

Assignment 1: Automated Instance Management Using AWS Lambda and Boto3

Launch EC2 Instances

1. In the AWS Console, search for EC2 in the search bar and click EC2.

2. Click Launch instances.

Name: NikhilAutoStop

Tag:

Key: Action

Value: Auto-Stop

Name: NikhilAutoStart

Tag:

Key: Action

Value: Auto-Start

AMI : Ubuntu

Instance Type : t2.micro

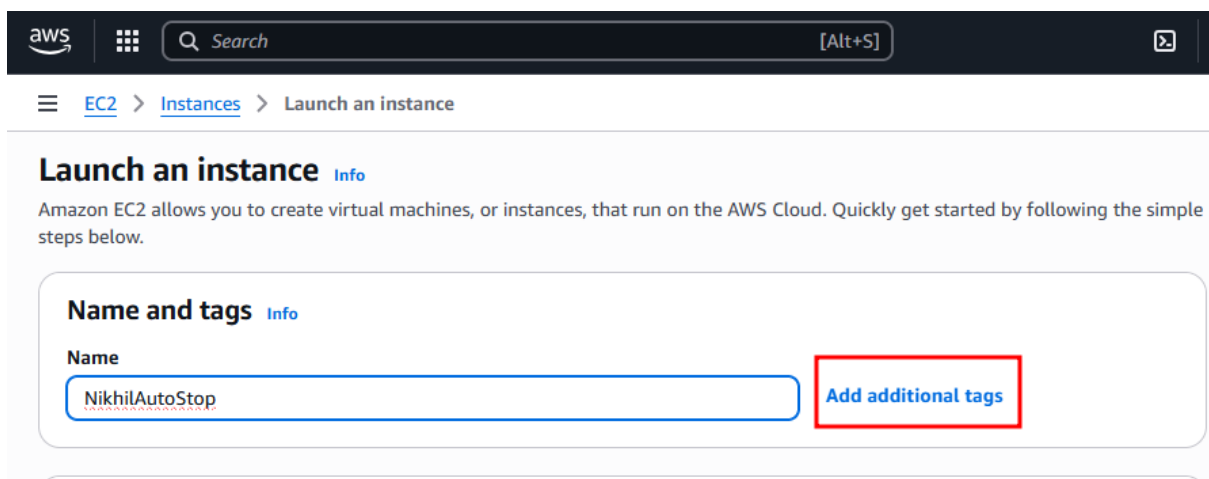
Key pair (login):

Network settings:

Allow SSH (port 22) from My IP.

Allow HTTP (port 80)

Click Launch instance



The screenshot shows the AWS Management Console interface for launching an EC2 instance. The top navigation bar includes the AWS logo, a search bar, and a [Alt+S] shortcut. The breadcrumb trail indicates the path: EC2 > Instances > Launch an instance. The main heading is 'Launch an instance' with an 'Info' link. Below this, a descriptive paragraph states: 'Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.' The 'Name and tags' section is expanded, showing a 'Name' label and a text input field containing 'NikhilAutoStop'. To the right of the input field is a red-bordered button labeled 'Add additional tags'.

aws

Search

[Alt+S]

Canada (Central)

Account ID: 9750-5002-4946

nikhilmathur1957@gmail.com

EC2 > Instances > Launch an instance

▼ Name and tags

Key

Info

Q Name

X

Value

Info

Q NikhilAutoStop

X

Resource types

Info

Select resource types

Instances

Remove

Key

Info

Q Action

X

Value

Info

Q Auto-Stop

X

Resource types

Info

Select resource types

Instances

Remove

Add new tag

You can add up to 48 more tags.

▼ Summary

Number of instances

Info

1

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd6...[read more](#)

ami-0c0a551d0459e9d39

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Cancel

Launch Instance

[Preview code](#)

aws

Search

[Alt+S]

EC2 > Instances > Launch an instance

Launch an instance

Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the steps below.

Name and tags

Info

Name

NikhilAutoStart

Add additional tags

Application and OS Image (Amazon Machine Image)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Key Info	Value Info	Resource types Info
<input type="text" value="Name"/>	<input type="text" value="NikhilAutoStart"/>	<div>Select resource types ▾</div> <div>Instances</div>
<div>Remove</div>		

Key Info	Value Info	Resource types Info
<input type="text" value="Action"/>	<input type="text" value="Auto-Start"/>	<div>Select resource types ▾</div> <div>Instances</div>
<div>Remove</div>		

Add new tag

You can add up to 48 more tags.

Instances (2) Info

Last updated less than a minute ago

Connect

Instance state ▾

Actions ▾

Launch instances

Find Instance by attribute or tag (case-sensitive)

All states ▾

nikhilauto

Clear filters

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
<input type="checkbox"/>	NikhilAutoStop	i-03fb7d87b3cfc797c	Running	t2.micro	2/2 checks passed	View alarms +	ca-central-1
<input type="checkbox"/>	NikhilAutoStart	i-0883fe058c0a89130	Running	t2.micro	Initializing	View alarms +	ca-central-1

Lambda Function Creation:

1. Create an IAM Role for Lambda

- AWS Console => IAM => Roles => Create role.
- Trusted Entity type: AWS Services
- Use Case: Lambda
- Click Next

The screenshot shows the 'Create role' wizard in the AWS IAM console. The left sidebar indicates the current step is 'Select trusted entity'. The main content area is titled 'Select trusted entity' and contains two sections: 'Trusted entity type' and 'Use case'. In the 'Trusted entity type' section, 'AWS service' is selected. In the 'Use case' section, 'Lambda' is selected from the 'Service or use case' dropdown. Below the dropdown, 'Choose a use case for the specified service.' shows 'Lambda' as the selected use case.

Permissions policies: AmazonEC2FullAccess

The screenshot shows the 'Add permissions' step in the AWS IAM console. The left sidebar indicates the current step is 'Add permissions'. The main content area is titled 'Add permissions' and contains a section for 'Permissions policies (1500)'. A search bar shows 'AmazonEC2FullAccess' with 1 match. Below the search bar, a table lists the policies. The 'AmazonEC2FullAccess' policy is highlighted with a red box. The table has columns for 'Policy name', 'Type', and 'Description'. Below the table, there is a link to 'Set permissions boundary - optional'. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons.

Policy name	Type	Description
AmazonEC2FullAccess	AWS managed	Provides full access to Amazon EC2 via th...

Role name: NikhilLambdaEC2ControlRole
Click Create role.

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Description
Add a short explanation for this role.

Step 1: Select trusted entities

Trust policy

```

1- {
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Effect": "Allow",
6-       "Action": [
7-         "sts:AssumeRole"
8-       ],
9-       "Principal": {
10-        "Service": [
11-          "lambda.amazonaws.com"
12-        ]
13-      }
14-    }
15-  ]

```

Role NikhilLambdaEC2ControlRole created. [View role](#)

Roles (594) [Info](#) [Delete](#) [Create role](#)

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.

1 match

<input type="checkbox"/>	Role name	Trusted entities	Last activity
<input type="checkbox"/>	NikhilLambdaEC2ControlRole	AWS Service: lambda	-

2. Create the Lambda Function

Go to AWS Console => Lambda.

Click Create function.

Select Author from scratch

Function name: **NikhilEC2TagBasedControl**

Runtime: Python 3.12 (or latest).

Permissions:

Expand Change default execution role.

Select Use an existing role.

Choose **NikhilLambdaEC2ControlRole** from the dropdown.

Note => I choose the role **prashantb12-role-9p53470y** for permission access to run the code.

Click Create function.

aws

Search

[Alt+S]

Canada (Central)

Account ID: 9750-5002-4946
nikhilmathur1957@gmail.com

Lambda > Functions > Create function

Create function [info](#)

Choose one of the following options to create your function.

☒ 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.

NikhilEC2TagBasedControl

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

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 3.13

Architecture

[info](#)

Choose the instruction set architecture you want for your function code.

☐ arm64

☒ x86_64

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.

aws

Search

[Alt+S]

Canada (Central)

Account ID: 9750-5002-4946
nikhilmathur1957@gmail.com

Lambda > Functions > Create function

x86_64

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.

NikhilLambdaEC2ControlRole

[View the NikhilLambdaEC2ControlRole role](#) on the IAM console.

► Additional configurations

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

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

aws

Search

[Alt+S]

Canada (Central)

Account ID: 9750-5002-4946
nikhilmathur1957@gmail.com

Lambda > Functions > NikhilEC2TagBasedControl

Successfully created the function NikhilEC2TagBasedControl. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

NikhilEC2TagBasedControl

[Throttle](#) [Copy ARN](#) [Actions](#)

▼ Function overview [info](#)

Diagram

Template

NikhilEC2TagBasedControl

Layers (0)

[+ Add trigger](#)

[+ Add destination](#)

Export to Infrastructure Composer

Download

Description

-

Last modified

3 seconds ago

Function ARN

arn:aws:lambda:ca-central-1:975050024946:function:NikhilEC2TagBasedControl

3. Add Python Code to Control EC2

- prints instance IDs with a given tag:
- Code to print instances with Auto-Stop (testing code)

```
Assignment1.py > ...
1
2
3 import boto3
4
5 def lambda_handler(event, context):
6     ec2 = boto3.client('ec2')
7
8     # Find instances with Action=Auto-Stop
9     response = ec2.describe_instances(
10         Filters=[
11             {'Name': 'tag:Action', 'Values': ['Auto-Stop']}
12         ]
13     )
14
15     print(response['Reservations'])
16
17     instances = []
18     for reservation in response['Reservations']:
19         for instance in reservation['Instances']:
20             instances.append(instance['InstanceId'])
21
22     print(f"Found instances with Auto-Stop tag: {instances}")
23     return {"instances": instances}
24
```

■ Full Code

```
Assignment1.py > lambda_handler
1 import boto3
2
3 def lambda_handler(event, context):
4     ec2 = boto3.client('ec2')
5
6     # --- Stop instances with Auto-Stop tag ---
7     stop_response = ec2.describe_instances(
8         Filters=[
9             {'Name': 'tag:Action', 'Values': ['Auto-Stop']},
10            {'Name': 'instance-state-name', 'Values': ['running']} # Only running ones
11        ]
12    )
13
14    stop_ids = []
15    for reservation in stop_response['Reservations']:
16        for instance in reservation['Instances']:
17            stop_ids.append(instance['InstanceId'])
18
19    if stop_ids:
20        ec2.stop_instances(InstanceIds=stop_ids)
21        print(f"Stopping instances: {stop_ids}")
22    else:
23        print("No running instances with Auto-Stop tag found.")
24
25    # --- Start instances with Auto-Start tag ---
26    start_response = ec2.describe_instances(
27        Filters=[
28            {'Name': 'tag:Action', 'Values': ['Auto-Start']},
29            {'Name': 'instance-state-name', 'Values': ['stopped']} # Only stopped ones
30        ]
31    )
32
```

```
Assignment1.py > lambda_handler

30     ]
31 )
32
33 start_ids = []
34 for reservation in start_response['Reservations']:
35     for instance in reservation['Instances']:
36         start_ids.append(instance['InstanceId'])
37
38 if start_ids:
39     ec2.start_instances(InstanceIds=start_ids)
40     print(f"Starting instances: {start_ids}")
41 else:
42     print("No stopped instances with Auto-Start tag found.")
43
44 return {
45     "StoppedInstances": stop_ids,
46     "StartedInstances": start_ids
47 }
48
```

Create the test case

Menu: Lambda > Functions > NikhilEC2TagBasedControl

Code | **Test** | Monitor | Configuration | Aliases | Versions

Executing function: succeeded (logs) Details

Test event info Delete CloudWatch Logs Live Tail Save Test

To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save.

Test event action

☐ Create new event ☒ Edit saved event

Event name

NikhilMathurTestEvent

Event JSON Format JSON

1 { }

Click for deploy the code

Menu: Lambda > Functions > NikhilEC2TagBasedControl

EXPLORER: NIKHILEC2TAGBASEDCONTROL > lambda_function.py

DEPLOY (UNDEPLOYED CHANGES) Deploy (Ctrl+Shift+U) Test (Ctrl+Shift+I)

TEST EVENTS [SELECTED: NIKHILMATHURTESTEVENT] Create new test event Private saved events NikhilMathurTestEvent

ENVIRONMENT VARIABLES

lambda_function.py

```
def lambda_handler(event, context):
    stop_ids = []
    for reservation in stop_response['Reservations']:
        for instance in reservation['Instances']:
            stop_ids.append(instance['InstanceId'])

    if stop_ids:
        ec2.stop_instances(InstanceIds=stop_ids)
        print(f"Stopping instances: {stop_ids}")
    else:
        print("No running instances with Auto-Stop tag found.")

    # --- Start instances with Auto-Start tag ---
    start_response = ec2.describe_instances(
        Filters=[
            {'Name': 'tag:Action', 'Values': ['Auto-Start']},
            {'Name': 'Instance-state-name', 'Values': ['stopped']} # Only stopped ones
        ]
    )

    start_ids = []
    for reservation in start_response['Reservations']:
```

PROBLEMS OUTPUT CODE REFERENCE LOGS TERMINAL

Request ID: 07ad97b2-b68a-4e64-b440-6fef72281b3a

Deploying code

Click on test for Output

Successfully updated the function NikhilEC2TagBasedControl.

DEPLOY

Deploy (Ctrl+Shift+U)

Test (Ctrl+Shift+I)

TEST EVENTS [SELECTED: NIKHILMATHURTESTEVEN...]

Create new test event

Private saved events

NikhilMathurTestEvent

ENVIRONMENT VARIABLES

45

PROBLEMS

OUTPUT

CODE REFERENCE LOG

TERMINAL

Execution Res

Response:

```
{
  "StoppedInstances": [],
  "StartedInstances": [
    "i-0737a00de076a0662",
    "i-0dd7161ab60933034",
    "i-094c8593c8a4e8ea5",
    "i-06146594887d6f02",
    "i-03fb7d87b3cfc797c",
    "i-0f57563aaf0fda17c",
    "i-03e903f6748810bb9",
    "i-033722e1f5f006ae3",
    "i-0f38a58ec7341c52e",
    "i-039ca85a1cac6c5ba"
  ]
}
```

Ln 28, Col 57 Spaces: 4 UTF-8 LF P

Verification

Instances (2) Info

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

Find Instance by attribute or tag (case-sensitive)

All states

NikhilAuto

Clear filters

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
<input type="checkbox"/>	NikhilAutoStop	i-03fb7d87b3cfc797c	Stopped	t2.micro	-	View alarms +	ca-central-1b	-
<input type="checkbox"/>	NikhilAutoStart	i-0883fe058c0a89130	Running	t2.micro	2/2 checks passed	View alarms +	ca-central-1b	ec2-16-5