{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "AWS Service Catalog sample template. Creates an Amazon EC2 instance

running the Amazon Linux AMI. The AMI is chosen based on the region

in which the stack is run. This example creates an EC2 security

group for the instance to give you SSH access. \*\*WARNING\*\* This

template creates an Amazon EC2 instance. You will be billed for the

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

"Parameters" : {

"KeyName": {

"Description" : "Name of an existing EC2 key pair for SSH access to the EC2 instance.",

"Type": "AWS::EC2::KeyPair::KeyName"

},

"InstanceType" : {

"Description" : "EC2 instance type.",

"Type" : "String",

"Default" : "t2.micro",

"AllowedValues" : [ "t2.micro", "t2.small", "t2.medium", "m3.medium", "m3.large",

"m3.xlarge", "m3.2xlarge" ]

},

"SSHLocation" : {

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

"Type": "String",

"MinLength": "9",

"MaxLength": "18",

"Default": "0.0.0.0/0",

"AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",

"ConstraintDescription": "Must be a valid IP CIDR range of the form x.x.x.x/x."

}

},

"Metadata" : {

"AWS::CloudFormation::Interface" : {

"ParameterGroups" : [{

"Label" : {"default": "Instance configuration"},

"Parameters" : ["InstanceType"]

},{

"Label" : {"default": "Security configuration"},

"Parameters" : ["KeyName", "SSHLocation"]

}],

"ParameterLabels" : {

"InstanceType": {"default": "Server size:"},

"KeyName": {"default": "Key pair:"},

"SSHLocation": {"default": "CIDR range:"}

}

}

},

"Mappings" : {

"AWSRegionArch2AMI" : {

"us-east-1" : { "HVM64" : "ami-08842d60" },

"us-west-2" : { "HVM64" : "ami-8786c6b7" },

"us-west-1" : { "HVM64" : "ami-cfa8a18a" },

"eu-west-1" : { "HVM64" : "ami-748e2903" },

"ap-southeast-1" : { "HVM64" : "ami-d6e1c584" },

"ap-northeast-1" : { "HVM64" : "ami-35072834" },

"ap-southeast-2" : { "HVM64" : "ami-fd4724c7" },

"sa-east-1" : { "HVM64" : "ami-956cc688" },

"cn-north-1" : { "HVM64" : "ami-ac57c595" },

"eu-central-1" : { "HVM64" : "ami-b43503a9" }

}

},

"Resources" : {

"EC2Instance" : {

"Type" : "AWS::EC2::Instance",

"Properties" : {

"InstanceType" : { "Ref" : "InstanceType" },

"SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],

"KeyName" : { "Ref" : "KeyName" },

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

}

},

"InstanceSecurityGroup" : {

"Type" : "AWS::EC2::SecurityGroup",

"Properties" : {

"GroupDescription" : "Enable SSH access via port 22",

"SecurityGroupIngress" : [ {

"IpProtocol" : "tcp",

"FromPort" : "22",

"ToPort" : "22",

"CidrIp" : { "Ref" : "SSHLocation"}

} ]

}

}

},

"Outputs" : {

"PublicDNSName" : {

"Description" : "Public DNS name of the new EC2 instance",

"Value" : { "Fn::GetAtt" : [ "EC2Instance", "PublicDnsName" ] }

},

"PublicIPAddress" : {

"Description" : "Public IP address of the new EC2 instance",

"Value" : { "Fn::GetAtt" : [ "EC2Instance", "PublicIp" ] }

}

}

}

Step 2: Create a Key Pair

"Parameters" : {

"KeyName": {

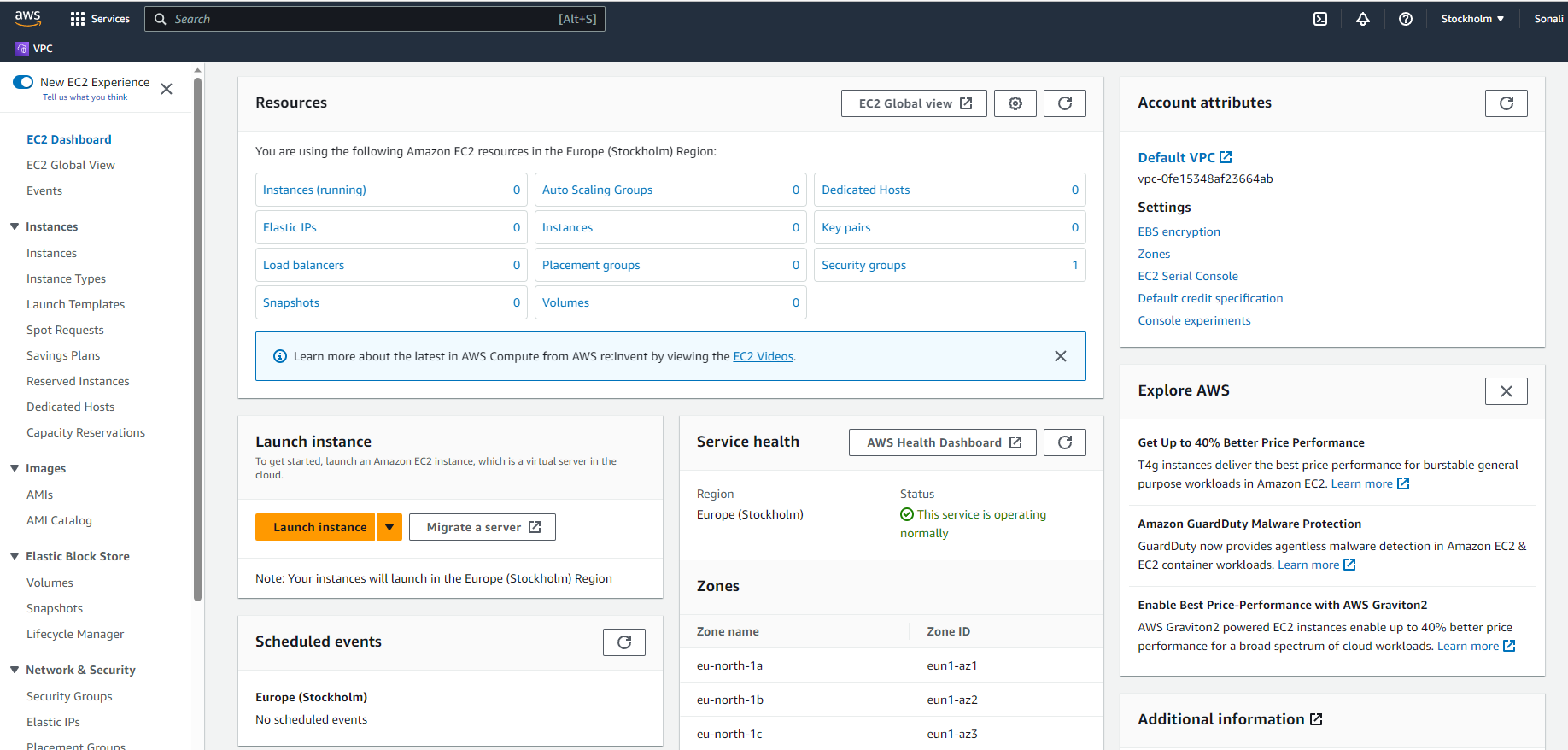
"Description" : "Name of an existing EC2 key pair for SSH access to the EC2 instance.",

"Type": "AWS::EC2::KeyPair::KeyName"

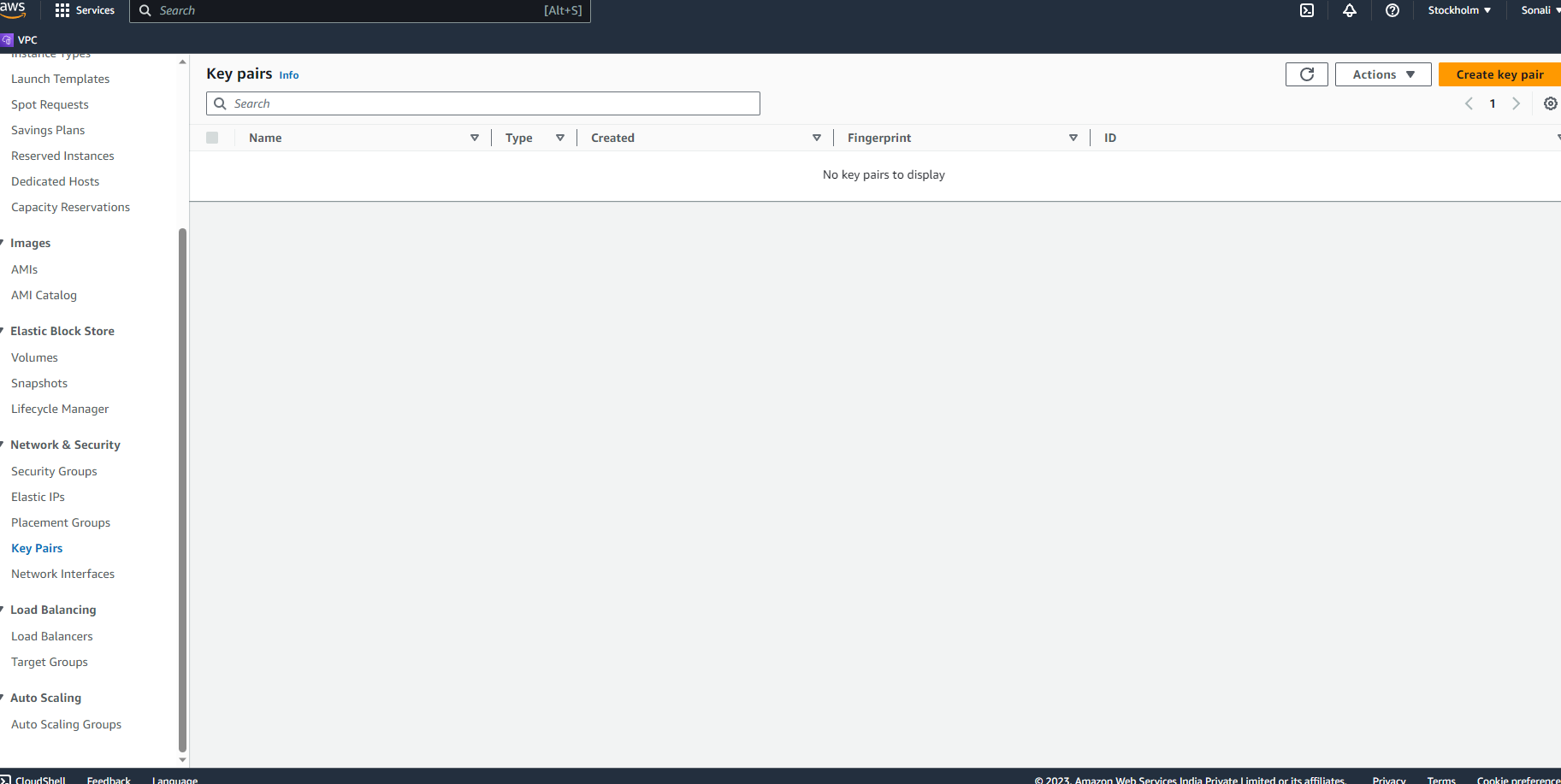
},

###### To create a key pair

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.



1. In the navigation pane, under **Network & Security**, choose **Key Pairs**.



1. On the **Key Pairs** page, choose **Create Key Pair**.
2. For **Key pair name**, type a name that is easy for you to remember, and then choose **Create**.
3. When the console prompts you to save the private key file, save it in a safe place.

# Step 3: Create a Service Catalog Portfolio

[**PDF**](https://docs.aws.amazon.com/pdfs/servicecatalog/latest/adminguide/service-catalog-ag.pdf#getstarted-portfolio)

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

###### To create a portfolio

1. Open the Service Catalog console at <https://console.aws.amazon.com/servicecatalog/>.
2. In the left navigation panel, choose **Portfolios**, and then choose **Create portfolio**.
3. Type the following values:
   * **Portfolio name** – **Engineering Tools**
   * **Portfolio description** – **Sample portfolio that contains a single product.**
   * **Owner** – **IT (it@example.com)**
4. Choose **Create**.

[Step 4: Create a Service Catalog Product - Service Catalog (amazon.com)](https://docs.aws.amazon.com/servicecatalog/latest/adminguide/getstarted-product.html)

# Step 4: Create a Service Catalog Product

[**PDF**](https://docs.aws.amazon.com/pdfs/servicecatalog/latest/adminguide/service-catalog-ag.pdf#getstarted-product)

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

###### To create a product

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

# Step 5: Add a Template Constraint to Limit Instance Size

[**PDF**](https://docs.aws.amazon.com/pdfs/servicecatalog/latest/adminguide/service-catalog-ag.pdf#getstarted-constraint)

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 Service Catalog Constraints](https://docs.aws.amazon.com/servicecatalog/latest/adminguide/constraints.html).

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](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/t2-instances.html) 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, choose **Constraints**, then choose **Create constraint**.
2. In the **Create constraint** page, for **Product,** choose **Linux Desktop**. Then, for **Constraint type**, choose **Template**.
3. In the T**emplate constraint** section, choose **Text editor**.
4. Paste the following into the text editor:
5. {
6. "Rules": {
7. "Rule1": {
8. "Assertions": [
9. {
10. "Assert" : {"Fn::Contains": [["t2.micro", "t2.small"], {"Ref": "InstanceType"}]},
11. "AssertDescription": "Instance type should be t2.micro or t2.small"
12. }
13. ]
14. }
15. }

}

1. For **Constraint description**, enter **Small instance sizes**.
2. Choose **Create**.

# Step 6: Add a launch constraint to assign an IAM role

[**PDF**](https://docs.aws.amazon.com/pdfs/servicecatalog/latest/adminguide/service-catalog-ag.pdf#getstarted-launchconstraint)

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

For this step, you add a launch constraint to the Linux Desktop product, so Service Catalog can use the IAM resources that make up the product's AWS CloudFormation template.

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

1. AWS CloudFormation
2. Services in the AWS CloudFormation template for the product
3. Read access to the AWS CloudFormation template in a service-owned Amazon S3 bucket.

This launch constraint enables the end user to launch the product and, after launch, manage it as a provisioned product. For more information, see [Service Catalog Launch Constraints](https://docs.aws.amazon.com/servicecatalog/latest/adminguide/constraints-launch.html).

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

Using a launch constraint allows you follow the IAM best practice of keeping end user IAM permissions to a minimum. For more information, see [Grant least privilege](https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege) in the IAM User Guide.

###### To add a launch constraint

1. Follow the instructions to [Create new policies on the JSON tab](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create-console.html) in the IAM User guide.
2. Paste the following JSON policy document:
   * cloudformation– Allows Service Catalog full permissions to create, read, update, delete, list, and tag AWS CloudFormation stacks.
   * ec2— Allows Service Catalog full permissions to list, read, write, provision, and tag Amazon Elastic Compute Cloud (Amazon EC2) resources that are part of the Service Catalog product. Depending on the AWS resource that you want to deploy, this permission might change.
   * ec2– Creates a new managed policy for you AWS account and attaches the specified managed policy to the specified IAM role.
   * s3— Allows access to Amazon S3 buckets owned by Service Catalog. To deploy the product, Service Catalog requires access to provisioning artifacts.
   * servicecatalog— Allows Service Catalog permissions to list, read, write, tag, and launch resources on behalf of the end-user.
   * sns— Allows Service Catalog permissions to list, read, write, and tag Amazon SNS topics for the launch constraint.

###### Note

Depending on the underlying resources that you want to deploy, you might need to modify the example JSON policy.

{

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

"Statement": [

{

"Effect": "Allow",

"Action": [

"cloudformation:CreateStack",

"cloudformation:DeleteStack",

"cloudformation:DescribeStackEvents",

"cloudformation:DescribeStacks",

"cloudformation:GetTemplateSummary",

"cloudformation:SetStackPolicy",

"cloudformation:ValidateTemplate",

"cloudformation:UpdateStack",

"ec2:\*",

"servicecatalog:\*",

"sns:\*"

],

"Resource": "\*"

},

{

"Effect":"Allow",

"Action":[

"s3:GetObject"

],

"Resource":"\*",

"Condition":{

"StringEquals":{

"s3:ExistingObjectTag/servicecatalog:provisioning":"true"

}

}

}

]

}

1. Choose **Next**, **Tags**.
2. Choose **Next,** **Review**.
3. In the **Review policy** page, for the **Name**, enter **linuxDesktopPolicy**.
4. Choose **Create policy**.
5. In the navigation pane, choose **Roles**. Then choose **Create role** and do the following:
   * For **Select trusted entity**, choose **AWS service** and then under **Use case for other AWS services**choose **Service Catalog**. Select the Service Catalog use case and then choose **Next**.
   * Search for the **linuxDesktopPolicy** policy and then select the checkbox.
   * Choose **Next**.
   * For **Role name**, type **linuxDesktopLaunchRole**.
   * Choose **Create role**.
6. Open the AWS Service Catalog console at [https://console.aws.amazon.com/servicecatalog](https://console.aws.amazon.com/servicecatalog.).
7. Choose the **Engineering Tools** portfolio.
8. On the **Portfolio details** page, choose the **Constraints** tab, and then choose **Create constraint**.
9. For **Product**, choose **Linux Desktop**, and for **Constraint type**, choose **Launch**.
10. Choose **Select IAM role**. Next choose **linuxDesktopLaunchRole**, and then choose **Create**.

# Step 7: Grant end users access to the portfolio

[**PDF**](https://docs.aws.amazon.com/pdfs/servicecatalog/latest/adminguide/service-catalog-ag.pdf#getstarted-deploy)

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 Service Catalog end users](https://docs.aws.amazon.com/servicecatalog/latest/adminguide/getstarted-iamenduser.html).

###### To provide access to the portfolio

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

# Test the End User Experience

[**PDF**](https://docs.aws.amazon.com/pdfs/servicecatalog/latest/adminguide/service-catalog-ag.pdf#getstarted-verify)

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

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

1. Follow the instructions to [Sign in as an IAM user](https://docs.aws.amazon.com/IAM/latest/UserGuide/console.html) in the IAM User guide.
2. In the menu bar, choose the AWS Region in which you created the Engineering Tools portfolio. For this tutorial, choose **us-east-1 region**.
3. Open the AWS Service Catalog console at <https://console.aws.amazon.com/servicecatalog/> to see:
   * **Products** – The products that the user can use.
   * **Provisioned products** – The provisioned products that the user has launched.

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

Note that for this tutorial, choose**us-east-1 region**.

1. In the **Products** section of the console, choose **Linux Desktop**.
2. Choose **Launch product** to start the wizard that configures your product.
3. On the **Launch: Linux Desktop** page, enter **Linux-Desktop** for the provisioned product name.
4. On the **Parameters** page, enter the following and choose **Next**:
   * **Server size** – Choose **t2.micro**.
   * **Key pair** – Select the key pair that you created in [Step 2: Create a Key Pair](https://docs.aws.amazon.com/servicecatalog/latest/adminguide/getstarted-keypair.html).
   * **CIDR range** – Enter a valid CIDR range for the IP address to connect to the instance. You can use the default value (0.0.0.0/0) to allow access from any IP address, then your IP address, followed by **/32** to restrict access to your IP address only, or something in between.
5. Choose **Launch product**to launch the stack. The console displays the stack details page for the Linux-Desktop stack. The initial status of the product is **Under change**. It takes several minutes for AWS Service Catalog to launch the product. To see the current status, refresh your browser. After the product launches, the status is A**vailable**.

## >>>>>>>>>>>>Gathering Instance Metadata<<<<<<<<<<<<<<<<<<<<<

To gather instance metadata within AWS::CloudFormation::Init, you can utilize the AWS::CloudFormation::Init metadata section and the AWS::CloudFormation::Init config sets. Let’s dive into the steps involved:

### Step 1: Define Metadata

In your CloudFormation template, you need to define the AWS::CloudFormation::Init metadata section. This section contains configurations for various resources, including AWS::CloudFormation::Init, which is where we will focus.

**Resources:**

**MyInstance:**

**Type:** AWS::EC2::Instance

**Metadata:**

**AWS::CloudFormation::Init:**

**configSets:**

**myConfigSet:**

- gatherInstanceMetadata

**gatherInstanceMetadata:**

**commands:**

**01\_get\_instance\_metadata:**

**command:** "curl http://169.254.169.254/latest/meta-data/ -o /tmp/instance\_metadata.txt"

In the above example, we define an EC2 instance resource named MyInstance and include the AWS::CloudFormation::Init metadata section. We define a config set named myConfigSet and specify that it should run the gatherInstanceMetadata configuration. Inside the gatherInstanceMetadata configuration, we define a command named 01\_get\_instance\_metadata that uses the curl command to retrieve the instance metadata and save it to a file named instance\_metadata.txt in the /tmp directory.

### Step 2: Apply Metadata during Stack Creation

To apply the defined metadata during stack creation, you need to include the AWS::CloudFormation::Init in the UserData property of your EC2 instance resource.

**Resources:**

**MyInstance:**

**Type:** AWS::EC2::Instance

**Properties:**

**UserData:**

"Fn::Base64"**:**

!Sub |

*#!/bin/bash -xe*

/opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource MyInstance --configsets myConfigSet --region ${AWS::Region}

In the above snippet, we pass the myConfigSet config set to the --configsets option of the cfn-init command. This ensures that the instance metadata gathering configuration is applied during the stack creation process.

### Step 3: Access the Instance Metadata

Once the stack creation is complete, you can access the gathered instance metadata. In our example, we saved the metadata to the /tmp/instance\_metadata.txt file. You can SSH into the EC2 instance or use AWS Systems Manager Session Manager to access the instance without SSH access. Then, you can read the contents of the file to retrieve the metadata.

$ cat /tmp/instance\_metadata.txt

The above command will display the contents of the instance\_metadata.txt file, which should contain the gathered instance metadata.

## Conclusion

In this article, we explored how to gather instance metadata within AWS::CloudFormation::Init. By leveraging the power of AWS CloudFormation, you can automate the process of collecting valuable information about your instances during the stack creation process. This information can be used for various automation tasks, application configurations, or any other use case that requires access to instance metadata. With the ability to define metadata, commands, and files, AWS::CloudFormation::Init provides a flexible and efficient solution for customizing and configuring your EC2 instances.