. This does not affect actions performed by end users, such as requesting a Provisioned Product or manually creating an OpsItem in Jira, which as usual will be recorded as being done by that end user.

This user should have global admin permissions, JSM permissions, and admin access to each of the AWS-enabled projects.

4. Choose **Save**.

Note
We recommend no changes to entities that the plugin created, such as the addition of fields, workflows, issue types, screens, and so on.

Configuring Automated Tags (AWS Service Catalog)

The AWS Service Management Connector v1.7.1 enables Jira administrators to add tags (metadata) to AWS Service Catalog provisioned products globally across the add-on or granularly at the portfolio level. These tags are not visible to end users.

Two tag types are available in this release:

- Generic tags in which the admin can enter the key and value.
- AWS Service Catalog Request Type tags in which the admin can enter the following syntax for key and value:

AWS Service Catalog Request Type tags

Key	Value
Project Code	${PROJECT_CODE}
Project Name	${PROJECT_NAME}
Project Name	${ISSUE_ID}
Username	${USERNAME}
Opened By	${OPENED_BY}

To add generic AWS tags to AWS Service Catalog provisioned products in Jira Service Management

1. In the left navigation menu, under **AWS Service Management**, select **Automated Tags**.
2. For Global level tags, enter the Key and Value entries. Under **Portfolio**, select **Global** (set by default). Select the **+** icon to insert.

3. For Portfolio level tags, enter the Key and Value entries. Under **Portfolio**, select the Portfolio dropdown to choose the portfolio associated to associate tag. Select the **+** icon to insert.

To add in-scope request type AWS tags to AWS Service Catalog provisioned products derived from Jira Service Management

1. In the left navigation menu, under **AWS Service Management**, select **Automated Tags**.
2. For Global level tags, enter the Key and Value entries. Under **Portfolio**, select **Global** (set by default). Select the **+** icon to insert.
3. For Portfolio level tags, enter the Key and Value entries. Under **Portfolio**, select the Portfolio dropdown to choose the portfolio associated to associate tag. Select the **+** icon to insert.

   Once products are provisioned, you can see in the AWS console that these tags are associated to the resource.

Configuring Project Request Type Groups

The AWS request type must be in a group for users to be able to access it in Jira Service Management. Enabling Jira projects, as described in Configuring Connector Settings (Jira Project Enablement and Request Type) (p. 140), makes AWS product request types available, but Jira Service Management users won't see the request type until it is added to a **Request Type Group**.

To configure request types

1. In the AWS Service Management Connector for Jira Service Management , go to the **Connector settings** page.
2. In the **Projects** section, choose **add the AWS request type**.
3. Select **Add existing request type** in the upper right-hand corner.
4. Select **Request AWS product** from the available request type.
5. Select **Edit Groups** for the **Request AWS product** request type.
6. On the **Edit groups** form, select **General**, then choose **Save**.

Note
A custom **Request AWS Product** request type was created for the Connector for Jira Service Management, so edits to the **Request AWS Product** request type are not required. You can add a request type to an existing group. If you don't have a group, create a new group and add the request type to it.

IT Lifecycle Management Setup and Use Case

The AWS Service Management Connector for Jira Service Management allows Jira Service Management end-users to provision, manage, and operate AWS resources natively through Atlassian's Jira Service Management. To enable the IT Lifecycle Management scenario, you need to configure:

- AWS Config Linked Resources
- Suggested AWS Systems Manager Remediations for an Issue
- Sample Use Case: Automatically Creating Jira Issues for IT Lifecycle Management - Remediating non-compliant public S3 buckets

AWS Config and suggested AWS Systems Manager remediations for any Jira issue

The Connector provides two fields to use for any issue.

- *AWS Config Linked Resources*: this field allows any resource with an entry in AWS Config to have its AWS Config information displayed on the issue in Jira. The information can be expanded and information on relations also displayed. Multiple AWS resources can be linked to an issue.
- *AWS Systems Manager Automation Suggested Remediation*: this field allows SSM automation documents to be recorded against an issue. These are then displayed as suggested ways to correct the issue. When a Jira user views the issue, they can see these suggested remediations and click to apply them. Multiple suggested remediations can be attached to an issue.

The two fields can be used individually but they work very well together. When an incident is detected on an AWS resource or set of resources, setting both allows a Jira user to see the configuration information to confirm or better understand the problem, apply remediations to fix common problems, and then confirm in the AWS Config information that the problem has been fixed.

To add AWS fields to an existing issue

1. The project or projects must be enabled for the Connector in **Connector Settings** under **Admin -> Manage Add-Ons**, as described in the Connector setup guide.
2. Go to **Admin**, then **Projects**, and open the project you want to use these fields with.
3. Choose the issue type you want to use in the menu at left.
4. Select to view *Fields* in the top right (if not already selected). It should then show a list of fields enabled for the screen.
5. Scroll to the bottom where there should be a textbox where you can type additional fields. Enter **AWS**, then select the AWS field you want to use.
6. Choose **Add** to apply.
7. Repeat the previous step for the other field if you wish to use it.
8. Repeat these steps for each issue type you wish you use these fields with. Some issue types may share screens so the field may already be added for some.

It is important also to make a note of the field ID for the field or fields you are using. Choose **Admin -> Issues -> Custom fields** and select **Configure** on each field.

Inspect the URL that is opened to see the numeric field ID. It should be a 5-digit number. Alternatively, for any issue in a project where you've added the field (following the instructions above), the REST API at `/rest/api/2/issue/PRJ-1/editmeta` (for example, `http://localhost:2990/jira/rest/api/2/issue/PRJ-1/editmeta`) will include information on the fields and should contain an entry `customfield_#####: { ..., name: "AWS Config Linked Resources", ... }`, where `#####` is the numeric field ID.

Once these fields are enabled for projects and issue types, use the Jira REST API to create or update issues with values for these fields. You can use tools such as CloudWatch, AppDynamics, Jenkins, or a Systems Manager Automation Document (provided in the next section). The REST API endpoint to update an issue is `/rest/api/2/issue/issue-key` and the general schema to pass to set a value is

```
                    { "update": {
    "customfield_field-ID": [ {
      "set": "value"
    } ]
} }
```

See the examples below, or for more information on the REST API, see JIRA Developer Documentation : Updating an Issue through the JIRA REST APIs.

AWS Config Linked Resources

The **AWS Config Linked Resources** field should be set to the JSON string representation of a list of objects (maps) corresponding to the linked resources, each with the following keys:

- *resourceId*: the ID of the resource in AWS Config
- *resourceType*: the type of the resource in AWS Config
- *accountName*: the name or alias of the AWS account configured in Jira that should be used to access this resource
- *region*: the region where AWS Config should be accessed to get information on this resource

For example, the following value would show information on the S3 bucket `my-bucket` in `eu-central-1`, using the account and end user credentials that are specified in Jira for the AWS account identified in Jira as `MyAccount1`:

```
            [ { "resourceId": "my-bucket",
"resourceType": "AWS::S3::Bucket",
"accountName": "MyAccount1",
"region": "eu-central-1" } ]
```

AWS Systems Manager Automation Suggested Remediation

The **AWS Systems Manager Automation Suggested Remediation** field should be set to the JSON string that represents a list of objects (maps) that correspond to the automation documents as remediations, each with the following keys:

- *documentName*: the name of the Systems Manager automation document
- *description*: a description of the remediation to display in Jira; this may be different to the document description in AWS and might explain why it is a good remediation for the issue where this is being set
- *accountName*: the name/alias of the AWS account configured in Jira that should be used to access this resource
- *region*: the region where AWS Config should be accessed to get information on this resource

For example, the following value would suggest the `AWS-DisableS3BucketPublicReadWrite` automation document, with a description to show in Jira, to apply in `eu-central-1`, using the account and end-user credentials that is specified in Jira for the AWS account identified in Jira as `MyAccount1`:

```
                        [ { "documentName": "AWS-DisableS3BucketPublicReadWrite",
"description": "This will make the bucket private, resolving the issue.",
"accountName": "MyAccount1",
"region": "eu-central-1" } ]
```

Scripting Field Creation

As an example, the following bash script using curl will link the above-noted resource to an issue and attach a suggested remediation. The values used below assume Jira is at *localhost:2990/jira* with login *admin:admin*, the issue is *PRJ-1*, and the field IDs are 10011 (AWS Config linked resources) and 10010 (suggested remediation). These should be changed to reflect your environment.

1. Set the following to correspond to your environment and issue:

   JIRA_BASE_URL=http://localhost:2990/jira

   JIRA_USER_PASS=admin:admin

   ISSUE_KEY=PRJ-1

2. Set the field ID and edit the JSON record for an AWS Config resource to link.

```
                          CUSTOM_FIELD_ID=customfield_10011
cat > value.json  EOF
    [ { "resourceId": "my-bucket",
        "resourceType": "AWS::S3::Bucket",
        "accountName": "MyAccount1",
        "region": "eu-central-1" } ]
EOF
```

3. Define a helper function to escape the JSON

```
                         json_escape () {
    printf '%s' "$1" | python -c \
      'import json,sys; print(json.dumps(sys.stdin.read()))'
}
```

4. Make the REST call to set the AWS Config Linked Resource field

```
                         curl -v -D- -X PUT  -H "Content-Type: application/json" \
  --data '{ "update": { "'${CUSTOM_FIELD_ID}'": [ {"set": '"$(
     json_escape "$(cat value.json)")"' } ] } }' \
  -u admin:admin ${JIRA_BASE_URL}/rest/api/2/issue/${ISSUE_KEY}
```

5. Set the field ID and edit the JSON record for a suggested remediation to attach.

```
                         CUSTOM_FIELD_ID=customfield_10010
cat > value.json  EOF
    [ { "documentName": "AWS-DisableS3BucketPublicReadWrite",
        "description": "This will make the bucket private, resolving the issue.",
        "accountName": "MyAccount1",
        "region": "eu-central-1" } ]
EOF
```

6. Make the REST call to set the **AWS Systems Manager Automation Suggested Remediations** field .

```
                    curl -v -D- -X PUT  -H "Content-Type: application/json" \
--data '{ "update": { "'${CUSTOM_FIELD_ID}'": [ {"set": '"$(
   json_escape "$(cat value.json)")"' } ] } }' \
-u ${JIRA_USER_PASS} ${JIRA_BASE_URL}/rest/api/2/issue/${ISSUE_KEY}
```

The issue should then show AWS Config for the bucket and a suggested remediation to make it private.

Creating Issues with Suggestions and a Linked AWS Resource from AWS Systems Manager

A Systems Manager Automation Document can automatically create a Jira issue with the fields set to have a linked AWS resource and up to three suggested remediation documents.

To install this automation document, download and extract the JSM Connector Create Remediation Issue Automation and IT Lifecycle Demo.zip that contains two files:

- *JSMConnector-CreateRemediationIssue.ssmdoc.yaml*
- *JSMConnector-function.zip*

Follow these steps

1. Upload the file *JSMConnector-function.zip* to a bucket. The following command replace ${BUCKET} with the appropriate bucket:

   ```
   aws s3 cp JSMConnector-function.zip s3://${BUCKET}/function.zip
   ```

2. Create the Systems Manager Automation Document, called **JSMConnector-CreateRemediationIssue**, with the contents from the file *JSMConnector-CreateRemediationIssue.ssmdoc.yam*l and an attachment *Key=SourceUrl,Values=s3://${BUCKET}/*, using the bucket name from the previous step as ${BUCKET}. The following command replaces ${BUCKET}):

   ```
   aws ssm create-document --name "JSMConnector-CreateRemediationIssue" --content "file://
   JSMConnector-CreateRemediationIssue.ssmdoc.yaml" --document-type "Automation" --
   document-format "YAML" --attachments "Key=SourceUrl,Values=s3://${BUCKET}/"
   ```

Once installed, enter the parameters and run it. Note that it requires many of the same parameters, as described previously to connect to Jira.

You should then see an issue in Jira with AWS Config information and the suggested remediation shown.

Sample Use Case: Automatically Creating Issues for IT Lifecycle Management - Remediating non-compliant public S3 buckets

Once the fields are enabled to an issue and the Systems Manager Automation Document is created, you can set up rules to automatically create Jira issues for common problem categories in AWS. You can also include suggested remediations to make it easy for Jira agents and end users to see problems and fix them.

This demo creates a Config Rule in AWS, which detects public S3 buckets and makes it possible for Jira agents or end users to disable public access directly from Jira.

You will need to set up prerequisites, roles for the automation and lambda to execute, and the Jira password as a secure string in Systems Manager Parameter Store.

To store the Jira password securely in Parameter Store

1. Open the AWS Console and go to **Systems Manager -> Parameter Store**.
2. Choose **Create parameter**.
3. Set the name as **`jira_password`**.
4. Set the type as **`SecureString`**.
5. Set the value as the password for the Jira user to create issues.
6. To save, choose **Create parameter**.

An AWS CloudFormation template is provided to assist setting up the role and configuration rule: **JSMConnector-CreateRemediationIssue-MakePublicBucketsPrivateConfigRule.cfn.yaml**

Install the template, setting the following parameters:

- **JiraURL**: the base URL to your Jira, such that appending */rest/...* after it accesses the REST API
- **JiraUsername**: the username to log in to Jira (with the password specified in *jira_password*)
- **SSMParameterName**: *jira_password* (the parameter containing the Jira password)
- **ProjectKey**: the key of the project (the token before the *-n an issue*), such as *PRJ*.
- **IssueTypeName**: must exactly match the name of the issue type on the project in Jira
- **JiraAwsAccountName**: the name of the AWS Account as configured in the Connector in Jira
- **JiraAwsAccountRegion**: the region where this violating resource is found, e.g. *us-east-1*
- **JiraAwsResourceFieldId**: enter the field ID of the AWS Config Linked Resources field in Jira, such as *customfield_10011*.
- **JiraRemediationsFieldId**: enter the field ID of the **AWS Systems Manager Automation Suggested Remediation** field in Jira, such as *customfield_10010*.

The Config Rule runs automatically within the period specified. To see it in action immediately:

1. Create a public Amazon S3 bucket.
2. Open the Config Rule in AWS Config and choose **Re-evaluate**. The rule and the automation can take a short while to run, but within a few minutes you should see a new issue in Jira with AWS Config information for the bucket, which is in violation and suggests the **DisableS3BucketPublicReadWrite** automation document as a remediation.

Validating Configurations

You are now ready to validate the AWS Service Management Connector for Jira Service Management installation procedures.

AWS Service Catalog

To order an AWS Service Catalog product

1. Log in to your Jira Service Management customer portal as the end user.
2. In the Jira Service Management customer portal, choose **Request AWS product**.

3. Enter **Summary** details.
4. Open the **AWS product request detail** menu and select a product to provision.
5. Fill in the product request details, including product reference name, parameters, and tags.
6. Choose **Create** to submit the Jira Service Management request and provision the AWS Service Catalog product.
7. After the request processes, a message appears indicating that your request was created. When the product is ready to provision, the end user is notified that the product is launching.

To view provisioned products

1. In the Jira Service Management customer portal, go to **Requests** in the upper right corner.
2. Select **My Requests** in the Jira Service Management customer portal view.
3. Select the AWS product you requested.
4. The AWS product details will display, including the status of the product request, product events, and activities.
5. AWS Config information is displayed, if that Connector feature is available. You can expand Configuration Items or Relationships to see more information. Related resources can be loaded by continuing to expand them underneath the Relationships section.
6. Once the product is in the **Available** status, end users can request post-provision operations actions such as **Request update**, **Request termination**, and **Request self-service actions**. These actions render additional product events and activities within the request. Once the product is terminated, the request closes in a resolved state.

AWS Systems Manager Automation

To execute an automation document

1. Log in to your Jira Service Management customer portal as the end user.
2. In the Jira Service Management customer portal, choose **Request AWS automation**.
3. Enter **Summary** details.
4. Open the **AWS automation request detail** menu and select an automation document to execute.
5. Fill in the automation request details, parameters and tags.
6. Choose **Create** to submit the Jira Service Management request and execute the AWS Systems Manager Automation Document.
7. After the request processes, a message appears indicating that your request was created. As the automation executes, the end user is notified of progress.

To view automation executions

1. In the Jira Service Management customer portal, go to **Requests** in the upper right corner.
2. Select **My Requests** in the Jira Service Management customer portal view.
3. Select the AWS automation execution you requested. The AWS automation execution details will display, including the status of the execution, request details, and steps.

AWS Systems Manager OpsCenter

To view OpsItems in Jira Service Management from AWS Systems Manager

1. Log in to your Jira Agent view as an end user.

2. In the Jira Service Management Jira Agent view, choose the Jira project associated to OpsCenter
3. Select Open Issues and select the OpsItem from AWS that you want to view.

To create AWS Systems Manager OpsItems in Jira Service Management

1. Log in to your Jira Agent view as an end user.
2. In the Jira Service Management Jira Agent view, select Create.
3. In the **Create Issue** field input the following details:
   - Project: Auto-populated
   - Issue Type: Select AWS OpsItem if you have multiple issue types
   - Summary: Input Summary Details
   - Description: Input Description
   - Priority: Select the appropriate Priority (default value is Low)
   - Severity: Select the appropriate Severity (required for AWS OpsItem)
   - Category: Select the appropriate Category (required for AWS OpsItem)
   - Region: Select the appropriate AWS Region (required for AWS OpsItem)
4. Select **Create**.

   Note
   The newly created OpsItem from Jira Service Management will display in the AWS account view of OpsItem on the next sync between AWS and Jira Service Management.

To update AWS Systems Manager OpsItems in Jira Service Management

1. Log in to your Jira Agent view as an end user.
2. In the Jira Service Management Jira Agent view, choose the Jira project associated to OpsCenter.
3. Select **Open Issues** and select the **OpsItem** from AWS that you want to update.
4. Select **Edit Issue**.
5. Update fields available such as Summary, Description, Priority, Severity, Category. The **Resolved** button in the OpsItem issue is also available to select upon resolution.

   Note
   Updates to OpsItem fields from Jira Service Management will display in the AWS account view of OpsItem on the next sync between AWS and Jira Service Management.

To view AWS related resources in AWS Systems Manager OpsItems through Jira Service Management

1. Log in to your Jira Agent view as an end user.
2. In the Jira Service Management Jira Agent view, choose the Jira project associated to OpsCenter.
3. Select **Open Issues** and select the **OpsItem** from the OpsItem from AWS.
4. Go to AWS related resource section of the OpsItem selected. This section displays the related resource details.

To execute runbooks on AWS Systems Manager OpsItems through Jira Service Management

1. Log in to your Jira Agent view as an end user.
2. In the Jira Service Management Jira Agent view, choose the Jira project associated to OpsCenter.
3. Select **Open Issues** and select the **OpsItem**.

4. Go to the OpsItem section AWS Runbooks. OpsItem that contain associated runbooks display a list of automation documents available highlighted by a star shaped symbol.

   - Select Execute on the desired runbook. An Execute Runbook from OpsItem screen displays.
   - Enter the workflow parameter details associated to the runbook. **Note**: The runbook will not execute successfully without the correct parameter inputs.
   - Enter metadata tags details if applicable.
   - **Select Create**. An Execute AWS Systems Manager Automation Request issue is generated and provides the execution status.

   OpsItems without associated runbooks are still able to run automated documents.

To run automated documents not associated with runbooks

1. In the OpsItem, select **Show All Runbooks**. A list on AWS runbooks is displayed.
2. To narrow the list of runbooks available, enter details into the search bar above the first listed runbook.
3. Select **Execute** on the desired runbook. An Execute Runbook from OpsItem screen displays.
4. Enter the workflow parameter details associated to the runbook. Note: The runbook will not execute successfully without the correct parameter inputs.
5. Enter metadata tags details if applicable.
6. Select **Create**. An Execute AWS Systems Manager Automation Request issue is generated and provides the execution status.

Jira Additional Administrator Features

The following sections describe approvals and access controls that are available in Jira.

Approvals

The approval agent has access to a screen with the options to approve or reject the product request. For a rejection, the agent can add a comment explaining why they are rejecting the request. The requester is able to see the status of the request, such as *Waiting for Approval*, *Scheduled*, *Launching*, or *Available*.

Changes to approver group members do not impact approvers identified for pre-existing issues but do affect whether the approval is permitted. Approver users are assigned to the issue at the time the issue is created, and only these users are given the option to approve the request. The approver user must still be a member of the group when the approval is made, otherwise the approval is rejected.

As with AWS Service Catalog, all post-provision actions, including termination, are pre-approved for the user or group who is approved to provision it.

Access Controls

Access controls can be set on portfolios, as described earlier in this guide. Those access controls are in addition to the per-project enablement: users must have access to an AWS Connector-enabled project and belong to the groups enabled for a portfolio in order to provision products in that portfolio.

Document History

The following table describes important additions to the AWS Service Catalog documentation.

Feature	Description	Release Date
Latest version of Connector for ServiceNow	To learn about the updates to the Connector for ServiceNow, see AWS Service Management Connector for ServiceNow (p. 90).	September 24, 2020
AWS Service Quotas	To learn about how AWS Service Catalog works with AWS Service Quotas, see AWS Service Catalog default service quotas (p. 4).	March 24, 2020
Getting Started Library	To learn about the library of well-architected product templates offered by AWS Service Catalog, see Getting Started Library (p. 17).	March 10, 2020
Version guidance	To learn about product version guidance, see Version Guidance (p. 46).	December 17, 2019
Connector for Jira Service Desk	To begin using the Connector for Jira Service Desk, see AWS Service Management Connector for Jira Service Management (p. 128).	November 21, 2019
New version of Connector for ServiceNow	To learn about the updates to the Connector for ServiceNow, see AWS Service Management Connector for ServiceNow (p. 90).	November 18, 2019
New security chapter	To learn about security in AWS Service Catalog, see Security in AWS Service Catalog (p. 19).	October 31, 2019
Changing provisioned product owner	To learn about how to change the owner of provisioned products, see Changing Provisioned Product Owner (p. 66).	October 31, 2019
New resource update constraint	To learn about how to use the `RESOURCE_UPDATE` constraint to update tags in provisioned products, see AWS Service Catalog Tag Update Constraints (p. 49).	April 17, 2019
Connector for ServiceNow	To begin using the Connector for ServiceNow, see AWS Service Management Connector for ServiceNow (p. 90).	March 19, 2019

Feature	Description	Release Date
Support for AWS CloudFormation StackSets	To begin using AWS CloudFormation StackSets, see Using AWS CloudFormation StackSets (p. 62).	November 14, 2018
Self-service actions	To begin using self-service actions, see AWS Service Catalog Service Actions (p. 53).	October 17, 2018
CloudWatch metrics	To learn about CloudWatch metrics, see AWS Service Catalog CloudWatch Metrics (p. 77).	September 26, 2018
Support for TagOptions	To manage tags, see AWS Service Catalog TagOption Library (p. 72).	June 28, 2017
Importing a portfolio	To import a portfolio shared from another AWS account, see Importing a Portfolio (p. 38).	February 16, 2016
Updates to permissions information	To grant access to the end user console view, see Console access for end users (p. 22).	February 16, 2016
Initial release	This is the initial release of the *AWS Service Catalog Administrator Guide*.	July 9, 2015