

Toronto - Self service+Incident management copy

#3 Self Service

In this subtask, you will expose your catalog to your ServiceNow which is a popular ITSM tool and enable self-service of IT resources.

What is ServiceNow: [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.

The catalog you set up in task 1 and 2 can be exposed to ServiceNow, such that your end users will be able to request an EC2 instance/Workspaces desktop from ServiceNow. To enable this connection, AWS has created a ServiceNow-Service Catalog connector. With the connector, you will be able to manage your catalog using AWS Service Catalog and set up approvals/workflows in ServiceNow. [ServiceNow Service Catalog](#) is a self-service application that end users can use to order IT services based on request fulfillment approvals and workflows. A detailed step-by-step-walkthrough blog on how to configure the AWS side and the ServiceNow side is available on this blog

- <https://aws.amazon.com/blogs/mt/how-to-install-and-configure-the-aws-service-catalog-connector-for-servicenow/>

In this workshop, for your convenience, the setup process has been simplified and some pre-configuration has been done for you. Also, following ServiceNow connection specific roles have been created for you:

- a. **SCEC2ConnectLaunchRole** - This role is an internal role created for your ServiceNow-Service Catalog connector. This role is associated with **EC2 product** while setting up your **SNOW-SC test portfolio**.
- b. **SCWorkspacesConnectLaunchRole** - This role is an internal role created for your ServiceNow-Service Catalog connector. This role is associated with **Workspaces product** while setting up your **SNOW-SC test portfolio**.
- c. **SnowEndUser** - This role is mapped to the ServiceNow side role to which access to launch the products from a portfolio is given. This role has **AWSServiceCatalogEndUserFullAccess** managed policy associated with it and will have access to **SNOW-SC test portfolio** from your catalog.
- d. **SCSyncUser** - This user has **ServiceCatalogAdminReadOnlyAccess** policy associated with it. This user is used by Service catalog- Service now connector for synchronizing portfolios, products, etc.
- e. **SCEndUser** - Within ServiceNow, the SCEndUser is mapped to **snow-stsuser-account**. SCEndUser has access to assume SnowEndUser role.

In this task:

- you will create two service-now accounts, and map them to SCEndUser and SCSyncUser using credentials.
- Next, you will run Sync all accounts scheduled job which will pull Service catalog resources into ServiceNow.
- Next, you will associate SnowEndUser (from AWS) with the snow-stsuser-account and order_aws_sc_products.
- You will launch an EC2 instance from ServiceNow (brownie points task)

SUBTASK 3.1 CONFIGURE ACCOUNTS

In the **AWS Service Catalog scoped app Accounts** menu, you will create two accounts, one for sync and one for provisioning. The snow-stsuser-account account has no Regions configured. The snow-sync-account user has one region configured, matching the Region where the AWS Service Catalog is defined.

ServiceNow-AWS Correlations	
ServiceNow Account	AWS User
snow-sync-account	SCSyncUser
snow-stsuser-account	SCEndUser

To configure **snow-sync-account** :

1. Log in to ServiceNow as the **System Administrator** using credentials provided to you (you will be given a link, and credentials for ServiceNow instance).
2. In the navigation panel on the left, choose **Accounts** (type **Accounts** in filter navigator search box) under **AWS Service Catalog**.
3. Choose **New** to create a new account.

4. Specify **snow-sync-account** as the **Name**.
5. Under **Access Key**, specify the value of **SCSyncUserAccessKey** provided to you in the outputs section of Cloudformation.
6. Under **Secret Access Key**, specify the value of **SCSyncUserSAK** provided to you, in the outputs section of Cloudformation. Next, click on **submit**, and open the **snow-sync-account**.

Account Synchronizations	
US East (N. Virginia)	

7. Under **Account Synchronizations**, choose region specified in the output section by clicking the tick icon (you can double click on “insert a new row” in the Region column to see regions list) of Cloudformation and then click on **update**.
8. Next, open the **snow-sync-account** entry. Choose **Validate Regions**, in a few seconds, it should show following message. This means your ServiceNow is able to communicate with AWS.

① Successfully performed AWS Service Catalog SearchProductsAsAdmin action in each referenced region.



Note: If it doesn't show you this message, check the **Access Key** and the **Secret Access Key** you entered and validate the regions again.

To configure **snow-stsuser-account** :

9. Go back to **Accounts** screen, Choose **New** to create a new account.

10. Specify **snow-stsuser-account** as the **Name**.

11. Under **Access Key**, specify value of **SCEndUserAccessKey** provided to you in the outputs section of Cloudformation.

12. Under **Secret Access Key**, specify value of **SCEndUserSAK** provided to you in the outputs section of Cloudformation.

13. Choose **Submit**.

SUBTASK 3.2 SCHEDULED JOBS (INITIAL MANUAL SYNC)

During the initial setup, manually execute the sync(synchronize) job instead of waiting for the Scheduled Jobs to occur. To synchronize the accounts manually, do the following:

1. As **System Administrator**, in ServiceNow console, find **Scheduled Jobs** under **System Definition** in the filter navigator panel on the left.

Name	Active	Class	Updated
Sync all Accounts	true	Scheduled Script Execution	2017-11-30 11:36:47
Sync all Provisioned AWS Service Catalog Products	true	Scheduled Script Execution	2017-10-27 04:02:55
Synchronous Import Set Completer	true	Scheduled Script Execution	2009-01-03 23:39:56
Team Development Update Push Status	true	Scheduled Script Execution	2013-07-22 11:20:17
test import from attached	false	Scheduled Data Import	2007-06-11 15:23:40

Next, Search by name for job called **Sync all Accounts**, open it by clicking it, and then choose **Execute Now**.

Scheduled Script Execution
Sync all Accounts

This record is in the **AWS Service Catalog** application, but **Global** is the current application. To edit this record click [here](#).

Name: Sync all Accounts

Active: ☒

Run: Periodically

Application: AWS Service Catalog

Repeat Interval: 30 Minutes

Starting: 2017-09-15 02:45:51

Conditional: ☐

Execute Now

SUBTASK 3.3 GRANT ACCESS TO PORTFOLIOS

Data will be visible in the AWS Service Catalog scoped app menus after the adapter's scheduled synchronization job has run.

1. In ServiceNow, as an administrator user, in the navigation panel on the left, choose **Identities** under **AWS Service Catalog**.
2. Open the identity with **SnowEndUser** ARN. E.g. `arn:aws:iam::0123456789012:role/SnowEndUser`
3. Under **Account**, choose **snow-stsuser-account** (you can also find this by clicking on search/look-up button);, next choose **Update** to update the record.

AWS Identity
arn:aws:iam:::role/SnowEndUser

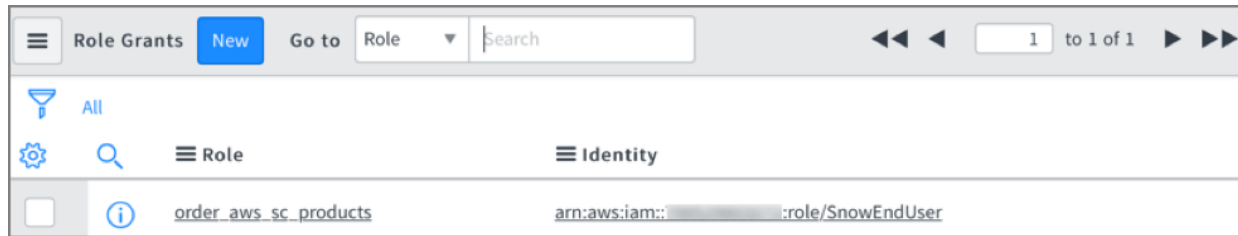
ARN: arn:aws:iam:::role/SnowEndUser

Account: snow-stsuser-account

Update **Delete**

4. Next, In ServiceNow, In the navigation panel on the left, choose **Role Grants** under **AWS Service Catalog**.

5. Choose **New** to create a new **Role Grant**. Under **role**, specify **order_aws_sc_products**.
6. Under Identity, specify **SnowEndUser**'s ARN(you can also find this by clicking on search button) specified in the outputs section of Cloudformation, choose **Submit**.



Then click on the the identity from role grant you just created. Wait for a few seconds and then choose **Test authorization** to test whether it is working or not.

① Successfully performed SearchProducts action as arn:aws:iam::306633430434:role/SnowEndUser

Once the operation succeeds, you will see a message like the one shown above. If you don't get above message, wait for a minute and click on **Test authorization** again.

Given the set up above, since **order_aws_sc_products** has been associated with the **SCEndUser01** user, **SCEndUser01** can now order products from AWS Service Catalog in ServiceNow. You dont need to order products now, you will order it as part of next task.

#4 Incident management

Customers want to leverage Amazon CloudWatch, AWS Config, and ServiceNow together to receive notification of important events and to quickly orchestrate a remediation. In this task, you will see how these services can work together to enable this outcome. The blog on which this task is based can be found [here](#).

Subtasks:

- **Subtask 4.1:** You will see that we have set up an AWS Config rule which becomes non-compliant every time an EC2 instance goes out of compliance. We have also set up a CloudWatch events rule which publishes an SNS notification on an SNS topic every time compliance status changes.
- **Subtask 4.2:** You will Subscribe to the topic from ServiceNow.
- **Subtask 4.3:** Launch an EC2 server of the type t2.medium and see an incident getting created in ServiceNow.

Here is a quick recap of important concepts that you would use in this lab:

- [Amazon Simple Notification Service \(Amazon SNS\)](#) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.
 - Using SNS you can create a topic on which you can publish messages and subscribers of this topic will receive messages.
 - The type of subscriber can be an email, a text message, a Lambda function, or even an SQS queue.
- An **AWS Config rule** represents your desired configuration settings for specific AWS resources or for an entire AWS account. AWS Config provides customizable, predefined rules to help you get started. If a resource violates a rule, AWS Config flags the resource and the rule as noncompliant, and AWS Config notifies you through Amazon SNS.

- **Amazon CloudWatch** is a monitoring and management service that enables you to monitor your applications, understand and respond to system-wide performance changes, optimize resource utilization, and get a unified view of operational health. Using **CloudWatch Event rules** you can quickly set up, you can match events and route them to one or more target functions or streams. This enables different parts of an organization to look for and process the events that are of interest to them. A rule can customize the JSON sent to the target, by passing only certain parts or by overwriting it with a constant. For more information on [CloudWatch Events rules](#).
- **Service Now Incidents:** ServiceNow routes incidents to the right people in your organization who need to take action when something meaningful happens in your AWS environment. In this task, we see how both can work together. Here is the outline of subtasks that you will do as part of this task.

SUBTASK 4.1: SET UP AN AWS CONFIG RULE & A CLOUDWATCH EVENTS RULE THAT PUBLISHES A SNS NOTIFICATION EVERYTIME AN EC2 INSTANCE GOES OUT OF COMPLIANCE.

First review the environment. We have pre-provisioned a custom AWS Config rule that flags any EC2 instance that does not have **t2.micro** as an instance type, as a non-compliant resource. To view the rules available:

1. In your AWS console, You need to go back to switch your role back to **awsstudent** user (you can find the URL to launch in cloudformation output under key **SwitchRoleAwsStudent**).
2. Next, Navigate to **config** service.
3. Choose **Get Started**
4. In **Settings** screen under **resource types to record**, **uncheck All resources**,
5. under **Specific types**, specify **EC2: Instance** as Resource types to record.
6. Under Amazon S3 bucket, choose **Choose a bucket from your account** and specify bucket provided to you in the outputs section of the Cloudformation(the bucket will have **configbucket** in its name)
7. Under AWS Config role, choose **"Use an existing AWS Config service-linked role"** and then choose **Next**.
8. On AWS Config Rules screen, choose **Skip**.
9. On Review screen, choose **Confirm**.

Resource types to record

Select the types of AWS resources for which you want AWS Config to record configuration changes. By default, AWS Config records configuration changes for all supported resources. resources in this region.

All resources

- ☐ Record all resources supported in this region ⓘ
- ☐ Include global resources (e.g., AWS IAM resources) ⓘ

Specific types

EC2: Instance x

Amazon S3 bucket*

Your bucket receives configuration history and configuration snapshot files, which contain details for the resources that AWS Config records.

- ☐ Create a bucket
- ☒ Choose a bucket from your account
- ☐ Choose a bucket from another account ⓘ

Bucket name* config-configbucket-a0q741fg... / Prefix (optional) / AWSLogs/883739019404/Config/us-east-1

Amazon SNS topic

- ☐ Stream configuration changes and notifications to an Amazon SNS topic.

AWS Config role*

Grant AWS Config read-only access to your AWS resources so that it can record configuration information, and grant it permission to send this information to Amazon S3 and Amazon

- ☒ Use an existing AWS Config service-linked role
- ☐ Choose a role from your account

1. Choose **Rules** from the left panel, this will open the AWS Config rules that have been configured in your AWS account.
2. Next, choose the **ConfigRuleForCheckIfInstanceIsNotOfTypeT2MicroVerification** rule to view the rule details.
3. Note the **trigger type** - the values are configuration changes and periodic. The rule has been configured to trigger re-evaluation every time an EC2 instance starts/stops/terminates.
4. Note the **Resources evaluated** section. Later, after you launch an EC2 instance in your environment, you will find that non-compliant as well as, compliant resources appear here.

Next, you need to view the CloudWatch Events rule which publishes compliance change notifications for the config rule **ConfigRuleForCheckIfInstanceIsNotOfTypeT2MicroVerification**.

To view the CloudWatch Events rule:

1. Open CloudWatch by launching following link - <https://console.aws.amazon.com/cloudwatch/>
2. Ensure you are in correct region.
3. Choose **Rules** in left panel, choose the name **T2MicroConfigRuleComplianceChangeNotification** to open it. You will see that the CloudWatch events rule publishes a notification everytime compliancechangenotification message is published by the Config rule you viewed earlier, on **T2MicroConfigRuleTopic** SNS topic.

Rules > T2MicroConfigRuleComplianceChangeNotification

Summary

ARN ⓘ `arn:aws:events:eu-west-1:836555555555:rule/T2MicroConfigRuleComplianceChangeNotification`

Event pattern ⓘ

```
{
  "detail-type": [
    "Config Rules Compliance Change"
  ],
  "source": [
    "aws.config"
  ],
  "detail": {
    "configRuleName": [
      "ConfigRuleForCheckIfInstanceIsNotOfTypeT2MicroVerification"
    ],
    "messageType": [
      "ComplianceChangeNotification"
    ]
  }
}
```

Status Enabled

Description T2MicroConfigRuleComplianceChangeNotification Event Rule

Monitoring [Show metrics for the rule](#)

Targets

Filter: <input type="text"/>				
Type	Name	Input	Role	Additional paramet
SNS topic	T2MicroConfigRuleTopic	Matched event		

Next, you need to need to create a subscription to the **T2MicroConfigRuleTopic** topic.

1. Open SNS service by launching <https://console.aws.amazon.com/sns/> in a browser tab, ensure that you are in correct region.
2. Choose **Topics** from left panel and then click on ARN of the **T2MicroConfigRuleTopic** topic to view the topic details.
3. Under **Subscriptions**, choose **Create Subscription** to create a subscription for ServiceNow.
4. Choose **Https** as the protocol
5. Specify your ServiceNow instance URL as the endpoint:
 - a. `https://admin:<ServiceNow admin password>@<your developer instance>.service-now.com/api/x_snc_aws_sns/aws_sns`
 - b. Note: `https://rel-oct12shm-005.lab.service-now.com/` is your ServiceNow URL, then **rel-oct12shm-005.lab** is the value of your developer instance)
 - c. Here is an example of a value: `https://admin:mypassword@rel-decxxxshm-005.lab.service-now.com/api/x_snc_aws_sns/aws_sns`
6. Before you click on **Create subscription**, specify password and developer instance name in the endpoint.
7. Next, Choose **Create Subscription**.

Create subscription

Topic ARN

arn:aws:sns:us-east-1:██████████:MicroConfigRuleTopic

Protocol

HTTPS

Endpoint

https://admin.██████████@██████████.service-now.com/api/x_snc_aws_s

Cancel

Create subscription

SUBTASK 4.2: SUBSCRIBE TO THE TOPIC FROM SERVICENOW.

Now you will log in to your ServiceNow instance and accept the pending subscription.

Before SNS is allowed to send messages to ServiceNow, you must confirm the subscription on ServiceNow. At this point, AWS has already sent a handshake request, and it's awaiting confirmation inside your ServiceNow instance.

Next, choose **Subscriptions** under **SNS** by using the filter in the left panel of your ServiceNow screen, and notice that a new record has been created by AWS.

Subscriptions	New	Go to	Name	Search	1 to 1 of 1
All					
			Name	State	Subscribe URL
			ServiceNow	Pending Confirmation	https://us-east-1.amazonaws.com/Subscription?TopicArn=us-east-1:██████████:MicroConfigRuleTopic

Open the subscription by choosing the name displayed(E.g. in above screenshot, **ServiceNow**), and then choosing **Confirm Subscription**.

Subscription ServiceNow

Name: ServiceNow

State: Pending Confirmation

Resource Names

Topic ARN: arn:aws:sns:us-east-1:757136350500:ServiceNow

Subscription ARN:

Update Confirm Subscription Delete

Handlers New Edit... Go to Order Search

Subscription = ServiceNow

Handler Order Active

Stay on this page, because you will need to create a handler next.

Now let's do something meaningful whenever SNS sends a notification. ServiceNow provides a script "Handler" that is invoked when SNS sends an alarm message. To configure a handler to create an incident, follow the instructions below:

- At the bottom of the **Subscription** form, find the Handlers section.
- Choose **New** and type a name for the handler, such as "Create SNS Non-micro launch Incident."
- Replace line 3-6 with following code (just inside the function):

```
if(message.detail.newEvaluationResult.complianceType == 'NON_COMPLIANT'){
    var incident = new GlideRecord("incident");
    incident.initialize();
    incident.short_description = "SNS Alarm: " + message.detail.configRuleName;
    incident.description = "AWS Account ID: " + message.account + "\nRegion: " + message.detail.awsRegion +
    "\nCompliance status: " + message.detail.newEvaluationResult.complianceType;
    incident.insert();
}
```

- Choose **Submit** to save the handler. Here is how your code will look.

SUBTASK 4.3: LAUNCH AN EC2 SERVER FROM SERVICENOW OF THE TYPE T2.MEDIUM AND SEE AN INCIDENT GETTING CREATED IN SERVICENOW.

1. Go back to the ServiceNow browser tab, open the **system administrator** menu (Top right of the screen) and then Choose **Impersonate User**, choose **SCEndUser01** as the impersonation.
2. Open **Service Catalog** under **Self Service** from the left navigation panel. This will display Service Catalog Home Screen. Choose **AWS Service Catalog**.

Service Catalog



AWS Service Catalog


Order products from the AWS Service Catalog.

4. You will see two products here- One an EC2 instance product and another an Amazon workspaces desktop.
5. Choose **EC2 instance**, then specify Product name as **My-Snow-EC2-Instance**.
 1. Subnet ID – Specify Public Subnet ID provided to you in the outputs section of the Cloudformation
 2. Security Group - Specify Security group provided to you in the outputs section of the Cloudformation
 3. AMI - Specify AMI provided to you in the outputs section of the Cloudformation
 4. InstanceType – `t2.medium`
5. Choose **Order Now** from Top Right of the screen. This will start the creation of an EC2 instance in your AWS account.
6. Next, open **My Assets** from navigation panel in left.
7. Scroll down, and at the end of the page, under “**My Asset Requests**”, choose **My-Snow-EC2-Instance**, under configuration item column.
8. On AWS Service Catalog product screen, once **product status** becomes **provisioned** (you can refresh the screen in few seconds if you dont see this yet), you will see that **Request Termination** & **Request update** options have become available. This means that your EC2 instance have been launched.
9. You can optionally click on **Request Self-Service action** to stop/start your EC2 instance via ServiceNow. Here is how it looks, when you click on “Request Self-service action” you can see self-service actions you configured in your AWS Service catalog. :

Assigned to

Product Status **Provisioned**

Related Links

[Request Self-Service Action](#) 

[Request Termination](#)

[Request Update](#)

[Subscribe](#)

[Sync Product](#)



- ☐ **AWS-StopEC2Instance**
Stop EC2 instances(s)
- ☐ **AWS-StartEC2Instance**
Start EC2 instances(s)

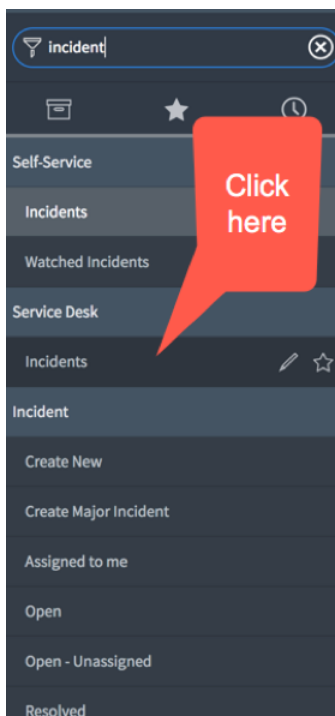
Select a self-service action.

For this workshop, you do not need to click on **Execute**. This was just for you to know that service-actions can be made available via ServiceNow as well.

Now that you have launched an EC2 instance that makes Config rule non-compliant, you should be able to see an incident getting created in ServiceNow.

Note: An AWS Config rule may take some time to run and report results. If it is taking longer, open AWS Config service, from the left panel, choose rules. Next, choose the **ConfigRuleForCheckIfInstanceIsNotOfTypeT2MicroVerification** rule to view the rule details. Once you see **evaluation details** section populated with the newly non-compliant EC2 instance details, you are ready to check ServiceNow.

1. Navigate to ServiceNow. On ServiceNow browser tab, open the **SCEndUser01** menu (Top right of the screen) and then Choose **Impersonate User**, and then select **system administrator**.
2. In the left panel, type incidents and then choose incidents available under Service Desk menu



You will see an incident. This means you have successfully integrated SNS and ServiceNow.

The principle can be extended to any type of SNS topic that notifies ServiceNow whenever anything meaningful happens inside your AWS Cloud environment. For example, you can configure ServiceNow to provision AWS resources by leveraging the [AWS Service Catalog Connector for ServiceNow](#) and configure it such that any alarms from CloudWatch for newly created resources would create an incident in ServiceNow. Within ServiceNow's SNS Handlers,

you can create any type of ServiceNow record you like. It can kick off an automated workflow, or create Events/Alerts/Notifications, update CMDB or even automatically orchestrate a remediation.

Important performance note

This example is meant for low-volume scenarios only, such as occasional billing alarms. It is not designed to handle full operational monitoring scale of processing and will very quickly consume all your API semaphores. ServiceNow offers fully-supported solutions for high-volume operational monitoring included with Event Management:

[AWS Config with ServiceNow](#)

[CloudWatch Alarms with ServiceNow](#)

[CloudWatch Metrics with ServiceNow](#)

If you have spare time left, try out the cool brownie point tasks otherwise, you can go to CloudFormation console and end the qwiklab.

Brownie point task 1 (optional)

In this section, you will learn how to create a template constraint to limit the values your end-users can specify for CloudFormation input parameters. As part of this task you will add a constraint outside CloudFormation template to allow your end users to launch only a t2.micro EC2 instance.

Note: You will use template constraint instead of modifying CloudFormation template because you would like to also use the same EC2 instance product in another portfolio created for deployment-specialists who need to higher instance types.

Next, you will configure your Portfolio to allow creation of only **t2.micro** EC2 instance. To do so:

1. Next, you need to log out of your AWS account and log back in.
2. You will be switching role **service_catalog_administrator**. To do this, copy the URL present in the **SwitchRoleSCAdmin** field in the outputs section of Cloudformation and launch it in your browser, then click on Switch Role button.
3. In AWS console, Choose the region specified in the outputs section of the Cloudformation.
4. Open AWS Service catalog by choosing the **Service catalog** service from **services** menu.
5. Navigate to the portfolio management screen of **SNOW-SC Test Portfolio**.
6. Expand the **Constraints** section, choose **ADD CONSTRAINTS**.
7. Choose **EC2 instance** as the product, **Template** as the constraint.
8. Choose **Continue**.

1. Specify Description as Enforcing instance type to be t2.micro
2. Specify following in Template Constraint textarea:

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

7. Choose **Submit**.
8. Sign out of AWS management console by choosing Sign out from the menu (expand menu left to the region menu in the top right of your screen).

You have just created a template constraint to ensure that end user can launch only a t2.micro EC2 instance and not any other EC2 instance-type. Now, you will verify whether user can launch t2.medium EC2 instance or not.

1. Next, you would switch roles to **service_catalog_end_user**. To do this, copy the URL present in the **SwitchRoleSCEndUser** field in the outputs section and launch it in your browser, then click on Switch Role button. Next, open the AWS Service Catalog console by launching <https://console.aws.amazon.com/servicecatalog/>.
2. In AWS console, Choose the region specified in the outputs section of the Cloudformation.
3. In the left panel, choose **Products list**. In products list page, choose **EC2 instance** and then choose **Launch product**.

4. On the Product version page, for Name, type My-test-EC2-instance.
5. In the Version table, choose v1.0
6. Choose **Next**.
7. On the Parameters page, try choosing/typing t2.medium. You will notice that you are not able to.
 - Choose **Cancel**. You do not need to launch a t2.micro instance.

You just learned how to use template constraints. By creating template constraints specific to portfolio, you can reuse the product across multiple portfolios. Things like subnet/security groups/tagoptions/cloudformation parameter values can be enforced at the portfolio level. This way, you do not need to manage redundant copies of same CloudFormation template with specific parameter values for certain groups.

To know how to dynamically constrain parameter options in AWS Service Catalog based on specific tag values, see [documentation](#).

Finally, you can go and shutdown EC2 instances created to avoid charges.