AWS Service Catalog: Administrator Guide
Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.
# Table of Contents

What Is AWS Service Catalog? ................................................................................................. 1
  Overview ............................................................................................................................ 1
    Users ............................................................................................................................... 1
    Products ......................................................................................................................... 2
    Provisioned Products .................................................................................................... 2
    Portfolios ....................................................................................................................... 2
    Versioning ...................................................................................................................... 2
    Permissions .................................................................................................................... 2
    Constraints ..................................................................................................................... 3
    Initial Administrator Workflow .................................................................................... 3
    Initial End User Workflow ........................................................................................... 3
  Quotas ............................................................................................................................. 4

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

Getting Started .................................................................................................................. 9
  Step 1: Download the Template ..................................................................................... 9
    Template Download ...................................................................................................... 9
    Template Overview ....................................................................................................... 9
  Step 2: Create a Key Pair ............................................................................................... 12
  Step 3: Create a Portfolio ............................................................................................... 12
  Step 4: Create a Product ............................................................................................... 13
  Step 5: Add a Template Constraint ............................................................................... 13
  Step 6: Add a Launch Constraint .................................................................................. 14
  Step 7: Grant End Users Access to the Portfolio ............................................................. 15
  Step 8: Test the End User Experience ............................................................................ 16

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

Security .................................................................................................................................. 19
  Data Protection ............................................................................................................... 19
    Protecting Data with Encryption ................................................................................... 20
  Identity and Access Management .................................................................................... 20
    Audience ...................................................................................................................... 21
    Controlling Access ....................................................................................................... 21
    Predefined AWS Managed Policies ............................................................................ 21
    Console Access for End Users ..................................................................................... 22
    Product Access for End Users ....................................................................................... 23
    Example Policies ......................................................................................................... 23
  Logging and Monitoring ................................................................................................. 26
  Compliance Validation ..................................................................................................... 26
  Resilience ......................................................................................................................... 27
  Infrastructure Security .................................................................................................... 27
  Security Best Practices .................................................................................................... 27

Managing Catalogs ............................................................................................................. 29
  Managing Portfolios ....................................................................................................... 29
    Creating, Viewing, and Deleting Portfolios ................................................................. 29
    Viewing Portfolio Details ............................................................................................. 30
    Creating and Deleting Portfolios .................................................................................. 30
    Adding Products .......................................................................................................... 30

Audience ............................................................................................................................. 21

Logging and Monitoring .................................................................................................... 26
Compliance Validation ........................................................................................................ 26
Resilience ............................................................................................................................ 27
Infrastructure Security ....................................................................................................... 27
Security Best Practices ....................................................................................................... 27
Managing Catalogs ............................................................................................................. 29
Managing Portfolios ........................................................................................................... 29
Creating, Viewing, and Deleting Portfolios ....................................................................... 29
Viewing Portfolio Details .................................................................................................. 30
Creating and Deleting Portfolios ....................................................................................... 30
Adding Products ................................................................................................................ 30
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background</td>
<td>79</td>
</tr>
<tr>
<td>Getting Started</td>
<td>80</td>
</tr>
<tr>
<td>Release Notes</td>
<td>80</td>
</tr>
<tr>
<td>Baseline Permissions</td>
<td>81</td>
</tr>
<tr>
<td>Configuring AWS Service Catalog</td>
<td>86</td>
</tr>
<tr>
<td>Creating StackSet Constraints</td>
<td>61</td>
</tr>
<tr>
<td>Relating Budgets to Products and Portfolios</td>
<td>87</td>
</tr>
<tr>
<td>Configuring ServiceNow</td>
<td>87</td>
</tr>
<tr>
<td>Validating Configurations</td>
<td>92</td>
</tr>
<tr>
<td>ServiceNow Additional Administrator Features</td>
<td>93</td>
</tr>
<tr>
<td>Upgrade Instructions</td>
<td>94</td>
</tr>
<tr>
<td>Connector for Jira Service Desk</td>
<td>98</td>
</tr>
<tr>
<td>Background</td>
<td>98</td>
</tr>
<tr>
<td>Jira Service Desk Supported Versions and Releases</td>
<td>99</td>
</tr>
<tr>
<td>Getting Started</td>
<td>99</td>
</tr>
<tr>
<td>Release Notes</td>
<td>100</td>
</tr>
<tr>
<td>Baseline Permissions</td>
<td>101</td>
</tr>
<tr>
<td>Configuring AWS Service Catalog</td>
<td>104</td>
</tr>
<tr>
<td>Configuring Jira Service Desk</td>
<td>106</td>
</tr>
<tr>
<td>IT Lifecycle Management Setup and Use Case</td>
<td>110</td>
</tr>
<tr>
<td>Validating Configurations</td>
<td>115</td>
</tr>
<tr>
<td>Jira Additional Administrator Features</td>
<td>116</td>
</tr>
<tr>
<td>Document History</td>
<td>117</td>
</tr>
</tbody>
</table>
What Is AWS Service Catalog?

AWS Service Catalog enables organizations to create and manage catalogs of IT services that are approved for use on AWS. These IT services can include everything from virtual machine images, servers, software, and databases to complete multi-tier application architectures. AWS Service Catalog allows organizations to centrally manage commonly deployed IT services, and helps organizations achieve consistent governance and meet compliance requirements. End users can quickly deploy only the approved IT services they need, following the constraints set by your organization.

AWS Service Catalog provides the following benefits:

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

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

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

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

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

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

Overview of AWS Service Catalog

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

**Users**

AWS Service Catalog supports the following types of users:

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

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

## Products

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

## Provisioned Products

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

## Portfolios

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

## Versioning

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

## Permissions

Granting a user access to a portfolio enables that user to browse the portfolio and launch the products in it. You apply AWS Identity and Access Management (IAM) permissions to control who can view and
modify your catalog. IAM permissions can be assigned to IAM users, groups, and roles. When a user launches a product that has an IAM role assigned to it, AWS Service Catalog uses the role to launch the product's cloud resources using AWS CloudFormation. By assigning an IAM role to each product, you can avoid giving users permissions to perform unapproved operations and enable them to provision resources using the catalog.

Constraints

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

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

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

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

Initial Administrator Workflow

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

Initial End User Workflow

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

Your AWS account has the following default quotas related to AWS Service Catalog.

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

Regional quotas

- Portfolios: 100
- Products: 350

Portfolio quotas

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

Product quotas

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

**Provisioned product quotas**

• Tags per provisioned product: 50

**Constraint quotas**

• Constraints per product per portfolio: 100

**Service action quotas**

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

**TagOption quotas**

• TagOptions per resource: 25
• Values per TagOption: 25
Setting Up for AWS Service Catalog

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

Sign Up for Amazon Web Services

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

**To sign up for an AWS account**

2. Follow the online instructions.

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

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

Grant Permissions to Administrators and End Users

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

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

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

Grant Permissions to AWS Service Catalog Administrators

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

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

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

To grant permissions to a catalog administrator

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

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ec2:CreateKeyPair",
            "iam:AddRoleToInstanceProfile",
            "iam:AddUserToGroup",
            "iam:AttachGroupPolicy",
            "iam:CreateAccessKey",
            "iam:CreateGroup",
            "iam:CreateInstanceProfile",
            "iam:CreateLoginProfile",
            "iam:CreateRole",
            "iam:CreateUser",
            "iam:Get*",
            "iam:List*",
            "iam:PutRolePolicy",
            "iam:UpdateAssumeRolePolicy"
         ],
         "Resource": [
            "*
         ]
      }
   ]
}
```
   c. Choose Review policy.
   d. For Policy Name, type ServiceCatalogAdmin-AdditionalPermissions.
e. (Optional) You must grant administrators additional permissions for Amazon S3 if they need to use a private CloudFormation template. For more information, see User Policy Examples in the Amazon Simple Storage Service Developer Guide.

f. Choose Create Policy.

5. Return to the browser window with the permissions page and choose Refresh.

6. In the search field, type ServiceCatalog to filter the policy list.

7. Select the checkboxes for the AWSServiceCatalogAdminFullAccess and ServiceCatalogAdminAdditionalPermissions policies, and then choose Next: Review.

8. If you are updating a user, choose Add permissions.

If you are creating a user, choose Create user. You can download or copy the credentials and then choose Close.

9. To sign in as the catalog administrator, use your account-specific URL. To find this URL, choose Dashboard in the navigation pane and choose Copy Link. Paste the link in your browser, and use the name and password of the IAM user you created or updated in this procedure.

Grant Permissions to AWS Service Catalog End Users

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

To grant permissions to an end user group

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

2. In the navigation pane, choose Groups.

3. Choose Create New Group and do the following:

   a. For Group Name, type Endusers, and then choose Next Step.

   b. In the search field, type AWSServiceCatalog to filter the policy list.

   c. Select the checkbox for the AWSServiceCatalogEndUserFullAccess policy, and then choose Next Step. You also have the option to choose AWSServiceCatalogEndUserReadOnlyAccess instead.


4. In the navigation pane, choose Users.

5. Choose Add user and do the following:

   a. For User name, type a name for the user.

   b. Select AWS Management Console access.

   c. Choose Next: Permissions.

   d. Choose Add user to group.

   e. Select the checkbox for the Endusers group and choose Next: Tags and then Next: Review.

   f. On the Review page, choose Create user. Download or copy the credentials and then choose Close.
Getting Started

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

Before You Begin

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

Tasks
- Step 1: Download the AWS CloudFormation Template (p. 9)
- Step 2: Create a Key Pair (p. 12)
- Step 3: Create an AWS Service Catalog Portfolio (p. 12)
- Step 4: Create an AWS Service Catalog Product (p. 13)
- Step 5: Add a Template Constraint to Limit Instance Size (p. 13)
- Step 6: Add a Launch Constraint to Assign an IAM Role (p. 14)
- Step 7: Grant End Users Access to the Portfolio (p. 15)
- Step 8: Test the End User Experience (p. 16)

Step 1: Download the AWS CloudFormation Template

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

Template Download

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

Template Overview

The text of the sample template follows:

```json
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Description" : "AWS Service Catalog sample template. Creates an Amazon EC2 instance running the Amazon Linux AMI. The AMI is chosen based on the region in which the stack is run. This example creates an EC2 security group for the instance to give you SSH access. **WARNING** This template creates an Amazon EC2 instance. You will be billed for the
```
AWS resources used if you create a stack from this template.

"Parameters" : {
    "KeyName": {
        "Description": "Name of an existing EC2 key pair for SSH access to the EC2 instance.",
        "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-fd4772cc" },
        "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" }
        }
    }
}
Template Overview

```json

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

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

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

Template Resources

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

- **AWSTemplateFormatVersion** – The version of the AWS Template Format used to create this template.
- **Description** – A description of the template.
- **Parameters** – The parameters that your user must specify to launch the product. For each parameter, the template includes a description and constraints that must be met by the value typed. For more information about constraints, see Using AWS Service Catalog Constraints (p. 37).

The KeyName parameter allows you to specify an Amazon Elastic Compute Cloud (Amazon EC2) key pair name that end users must provide when they use AWS Service Catalog to launch your product. You will create the key pair in the next step.

- **Metadata** – An optional section that defines details about the template. The AWS::CloudFormation::Interface key defines how the end user console view displays parameters. The ParameterGroups property defines how parameters are grouped and headings for those groups. The ParameterLabels property defines friendly parameter names. When a user is specifying parameters to launch a product that is based on this template, the end user console view displays the parameter labeled Server size: under the heading Instance configuration, and it displays the parameters labeled Key pair: and CIDR range: under the heading Security configuration.
- **Mappings** – A list of regions and the Amazon Machine Image (AMI) that corresponds to each. AWS Service Catalog uses the mapping to determine which AMI to use based on the region that the user selects in the AWS Management Console.
- **Resources** – An EC2 instance running Amazon Linux and a security group that allows SSH access to the instance. The Properties section of the EC2 instance resource uses the information that the user types to configure the instance type and a key name for SSH access.
AWS CloudFormation uses the current region to select the AMI ID from the mappings defined earlier and assigns a security group to it. The security group is configured to allow inbound access on port 22 from the CIDR IP address range that the user specifies.

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

---

**Step 2: Create a Key Pair**

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

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

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

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

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

**To create a key pair**

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

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

---

**Step 3: Create an AWS Service Catalog Portfolio**

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

**To create a portfolio**

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. If you are using the AWS Service Catalog administrator console for the first time, choose **Get started** to start the wizard for configuring a portfolio. Otherwise, choose **Create portfolio**.
Step 4: Create an AWS Service Catalog Product

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

To create a product

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

Step 5: Add a Template Constraint to Limit Instance Size

Constraints add another layer of control over products at the portfolio level. Constraints can control the launch context of a product (launch constraints), or add rules to the AWS CloudFormation template (template constraints). For more information, see Using AWS Service Catalog Constraints (p. 37).

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

To add a template constraint to the Linux Desktop product

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

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

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

A launch constraint designates an IAM role that AWS Service Catalog assumes when an end user launches a product. For this step, you will add a launch constraint to the Linux Desktop product so that AWS Service Catalog can use the AWS resources that are part of the product’s AWS CloudFormation template. This launch constraint will enable the end user to launch the product and, after it is launched, manage it as a provisioned product. For more information, see AWS Service Catalog Launch Constraints (p. 37).

Without a launch constraint, you would need to grant additional IAM permissions to your end users before they could use the Linux Desktop product. For example, the ServiceCatalogEndUserAccess policy grants the minimum IAM permissions required to access the AWS Service Catalog end user console view. By using a launch constraint, you can keep your end users’ IAM permissions to a minimum, which is an IAM best practice. For more information, see Grant least privilege in the IAM User Guide.

To add a launch constraint

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Policies. Choose Create policy and do the following:
   a. On the Create policy page, choose the JSON tab.
   b. Copy the following example policy and paste it in Policy Document, replacing the placeholder JSON in the text field:

   ```json
   {
       "Version": "2012-10-17",
   ```
Step 7: Grant End Users Access to the Portfolio

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

**Prerequisites**

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

**To provide access to the portfolio**

1. On the portfolio details page, choose the Groups, roles, and users tab.
2. Choose Add groups, roles, users.
3. On the Groups tab, select the checkbox for the IAM group for the end users.
Step 8: Test the End User Experience

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

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

1. To sign in as the IAM user, use account-specific URL. To find this URL, open the IAM console, choose Dashboard in the navigation pane, and choose Copy Link. Paste the link in your browser, and use the name and password of the IAM user.
2. In the menu bar, choose the region in which you created the Engineering Tools portfolio.
3. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/ and select Service Catalog, Dashboard to see the following:
   - Products – The products that the user can use.
   - Provisioned products – The provisioned products that the user has launched.

To verify that the end user can launch the Linux Desktop product

1. In the Products section of the console, choose Linux Desktop.
2. Choose Launch product to start the wizard for configuring your product.
3. On the Product version page, for Name, type Linux-Desktop.
4. In the Version table, choose v1.0.
5. Choose Next.
6. On the Parameters page, type the following and choose Next:
   - Server size – Choose t2.micro.
   - Key pair – Select the key pair that you created in Step 2: Create a Key Pair (p. 12).
   - CIDR range – Type a valid CIDR range for the IP address from which you will connect to the instance. This can be the default value (0.0.0.0/0) to allow access from any IP address, your IP address followed by /32 to restrict access to your IP address only, or something in between.
7. On the Review page, review the information that you typed, and then choose Launch to launch the stack. The console displays the stack details page for the Linux-Desktop stack. The initial status of the product is Launching. It takes several minutes for AWS Service Catalog to launch the product. To see the current status, refresh your browser. After the product is launched, the status is Available.
Getting Started Library

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

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

Prerequisites

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

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

Reference Architectures

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

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

To view the Reference Architectures portfolio in the administrator console

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

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

High Reliability Architectures

Our High Reliability Architectures portfolio is a repository of well-architected, multi-region blueprints. Each blueprint provides prescriptive implementation guidance for AWS services commonly used to
build multi-region workloads. Examples include patterns for managing infrastructure changes and data storage backup and recovery for user identity, key-value, and object data across multiple regions.

Learn More

- For more information about the well-architected framework, see AWS Well-Architected.
Security in AWS Service Catalog

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

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

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

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

Topics
- Data Protection in AWS Service Catalog (p. 19)
- Identity and Access Management in AWS Service Catalog (p. 20)
- Logging and Monitoring in AWS Service Catalog (p. 26)
- Compliance Validation for AWS Service Catalog (p. 26)
- Resilience in AWS Service Catalog (p. 27)
- Infrastructure Security in AWS Service Catalog (p. 27)
- Security Best Practices for AWS Service Catalog (p. 27)

Data Protection in AWS Service Catalog

AWS Service Catalog conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN Partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

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

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
Protecting Data with Encryption

Encryption at rest

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

Encryption in transit

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

Identity and Access Management in AWS Service Catalog

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

Topics

- Audience (p. 21)
- Controlling Access (p. 21)
- Predefined AWS Managed Policies (p. 21)
- Console Access for End Users (p. 22)
Audience

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

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

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

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

Controlling Access

An AWS Service Catalog portfolio gives your administrators a level of access control for your groups of end users. When you add users to a portfolio, they can browse and launch any of the products in the portfolio. For more information, see the section called “Managing Portfolios” (p. 29).

Constraints

Constraints control which rules are applied to your end users when launching a product from a specific portfolio. You use them to apply limits to products for governance or cost control. For more information about constraints, see the section called “Using Constraints” (p. 37).

AWS Service Catalog launch constraints give you more control over permissions needed by an end user. When your administrator creates a launch constraint for a product in a portfolio, the launch constraint associates a role ARN that is used when your end users launch the product from that portfolio. Using this pattern, you can control access to AWS resource creation. For more information, see the section called “Launch Constraints” (p. 37).

Predefined AWS Managed Policies

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

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

**Administrators**

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

**End users**

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

**To attach a policy to an IAM user**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users**.
3. Choose the name (not the check box) of the IAM user.
4. On the **Permissions** tab, choose **Add permissions**.
5. On the **Add permissions** page, choose **Attach existing policies directly**.
6. Select the check box next to the managed policy for AWS Service Catalog, and then choose **Next: Review**.
7. On the **Permissions summary** page, choose **Add permissions**.
8. (Optional) You must grant administrators additional permissions for Amazon S3 if they need to use a private CloudFormation template. For more information, see **User Policy Examples** in the *Amazon Simple Storage Service Developer Guide*.

**Deprecated Policies**

The following managed policies are deprecated:

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

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

**To migrate from the deprecated policies to the current policies**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**.
3. In the search field, type **ServiceCatalog** to filter the policy list. Choose the name (not the check box) for **ServiceCatalogAdminFullAccess**.
4. For each attached entity (user, group, or role), do the following:
   a. Open the summary page for the entity.
   b. Add one of the current policies, as described in the procedure **To attach a policy to an IAM user** (p. 22).
   c. On the **Permissions** tab, next to **ServiceCatalogAdminFullAccess**, choose **Detach Policy**. When prompted for confirmation, choose **Detach**.
5. Repeat the process for **ServiceCatalogEndUserFullAccess**.

**Console Access for End Users**

The **AWSServiceCatalogEndUserFullAccess** and **AWSServiceCatalogEndUserReadOnlyAccess** policies grant access to the AWS Service Catalog end user console view. When a user who has either of these policies chooses AWS Service Catalog in the AWS Management Console, the end user console view displays the products they have permission to launch.
Before end users can successfully launch a product from AWS Service Catalog to which you give access, you must provide them additional IAM permissions to allow them to use each of the underlying AWS resources in a product's AWS CloudFormation template. For example, if a product template includes Amazon Relational Database Service (Amazon RDS), you must grant the users Amazon RDS permissions to launch the product.

To learn about how to enable end users to launch products while enforcing least-access permissions to AWS resources, see the section called “Using Constraints” (p. 37).

If you apply the `AWSServiceCatalogEndUserReadOnlyAccess` policy, your users have access to the end user console, but they won’t have the permissions that they need to launch products and manage provisioned products. You can grant these permissions directly to an end user using IAM, but if you want to limit the access that end users have to AWS resources, you should attach the policy to a launch role. You then use AWS Service Catalog to apply the launch role to a launch constraint for the product. For more information about applying a launch role, launch role limitations, and a sample launch role, see AWS Service Catalog Launch Constraints (p. 37).

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

**Product Access for End Users**

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

If you apply the `ServiceCatalogEndUserAccess` policy, your users have access to the end user console view, but they won't have the permissions that they need to launch products and manage provisioned products. You can grant these permissions directly to an end user in IAM, but if you want to limit the access that end users have to AWS resources, you should attach the policy to a launch role. You then use AWS Service Catalog to apply the launch role to a launch constraint for the product. For more information about applying a launch role, launch role limitations, and a sample launch role, see AWS Service Catalog Launch Constraints (p. 37).

**Example Policies for Managing Provisioned Products**

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

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

**Examples**
• Example: Full Admin Access to Provisioned Products (p. 24)
• Example: End-user Access to Provisioned Products (p. 24)
• Example: Partial Admin Access to Provisioned Products (p. 25)

Example: Full Admin Access to Provisioned Products

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

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

This policy is functionally equivalent to the following policy:

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

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

Example: End-user Access to Provisioned Products

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

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["servicecatalog:DescribeProduct",
          
```
Example: Partial Admin Access to Provisioned Products

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

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

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

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

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

You can also use notification constraints to set up Amazon SNS notifications about stack events. For more information, see the section called “Notification Constraints” (p. 39).

Compliance Validation for AWS Service Catalog

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

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

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

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

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

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
AWS Service Catalog Administrator Guide
Resilience

- **AWS Security Hub** – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

**Resilience in AWS Service Catalog**

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

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

In addition to the AWS global infrastructure, AWS Service Catalog offers AWS Service Catalog self-service actions. With self-service actions, customers can reduce administrative maintenance and end-user training while adhering to compliance and security measures. With self-service actions, as the administrator, you can enable end users to perform operational tasks such as backup and restore, troubleshoot issues, run approved commands, and request permissions in AWS Service Catalog. To learn more, see the section called “Using Service Actions” (p. 49).

**Infrastructure Security in AWS Service Catalog**

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

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

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

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

**Security Best Practices for AWS Service Catalog**

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

You can define rules that limit the parameter values that a user enters when launching a product. These rules are called template constraints because they constrain how the AWS CloudFormation template for the product is deployed. You use a simple editor to create template constraints, and you apply them to individual products.
AWS Service Catalog applies constraints when provisioning a new product or updating a product that is already in use. It always applies the most restrictive constraint among all constraints applied to the portfolio and the product. For example, consider a scenario where the product allows all Amazon EC2 instances to be launched and the portfolio has two constraints: one that allows all non-GPU type EC2 instances to be launched and one that allows only t1.micro and m1.small EC2 instances to be launched. For this example, AWS Service Catalog applies the second, more restrictive constraint (t1.micro and m1.small).

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

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

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

**Tasks**
- Managing Portfolios (p. 29)
- Managing Products (p. 33)
- Using AWS Service Catalog Constraints (p. 37)
- AWS Service Catalog Service Actions (p. 49)
- Adding AWS Marketplace Products to Your Portfolio (p. 53)
- Portfolio Sharing (p. 58)
- Using AWS CloudFormation StackSets (p. 60)
- Managing Budgets (p. 61)

Managing Portfolios

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

**Tasks**
- Creating, Viewing, and Deleting Portfolios (p. 29)
- Viewing Portfolio Details (p. 30)
- Creating and Deleting Portfolios (p. 30)
- Adding Products (p. 30)
- Adding Constraints (p. 32)
- Granting Access to Users (p. 33)

Creating, Viewing, and Deleting Portfolios

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

**To view the Portfolios page**

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

Viewing Portfolio Details

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

To view the Portfolio details page

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

Creating and Deleting Portfolios

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

To create a new portfolio

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

To delete a portfolio

1. Navigate to the Portfolios page.
2. Select the portfolio by clicking the corresponding radio button or anywhere on the listing except on the portfolio title.
3. Choose Delete portfolio.
4. Choose Continue.

Adding Products

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

Note

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

Adding a New Product

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

1. Navigate to the Portfolios page, and then choose the name of the portfolio to which you want to add the product.
2. On the portfolio details page, expand the Products section, and then choose Upload new product.
3. For Enter product details, enter the following:
   - **Product name** – The name of the product.
   - **Short description** – The short description. This description appears in search results to help the user choose the correct product.
   - **Description** – The full description. This description is shown in the product listing to help the user choose the correct product.
   - **Provided by** – The name or email address of your IT department or administrator.
   - **Vendor** (optional) – The name of the application's publisher. This field allows users to sort their products list to makes it easier to find the products that they need.

   Choose Next.

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

   Choose Next.

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

   Choose Next.

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

Adding an Existing Product

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

**To add an existing product to a portfolio**

1. Navigate to the Portfolios page.
2. Choose a portfolio, and then choose Add product.
3. Choose a product, and then choose Add product to portfolio.
Removing a Product from a Portfolio

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

To remove a product from a portfolio

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

Adding Constraints

To control how users are able to use products, add constraints. For more information about the types of constraints that AWS Service Catalog supports, see Using AWS Service Catalog Constraints (p. 37).

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

To add a constraint to a product

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose Portfolios and select a portfolio.
3. In the portfolio details page, expand the Constraints section and choose Add constraints.
4. For Product, select the product to which to apply the constraint.
5. For Constraint type, choose one of the following options:
   - Launch – The IAM role that AWS Service Catalog uses to launch and manage the product. For more information, see AWS Service Catalog Launch Constraints (p. 37).
   - Notification – The Amazon SNS topic specified to receive notifications. For more information, see AWS Service Catalog Notification Constraints (p. 39).
   - Template – A JSON–formatted text file that contains one or more rules. Rules are added to the AWS CloudFormation template used by the product. For more information, see Template Constraint Rules (p. 42).
   - Stack Set – Uses AWS CloudFormation StackSets to specify multiple accounts and regions for the AWS Service Catalog product launch. For more information, see AWS Service Catalog Stack Set Constraints (p. 40).
   - Tag Update – Allows you to update tags after the product has been provisioned. For more information, see AWS Service Catalog Tag Update Constraints (p. 40).
6. Choose Continue.

To edit a constraint

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

### Granting Access to Users

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

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

**To grant portfolio access to users or groups**

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

**Tip**

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

**To remove access to a portfolio**

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

### Managing Products

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

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

**Tasks**

- Viewing the Products Page (p. 33)
- Creating Products (p. 34)
- Adding Products to Portfolios (p. 35)
- Updating Products (p. 35)
- Deleting Products (p. 36)
- Managing Versions (p. 36)

### Viewing the Products Page

You manage products from the **Products** page in the AWS Service Catalog administrator console.
To view the Products page
1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose Service Catalog in the navigation bar.
3. Choose Products.

Creating Products

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

To create a new AWS Service Catalog product
1. Navigate to the Products page.
2. Choose Upload new product.
3. For Enter product details, enter the following:
   - Product name – The name of the product.
   - Short description – The short description. This description appears in search results to help the user choose the correct product.
   - Description – The full description. This description is shown in the product listing to help the user choose the correct product.
   - Provided by – The name of your IT department or administrator.
   - Vendor (optional) – The name of the application's publisher. This field allows users to sort their products list to makes it easier to find the products that they need.
4. Choose Next.
5. 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.
4. Choose Next.
6. For Version details, enter the following:
   - Select template – An AWS CloudFormation template from a local drive or a URL that points to a template stored in Amazon S3. If you specify an Amazon S3 URL, it must begin with https://. The extension for the template file must be .template.
   - Version title – the name of the product version (e.g., "v1", "v2beta"). No spaces are allowed.
   - Description (optional) – A description of the product version including how this version differs from the previous version.
   - Guidance – By default, product versions don't have any guidance, so end users can use that version to update and launch provisioned products. If you set the guidance to deprecated, users can make updates to a provisioned product but can't launch new provisioned products of that version.
7. Choose Next.
8. On the Review page, verify that the information is correct, and then choose Confirm and upload. After a few seconds, the product appears on the Products page. You might need to refresh your browser to see the product.
You can also use CodePipeline to create and configure a pipeline to deploy your product template to AWS Service Catalog and deliver changes you have made in your source repository. For more information, see Tutorial: Create a Pipeline That Deploys to AWS Service Catalog.

Adding Products to Portfolios

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

To add a product from your catalog to a portfolio

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

Updating Products

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

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

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

To create a new product version

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

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

Deleting Products

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

If your product has a budget associated to it, you will need to disassociate the budget before you can delete the product. For more information on disassociating a budget, see the section called “Managing Budgets” (p. 61).

To delete a product from your catalog

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

Managing Versions

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

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

Version Status

A version can have one of three statuses:

- **Active** - An active version appears in the version list and allows users to launch it.
- **Inactive** - An inactive version is hidden from the version list. Existing provisioned products launched from this version will not be affected.
- **Deleted** - If a version is deleted, it is removed from the version list. Deleting a version can't be undone.

Version Guidance

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

There are two options for version guidance:

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

Updating Versions

You assign product versions when creating a product, and you can also update a version any time. For more information about creating a product, see Creating Products (p. 34).
To update a product version

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

Using AWS Service Catalog Constraints

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

Constraints

- AWS Service Catalog Launch Constraints (p. 37)
- AWS Service Catalog Notification Constraints (p. 39)
- AWS Service Catalog Tag Update Constraints (p. 40)
- AWS Service Catalog Stack Set Constraints (p. 40)
- AWS Service Catalog Template Constraints (p. 41)

AWS Service Catalog Launch Constraints

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

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

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

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

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

Configuring a Launch Role

The IAM role that you assign to a product as a launch constraint must have permissions to use the following:
- AWS CloudFormation
- Services used in the AWS CloudFormation template for the product
- Read access to the AWS CloudFormation template in Amazon S3

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

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

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

**To attach a policy to the new role**
1. Choose the role that you created to view the role details page.
2. Choose the Permissions tab, and expand the Inline Policies section. Then, choose click here.
3. Choose Custom Policy, and then choose Select.
4. Enter a name for the policy, and then paste the following into the Policy Document editor:

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

```json
        "rds:CreateDBInstance",
        "rds:DeleteDBInstance",
```
6. Choose Apply Policy.

Applying a Launch Constraint

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

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

If you specify the role name, when an account uses the launch constraint, the IAM role with that name in the account will be used. This allows launch-role constraints to be account-agnostic so you can create fewer resources per shared account.

**Note**
The given role name must exist in the account used to create the launch constraint and the account of the user who launches a product with this launch constraint.

6. After specifying the IAM role, choose Create.

Verify That the Launch Constraint Is Applied

Verify that AWS Service Catalog uses the role to launch the product and that the provisioned product is created successfully by launching the product from the AWS Service Catalog console. To test a constraint prior to releasing it to users, create a test portfolio that contains the same products and test the constraints with that portfolio.

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

AWS Service Catalog Notification Constraints

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

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

2. Choose Create topic.
3. Type a topic name and then choose Create topic.
4. Choose Create subscription.
5. For Protocol, select Email. For Endpoint, type an email address that you can use to receive notifications. Choose Create subscription.
6. You’ll receive a confirmation email with the subject line AWS Notification - Subscription Confirmation. Open the email and follow the directions to complete your subscription.

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

To apply a notification constraint to a product

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

AWS Service Catalog Tag Update Constraints

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

To enable tag updates to a product

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose the portfolio that contains the product you want to update.
3. Choose the Constraints tab and choose Add constraints.
4. Under Constraint type, choose Tag Update.
5. Choose the product from Product, then choose Continue.
6. On the Tag Updates page, select Enable Tag Updates.
7. Choose Submit.

AWS Service Catalog Stack Set Set Constraints

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

A stack set constraint allows you to configure product deployment options using AWS CloudFormation StackSets. You can specify multiple accounts and regions for the product launch.
**To apply a stack set constraint to a product**

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose the portfolio that contains the product.
3. Expand **Constraints** and choose **Add constraints**.
4. Choose the product from **Product** and set **Constraint type** to **Stack Set**. Choose **Continue**.
5. On the **Stack Set constraint page**, enter a description.
6. Choose the account(s) in which you want to create products.
7. Choose the region(s) in which you want to deploy products. Products are deployed in these regions in the order that you specify.
8. Choose an IAM StackSet Administrator Role that will be used to manage your target accounts. If you don't choose a role, StackSets will use the default ARN. Learn more about setting up stack set permissions.
9. Choose **Submit**.

**AWS Service Catalog Template Constraints**

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

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

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

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

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

You define template constraint rules by using a rule editor or by writing the rules as JSON text in the AWS Service Catalog administrator console. For more information about rules, including syntax and examples, see **Template Constraint Rules** (p. 42).

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

**To apply template constraints to a product**

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. On the **Portfolios** page, choose the portfolio that contains the product to which you want to apply a template constraint.

3. Expand the **Constraints** section and choose **Add constraints**.

4. In the **Select product and type** window, for **Product** choose the product for which you want to define the template constraints. Then, for **Constraint type**, choose **Template**. Choose **Continue**.

5. On the **Template constraint builder** page, edit the constraint rules by using the JSON editor or the rule builder interface.

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

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

   When you have finished building a rule, choose **Add rule**. The rule appears in the table on the **Rule Builder** tab. To review and edit the JSON output, choose the **Constraint Text Editor** tab.

6. When you are done editing the rules for your constraint, choose **Submit**. To see the constraint, go to the portfolio details page and expand **Constraints**.

---

### Template Constraint Rules

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

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

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

### Syntax

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

A rule can include a **RuleCondition** property and must include an **Assertions** property. For each rule, you can define only one rule condition; you can define one or more asserts within the **Assertions** property. You define a rule condition and assertions by using rule-specific intrinsic functions, as shown in the following pseudo template:
"Rules" : {  
  "Rule01" : {  
    "RuleCondition" : { Rule-specific intrinsic function },  
    "Assertions" : [  
      {  
        "Assert" : { Rule-specific intrinsic function },  
        "AssertDescription" : "Information about this assert"  
      },  
      {  
        "Assert" : { Rule-specific intrinsic function },  
        "AssertDescription" : "Information about this assert"  
      }  
    ]  
  },  
  "Rule02" : {  
    "Assertions" : [  
      {  
        "Assert" : { Rule-specific intrinsic function },  
        "AssertDescription" : "Information about this assert"  
      }  
    ]  
  }  
}

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

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" : [{"Ref" : "InstanceType"}],
          "AssertDescription" : "For the prod environment, the instance type must be m1.large"
        }
      }
    ]
  }
}

AWS Service Catalog Rule Functions

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

Functions

- Fn::And (p. 44)
- Fn::Contains (p. 45)
- Fn::EachMemberEquals (p. 45)
- Fn::EachMemberIn (p. 46)
- Fn::Equals (p. 46)
- Fn::Not (p. 47)
- Fn::Or (p. 47)
- Fn::RefAll (p. 47)
- Fn::ValueOf (p. 48)
- Fn::ValueOfAll (p. 48)
- Supported Functions (p. 49)
- Supported Attributes (p. 49)

Fn::And

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

Declaration

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

Parameters

condition

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

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

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

**Fn::Contains**

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

**Declaration**

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

**Parameters**

- **list_of_strings**
  
  A list of strings, such as "A", "B", "C".

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

**Example**

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

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

**Fn::EachMemberEquals**

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

**Declaration**

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

**Parameters**

- **list_of_strings**
  
  A list of strings, such as "A", "B", "C".

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

**Example**

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

```
"Fn::EachMemberEquals" : [[list_of_strings], string]
```
"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. 49).

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. 49).

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::Value 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

AWS Service Catalog Service Actions

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

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

Prerequisites

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

Topics

- Step 1: Configure end user permissions (p. 50)
- Step 2: Create a service action (p. 51)
- Step 3: Associate the service action with a product version (p. 51)
- Step 4: Test the end user experience (p. 52)
- Step 5: Troubleshooting (p. 52)

Step 1: Configure end user permissions

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

To update permissions

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

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1536341175150",
      "Action": [
        "servicecatalog:ListServiceActionsForProvisioningArtifact",
        "servicecatalog:ExecuteprovisionedProductServiceAction",
        "ssm:DescribeDocument",
      ]
    }
  ]
}
```
Step 2: Create a service action

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

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/sc/.
2. From the menu, choose Service actions.
3. On the service actions page, choose Create new action.
4. On the Create action page, choose an AWS Systems Manager document to define the service action. The Amazon EC2 Instance Restart action is defined by an AWS Systems Manager document, so we keep the default option on the drop-down menu, Amazon documents.
5. Choose the AWS-RestartEC2Instance action.
6. Provide a name and description for the action that make sense for your environment and team. The end user will see this description, so choose something that helps them understand what the action does.
7. Under Parameter and target configuration, choose the SSM document parameter that will be the target of the action (for example, the Instance ID), and choose the target of the parameter. Choose Add parameter to add additional parameters.
8. Under Permissions, choose a role. We are using default permissions for this example. Other permission configurations are possible and are defined on this page.
9. After you have reviewed the configuration, choose Create action.
10. On the next page, a confirmation appears when the action has been created and is ready to use.

Step 3: Associate the service action with a product version

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

1. On the Service actions page, choose AWS-RestartEC2Instance, and then choose Associate action.
2. On the Associate action page, choose the product that you want your end users to take the service action on. In this example, we choose Linux Desktop.
3. Select a product version. Note that you can use the topmost check box to select all versions.
4. Choose Associate action.
5. On the next page, a confirmation message appears.
You have now created the service action in AWS Service Catalog. The next step of this tutorial is to use the service action as an end user.

**Step 4: Test the end user experience**

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

**To access the service action as an end user**

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

**Step 5: Troubleshooting**

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

**Note**

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

**Internal failure**

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

**An error occurred (ThrottlingException) when calling the StartAutomationExecution operation**

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

**Access denied while assuming the role**

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

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

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

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

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

**Document with that name does not exist**
Adding AWS Marketplace Products to Your Portfolio

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

AWS Marketplace is an online store in which you can find, subscribe to, and immediately start using a large selection of software and services. The types of products in AWS Marketplace include databases, application servers, testing tools, monitoring tools, content management tools, and business intelligence software. AWS Marketplace is available at https://aws.amazon.com/marketplace.

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

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

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

Managing and Adding AWS Marketplace Products Manually

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

To subscribe to an AWS Marketplace product

2. Browse the products or search to find the product that you want to add to your AWS Service Catalog portfolio. Choose the product to view the product details page.
3. Choose Continue to view the fulfillment page, and then choose the Manual Launch tab.
   The information on the fulfillment page includes the supported Amazon Elastic Compute Cloud (Amazon EC2) instance types, the supported AWS regions, and the Amazon Machine Image (AMI) ID that the product uses for each AWS region. Note that some choices will affect cost. You will use this information to customize the AWS CloudFormation template in later steps.
4. Choose Accept Terms to subscribe to the product.

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

To define your AWS Marketplace product in an AWS CloudFormation template

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

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

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

6. Customize the InstanceType parameter so that it includes only EC2 instance types that are supported by your product. If your template includes unsupported EC2 instance types, the product will fail to launch for your end users.
a. On the product fulfillment page in AWS Marketplace, view the supported EC2 instance types in the **Pricing Details** section, as in the following example:

![Pricing Details](image)

*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.

<table>
<thead>
<tr>
<th>EC2 Instance Type</th>
<th>EC2 Usage</th>
<th>Software</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1.micro</td>
<td>$0.02/hr</td>
<td>$0.00/hr</td>
<td>$0.02/hr</td>
</tr>
<tr>
<td>m1.small</td>
<td>$0.044/hr</td>
<td>$0.00/hr</td>
<td>$0.044/hr</td>
</tr>
<tr>
<td>m1.medium</td>
<td>$0.087/hr</td>
<td>$0.00/hr</td>
<td>$0.087/hr</td>
</tr>
<tr>
<td>m1.large</td>
<td>$0.175/hr</td>
<td>$0.00/hr</td>
<td>$0.175/hr</td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>$0.35/hr</td>
<td>$0.00/hr</td>
<td>$0.35/hr</td>
</tr>
<tr>
<td>m2.xlarge</td>
<td>$0.245/hr</td>
<td>$0.00/hr</td>
<td>$0.245/hr</td>
</tr>
<tr>
<td>m2.2xlarge</td>
<td>$0.49/hr</td>
<td>$0.00/hr</td>
<td>$0.49/hr</td>
</tr>
<tr>
<td>m2.4xlarge</td>
<td>$0.98/hr</td>
<td>$0.00/hr</td>
<td>$0.98/hr</td>
</tr>
<tr>
<td>c1.medium</td>
<td>$0.13/hr</td>
<td>$0.00/hr</td>
<td>$0.13/hr</td>
</tr>
<tr>
<td>c1.xlarge</td>
<td>$0.52/hr</td>
<td>$0.00/hr</td>
<td>$0.52/hr</td>
</tr>
<tr>
<td>hi1.4xlarge</td>
<td>$3.10/hr</td>
<td>$0.00/hr</td>
<td>$3.10/hr</td>
</tr>
<tr>
<td>hs1.8xlarge</td>
<td>$4.60/hr</td>
<td>$0.00/hr</td>
<td>$4.60/hr</td>
</tr>
<tr>
<td>m3.medium</td>
<td>$0.067/hr</td>
<td>$0.00/hr</td>
<td>$0.067/hr</td>
</tr>
<tr>
<td>m3.large</td>
<td>$0.133/hr</td>
<td>$0.00/hr</td>
<td>$0.133/hr</td>
</tr>
<tr>
<td>m3.xlarge</td>
<td>$0.266/hr</td>
<td>$0.00/hr</td>
<td>$0.266/hr</td>
</tr>
<tr>
<td>m3.2xlarge</td>
<td>$0.532/hr</td>
<td>$0.00/hr</td>
<td>$0.532/hr</td>
</tr>
<tr>
<td>c3.large</td>
<td>$0.105/hr</td>
<td>$0.00/hr</td>
<td>$0.105/hr</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>$0.21/hr</td>
<td>$0.00/hr</td>
<td>$0.21/hr</td>
</tr>
<tr>
<td>c3.2xlarge</td>
<td>$0.42/hr</td>
<td>$0.00/hr</td>
<td>$0.42/hr</td>
</tr>
<tr>
<td>c3.4xlarge</td>
<td>$0.64/hr</td>
<td>$0.00/hr</td>
<td>$0.64/hr</td>
</tr>
<tr>
<td>c3.8xlarge</td>
<td>$1.68/hr</td>
<td>$0.00/hr</td>
<td>$1.68/hr</td>
</tr>
</tbody>
</table>

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:

```json
"InstanceType" : {
    "Description" : "EC2 instance type",
    "Type" : "String",
    "Default" : "m1.small",
    "AllowedValues" : [ "t1.micro", "m1.small", "m1.medium", "m1.large", "m1.xlarge",
                      "m2.xlarge", "m2.2xlarge", "m2.4xlarge", "c1.medium", "c1.xlarge", "c3.large",
                      "c3.xlarge", "c3.2xlarge", "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:

```json
"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:
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](https://aws.amazon.com/documentation/cloudformation) to learn more about templates.

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

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 to your portfolio. It is now available to end users who have access to the portfolio.

**Portfolio Sharing**

To make your AWS Service Catalog products available to users who are not in your AWS account, such as users who belong to other organizations or to other AWS accounts in your organization, you share your portfolios with their AWS accounts.

When you share a portfolio, you allow an AWS Service Catalog administrator of another AWS account to import your portfolio into his or her account and distribute the products to end users in that account. This imported portfolio isn't an independent copy. The products and constraints in the imported portfolio stay in sync with changes that you make to the **shared portfolio**, the original portfolio that you shared. The recipient administrator, the administrator with whom you share a portfolio, cannot change the products or constraints, but can add AWS Identity and Access Management (IAM) access for end users. For more information, see **Granting Access to Users** (p. 33).

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

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

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

If you apply a launch constraint to a product in a shared portfolio, it propagates to all imported instances of the product. To override this launch constraint, the recipient administrator adds the product to a local portfolio and then applies a different launch constraint to it. The launch constraint that is in effect sets a launch role for the product. A **launch role** is an IAM role that AWS Service Catalog uses to provision AWS resources (such as EC2 instances or RDS databases) when an end user launches the product. As an administrator you can choose to designate a specific launch role ARN or a local role name. If you use the role ARN, the role will be used even if the end user belongs to a different AWS account than the one that owns the launch role. If you use a local role name, the IAM role with that name in the end user's account will be used. For more information about launch constraints and launch roles, see **AWS Service Catalog Launch Constraints** (p. 37). The AWS account that owns the launch role provisions the AWS resources, and this account incurs the usage charges for those resources. For more information, see **AWS Service Catalog Pricing**.

**Note**

You cannot re-share products from a portfolio that has been imported or shared.
Relationship Between Shared and Imported Portfolios

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

<table>
<thead>
<tr>
<th>Element of Shared Portfolio</th>
<th>Relationship to Imported Portfolio</th>
<th>Recipient Administrator Can</th>
<th>Recipient Administrator Cannot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Products and product versions</td>
<td>Inherited.</td>
<td>Add imported products to local portfolios. Products stay in sync with shared portfolio.</td>
<td>Upload or add products to the imported portfolio or remove products from the imported portfolio.</td>
</tr>
<tr>
<td></td>
<td>If the portfolio creator adds products to or removes products from the shared portfolio, the change propagates to the imported portfolio.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Launch constraints</td>
<td>Inherited.</td>
<td>In a local portfolio, the administrator can override the imported launch constraint by applying a different one to the product.</td>
<td>Add launch constraints to or remove launch constraints from the imported portfolio.</td>
</tr>
<tr>
<td></td>
<td>If the portfolio creator adds launch constraints to or removes launch constraints from a shared product, the change propagates to all imported instances of the product.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Template constraints</td>
<td>Inherited.</td>
<td>In a local portfolio, the administrator can add template constraints that take effect in addition to the imported constraints.</td>
<td>Remove the imported template constraints.</td>
</tr>
<tr>
<td></td>
<td>If the portfolio creator adds a template constraint to or removes a template constraint from a shared product, the change propagates to all imported instances of the product.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If the recipient administrator adds an imported product to</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Sharing a Portfolio**

To enable an AWS Service Catalog administrator for another AWS account to distribute your products to end users, share your AWS Service Catalog portfolio with that administrator’s AWS account.

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

**To share a portfolio**

2. On the **Portfolios** page, select the portfolio that you want to share, and choose **Share Portfolio**.
3. In the **Enter AWS account ID** window, type the account ID of the AWS account that you are sharing with. Then, choose **Share**. If sharing succeeds, a message on the **Portfolios** page confirms that the portfolio is linked with the target account. It also provides a URL that the recipient administrator must use to import the portfolio.
4. Send the URL to the AWS Service Catalog administrator of the target account. The URL opens the **Import Portfolio** page with the ARN of the shared portfolio automatically provided.

**Importing a Portfolio**

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

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

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

**Using AWS CloudFormation StackSets**

**Note**

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

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

**Stack sets vs. stack instances**

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

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

For more information, see StackSets Concepts.

**Stack set constraints**

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

For more information, see the section called “Stack Set Constraints” (p. 40).

**Managing Budgets**

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

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

**Tasks**

- Prerequisites (p. 61)
- Creating a Budget (p. 62)
- Associating a Budget (p. 63)
- Viewing a Budget (p. 63)
- Disassociating a Budget (p. 64)

**Prerequisites**

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

**Note**

Tags take up to 24 hours to activate.

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

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

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

To create a budget

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

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

   In order for spending that occurs on a portfolio or product to reflect on the associated budget, they must have the same tag. Note that a tag being used for the first time can take 24 hours to activate. For more information, see the section called “Prerequisites” (p. 61).

6. Choose **Continue**.
7. You will be taken to the **Set up your budget** page. Continue your budget setup by following the steps on Creating a Budget.

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

### Associating a Budget

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

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

**Note**

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

**To associate a budget**

2. Choose either **Products** or **Portfolios**.
3. Select the product or portfolio you want to associate a budget to.
4. Open the **Actions** menu, then choose **Associate budget**.
5. On the **Budget association** page, select an existing budget. Then choose **Continue**.
6. The **Portfolios** or **Products** table will now include data for the budget you just added.

### Viewing a Budget

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

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

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

You can disassociate a budget from a portfolio or product.

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

**To disassociate a budget**

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

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

Topics
- Managing All Provisioned Products as Administrator (p. 65)
- Changing Provisioned Product Owner (p. 65)
- Tutorial: Identifying User Resource Allocation (p. 66)

Managing All Provisioned Products as Administrator

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

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

Changing Provisioned Product Owner

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

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

To change the owner of a provisioned product
1. In the AWS Service Catalog console, choose Provisioned products list.
2. Locate the provisioned product you want to update, then choose the three dots beside it and choose Change provisioned product owner. You can also find the Change owner option on the provisioned product's detail page, in the Actions menu.
3. In the dialog box, enter the ARN of the user or role you want to set as the new owner. An ARN begins with `arn:` and includes other information separated by colons or slashes, for example, `arn:aws:iam::123456789012:user/NewOwner`.

4. Choose **Submit**. You will see a success message when the owner has been updated.

### See Also
- **UpdateProvisionedProductProperties**

### Tutorial: Identifying User Resource Allocation

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

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

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

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

![Provisioned products console](image)

2. In the **Provisioned products** pane, for **View**, choose **Account**.
3. Identify the provisioned product to investigate, and select the provisioned product.

4. Expand the Events section and note the Provisioned product ID and CloudformationStackARN values.

5. Use the provisioned product ID to identify the CloudTrail record that corresponds to this launch and identify the requesting user (typically, this is entered as an email address during federation). In this example, it is "steve".

```json
{
  "eventVersion":"1.03","userIdentity":
  {"type":"AssumedRole","principalId":"
  "arn":"
  "accountid":"
  "sessionContext":
  {"attributes":
  {"mfaAuthenticated":false,"creationDate":"
  },
  "sessionIssuer":
  {"type":"Role","principalId":""}
  }
  }
}
6. Use the CloudformationStackARN value to identify AWS CloudFormation events to find information about resources created. You can also use the AWS CloudFormation API to obtain this information. For more information, see AWS CloudFormation API Reference.
Note that you can perform steps 1 through 4 using the AWS Service Catalog API or the AWS CLI. For more information, see AWS Service Catalog Developer Guide and AWS Service Catalog Command Line Reference.
Managing Tags in AWS Service Catalog

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

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

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

Topics
- AWS Service Catalog AutoTags (p. 70)
- AWS Service Catalog TagOption Library (p. 71)

AWS Service Catalog AutoTags

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

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

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

AutoTag details

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

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

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

The TagOption library makes it easier to enforce the following:

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

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

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

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

Contents

- Launching a Product with TagOptions (p. 71)
- Managing TagOptions (p. 74)

Launching a Product with TagOptions

When a user launches a product that has TagOptions, AWS Service Catalog performs the following actions on your behalf:
• Collects all TagOptions for the product and the launching portfolio.
• Ensures that only TagOptions with unique keys are used in a tag on the provisioned product. Users get a multiple-choice value lists for a key. After the user chooses a value, it becomes a tag on the provisioned product.
• Allows users to add non-conflicting tags to the product during provisioning.

The following use cases demonstrate how TagOptions work during launch.

**Example 1: A Unique TagOption Key**

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

---

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

An administrator has placed two TagOptions with the same key on a portfolio, and there are no TagOptions with the same key on any products within that portfolio. During launch, the user must select one of the two values associated with the key. The provisioned product is then tagged with the key and the user-selected value.
Example 3: A Set of TagOptions with the Same Key on Both the Portfolio and a Product in that Portfolio

An administrator has placed several TagOptions with the same key on a portfolio, and there are also several TagOptions with the same key on the product within that portfolio. AWS Service Catalog creates a set of values from the aggregation (logical AND operation) of the TagOptions. When the user launches the product, he or she sees and selects from this set of values. The provisioned product is tagged with the key and the user-selected value.
Example 4: Multiple TagOptions with the Same Key and Conflicting Values

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

Managing TagOptions

As an administrator, you can create, remove, and edit TagOptions, and associate and disassociate TagOptions with a portfolio or product.

To create a TagOption (console)

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

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

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

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

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

Alternatively, from a portfolio or product detail page, choose Add TagOption and select the TagOption to associate the TagOption with.
3. Choose Save.

To associate a TagOption with a portfolio or product using the AWS Service Catalog API, see AssociateTagOptionWithResource.
To remove (disassociate) a TagOption from a portfolio or product (console)

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

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

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

To edit a TagOption

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

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

For a list of available metrics and dimensions, see AWS Service Catalog CloudWatch Metrics (p. 76).

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

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

Monitoring Tools

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

Automated Monitoring Tools

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

- Amazon CloudWatch alarms – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Amazon EC2 Auto Scaling policy. CloudWatch alarms do not invoke actions simply because they are in a particular state; the state must have changed and been maintained for a specified number of periods. To learn how to create an alarm, see Creating Amazon CloudWatch Alarms. For more information on using Amazon CloudWatch metrics with AWS Service Catalog, see AWS Service Catalog CloudWatch Metrics (p. 76).

AWS Service Catalog CloudWatch Metrics

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

Topics
- Enabling CloudWatch Metrics (p. 77)
- Available Metrics and Dimensions (p. 77)
- Viewing AWS Service Catalog Metrics (p. 78)

Enabling CloudWatch Metrics

Amazon CloudWatch metrics are enabled by default.

Available Metrics and Dimensions

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

AWS Service Catalog Metrics

The AWS/ServiceCatalog namespace includes the following metrics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProvisionedProductLaunch</td>
<td>The number of provisioned products launched for a given product and provisioning artifact in a specified time period. Units: Count Valid statistics: Minimum, Maximum, Sum, Average</td>
</tr>
</tbody>
</table>

Dimensions for AWS Service Catalog Metrics

AWS Service Catalog sends the following dimensions to CloudWatch.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>This dimension filters the data you request for all provisioned products launched with this specified state. This helps you categorize your data by the state of launch. Valid State: SUCCEEDED, FAILED</td>
</tr>
<tr>
<td>ProductId</td>
<td>This dimension filters the data you request for the identified product id only. This helps you to pinpoint an exact product from which to be launched.</td>
</tr>
<tr>
<td>ProvisioningArtifactId</td>
<td>This dimension filters the data you request for the identified provisioning artifact id only. This helps you to pinpoint an exact version of products from which to be launched.</td>
</tr>
</tbody>
</table>
Viewing AWS Service Catalog Metrics

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

Topics
- Viewing AWS Service Catalog Metrics in the CloudWatch Console (p. 78)

Viewing AWS Service Catalog Metrics in the CloudWatch Console

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

To view metrics in the CloudWatch console

2. In the Metrics section in the left navigation, choose Service Catalog.
3. Choose the metrics to view.
Product and Service Integrations with AWS Service Catalog

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

Topics
- AWS Service Catalog Connector for ServiceNow (p. 79)
- AWS Service Management Connector for Jira Service Desk (p. 98)

AWS Service Catalog Connector for ServiceNow

To help customers integrate provisioning secure, compliant, and pre-approved AWS products into their ServiceNow portal, AWS created the AWS Service Catalog Connector for ServiceNow.

AWS Service Catalog Connector for ServiceNow synchronizes AWS Service Catalog portfolios and products with the ServiceNow Service Catalog to enable ServiceNow users to request approved AWS products via ServiceNow.

Topics
- Background (p. 79)
- Getting Started (p. 80)
- Release Notes (p. 80)
- Baseline Permissions (p. 81)
- Configuring AWS Service Catalog (p. 86)
- Creating StackSet Constraints (p. 61)
- Relating Budgets to Products and Portfolios (p. 87)
- Configuring ServiceNow (p. 87)
- Validating Configurations (p. 92)
- ServiceNow Additional Administrator Features (p. 93)
- Upgrade Instructions (p. 94)

Background

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

ServiceNow is an enterprise service management platform that places a service-oriented lens on the activities, tasks, and processes that make up day-to-day work life to enable a modern work environment. ServiceNow Service Catalog is a self-service application that end users can use to order IT services based on request fulfillment approvals and workflows.
Getting Started

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

For a video showing how to integrate AWS products into your ServiceNow portal, see Integrate AWS products into Your ServiceNow Portal.

AWS prerequisites

To get started you need an AWS account to configure your AWS portfolios and products. For details, see Setting Up for AWS Service Catalog (p. 6).

For each AWS account, the Connector for ServiceNow also requires two AWS Identity and Access Management (IAM) users and two IAM roles:

- An IAM user to sync AWS Service Catalog portfolios and products to ServiceNow Service Catalog items.
- An IAM role configured as an AWS Service Catalog end user and assigned to each portfolio.
- An IAM end user to assume the previous end user role. This end user has a baseline of permissions to provision AWS services in the ServiceNow Service Catalog. This ServiceNow end user is linked to the end user role in IAM.
- An IAM launch role used to place baseline AWS service permissions into the AWS Service Catalog launch constraints. Configuring this role enables segregation of duty by provisioning product resources on behalf of the ServiceNow end user.

Baseline permissions enable an end user to provision the following AWS services: Amazon Simple Storage Service and Amazon Elastic Compute Cloud. To allow end users to provision AWS services beyond the baseline permissions, you must include the additional AWS service permissions to the launch role. For information about initial permissions setup actions, see the section called “Baseline Permissions” (p. 81).

Note

To use an AWS CloudFormation template to set up the AWS configurations of the Connector for ServiceNow, see the two JSON AWS Configurations for: Connector for ServiceNow v2.3.4 - AWS Commercial Regions and Connector for ServiceNow v2.3.4 - AWS GovCloud West Region.

ServiceNow Prerequisites

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

Release Notes

Version 2.3.4 of the AWS Service Catalog Connector for ServiceNow includes a fix to the ServiceNow platform regression supporting the scoped app call Object.getPrototypeOf(…)
return an object safe for cross-scope access. Prior to this fix, customers would receive a prototype_not_allowed error when validating regions within the AWS Service Catalog Connector on the latest ServiceNow platform releases or patches (Orlando, New York, and Madrid).

This version also includes prior AWS Service Catalog Connector for ServiceNow features such as:

- The ability for administrators to view portfolio and product budgets and actual costs. (Requires budgets to be associated within AWS Service Catalog.)
• Support for AWS GovCloud West region.
• The ability for end users to choose accounts and regions for StackSet deployments.
• The ability to view provisioned product events and outputs in the ServiceNow request item. This includes closure of ServiceNow request items when products are terminated.

This version also includes prior AWS Service Catalog Connector for ServiceNow features such as:

• Support for AWS CloudFormation StackSets, enabling launch of AWS Service Catalog products across multiple regions and accounts.
• Support for AWS CloudFormation Change Sets, enabling a preview of resource changes from a launch or update.
• Display of AWS Service Catalog portfolios (including correlated products) as sub-categories in the ServiceNow Service Catalog.
• Support AWS Service Catalog self-service actions using AWS Systems Manager documents.
• Support AWS Service Catalog post-provision operational actions to update and terminate products.
• Rendering of AWS Service Catalog products in the ServiceNow Portal page.
• Multi-account support.
• Validation of AWS Regions and identities associated with syncing AWS and ServiceNow.
• Synchronization of product details in the My Asset/CMDB view.

Baseline Permissions

This section provides instructions on how to set up the baseline AWS users and permissions needed for the AWS Service Catalog Connector for ServiceNow. For each AWS account, the Connector for ServiceNow requires two IAM users and three roles:

• **AWS Service Catalog Sync User**: IAM user to sync AWS portfolios and products to ServiceNow catalog items (**AWSServiceCatalogAdminReadOnlyAccess** managed policy).
• **AWS Service Catalog End User role**: IAM role configured as an AWS Service Catalog end user and assigned to each AWS Service Catalog portfolio.
• **AWS Service Catalog End User**: Enables Connector for ServiceNow to provision AWS products by assuming a role that contains the trust relationship with the account and policies needed for the end user privileges in AWS Service Catalog.
• **SCConnect Launch role**: IAM role used to place baseline AWS service permissions into the AWS Service Catalog launch constraints. Configuring this role enables segregation of duty through provisioning product resources on behalf of the ServiceNow end user. The SCConnectLaunch role baseline contains permissions to Amazon EC2 and Amazon S3 services. If your products contain more AWS services, you must either include those services in the SCConnectLaunch role or create new launch roles.

Creating AWS Service Catalog Sync User

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

**To create AWS Service Catalog sync user**

1. Go to Creating IAM Policies. Following the instructions there, create a policy called **SCConnectorAdmin** for ServiceNow administrators to delete AWS Service Catalog products in ServiceNow that do not have self-service actions associated. ServiceNow administrators can also view budgets associated to AWS Service Catalog portfolios and products. Copy the following policy and paste it into **Policy Document**:
2. Go to Creating an IAM User in Your AWS Account. Following the instructions there, create a sync user (that is, SCSyncUser). The user needs programmatic and AWS Management Console access to follow the Connector for ServiceNow installation instructions.

3. Set permissions for your sync user (SCSyncUser). Choose Attach existing policies directly and select the AWSServiceCatalogAdminReadOnlyAccess and SCConnectorAdmin policies.

   **Note**
   
   The ServiceCatalogAdminReadOnlyAccess policy was deprecated. If you are using a current version of the Connector for ServiceNow, update your SCSyncUser with the correct managed policy: AWSServiceCatalogAdminReadOnlyAccess.

4. Review and choose Create User.

5. Note the Access and Secret Access information. Download the .csv file that contains the user credential information.

### Creating AWS Service Catalog End User

The following section describes how to create the AWS Service Catalog end user and associate the appropriate IAM permission. This AWS end user (SCEndUser) requires you to first create an AWS role (such as SnowEndUser) with the required IAM permissions. The AWS end user will assume the AWS role.

To perform this task, you need IAM permissions to create new users.

If you are upgrading from an earlier version of the Connector, note that the ServiceCatalogServiceNowAdditionalPermissions AWS policy is no longer needed for the Connector for ServiceNow. Proceed to the Create a SnowEndUser role step.

**To create AWS Service Catalog end user**

1. You need to first create the AWS role (such as SnowEndUser). Go to Create a role.

   For products using AWS CloudFormation StackSets, you need to create a StackSet inline policy. With AWS CloudFormation StackSets, you are able to create products that are deployed across multiple accounts and regions. Using an administrator account, you define and manage an AWS Service Catalog product, and use it as the basis for provisioning stacks into selected target accounts across specified regions. You need to have the necessary permissions defined in your AWS accounts.

   To set up the necessary permissions, go to Granting Permissions for Stack Set Operations. Following the instructions there, create an AWSCloudFormationStackSetAdministrationRole and an AWSCloudFormationStackSetExecutionRole.
Create the StackSet inline policy to enable provisioning a product across multiple regions within one account.

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

**Note**
Replace 123456789123 with your account information. The Connector for ServiceNow v2.3.4 - AWS Commercial Regions and Connector for ServiceNow v2.3.4 - AWS GovCloud West Region files include the stack set permissions.

2. Add the following permissions (policies) to the SnowEndUser role:

- **AWSServiceCatalogEndUserFullAccess** - Note: The ServiceCatalogEndUserFullAccess policy was deprecated. If you are using a current version of the Connector for ServiceNow, update the SCSyncUser with the correct AWS managed policy.
- **StackSet (inline policy)**
- **AmazonEC2ReadOnlyAccess**
- **AmazonS3ReadOnlyAccess** - Note: For AWS Service Catalog products using AWS CloudFormation StackSets, you need to modify the SnowEndUser role to include the ReadOnly permissions for the service(s) you want to provision. For example, to provision an Amazon S3 bucket, include the AmazonS3ReadOnlyAccess policy to the SnowEndUser role.
- Create a trust relationship on the SnowEndUser role to the account. Copy and paste the following text into the Trust Relationship (replacing the number string for ARN with your account information):

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789123:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}  
    }
  ]
}
```
3. Go to Create a policy. Following the instructions there, create a policy called StsAssume-SC. Copy and paste the following text into the JSON editor (replacing the number string for ARN with your account information):

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam:: 123456789123:role/SnowEndUser"
    }
  ]
}
```

4. Go to Creating an IAM User in Your AWS Account. Following the instructions there, create a user (such as SCEndUser). The user needs programmatic and AWS Management Console access to follow the ServiceNow Connector installation instructions.

5. Attach the assume policy (StsAssume-SC) to your end user (SCEndUser). Choose Attach existing policies directly and select StsAssume-SC.

6. Review and choose Create User.

7. Note the Access and Secret Access information. Download the .csv file that contains the user credential information.

Creating SCConnectLaunch Role

The following section describes how to create the SCConnectLaunch role. This role is used to place baseline AWS service permissions into the AWS Service Catalog launch constraints. For more information, see AWS Service Catalog Launch Constraints.

To create SCConnectLaunch role

1. Create the AWSCloudFormationFullAccess policy. Choose create policy and then paste the following in the JSON editor:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:DescribeStackResource",
        "cloudformation:DescribeStackResources",
        "cloudformation:GetTemplate",
        "cloudformation:List*",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStacks",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStackEvents",
```
"cloudformation:DescribeStacks",
"cloudformation:GetTemplateSummary",
"cloudformation:SetStackPolicy",
"cloudformation:ValidateTemplate",
"cloudformation:UpdateStack",
"cloudformation:CreateChangeSet",
"cloudformation:DescribeChangeSet",
"cloudformation:ExecuteChangeSet",
"cloudformation:DeleteChangeSet",
"s3:GetObject"
],
"Resource": "*"
}

Note
AWSCloudFormationFullAccess now includes additional permissions for ChangeSets.

2. Create a policy called ServiceCatalogSSMActionsBaseline. Follow the instructions on Creating IAM Policies, and paste the following into the JSON editor:

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

3. Create the SCConnectLaunch role. Assign the trust relationship to AWS Service Catalog.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
            "Service": "servicecatalog.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
      }
   ]
}
```
4. Attach the relevant policies to the **SCConnectLaunch** role. Attach the following baseline IAM policies:

   - **AmazonEC2FullAccess** (AWS managed policy)
   - **AmazonS3FullAccess** (AWS managed policy)
   - **AWSCloudFormationFullAccess** (custom managed policy)
   - **ServiceCatalogSSMActionsBaseline**

### Configuring AWS Service Catalog

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

**To configure AWS Service Catalog**

1. Create a portfolio by following the steps at Create an AWS Service Catalog Portfolio (p. 12).
2. To add the Amazon S3 bucket product to the portfolio you just created, in the AWS Service Catalog console, on the **Upload new product** page, enter product details.
3. For **Select template**, choose the Amazon S3 bucket AWS CloudFormation template you saved to your device.
4. Set **Constraint type** to **Launch** for the product that you just created with the SCConnectLaunch role in the baseline permissions. For additional launch constraint instructions, see AWS Service Catalog Launch Constraints (p. 37).

   **Note**

   The AWS configuration design requires each AWS Service Catalog product to have a launch constraint. Failure to follow this step may result in an “Unable to Retrieve Parameter” message within ServiceNow Service Catalog.

5. Add the SnowEndUser IAM role to the AWS Service Catalog portfolio. For additional user access instructions, see Granting Access to Users (p. 33).

### Creating StackSet Constraints

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

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

Note that the Connector for ServiceNow v2.3.4 - AWS Commercial Regions and Connector for ServiceNow v2.3.4 - AWS GovCloud West Region templates create the permissions as well as outputs needed for stack set constraints. Example stack set outputs:

<table>
<thead>
<tr>
<th>Role Name</th>
<th>ARN</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCStackSetAdministratorRoleARN</td>
<td>arn:aws:iam::123456789123:role/AWSCloudFormationStackSetAdministrationRole</td>
</tr>
<tr>
<td>SCIAMStackSetExecutionRoleName</td>
<td>AWSCloudFormationStackSetExecutionRole</td>
</tr>
<tr>
<td>SCIAMAdminRoleARN</td>
<td>arn:aws:iam::123456789123:role/AWSCloudFormationStackSetAdministrationRole</td>
</tr>
</tbody>
</table>

Note
Replace the 123456789123 with your account information.

Relating Budgets to Products and Portfolios

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

For more information on creating and associating budgets, see the section called “Managing Budgets” (p. 61).

Configuring ServiceNow

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

- Clear the ServiceNow platform cache.
- Clear the web browser cache.
- Install the ServiceNow Connector scoped application, and upload and commit the ServiceNow Connector Update Set.
- Configure ServiceNow platform system admin components.
- Configure AWS Service Catalog Connector scoped application, including accounts, scheduled jobs sync, and permissions.

Clear the ServiceNow Platform Cache

Before installing the AWS Service Catalog scoped app, we recommend that you clear the ServiceNow platform cache by typing in the following URL: https://[InsertServiceNowInstanceNameHere]/cache.do

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

Clear the Web Browser Cache

Clear the web browser cache to clear previous rendered product forms.
Installing ServiceNow Connector Scoped Application

The AWS Service Catalog Connector for ServiceNow is released as a conventional ServiceNow scoped application via a ServiceNow Update Set. ServiceNow update sets are code changes to the out-of-the-box platform and enable developers to move code across ServiceNow instance environments. The Connector for ServiceNow update set is available to download in the ServiceNow store. For users installing the update set on a ServiceNow Personal Developer Instance (PDI), download the code from Connector for ServiceNow version 2.3.4 update set.

The Connector for ServiceNow version 2.3.4 update set may be applied to a "Madrid," "New York," or "Orlando" platform release of ServiceNow.

If you do not already have a ServiceNow instance, begin with the first step below. If you already have a ServiceNow instance, proceed to To download AWS Service Catalog Connector for ServiceNow.

To obtain a ServiceNow instance

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

To download AWS Service Catalog Connector for ServiceNow

1. From your ServiceNow dashboard, type plugins into the navigation panel in the upper left.
2. When the System Plugins page populates, next to the dropdown that says Name, search for user criteria.
3. Choose User Criteria Scoped API and then choose Activate.
4. From the ServiceNow Store, download the AWS Service Catalog Connector. When prompted, log in with your administrator credentials.

To install the update set

1. From your ServiceNow dashboard, type update sets into the navigation panel in the upper left.
2. Choose Retrieved Update Sets from the results.
3. Select Import Update Set from XML and upload the release XML file.
4. Select the AWS Service Catalog - 2.3.4 update set.
5. Choose Preview Update Set, which makes ServiceNow validate the connector update set.
6. Choose Update.
7. Choose Commit Update Set to apply the update set and create the application. This procedure should complete 100%.

Configuring ServiceNow Platform System Admin Components

To enable the AWS Service Catalog Connector for ServiceNow scoped application named AWS Service Catalog, the system admin must configure specific platform tables, forms, and views.

Note
If you are upgrading from an earlier version, the permissions on ServiceNow Platform tables (User Criteria, Catalog Variable Set, and Category) are no longer needed for the Connector for ServiceNow.
Enable permissions on ServiceNow Platform table (Catalog Item Category)

For AWS products to display under AWS portfolios as sub-categories in the ServiceNow Service Catalog, you need to modify the Application Access form for Catalog Item Category tables. This action is necessary because a ServiceNow scoped API is not available for the Catalog Item Category table.

1. Enter “Tables” in the Navigator and choose System Definition, then choose Tables.
2. In the list of tables, search for a table with label “Catalog Item Category” (or with the name “sc_cat_item_category”). The list of tables will be displayed. Choose Category to view the form defining the table.
3. Choose the “Application Access” tab on the form and choose the “Can Create”, “Can Update, and “Can delete” checkboxes on the form. Choose the ”Update“ button.

ServiceNow Permissions for Administrators of the ConnectorScoped App.

The AWS Service Catalog scoped app comes with two ServiceNow roles that enable access to configure the application. This enables system admins to grant one or more users privileges to administer the application without having to open up full sysadmin access to them. These roles can be assigned either to individual users or to one administrator user.

To set up application administrator privileges

1. Type Users in the navigator and select System Security – Users.
2. Select a user to grant one or both previous roles (such as admin) to. You can also Create a User.
3. Choose Edit on the Roles tab of the form.
4. Filter the collection of roles by the prefix “x_.”.
5. Choose one or both of the following and add them to the user: x_126749_aws_sc_account_admin, x_126749_aws_sc_portfolio_manager
6. Choose Save.

To add AWS Service Catalog to ServiceNow Service Catalog categories

1. Navigate to Self Service | Service Catalog and select the Add content icon in the upper right.
2. Select the AWS Service Catalog Product entry. Add it to your catalog home page by choosing the first Add Here link on the second row of the selection panel at the bottom of the page.

To add a change request type

1. If you are upgrading from a previous version of the AWS Service Catalog scoped app, you must remove the AWS Product Termination change request type before creating a new change request type.
2. You must add a new change request type called AWS Provisioned Product Event for the scoped application to trigger an automated change request in Change Management. For instructions, see Add a new change request type.
3. Open an existing change request.
4. Open the context (right-click) menu for Type and then choose Show Choice List.
5. Choose New and fill in the following fields:
   - Table: Change Request
   - Label: AWS Provisioned Product Event
   - Value: AWSProvisionedProductEvent
6. Submit the form.

Configuring AWS Service Catalog Connector Scoped Application

Having installed and configured the AWS Service Catalog Connector for ServiceNow in the previous procedure, you must configure the AWS Service Catalog scoped application and applicable roles.

To configure the AWS Service Catalog scoped application and applicable roles

1. On your ServiceNow dashboard, create a role called `order_aws_sc_products`. This role is granted to any users with permission to order AWS Service Catalog products. For instructions, see Create a role.
2. Grant roles to the following users:
   - **System Administrator (admin)**: For simplicity in this example, user `admin` is designated as the administrator of the AWS Service Catalog scoped application. Grant this user both of the administrative permissions from the adapter, `x_126749_aws_sc_portfolio_manager` and `x_126749_aws_sc_account_admin`. In a real scenario, these roles would likely be granted to two different users.
   - **Abel Tuter**: The user `abel.tuter` is chosen as an illustrative end user. Grant Abel the new role `order_aws_sc_products`. This allows him to order products from AWS.

Fix Script for AWS Account Type Entries

Version 2.3.4 of the Connector for ServiceNow enables ServiceNow administrators to identify AWS account type entries as End User or Sync User. If you’ve installed previous versions of the Connector for ServiceNow, version 2.3.4 comes with a ServiceNow Fix Script that automatically updates the account type entries.

To run the Fix Script

1. In the ServiceNow dashboard, in the left menu, search for Fix scripts.
2. Find the script named `Update AWS account type`.
3. Choose Run Fix Script.

Configuring Accounts

1. Log in as the system administrator.
2. In the AWS Service Catalog scoped app Accounts menu, create two entries for every AWS account, one for sync and one for provisioning: `snow-stsuser-account` and `snow-sync-account`. You need to use the keys and secret keys from the users you created in AWS. Note that the names here are chosen for convenience to make it easy to see which IAM user they correspond to (these are the users created in the AWS setup).

   For the GovCloud regions and account entry type "Type" are required for the Connector to point to the appropriate GovCloud endpoints. There are two new regions available: **US GovCloud (US West) - FIPS 2**, and **US GovCloud (US West)**, used for Non-FIPS.

   To create SyncUser account entry

1. Enter the name as an account entry identifier, such as `snow-sync-account` (for Commercial region), or `snow-sync-account_GovCloud` (for GovCloud region).
2. Enter AWS access key and secret access key from the AWS account sync user IAM configurations.
3. Enter account entry type of **Sync User**. Note that this feature was added to address the unique GovCloud endpoints.
4. Choose the Commercial or GovCloud region. Validate this in the next section.
5. Save or update the sync user information.

**To create EndUser account entry**

1. Enter the name as an account entry identifier, such as **snow-stsuser-account** (for Commercial region), or **snow-stsuser-account_GovCloud** (for GovCloud region).
2. Enter AWS access key and secret access key from the AWS account sync user IAM configurations.
3. Enter account entry type of **End User**. Note that this feature was added to address the unique GovCloud endpoints.
4. For GovCloud users only: select the GovCloud region. This is required to address the unique service endpoints in GovCloud.
   
   **Note**
   
   For commercial regions, do not select a region in the End User account entry.
5. Save or update the end user information.

**Validating Connectivity to AWS Regions**

You can now validate connectivity to AWS regions between the ServiceNow snow-sync-account and the AWS IAM SyncUser.

**To validate connectivity to AWS regions**

1. In the AWS Service Catalog scoped app, choose **Accounts**.
2. Select **snow-sync-account** and choose **Validate Regions**.
3. A successful connection result in the message, "Successfully performed AWS Service Catalog SearchProductsAsAdmin action in each referenced Region."

If the AWS IAM access key or secret access key are incorrect, you will receive the message similar to the following: *Error performing AWS Service Catalog SearchProductsAsAdmin action in one or more Regions: us-east-1: The security token included in the request is invalid. Check that the access key and secret access key are correct.*

**Manually Syncing Scheduled Jobs**

During the initial setup, manually execute the sync instead of waiting for Scheduled Jobs to run.

**To sync the accounts manually**

1. Log in as system administrator.
2. Find **Scheduled Jobs** in the navigator panel.
3. Search for job **Sync all Accounts**, select it, and choose **Execute Now**.
   
   **Note**
   
   If you do not see **Execute Now** in the upper left corner, choose **Configure Job Definition**. **Execute Now** will be visible.
Granting Access to Portfolios

Data is visible in the AWS Service Catalog scoped app menus after the adapter’s scheduled synchronization job has run.

To grant access to AWS Service Catalog products in ServiceNow, you must establish a link between the AWS SnowEndUser role discovered from the Sync All Scheduled Job and snow-stsuser-account entry created in the ServiceNow AWS Service Catalog scoped app.

To grant access to AWS Service Catalog products in ServiceNow

1. In the AWS Service Catalog scoped app, choose the Identities module.
2. Select the ARN address for the AWS SnowEndUser role and assign it to account snow-stsuser-account. You can double-click the cell in the account column, or click the SConedUser user name and edit the form presented.
   
   Role Grants is available within the Identities modules to conveniently associate the ServiceNow role order_aws_sc_products to the AWS SnowEndUser role identity.
3. Choose New and enter the Role of order_aws_sc_products and the SnowEndUser identity.
4. Choose Submit.

The Identities module now has a view of the associated role. You can test the AWS identity to determine if the ServiceNow end user with the order_aws_sc_products role can order an AWS Service Catalog product.

To test access to portfolios

1. Choose the Test Authorization button shown in the AWS identity module.
2. If the test is successful, the message Successfully performed SearchProducts action as arn:aws:iam::AWS Account:role/SnowEndUser is returned.
3. An unsuccessful test returns the message Error using account...
4. Given the preceding setup, Abel Tuter can now order products from AWS Service Catalog in ServiceNow.

Validating Configurations

You are now ready to validate the AWS Service Catalog Connector for ServiceNow installation procedures.

To validate the configuration of the Connector

1. Log into your ServiceNow instance as the end user (for example, Abel Tuter).
2. Type Service Catalog in the navigation filter and choose Service Catalog.
3. The user interface view displays the AWS Service Catalog category.

To order a product

1. Select the AWS Service Catalog S3 Storage product to provision.
2. Fill in the product request details including product name, parameters, and tags.
3. Choose Order Now to submit the ServiceNow request and provision the AWS Service Catalog product.
4. After approximately one minute, you receive an order status indicating that your request was submitted.
To view provisioned products

End users can view products in two places on the ServiceNow portal through request items (Requests) or My AWS Products widgets.

To view products in Service Portal Requests

1. Choose Requests in the home page navigation bar.
2. Select the request item of your choice (contains the AWS Service Catalog product and request item number).

   **Note**
   The request item is updated with AWS product events and outputs. When the AWS product is terminated, the ServiceNow request item will go into a state of Closed Complete.

To view products in the My AWS Products widget Service Portal Requests

1. Go to the My AWS Products widget
2. Choose the AWS Select Product name that you entered into the request form.
3. View the Status and Product Events
4. If you want to perform post-provisioned operational actions, choose Request Update, Request Self-Service Action, or Terminate.

ServiceNow Additional Administrator Features

This section provides information about additional administrator features for the AWS Service Catalog Connector for ServiceNow.

Deleting AWS Service Catalog Products

The Connector for ServiceNow enables ServiceNow administrators with x_126749_aws_sc_account_admin permission the ability to delete AWS Service Catalog products that do not have self-service actions associated.

   **Note**
   You must disassociate self-service actions from AWS Service Catalog products within the AWS Management Console before managing products with the ServiceNow platform.

To delete AWS Service Catalog products

1. In the Connector, go to AWS Service Catalog - Products. Choose the AWS Service Catalog product to delete.
2. Choose Manage Product.
3. Choose Delete Product.
4. A warning appears. Choose OK.
5. After the deletion is complete, a message appears telling you the product has been deleted.

Ordering AWS Service Catalog Products Through the ServiceNow Service Portal

The Connector for ServiceNow version 1.6.7 and above supports ordering AWS Service Catalog products through Service Portal by using the Service Catalog and Order Something views. The release also includes pages and widgets that you can add to Service Portal that enable users to view their provisioned products.
Note
The audience for the Service Portal Features section is a ServiceNow administrator or equivalent. The ServiceNow user requires permissions to modify the Service Portal.

Service Portal Widgets

The Connector for ServiceNow includes widgets that you can add to your Service Portal. It also includes two alternative view Portal Pages for the following:

- **My AWS Products** – Provides an overview of all provisioned products owned by the user.
- **AWS Product Details** – Provides details of a single provisioned product.

To access the new widgets, you need to update the Service Portal Designer.

**To update the Service Portal Designer**

1. Go to Create and edit a page using the Service Portal Designer.
2. Following the instructions, choose the Service Portal Index page.
3. Under the Order Something container, add the My AWS Products widget.
4. The new widget appears on your main Service Portal view.

Service Portal Pages

The following section describes the two new pages available in the Service Portal Beta release of the AWS Service Catalog Connector, **My AWS Products** and **AWS Product Details**. You can add links to these pages on the Service Portal home page or other pages by using the usual page configuration mechanism in Service Portal.

**My AWS Products**

Provides an overview of all provisioned products owned by the user. Terminated products are displayed separately from current products in a panel that is collapsed on initial page load.

The **My AWS Products** page is available using the following format:

http://<insertinstancename>.service-now.com/sp?id=aws_sc_pp

**AWS Product Details**

Provides details of a single provisioned product.

The **AWS Product Details** page is available using the following format:

http://<insertinstancename>.service-now.com/sp?id=aws_sc_pp_details&sys_id=< provisioned product id>

Upgrade Instructions

This section provides steps for upgrading from an earlier version of the AWS Service Catalog Connector for ServiceNow.

**To upgrade to the latest version of the Connector**

1. Clear the ServiceNow platform cache by typing in the following URL: https:// [InsertServiceNowInstanceNameHere]/cache.do
Note
Make sure you are installing the update set in a non-production/sandbox environment.
Consult a ServiceNow system administrator if you need approval to clear the ServiceNow platform cache.

2. Clear the web browser cache.
3. If you are upgrading from an earlier version, the permissions on ServiceNow Platform tables (User Criteria and Catalog Variable Set) are no longer needed for the Connector for ServiceNow.
4. Enable permissions on ServiceNow Platform tables (Category and Catalog Item Category).
5. For AWS products to display under AWS portfolios as sub-categories in the ServiceNow Service Catalog, you need to modify the Application Access form for Category and Catalog Item Category tables.
6. Enter “Tables” in the Navigator and choose System Definition, Tables.
7. In the list of tables, search for a table with label “Category” (or name “sc_category”). The list of tables will be displayed. Choose Category to view the form defining the table.
8. Choose the “Application Access” tab on the form and choose the “Can Create”, “Can Update, and “Can delete” checkboxes on the form. Choose the “Update” button.
9. Repeat the steps used on the Category table above for the “Catalog Item Category” table (type sc_cat_item_category in the “Go to Name Search” field).
10. From your ServiceNow dashboard, type plugins in the navigation panel in the upper left.
11. When the System Plugins page populates, next to the dropdown that says Name, search for user criteria.
12. Choose User Criteria Scoped API and then choose Activate.
13. Download the Connector for ServiceNow update set from the ServiceNow store. For users installing the update set on a ServiceNow Personal Developer Instance (PDI), download the code from Connector for ServiceNow version 2.3.4 update set.

The Connector for ServiceNow version 2.3.4 update set may be applied to a “Madrid,” "New York," or "Orlando" platform release of ServiceNow.
14. From your ServiceNow dashboard, type update sets in the navigation panel in the upper left.
15. Choose Retrieved Update Sets from the results.
16. Select Import Update Set from XML and upload the release XML file.
17. Select the AWS Service Catalog Connector for ServiceNow update set.
18. Choose Preview Update Set, which makes ServiceNow validate the connector update set.
19. Choose Update.
20. Choose Commit Update Set to apply the update set and create the application. This procedure should complete 100%.

To update AWS permissions
1. Go to Creating IAM Policies. Following the instructions there, create a policy called SCConnectorAdmin for ServiceNow administrators to delete AWS Service Catalog products in ServiceNow that do not have self-service actions associated. ServiceNow administrators can also view budgets associated to AWS Service Catalog portfolios and products. Copy the following policy and paste it into Policy Document:

```json
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "VisualEditor0",
        "Effect": "Allow",
```
"Action": [
  "servicecatalog:DisassociateProductFromPortfolio",
  "servicecatalog:DeleteProduct",
  "servicecatalog:DeleteConstraint",
  "servicecatalog:DeleteProvisionedProductPlan",
  "servicecatalog:DeleteProvisioningArtifact",
  "servicecatalog:ListBudgetsForResource",
  "budgets:ViewBudget"
],
"Resource": "*"
}

**Note**

The **ServiceCatalogServiceNowAdditionalPermissions** AWS policy is no longer needed for the Connector for ServiceNow.

2. Create a policy called **ServiceCatalogSSMActionsBaseline**. Follow the instructions at Creating IAM Policies, and paste the following into the JSON editor:

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

3. Update the **AWSCloudFormationFullAccess** policy. Choose create policy and then paste the following in the JSON editor:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:DescribeStackResource",
        "cloudformation:DescribeStackResources",
        "cloudformation:GetTemplate",
        "cloudformation:List*",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStacks",
        "cloudformation:CreateStack",
```
"cloudformation:DeleteStack",
"cloudformation:DescribeStackEvents",
"cloudformation:DescribeStacks",
"cloudformation:GetTemplateSummary",
"cloudformation:SetStackPolicy",
"cloudformation:ValidateTemplate",
"cloudformation:UpdateStack",
"cloudformation:CreateChangeSet",
"cloudformation:DescribeChangeSet",
"cloudformation:ExecuteChangeSet",
"cloudformation:DeleteChangeSet",
"s3:GetObject"
],
"Resource": "*
"

Note

AWSCloudFormationFullAccess now includes additional permissions for ChangeSets.

4. Attach the ServiceCatalogSSMActionsBaseline and AWSCloudFormationFullAccess IAM policies to the SCConnectLaunch role, which were created during the the section called "Baseline Permissions" (p. 81) setup.

5. The Connector for ServiceNow includes the ability for ServiceNow administrators to view budgets related to AWS Service Catalog products and portfolios. AWS Service Catalog administrators can create or associate budgets to portfolios and products. For information about creating and associating budgets, see the section called "Managing Budgets" (p. 61).

To add a change request type

1. When upgrading from a previous version of the AWS Service Catalog scoped app, you must remove the AWS Product Termination change request type before creating a new change request type.

2. You also must add a new change request type called AWS Provisioned Product Event for the scoped application to trigger an automated change request in Change Management. For instructions, see Add a new change request type.

   • Open an existing change request.
   • Open the context menu for Type and then choose Show Choice List.
   • Choose New and fill in the following fields:

<table>
<thead>
<tr>
<th>Table</th>
<th>Change request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>AWS Provisioned Product Event</td>
</tr>
<tr>
<td>Value</td>
<td>AWSProvisionedProductEvent</td>
</tr>
<tr>
<td>Sequence</td>
<td>pick the next unused value</td>
</tr>
</tbody>
</table>

   • Submit the form.
AWS Service Management Connector for Jira Service Desk

To help customers integrate provisioning secure, compliant, and pre-approved AWS products into their Jira Service Desk portal, AWS created the AWS Service Management Connector for Jira Service Desk (formerly the AWS Service Management Connector).

The AWS Service Management Connector for Jira Service Desk allows Jira Service Desk end users to provision, manage, and operate AWS resources natively via Atlassian's Jira Service Desk. Jira Service Desk administrators can provide pre-approved, secured, and governed AWS resources to end users via AWS Service Catalog, execute automation playbooks via AWS Systems Manager, and track resources in a configuration item view powered by AWS Config seamlessly on the Jira Service Desk with the AWS Service Management Connector.

Jira Service Desk end users can browse, request, and provision pre-secured AWS solutions, view configuration item details on provisioned products, and execute workflows within Jira Service Desk on AWS resources. This simplifies AWS product request actions for Jira Service Desk users and provides Jira Service Desk governance and oversight over AWS products.

The AWS-supplied connector is available at no charge in the Atlassian Marketplace. This new feature is generally available in all AWS Regions where AWS Service Catalog, AWS Config, and AWS Systems Manager services are available.

Topics

- Background (p. 98)
- Jira Service Desk Supported Versions and Releases (p. 99)
- Getting Started (p. 99)
- Release Notes (p. 100)
- Baseline Permissions (p. 101)
- Configuring AWS Service Catalog (p. 104)
- Configuring Jira Service Desk (p. 106)
- IT Lifecycle Management Setup and Use Case (p. 110)
- Validating Configurations (p. 115)
- Jira Additional Administrator Features (p. 116)

Background

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

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

AWS Systems Manager gives you visibility and control of your infrastructure on AWS. Systems Manager provides a unified user interface so you can view operational data from multiple AWS services and allows you to automate operational tasks across your AWS resources.

Atlassian Jira Service Desk is service desk software for modern IT teams. Jira Service Desk request types enable self-service for developers and end users to order IT services based on request fulfillment approvals and workflows.
Jira Service Desk Supported Versions and Releases

The AWS Service Management Connector for Jira Service Desk supports Jira Service Desk Server and Data Center versions. Jira software (Jira Service Desk) releases are supported for the current and one previous version in each of the major, minor, and point release streams:

- Jira 8.8.0 (JSD 4.8.0)
- Jira 8.7.1 (JSD 4.7.1)
- Jira 8.7.0 (JSD 4.7.0)
- Jira 8.6.1 (JSD 4.6.1)
- Jira 8.5.4 (JSD 4.5.4)
- Jira 7.13.13 (JSD 3.16.13)

A Jira Service Desk Cloud Connector is also available in the Atlassian Marketplace. For more information, see AWS Service Catalog for Jira Service Desk Cloud.

Getting Started

Before installing the AWS Service Management Connector for Jira Service Desk, you need an AWS account and an Atlassian instance with Jira Service Desk pre-installed. Verify that you have the necessary permissions in your AWS account and Jira Service Desk software.

For a video showing how to integrate AWS products into your Jira Service Desk portal, see Integrate AWS Products into Your Jira Service Desk Portal.

For a zip file containing Connector add-on code as well as AWS Configuration files, download and extract the AWS Service Management Connector for JSD-Configuration Files.

Note
The Jira Products on AWS Reference Deployment Quick Start is available to use AWS resources for infrastructure required to install Jira Service Desk data center version.

AWS prerequisites

To get started:

- To use AWS Service Catalog with the Connector, you need an AWS account to configure your AWS portfolios and products. For details, see Setting Up for AWS Service Catalog (p. 6).
- To see AWS Config details, the service settings need to be configured to record data for the resource types of interest. It is recommended to include provisioned products and AWS CloudFormation stacks in addition to the major resource types used by your team. For details, see Setting Up AWS Config with the Console.
- To use AWS Systems Manager Automation with the Connector, no AWS-side setup is required. A number of automation documents are provided by AWS as standard. If you have additional automation documents you wish to use, they will be available in the Connector. For details, see Working with Automation Documents (Playbooks).

For each AWS account, the Connector for Jira Service Desk also requires API access with the section called "Baseline Permissions" (p. 81) as described below.

Jira Service Desk prerequisites

In addition to your AWS account, you need the Jira Service Desk software installed on your Atlassian instance before you can install the AWS Service Management Connector add-on. The Jira Service Desk administrator needs the admin role in order to install the AWS Service Management Connector add-on.
Before configuring your AWS connector, ensure that you follow Atlassian recommendations for securing your Jira Service Desk instances. For more information, see Preventing Security Attacks.

The Connector for Jira Service Desk add-on is available to download in the Atlassian Marketplace.

**Release Notes**

**Version 1.5.0** of the AWS Service Management Connector for Jira Service Desk (formerly the AWS Service Management Connector) includes:

**AWS Service Catalog integration features**

- Rendering AWS Service Catalog portfolios and products in the Jira Service Desk Customer Portal and Jira Agent views.
- The ability for Jira Service Desk administrators to associate Jira Service Desk approval groups to AWS Service Catalog portfolios to require approvals for Jira Service Desk user product requests.
- The ability for Jira Service Desk users to request AWS Service Catalog products through Jira Service Desk.
- The ability for administrators to view portfolio and product budgets and actual costs. (Requires budgets to be associated within AWS Service Catalog.)
- Support for AWS Service Catalog self-service actions for Jira Service Desk users to update and terminate products.
- Support for AWS CloudFormation StackSets, enabling launch of AWS Service Catalog products across multiple regions and accounts.
- Support for CloudFront Change Sets, enabling a preview of resource changes prior to a launch or update.

**AWS Config integration features**

- Rendering of AWS Config configuration item details on provisioned AWS products via Jira Service Desk request.
- The ability to view the configuration item relationships in a tree structure.
- The ability to associate AWS Config items details to Jira issues.

**AWS Systems Manager integration features**

- Rendering of AWS Systems Manager automation documents in the Jira Service Desk Customer Portal and Jira Agent views.
- The ability for Jira Service Desk administrators to associate AWS Systems Manager automation to Jira projects.
- The ability for Jira Service Desk users to request and execute AWS Systems Manager automation documents through Jira Service Desk.
- The ability to create Jira issues (incidents) that provide actionable remediation suggestions via a Connector specific AWS Systems Manager automation document.

- Support for multiple AWS accounts.
- Support for FIPS endpoints and usage in the AWS GovCloud West region.
- Support for the latest releases of Jira Service Desk Server and Data Center versions.
Baseline Permissions

This section provides instructions on how to set up the baseline AWS users and permissions needed for the AWS Service Management Connector for Jira Service Desk. For each AWS account, the Connector for Jira Service Desk requires two sets of an access key identifier and a secret key for API access. These correspond to users in AWS Identity and Access Management (IAM). Specifically, you should set up:

- An IAM user to sync AWS resources to Jira Service Desk.
- An IAM user able to perform end user functionality to provision and execute requests exposed through Jira Service Desk, including assuming any roles required to perform the provisioning and execution (launch roles are recommended for AWS Service Catalog).

These can be the same user and can be an existing user, but in line with the best practice to give minimal permissions it is recommended these be two new users for Connector.

Full details of the permissions required by these users is included below, with an AWS CloudFormation template to facilitate setting this up. Baseline permissions enable an end user to provision the following AWS services: Amazon Simple Storage Service and Amazon Elastic Compute Cloud. To allow end users to provision AWS services beyond the baseline permissions, you must include the additional AWS service permissions to the appropriate roles.

**Note**

To use an AWS CloudFormation template to set up the AWS configurations of the Connector for Jira Service Desk, see the two JSON AWS Configurations for Connector for Jira Service Desk v1.5.0 - AWS Commercial Regions and Connector for Jira Service Desk v1.5.0 - AWS GovCloud West Region.

Creating AWS Service Management Connector Sync User

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

**To create AWS Service Management Connector sync user**

1. Go to Creating an IAM User in Your AWS Account. Following the instructions there, create a sync user (that is, SCSyncUser). The user needs programmatic and AWS Management Console access to follow the Connector for Jira Service Desk installation instructions.
2. Set permissions for your sync user (SCSyncUser). Choose Attach existing policies directly and select the AWSServiceCatalogAdminReadOnlyAccess policy and the AmazonSSMReadOnlyAccess policy.
3. Also add a policy allowing budgets:ViewBudget on all resources (*)).
4. Review and choose Create User.
5. Note the access and secret access information. Download the .csv file that contains the user credential information.

Creating AWS Service Management Connector End User

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

**To create AWS Service Management Connector end user**

1. Go to Creating an IAM User in Your AWS Account. Following the instructions there, create a user (such as SCEndUser). The user needs programmatic and AWS Management Console access to follow the Connector for Jira Service Desk installation instructions.
2. For products with AWS CloudFormation StackSets, you need to create a stack set inline policy. With AWS CloudFormation StackSets, you can create products that are deployed across multiple accounts and regions. Using an administrator account, you define and manage an AWS Service Catalog product and use it as the basis for provisioning stacks into selected target accounts across specified regions. You need to have the necessary permissions defined in your AWS accounts.

To set up the necessary permissions, go to Granting Permissions for Stack Set Operations. Following the instructions there, create an AWSCloudFormationStackSetAdministrationRole and an AWSCloudFormationStackSetExecutionRole.

3. Create the stack set inline policy to enable provisioning a product across multiple regions in one account, replacing the arn number string with your account number.

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

**Note**

The Connector for Jira Service Desk v1.5.0 - AWS Commercial Regions and Connector for Jira Service Desk v1.5.0 - AWS GovCloud West Region templates include the AWS CloudFormation StackSet permissions.

4. Add the following permissions (policies) to the user SCEndUser:

- AWSServiceCatalogEndUserFullAccess - (AWS managed policy)
- StackSet - (inline policy)
- AmazonS3ReadOnlyAccess
- AmazonEC2ReadOnlyAccess
- AWSConfigUserAccess

**Note**

For AWS Service Catalog products with AWS CloudFormation StackSets, you need to include the read only permissions for the services you want to provision. For example, to provision an Amazon S3 bucket, include the AmazonS3ReadOnlyAccess policy to the SCEndUser role.
5. Also add a policy allowing the following on all resources (*): ssm:DescribeAutomationExecutions, ssm:DescribeDocument, and ssm:StartAutomationExecution.

6. Review and choose Create User.

7. Note the access and secret access information. Download the .csv file that contains the user credential information.

Creating SCConnectLaunch Role

The following section describes how to create the SCConnectLaunch role. This role is used to place baseline AWS service permissions into the AWS Service Catalog launch constraints. For more information, see AWS Service Catalog Launch Constraints (p. 37).

To create SCConnectLaunch role

1. Create the AWSCloudFormationFullAccess policy. Choose create policy and then paste the following in the JSON editor.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:DescribeStackResource",
                "cloudformation:DescribeStackResources",
                "cloudformation:GetTemplate",
                "cloudformation:List*",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStacks",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStacks",
                "cloudformation:GetTemplateSummary",
                "cloudformation:SetStackPolicy",
                "cloudformation:ValidateTemplate",
                "cloudformation:UpdateStack",
                "cloudformation:CreateChangeSet",
                "cloudformation:DescribeChangeSet",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:DeleteChangeSet",
                "s3:GetObject"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Create a policy called ServiceCatalogSSMActionsBaseline. Follow the instructions on Creating IAM Policies, and paste the following into the JSON editor.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1536341175150",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeAutomationExecutions",
                "ssm:DescribeDocument",
                "ssm:StartAutomationExecution"
            ],
            "Resource": "*"
        }
    ]
}
```
3. Create the **SCConnectLaunch** role. Assign the trust relationship to AWS Service Catalog.

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

4. Attach the relevant policies to the **SCConnectLaunch** role. Attach the following baseline IAM policies:

- **AmazonEC2FullAccess** (AWS managed policy)
- **AmazonS3FullAccess** (AWS managed policy)
- **AWSCloudFormationFullAccess** (custom managed policy)
- **ServiceCatalogSSMActionsBaseline** (custom managed policy)

**Note**
To use an AWS CloudFormation template to set up the AWS configurations of the Connector for Jira Service Desk, see the two JSON AWS Configurations for **Connector for Jira Service Desk v1.5.0 - AWS Commercial Regions** and **Connector for Jira Service Desk v1.5.0 - AWS GovCloud West Region**.

## Configuring AWS Service Catalog

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

1. Create a portfolio by following the steps at Step 3: Create an AWS Service Catalog Portfolio (p. 12).
2. To add the Amazon S3 bucket product to the portfolio you just created, in the AWS Service Catalog console, on the Upload new product page, enter product details.
3. For Select template, choose the Amazon S3 bucket AWS CloudFormation template you saved to your device.
4. Set Constraint type to Launch for the product that you just created with the SCConnectLaunch role in the baseline permissions. For additional launch constraint instructions, see AWS Service Catalog Launch Constraints (p. 37).

**Note**
The AWS configuration design requires each AWS Service Catalog product to have either a launch or StackSet constraint. Failure to follow this step may result in an Unable to Retrieve Parameter message within Jira Service Desk Service Catalog.

Creating Stack Set Constraint

AWS CloudFormation StackSets enable users to create products that are deployed across multiple accounts and regions. In AWS Service Catalog, a stack set constraint allows you to configure product deployment options.

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

1. Go to AWS Service Catalog as a catalog administrator.
2. Choose the portfolio that contains the product you want to apply a constraint to.
3. Expand Constraints and choose Add constraints.
4. Choose the product from Product and set Constraint type to Stack Set. Choose Continue.
5. On the Stack set constraint page, enter a description.
6. Choose the accounts in which you want to create products.
7. Choose the regions in which you want to deploy products. Products are deployed in these regions in the order that you specify.
8. Choose the AWSCloudFormationStackSetAdministratorRole role that will be used to manage your target accounts.
9. Choose the AWSCloudFormationStackSetExecutionRole role that the administrator role will assume.
10. Choose Submit.

**Note** that the Connector for Jira Service Desk v1.5.0 - AWS Commercial Regions and Connector for Jira Service Desk v1.5.0 - AWS GovCloud West Region templates create the permissions as well as the outputs needed for stack set constraints. Example stack set outputs:

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

Note that AWS Service Catalog products can have either a stack set or a launch constraint, but not both.
Configuring Jira Service Desk

The AWS Service Management Connector for Jira Service Desk is released as a conventional Jira Service Desk add-on. Add-ons are code changes to the Jira software that extend its functionality or extend the functionality of Jira Service Desk software. The Connector for Jira Service Desk add-on is available to download in the Atlassian Marketplace.

After completing the IAM and AWS Service Catalog configurations, you must configure Jira Service Desk. Installation tasks within Jira Service Desk include:

- Clear your web browser cache.
- Install the Jira Service Desk Connector add-on.
- Configure AWS Service Management Connector add-on, including accounts, schedule sync, and request/approval permissions.

Clear Web Browser Cache

Clear your web browser cache to remove previously rendered Jira Service Desk forms.

Installing Jira Service Desk Connector Add-on

1. Log in to your Jira instance as an admin.
2. Open the admin menu and choose **Add-ons**.
3. On the **Manage add-ons** screen, choose **Find new apps** or **Find new add-ons** from the left side of the page.
4. Find AWS Service Management Connector for JSD. The search results should include app versions compatible with your Jira instance.
5. Choose **Install** to download and install your app.
6. Proceed to **Configuring AWS Accounts and Regions**.

Alternatively, you can download the code from the OBR file: AWS Service Management Connector for Jira Service Desk v1.5.0 OBR.

1. Go to **Manage apps**.
2. Select **Upload app** and upload the OBR file.
3. Proceed to **Configuring AWS Accounts and Regions**.

The Connector for Jira Service Desk version 1.5.0 add-on can be applied to the following Jira software (Jira Service Desk) releases: 7.11.2 (JSD 3.14.2) to Jira 8.5.1 (JSD 4.5.1).

Configuring AWS Accounts and Regions

Once the AWS Service Management Connector is installed, configure it by choosing the Jira administration icon in the top right, then choosing **Add-ons**.

1. From the AWS Service Catalog section on the left navigation menu, choose **AWS Accounts**.
2. Choose **Connect new account**.
3. Enter the account alias (used to identify the AWS account in the Connector).
4. Enter the credentials for SC-sync-user. This is the access key identity and credentials for a sync user saved from the AWS configuration. SC-sync-user credentials are used to retrieve portfolios and
products to make them available through Jira Service Desk. You will have the opportunity to set the groups allowed to access these.

5. Enter the credentials for SC-end-user. This is the access key identity and credentials for the end user saved from the AWS configuration. The SC-end-user credentials are used to provision products on behalf of a Jira user.

6. Add AWS Regions containing AWS Service Catalog products and portfolios that you want available in Jira Service Desk.

7. Choose **Test Connectivity**.

8. Upon successful connection status, choose **Connect**.

**Note**
We recommend that the sync user and end user be new users in AWS used only with AWS Service Management Connectors. These users should have minimum required privileges. An AWS CloudFormation template with the minimal permissions for AWS Service Management Connectors is available.

### Configuring AWS Service Catalog Portfolios within Jira

**AWS Product Access**

This section describes how to configure AWS Service Catalog portfolios within Jira.

Once your account or accounts are set up and connectivity is successful, use the **AWS Account** page to manage, for each account, which groups are permitted to access each portfolio in each Region. You can expand and collapse each Region and edit and add groups for each portfolio. Only users in the designated groups have access to those products. By default, no groups have access.

**Note**
At least one group must be associated to an AWS Service Catalog portfolio in order for Jira Service Desk end users to request AWS products.

#### To provision products and portfolios

1. Choose **AWS Accounts**.

2. Choose **Manage** for the AWS account on which you want to configure portfolios.

3. Under **Portfolios**, expand the Region associated with the account. Portfolios are displayed under each Region.

4. In the **Permission to request** column, choose **Add groups** for the portfolios that you want to make visible in Jira Service Desk. Select the group that you want to be able to see and request AWS Service Catalog products.

   **Note**
   Because the AWS Service Management Connector for Jira Service Desk allows Jira users to provision AWS products in the portfolios their groups have access to, and to control those provisioned products, users should be reminded of the importance of maintaining the security of their Jira accounts.

5. If products in this portfolio do not require approvals, choose **Save**.

**Jira Service Desk Approvals for Products in AWS Service Catalog Portfolios**

The AWS Service Management Connector for Jira Service Desk enables administrators to configure approvals for products at the portfolio level. All products within a portfolio that contains approval permissions will require approval, so AWS and Jira administrators may need to collaborate on the AWS Service Catalog portfolio structure.
To configure the approval process
1. Choose **AWS Accounts**.
2. Choose **Manage** on the AWS account for which you want to configure portfolio approvals.
3. In the **Permission to approve** column, choose **Add groups** for the portfolios that require product approvals.
4. Select **Require approval for provisioning**.
5. Under **Permission to approve**, choose **Add group**.
6. Choose **Save**.

**Note**
If a portfolio only has a group associated with **Permissions to request**, products within the portfolio immediately provision when the product request is submitted.

**Viewing Products and Budgets**

Two other tabs in the **Admin -> AWS Accounts -> Manage** section let you view information on portfolios, for reference. The **Available Products** tab lists the products in the portfolio and budgetary information on each. The **Budgets** tab gives overall budgetary information on the portfolio.

**Account Settings for Other Connector Features**

There are no per-account settings for the other aspects of AWS exposed through the JSD Connector, such as AWS Config and AWS Systems Manager Automation.

**Configuring Connector Settings (Jira Project Enablement and Request Type)**

Next, the AWS Service Management Connector for Jira Service Desk requires the add-on to be associated to a Jira project and a request type. You can configure which Connector features are enabled for each Jira project.

For a new installation of Connector, the default project configuration is for all Connector features (AWS Service Catalog, AWS Config, and AWS Systems Manager) to be enabled. If you are upgrading an existing installation, for security reasons, new features are initially not enabled.

**To configure the default Connector features for specific AWS services**

1. In the left navigation menu, under **AWS Service Catalog**, select **Connector settings**.
2. At the top, under **Connector features enabled by default**, select each feature depending whether you want projects using the default configuration to be able to use them or not.
3. Choose **Save**.

**To configure the Jira projects for which the AWS Service Management Connector for Jira Service Desk is enabled**

1. In the left navigation menu, under **AWS Service Management Connector**, select **Connector settings**.
2. Under **Projects enabled for Connector**, you must enable at least one Jira project. You can **create a new Jira Service Desk project** or add an existing one. Only users with access to the associated project will be able to access the Connector. When this update is applied, the Connector adds the necessary
issue types and other Jira items needed for AWS Service Catalog products to be available in those projects. You can return to this screen and add or remove projects at any time.

3. Projects initially take the default configuration in regards which Connector features are enabled. Click Edit in a project row to change the configuration for individual projects. It is permitted for projects to use more features than the default.

4. Choose Save.

**Note**

For end users to be able to request AWS Service Catalog products, one or more projects must be enabled and users must have Jira permissions to create issues in the Jira project and Permission to Request in the Jira settings for the AWS Account for at least one portfolio with products.

**AWS Systems Manager enablement considerations:** It is not currently supported to have fine-grained permissions in Jira for which users/groups should be allowed to access which AWS Systems Manager automation documents. If a project is enabled for Systems Manager Automation, then any user with permission to create issues in that project will have the ability to run any of the automations. Access can be restricted by limiting which users have access to projects with AWS Systems Manager Automation enabled.

**To configure operational settings for the AWS Service Management Connector for Jira Service Desk**

1. In the left navigation menu, under AWS Service Management Connector, select Connector settings.

2. Under Operational settings, in the Synchronization interval field, you can change the sync interval if you want. This interval determines how often Jira Service Desk syncs with AWS. Increasing this number will reduce the number of API calls to AWS but will mean it takes longer for updates in AWS portfolios and automation documents to be reflected in the Connector. Information on actively provisioning products and ongoing automation executions updates more frequently.

3. Choose Save.

**Configuring Project Request Type Groups**

The AWS request type must be in a group for users to be able to access it in Jira Service Desk. Enabling Jira projects, as described in Configuring Connector Settings (Jira Project Enablement and Request Type) (p. 108), makes AWS product request types available, but Jira Service Desk users won't see the request type until it is added to a Request Type Group.

**To configure request types**

1. In the AWS Service Management Connector for Jira Service Desk, go to the Connector settings page.

2. In the Projects section, choose add the AWS request type.

3. Select Add existing request type in the upper right-hand corner.

4. Select Request AWS product from the available request type.

5. Select Edit Groups for the Request AWS product request type.

6. On the Edit groups form, select General, then choose Save.

**Note**

A custom Request AWS Product request type was created for the Connector for Jira Service Desk, so edits to the Request AWS Product request type are not required. You can add a request type to an existing group. If you don't have a group, create a new group and add the request type to it.
IT Lifecycle Management Setup and Use Case

The AWS Service Management Connector for Jira Service Desk allows Jira Service Desk end-users to provision, manage and operate AWS resources natively via Atlassian's Jira Service Desk. To enable the IT Lifecycle Management scenario, you need to configure:

- AWS Config Linked Resources
- Suggested AWS Systems Manager Remediations for an Issue
- Sample Use Case: Automatically Creating Jira Issues for IT Lifecycle Management, Rremediating non-compliant public S3 buckets

AWS Config and suggested AWS Systems Manager remediations for any Jira issue

The Connector provides two fields that can be used and displayed for any issue.

- **AWS Config Linked Resources**: this field allows any resource with an entry in AWS Config to have its AWS Config information displayed on the issue in Jira. The information can be expanded and information on relations also displayed. Multiple AWS resources can be linked to an issue.
- **AWS Systems Manager Automation Suggested Remediation**: this field allows SSM automation documents to be recorded against an issue. These are then displayed as suggested ways to correct the issue. When a Jira user views the issue, they can see these suggested remediations and click to apply them. Multiple suggested remediations can be attached to an issue.

The two fields can be used individually but they work very well together: when an incident is detected on an AWS resource or set of resources, setting both allows a Jira user to see the Config information to confirm or better understand the problem, apply remediations which fix common problems, and then confirm in the AWS Config information that the problem has been fixed.

To add AWS fields to an existing issue

1. The project or projects must be enabled for the Connector. This is done in **Connector Settings** under **Admin -> Manage Add-Ons**, as described elsewhere in the Connector setup guide.
2. Go to **Admin**, then **Projects**, and open the project you wish to use these fields with.
3. Choose the issue type you wish to use in the menu at left.
4. Select to view "Fields" in the top right (if not already selected). It should then show a list of fields enabled for the screen.
5. Scroll to the bottom where there should be a textbox where you can type additional fields. Start typing "AWS" then select the AWS field you want to use.
6. Choose **Add** to apply.
7. Repeat the previous step for the other field if you wish to use it.
8. Repeat these steps for each issue type you wish you use these fields with. Some issue types may share screens so the field may already be added for some.

It is important also to make a note of the field ID for the field or fields you are using. This can be done by going to **Admin -> Issues -> Custom fields** and selecting **Configure** on each such field.

Inspect the URL that is opened to see the numeric field ID. It should be a 5-digit number. Alternatively, for any issue in a project where you've added the field (following the instructions above), the REST API at `/rest/api/2/issue/PRJ-1/editmeta` (for example, `http://localhost:2990/jira/rest/`)
api/2/issue/PRJ-1/editmeta) will include information on the fields and should contain an entry
customfield_#####: { ..., name: "AWS Config Linked Resources", ... }, where ####
is the numeric field ID.

Once these fields are enabled for projects and issue types, use the Jira REST API to create or update
issues with values for these fields. This can be used from tools such as CloudWatch or AppDynamics
or Jenkins or a Systems Manager Automation Document (provided in the next section). The REST API
endpoint to update an issue is /rest/api/2/issue/issue-key and the general schema to pass to set
a value is:

```json
{ "update": {
    "customfield_field-ID": [ {
        "set": "value"
    }]
}
```

See the examples below, or for more information on the REST API, see JIRA Developer Documentation:
Updating an Issue via the JIRA REST APIs.

### AWS Config Linked Resources

The **AWS Config Linked Resources** field should be set to the JSON string representation of a list of
objects (maps) corresponding to the linked resources, each with the following keys:

- **resourceId**: the ID of the resource in AWS Config
- **resourceType**: the type of the resource in AWS Config
- **accountName**: the name/alias of the AWS account configured in Jira that should be used to access this
  resource
- **region**: the region where AWS Config should be accessed to get information on this resource

For example the following value would show information on the S3 bucket my-bucket in eu-
central-1, using the account and end user credentials specified in Jira for the AWS Account identified
in Jira as MyAccount1:

```json
[ { "resourceId": "my-bucket",
    "resourceType": "AWS::S3::Bucket",
    "accountName": "MyAccount1",
    "region": "eu-central-1" } ]
```

### AWS Systems Manager Automation Suggested Remediation

The **AWS Systems Manager Automation Suggested Remediation** field should be set to the JSON string
representation of a list of objects (maps) corresponding to the automation documents to suggest as
remediations, each with the following keys:

- **documentName**: the name of the Systems Manager automation document
- **description**: a description of the remediation to display in Jira; this may be different to the document
description in AWS and might explain why it is a good remediation for the issue where this is being set
- **accountName**: the name/alias of the AWS account configured in Jira that should be used to access this
  resource
• **region**: the region where AWS Config should be accessed to get information on this resource

For example, the following value would suggest the `AWS-DisableS3BucketPublicReadWrite` automation document, with a nice description to show in Jira, to be applied in `eu-central-1`, using the account and end-user credentials specified in Jira for the AWS Account identified in Jira as **MyAccount1**:

```json
[ { "documentName": "AWS-DisableS3BucketPublicReadWrite", "description": "This will make the bucket private, resolving the issue.", "accountName": "MyAccount1", "region": "eu-central-1" } ]
```

### Scripting Field Creation

As an example, the following bash script using `curl` will link the above-noted resource to an issue and attach a suggested remediation. The values used below assume Jira is at `localhost:2990/jira` with login `admin:admin`, the issue is `PRJ-1`, and the field IDs are 10011 (AWS Config linked resources) and 10010 (suggested remediation). These should be changed to reflect your environment.

1. Set these corresponding to your environment and issue:

   ```bash
   JIRA_BASE_URL=http://localhost:2990/jira
   JIRA_USER_PASS=admin:admin
   ISSUE_KEY=PRJ-1
   ```

2. Set the field ID and edit the JSON record for an AWS Config resource to link

   ```bash
   CUSTOM_FIELD_ID=customfield_10011
   cat > value.json
   EOF
   ```

```json
{ "resourceId": "my-bucket", "resourceType": "AWS::S3::Bucket", "accountName": "MyAccount1", "region": "eu-central-1" }
```  

EOF

3. Define a helper function to escape the JSON

   ```bash
   json_escape () {
   printf '\%s' "$1" | python -c \n   'import json,sys; print(json.dumps(sys.stdin.read()))'
   }
   ```

4. Make the REST call to set the AWS Config Linked Resource field

   ```bash
   curl -v -D- -X PUT -H "Content-Type: application/json" \
   --data '{ "update": [ {"$({CUSTOM_FIELD_ID})": [ {"set": "$({json_escape "$(cat value.json)"})" } ] } ] }' \
   -u admin:admin ${JIRA_BASE_URL}/rest/api/2/issue/${ISSUE_KEY}
   ```
5. Set the field ID and edit the JSON record for a suggested remediation to attach

```
CUSTOM_FIELD_ID=customfield_10010
cat > value.json EOF
[ { "documentName": "AWS-DisableS3BucketPublicReadWrite",
  "description": "This will make the bucket private, resolving the issue.",
  "accountName": "MyAccount1",
  "region": "eu-central-1" } ] EOF
```

6. Make the REST call to set the AWS Systems Manager Automation Suggested Remediations field

```
curl -v -D- -X PUT -H "Content-Type: application/json" \
--data '{ "update": { ""$({CUSTOM_FIELD_ID})": [ {"set": "$(
  json_escape "$(cat value.json))")" } ] } }' \
-u ${JIRA_USER_PASS} ${JIRA_BASE_URL}/rest/api/2/issue/${ISSUE_KEY}
```

The issue should then show AWS Config for the bucket and a suggested remediation to make it private.

Creating Issues with Suggestions and a Linked AWS Resource from AWS Systems Manager

A Systems Manager Automation Document is provided which can automatically create a Jira issue with the fields set to have a linked AWS resource and up to three suggested remediation documents. To install this automation document, download and extract the JSD Connector Create Remediation Issue Automation and IT Lifecycle Demo.zip that contains two files:

- JSDConnector-CreateRemediationIssue.ssmdoc.yaml
- JSDConnector-function.zip

Follow these steps

1. Upload the file `JSDConnector-function.zip` to a bucket. The following command will do this (replacing `$(BUCKET)` with the appropriate bucket):

```
aws s3 cp JSDConnector-function.zip s3://$(BUCKET)/
```

2. Create the Systems Manager Automation Document, called `JSDConnector-CreateRemediationIssue`, with the contents taken from the file `JSDConnector-CreateRemediationIssue.ssmdoc.yaml` and an attachment `Key=SourceUrl,Values=s3://$(BUCKET)/`, using the bucket name from the previous step as $(BUCKET). The following command will do this (replacing $(BUCKET)):

```
aws ssm create-document --name "JSDConnector-CreateRemediationIssue" --content "file://JSDConnector-
```
Once installed, use it as you would any other Systems Manager automation document, filling out the parameters and running it. Note this requires many of the same parameters as described previously in order to connect to Jira.

You should then see an issue in Jira with AWS Config information and the suggested remediation shown.

**Sample Use Case: Automatically Creating Issues for IT Lifecycle Management - Remediating non-compliant public S3 buckets**

Once the fields are enabled to an issue and the Systems Manager Automation Document is created, you can set up rules to automatically create Jira issues for common problem categories in AWS and include suggested remediations to make it easy for a Jira agents and end users to see problems and fix them.

This demo will create a Config Rule is AWS which detects public S3 buckets and makes it possible for a Jira agents or end users to disable public access directly from Jira.

You will need to set up prerequisites, roles for the automation and lambda to execute as, and the Jira password as a secure string in Systems Manager Parameter Store.

**To store the Jira password securely in Parameter Store**

1. Open the AWS Console and go to Systems Manager -> Parameter Store.
2. Choose Create parameter.
3. Set the name as `jira_password`.
4. Set the type as SecureString.
5. Set the value as the password for the Jira user to be used to create issues.
6. To save, choose Create parameter.

An AWS CloudFormation template is provided to assist setting up the role and config rule:
`JSDConnector-CreateRemediationIssue-MakePublicBucketsPrivateConfigRule.cfn.yaml`

Install the template, setting the following parameters:

- **JiraURL**: the base URL to your Jira, such that appending `/rest/...` after it accesses the REST API
- **JiraUsername**: the username to use to log in to Jira (with the password specified in `jira_password`)
- **SSMParameterName**: `jira_password` (the parameter containing the Jira password)
- **ProjectKey**: the key of the project (the token before the `-` in an issue), e.g. `PRJ`
- **IssueTypeName**: this must exactly match the name of the issue type on the project in Jira
- **JiraAwsAccountName**: the name of the AWS Account as configured in the Connector in Jira
- **JiraAwsAccountRegion**: the region where this violating resource is found, e.g. `us-east-1`
- **JiraAwsResourceFieldId**: enter the field ID of the AWS Config Linked Resources field in Jira, e.g. `customfield_10011`
- **JiraRemediationsFieldId**: enter the field ID of the AWS Systems Manager Automation Suggested Remediation field in Jira, e.g. `customfield_10010`

The Config Rule will run automatically within the period specified. To see it in action immediately:

1. Create a public Amazon S3 bucket.
Validating Configurations

You are now ready to validate the AWS Service Management Connector for Jira Service Desk installation procedures.

AWS Service Catalog

To order an AWS Service Catalog product

1. Log in to your Jira Service Desk customer portal as the end user.
2. In the Jira Service Desk customer portal, choose Request AWS product.
3. Enter Summary details.
4. Open the AWS product request detail menu and select a product to provision.
5. Fill in the product request details, including product reference name, parameters, and tags.
6. Choose Create to submit the Jira Service Desk request and provision the AWS Service Catalog product.
7. After the request processes, a message appears indicating that your request was created. When the product is ready to provision, the end user is notified that the product is launching.

To view provisioned products

1. In the Jira Service Desk customer portal, go to Requests in the upper right corner.
2. Select My Requests in the Jira Service Desk customer portal view.
3. Select the AWS product you requested.
4. The AWS product details will display, including the status of the product request, product events, and activities.
5. Once the product is in the Available status, end users can request post-provision operations actions such as Request update, Request termination, and Request self-service actions. These actions render additional product events and activities within the request. Once the product is terminated, the request closes in a resolved state.

Systems Manager Automation

To execute an automation document

1. Log in to your Jira Service Desk customer portal as the end user.
2. In the Jira Service Desk customer portal, choose Request AWS automation.
3. Enter Summary details.
4. Open the AWS automation request detail menu and select an automation document to execute.
5. Fill in the automation request details, parameters and tags.
6. Choose Create to submit the Jira Service Desk request and execute the AWS Systems Manager Automation Document.
7. After the request processes, a message appears indicating that your request was created. As the automation executes, the end user is notified of progress.
To view automation executions

1. In the Jira Service Desk customer portal, go to **Requests** in the upper right corner.
2. Select **My Requests** in the Jira Service Desk customer portal view.
3. Select the AWS automation execution you requested. The AWS automation execution details will display, including the status of the execution, request details, and steps.

Jira Additional Administrator Features

The following sections describe approvals and access controls that are available in Jira.

**Approvals**

The approval agent has access to a screen with the options to approve or reject the product request. For a rejection, the agent can add a comment explaining why they are rejecting the request. The requester is able to see the status of the request, such as **Waiting for Approval**, **Scheduled**, **Launching**, or **Available**.

Changes to approver group members do not impact approvers identified for pre-existing issues but do affect whether the approval is permitted. Approver users are assigned to the issue at the time the issue is created, and only these users are given the option to approve the request. The approver user must still be a member of the group when the approval is made, otherwise the approval is rejected.

As with AWS Service Catalog, all post-provision actions, including termination, are pre-approved for the user or group who is approved to provision it.

**Access Controls**

Access controls can be set on portfolios, as described earlier in this guide. Those access controls are in addition to the per-project enablement: users must have access to an AWS Connector-enabled project and belong to the groups enabled for a portfolio in order to provision products in that portfolio.
## Document History

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Service Quotas</td>
<td>To learn about how AWS Service Catalog works with AWS Service Quotas, see <a href="https://aws.amazon.com/documentation/servicecatalog/default-service-quotas/">AWS Service Catalog default service quotas (p. 4)</a>.</td>
<td>March 24, 2020</td>
</tr>
<tr>
<td>Getting Started Library</td>
<td>To learn about the library of well-architected product templates offered by AWS Service Catalog, see <a href="https://aws.amazon.com/documentation/servicecatalog/getting-started-library/">Getting Started Library (p. 17)</a>.</td>
<td>March 10, 2020</td>
</tr>
<tr>
<td>Version Guidance</td>
<td>To learn about product version guidance, see <a href="https://aws.amazon.com/documentation/servicecatalog/version-guidance/">Version Guidance (p. 36)</a>.</td>
<td>December 17, 2019</td>
</tr>
<tr>
<td>Connector for Jira Service Desk</td>
<td>To begin using the Connector for Jira Service Desk, see <a href="https://aws.amazon.com/documentation/servicecatalog/connector-for-jira-service-desk/">AWS Service Management Connector for Jira Service Desk (p. 98)</a>.</td>
<td>November 21, 2019</td>
</tr>
<tr>
<td>New Version of Connector for ServiceNow</td>
<td>To learn about the updates to the Connector for ServiceNow, see <a href="https://aws.amazon.com/documentation/servicecatalog/connector-for-service-now/">AWS Service Catalog Connector for ServiceNow (p. 79)</a>.</td>
<td>November 18, 2019</td>
</tr>
<tr>
<td>Changing Provisioned Product Owner</td>
<td>To learn about how to change the owner of provisioned products, see <a href="https://aws.amazon.com/documentation/servicecatalog/changing-provisioned-product-owner/">Changing Provisioned Product Owner (p. 65)</a>.</td>
<td>October 31, 2019</td>
</tr>
<tr>
<td>New Resource Update Constraint</td>
<td>To learn about how to use the RESOURCE_UPDATE constraint to update tags in provisioned products, see <a href="https://aws.amazon.com/documentation/servicecatalog/tag-update-constraints/">AWS Service Catalog Tag Update Constraints (p. 40)</a>.</td>
<td>April 17, 2019</td>
</tr>
<tr>
<td>Connector for ServiceNow</td>
<td>To begin using the Connector for ServiceNow, see <a href="https://aws.amazon.com/documentation/servicecatalog/connector-for-service-now/">AWS Service Catalog Connector for ServiceNow (p. 79)</a>.</td>
<td>March 19, 2019</td>
</tr>
<tr>
<td>Support for AWS CloudFormation StackSets</td>
<td>To begin using AWS CloudFormation StackSets, see <a href="https://aws.amazon.com/documentation/servicecatalog/support-for-cloudformation-stacksets/">Using AWS CloudFormation StackSets (p. 60)</a>.</td>
<td>November 14, 2018</td>
</tr>
<tr>
<td>Self-service actions</td>
<td>To begin using self-service actions, see <a href="https://aws.amazon.com/documentation/servicecatalog/service-actions/">AWS Service Catalog Service Actions (p. 49)</a>.</td>
<td>October 17, 2018</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>CloudWatch metrics</td>
<td>To learn about CloudWatch metrics, see <em>AWS Service Catalog CloudWatch Metrics</em> (p. 76).</td>
<td>September 26, 2018</td>
</tr>
<tr>
<td>Support for TagOptions</td>
<td>To manage tags, see <em>AWS Service Catalog TagOption Library</em> (p. 71).</td>
<td>June 28, 2017</td>
</tr>
<tr>
<td>Importing a portfolio</td>
<td>To import a portfolio shared from another AWS account, see <em>Importing a Portfolio</em> (p. 60).</td>
<td>February 16, 2016</td>
</tr>
<tr>
<td>Updates to permissions information</td>
<td>To grant access to the end user console view, see <em>Console Access for End Users</em> (p. 22).</td>
<td>February 16, 2016</td>
</tr>
<tr>
<td>Initial release</td>
<td>This is the initial release of the <em>AWS Service Catalog Administrator Guide</em>.</td>
<td>July 9, 2015</td>
</tr>
</tbody>
</table>