

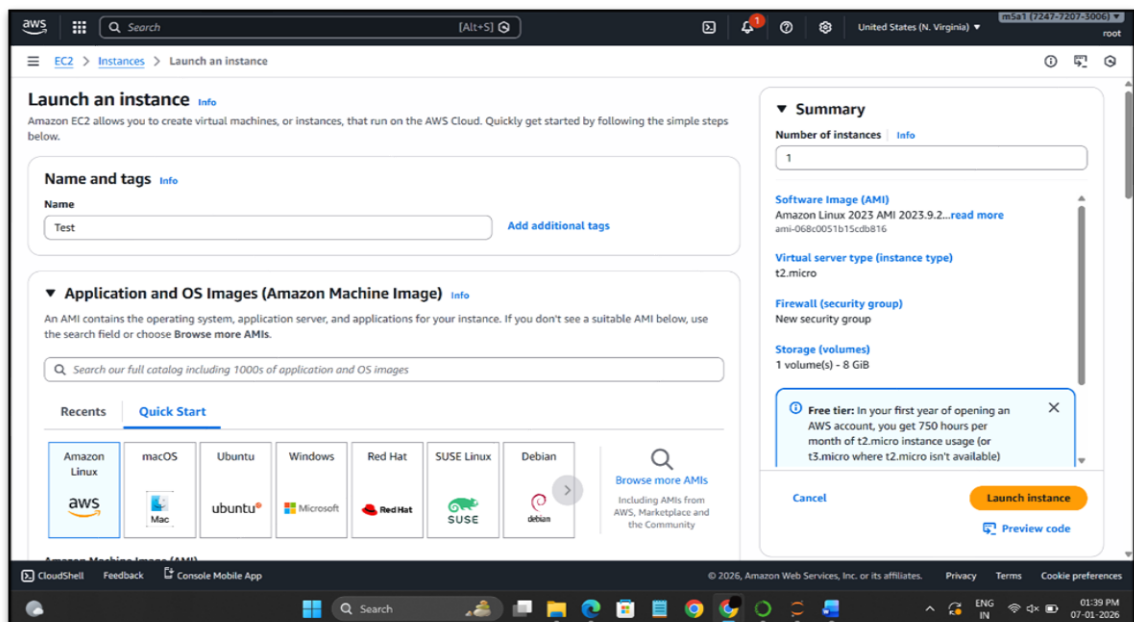
# To Stop and Start Amazon EC2 Instances at regular time intervals using Lambda.

## Requirements:

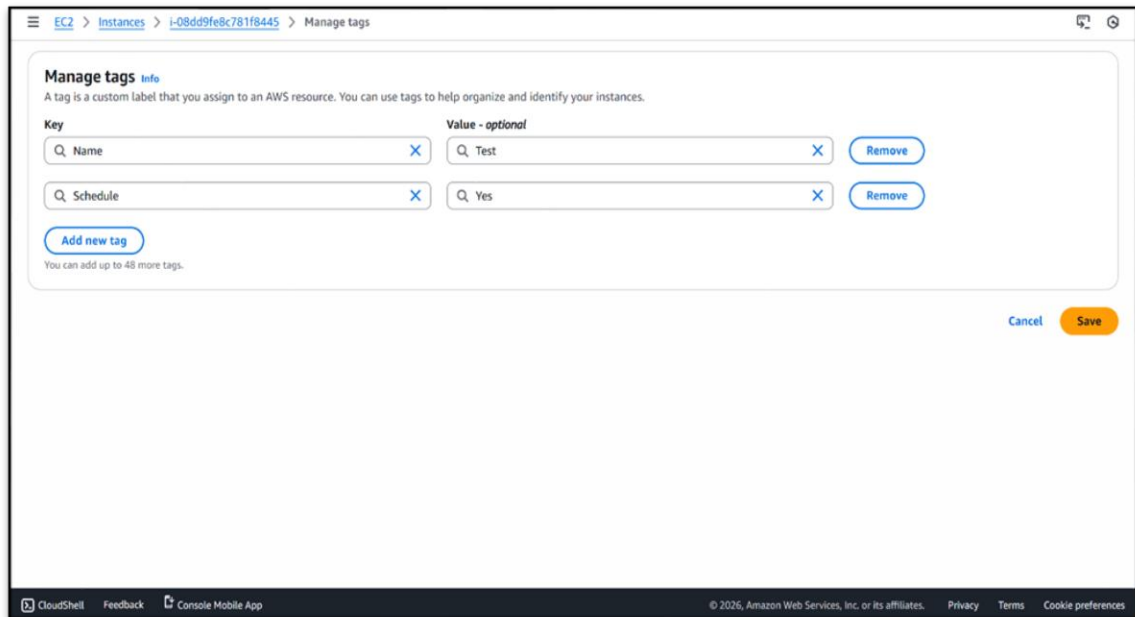
- Reduce EC2 Usage by scheduling start and stop automatically.
- Create IAM policy and execution role for Lambda function
- Create Lambda function to start and stop EC2 instances.
- Create EventBridge to rule Lambda function.

## EC2:

- Open the AWS console → Navigate to EC2 → Launch instance → Name → Key pair → Allow SSH and HTTP in security group → Launch instance.



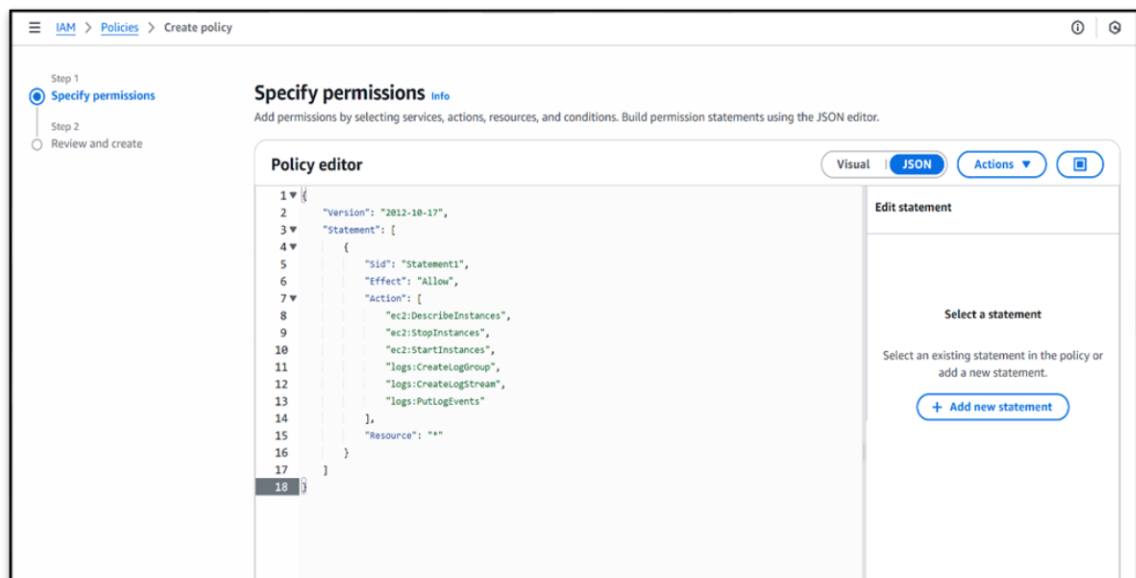
- After creation of instance select the instance → Navigate to tags → Manage tags → Add tags → Key='Schedule', Value='Yes' → Save. (This is done to help Lambda identify the instance)



## IAM:

➤ Navigate to IAM policies → Create policy → JSON → Add the required permissions:

- EC2
- Logs



➤ Review and create: Policy name → "EC2LambdaPolicy" → Create policy.

Permissions defined in this policy [info](#) [Edit](#)

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Search

Allow (2 of 461 services) [Show remaining 459 services](#)

Service	Access level	Resource	Request condition
<a href="#">CloudWatch Logs</a>	Limited: Write	All resources	None
<a href="#">EC2</a>	Limited: List, Write	All resources	None

**Add tags - optional** [info](#)

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

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

- Navigate to Roles→Create Execution role→Select Lambda as service→Attach EC2 Lambda Policy→Next→Name:"Lambda EC2Role"→Create role.

Step 2: Add permissions [Edit](#)

**Permissions policy summary**

Policy name	Type	Attached as
<a href="#">EC2LambdaPolicy</a>	Customer managed	Permissions policy

**Step 3: Add tags**

**Add tags - optional** [info](#)

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

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

## Lambda:

- Navigate to Lambda Functions→Create function→Function name→Runtime(Python 3.12)→Use the existing Lambda EC2 Role→Create function.

**Permissions** [info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **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.

LambdaEC2Role [View the LambdaEC2Role role on the IAM console.](#)

► **Additional configurations**

Use additional configurations to set up networking, security, and governance for your function. These settings help secure and customize your Lambda function deployment.

[Cancel](#) [Create function](#)

- Provide necessary code for the **Lambda Stop Function**→ Deploy.

**Code source** [info](#)

[Open in Visual Studio Code](#) [Upload from](#)

EXPLORE

- STOPEC2FUNCTION
  - lambda\_function.py

DEPLOY [Deploy \(Ctrl+Shift+U\)](#) [Test \(Ctrl+Shift+I\)](#)

TEST EVENTS (SELECTED: TESTSTOP)

- + Create new test event
- Private saved events
  - TestStop

```

1 import json
2
3 import boto3
4
5 def lambda_handler(event, context):
6     ec2 = boto3.client('ec2', region_name='us-east-1')
7     response = ec2.describe_instances(Filters=[{'Name': 'tag:Schedule', 'Values': ['Yes']}])
8     instance_ids = []
9     for reservation in response['Reservations']:
10         for instance in reservation['Instances']:
11             if instance['State']['Name'] == 'running':
12                 instance_ids.append(instance['InstanceId'])
13
14 if instance_ids:
15     ec2.stop_instances(InstanceIds=instance_ids)
16     return f'Stopped: {instance_ids}'
17
18 return 'No running instances tagged'
  
```

PROBLEMS OUTPUT CODE REFERENCE LOG TERMINAL

Tasks

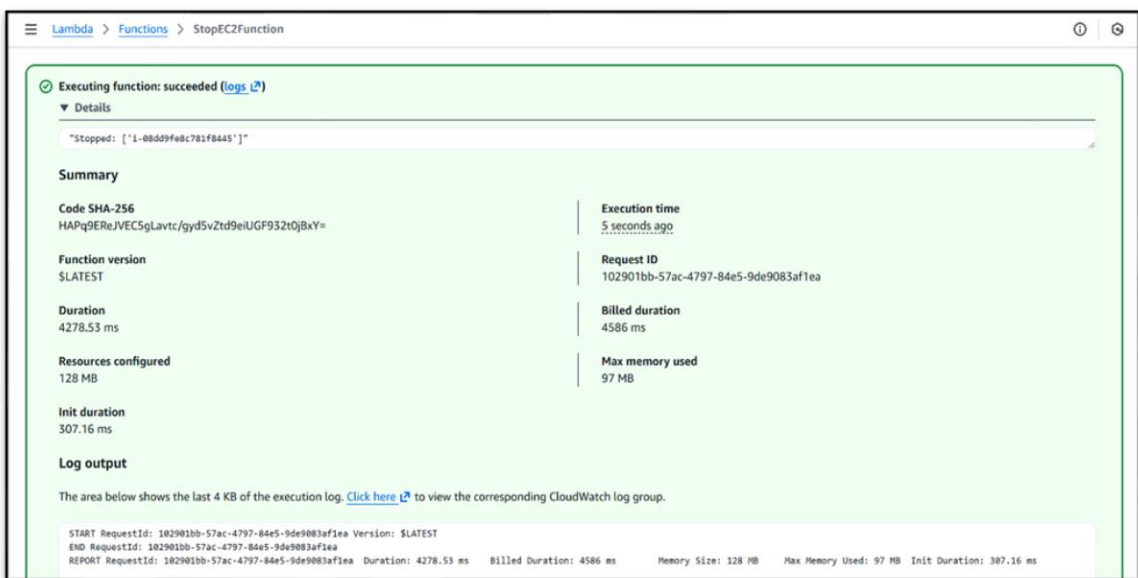
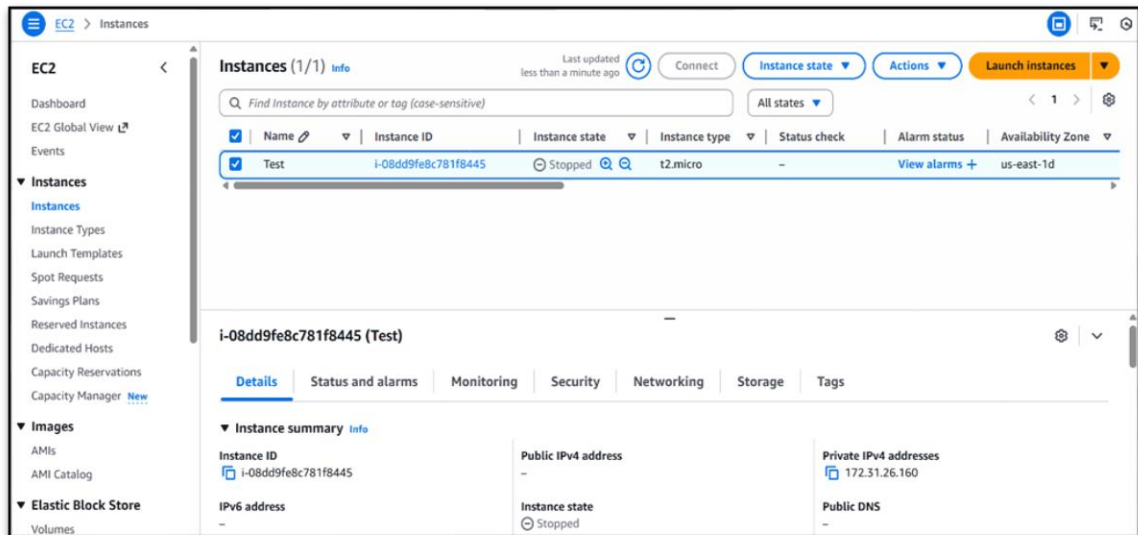
- Configure the code by navigating to configuration→Enter the timeout seconds→Save.

The screenshot shows the 'Edit basic settings' page for a Lambda function named 'StopEC2Function'. The breadcrumb navigation at the top is 'Lambda > Functions > StopEC2Function > Edit basic settings'. The page contains several configuration sections: 'Ephemeral storage' with a value of 512 MB; 'SnapStart' set to 'None'; 'Timeout' set to 0 minutes and 5 seconds; 'Execution role' with the option 'Use an existing role' selected; and 'Existing role' set to 'LambdaEC2Role'. At the bottom right, there are 'Cancel' and 'Save' buttons.

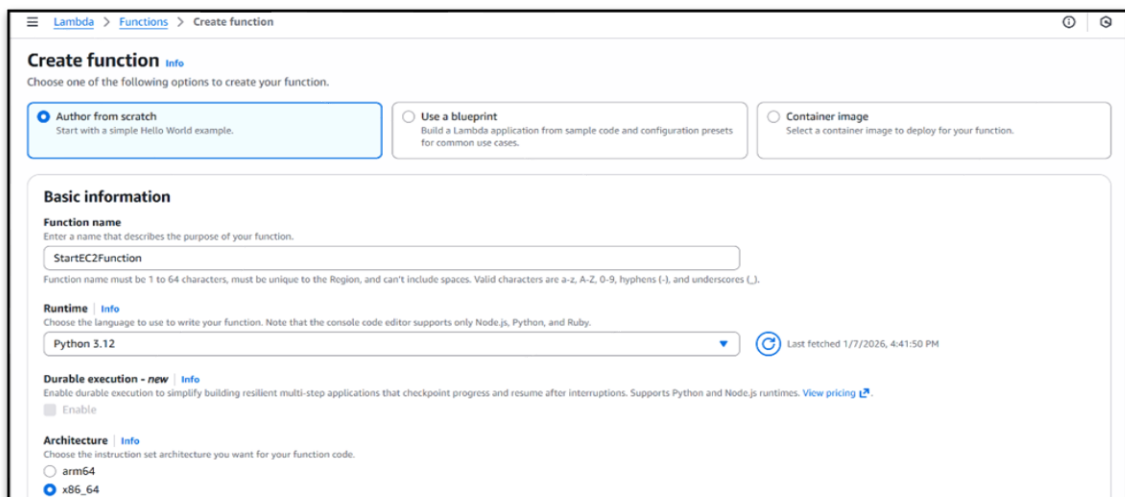
- Test the Lambda function by navigating to Test→Proceed with empty JSON code→Save→Test.

This screenshot is identical to the one above, showing the 'Edit basic settings' page for the 'StopEC2Function' Lambda function. It displays the same configuration options: 512 MB ephemeral storage, SnapStart set to None, a 5-second timeout, and the 'LambdaEC2Role' as the execution role. The 'Save' button is highlighted in orange.

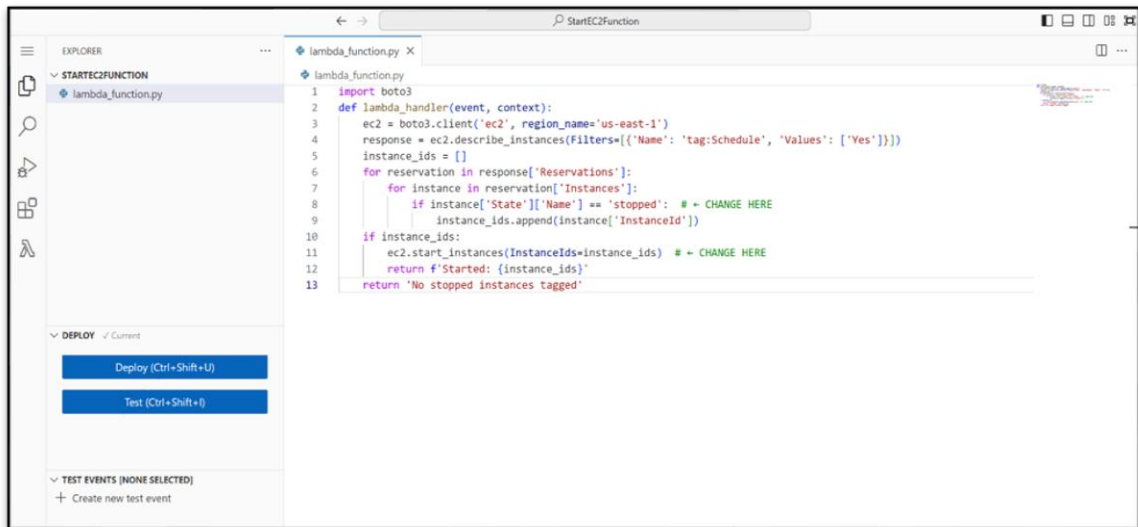
- You can find that the instance has been stopped and the stop function has been executed successfully.



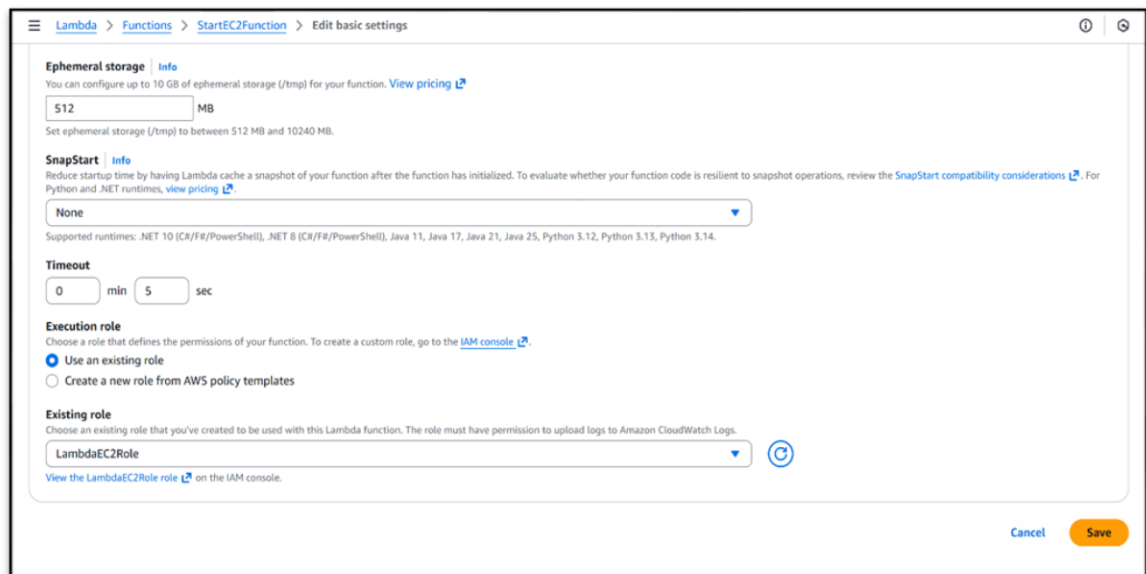
- Create Start Function by navigating to Lambda → Create function → Name: "StartEC2Function" → Provide the existing Lambda EC2 Role → Create function.



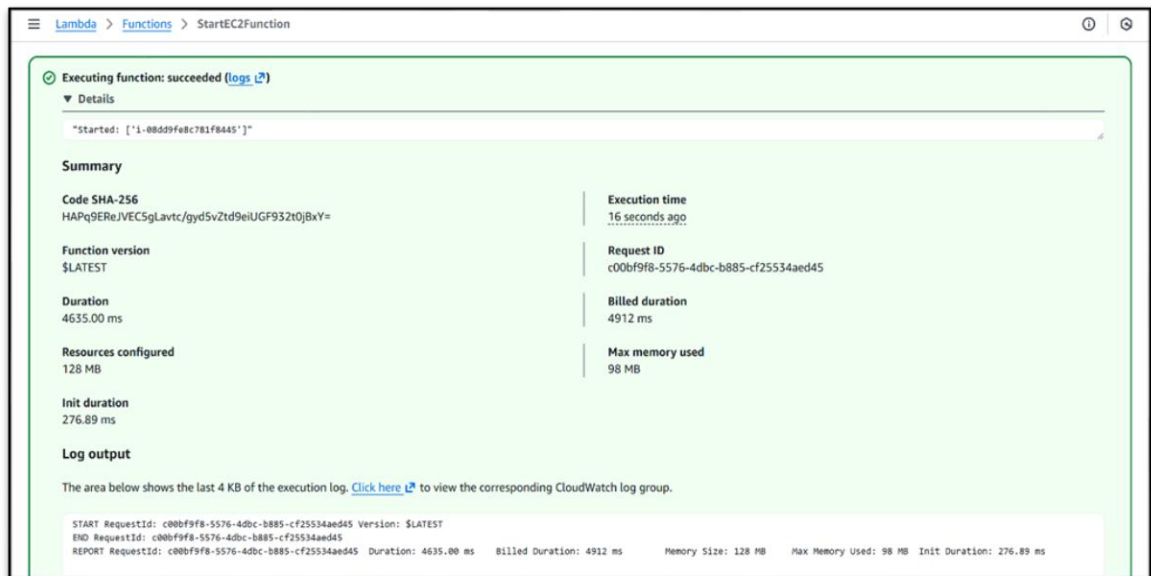
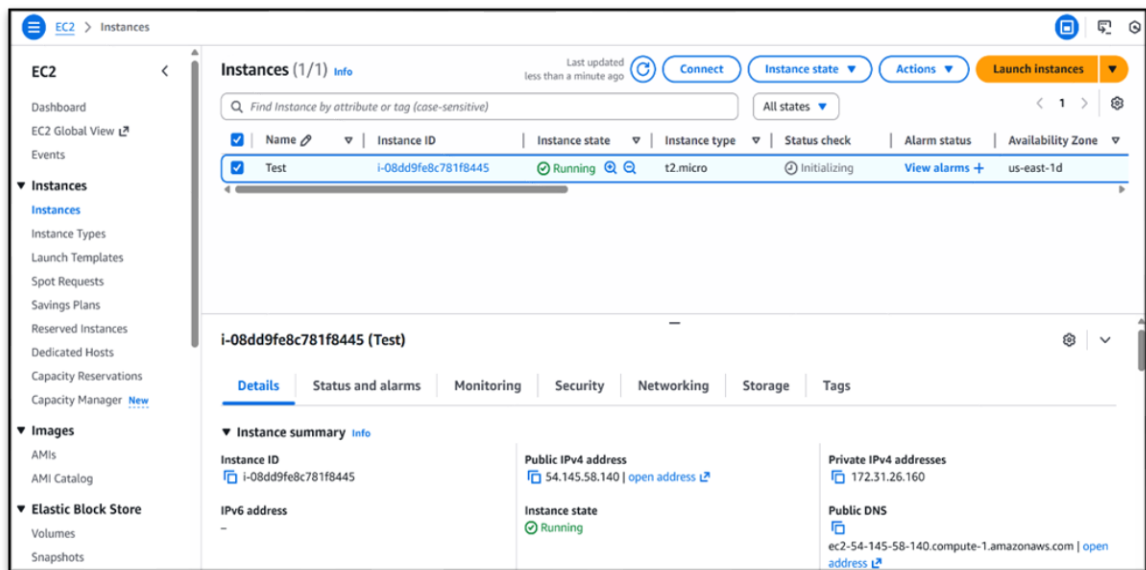
- Provide necessary code for the **Lambda Start Function**→  
Deploy.



- Configure the code by navigating to configuration→Enter the timeout seconds→Save.



- Test the Lambda function by navigating to Test→Proceed with empty JSON code→Save→Test.
- You can find that the instance has been started and the start function has been executed successfully.



## EventBridge:

- Navigate to Amazon Eventbridge Console→Rules→Create rule  
→Provide the name and description→Next.
- Define schedule by entering appropriate cron expression→Next.



The screenshot shows the 'Define schedule' step of the 'Create rule' wizard in Amazon EventBridge. The left sidebar contains navigation links for Dashboard, Developer resources, Buses, Pipes, Scheduler, and Integration. The main content area is titled 'Define schedule' and includes a 'Schedule pattern' section with two radio buttons: 'A fine-grained schedule that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.' (selected) and 'A schedule that runs at a regular rate, such as every 10 minutes.' Below this is a 'Cron expression' section with a text input field containing '0 12 \* \* ? \*' and a dropdown menu set to 'UTC'. A list of 'Next 10 trigger date(s)' is displayed, ranging from Wednesday, 07 Jan 2026 to Friday, 16 Jan 2026. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.

- Select the target type as Lambda Function → Stop EC2 Function → Next → Review and Create.

The screenshot shows the 'Review and create' step of the 'Create rule' wizard. The left sidebar is the same as the previous screenshot. The main content area is titled 'Review and create' and contains two sections: 'Step 1: Define rule detail' and 'Step 2: Build schedule'. 'Step 1' shows the rule name 'TestStop6p', description 'Stop at 6pmIST', status 'Enabled', and rule type 'Scheduled rule'. 'Step 2' shows the cron expression '0 12 \* \* ? \*' and the same 'Next 10 trigger date(s)' list. At the bottom right are 'Edit' buttons for both steps.

- Follow the same steps and create another rule for Start EC2 Function.

Amazon EventBridge > Rules > Create rule

**Review and create**

**Step 1: Define rule detail** [Edit](#)

**Define rule detail**

<b>Rule name</b> TestStart7pm	<b>Status</b> Enabled	<b>Event bus</b> default
<b>Description</b> Start at 7pmIST	<b>Rule type</b> Scheduled rule	

**Step 2: Build schedule** [Edit](#)

**Event schedule** [Info](#)

Cron expression  
0 13 \* \* \*

Next 10 trigger date(s) UTC

Wed, 07 Jan 2026 13:00:00 UTC  
Thu, 08 Jan 2026 13:00:00 UTC  
Fri, 09 Jan 2026 13:00:00 UTC  
Sat, 10 Jan 2026 13:00:00 UTC  
Sun, 11 Jan 2026 13:00:00 UTC  
Mon, 12 Jan 2026 13:00:00 UTC

- Both the Start and Stop rule have been successfully created to schedule the EC2.

Amazon EventBridge > Rules

**Rules**

A rule watches for specific types of events. When a matching event occurs, the event is routed to the targets associated with the rule. A rule can be associated with one or more targets.

**Select event bus**

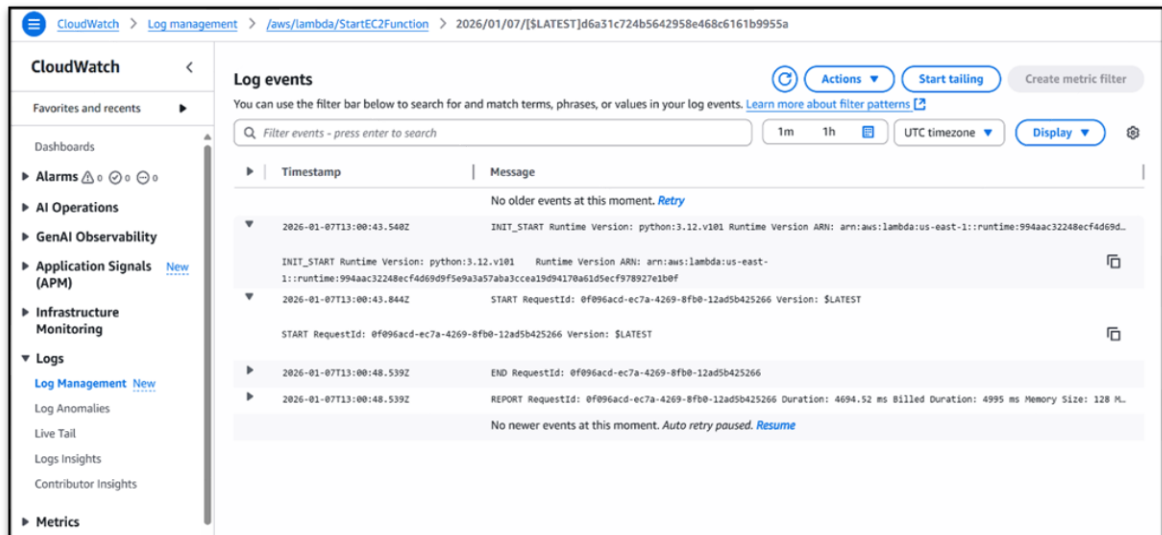
**Event bus**  
Select or enter event bus name  
default

**Rules on default event bus (2)** [Refresh](#) [Delete](#) [Enable](#) [Edit](#) [CloudFormation Template](#) [Create rule](#)

Any status

<input type="checkbox"/>	Name	Status	Type	Event bus	ARN	Description
<input type="checkbox"/>	<a href="#">TestStart7pm</a>	Enabled	Scheduled Standard	default	arn:aws:events:us-east-1:72477207:3006:rule/TestStart7pm	Start at 7pmIST
<input type="checkbox"/>	<a href="#">TestStop6pm</a>	Enabled	Scheduled Standard	default	arn:aws:events:us-east-1:72477207:3006:rule/TestStop6pm	Stop at 6pmIST

- Through the log events we could find that instance have stopped and started at the specified time intervals.



- The EC2 instances have been scheduled to stop and start at the specified time using IAM policy, Lambda functions and Eventbridge.

## Conclusion:

- This AWS EC2 auto stop/start project successfully implemented serverless automation using Lambda and EventBridge, achieving 75% cost savings by stopping instances during specified times.