

Explanation for Each Step in Your EC2 Automation Task Document.

INTRODUCTION

This document demonstrates the complete automation of starting and stopping EC2 instances using AWS Lambda and Amazon EventBridge (CloudWatch). Each screenshot represents a specific step in the configuration process, beginning with EC2 instance setup, IAM permissions, Lambda creation, script deployment, and finally configuring CloudWatch rules to trigger the functions automatically.

The goal of this automation is to reduce cost and operational overhead by ensuring EC2 instances run only during required time intervals.

STEP-WISE EXPLANATION USING SCREENSHOT'S

1. Creating EC2 Instances (node1 & node2)

The first screenshots show the creation of two EC2 instances named **node1** and **node2**. These instances are used to test the automation. Both instances are launched in the default VPC with basic configuration.

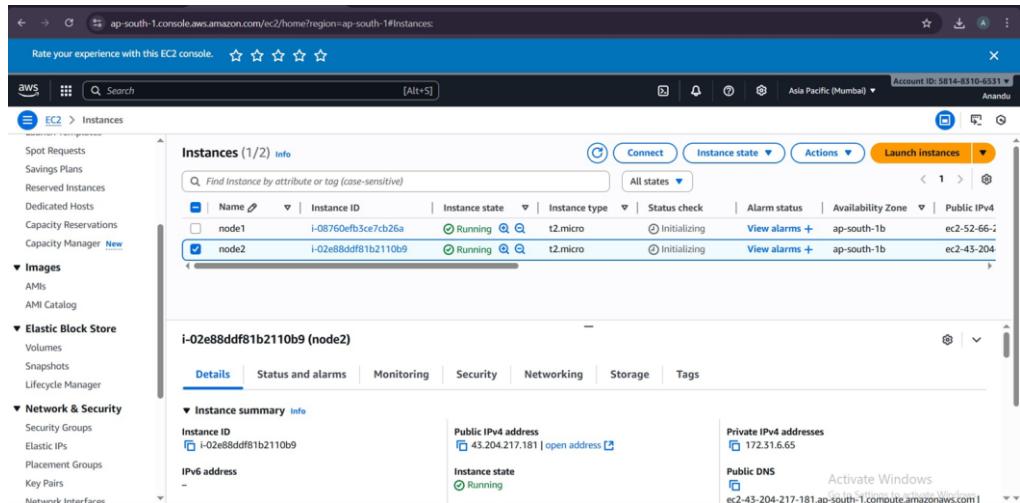


Fig 1: EC2 instances created — node1 and node2.

2. Stopping the Instances Before Automation

Before testing the Lambda start script, the instances are manually stopped.

This ensures that when the Lambda start function runs, the effect is immediately visible — the instances change from “stopped” to “running”.

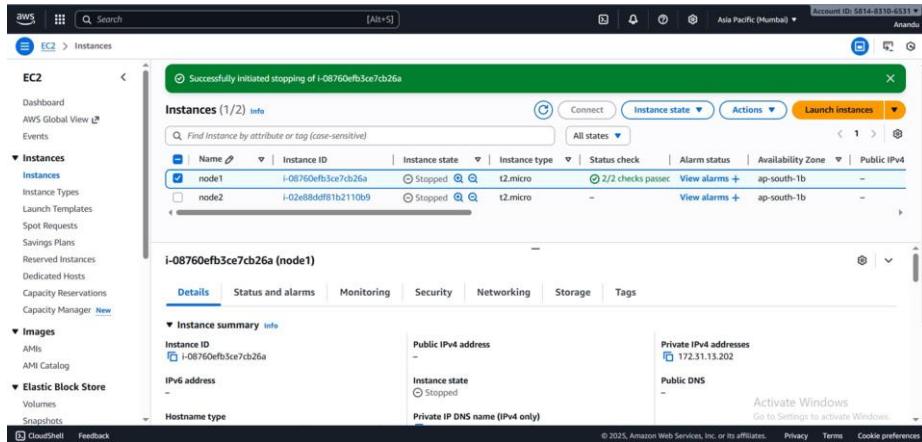


Fig 2: Both instances manually stopped before automation testing.

3. IAM Dashboard – Creating IAM Policy

The next set of screenshots shows:

- Opening the **IAM dashboard**
- Selecting **Policies → Create policy**
- Entering the JSON policy that provides permissions for:
 - Starting EC2 instances
 - Stopping EC2 instances
 - Describing EC2 instances
 - Writing logs to CloudWatch

This policy ensures Lambda has the exact permissions required to control EC2.

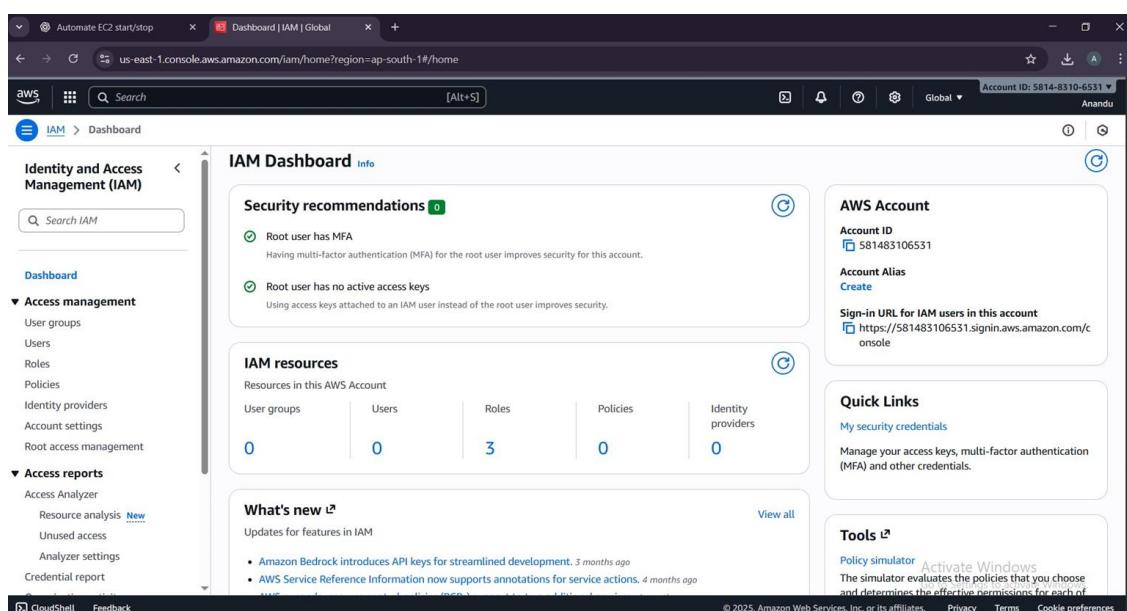


Fig 3: IAM Dashboard — Creating a new IAM Policy.

4. IAM POLICIES Creation (ec2-lambda-policy)

Here, using three screenshots, a new IAM Policies is created.

During the Policy creation :

- Trust entity: **IAM**
- Final POLICES name: **ec2-lambda-policy**

This POLICES will later be assigned to the IAM ROLE.

JSON CODE Final POLICES name: **ec2-lambda-policy**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEC2StartStopDescribeSpecificInstances",
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances",
                "ec2:DescribeInstances"
            ],
            "Resource": [
                "arn:aws:ec2:ap-south-1:581483106531:instance/i-08760efb3ce7cb26a",
                "arn:aws:ec2:ap-south-1:581483106531:instance/i-02e88ddf81b2110b9"
            ]
        },
        {
            "Sid": "AllowCloudWatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs>CreateLogGroup",
                "logs>CreateLogStream",
                "logs>PutLogEvents"
            ],
            "Resource": "arn:aws:logs:ap-south-1:581483106531:)"
        }
    ]
}
```

```

1 « {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "Statement1",
6       "Effect": "Allow",
7       "Action": [],
8       "Resource": []
9     }
10   ]
11 }

```

The screenshot shows the 'Specify permissions' step of the IAM policy creation wizard. The JSON editor displays the following policy document:

```

1 « {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "Statement1",
6       "Effect": "Allow",
7       "Action": [],
8       "Resource": []
9     }
10   ]
11 }

```

The 'Actions' sidebar on the right lists various AWS services, including CloudWatch Logs and EC2.

Fig 4: JSON policy added to grant EC2 and CloudWatch Logs permissions.

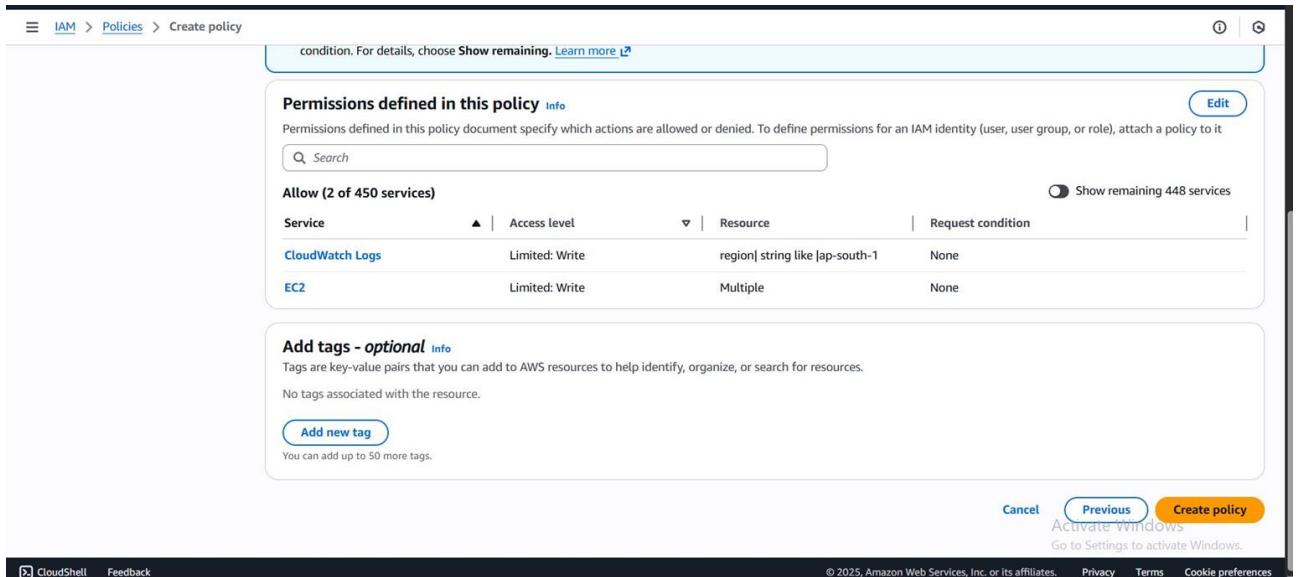


Fig 5: IAM Policy successfully created (ec2-lambda-policy)

5. IAM Role Creation (lambdarole)

Here, using three screenshots, a new IAM role is created.

During the role creation:

- Trust entity: **AWS Lambda**
- Permission attached: The previously created policy

- Final role name: **lambdarole**

This role will later be assigned to the Lambda functions.

The screenshot shows the AWS IAM Roles page. On the left, there's a navigation sidebar with options like Dashboard, Access management (selected), Roles, Policies, Identity providers, Account settings, Root access management, Access reports, Access Analyzer, Resource analysis (New), Unused access, Analyzer settings, and Credential report. The main area displays a table of roles with columns for Role name, Trusted entities, and Last activity. Three roles are listed:

Role name	Trusted entities	Last activity
AWSServiceRoleForResourceExplorer	AWS Service: resource-explorer-2 (Service-Linked)	4 minutes ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked)	-

Below the table, there are sections for "Roles Anywhere" (info), "Access AWS from your non AWS workloads" (info), "X.509 Standard" (info), and "Temporary credentials" (info). A "Manage" button is located next to the "Temporary credentials" section. At the bottom right, there's a message about activating Windows.

Fig 6: IAM Role creation page — Selecting AWS Lambda as trusted entity.

The screenshot shows the "Add permissions" step of the IAM Role creation wizard. The steps are listed on the left: Step 1 (Select trusted entity), Step 2 (Add permissions, currently selected), and Step 3 (Name, review, and create). The main area is titled "Add permissions" and shows a search bar with "lambdapoli" and a filter dropdown set to "All types". One policy, "lambdapolicy", is listed under "Customer managed". Below the search bar, there's a note about setting a permissions boundary. At the bottom right, there are "Cancel", "Previous", and "Next" buttons, along with a message about activating Windows.

Fig 7: Attaching the custom EC2 policy to the new IAM role.

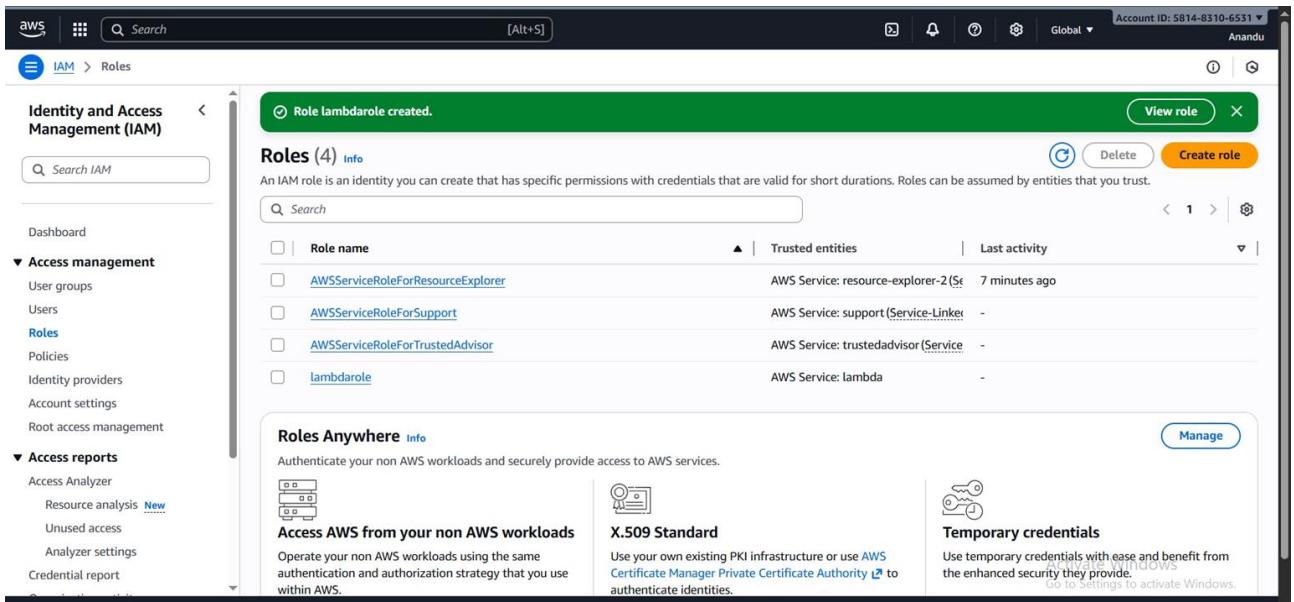


Fig 8: IAM Role successfully created (lambdarole).

6. Navigating to AWS Lambda

The screenshot shows how to access the Lambda service:

- In AWS Console → open **Services**
- Under **Compute** → select **Lambda**

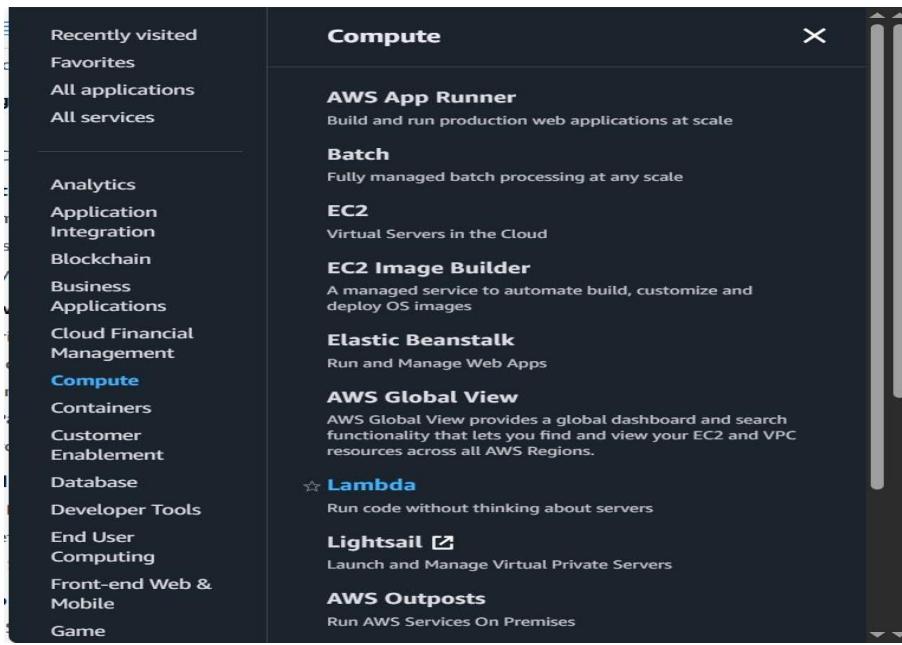


Fig 9: Navigating to AWS Lambda from the Compute section.

7. Lambda Homepage – Selecting “Create Function”

This screenshot shows the Lambda dashboard and the **Create function** button. This is where new automation functions are defined.

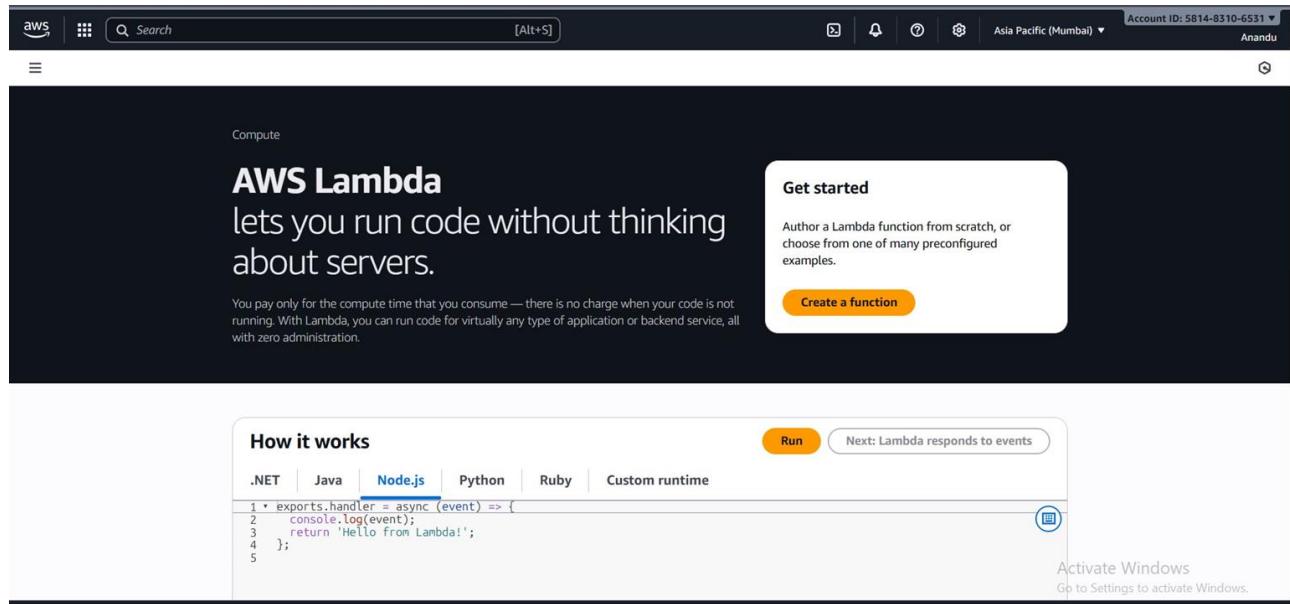


Fig 10: Lambda homepage — Selecting “Create Function

8. Choosing “RUNTIME LANGUAGE AS PYTHON”

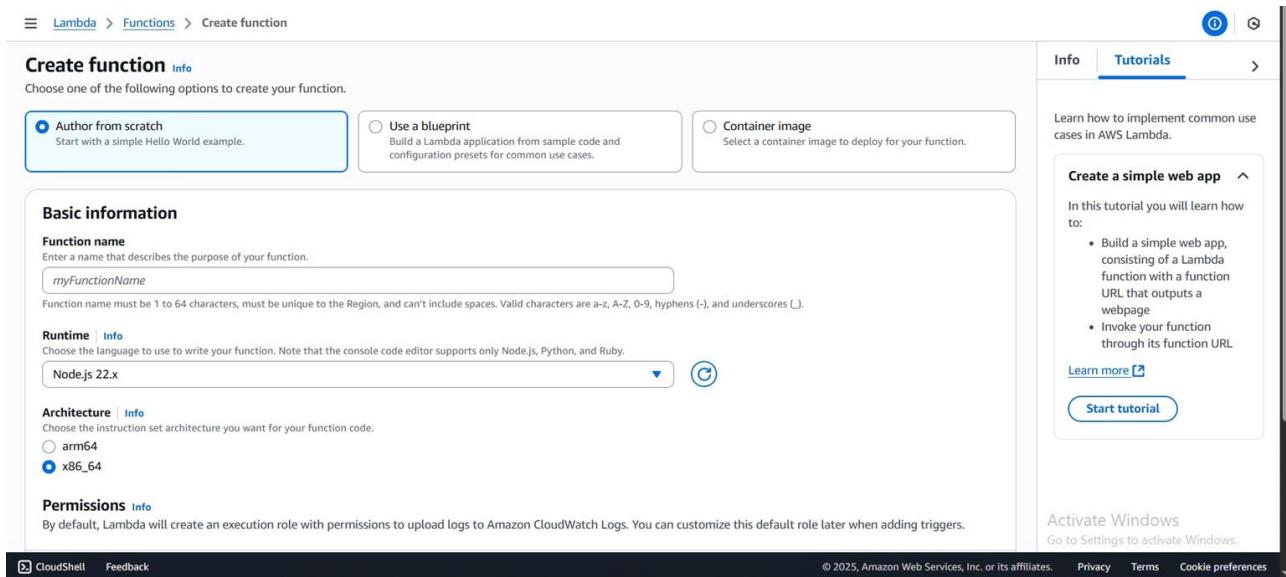


Fig 11: Choosing Python as the runtime language for the Lambda function.

9. Choosing “Use an Existing Role”

While creating the function, the screenshot highlights:

- “Change default execution role”
- Selecting **Use an existing role**

This ensures the Lambda function uses the IAM role **lambdarole**, which already contains the required EC2 permissions.

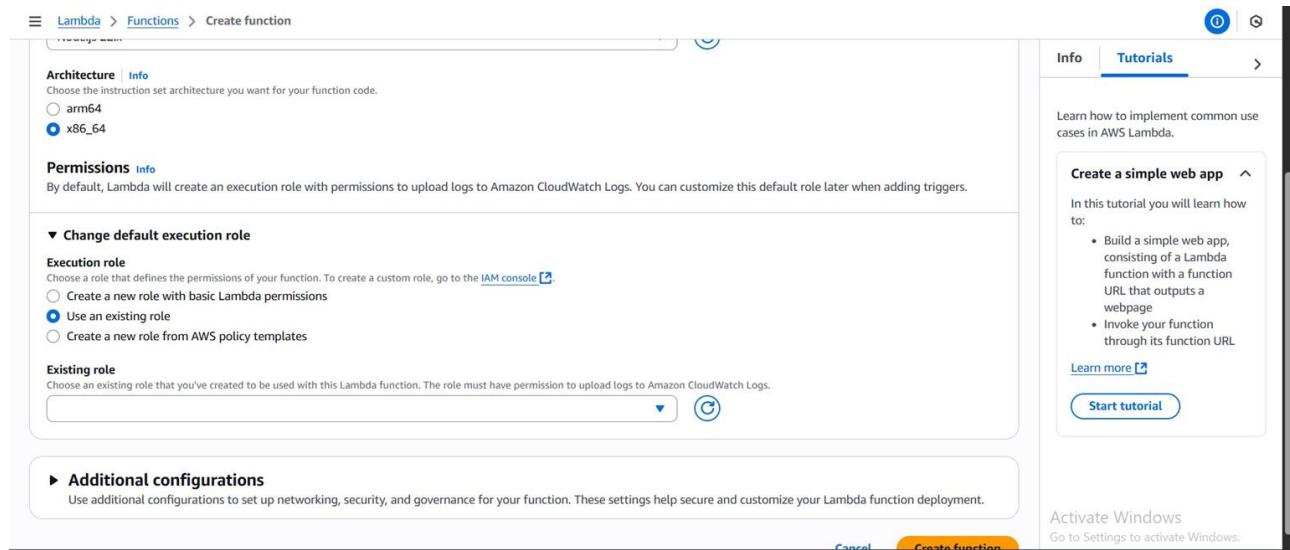


Fig 12: Selecting “Use an existing role” for execution permissions.

10. Selecting the Role ‘lambdarole’

The screenshot here confirms that the execution role chosen for the Lambda function is **lambdarole**.

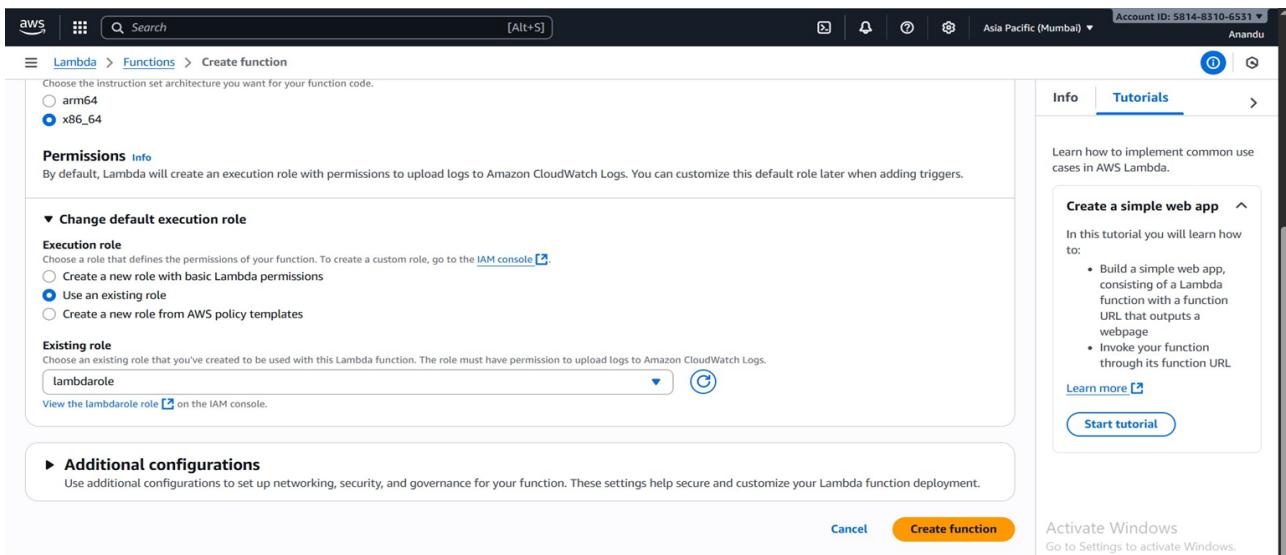


Fig 13: Assigning the IAM role *lambdarole* to the Lambda function.

11. Lambda Function Successfully Created

This image shows the successfully created Lambda function named **lambdafunction**, along with available options to edit its code and configuration.

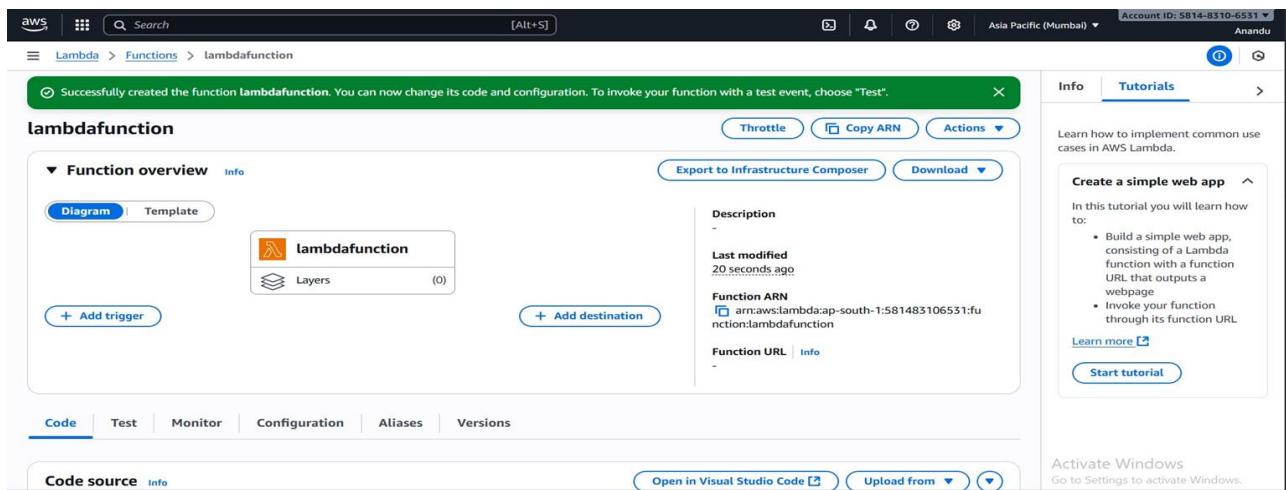


Fig 14: Lambda function created successfully with default configuration.

12. Uploading & Deploying lambdafunction.py

This screenshot shows the Lambda function updated with the Python script (**lambdafunction.py**) responsible for starting the EC2 instances.

After upload, the code is deployed and tested successfully.

```

import boto3

# EC2 Start Script for Mumbai Region
region = 'ap-south-1'
instances = ['i-08760efb3ce7cb26a', 'i-02e88ddf81b2110b9']
ec2 = boto3.client('ec2', region_name=region)

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

```

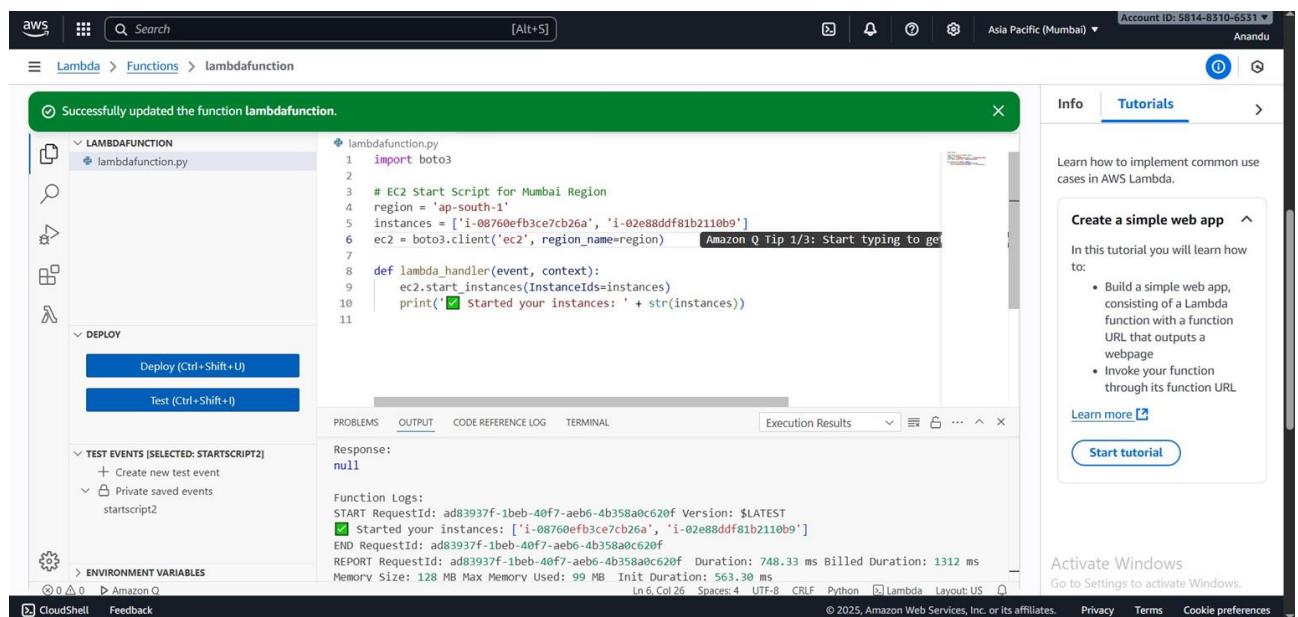


Fig 15: Lambda function updated with lambdafunction.py and deployed successfully.

13. Runtime Settings

The screenshot shows:

- Runtime: **Python 3.13**

- Handler: `lambda_function.lambda_handler`

This verifies that Lambda will execute the correct entry function.

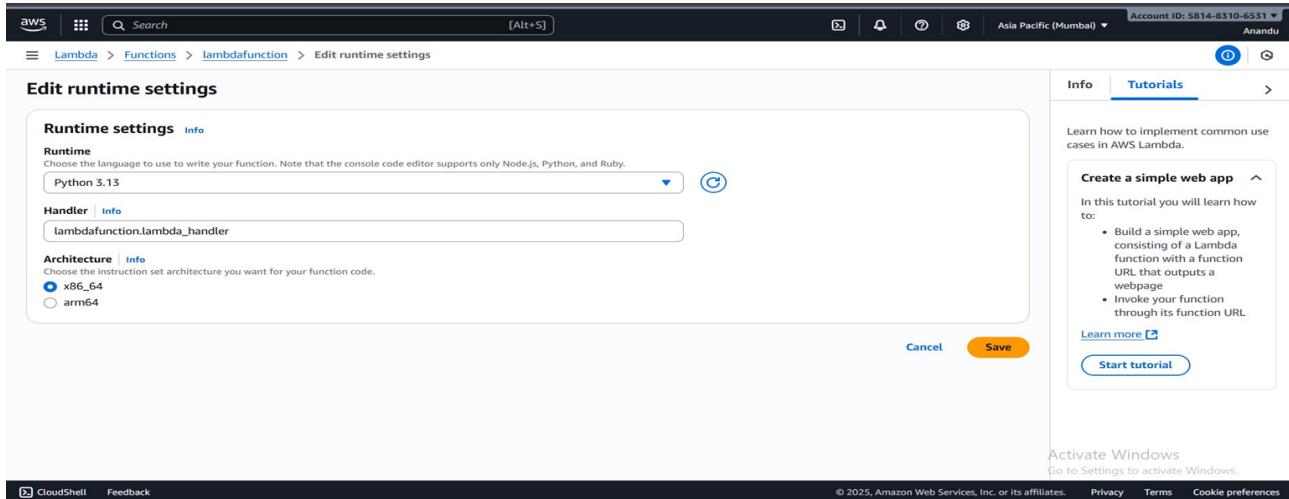


Fig 16: Runtime settings — Python 3.13 and correct Lambda handler configured.

14. Test Result – EC2 Instance Started

This screenshot shows that the previously stopped instance (node1/node2) is now running. This confirms the Lambda start script works correctly.

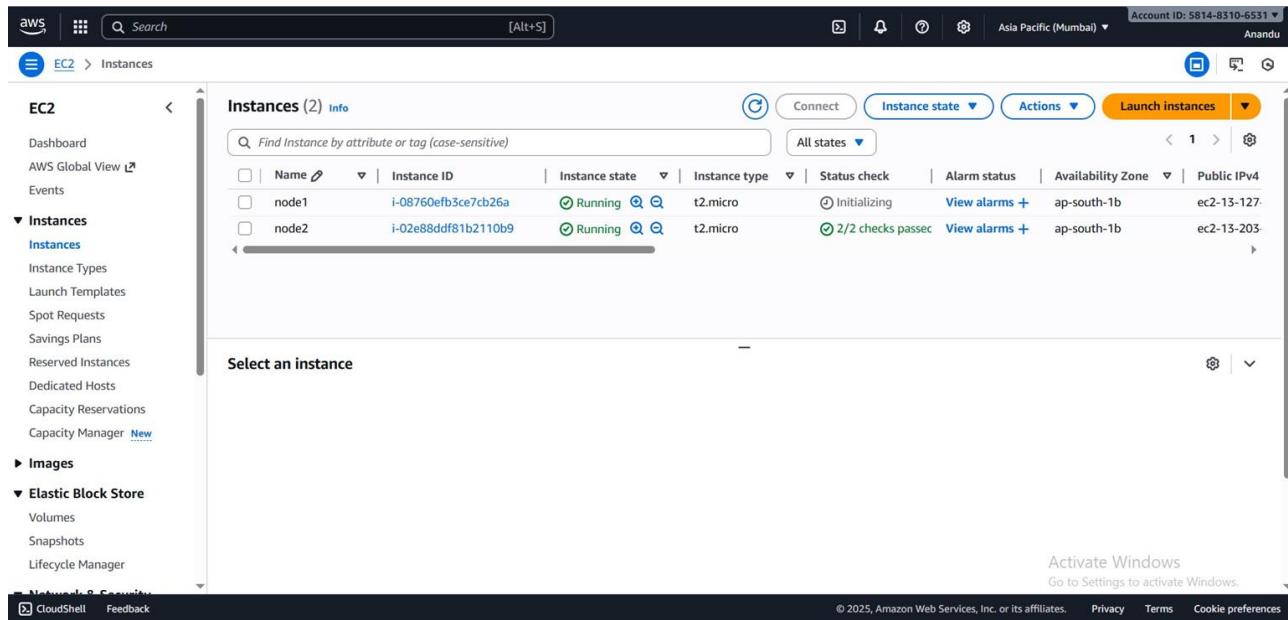


Fig 17: Test result — EC2 instance started successfully using start script.

15. Deploying EC2StopScript

Next screenshot shows uploading and deploying **EC2StopScript.py**, which stops the EC2 instances.

```
# EC2 Stop Script
```

```
import boto3

region = 'ap-south-1'
instances = ['i-08760efb3ce7cb26a', 'i-02e88ddf81b2110b9']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.stop_instances(InstanceIds=instances)
    print("Stopped your instances: " + str(instances))
```

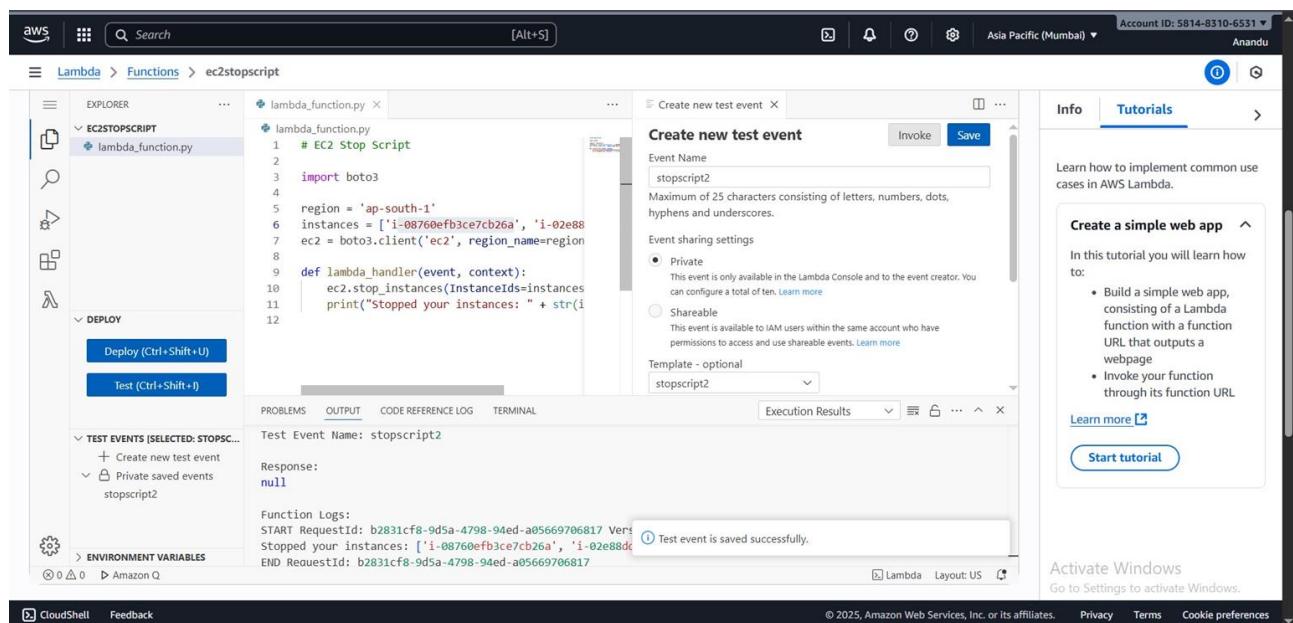


Fig 18: Deploying the EC2StopScript.py into the second Lambda function.

16. Test Result – EC2 Instance Stopped

Here, the EC2 instances transition from “running” to “stopped” due to the stop script deployment.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, AWS Global View, Events, Instances (selected), Images, Elastic Block Store, and more. The main area displays 'Instances (1/2) Info' with a search bar and filters for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. Two instances are listed: 'node2' (i-02e88ddf81b2110b9) which is Stopped, and 'node1' (i-08760efb3ce7cb26a) which is Stopping. Below this, a detailed view for 'node2' is shown with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The 'Details' tab is selected, showing fields for Instance ID (i-02e88ddf81b2110b9), Public IPv4 address (172.31.6.65), Private IPv4 addresses (172.31.6.65), Public DNS (172.31.6.65), and Instance state (Stopped).

Fig 19: Test result — EC2 instance stopped successfully using stop script.

17. Searching for CloudWatch

This screenshot shows how to open CloudWatch by searching in AWS Console.

The screenshot shows the AWS search results for 'cloudwatch'. The search bar at the top has 'cloudwatch' typed into it. The results are categorized into 'Services' and 'Features'. Under 'Services', 'CloudWatch' is listed as 'Monitor Resources and Applications'. Under 'Features', 'CloudWatch dashboard' is listed as a 'Systems Manager feature', and 'Cloud WAN' is listed as a 'VPC feature'. To the right of the search results, the AWS EC2 Instances page is visible, showing the same two instances as Fig 19: 'node2' (Stopped) and 'node1' (Stopping). The right-hand panel also displays private and public IP addresses and DNS names.

Fig 20: Searching for CloudWatch in the AWS console.

18. Opening Amazon EventBridge (CloudWatch)

This screenshot shows EventBridge, where automation rules will be created.

The screenshot shows the Amazon EventBridge dashboard. A blue banner at the top states: "You have been redirected. CloudWatch Events console is now deprecated. Use EventBridge console to create and manage event buses and rules." On the left, a sidebar menu includes options like Dashboard, Developer resources, Buses, Rules, Pipes, Scheduler, and Integration. The main area is titled "Rules" and shows a "Select event bus" section with a dropdown menu set to "default". Below this is a table titled "Rules on default event bus" which is currently empty, displaying the message "No rules. No rules to display." At the bottom right of the main area, there is a "Create rule" button. The top right corner of the screen shows account information: Account ID: 5814-8310-6531, Region: Asia Pacific (Mumbai), and User: Anandu.

Fig 21: Amazon EventBridge dashboard — Starting the rule creation process.

19. Time Zone (GMT) Reference

A screenshot explains that cron expressions use **GMT**.

Users must adjust schedules accordingly.

The screenshot shows a Google search results page with a dark theme. The search query "gmt time now" is entered in the search bar. The top result is a snippet showing the current time in Greenwich Mean Time (GMT): "11:50 am Thursday, 6 November 2025 Greenwich Mean Time (GMT)". Below this, under "People also ask", are several questions with dropdown answers: "What is GMT time for India?", "Is GMT 5.30 in India?", "Which country is GMT?", and "ଓট্টুয়ালৰ GMT 5.30 অৱেগমনি?". At the bottom right of the page, there is a "Activate Windows" message: "Go to Settings to activate Windows.".

Fig 22: Time zone reference showing cron expressions use GMT/UTC.

20. Creating a Cron Job Rule

This screenshot shows defining a cron expression to automatically trigger the Lambda start function at a scheduled time.

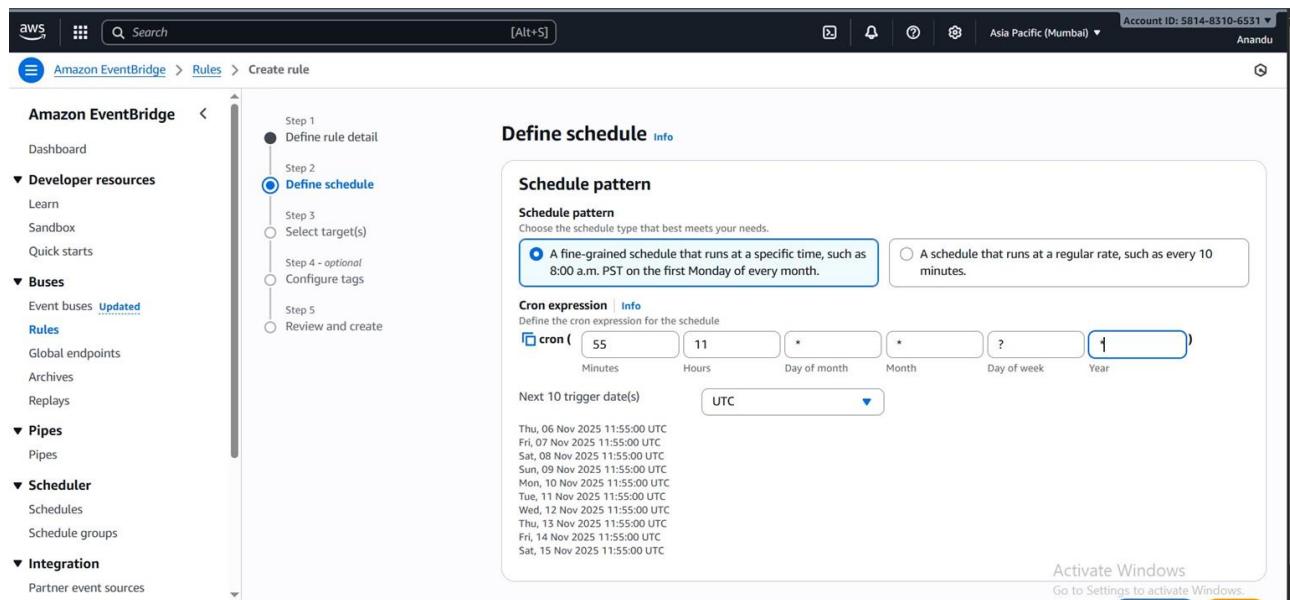


Fig 23: Creating a scheduled rule with a cron expression for EC2 start.

21. Selecting Lambda as the Target

Here you configure:

- **Target type:** AWS Lambda
- **Function name:** lambdafunction

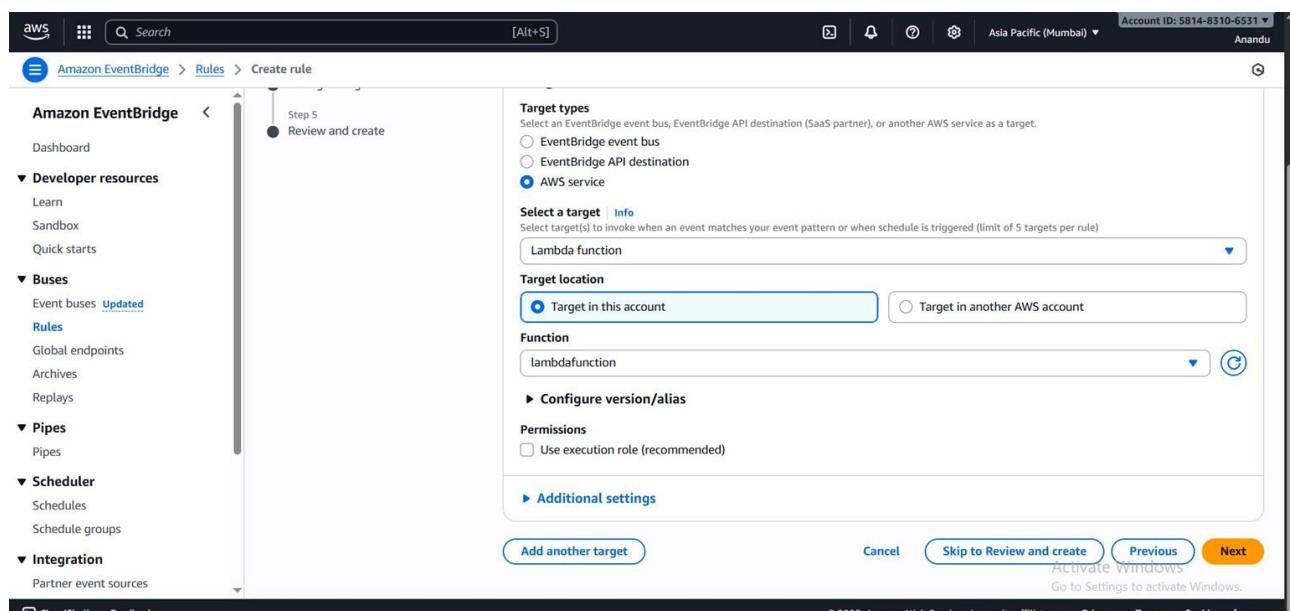


Fig 24: Selecting Lambda function as the target for the EventBridge rule.

- Permissions: Uncheck “Use execution role” (not needed because the role is already attached)

Then click **Next**.

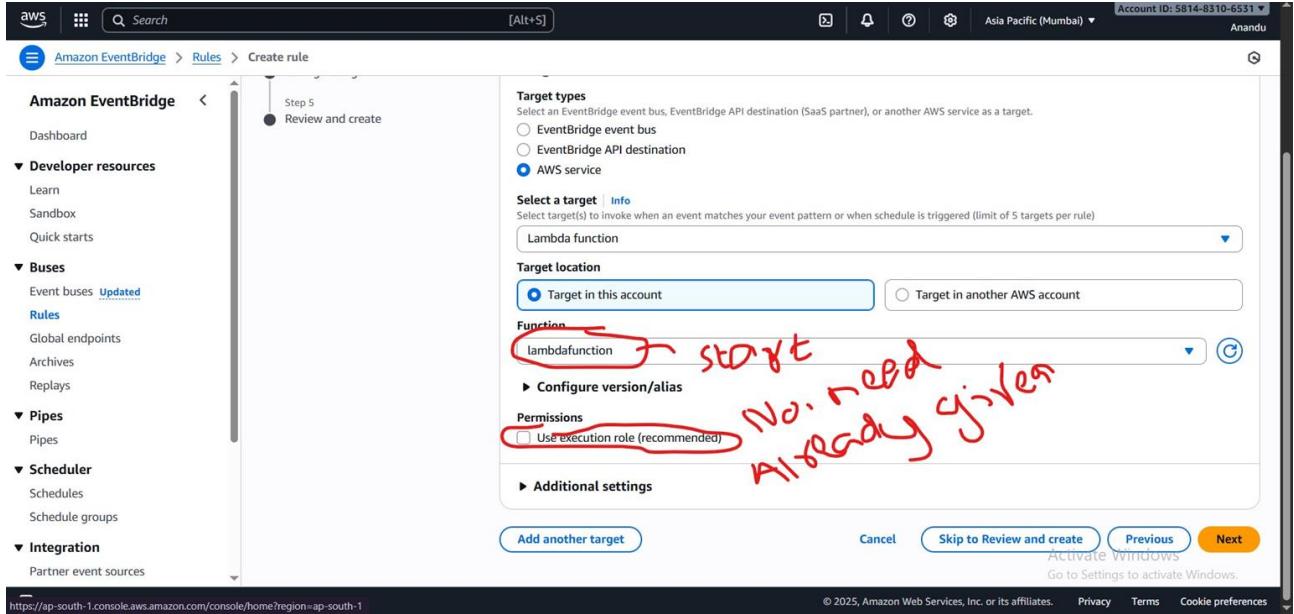


Fig 25: Permissions settings — Leaving execution role option unchecked.

22. Configure Tags (Optional)

This screenshot represents the tagging step. If tags are not required, you proceed without adding any.

Then click **Next**.

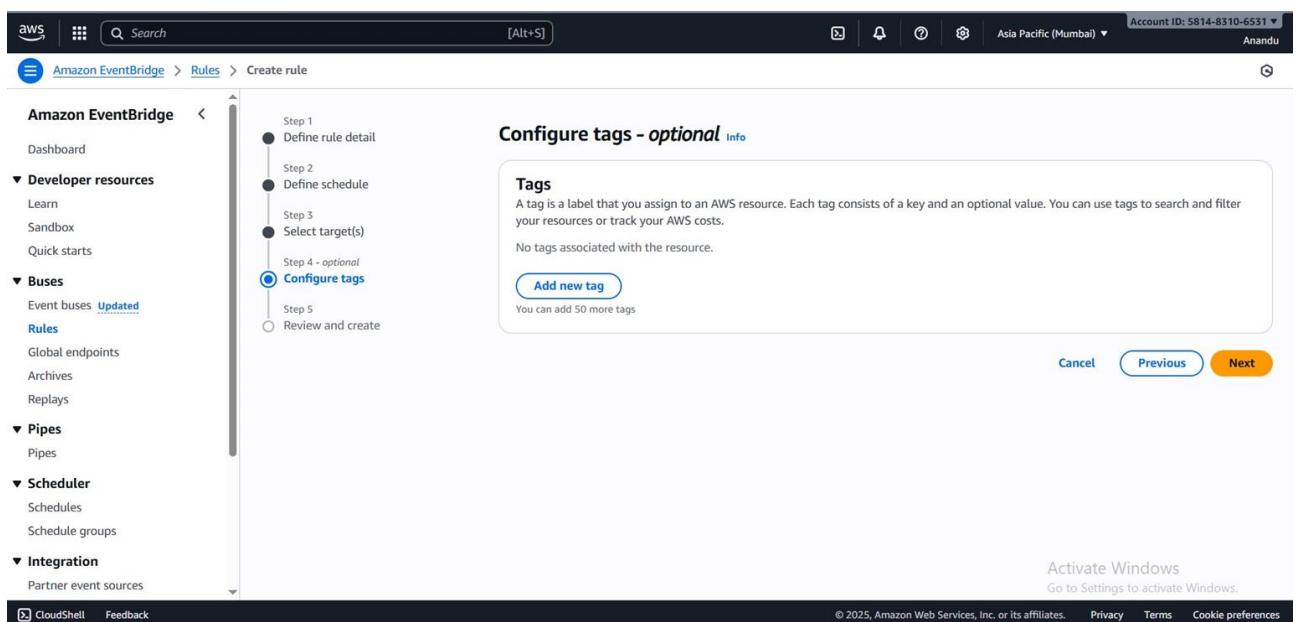


Fig 26: Optional tagging page — Proceeding without tags.

23. Review & Create Rule

This screenshot shows review of:

- Rule name, Target details, Schedule

Click **Create Rule**.

The screenshot shows the 'Create rule' review step in the Amazon EventBridge console. On the left, a sidebar navigation includes 'Event buses', 'Rules', 'Global endpoints', 'Archives', 'Replays', 'Pipes', 'Schedules', 'Schedule groups', 'Integration', 'Partner event sources', 'API destinations', and 'Connections'. The 'Rules' section is currently selected. The main area displays the 'Targets' configuration. A table lists one target: 'ec2stopscript' (Lambda function, ARN: arn:aws:lambda:ap-south-1:581483106531:function:ec2stopscript). Below the table, fields for 'Input to target' (Matched event), 'Additional parameters' (--), and 'Dead-letter queue (DLQ)' are shown. A 'Step 4: Configure tag(s)' section contains a table for adding tags, which is currently empty. At the bottom right are 'Cancel', 'Previous', 'Activate Windows' (button), 'Create rule' (button), and 'Go to Settings to activate Windows'.

Fig 27: Review page showing rule details before final creation.

24. Success Message – EC2-start Rule Created

This screenshot shows confirmation that the EventBridge rule **Ec2-start** was created successfully.

The screenshot shows the 'Rules' page in the Amazon EventBridge console. A green success message banner at the top reads 'Rule Ec2-start was created successfully'. The main content area shows the 'Select event bus' section with a dropdown set to 'default'. Below it is a table titled 'Rules on default event bus (1)'. The table has columns for Name, Status, Type, Event bus, ARN, and Description. It shows one rule named 'Ec2-start' with status 'Enabled', type 'Scheduled Standard', event bus 'default', ARN 'arn:aws:events:ap-south-1:581483106531:rule/Ec2-start', and description '-'. At the bottom right of the table are 'CloudFormation Template' and 'Create rule' buttons. The left sidebar navigation is identical to Fig 27.

Fig 28: Success message — EC2-start rule created successfully.

25. Instance Automatically Started

This screenshot shows the EC2 instance starting automatically due to the Ec2-start rule. This confirms the entire start automation pipeline works correctly.

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table titled "Instances (2) info" with the following data:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
node2	i-02e88ddf81b2110b9	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-13-204-
node1	i-08760efb5ce7cb26a	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-3-7-55-

Below the table, a section titled "Select an instance" is visible. The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows".

Fig 29: Instance automatically started by the EC2-start rule.

26. Creating the EC2-stop Rule

Similar to the start rule, a screenshot shows creation of an **Ec2-stop** scheduled rule.

Success Message – EC2-stop Rule Created

This screenshot shows confirmation that **Ec2-stop** rule was created successfully.

The screenshot shows the AWS Amazon EventBridge Rules page. The left sidebar is collapsed. The main area displays a table titled "Rules on default event bus (2)" with the following data:

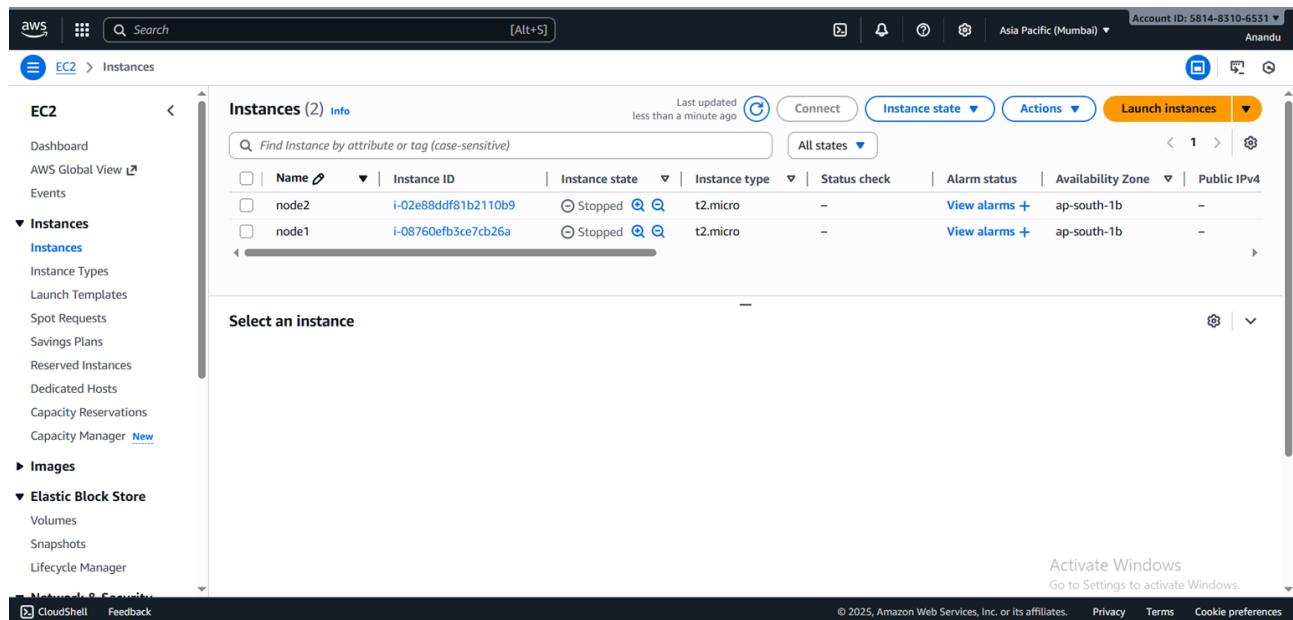
Name	Status	Type	Event bus	ARN	Description
Ec2-start	Enabled	Scheduled Standard	default	arn:aws:events:ap-south-1:58148310653:rule/Ec2-start	-
Ec2-stop	Enabled	Scheduled Standard	default	arn:aws:events:ap-south-1:58148310653:rule/Ec2-stop	-

The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows".

Fig 30: Success message — EC2-stop rule created successfully.

27. Instance Automatically Stopped

The last screenshot shows the EC2 instances transitioning to “stopped” automatically as a result of the Ec2-stop rule.



The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed, and the main content area displays the 'Instances (2) Info' table. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. Two instances are listed: 'node2' (Instance ID: i-02e88ddf81b2110b9) and 'node1' (Instance ID: i-08760efb3ce7cb26a). Both instances are in the 'Stopped' state. The 'Actions' dropdown menu is open, showing options like 'Start instances', 'Stop instances', and 'Reboot instances'. The top right corner shows the account ID: 5814-8310-6531 and the region: Asia Pacific (Mumbai).

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
node2	i-02e88ddf81b2110b9	Stopped	t2.micro	-	View alarms +	ap-south-1b	-
node1	i-08760efb3ce7cb26a	Stopped	t2.micro	-	View alarms +	ap-south-1b	-

Fig 31: Instance automatically stopped by the EC2-stop rule.

✓ CONCLUSION

In this project, AWS Lambda and Amazon EventBridge (CloudWatch) were successfully used to automate the lifecycle of EC2 instances. By creating IAM policies, roles, Lambda functions, and scheduled rules, the EC2 instances now start and stop automatically based on predefined cron expressions.

This automation helps reduce operational overhead and ensures cost efficiency by running compute resources only when required.

The workflow demonstrates real-world DevOps practices involving serverless functions, event scheduling, access control, and cloud resource management.

The screenshots and explanations together provide a complete understanding of how EC2 automation is implemented end-to-end in AWS.