

## 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

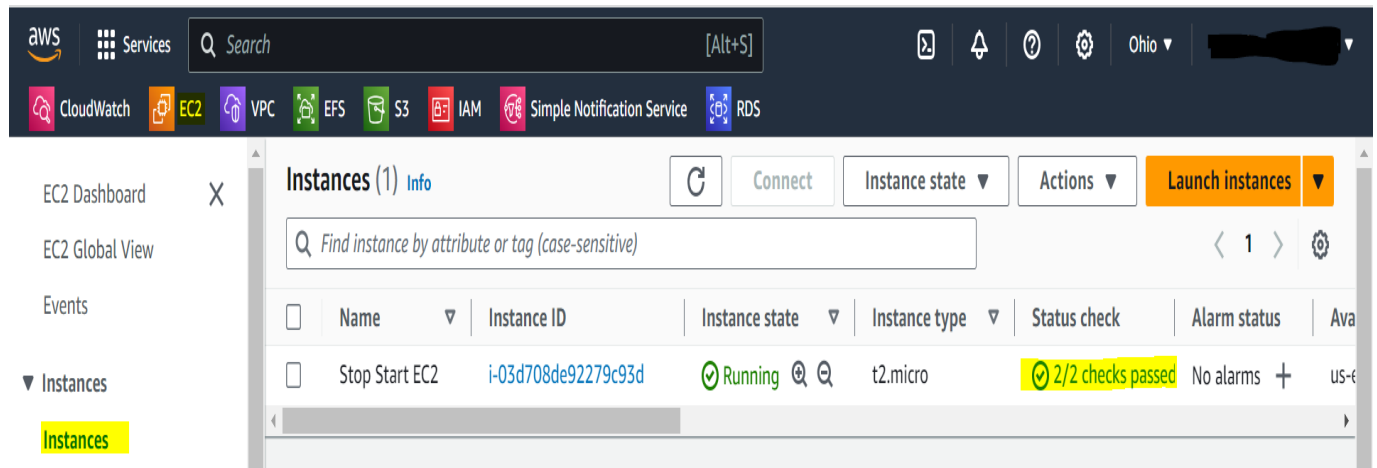
**Start performing Task:** -

### ***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**”.

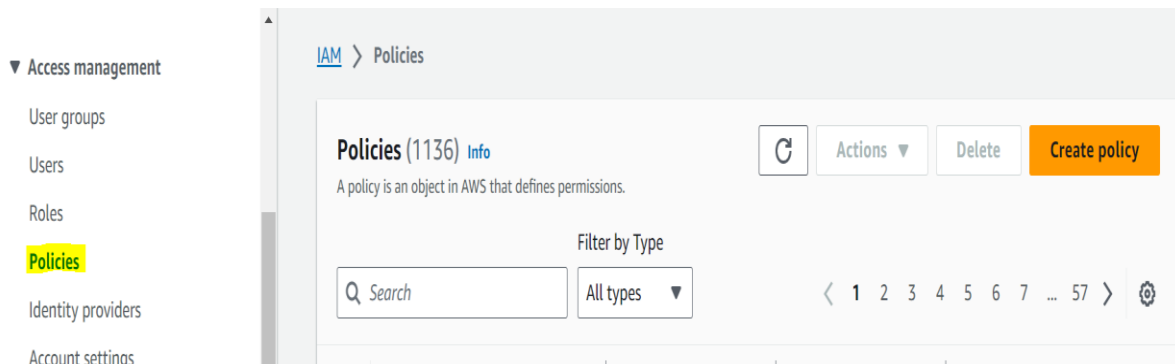


## Step 2: IAM Policy & Creation

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”**.

▼ **Write** (Selected 2/406)

☒ **StopInstances** Info ☐ TerminateClientVpnConnections Info ☐ TerminateInstances Info

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

Resource ARNs

+ Add more permissions

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

Cancel Next

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

**Policy details**

**Policy name**  
Enter a meaningful name to identify this policy.

Stopstartec2instance

Maximum 128 characters. Use alphanumeric and '+=,.\_-' characters.

**Description - optional**  
Add a short explanation for this policy.

stop and start the ec2 instance

Maximum 1,000 characters. Use alphanumeric and '+=,.\_-' characters.

Cancel Previous **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”**.


Access management

- User groups
- Users
- Roles**
- Policies

IAM > Roles

**Roles (9)** Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

 Delete **Create role**

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

**Trusted entity type**

☒ **AWS service**  
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**  
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**  
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**  
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**  
Create a custom trust policy to enable others to perform actions in this account.

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case  

Lambda

Choose a use case for the specified service.  
Use case  
☒ **Lambda**  
Allows Lambda functions to call AWS services on your behalf.

Cancel

Next

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

**add permissions** [Info](#)

**Permissions policies (1/889)** [Info](#)

Choose one or more policies to attach to your new role.

Q Stop X

Filter by Type  
All types ▼

1 match

< 1 > ⚙

<input checked="" type="checkbox"/>	Policy name <a href="#">↗</a>	Type	Description
<input checked="" type="checkbox"/>	Stopstartec2instance	Customer managed	stop and start the ec2 instance

► **Set permissions boundary - optional**

Cancel

Previous

Next

create a name and click on **"Create role"**.

## Name, review, and create

**Role details**

**Role name**  
Enter a meaningful name to identify this role.  
  
Maximum 64 characters. Use alphanumeric and '+=, @-\_' characters.

**Description**  
Add a short explanation for this role.  
  
Maximum 1000 characters. Use alphanumeric and '+=, @-\_' characters.

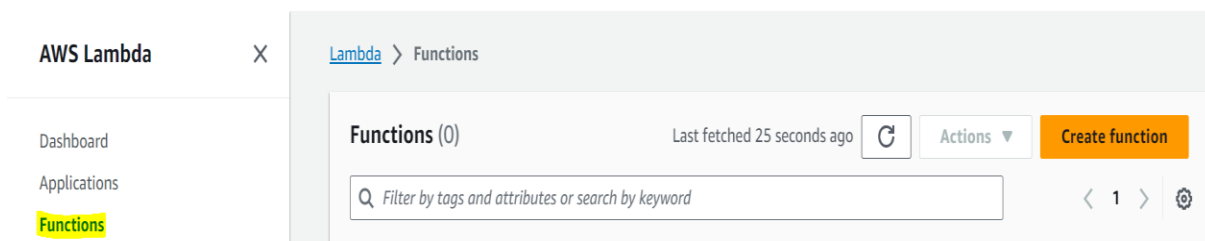
[Cancel](#) [Previous](#) [Create role](#)

### 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

**Create function** Info  
AWS Serverless Application Repository applications have moved to [Create application](#).

☒ **Author from scratch**  
Start with a simple Hello World example.

☐ **Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.

☐ **Container image**  
Select a container image to deploy for your function.

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.  
  
Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** Info  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

▼ **Change default execution role**

**Execution role**  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ **Use an existing role**

☐ Create a new role from AWS policy templates

**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
   
[View the Stopstartec2instancerole role](#) on the IAM console.

► **Advanced settings**

◆ Python code needed to **Stop**

**### Stop the instances by using python code: -**

```
import boto3
```

```
region = 'us-west-1'
```

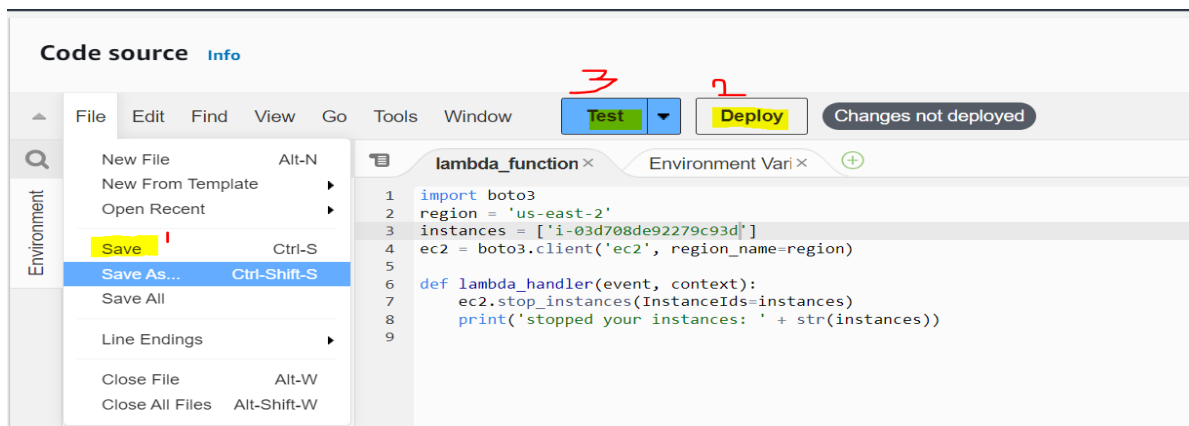
```
instances = ['i-12345cb6de4f78g9h', 'i-o8ce9b2d7eccf6d26']
```

```
ec2 = boto3.client('ec2', region_name=region)
```

```
def lambda_handler(event, context):
```

```
    ec2.stop_instances (InstanceIds=instances)
```

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



**Configure test event** [X]

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event ☐ Edit saved event

Event name

myevent

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

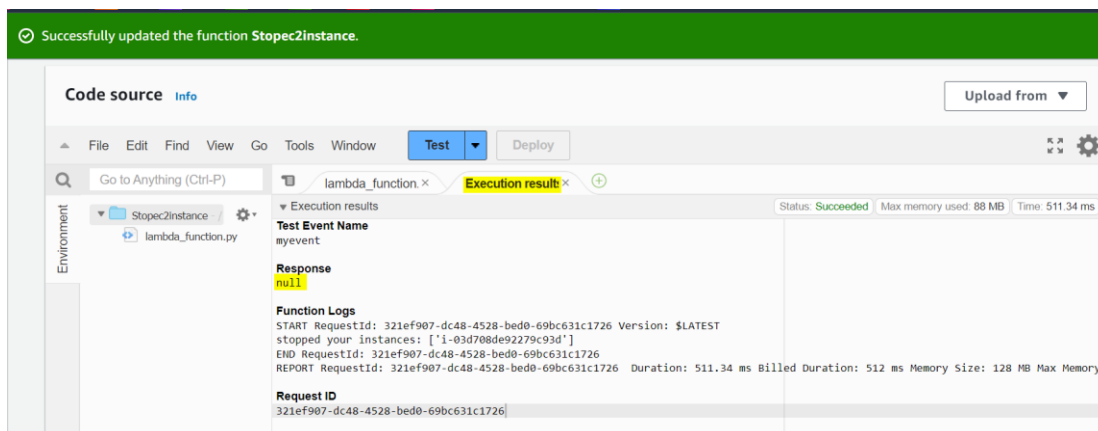
☒ Private  
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable  
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

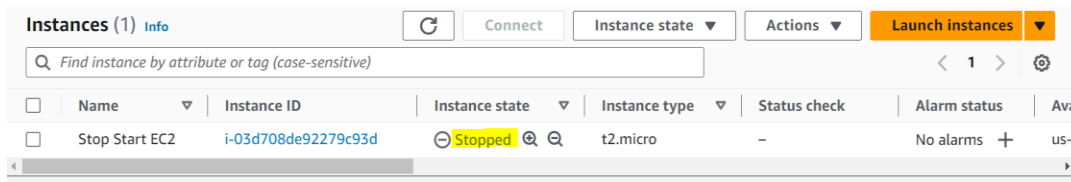
Template - optional

Cancel Invoke Save

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.



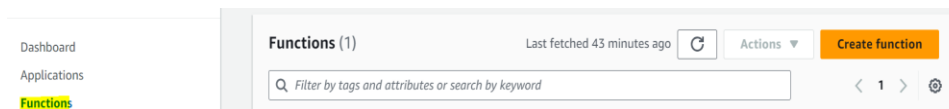
## Status of Ec2 Instance



The screenshot shows the AWS Management Console 'Instances' page. At the top, there are buttons for 'Refresh', 'Connect', 'Instance state', 'Actions', and 'Launch instances'. Below these is a search bar with the placeholder text 'Find instance by attribute or tag (case-sensitive)'. The main table lists the instances. The first instance is 'Stop Start EC2' with ID 'i-03d708de92279c93d', which is in a 'Stopped' state. The instance type is 't2.micro'. The status check shows a minus sign, and there are 'No alarms'.

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	Stop Start EC2	i-03d708de92279c93d	Stopped	t2.micro	-	No alarms	us-east-1a

### ◆ 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



Lambda > Functions > Create function

## Create function [Info](#)

AWS Serverless Application Repository applications have moved to [Create application](#).

☒ **Author from scratch**  
Start with a simple Hello World example.

☐ **Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.

☐ **Container image**  
Select a container image to deploy for your function.

### Basic information

**Function name**  
Enter a name that describes the purpose of your function.

Startec2instance

Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.11

▼ Change default execution role

**Execution role**  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions  
☒ **Use an existing role**  
☐ Create a new role from AWS policy templates

**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

Stopstartec2instanceroles

[View the Stopstartec2instanceroles role](#) on the IAM console.

► **Advanced settings**

Cancel **Create function**

**## Start** our ec2 instances by using python code.

```
import boto3

region = 'us-west-1'

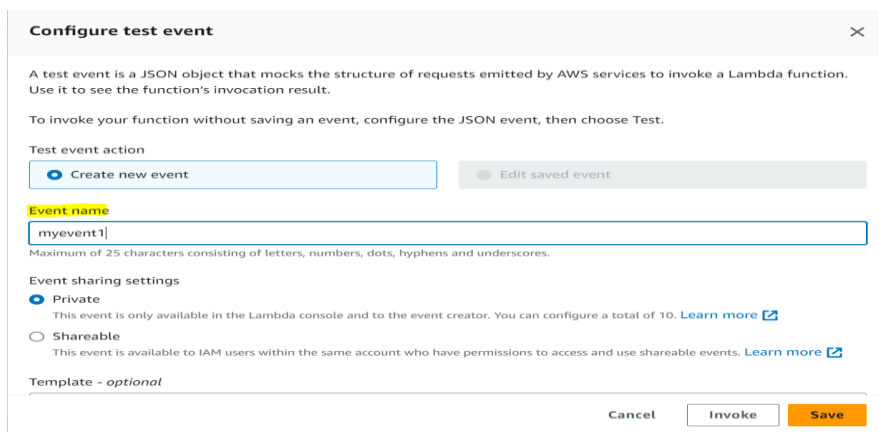
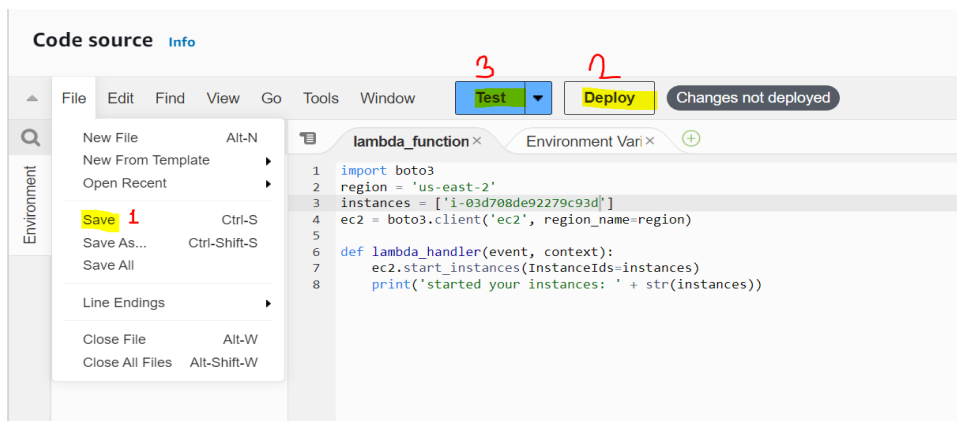
instances = ['i-12345cb6de4f78g9h', 'i-o8ce9b2d7eccf6d26']

ec2 = boto3.client('ec2', region_name=region)

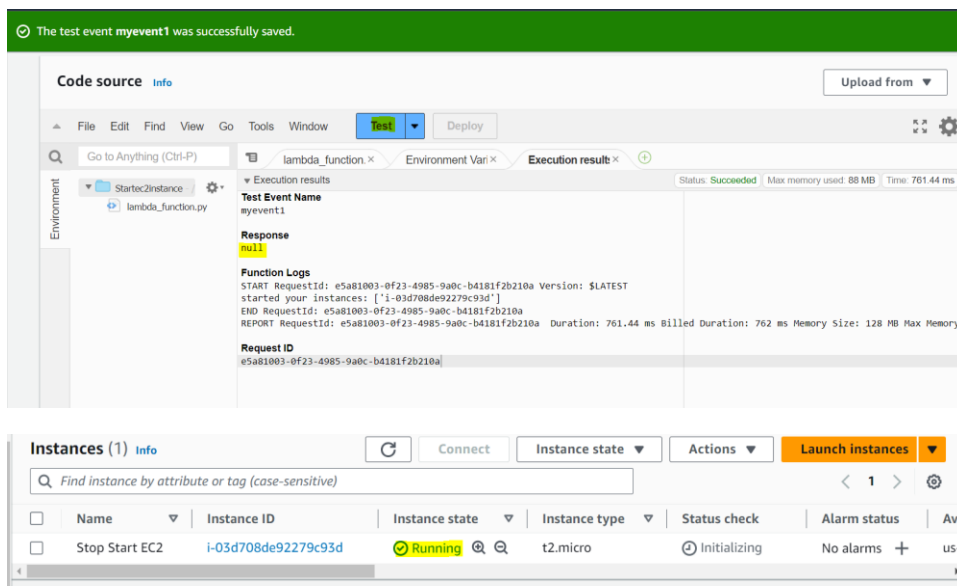
def lambda_handler(event, context):

    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.

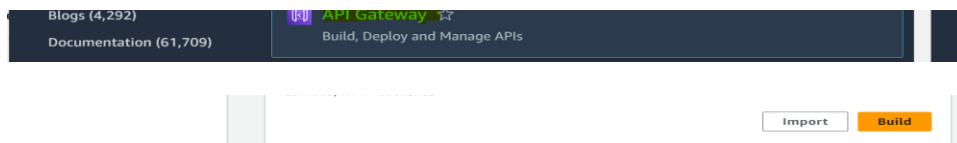


## **Step 4: Create the API Gateway and associate with the lambda function**

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.

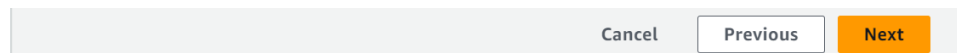
You'll be able to locate your **stop and Start** Lambda function in the dropdown list and fill your **API name** then click on **Next**

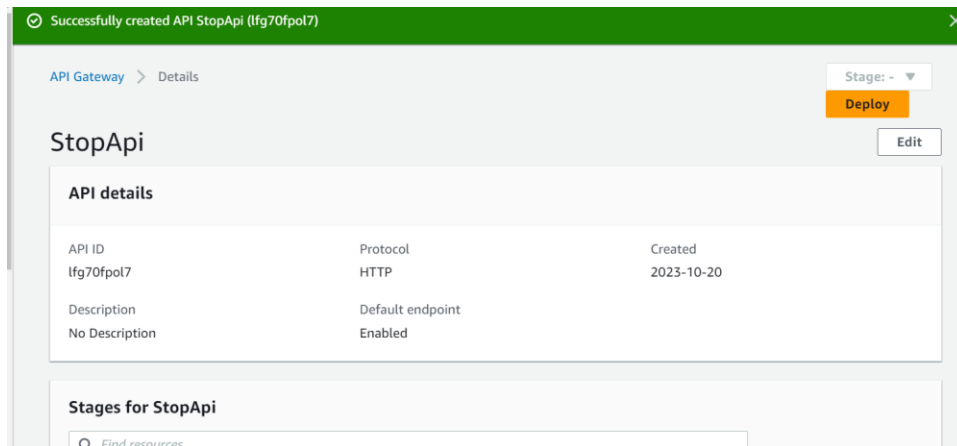


Configure your **API routes**. Then click on **Next**

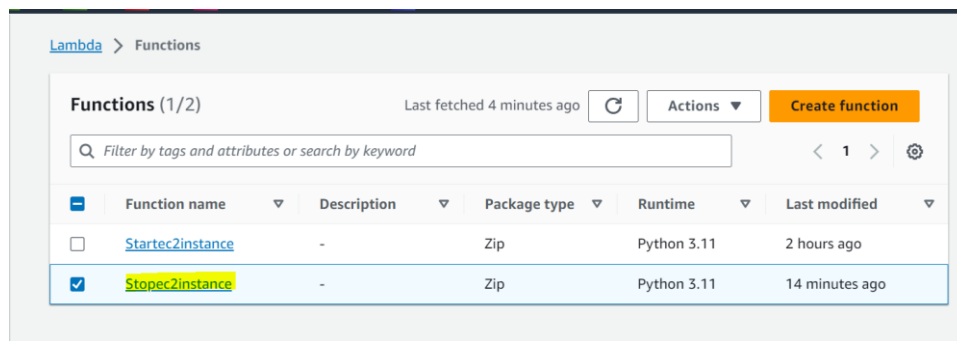


You can have several stages (stage, pre-prod, prod) for a route. Make sure to turn off the Auto-deploy feature. Add your **configure stages** then click on **Next** and then click on **create**

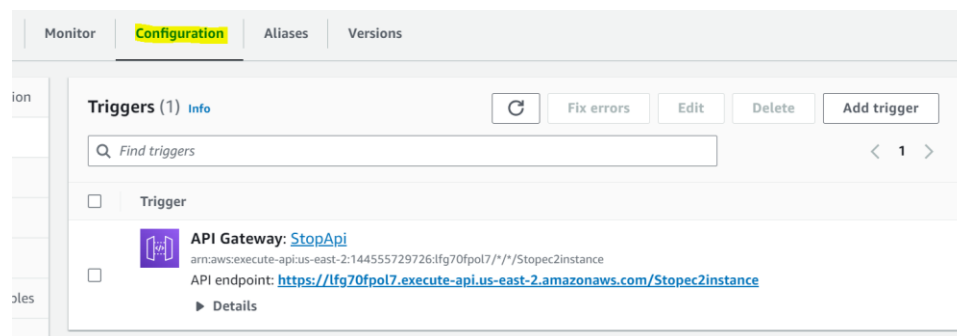




Go back to the **Lambda function** and choose **stop function**



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**

**Triggers (1/1)** [Info](#) Refresh Fix errors Edit Delete Add trigger

< 1 >

☒ **Trigger**

**API Gateway: StopApi**  
 arn:aws:execute-api:us-east-2:144555729726:lfq70fpol7/\*/\*/Stopec2instance  
 API endpoint: <https://lfq70fpol7.execute-api.us-east-2.amazonaws.com/Stopec2instance>  
 ▶ Details

## Add trigger

**Trigger configuration** [Info](#)

**API Gateway**  
 aws api application-services backend HTTP REST serverless

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. [Learn more](#)

**Intent**  
 Use an existing API or have us create one for you.

Selected API: StopApi

**HTTP APIs**

StopApi  
 lfq70fpol7

×

**Existing API**  
 Attach an existing API.  
 ×

**Deployment stage**  
 The name of your API's deployment stage.  
 Refresh

**Security**  
 Configure the security mechanism for your API endpoint.

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel Add

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

**Triggers (1)** [Info](#) Refresh Fix errors Edit Delete Add trigger

< 1 >

☐ **Trigger**

**API Gateway: StopApi**  
 arn:aws:execute-api:us-east-2:144555729726:lfq70fpol7/\*/\*/Stopec2instance  
 API endpoint: <https://lfq70fpol7.execute-api.us-east-2.amazonaws.com/Stopec2instance>  
 ▶ Details

Instances (1) <a href="#">info</a>							
<input type="text" value="Find instance by attribute or tag (case-sensitive)"/> <span>&lt; 1 &gt;</span>							
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Av
<input type="checkbox"/>	Stop Start EC2	i-03d708de92279c93d	<span>⏸ Stopping</span>	t2.micro	-	No alarms +	us-

## ♦ Start EC2 Instance by triggering API Endpoint

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

APIs (1)							
<input type="text" value="Find APIs"/> <span>&lt; 1 &gt;</span>							
<input type="radio"/>	Name	Description	ID	Protocol	Endpoint type	Created	
<input type="radio"/>	StopApi		lfg70fpol7	HTTP	Regional	2023-10-20	

Choose API Type **Click on Build**

### Choose an API type

#### HTTP API

Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.

Works with the following:  
Lambda, HTTP backends

## Integrations

Lambda ▼ Remove

AWS Region

Lambda function

Version [Learn more.](#)

us-east-2 ▼

Q arn:aws:lambda:us-east-2:144555729726:fu X

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



Remove

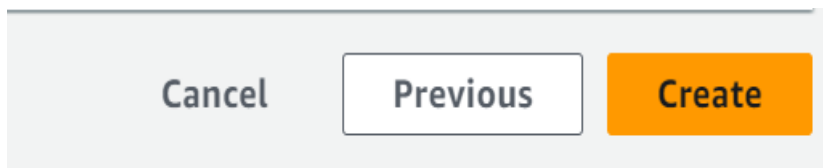
Add stage

Cancel

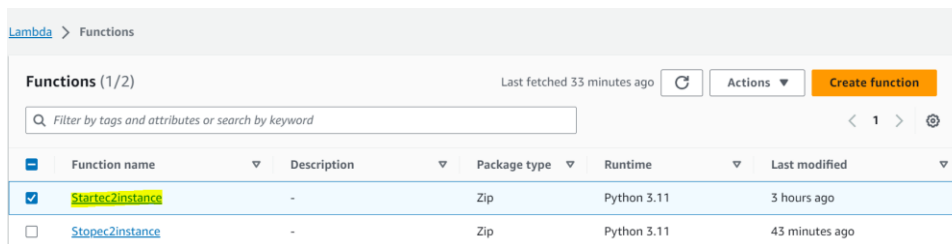
Previous

Next

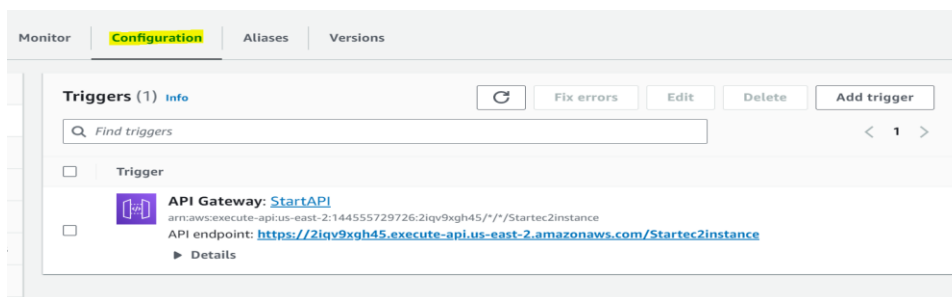
Then Click on **Create**



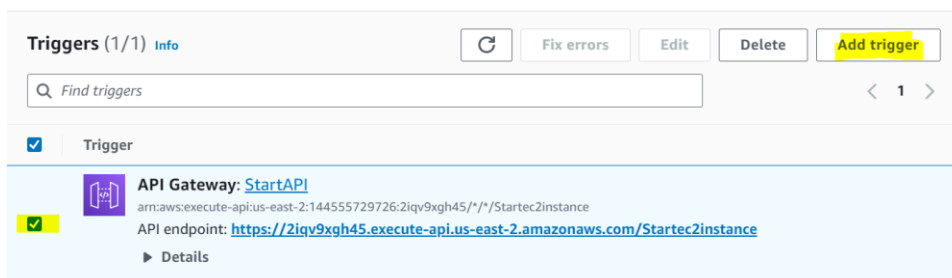
Go back to the **Lambda function** and choose **Start function**



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**





**API Gateway**  
aws api application-services backend HTTP REST serverless

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. [Learn more](#)

Selected API: StartAPI

**HTTP APIs**

StartAPI  
2iqv9xgh45

StopApi  
lfg70fpol7

Q 2iqv9xgh45

Use an existing api or have us create one for you.

☐ Create a new API

☒ Use existing API

**Existing API**  
Attach an existing API.

Q 2iqv9xgh45

**Deployment stage**  
The name of your API's deployment stage.

\$default

**Security**  
Configure the security mechanism for your API endpoint.

Open

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. [Learn more](#)

Cancel Add

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

**Triggers (1)** Info

Find triggers

Trigger

API Gateway: [StartAPI](#)

arn:aws:execute-api:us-east-2:144555729726:2iqv9xgh45/\*/\*/Startec2instance

API endpoint: <https://2iqv9xgh45.execute-api.us-east-2.amazonaws.com/Startec2instance>

Details

**Instances (1)** Info

Find instance by attribute or tag (case-sensitive)

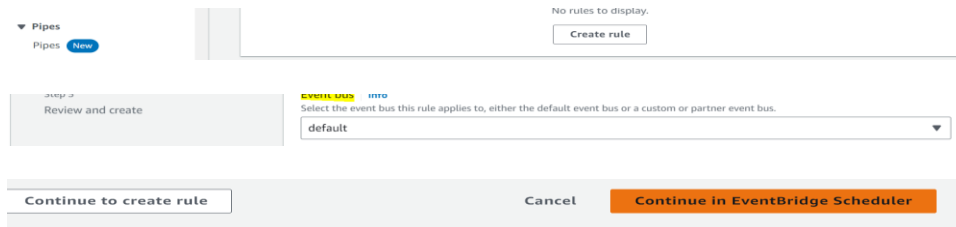
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Stop Start EC2	i-03d708de92279c93d	Running	t2.micro	2/2 checks passed	No alarms	us-east-2

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

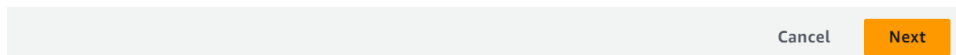
In this step, we are going to create an **EventBridge** Rule that triggers our EC2 instances. Being that Lambda is triggered by events, whenever an EC2 instance is stopped, it's going to send an event that will trigger our function.

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”**.

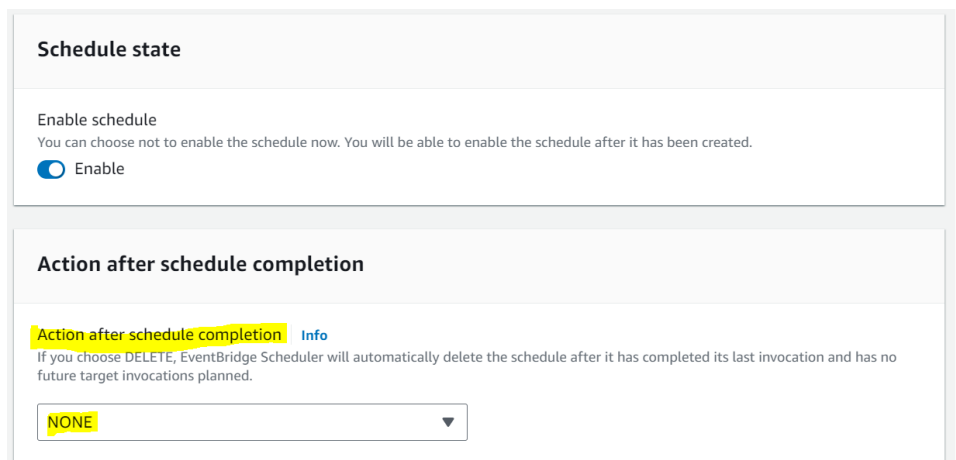
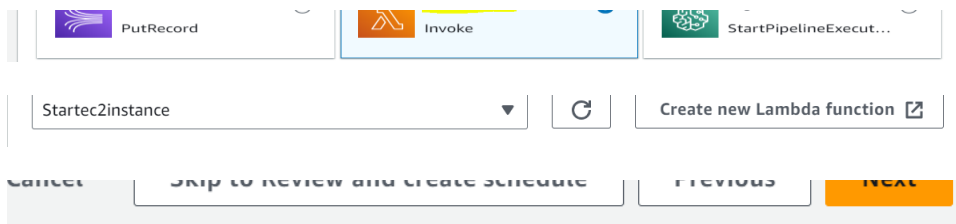
Provide a name and a description and under rule type select **“rule with an event pattern”**.



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



For event source select **AWS Services**.



**Permissions** [Info](#)

**Permissions**  
EventBridge Scheduler requires permission to send events to the target, and based on the preferences you select, integrate with other AWS services such as AWS KMS and Amazon SQS.

**Execution role**

☒ **Create new role for this schedule** ☐ Use existing role

**Role name**  
This is the role name we will be creating on your behalf. You can change the name.

[Go to IAM console](#)

[Cancel](#) [Previous](#) [Next](#)

[Cancel](#) [Previous](#) [Create schedule](#)

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

Instances (1) <a href="#">Info</a>							
<input type="text" value="Find instance by attribute or tag (case-sensitive)"/>							
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability zone
<input type="checkbox"/>	Stop Start EC2	i-03d708de92279c93d	<span>Running</span>	t2.micro	Initializing	No alarms	us-east-1

## 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***”.

The screenshot shows the 'Select event bus' dialog in the Amazon EventBridge console. On the left is a sidebar with navigation links: 'Developer resources' (Learn, Sandbox, Quick starts), 'Buses' (Event buses, **Rules**, Global endpoints, Archives, Replays), and 'Pipes' (Pipes). The main area is titled 'Select event bus'. It has a section 'Event bus' with a dropdown menu currently set to 'default'. Below this is a 'Rules (0)' section with a search bar, a status filter set to 'Any status', and a table with columns 'Name', 'Status', 'Type', and 'ARN'. The table is empty, showing 'No rules' and 'No rules to display'. There are buttons for 'Delete', 'Enable', 'Edit', 'CloudFormation Template', and a prominent orange 'Create rule' button.

The screenshot shows the 'Rule detail' form in the Amazon EventBridge console. It contains the following fields and options:

- Name:** A text input field containing 'StopEc2Rule'. Below it, a note states: 'Maximum of 64 characters consisting of numbers, lower/upper case letters, -, \_, .'.
- Description - optional:** A text input field containing 'StopEc2'.
- Event bus:** A dropdown menu set to 'default'. Below it, a note states: 'Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.'
- Enable the rule on the selected event bus:** A toggle switch that is currently turned on.
- Rule type:** Two radio button options:
  - Rule with an event pattern:** A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target. (This option is currently selected.)
  - Schedule:** A rule that runs on a schedule.

The screenshot shows the bottom of the Amazon EventBridge console interface. It features two buttons: a grey 'Cancel' button on the left and a prominent orange 'Continue in EventBridge Scheduler' button on the right.

## Schedule pattern

### Occurrence [Info](#)

You can define an one-time or recurring 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



© Shipping Tool

**Permissions** [Info](#)

Permissions

EventBridge Scheduler requires permission to send events to the target, and based on the preferences you select, integrate with other AWS services such as AWS KMS and Amazon SQS.

**Execution role**

☒ **Create new role for this schedule**

☐ Use existing role

Role name

This is the role name we will be creating on your behalf. You can change the name.

Amazon\_EventBridge\_Scheduler\_LAMBDA\_5392254861

Go to IAM console [↗](#)

Cancel

Previous

Next

Cancel

Previous

Create schedule

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

Instances (1) <a href="#">Info</a>							
<div><div>Find instance by attribute or tag (case-sensitive)</div><div><div>Refresh</div><div>Connect</div><div>Instance state ▼</div><div>Actions ▼</div><div>Launch Instances ▼</div></div></div>							
<div>&lt; 1 &gt; </div>							
<input type="checkbox"/>	Name ▼	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status	Availability zone
<input type="checkbox"/>	Stop Start EC2	i-03d708de92279c93d	Stopped	t2.micro	-	No alarms	us-east-1