# 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

Limits ......................................................................................................................................................... 4

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 .......................................................................................................................................... 10
  Step 1: Download the Template .............................................................................................................. 10
  Template Download ................................................................................................................................. 10
  Template Overview .................................................................................................................................. 10
  Step 2: Create a Key Pair ......................................................................................................................... 13
  Step 3: Create a Portfolio ......................................................................................................................... 13
  Step 4: Create a Product ............................................................................................................................ 14
  Step 5: Add a Template Constraint ......................................................................................................... 14
  Step 6: Add a Launch Constraint .............................................................................................................. 15
  Step 7: Grant End Users Access to the Portfolio ...................................................................................... 16
  Step 8: Test the End User Experience ..................................................................................................... 16

Authentication and Access Control .............................................................................................................. 18
  Predefined AWS Managed Policies ........................................................................................................... 18
    Deprecated Policies .................................................................................................................................. 19
  Console Access for End Users .................................................................................................................... 19
  Product Access for End Users .................................................................................................................... 20
  Example Policies ....................................................................................................................................... 20
    Example: Full Admin Access to Provisioned Products ............................................................................ 20
    Example: End-user Access to Provisioned Products ............................................................................... 21
    Example: Partial Admin Access to Provisioned Products ...................................................................... 22

Managing Catalogs ........................................................................................................................................ 23
  Managing Portfolios ................................................................................................................................. 23
    Creating, Viewing, and Deleting Portfolios .......................................................................................... 23
    Viewing Portfolio Details ....................................................................................................................... 24
    Creating and Deleting Portfolios .......................................................................................................... 24
    Adding Products .................................................................................................................................... 24
    Adding Constraints ............................................................................................................................... 26
    Granting Access to Users ....................................................................................................................... 26

Managing Products ....................................................................................................................................... 27
  Viewing the Products Page ....................................................................................................................... 27
  Creating Products .................................................................................................................................... 28
  Adding Products to Portfolios ................................................................................................................... 28
  Updating Products .................................................................................................................................... 29
  Deleting Products .................................................................................................................................... 29

Using Constraints ......................................................................................................................................... 29
  Launch Constraints ................................................................................................................................... 30
  Notification Constraints ............................................................................................................................ 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template Constraints</td>
<td>33</td>
</tr>
<tr>
<td>Adding AWS Marketplace Products to Your Portfolio</td>
<td>41</td>
</tr>
<tr>
<td>Managing AWS Marketplace Products Using AWS Service Catalog</td>
<td>42</td>
</tr>
<tr>
<td>Managing and Adding AWS Marketplace Products Manually</td>
<td>42</td>
</tr>
<tr>
<td>Portfolio Sharing</td>
<td>46</td>
</tr>
<tr>
<td>Relationship Between Shared and Imported Portfolios</td>
<td>46</td>
</tr>
<tr>
<td>Sharing a Portfolio</td>
<td>48</td>
</tr>
<tr>
<td>Importing a Portfolio</td>
<td>48</td>
</tr>
<tr>
<td>Managing Provisioned Products</td>
<td>49</td>
</tr>
<tr>
<td>Managing All Provisioned Products as Administrator</td>
<td>49</td>
</tr>
<tr>
<td>Tutorial: Identifying User Resource Allocation</td>
<td>49</td>
</tr>
<tr>
<td>TagOption Library</td>
<td>53</td>
</tr>
<tr>
<td>Launching a Product with TagOptions</td>
<td>54</td>
</tr>
<tr>
<td>Example 1: A Unique TagOption Key</td>
<td>54</td>
</tr>
<tr>
<td>Example 2: A Set of TagOptions with the Same Key on a Portfolio</td>
<td>54</td>
</tr>
<tr>
<td>Example 3: A Set of TagOptions with the Same Key on Both the Portfolio and a Product in that Portfolio</td>
<td>55</td>
</tr>
<tr>
<td>Example 4: Multiple TagOptions with the Same Key and Conflicting Values</td>
<td>56</td>
</tr>
<tr>
<td>Managing TagOptions</td>
<td>56</td>
</tr>
<tr>
<td>Document History</td>
<td>58</td>
</tr>
</tbody>
</table>
What Is AWS Service Catalog?

AWS Service Catalog enables organizations to create and manage catalogs of IT services that are approved for use on AWS. These IT services can include everything from virtual machine images, servers, software, and databases 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 AWS 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 AWS 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 AWS 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 Limits

Your AWS account has the following limits related to AWS Service Catalog. To request a limit increase, use the Service Limit Increase form.

Regional Limits

- Portfolios: 25
- Products: 100

Portfolio Limits

- Users, groups, and roles per portfolio: 25
- Products per portfolio: 25
- Tags per portfolio: 20

Product Limits

- Product versions per product: 50
- Tags per product: 20

Provisioned Product Limits

- Tags per provisioned product: 50
Constraint Limits

- Constraints per product per portfolio: 25
Setting Up for 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://aws.amazon.com/, and then choose Create an AWS Account.

   Note
   This might be unavailable in your browser if you previously signed into the AWS Management Console. In that case, choose Sign in to a different account, and then choose Create a new AWS account.

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using 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.

- Authentication and Access Control for AWS Service Catalog (p. 18)
- 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. For Create Your Own Policy, choose Select.
   b. For Policy Name, type ServiceCatalogAdmin-AdditionalPermissions.
   c. Copy the following example policy and paste it in Policy Document:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateKeyPair",
        "iam:AddRoleToInstanceProfile",
        "iam:AddUserToGroup",
        "iam:AttachGroupPolicy",
        "iam:AddAccessKey",
        "iam:CreateAccessKey",
        "iam:CreateGroup",
        "iam:CreateInstanceProfile",
        "iam:CreateLoginProfile",
        "iam:CreateRole",
        "iam:CreateUser",
        "iam:Get*",
        "iam:List*",
        "iam:GetGroupPolicy",
        "iam:GetRolePolicy",
        "iam:GetUserPolicy",
        "iam:GetInstanceProfilePolicy",
        "iam:GetLoginProfilePolicy",
        "iam:GetRolePolicy",
        "iam:GetUserPolicy",
        "iam:List*",
        "iam:PutRolePolicy",
        "iam:UpdateAssumeRolePolicy"
      ],
      "Resource": [
        "*"
      ],
    }
  ]
}
```
d. (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.

e. 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 ServiceCatalogEndUserAccess policy to an IAM group. For more information, see Predefined AWS Managed Policies (p. 18).

To allow an end user to launch a product, you must grant access to the ProvisionProduct action. You can do so using an inline policy, as shown in the following procedure.

To grant permissions to an end user

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

2. In the navigation pane, choose Policies.

3. Choose Create policy and do the following:

   a. For Create Your Own Policy, choose Select.

   b. For Policy Name, type ServiceCatalogEndusers-AdditionalPermissions.

   c. Copy the following example policy and paste it in Policy Document:

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

   d. Choose Create Policy.

4. In the navigation pane, choose Groups.

5. 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 **ServiceCatalog** to filter the policy list.
c. Select the checkboxes for the **ServiceCatalogEndUserAccess** and **ServiceCatalogEndusers-AdditionalPermissions** policies, and then choose **Next Step**.
d. On the **Review page**, choose **Create Group**.

6. In the navigation pane, choose **Users**.

7. 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: 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 for AWS Service Catalog (p. 6).

Tasks

- Step 1: Download the AWS CloudFormation Template (p. 10)
- Step 2: Create a Key Pair (p. 13)
- Step 3: Create an AWS Service Catalog Portfolio (p. 13)
- Step 4: Create an AWS Service Catalog Product (p. 14)
- Step 5: Add a Template Constraint to Limit Instance Size (p. 14)
- Step 6: Add a Launch Constraint to Assign an IAM Role (p. 15)
- Step 7: Grant End Users Access to the Portfolio (p. 16)
- Step 8: Test the End User Experience (p. 16)

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:

```json
{
  "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."
  },
  "InstanceType": {
    "Description" : "EC2 instance type."
  },
  "SSHLocation": {
    "Description" : "The IP address range that can SSH to the EC2 instance."
  }
},

"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-35072384" },
    "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. 29).

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:

```json
  "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. 13). 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 **Get started** to start the wizard for configuring a portfolio. Otherwise, choose **Create portfolio**.
3. Type the following values:
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 the name **Engineering Tools** to open the portfolio details page, 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.**
   - **Provided by** – **IT**
   - **Vendor** – (blank)
4. 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.**
5. 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**
6. On the **Review** page, choose **Create**.

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. 29).

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:

   ```json
   {  
       "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 will add a launch constraint to the Linux Desktop product so that AWS Service Catalog can use the AWS resources that are part of the product's AWS CloudFormation template. This launch constraint will enable the end user to launch the product and, after it is launched, manage it as a provisioned product. For more information, see AWS Service Catalog Launch Constraints (p. 30).

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. For Create Your Own Policy, choose Select.
   b. For Policy Name, type linuxDesktopPolicy.
   c. Copy the following example policy and paste it in Policy Document:

   ```json
   {  
       "Version": "2012-10-17",  
       "Statement": [  
           {  
               "Effect": "Allow",  
               "Action": [  
               
               ]  
           }  
       ]  
   }
   ```
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, expand the Users, groups and roles section.
2. Choose Add user, group or role.
3. On the Groups tab, select the checkbox for the IAM group for the endusers.

Step 8: Test the End User Experience

To verify that 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 account-specific URL. To find this URL, open the IAM console, 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.
2. In the menu bar, choose the region in which you created the Engineering Tools portfolio.
3. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/ and select Service Catalog, Dashboard to see the following:
   - **Products** – The products that the user can use.
   - **Provisioned products** – The provisioned products that the user has launched.

To verify that 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 for configuring your product.
3. On the Product version page, for Name, type Linux-Desktop.
4. In the Version table, choose v1.0.
5. Choose Next.
6. On the Parameters page, type 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. 13).
   - **CIDR range** – Type a valid CIDR range for the IP address from which you will connect to the instance. This can be the default value (0.0.0.0/0) to allow access from any IP address, your IP address followed by /32 to restrict access to your IP address only, or something in between.
7. On the Review page, review the information that you typed, and then 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 is launched, the status is Available.
Authentication and Access Control for AWS Service Catalog

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 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.

Contents
- Predefined AWS Managed Policies (p. 18)
- Console Access for End Users (p. 19)
- Product Access for End Users (p. 20)
- Example Policies for Managing Provisioned Products (p. 20)

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.
- ServiceCatalogAdminReadOnlyAccess — 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.
- ServiceCatalogEndUserAccess — 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 Attach Policy.
5. On the Attach Policy page, select the check box next to the managed policy for AWS Service Catalog, and then choose Attach Policy.
6. (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.
Deprecation Policies

The following managed policies are deprecated:

- `ServiceCatalogAdminFullAccess` — Use `AWSServiceCatalogAdminFullAccess` instead.
- `ServiceCatalogEndUserFullAccess` — Use `AWSServiceCatalogEndUserFullAccess` 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, as described in the procedure *To attach a policy to an IAM user* (p. 18).
   c. On the *Permissions* tab, next to `ServiceCatalogAdminFullAccess`, choose *Detach Policy*. When prompted for confirmation, choose *Detach*.
5. Repeat the process for `ServiceCatalogEndUserFullAccess`.

Console 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.

The `ServiceCatalogEndUserFullAccess` and `ServiceCatalogEndUserAccess` policies grant access to the AWS Service Catalog end user console view. When a user who has either of these policies chooses `Service Catalog` in the AWS Management Console, the end user console view displays.

If you apply the `ServiceCatalogEndUserAccess` 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. 30).

If you grant users the following IAM permissions, which are meant for AWS Service Catalog administrators, the administrator console view displays instead:

- `catalog-admin:ListPortfolios`
- `catalog-admin:SearchListings`

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. 30).

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, Scan provisioned products, 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. 20)
- Example: End-user Access to Provisioned Products (p. 21)
- Example: Partial Admin Access to Provisioned Products (p. 22)

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.

```json
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":
```
This policy is functionally equivalent to the following policy:

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

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"
        }
      }
    }
  ]
}
```
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 [Authentication and Access Control for AWS Service Catalog](p. 18).

**Tasks**
- Managing Portfolios (p. 23)
- Managing Products (p. 27)
- Using AWS Service Catalog Constraints (p. 29)
- Adding AWS Marketplace Products to Your Portfolio (p. 41)
- Portfolio Sharing (p. 46)

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. 23)
- Viewing Portfolio Details (p. 24)
- Creating and Deleting Portfolios (p. 24)
- Adding Products (p. 24)
- Adding Constraints (p. 26)
- Granting Access to Users (p. 26)

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**

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**

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. Select the portfolio by clicking the corresponding radio button or anywhere on the listing except on the portfolio title.
3. Choose **Delete portfolio**.
4. Choose **Continue**.

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. 29).

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.
   - 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. 34).
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 GuideUsing IAM.
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 Authentication and Access Control for AWS Service Catalog (p. 18).

**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**.

### 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. 27)
- Creating Products (p. 28)
- Adding Products to Portfolios (p. 28)
- Updating Products (p. 29)
- Deleting Products (p. 29)

### 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

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 make 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.

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.

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.

Choose Save.

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.

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.

Using AWS Service Catalog Constraints

To control which rules are applied when the end user launches a product from a specific portfolio, you apply constraints. You apply constraints to products from the portfolio details page. Constraints are active as soon as you create them and apply to all current versions of a product that are not already launched when you create the constraint.
AWS Service Catalog Administrator Guide

Launch Constraints

Constraints

- AWS Service Catalog Launch Constraints (p. 30)
- AWS Service Catalog Notification Constraints (p. 32)
- AWS Service Catalog Template Constraints (p. 33)

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. 15).

Launch constraints are associated with a product within the portfolio (product-portfolio association), not 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, the AWS services used by the products, and AWS Service Catalog. By using a launch role, you can instead limit the end users' permissions to the minimum that they require for that product. For more information about end user permissions, see Authentication and Access Control for AWS Service Catalog (p. 18).

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:

- AWS CloudFormation
- Services used in the AWS CloudFormation template for the product
- Read access to the AWS 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:

   ```json
   
   "Version":"2012-10-17",
   "Statement":[
     
     "Effect":"Allow",
     "Action":[
       "catalog-user:*",
       "cloudformation:CreateStack",
       "cloudformation:DeleteStack",
       "cloudformation:DescribeStackEvents",
       "cloudformation:DescribeStacks",
       "cloudformation:GetTemplateSummary",
       "cloudformation:SetStackPolicy",
       "cloudformation:ValidateTemplate",
       "cloudformation:UpdateStack",
       "s3:GetObject"
     ],
     "Resource":"
   ]
   
   "rds:*"
   }
   ```

5. Add a line to the policy for each additional service that the product uses. For example, to add permission for Amazon Relational Database Service (Amazon RDS), type 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**

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

**To assign the role to a product**

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 **Launch**. Choose **Continue**.
5. For **IAM role**, choose the launch role. Choose **Submit**.

**Verify That the Launch Constraint Is Applied**

Verify that AWS Service Catalog uses the role to launch the product and that the provisioned product is created successfully by launching 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/v2/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**.
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. 34).

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" : [
```

34
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.

You can use the following rule-specific intrinsic functions to define rule conditions and assertions:

- `Fn::And`
- `Fn::Contains`
- `Fn::EachMemberEquals`
- `Fn::EachMemberIn`
- `Fn::Equals`
- `Fn::If`
- `Fn::Not`
- `Fn::Or`
- `Fn::RefAll`
- `Fn::ValueOf`
- `Fn::ValueOfAll`

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.

```json
"Rules" : {
    "testInstanceType" : {
        "RuleCondition" : {"Fn::Equals":["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":["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 Rule Functions

In the condition or assertions of a rule, you can use intrinsic functions, such as `Fn::Equals`, `Fn::Not`, and `Fn::RefAll`. The condition property determines if AWS CloudFormation applies the assertions. If the condition evaluates to `true`, AWS CloudFormation evaluates the assertions to verify whether a parameter value is valid when a provisioned product is created or updated. If a parameter value is invalid, AWS CloudFormation does not create or update the stack. If the condition evaluates to `false`, AWS CloudFormation doesn't check the parameter value and proceeds with the stack operation.

Functions

- `Fn::And (p. 36)`
- `Fn::Contains (p. 36)`
- `Fn::EachMemberEquals (p. 37)`
- `Fn::EachMemberIn (p. 37)`
- `Fn::Equals (p. 38)`
- `Fn::Not (p. 38)`
- `Fn::Or (p. 39)`
- `Fn::RefAll (p. 39)`
- `Fn::ValueOf (p. 40)`
- `Fn::ValueOfAll (p. 40)`
- `Supported Functions (p. 41)`
- `Supported Attributes (p. 41)`

**Fn::And**

Returns `true` if all the specified conditions evaluate to `true`; returns `false` if any one of the conditions evaluates to `false`. `Fn::And` acts as an AND operator. The minimum number of conditions that you can include is two, and the maximum is ten.

Declaration

```
"Fn::And" : [{condition}, {...}]
```

Parameters

`condition`

A rule-specific intrinsic function that evaluates to `true` or `false`.

Example

The following example evaluates to `true` if the referenced security group name is equal to `sg-mysggroup` and if the `InstanceType` parameter value is either `m1.large` or `m1.small`:

```
"Fn::And" : [
  {"Fn::Equals" : ["sg-mysggroup", {"Ref" : "ASecurityGroup"}]},
  {"Fn::Contains" : ["m1.large", "m1.small"], {"Ref" : "InstanceType"}]}
]
```

**Fn::Contains**

Returns `true` if a specified string matches at least one value in a list of strings.
Declaration

"Fn::Contains" : [[list_of_strings], string]

Parameters

list_of_strings

A list of strings, such as "A", "B", "C".

string

A string, such as "A", that you want to compare against a list of strings.

Example

The following function evaluates to true if the InstanceType parameter value is contained in the list (m1.large or m1.small):

"Fn::Contains" : [
    ["m1.large", "m1.small"], {"Ref" : "InstanceType"}
]

Fn::EachMemberEquals

Returns true if a specified string matches all values in a list.

Declaration

"Fn::EachMemberEquals" : [[list_of_strings], string]

Parameters

list_of_strings

A list of strings, such as "A", "B", "C".

string

A string, such as "A", that you want to compare against a list of strings.

Example

The following function returns true if the Department tag for all parameters of type AWS::EC2::VPC::Id have a value of IT:

"Fn::EachMemberEquals" : [
    {"Fn::ValueOfAll" : ["AWS::EC2::VPC::Id", "Tags.Department"], "IT"}
]

Fn::EachMemberIn

Returns true if each member in a list of strings matches at least one value in a second list of strings.

Declaration

"Fn::EachMemberIn" : [[strings_to_check], strings_to_match]
Parameters

strings_to_check

A list of strings, such as "A", "B", "C". AWS CloudFormation checks whether each member in the strings_to_check parameter is in the strings_to_match parameter.

strings_to_match

A list of strings, such as "A", "B", "C". Each member in the strings_to_match parameter is compared against the members of the strings_to_check parameter.

Example

The following function checks whether users specify a subnet that is in a valid virtual private cloud (VPC). The VPC must be in the account and the region in which users are working with the stack. The function applies to all parameters of type AWS::EC2::Subnet::Id.

```
"Fn::EachMemberIn" : [
   {"Fn::ValueOfAll" : ["AWS::EC2::Subnet::Id", "VpcId"]}, {"Fn::RefAll" : "AWS::EC2::VPC::Id"}
]
```

Fn::Equals

Compares two values to determine whether they are equal. Returns true if the two values are equal and false if they aren't.

Declaration

```
"Fn::Equals" : ["value_1", "value_2"]
```

Parameters

value

A value of any type that you want to compare with another value.

Example

The following example evaluates to true if the value for the EnvironmentType parameter is equal to prod:

```
"Fn::Equals" : [{"Ref" : "EnvironmentType"}, "prod"]
```

Fn::Not

Returns true for a condition that evaluates to false, and returns false for a condition that evaluates to true. Fn::Not acts as a NOT operator.

Declaration

```
"Fn::Not" : [{condition}]
```
Parameters

condition

A rule-specific intrinsic function that evaluates to true or false.

Example

The following example evaluates to true if the value for the EnvironmentType parameter is not equal to prod:

```
"Fn::Not" : [{"Fn::Equals" : [{"Ref" : "EnvironmentType"}, "prod"]}
```

Fn::Or

Returns true if any one of the specified conditions evaluates to true; returns false if all of the conditions evaluate to false. Fn::Or acts as an OR operator. The minimum number of conditions that you can include is two, and the maximum is ten.

Declaration

```
"Fn::Or" : [{ condition }, {...}]
```

Parameters

condition

A rule-specific intrinsic function that evaluates to true or false.

Example

The following example evaluates to true if the referenced security group name is equal to sg-mysggroup or if the InstanceType parameter value is either m1.large or m1.small:

```
"Fn::Or" : [
    {"Fn::Equals" : ["sg-mysggroup", {"Ref" : "ASecurityGroup"}]},
    {"Fn::Contains" : [["m1.large", "m1.small"], {"Ref" : "InstanceType"}]}
]
```

Fn::RefAll

Returns all values for a specified parameter type.

Declaration

```
"Fn::RefAll" : "parameter_type"
```

Parameters

parameter_type

An AWS-specific parameter type, such as AWS::EC2::SecurityGroup::Id or AWS::EC2::VPC::Id. For more information, see Parameters in the AWS CloudFormation User Guide.
Example

The following function returns a list of all VPC IDs for the region and AWS account in which the stack is being created or updated:

```
"Fn::RefAll" : "AWS::EC2::VPC::Id"
```

**Fn::ValueOf**

Returns an attribute value or list of values for a specific parameter and attribute.

**Declaration**

```
"Fn::ValueOf" : [ "parameter_logical_id", "attribute" ]
```

**Parameters**

- **attribute**
  
  The name of an attribute from which you want to retrieve a value. For more information about attributes, see Supported Attributes (p. 41).

- **parameter_logical_id**
  
  The name of a parameter for which you want to retrieve attribute values. The parameter must be declared in the Parameters section of the template.

**Examples**

The following example returns the value of the Department tag for the VPC that is specified by the ElbVpc parameter:

```
"Fn::ValueOf" : ["ElbVpc", "Tags.Department"]
```

If you specify multiple values for a parameter, the Fn::ValueOf function can return a list. For example, you can specify multiple subnets and get a list of Availability Zones where each member is the Availability Zone of a particular subnet:

```
"Fn::ValueOf" : ["ListOfElbSubnets", "AvailabilityZone"]
```

**Fn::ValueOfAll**

Returns a list of all attribute values for a given parameter type and attribute.

**Declaration**

```
"Fn::ValueOfAll" : ["parameter_type", "attribute"]
```

**Parameters**

- **attribute**
  
  The name of an attribute from which you want to retrieve a value. For more information about attributes, see Supported Attributes (p. 41).

- **parameter_type**
  
  An AWS-specific parameter type, such as AWS::EC2::SecurityGroup::Id or AWS::EC2::VPC::Id. For more information, see Parameters in the AWS CloudFormation User Guide.
Example

In the following example, the `Fn::ValueOfAll` function returns a list of values, where each member is the `Department` tag value for VPCs with that tag:

```
"Fn::ValueOfAll" : ["AWS::EC2::VPC::Id", "Tags.Department"]
```

Supported Functions

You cannot use another function within the `Fn::ValueOf` and `Fn::ValueOfAll` functions. However, you can use the following functions within all other rule-specific intrinsic functions:

- `Ref`
- Other rule-specific intrinsic functions

Supported Attributes

The following list describes the attribute values that you can retrieve for specific resources and parameter types:

The `AWS::EC2::VPC::Id` parameter type or VPC IDs

- `DefaultNetworkAcl`
- `DefaultSecurityGroup`
- `Tags.tag_key`

The `AWS::EC2::Subnet::Id` parameter type or subnet IDs

- `AvailabilityZone`
- `Tags.tag_key`
- `Vpcid`

The `AWS::EC2::SecurityGroup::Id` parameter type or security group IDs

- `Tags.tag_key`

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.

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

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:

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

**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.

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.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:

```
"AWSRegionArch2AMI" : {
  "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"  }
},
```
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.

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.
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.

**Portfolio Sharing**

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 their AWS accounts.

When you share a portfolio, 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. 26).

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. This launch role is used even if the end user belongs to a different AWS account than the one that owns the launch role. For more information about launch constraints and launch roles, see AWS Service Catalog Launch Constraints (p. 30). 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.

**Note**

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

**Relationship Between Shared and Imported Portfolios**

The following 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.
<table>
<thead>
<tr>
<th>Element of Shared Portfolio</th>
<th>Relationship to Importe Portfolio</th>
<th>Recipient Administrator Can</th>
<th>Recipient Administrator Cannot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Products and product versions</td>
<td>Inherited.</td>
<td>Add imported products to local portfolios. Products stay in sync with shared portfolio.</td>
<td>Upload or add products to the imported portfolio or remove products from the imported portfolio.</td>
</tr>
<tr>
<td></td>
<td>If the portfolio creator adds products to or removes products from the shared portfolio, the change propagates to the imported portfolio.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Launch constraints</td>
<td>Inherited.</td>
<td>In a local portfolio, the administrator can override the imported launch constraint by applying a different one to the product.</td>
<td>Add launch constraints to or remove launch constraints from the imported portfolio.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Template constraints</td>
<td>Inherited.</td>
<td>In a local portfolio, the administrator can add template constraints that take effect in addition to the imported constraints.</td>
<td>Remove the imported template constraints.</td>
</tr>
<tr>
<td></td>
<td>If the portfolio creator adds a template constraint to or removes a template constraint from a shared product, the change propagates to all imported instances of the product.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IAM users, groups, and roles</td>
<td>Not inherited.</td>
<td>Add IAM users, groups, and roles that are in</td>
<td>Not applicable.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 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 that administrator’s AWS account.

To complete these steps, you must obtain the account ID of the target AWS account. The ID is provided on the **My Account** page in the AWS Management Console of the target account.

**To share a portfolio**

2. On the **Portfolios** page, select the portfolio that you want to share, and choose **Share Portfolio**.
3. In the **Enter AWS account ID** window, type the account ID of the AWS account that you are sharing with. Then, choose **Share**. If sharing succeeds, a message on the **Portfolios** page confirms that the portfolio is linked with the target account. It also provides a URL that the recipient administrator must use to import the portfolio.
4. 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 for importing the portfolio from the administrator.

Open the URL, and on the **Import Portfolio** page, choose **Import**. The **Portfolios** page displays, and the portfolio is shown in the **Imported Portfolios** table.
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.

Contents

- Managing All Provisioned Products as Administrator (p. 49)
- Tutorial: Identifying User Resource Allocation (p. 49)

Managing All Provisioned Products as Administrator

To manage all provisioned products for the account, you will need ServiceCatalogAdminFullAccess or equivalent access to the provisioned product write operations. For more information, see Authentication and Access Control for AWS Service Catalog (p. 18).

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.

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 ServiceCatalogAdminFullAccess or equivalent access to the provisioned product write operations. For more information, see Authentication and Access Control for AWS Service Catalog (p. 18).

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".

```json
{
    "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":
```
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.
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

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 create, remove, and edit TagOptions, and associate and disassociate TagOptions with a portfolio or product.

To create a TagOption (console)

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose TagOption library.
3. Either type a new value for one of the key groupings or choose Create new TagOption and type a new key and value.

After the new TagOption has been created, it's grouped by key-value pair and sorted alphabetically. You can delete a newly created TagOption by choosing Delete from library. This deletion feature is available only for newly created TagOptions. It's designed for quick management of mistyped TagOptions.

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

To associate a TagOption with a portfolio or product (console)

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose TagOption, select a TagOption, and select the portfolio or product to associate the TagOption with.

Alternatively, from a portfolio or product detail page, choose Add TagOption and select the TagOption to associate the TagOption with.
3. Choose Save.
To associate a TagOption with a portfolio or product using the AWS Service Catalog API, see
AssociateTagOptionWithResource.

To remove (disassociate) a TagOption from a portfolio or product (console)

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose TagOption, select a TagOption, and open the Detail page.
3. Select the small x to the right of the portfolio or product from which you want to remove the association.

   Alternatively, from a portfolio or product Detail page, choose the small x to the right of the TagOption that you want to remove.

To remove a TagOption using the AWS Service Catalog API, see DisassociateTagOptionFromResource.

To edit a TagOption

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose TagOption library, select a TagOption, and edit the key or value.
3. Choose Save.
## Document History

The following table describes important additions to the AWS Service Catalog documentation.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for TagOptions</td>
<td>To manage tags, see <a href="https://docs.aws.amazon.com/servicecatalog/latest/adminguide/tag-options.html">AWS Service Catalog TagOption Library (p. 53)</a></td>
<td>June 28, 2017</td>
</tr>
<tr>
<td>Importing a portfolio</td>
<td>To import a portfolio shared from another AWS account, see <a href="https://docs.aws.amazon.com/servicecatalog/latest/adminguide/import-portfolios.html">Importing a Portfolio (p. 48)</a></td>
<td>February 16, 2016</td>
</tr>
<tr>
<td>Updates to permissions information</td>
<td>To grant access to the end user console view, see <a href="https://docs.aws.amazon.com/servicecatalog/latest/adminguide/console-access-end-users.html">Console Access for End Users (p. 19)</a></td>
<td>February 16, 2016</td>
</tr>
<tr>
<td>Initial release</td>
<td>This is the initial release of the <a href="https://docs.aws.amazon.com/servicecatalog/latest/adminguide/">AWS Service Catalog Administrator Guide</a></td>
<td>July 9, 2015</td>
</tr>
</tbody>
</table>