AWS Service Catalog

Administrator Guide

AWS Service Catalog: Administrator Guide

Table of Contents

What Is AWS Service Catalog?

AWS Service Catalog allows 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, while enabling users to quickly deploy only the approved IT services they need.

- **Self-service discovery and launch**

  Users browse listings of services or applications that they have access to, locate the service or application that they want to use, and launch it all on their own. The system administrator can restrict where the product can be launched, the type of instance that can be used, and many other configuration options.
- **Fine-grain access controls and configuration**

  Administrators assemble portfolios of services and applications from their catalog, add rules that apply to a certain group of users (such as the tags that can be assigned to resources they launch), and then grant access to the portfolio through AWS Identity and Access Management (IAM) users and groups.
- **Extensibility and version control**

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

For more service highlights, see the AWS Service Catalog detail page.

Concepts

Understanding the basic components of AWS Service Catalog will help you get the most out of this service.

Topics

AWS Service Catalog Users

AWS Service Catalog users might be either of the following types, depending on the level of permissions that they have:

- **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 a launch context 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. End users may be granted permission to launch and manage all of the resources required by the products they use or only permission to service features.

Portfolio

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. 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.

Product

A product is an IT service that you want to make available for deployment on AWS. A product can comprise one or more AWS resource, 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 your products by importing AWS CloudFormation templates. These templates define the AWS resources required for the product, the relationships between resources, and the parameters that the end user can plug in when they launch the product to configure security groups, create key pairs, and perform other customizations.

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

Stack

AWS CloudFormation stacks make it easier to manage the lifecycle of your product by allowing you to provision, tag, update, and terminate your product instance as a single unit. An AWS CloudFormation stack includes an AWS CloudFormation template and its associated collection of resources. When an end user launches a product, the instance of the product that is provisioned by AWS Service Catalog is a stack of resources necessary to run the product.

Setting Up

To follow the tutorials in this guide, you will need to set up an account and obtain security credentials. This topic walks you through the setup process.

Topics

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 http://aws.amazon.com/, and then click **Sign Up**.
2. Follow the on-screen 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 http://aws.amazon.com/ and choosing **My Account**, **AWS Management Console**.

Managing Users and Groups in IAM

Administrators can provide permissions for users by using the AWS managed policies that are provided for AWS Service Catalog. These policies are provided in the AWS Identity and Access Management (IAM) service. You create IAM users or groups, and then you apply the appropriate AWS managed policies to desired users or groups. For more information, see Permissions (p. 38).

To complete the tutorial that is provided in Getting Started (p. 7), you must create an IAM user, create a group that has the end user policy applied to it, and add the end user to the group.

To create the IAM user and group for the Getting Started tutorial

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Users** in the navigation pane, and choose **Create New Users**.
3. Type **`Engineer`** for the user name and choose **Create**.
4. Choose **Close** twice to return to the **Users** page. Then choose **Engineer**.
5. Under **Security Credentials**, choose **Manage Password** and follow the prompts to create a password.
6. Choose **Groups** in the IAM menu, and choose **Create New Group**.
7. Type **`Engineers`** for the group name and choose **Next Step**.
8. Choose the checkbox for the **ServiceCatalogEndUser** policy, and choose **Next Step**. Then choose **Create Group**.
9. Choose **Engineers** to open the group, and choose **Add Users to Group**.
10. Choose the **Engineer** checkbox, and choose **Add Users**.
11. Choose **Dashboard** to return to the IAM dashboard.

You can now sign in with your IAM user at your account-specific URL, which is shown on the top of the IAM dashboard and has the form:

https://*`AccountID`*.signin.aws.amazon.com/console

You can customize this URL to replace the account ID with an alias that you specify; see the AWS Identity and Access Management User Guide for details. Navigate to the URL and enter the user name and password you created above and click **Sign In** to proceed. Make a note of the sign-in URL for later use.

> **Tip**
> Logging in to your IAM user will sign out your root user. If you prefer to keep your root console open while using the IAM user, you can log in to the console using a private browsing window (**Ctrl**+**Shift**+**P** in Firefox and Internet Explorer, **Ctrl**+**Shift**+**N** in Chrome).

Creating a Key Pair

You will assign a key pair to the cloud resources you launch and then use the private key to connect to the Amazon EC2 instance after it is launched.

To create a key pair

1. Log in to the AWS Management Console.
2. Choose the **Services** menu, and choose **EC2**.
3. Under **Network and Security**, choose **Key Pairs**, and then choose **Create Key Pair**.
4. Type a name for the key pair and choose **Create**.

The console will prompt you to download the private key.

> **Note**
> AWS does not store private key and cannot recover them if lost. This is the only opportunity that you will have to download the private key, so be sure to save it to a secure location.

AWS CloudFormation Templates (Optional)

AWS CloudFormation templates are available for use with this guide, but if you already have an application running on AWS that you would like to use to create products, you can use an AWS CloudFormation template configured to launch that application.

This guide will cover some basic aspects of AWS CloudFormation, but you should consult the *AWS CloudFormation User Guide* if you need to create a template for a complex application.

Getting Started

This tutorial shows how to create an AWS Service Catalog portfolio and product. You will start with an AWS CloudFormation template that defines the AWS resources used by the product, then, you will create a product, add it to a portfolio, and distribute it to users.

> **Note**
> Before you begin this tutorial, you must have an AWS account and have created the Administrator user, End User user, and Engineering group in AWS Identity and Access Management (IAM), as described in Managing Users and Groups in IAM (p. 4).

Topics

Step 1: Prepare the AWS CloudFormation Template

A simple AWS CloudFormation template will get you started. The provided template, development-environment, launches a single Linux instance configured for SSH access.

> **Note**
> All the resources (such as products and portfolios) created in AWS Service Catalog are region specific. Check which region your console is set to when you start this tutorial and keep the same region setting throughout. For more information, see Selecting a Region in the *AWS Management Console Getting Started Guide*.

Sample Template

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

The template declares the resources that will be created when the product is launched. It consists of seven sections:

- **AWSTemplateFormatVersion** – The version of the AWS Template Format used to author this template.
- **Description** – A description of the template's content.
- **Parameters** – Three 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 entered.
- **Metadata** – Information that AWS Service Catalog uses to format the parameter entry that is displayed to the end user when launching the product.
- **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 Amazon Elastic Compute Cloud (Amazon 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 enters 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 (the other resource defined by this template) 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.

The text of the provided template follows:

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

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

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

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

    "SSHLocation" : {
      "Description" : "The IP address range that can SSH to the EC2 instance.",

      "Type": "String",
```

```
      "MinLength": "9",
      "MaxLength": "18",
      "Default": "0.0.0.0/0",
      "AllowedPattern":
"(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription": "Must be a valid IP CIDR range of the form
x.x.x.x/x."
   }
  },

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

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

  },

  "Resources" : {
    "EC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "InstanceType" : { "Ref" : "InstanceType" },
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "KeyName" : { "Ref" : "KeyName" },
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" :
"AWS::Region" }, "HVM64" ] }
      }
    },

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
```

```
        "GroupDescription" : "Enable SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : { "Ref" : "SSHLocation"}
        } ]
      }
    }
  },

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

Step 2: Create a Portfolio

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

To create a portfolio

1. Log in to the AWS Management Console with your administrator IAM user name and password.
2. Choose **Services**, **Administration & Security**, and then **AWS Service Catalog** to open the administrator console.
3. Choose **Get started now** to start the wizard for configuring a portfolio.
4. Enter the following values:

   - **Portfolio name** – **`Engineering Tools`**
   - **Description** – **`Sample portfolio that contains a single product.`**
   - **Owner** – **`IT (it@example.com)`**

5. Choose **Create**. AWS Service Catalog creates the portfolio and displays the Portfolio details page. This page displays information about the portfolio and allows you to add products.
6. Expand the **Tags** section and add a tag with key **`Group`** and value **`Engineering`**.

Step 3: Create a 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, which is a cloud development environment that runs on Amazon Linux. If you've just completed the previous step, the portfolio details page will already be displayed. To return to this page later, choose **Service Catalog**, **Portfolios**, and then choose **Engineering Tools**.

To create a product

1. On the portfolio details page, choose **Upload New Product**, and then enter the following:

   - **Product name** – **`Linux Desktop`**
   - **Short Description** – **`Cloud development environment.`**
   - **Description** – **`Cloud development environment configured for engineering staff. Runs AWS Linux.`**
   - **Provided by** – **`IT`**
   - **Vendor** – *(blank)*

   Click **Next**
2. On the **Enter support details** page, enter the following:

   - **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.`**

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

   - **Select template** – Choose **Specify an Amazon S3 template URL** and enter **`https://awsdocs.s3.amazonaws.com/servicecatalog/development-environment.template`**.
   - **Version title** – **`v1.0`**
   - **Description** – **`Base Version`**

   Choose **Next**.
4. On the **Review** page, verify the information you provided to create the product.
5. Choose **Confirm and upload** to create the product and add it to the Engineering Tools portfolio.

Step 4: Add a 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).

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 will limit valid instance types to the two smallest types, t2.micro and t2.small .

To add a template constraint to the Linux Desktop product

1. Navigate to the **Portfolios** page, and then choose Engineering Tools.
2. For **Constraints**, choose **Add constraints**.
3. Enter the following:

   - **Product** – **`Linux Desktop`**
   - **Constraint type** – **`Template`**

Choose **Next**

4. For **Description**, enter **`Small instance sizes`**.
5. Paste the following into the **Template constraint** text box:

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

        }
      ]
    }
  }
}
```

6. Choose **Submit**.

Step 5: Grant End Users Access to Your Portfolio

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

To provide access to a portfolio

1. Navigate to the **Portfolios** page, and then choose Engineering Tools.
2. Choose **Users, groups, and roles**, and then choose **Add user, group or role** to see a list of the groups and users associated with your account.
3. Choose **Engineering**, and then choose **Add Access**.

   Note
   If no groups are listed, follow the instructions in Setting Up (p. 4) to create a group in AWS Identity and Access Management (IAM) and add a user to it.

Step 6: Verify That the IAM Account Works

Verify that end users can use the IAM account that you created earlier.

To verify that an end user with the IAM account created earlier has access

1. Log in to the IAM user from your account-specific sign-in page.
2. Choose **Services**, **Administration & Security**, and then **AWS Service Catalog**. The AWS Management Console checks the IAM user's permissions and redirects you to the end user console automatically.

   You now see a list of products and stacks:

   - **Products** – The products that the user can use.

- **Stacks** – The stacks that the user has launched.

Launch the Product

Next, use the user's account to launch the product so that you can verify that end users will have access.

To launch a product stack

1. In the **Products** section of the end user console, choose **Linux Desktop**.

   This page lists the product information that end users will see, including the product name, description, and support details.
2. Choose **Launch product** to start the wizard for configuring your product.
3. On the **Product version** page, enter a stack name (e.g., **`test-stack`**), and choose the base version. Choose **Next**.
4. On the **Parameters** page, enter the following:

   - **Server size** – Choose a large instance type to verify that the constraint has been applied. If you set the constraint correctly, you will see an error message. Then set the **Server size** to **`t2.micro`**.
   - **Key pair** – Select the key pair that you created for your end user in Setting Up (p. 4).
   - **CIDR range** – Enter 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.

   Choose **Next**.
5. On the **Tags** page, for the **Key** enter **`team`** and for the **Value** enter **`engineering`**, and then choose **Next**.
6. On the **Review** page, review the information that you entered, and then choose **Launch** to launch the stack.

   When the wizard has finished, AWS Service Catalog displays your active stack.

Creating and 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 catalog-admin privileges for AWS Identity and Access Management users. See Managing Users and Groups in IAM (p. 4) for instructions on creating an IAM user for AWS Service Catalog.

To access the AWS Service Catalog administrator's console, log in to the AWS Management Console and navigate to https://console.aws.amazon.com/catalog/. The first time you access the administrator's console, you are prompted to create a portfolio. Follow the instructions to create your first portfolio, and then proceed to the **Portfolios** page.

Topics

Default Service Limits

By default, AWS limits the products and portfolios you can create, the number of constraints that you can apply to products, and the number of tags that you can apply to portfolios. The following table lists the limits for each AWS Service Catalog resource.

Default Limits

Resource	Limit
Portfolios	25 per account
Products	25 per account

Resource	Limit
Product Versions	10 per product
Constraints	25 per product per portfolio
Tags	3 per product, 3 per portfolio, 10 per stack
Stacks	25 per account (AWS CloudFormation limit)

For more information on limits, including how to increase limits for your account, refer to AWS Service Limits in *AWS General Reference*.

Managing Portfolios

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

Using the Portfolios Page

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

- Sign in to the AWS Management Console and go to AWS Service Catalog at https://console.aws.amazon.com/catalog/portfolios.

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

Using the Portfolio Details Page

In the AWS Service Catalog administrator console the **Portfolio details** page lists all of the settings for a portfolio. Use this page to manage the products in the portfolio, grant users access to products, and apply tags and constraints to ensure that products are used in the right way.

To view the Portfolio details page

1. Sign in to the AWS Management Console and go to AWS Service Catalog at https://console.aws.amazon.com/catalog/.
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 **Portfolio Details** page, enter the requested information.
4. Choose **Continue**. 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 and Removing 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.

Uploading a New Product

You upload 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 upload 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**.

Configuring Constraints

To control how users are able to use products, add constraints. The types of constraints that AWS Service Catalog supports are detailed in the Applying Constraints (p. 26) section.

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

To add a constraint to a product

1. Navigate to the **Portfolios** page, and then choose a portfolio. The portfolio details page opens.
2. Expand the **Constraints** section, and then choose **Add Constraints**.

3. For the **Product**, choose the product that you want to apply the constraint to.
4. 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 document that contains one or more rules. Rules are added to the AWS CloudFormation template used by the product.

5. Click **Submit**.

For more information about constraint types and their use, see Applying Constraints (p. 26).

Tagging Portfolios

You can assign tags to portfolios to track products that are launched from that portfolio. When you add a tag to a portfolio, the tag is applied to all products launched from that portfolio. Tags are applied to all resources that are created when a product is launched. You can assign a maximum of three tags to a portfolio.

To add tags to a portfolio

1. Navigate to the **Portfolios** page, and then choose the portfolio.
2. On the portfolio details page, expand the **Tags** section.
3. Type a key and value for the tag.
4. Choose **Add Tag**.

To delete a tag, choose the check box next to its key, and then choose **Delete Tag**. Deleted tags are no longer applied to new stacks when they launch, but remain on resources that have already been launched.

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 *Using IAM* guide.

To grant portfolio access to users or groups

1. In the portfolio details page, expand the **Users, groups, and roles** section, 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**.

Note
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 Permissions (p. 38).

Managing Products

You create products by packaging a 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 stacks with just a few clicks.

Displaying the Products Page

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

To view the Products page

1. Sign in to the AWS Management Console, and then navigate to https://console.aws.amazon.com/catalog/.
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 makes it easier to find the products that they need.

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

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

Choose **Next**.

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

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

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.

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 to Portfolio**.

Updating Products

When you need to update a product's AWS CloudFormationtemplate, 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 stack of the previous version of the product can update their stack using the end user console. When a new version of a product is available, users can use the **Update Stack** command on either the **Stack list** or **Stack details** page.

> **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 detail** page, expand the **Versions** section, and then choose **Create New Version**.
4. For **Version Details**, enter the following:

   - **CloudFormation Template** – Choose **Browse**, and then either choose the updated AWS CloudFormation template that is stored locally or type the URL of the updated template that is stored in Amazon S3.
   - **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 it 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**.

Adding an AWS Marketplace Product 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 http://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 end user console.

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

To subscribe to an AWS Marketplace product

1. Go to AWS Marketplace at http://aws.amazon.com/marketplace.
2. Browse the products or search to find the product that you want to add to your AWS Service Catalog portfolio. Choose the product to view the product details page.
3. Choose **Continue** to view the fulfillment page, and then choose the **Manual Launch** tab.

   The information on the fulfillment page includes the supported Amazon Elastic Compute Cloud (Amazon EC2) instance types, the supported AWS regions, and the Amazon Machine Image (AMI) ID that the product uses for each AWS region. 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.

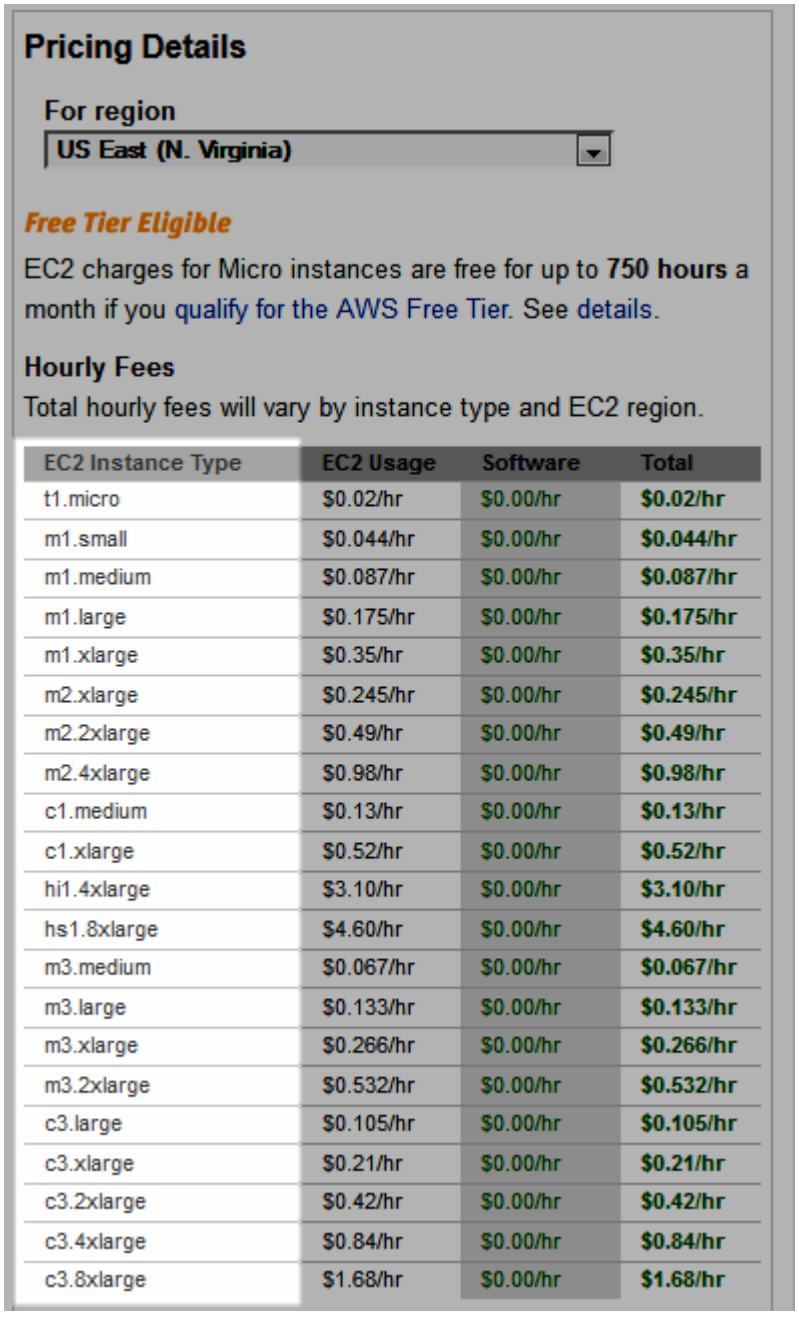
To define your AWS Marketplace product in an AWS CloudFormation template

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. You can view the sample templates in the *AWS CloudFormation User Guide* at http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-sample-templates.html.

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 **services**.
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 see the sample template, choose **View** , and then save a copy of the template locally so that you can edit it. Your local file must have the `.template` extension.
4. Open your template file in a text editor.
5. Customize the description at the top of the template. Your description might look like the following example:

   ```
   "Description": "Launches a LAMP stack from AWS Marketplace",
   ```
6. Customize the `InstanceType` parameter so that it includes only EC2 instance types that are supported by your product. If your template includes unsupported EC2 instance types, the product will fail to launch for your end users.

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



Pricing Details

For region

US East (N. Virginia)

Free Tier Eligible

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

Hourly Fees

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Applying Constraints

To control how a product is launched and 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.

Template constraints apply to a product only in the portfolio to which you applied the constraint. If you add the product to another portfolio, it will not have the same constraints.

> **Note**
> Constraints are active as soon as you create them. Constraints apply to all versions of a product that are not already launched when you create the constraint. 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.

Topics

Applying Launch Constraints

A launch constraint designates an AWS Identity and Access Management (IAM) role for AWS Service Catalog to assume for stack creation and management operations. An IAM role is a collection of permissions that an IAM user or AWS service can assume temporarily to use AWS services.

Without a launch constraint, end users must launch and manage products with 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 more information about end user permissions, see Permissions (p. 38).

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 services:

- AWS CloudFormation.
- The services that are in the AWS CloudFormation template for the product.
- Amazon Simple Storage Service (Amazon S3). AWS Service Catalog must have read access to the AWS CloudFormation template in Amazon S3.

The IAM role also must have a trust relationship with AWS Service Catalog. This allows AWS Service Catalog to assume the role during the launch process to create resources.

A sample launch role and trust policy are provided in the following steps.

> **Note**
> To create and assign IAM roles, you must have the following IAM administrative permissions: iam:CreateRole, iam:PutRolePolicy, iam:PassRole, iam:Get*, and iam:List*

To create a launch role

1. Sign in to the AWS Management Console and 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 **Amazon EC2**, 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. Under **Permissions**, choose **Inline Policies**, and then choose the link to create a new policy.
3. Choose **Custom Policy**, and then choose **Select**.
4. Enter a name for the policy, and then paste the following into the **Policy Document** box:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "catalog-admin:*",
            "servicecatalog:*",
            "cloudformation:GetTemplateSummary",
            "cloudformation:ValidateTemplate",
            "cloudformation:CreateStack",
            "cloudformation:UpdateStack",
            "cloudformation:DeleteStack",
            "cloudformation:DescribeStackEvents",
            "cloudformation:DescribeStacks",
            "s3:*"
         ],
         "Resource":"*"
      }
   ]
}
```

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**.

To update the role's trust relationship

1. Return to the **Role details** page.
2. Under **Trust Relationships**, choose **Edit Trust Relationship** .

3. Replace **`ec2.amazonaws.com`** with **`servicecatalog.amazonaws.com`**. Your trust policy will look like the following example:

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

4. Choose **Update Trust 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

1. Go to the AWS Service Catalog console at https://console.aws.amazon.com/catalog/.
2. Choose the portfolio that contains the product.
3. Expand the **Constraints** section and choose **Add constraints**.
4. Choose the product and set **Constraint type** to **Launch**. Choose **Continue**.
5. For the **IAM role**, choose the launch role Enter and choose **Submit**.

Verify That the Launch Constraint Is Applied

Verify that AWS Service Catalog uses the role to launch the product and that the product stack is created successfully by launching the product from the AWS Service Catalog end user console .

To launch the product

1. In the menu for the AWS Service Catalog administrator 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.

Defining 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 (such as t1.micro, m1.large, and so on) 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 stack launches and for all versions of the product.

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. 29).

To apply template constraints to a product

1. Sign in to the AWS Management Console and navigate to the AWS Service Catalog administrator console at https://console.aws.amazon.com/catalog/ .
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, choose the product for which you want to define the template constraints, and for the 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 **JSON 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** 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, click **Add rule**. The rule appears in the table on the **Rule** tab. To review and edit the JSON output, choose the **JSON editor** tab.
6. When you are done editing the rules for your constraint, choose Submit. To see the constraint, choose the **Portfolio details** page, and look at the **Constraints** section.

Template Constraint Rules

The `Rules` that define template constraints in an AWS Service Catalog portfolio describe when end users can use the template and which values they can specify for parameters that are declared in the AWS CloudFormation template used to create the product they are attempting to use. Rules are useful for

preventing end users from inadvertently specifying an incorrect value. For example, you can add a rule to verify whether end users specified a valid subnet in a given VPC or used `m1.small` instance types for test environments. AWS CloudFormation uses rules to validate parameter values before it creates the resources for the product.

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

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

Syntax

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

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

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

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

effect. `Rule02` always takes effect because it doesn't have a rule condition, which means the one assert is always evaluated and applied.

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.

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

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 stack is created or updated. If a parameter values 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.

Topics

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 of strings.

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. 37).

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 Avalibility 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. 37).

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*

Tags

Tags are words or phrases that act as metadata for identifying and organizing your AWS resources. Each resource can have up to ten tags, each of which consists of a key and a value. For more information about tagging, see What Is a Tag? in the *AWS Billing and Cost Management User Guide*.

You can add three tags to a portfolio and three tags to a product. When a product is launched, its tags and the tags of its portfolio are combined and applied to the stack automatically. For a portfolio, you might add a **`Cost Center`** or **`Group Name`** tag for the group that the portfolio is distributed to. For a product, you could add a **`Product Owner`**, **`License Type`**, or **`Operating System`** tag to specify information about the product.

When end users launches a stack, they can add tags to the stack beyond those inherited from the product or portfolio. If you want your users to add specific tags, specify them in the product's description. For example, you may want the user to enter a **`Name`** tag for the stack, or a **`User`** tag with their user name as the value.

Tags are assigned to stack resources during product launch and cannot be changed when a stack is updated or at any other time during a stack's lifecycle.

Note
AWS CloudFormation also adds three tags (`stack name`, `stack ID`, and `logical ID`) to each cloud resource when it is created. These tags do not count toward tag limits.

Using Tags

The tags you add appear as columns in the Amazon Elastic Compute Cloud (Amazon EC2) console and are automatically added to the list of filters in the search bar. You can sort by tag to find the resources you are looking for.

You can also manage tags collectively using Tag Editor in the AWS Management Console. For more information, see Working with Tag Editor.

Tip
Add a tag with the **Name** key to name stack resources at launch. The **Name** column is displayed by default in the Amazon EC2 console.

Permissions

You can control the level of access that AWS Service Catalog users have to AWS resources by applying permissions through AWS Identity and Access Management (IAM). If you are not familiar with IAM, see the IAM User Guide to learn more about granting permissions.

IAM includes AWS managed policies that are preconfigured for AWS Service Catalog. These polices are created and managed by AWS. They provide the permissions that administrators need and the initial permissions that end users need. You can attach these policies to the IAM users, groups, and roles that you use with AWS Service Catalog. For more information about AWS managed policies, see AWS Managed Policies in the IAM User Guide.

The following AWS managed policies for AWS Service Catalog are provided in IAM:

`ServiceCatalogAdmin`
: Provides access to the AWS Service Catalog administrator console. Allows administrators to create and manage products and portfolios.

`ServiceCatalogAdminReadOnly`
: Provides read-only access to the AWS Service Catalog administrator console.

`ServiceCatalogEndUser`
: Provides the minimal permissions required to access to the AWS Service Catalog end user console.

The `ServiceCatalogEndUser` policy does not include permissions that allow an end user to launch products or manage stacks, and it does not include the permissions to use the underlying AWS resources that are in a product's AWS CloudFormation template. To allow an end user to launch a product and manage stacks, you must create a policy with additional permissions. The policy must include the following AWS CloudFormation permissions:

```
"cloudformation:CreateStack",
"cloudformation:DeleteStack",
"cloudformation:UpdateStack"
```

You must also add permissions to use the underlying AWS resources for the product. For example, if a product template includes Amazon RDS, then the user will require Amazon RDS permissions to launch the product.

You can grant these permissions directly to an end user in IAM, but to limit the access that end users have with AWS, you instead 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. See Applying Launch Constraints (p. 26) for instructions and a sample launch role.

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 and add tags. For more information, see Granting Access to Users (p. 18) and Tagging Portfolios (p. 18).

The recipient administrator can distribute the products to end users who belong to the 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 stack 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 Applying Launch Constraints (p. 26). 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.

Summary of 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.

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

Sharing Your 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 AWS Account ID of the target AWS account. The ID is available on the **My Account** page in the AWS Management Console of the target account.

To share your portfolio

1. Sign in to the AWS Management Console and navigate to https://console.aws.amazon.com/catalog/.
2. On the **Portfolios** page, select the portfolio that you want to share and choose **Share Portfolio**.
3. In the **Enter AWS account ID** dialog box, type the account ID of the AWS account that you are sharing with, and 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.

Using the End User Console

Use the AWS Service Catalog end user console to start and stop the products you need to do your job. Also use the end user console to manage the computing resources (known collectively as a *stack*) needed to run those products. The home page for the end user console is the dashboard, which you can find at https://console.aws.amazon.com/servicecatalog/.

> **Note**
> If you see an error message when attempting to access the AWS Service Catalog end user console, contact your administrator to ensure that your account has both the permissions required to use the AWS Service Catalog service and access to one or more products.

Topics

Using the Dashboard

The AWS Service Catalog Dashboard displays a list of products and a list of stacks. From the Dashboard, you can launch products, and view, update, or delete stacks that you have created.

To view the AWS Service Catalog Dashboard

- Sign in to the AWS Management Console, and then navigate to https://console.aws.amazon.com/servicecatalog/.

While using AWS Service Catalog, you can return to the Dashboard at any time by choosing the link at the top of the page or by choosing **Dashboard** from the **Service Catalog** menu.

The Dashboard shows up to five products and five stacks. You can see a complete list of products and stacks on the **Product list** and **Stacks list** pages, which you can display by choosing them from the **Service Catalog** menu.

Using the Product List

The **Product list** shows the applications, tools, and cloud resources that your administrator has made available to you. You can use the **Product list** to launch an instance of those products and manage each stack you create.

To view the Product list

1. Sign in to the AWS Management Console, and then navigate to https://console.aws.amazon.com/servicecatalog/.
2. Choose **See All Products**.

You can return to the **Product list** at any time by choosing **Service Catalog** in the navigation bar, and then choosing **Product List**.

Using the Stack List

The **Stack list** displays all of the stacks that you have created by launching products. By default, the **Stack list** shows each stack's name, the time it was created, its current status, and a status message, if applicable. You can also use the column chooser to show stack ARNs (Amazon Resource Names) and the time they were last updated. Use the **Stack list** to search for stacks by name, update a stack to a new version, or delete a stack.

To view the Stack list

1. Sign in to the AWS Management Console, and then navigate to https://console.aws.amazon.com/servicecatalog/.
2. Choose **See All Stacks**.

While using AWS Service Catalog, you can return to the **Stack List** at any time by choosing **Service Catalog** in the navigation bar, and then choosing **Stack list**.

To change the columns that are visible

1. Choose the **Edit Columns** button (the gear icon at the top right of the **Stack List**).
2. Choose any of the available columns to show or hide them.
3. Choose **Save**.

Viewing Available Products

The **Product details** page displays information about a product, including a description of the product, details about product versions, and support information.

To view detailed information about a product

1. Navigate to the **Product List**.

2. Choose the product name.

Choosing the Product Version

If multiple versions of a product are available, you can decide which version to use by reading the version descriptions. Typically, you should use the latest version of a product.

Contexts

The launch context for the product includes identifiers for the product, the portfolio used to deliver it, and constraints or tags that are applied during launch. When you launch a product, the context determines the information needed to create and configure the stack, the parameters that you supply, and the constraints used to validate those parameters.

- **Path** – The ARN (Amazon Resource Name) of the product and the portfolio that contains the product.
- **Launch as** – The ARN of the role assumed by AWS Service Catalog to launch the product. If this field is blank, the product is launched with your user permissions.
- **Rules** – The names of template constraints applied to the product during launch.
- **Tags** – The names and values of tags that are inherited from the portfolio or product.

Tags

Tags are metadata assigned to a stack for tracking and analysis. In addition to the tags that you enter when you launch a product, a stack may have tags that were applied to the product or to the portfolio by the AWS Service Catalog administrator.

Support Details

Support details can include an email address, URL, or both. Support details are provided by the administrator when creating the product. Use this information to get help with your products.

Launching a Product

You can launch any product that appears in your AWS Service Catalog Using the Dashboard (p. 42) or Using the Product List (p. 43). Launching a product creates an instance of the product in an AWS stack. A stack in AWS is one or more cloud resources (compute instances, databases, networking components, etc.) that you manage as a single unit.

To launch a product

1. Choose the product in the AWS Service Catalog Using the Dashboard (p. 42) or Using the Product List (p. 43), and then choose **Launch product**.
2. On the **Product Version** page, enter a stack name. Stack names must start with a letter and can contain only letters, numbers, and dashes.
3. Choose the version of the product to launch, and then choose **Next**.
4. On the **Parameters** page, enter values for each parameter required by the product, and then choose **Next**. If a product has no parameters, AWS Service Catalog skips this step.
5. On the **Tags** page, add the tags that you would like to use with your product stack, and then choose **Next**. Tags can have a key and value and they help you identify resources in your stack. The most

common tag key is "Name"; AWS uses this key to populate the **Name** field for resources shown in other areas of the AWS Management Console.

A stack can inherit a maximum of three tags each from the product and portfolio, and can have a maximum of ten tags. Additional tags are added to some resources by AWS CloudFormation, but these do not apply toward the limit and do not appear on this page.

6. On the **Review** page, review the values that you entered, and then choose **Launch**.

When you choose **Launch**, you are redirected to the Using the Stack List (p. 43). If you want to see status message updates as resources are created and parameters are validated, choose **Refresh**.

If a problem occurs during launch, the status changes to **FAILED**. To identify the problem, choose the stack name to display the **Stack details** page by choosing the stack's name.

If the product launches successfully, the status changes to **SUCCESS**. To see output generated by the launch, click through to the **Stack Details** page.

Viewing Stack Information

Each stack has a **Stack details** page that displays information about the stack. The **Stack details** page is available from the time the product is first launched until the stack is deleted.

To view details about a stack

1. Navigate to the Using the Dashboard (p. 42) or Using the Stack List (p. 43).
2. Choose the stack.

Viewing Stack Status

Each stack that you launch changes state as AWS Service Catalog attempts to create and configure AWS resources using the product template and parameters that the user enters during launch. If all goes well, the stack advances from an initial status of **LAUNCHING** to **AVAILABLE**.

A stack's status is shown in the **Dashboard**, **Stack list**, and on the **Stack details** page. A status of **AVAILABLE** indicates that the product launched successfully and is ready for use.

If any of the cloud resources in a stack failed to start or if parameters failed to pass all constraints applied to the product, all of the resources are terminated and the stack has a status of **FAILED**. A failed stack cannot be recovered, but remains in the **Stack list** for troubleshooting.

When you update a stack to use a new version or different parameters, the stack's status is **UPDATING**. If the update succeeds, the stack's status changes to **AVAILABLE**.

The status of a deleted stacks is **TERMINATING** while resources are being terminated. When all of the resources have terminated, the stack is removed from AWS Service Catalog and no longer is listed.

The operations that you can perform on a stack depends on the stack's status. For example, stacks that are **AVAILABLE** can be updated or deleted, but stacks that are **LAUNCHING**, **UPDATING**, or **TERMINATING** cannot. **FAILED** stacks can only be viewed and deleted.

Viewing Outputs

Stacks provide information, called outputs, when a product is launching. Outputs usually display URLs, IP addresses, and database connection strings that are generated when the stack is launched. Each output has a key, value, and description.

How you use the information provided by outputs depends on the type of product you launch. For example, if the product launches an EC2 instance, the stack might generate the IP address of the instance, which you could use to connect to the instance using Remote Desktop Connection or SSH.

Viewing Events

AWS CloudFormation provides information during each step of the launch and updating processes. When a stack's status changes, resources are created, or errors occur, AWS CloudFormation logs an event with the following information:

- **Date** – The time that the event occurred, in local time.
- **Status** – The condition of a resource in a stack, as opposed to the Viewing Stack Status (p. 45).
- **Type** – The type of the resource that is the subject of the event. For details on resource types, see Resource Types in the *AWS CloudFormation User Guide*.
- **Logical ID** – The name of the resource, as defined in the underlying template.
- **Status reason** – Additional information about the stack's status, if available.
- **Physical ID** – The physical identifier of the resource, which appears when you choose an event.

Entering Parameters

You enter parameters when launching or updating a stack. If you enter an incorrect parameter value when you launch or update a stack, **CREATE_FAILED** will appear in the Viewing Events (p. 46) section.

Viewing Tags

Tags are metadata that are applied to the stack during launch. The **Stack details** page also shows tags that were inherited from the product and portfolio.

Viewing Support Details

If your AWS Service Catalog administrator provided support information in this optional section, you will find an email address or site link that you can use to get support if you encounter problems with your stack. It might also contain additional support information.

Updating Stacks

When you want to use a new version of a product or configure a running stack with updated parameter values, you update it. However, you cannot update a stack to change tags.

You can update stacks only if they have a status of **AVAILABLE**. You cannot update failed stacks or stacks that are in the process of starting, updating, or terminating. See Viewing Stack Status (p. 45) for more information on stack status.

To update a stack

1. Choose the stack, and then choose **Update stack**.
2. Choose the version that you want to update, and then choose **Next**.
3. Enter the parameters, and then choose **Next**.
4. Choose **Update**.

The stack status changes to **UPDATING**. To see output from the update operation, open the **Stack details** page and expand the **Events** section .

Deleting Stacks

To remove all AWS resources that a stack uses, delete the stack. Deleting a stack terminates all resources and removes the stack from your stack list. Delete a stack only if you no longer need it. Before deleting a stack, record any information about the stack or its resources that you might need later.

Before deleting a stack, ensure that it is in either the available or failed state. AWS Service Catalog can delete stacks only in these two states. For more information on stack status, see Viewing Stack Status (p. 45).

To delete a stack

1. Navigate to the Using the Dashboard (p. 42) or Using the Stack List (p. 43).
2. Select the stack, and then choose **Terminate Stack**.
3. Verify that you've chosen the stack that you want to delete, and then choose **Terminate**.

Document History

The following table describes the important changes to the documentation since the last release of AWS Service Catalog.

- **Latest documentation update**: July 31, 2015

Feature	Description	Release Date
Documentation updates for overriding imported launch constraints	The information in Portfolio Sharing (p. 39) is updated to explain how launch constraints are inherited by imported portfolios and how the administrator who imports a portfolio can override these launch constraints.	July 31, 2015
Documentation updates for IAM policies and sample launch role	Replaced the sample IAM policies in Permissions (p. 38) with references to the AWS managed policies in IAM, which are preconfigured for AWS Service Catalog. Corrected the sample launch role and added a sample trust policy in Applying Launch Constraints (p. 26).	July 22, 2015
New guide	This is the first release of *AWS Service Catalog Administrator Guide*.	July 9, 2015