Project Title: - Start and Stop AWS Ec2 Instance As per Prefer time.

Introduction: - Task about the rising cost of our production and need to save money by stopping our EC2 instances after all engineers are clocked out.

Pre-requisite: - AWS Account

AWS Services used in the Task: -

A) Ec2

B) IAM

C) Lambda

D) Cloud Watch

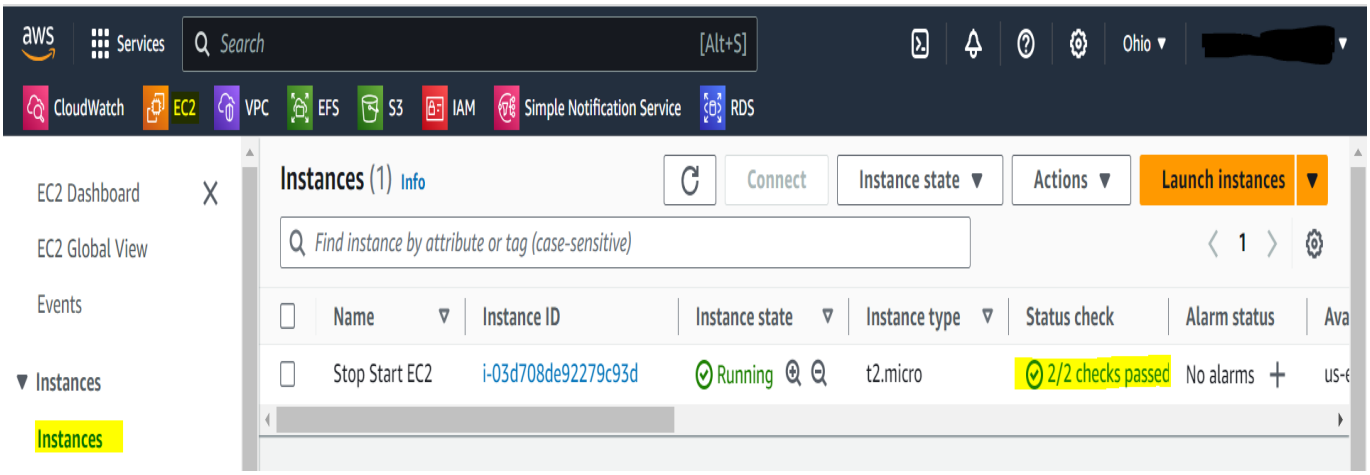
Start performing Task: -

Step 1: Manually Launch EC2

The first step in this project would be to manually launch any number of **EC2 instances**. These instances will be used solely as a means to test our lambda functions.

In the AWS console head over to the search bar and type in "EC2". EC2 can also be found in services under the "**Compute**" submenu.

Once the EC2 dashboard is displayed, on the left-hand side click on the orange button title "***Launch Instance***".

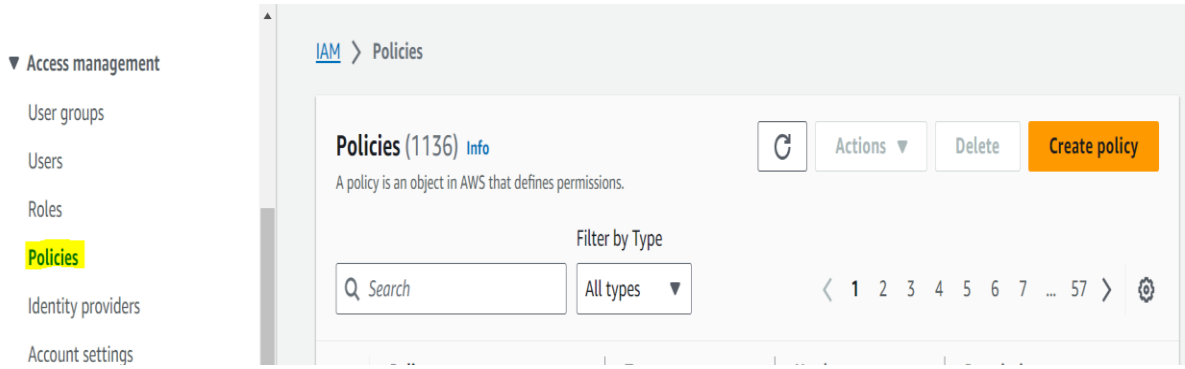


Step 2: IAM Policy & Creation

The second step in this project would be to create a policy and a role that will be used by our lambda function.

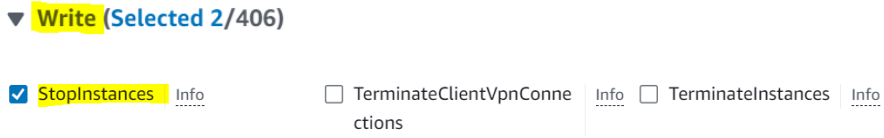
In the AWS console head over to the search bar and type in "IAM". IAM can also be found in services under the "**Security, Identity, & Compliance**" submenu.

Once the IAM dashboard is displayed, on the left-hand side click on "**Policies**" and then click on the orange button title "***Create policy***".

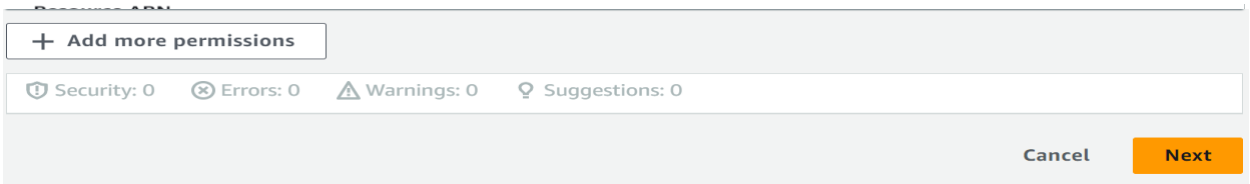


Under the "**Select Service**" menu type and select EC2. The next page should prompt you to select "**Access level**".

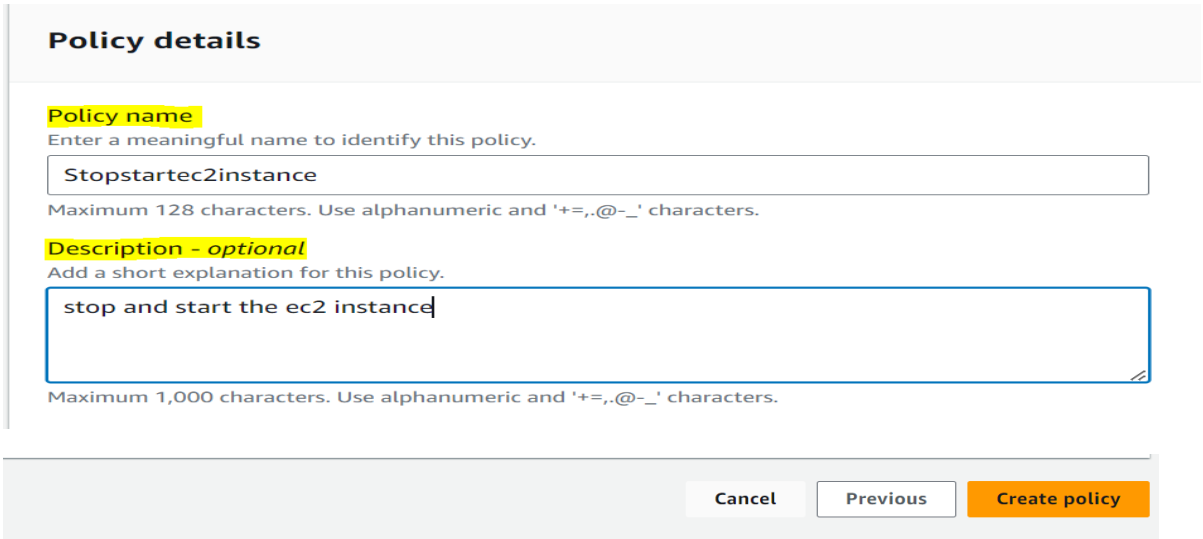
For Write access select "**StartInstances, StopInstances**".



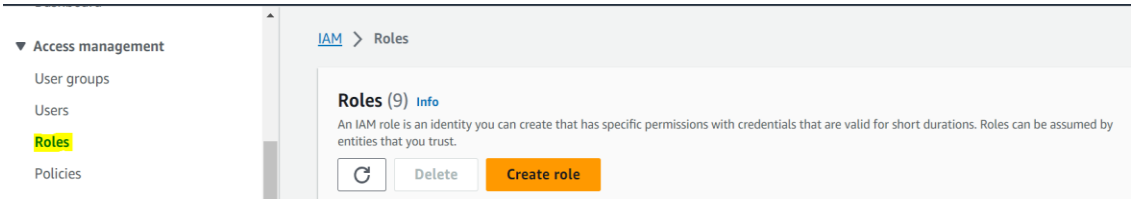
In the Resources click on **Add ARNs** After Filling all required information and click on **Add ARNs** then click on **Next**



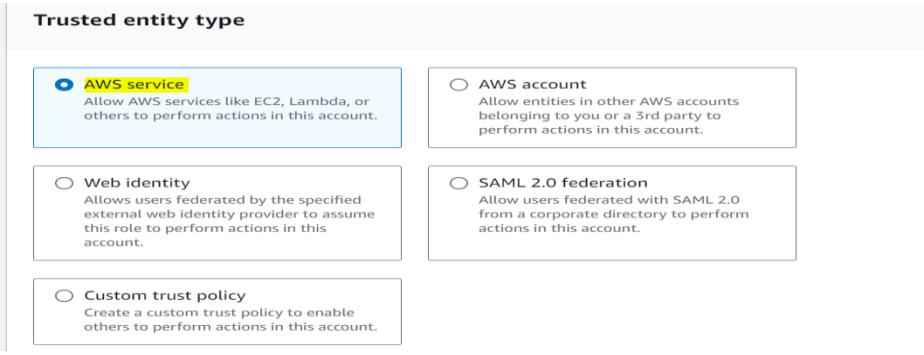
Add policy details and then click on **Create Policy**

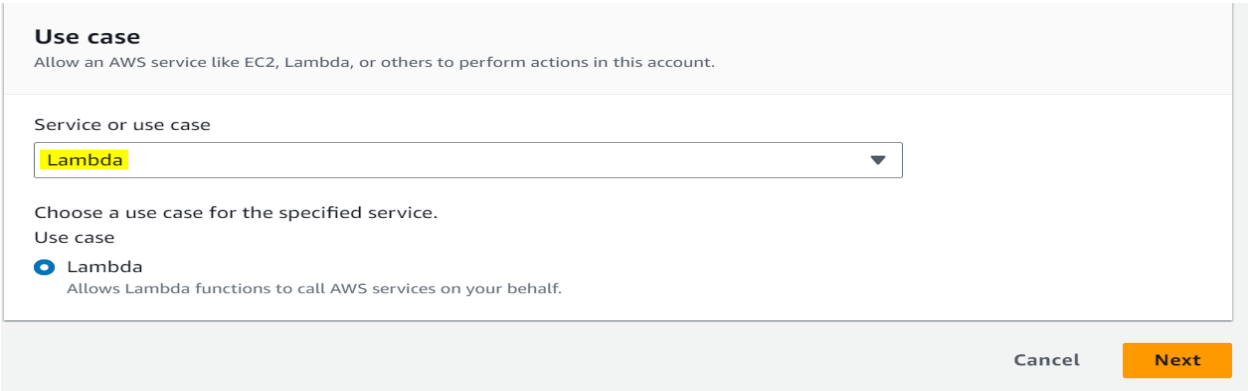


Once our policy has been created, on the left-hand side click on "**Roles**" and then click on the orange button title "***Create roles***".

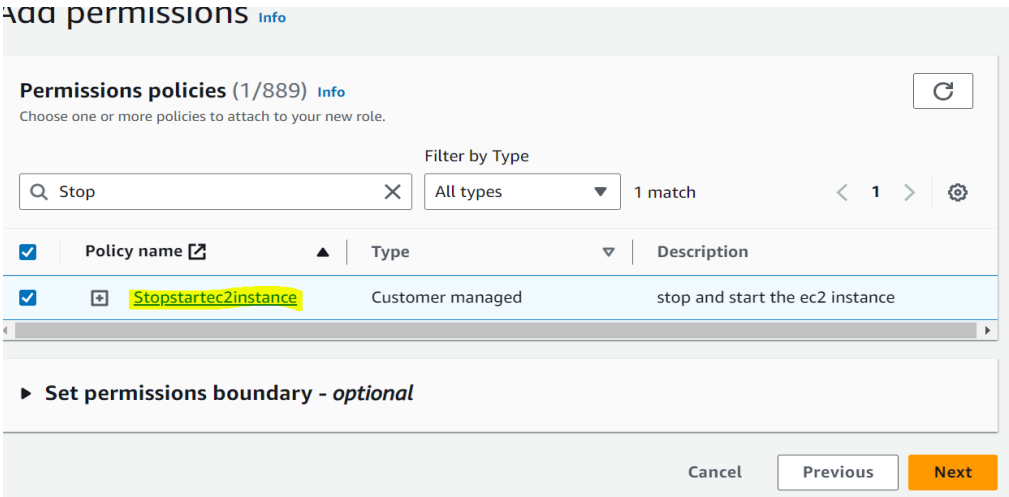


Select the **“AWS Service”** for “**Trusted entity type**” and **“Lambda”** for “**Use case**”, then click on **“next”**.

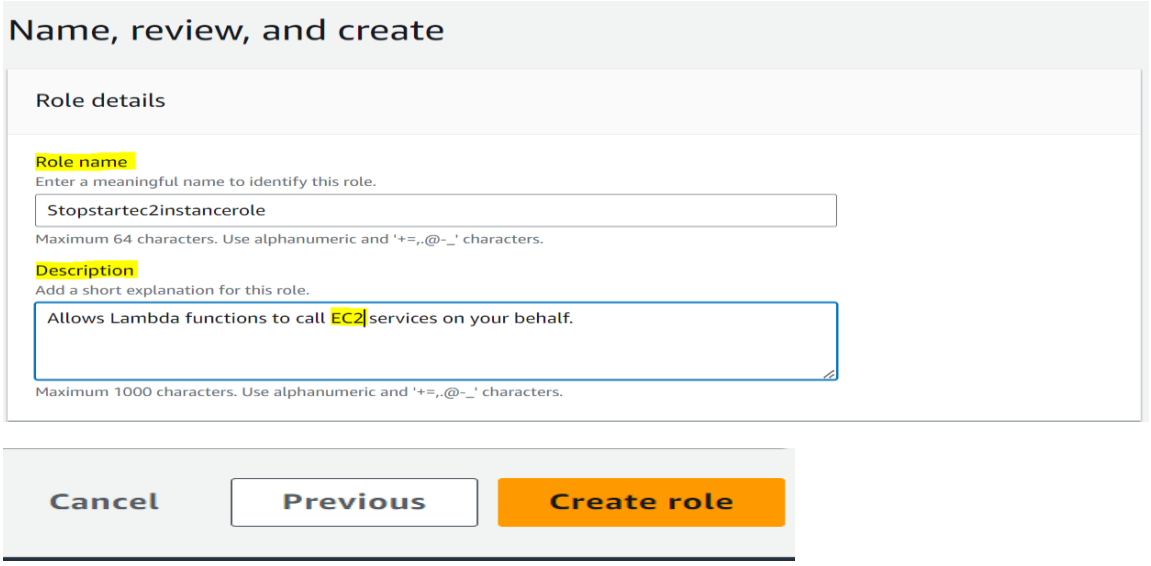




search and select the **policy** created in the steps above then click next.



create a name and click on **“Create role”**.

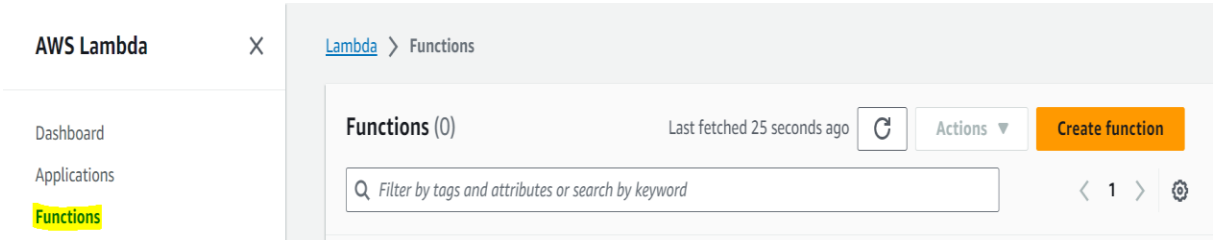


Step 3: Lambda Functions Creation

The third step in this project would be to create our lambda function. We are also going to assign the role we created in the step above in order for our function to be able to start and stop our instances as needed.

In the AWS console head over to the search bar and type in "Lambda". Lambda can also be found in services under the "**Compute**" submenu.

Once the Lambda dashboard is displayed, on the right-hand side click on the orange button title "***Create functions***".



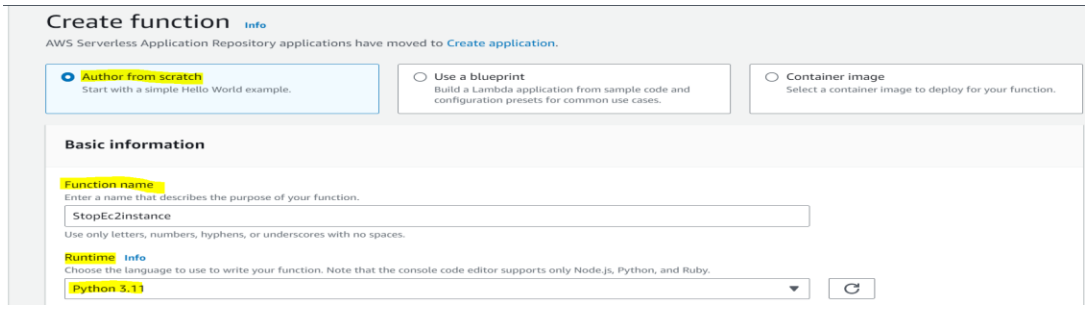
Author from scratch

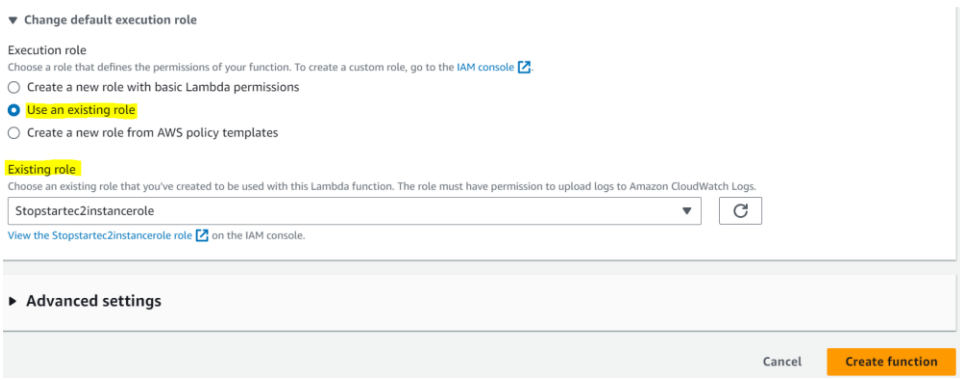
Create a name for our function.

Select **"Python 3.11"** for our runtime.

Select the role we created earlier under the **"Change default execution role"** option.

Then click on "**Create functions**" for **stop** running ec2 Instance



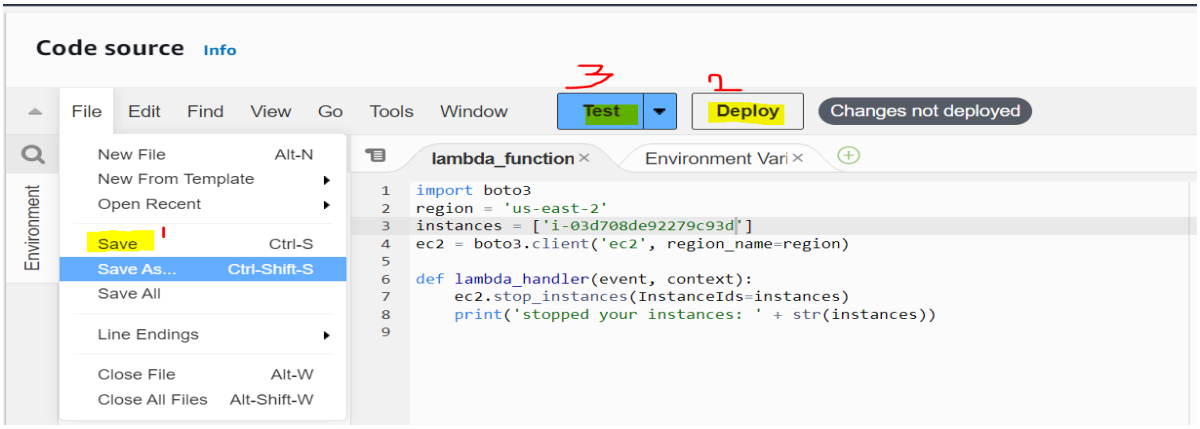


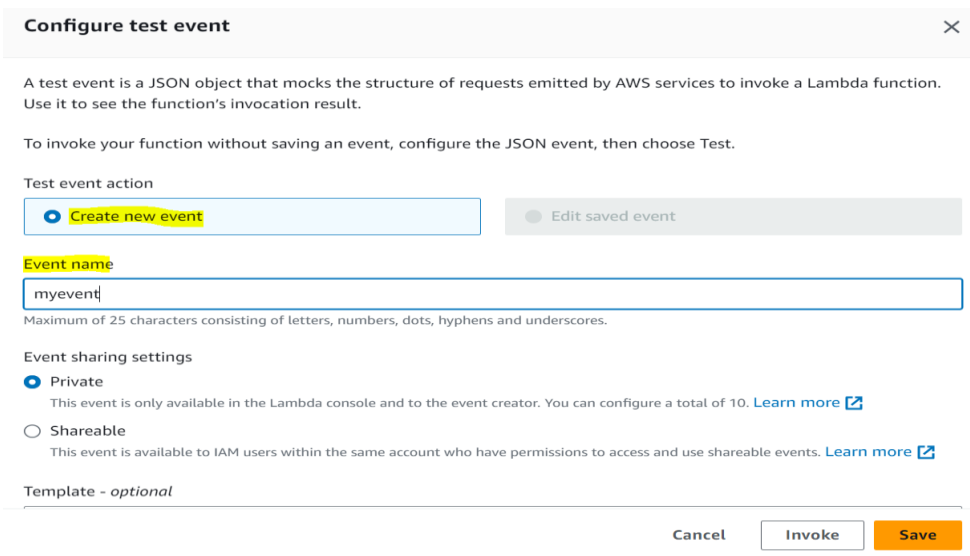
- Python code needed to **Stop**

Stop the instances by using python code: -

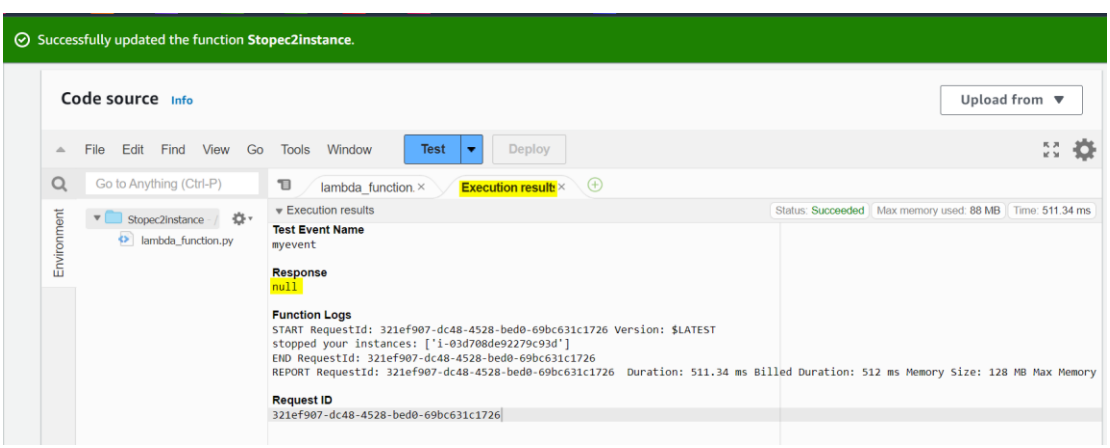
```
import boto3
region = 'us-west-1'
instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.stop_instances (InstanceIds=instances)
```

print ('stopped your instances: ' + str(instances))

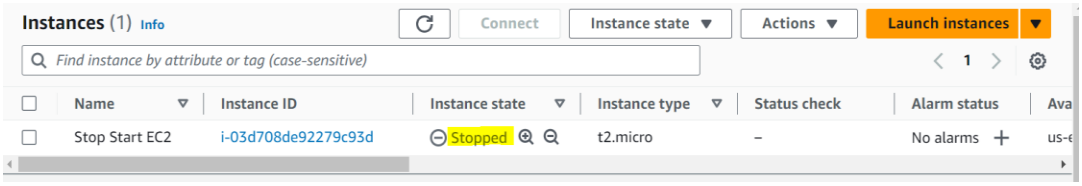




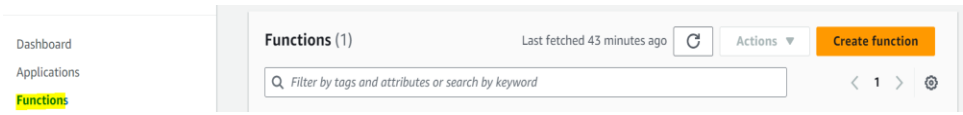
After creating Event then Click on **Test** and go back to the Ec2 and confirm our Ec2 is going to on stop stage or not.



Status of Ec2 Instance



- Create Lambda function for **Start Ec2** Instance



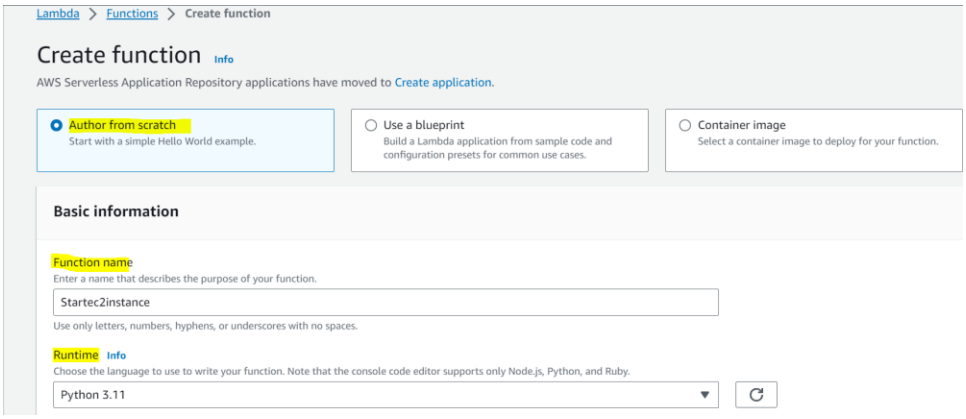
Author from scratch

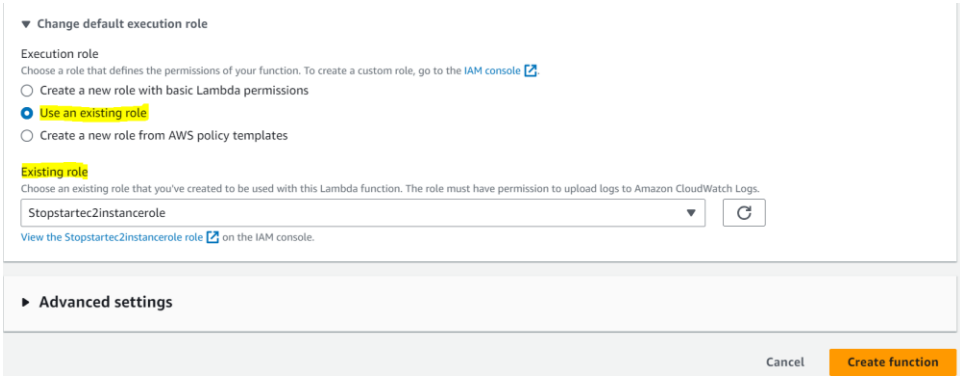
Create a name for our function.

Select **"Python 3.11"** for our runtime.

Select the role we created earlier under the **"Change default execution role"** option.

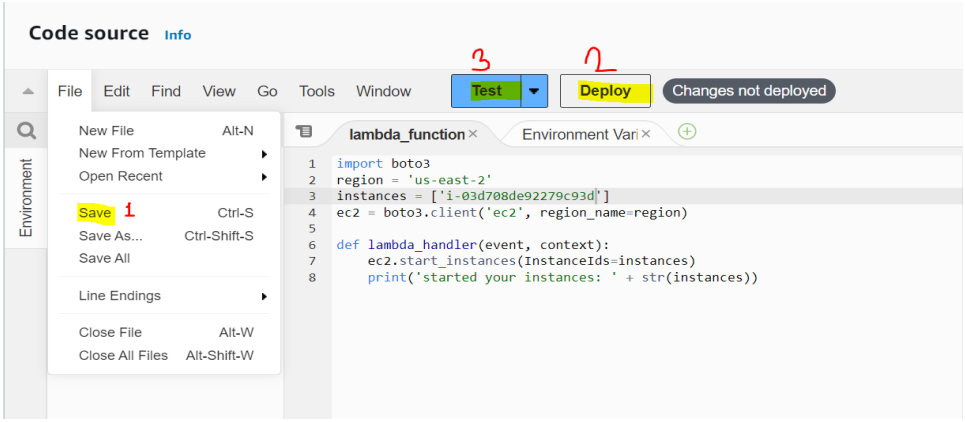
Then click on "**Create functions**" for **Start** Stopped ec2 Instance

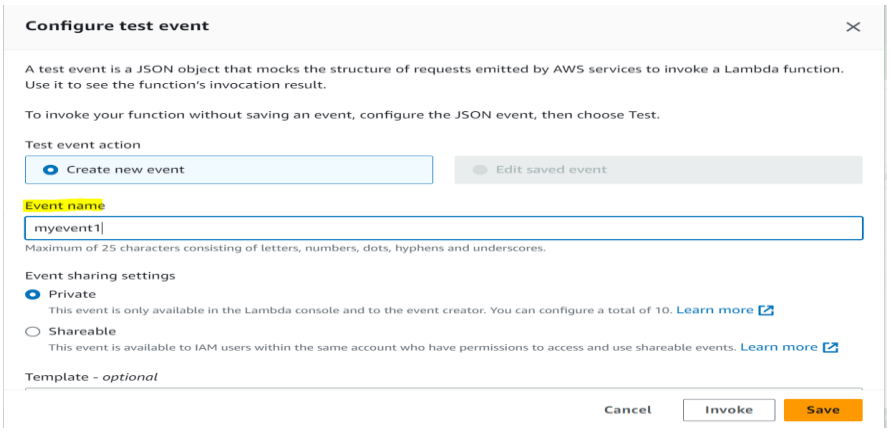




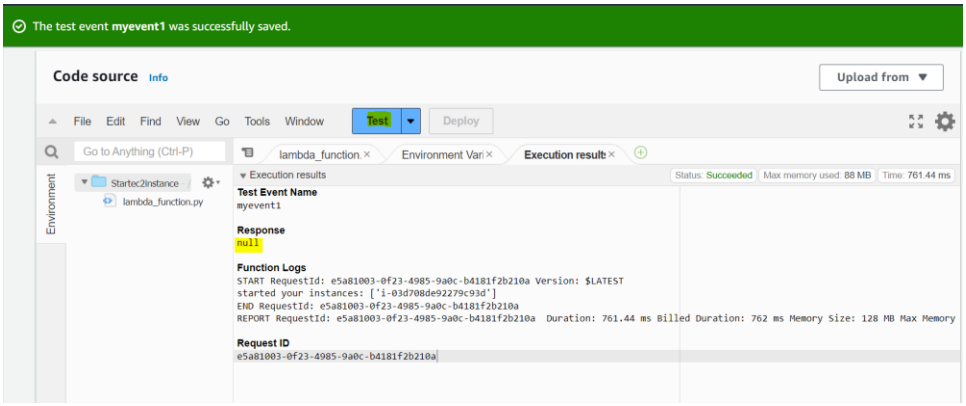
Start our ec2 instances by using python code.

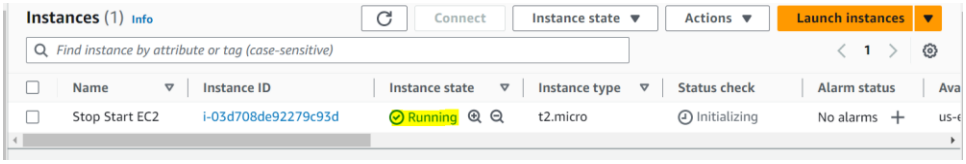
```
import boto3
region = 'us-west-1'
instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.start_instances (InstanceIds=instances)
    print ('started your instances: ' + str(instances))
```





After creating Event then Click on **Test** and go back to the Ec2 refresh and confirm our Ec2 is going to **start** or not.



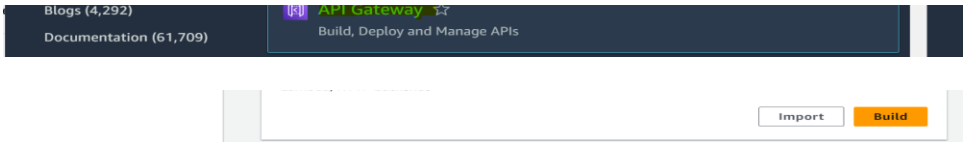


Step 4: Create the API Gateway and associate with the lambda function

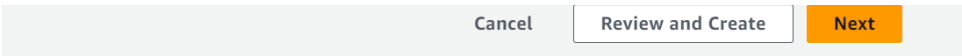
Go to API Gateway app

Go to Create API and select “HTTP API”

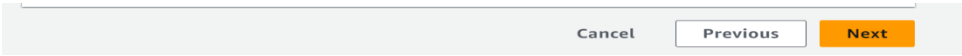
In the AWS console head over to the search bar and type in “API Gateway”



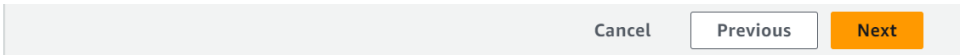
Click On **Build Then** Select **Lambda Integration type**, as a **Lambda** your region. You’ll be able to locate your **stop and Start** Lambda function in the dropdown list and fill your **API name** then click on **Next**

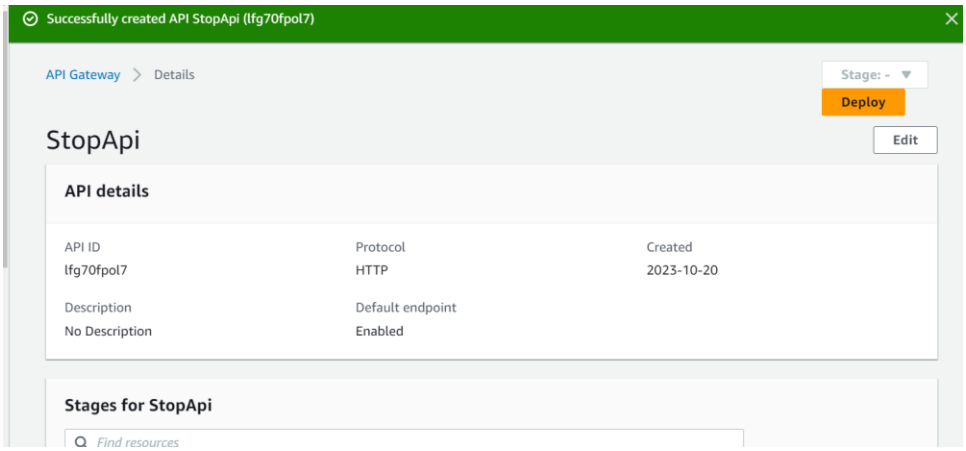


Configure your **API routes.** Then click on **Next**

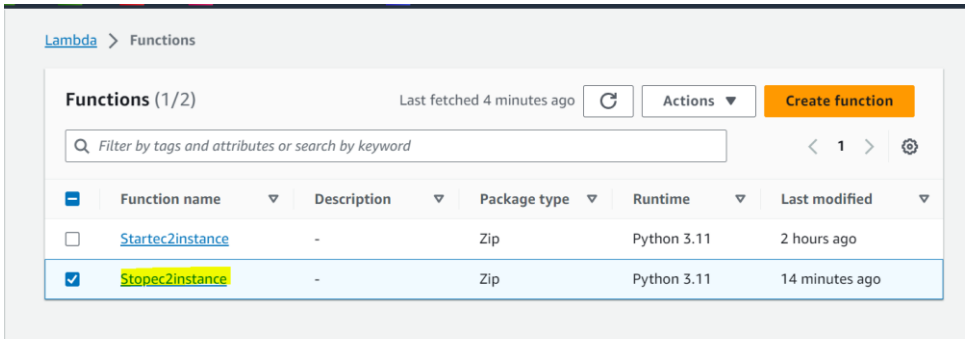


You can have several stages (stage, pre-prod, prod) for a route. Make sure to turn off the Auto-deploy feature. Add your **configure stages** then click on **Next** and then click on **create**

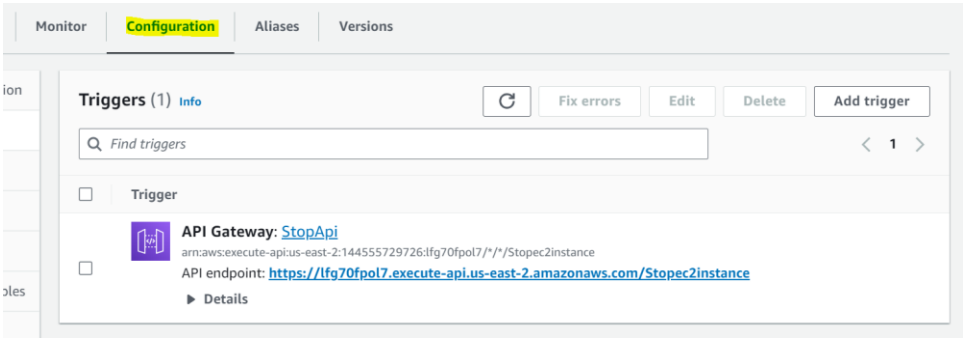




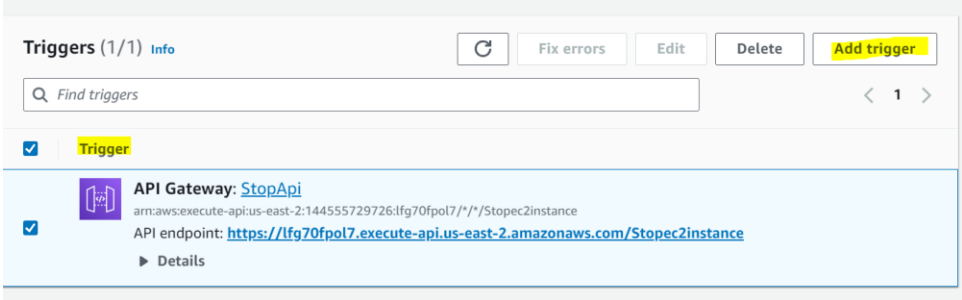
Go back to the **Lambda function** and choose **stop function**

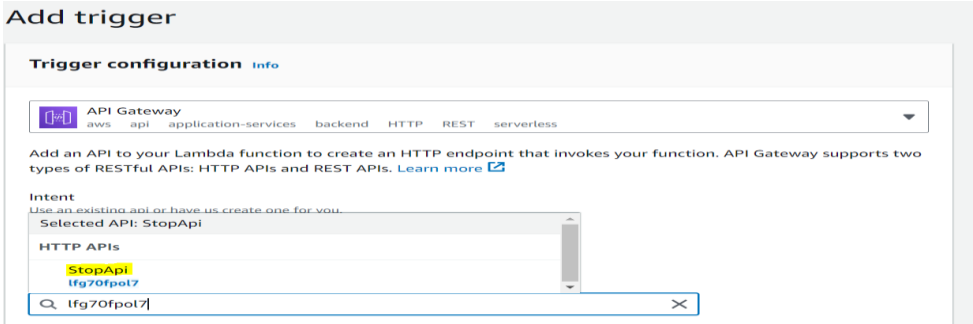


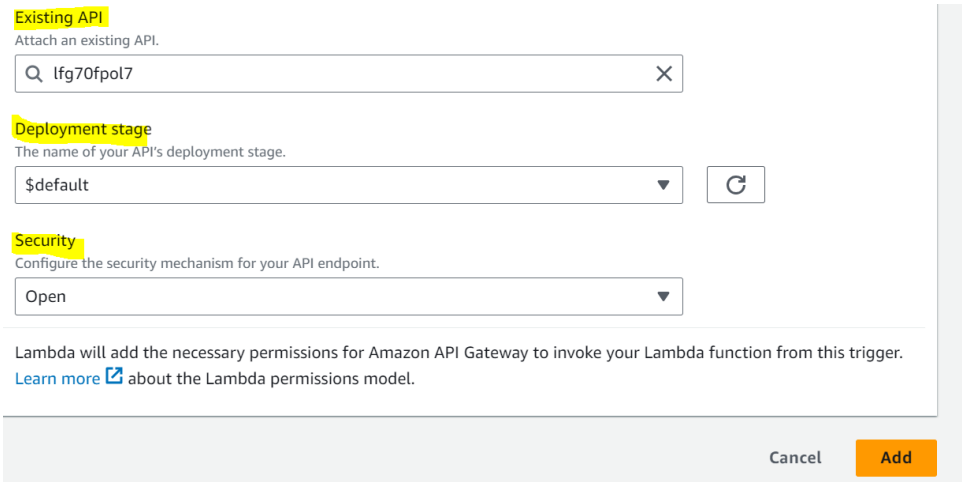
click on that **stop function** then click on **Configuration** then click on **Add**



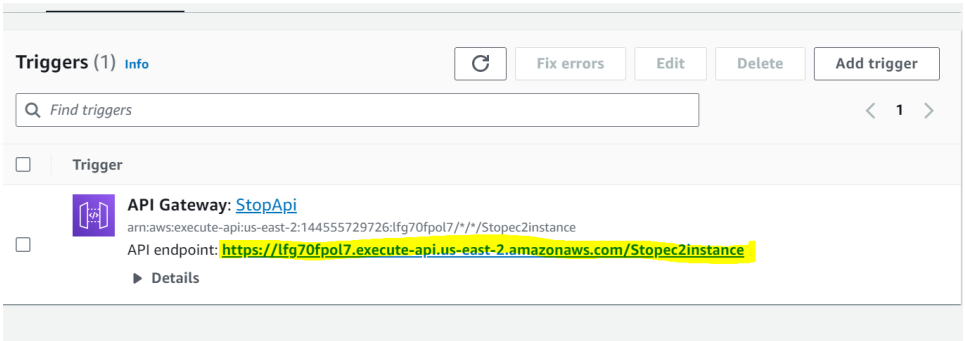
Select **API Gateway** and click on **Add trigger** and choose **API Gateway**

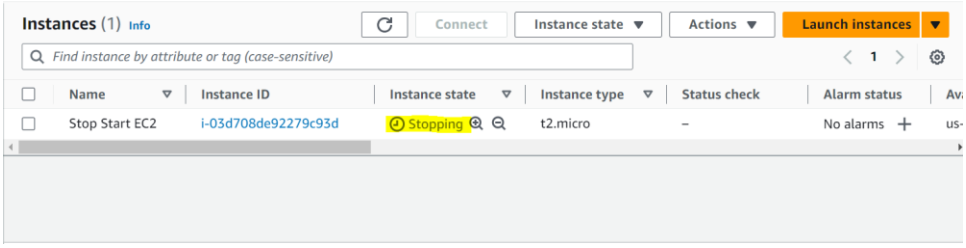






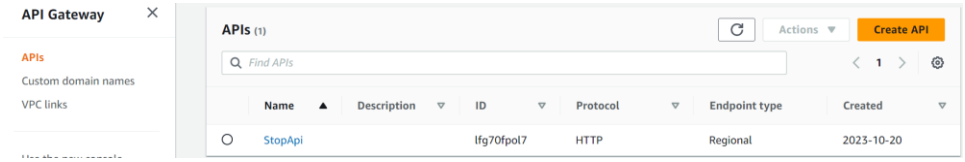
Click on **API Endpoint URL** and go back to **Ec2 dashboard** and check Instance status



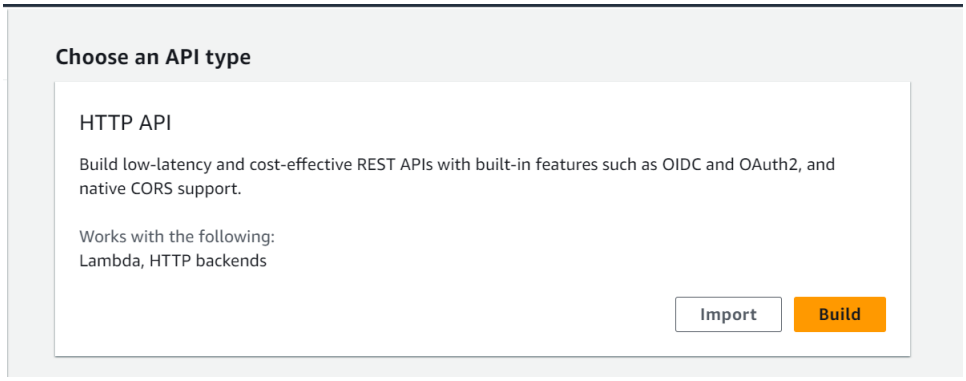


♦ Start EC2 Instance by triggering API Endpoint

Go to API Gateway then click on **Create API**



Choose API Type **Click on Build**



Integrations

Lambda | Remove

AWS Region: us-east-2

Lambda function: arn:aws:lambda:us-east-2:144555729726:fu

Version Learn more.: 2.0

Add integration

API name

An HTTP API must have a name. This name is cosmetic and does not have to be unique; you will use the API's ID (generated later) to programmatically refer to this API.

StartAPI

Cancel | Review and Create | Next

Configure routes

API Gateway uses routes to expose integrations to consumers of your API. Routes for HTTP APIs consist of two parts: an HTTP method and a resource path (e.g., GET /pets). You can define specific HTTP methods for your integration (GET, POST, PUT, PATCH, HEAD, OPTIONS, and DELETE) or use the ANY method to match all methods that you haven't defined on a given resource.

Method	Resource path		Integration target	
ANY	/Startec2instance	→	Startec2instance	Remove

Add route

Cancel | Previous | Next

Configure stages

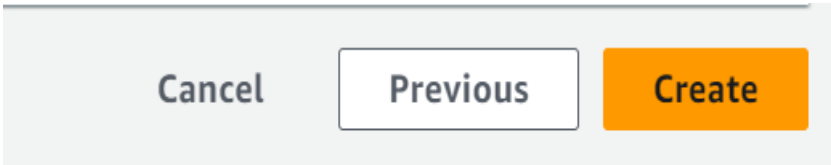
Stages are independently configurable environments that your API can be deployed to. You must deploy to a stage for API configuration changes to take effect, unless that stage is configured to autodeploy. By default, all HTTP APIs created through the console have a default stage named $default. All changes that you make to your API are autodeployed to that stage. You can add stages that represent environments such as development or production.

Stage name	Auto-deploy	
$default	ON	Remove

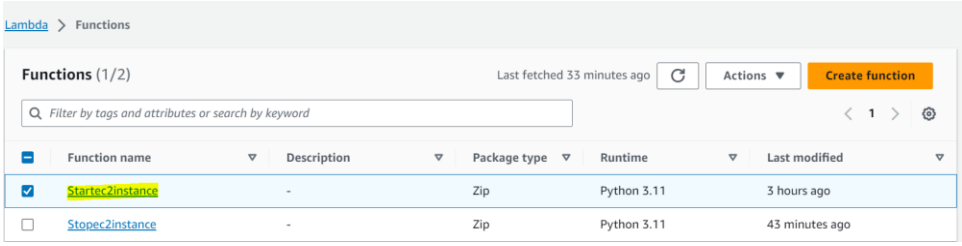
Add stage

Cancel | Previous | Next

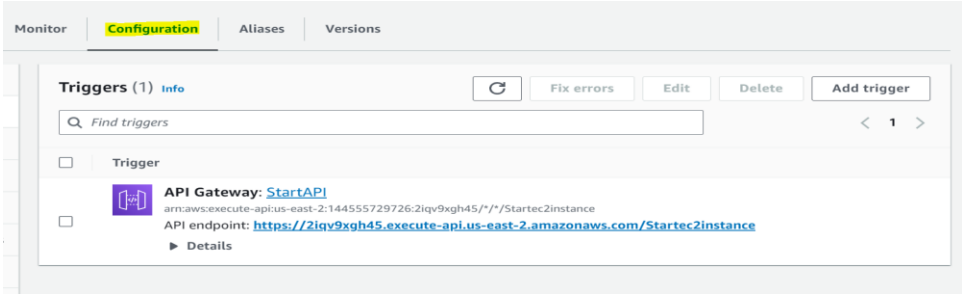
Then Click on **Create**



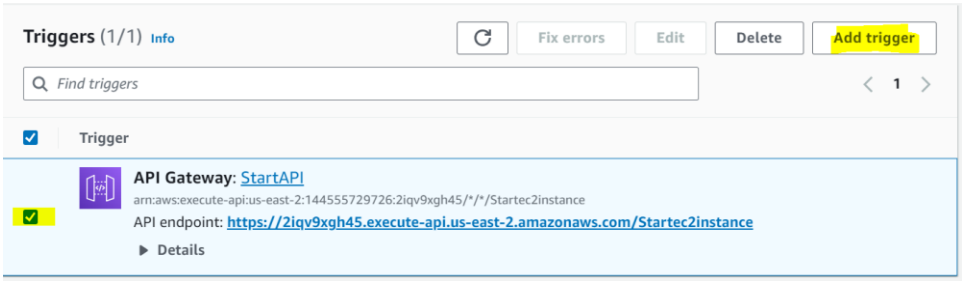
Go back to the **Lambda function** and choose **Start function**

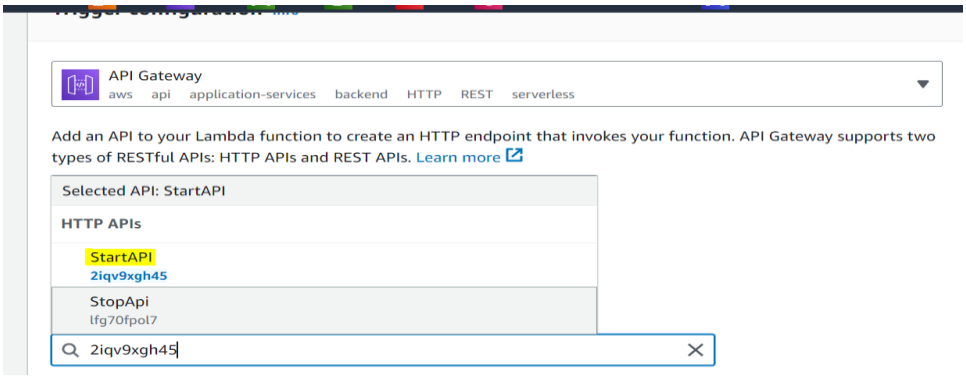


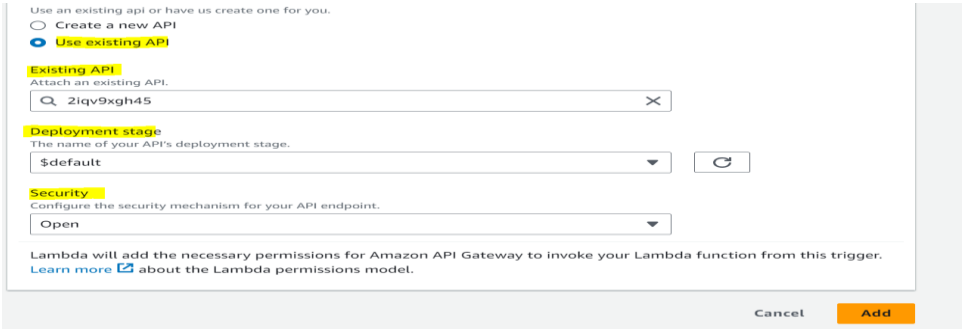
click on that **Start function** then click on **Configuration** then click on **Add**



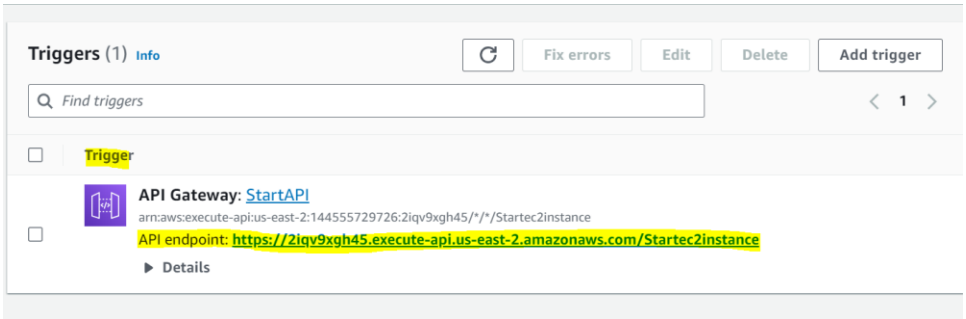
Select **API Gateway** and click on **Add trigger** and choose **API Gateway**

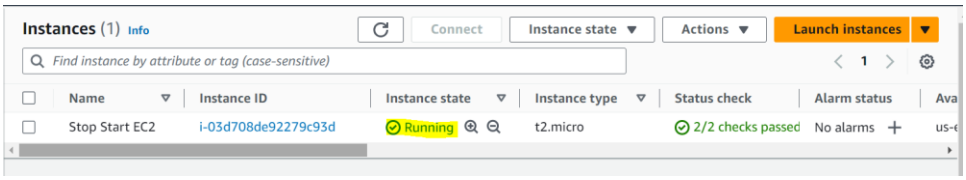






Click on **API Endpoint URL** and go back to **Ec2 dashboard** and check Instance status



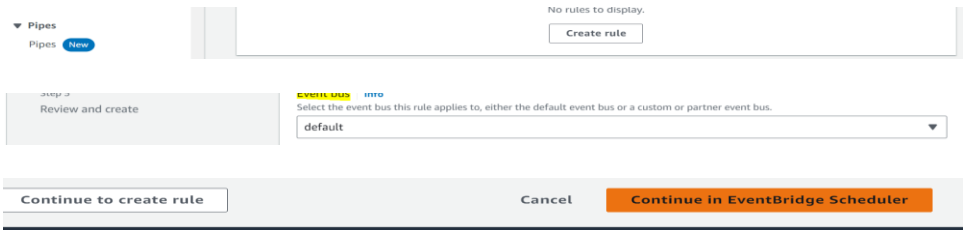


Step 4: Create EventBridge Rule to Trigger EC2 Instances By using AWS Cloud Watch Service and start

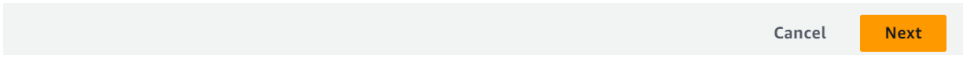
In this step, we are going to create an **EventBridge** Rule that triggers our EC2 instances. Being that Lambda is triggered by events, whenever an EC2 instance is stopped, it's going to send an event that will trigger our function.

In the AWS console head over to the search bar and type in **"EventBridge".** Once the page is displayed, on the right-hand side click on the orange button title ***"Create rule".***

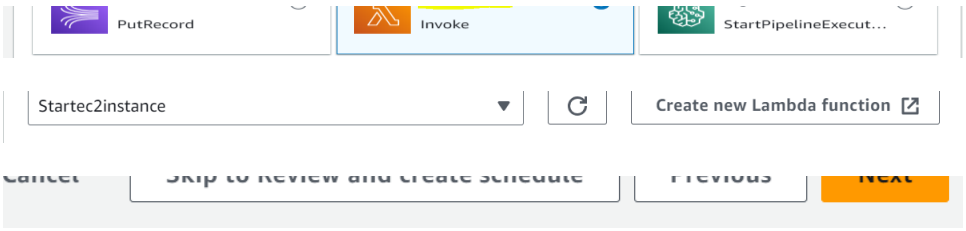
Provide a name and a description and under rule type select "***rule with an event pattern***".

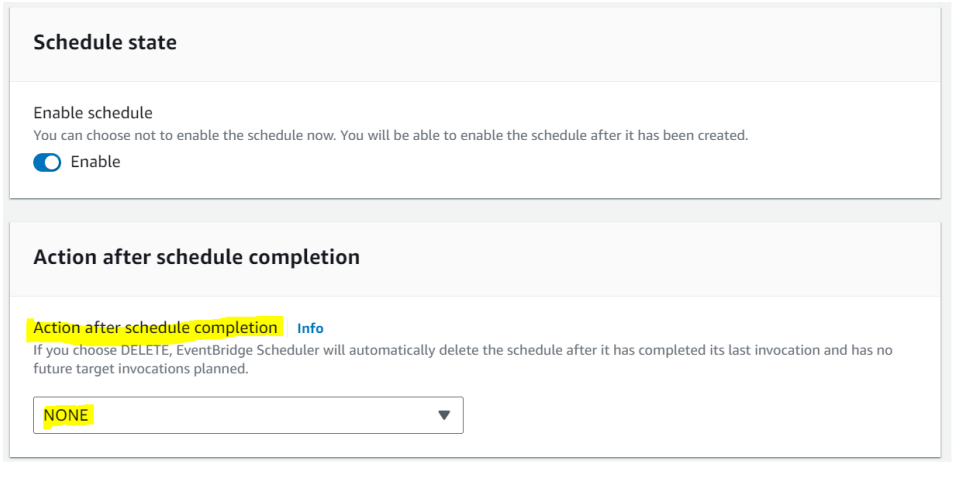


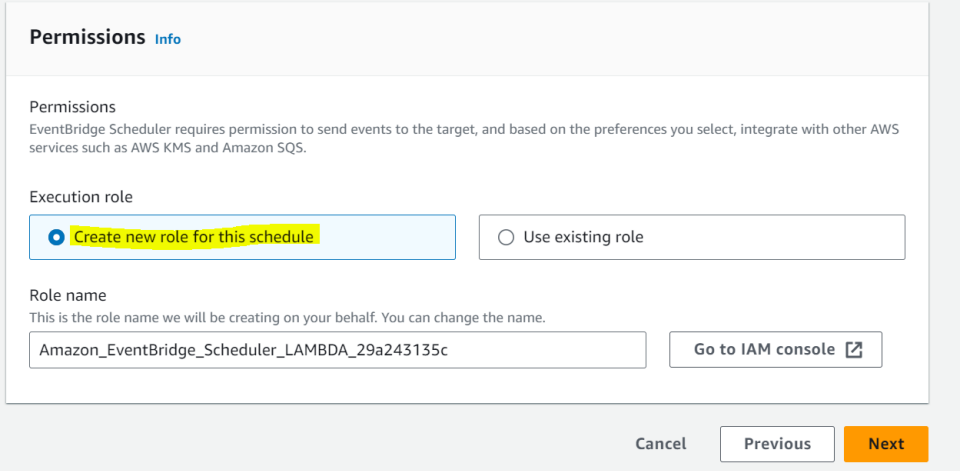
Fill all requirement as per your demand then click on **Next**



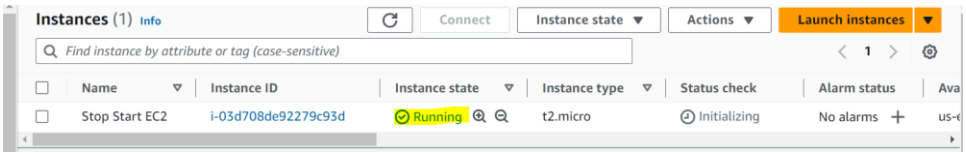
For event source select **AWS Services**.







After Clicking on create schedule Then go back to Ec2 console and check Instance status

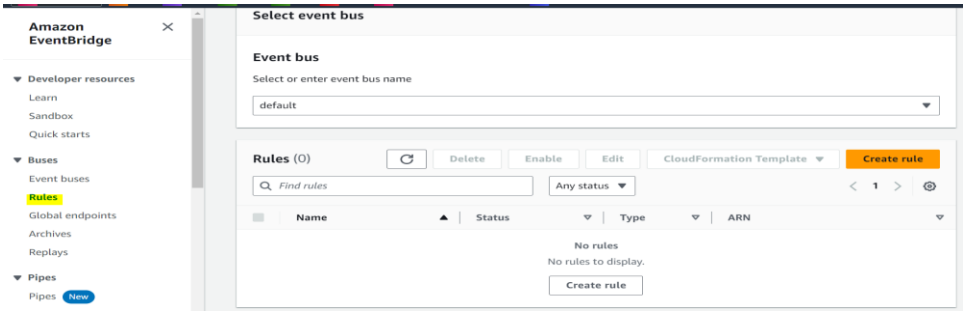


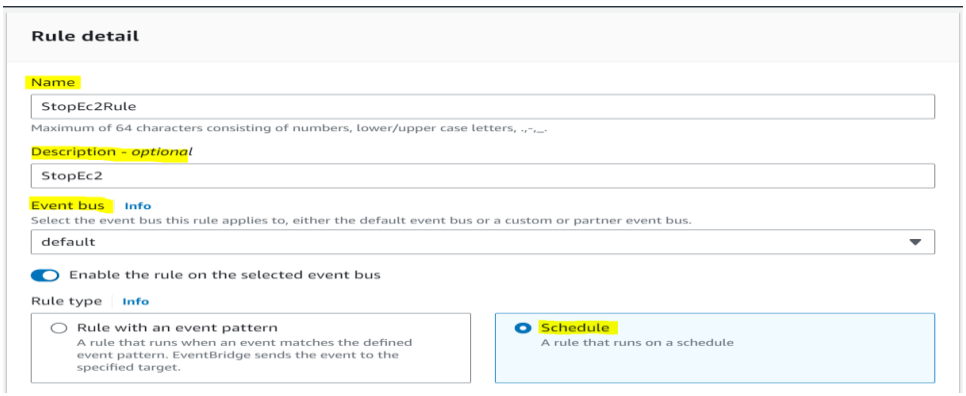
Create EventBridge to Stop the EC2

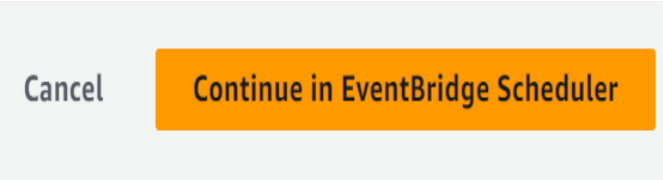
In this step, we are going to create an **EventBridge** Rule that triggers our EC2 instances. Being that Lambda is triggered by events, whenever an EC2 instance is stopped, it's going to send an event that will trigger our function.

In the AWS console head over to the search bar and type in **"EventBridge".** Once the page is displayed, on the right-hand side click on the orange button title ***"Create rule".*** For Stop the Ec2

Provide a name and a description and under rule type select "***rule with an event pattern***".







Schedule pattern

Occurrence Info

You can define an one-time or recurrent schedule.

- (●) One-time schedule
- ( ) Recurring schedule

Date and time

The date and time to invoke the target.

2023/10/20	23:20	(UTC+05:30) Asia/Calcutta
YYYY/MM/DD	Use 24-hour format timestamp (hh:mm)	Time zone

Flexible time window

If you choose a flexible time window, Scheduler invokes your schedule within the time window you specify. For example, if you choose 15 minutes, your schedule runs within 15 minutes after the schedule start time.

Off

Cancel Next

Target detail

Target API Info

Select an API that will be invoked as a target for your schedule.

- (●) Templated targets
- ( ) All APIs

CodeBuild StartBuild	CodePipeline StartPipelineExecut...	Amazon ECS RunTask
Amazon EventBridge PutEvents	Kinesis Data Firehose PutRecord	Amazon Inspector V1 StartAssessmentRun
Kinesis Data Streams PutRecord	AWS Lambda Invoke	SageMaker StartPipelineExecut...

Lambda function

Stopec2instance | Create new Lambda function

▶ Configure version/alias

Settings - *optional*

Schedule state

Enable schedule

You can choose not to enable the schedule now. You will be able to enable the schedule after it has been created.

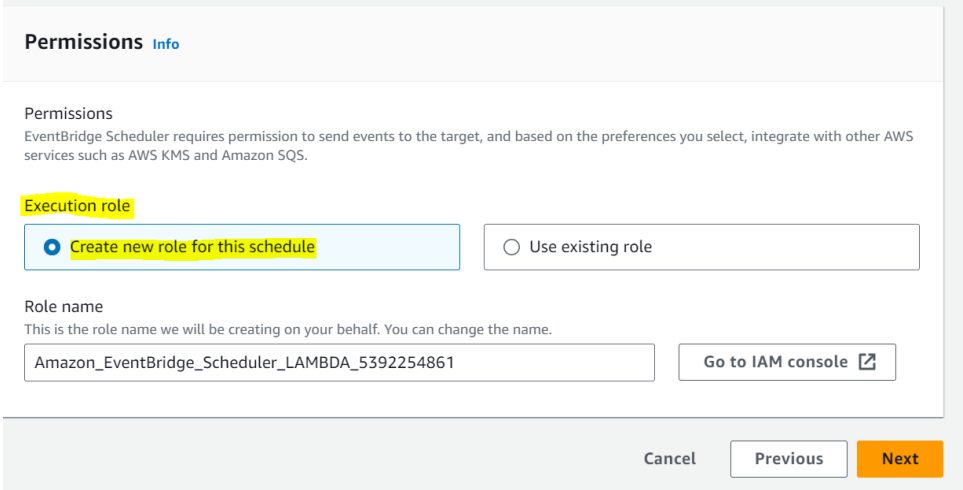
Enable

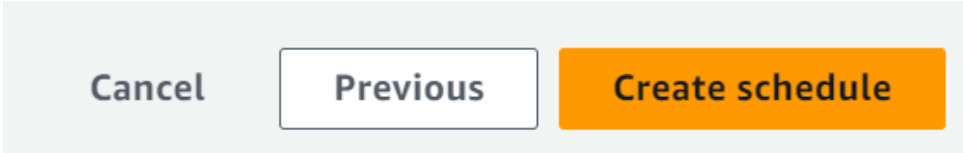
Action after schedule completion

Action after schedule completion Info

If you choose DELETE, EventBridge Scheduler will automatically delete the schedule after it has completed its last invocation and has no future target invocations planned.

NONE





After Clicking on create schedule Then go back to Ec2 console and check Instance status