

AWS Lambda Function with AWS CloudTrail



What is Lambda in AWS used for?

AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. These events may include changes in state or an update, such as a user placing an item in a shopping cart on an ecommerce website.

Create a Custom Managed Policy for AWS Lambda

- Create Policy with the Following Permissions
- AWS EC2 and CloudWatch Logs

The screenshot shows the AWS IAM console 'Specify permissions' page. The breadcrumb trail is 'IAM > Policies > Create policy'. The page is divided into two steps: 'Step 1: Specify permissions' (active) and 'Step 2: Review and create'. The main heading is 'Specify permissions' with an 'info' link. Below it, a subtitle reads: 'Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.' There are three tabs: 'Visual', 'JSON' (selected), and 'Actions'. The 'Policy editor' section shows a JSON snippet in a code editor with line numbers 1 through 11. The JSON is partially filled out:

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

 To the right of the code editor is the 'Edit statement' panel. It contains the text 'Select a statement' and 'Select an existing statement in the policy or add a new statement.' Below this is a button labeled '+ Add new statement'.

- You can use the following Json or Create your own Json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:Start*",
        "ec2:Stop*"
      ],
      "Resource": "*"
    }
  ]
}
```

- Name the Policy as LambdaPolicy
- Review and Create

The screenshot shows the 'Review and create' step in the AWS IAM console. The policy name is 'LambdaPolicy'. The description is optional. The permissions section shows that the policy allows actions on EC2 and CloudWatch Logs. The 'Add tags' section is optional.

Policy details

Policy name:

Description - optional:

Permissions defined in this policy

Service	Access level	Resource	Request condition
EC2	Limited Write	All resources	None
CloudWatch Logs	Limited Write	region: thing like /all	None

Create an IAM policy and IAM role for your Lambda function

- Create a Role with Identity and Access Management (IAM) and name the role as Lambda Role
- Search IAM in AWS Console

The screenshot shows the 'IAM Dashboard' in the AWS IAM console. It displays security recommendations, AWS account information, and IAM resources.

Security recommendations

- Add MFA for root user
- Deactivate or delete access keys for root user

AWS Account

Account ID: 666395655405

Account Alias: Create

Sign-in URL for IAM users in this account: <https://666395655405.signin.aws.amazon.com/console>

Quick Links

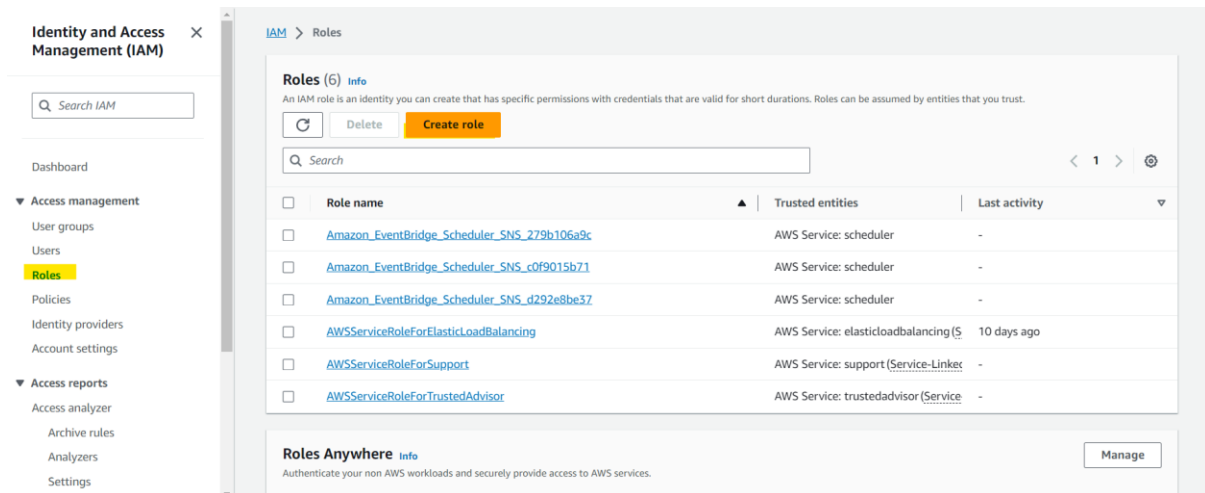
[My security credentials](#)

Manage your access keys, multi-factor authentication (MFA) and other credentials.

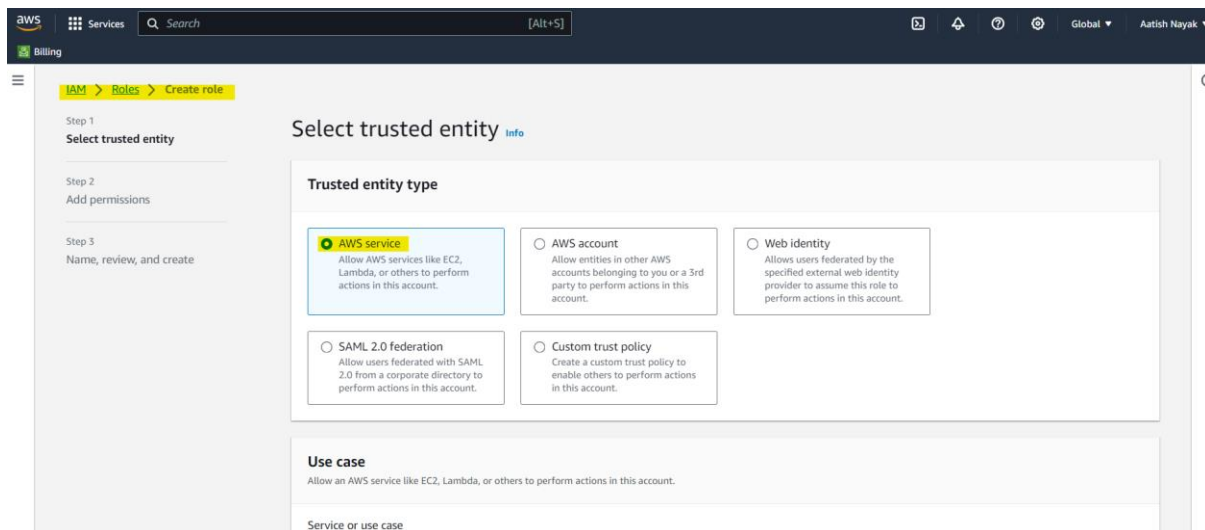
IAM resources

User groups	Users	Roles	Policies	Identity providers
0	0	6	4	0

- Click on Roles
- Click on create Role



- Select the Service as AWS Service
- Select the case use as Lambda
- Click Next Once You Select the Service



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

- Select the Custom Managed Policy

IAM > Roles > Create role

Step 1

Select trusted entity

Step 2

Add permissions

Step 3

Name, review, and create

Add permissions

Permissions policies (1/886)

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

Search

lambda

policy

Filter by Type

All types

1 match

<input checked="" type="checkbox"/>	Policy name	Type	Description
<input checked="" type="checkbox"/>	lambda	Customer managed	-

Set permissions boundary - optional

Cancel

Previous

Next

- Name, review, and create
- Name the Role as LambdaRole
- Review and Click on Create

Services

Search

[Alt+S]

Global

Astish Nayak

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Access analyzer

Role LambdaRole created.

View role

Roles (7)

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.

Refresh

Delete

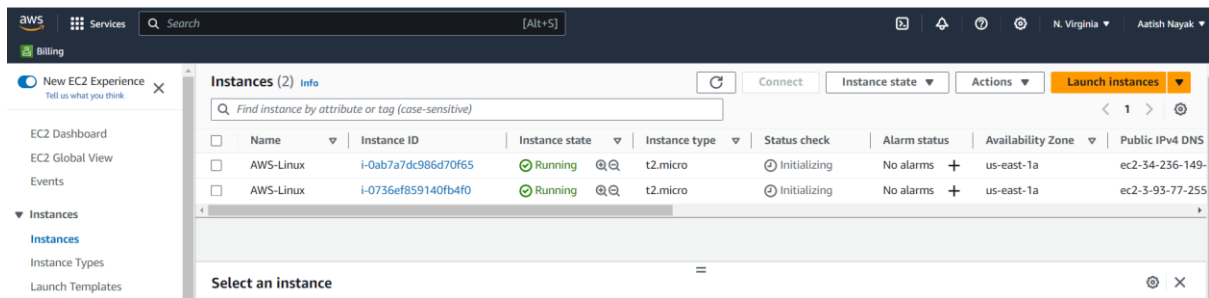
Create role

Search

1

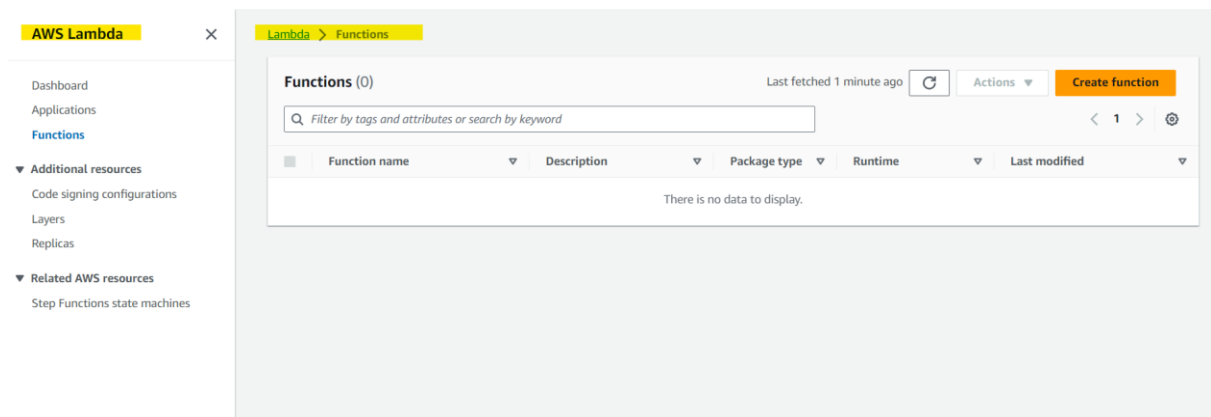
<input type="checkbox"/>	Role name	Trusted entities	Last activity
<input type="checkbox"/>	Amazon_EventBridge_Scheduler_SNS_c0f9015b71	AWS Service: scheduler	-
<input type="checkbox"/>	Amazon_EventBridge_Scheduler_SNS_d292e8be37	AWS Service: scheduler	-
<input type="checkbox"/>	AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (\$	10 days ago
<input type="checkbox"/>	AWSServiceRoleForSupport	AWS Service: support (Service-Link	-
<input type="checkbox"/>	AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-
<input checked="" type="checkbox"/>	LambdaRole	AWS Service: lambda	-

- Create two Ec2 instance
- You can create two Ec2 instance with operating System RedHat or AWS Linux

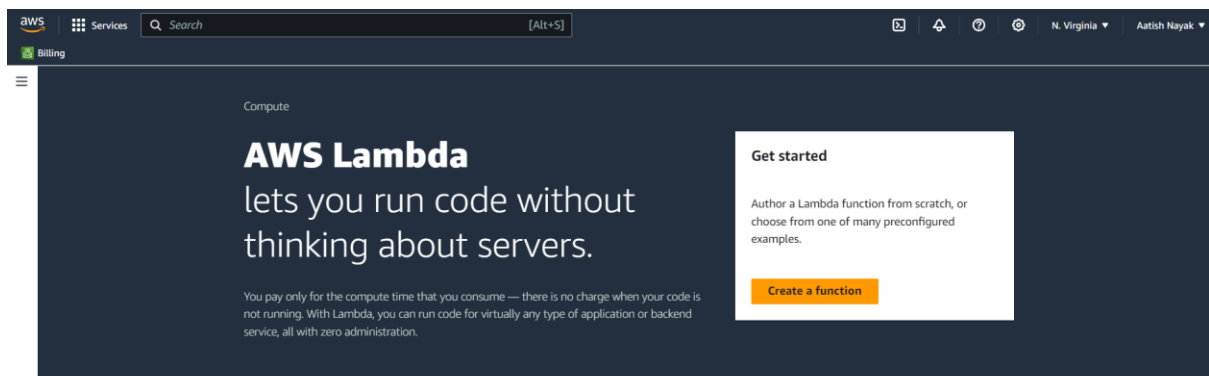


Create Lambda functions that stop and start your EC2 instances

- Open the Lambda console, and then choose Create function



- Click on Create function (Step1)
- Choose Author from scratch
- Under Basic information, enter the following information



- For **Function name**, enter a name that identifies it as the function that's used to stop your EC2 instances. For example, "Ec2start".
- For **Runtime**, choose **Python 3.9**.
- Under **Permissions**, expand **Change default execution role**.
- Under **Execution role**, choose **Use an existing role**.
- Under **Existing role**, choose the IAM role that you created.
- Choose **Create function**.

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.
☒ **Python 3.9** ▼ ↻

Architecture Info
Choose the instruction set architecture you want for your function code.
☒ **x86_64**
☐ arm64

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.

LambdaRole

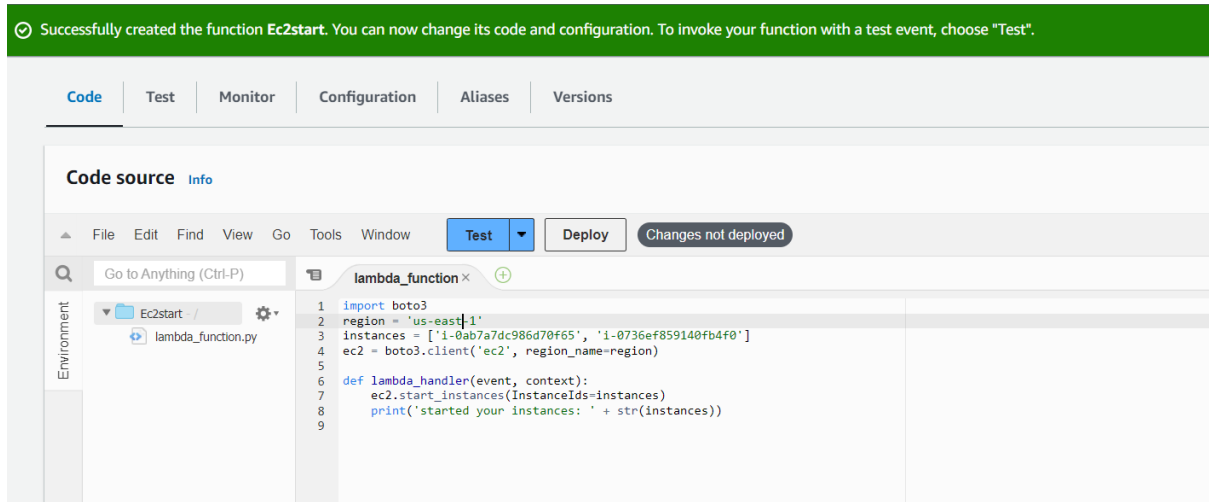
► **Advanced settings**

- On the Code tab, under Code source, paste the following code into the editor pane in the code editor on the lambda_function tab. This code starts the EC2 instances that you identify.

```
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))
```

- **Important:** For region, **replace** "us-east-1" with the AWS Region that your instances are in. For instances, replace the example EC2 instance IDs with the IDs of the specific instances that you want to stop and start.



- Click on Test Code
- Configure test event

Configure test event

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

Ec2Start

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

hello-world

Event JSON

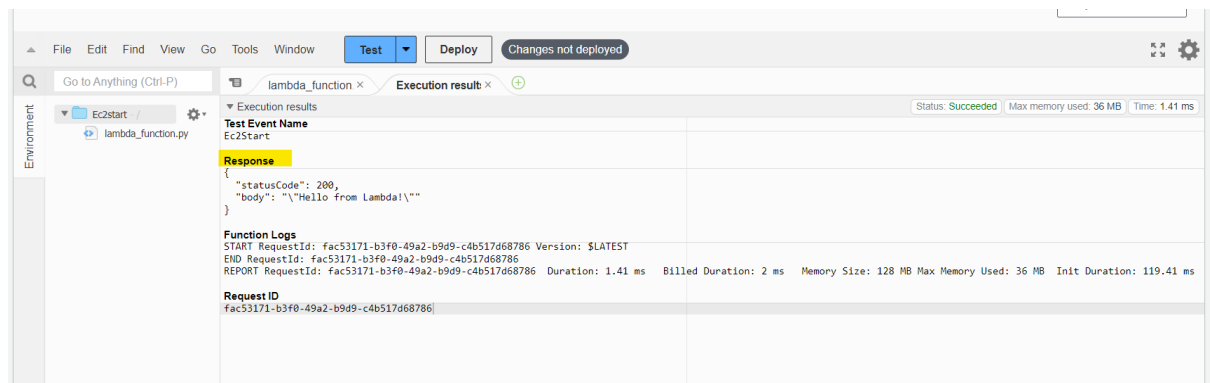
Format JSON

Cancel

Invoke

Save

- Click on Save option to save the Event Created (Step7)
- Once the Event is created click on Test Code to get the Response



- Click on Deploy if you want to start the Instance which is Stopped
- **Successfully updated the function Ec2start.** (Results After Deploying)
- On the **Configuration** tab, choose **General configuration**, and then choose **Edit**. Set **Timeout** to 10 seconds, and then choose **Save**.
- **Note:** [Configure the Lambda function settings](#) as needed for your use case. For example, to stop and start multiple instances, you might use a different value for **Timeout** and **Memory**.
- Repeat steps 1-7 to create another function. Complete the following steps differently so that this function starts your EC2 instances.

Python Code to Stop Ec2 Instance

```
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))
```

AWS CloudTrail

CloudTrail enables auditing, security monitoring, and operational troubleshooting by tracking user activity and API usage. CloudTrail logs, continuously monitors, and retains account activity related to actions across your AWS infrastructure, giving you control over storage, analysis, and remediation actions.

Note: You can use CloudTrail to check for events to confirm that the Lambda function stopped or started the EC2 instance.

1. Open the [CloudTrail console](#).
2. In the navigation pane, choose **Event history**.
3. Choose the **Lookup attributes** dropdown list, and then choose **Event name**.
4. In the search bar, enter **StopInstances** to review the results. Then, enter **StartInstances** in the search bar to review the results.

If there are no results, then the Lambda function didn't stop or start the EC2 instances.

Create EventBridge rules that run your Lambda functions

1. Open the [EventBridge console](#).
 2. Select **Create rule**.
 3. Enter a **Name** for your rule, such as "StopEC2Instances". (Optional) Enter a description for the rule in **Description**.
 4. For **Rule type**, choose **Schedule**, and then choose **Continue in EventBridge Scheduler**.
 5. For Schedule pattern, choose **Recurring schedule**.
 6. Under **Schedule pattern**, for **Occurrence**, choose **Recurring schedule**.
 7. For **Schedule type**, choose the type that's right for your need and complete the following steps:
When **Schedule type** is **Rate-based schedule**, for **Rate expression**, enter a rate value and choose an interval of time in minutes, hours, or days.
-or-
When **Schedule type** is **Cron-based schedule**, for **Cron expression**, enter an expression that tells Lambda when to stop your instance. For information on expression syntax, see [Schedule expressions for rules](#).
- Note:** Cron expressions are evaluated in UTC. Make sure that you adjust the expression for your preferred time zone.
8. In **Select targets**, choose **Lambda function** from the **Target** dropdown list.

9. For **Function**, choose the function that stops your EC2 instances.
10. Choose **Skip to review and create**, and then choose **Create**.
11. Repeat steps 1-10 to create a rule to start your EC2 instances. Complete the following steps differently:
Enter a name for your rule, such as "StartEC2Instances".
(Optional) In **Description**, enter a description for your rule, such as "Starts EC2 instances every morning at 7 AM."
In step 7, for **Cron expression**, enter an expression that tells Lambda when to start your instances.
In step 9, for **Function**, choose the function that starts your EC2 instances.

Note: Sometimes, a Lambda function can stop an Amazon EC2 instance and not be able to start it again. This can occur when an Amazon Elastic Block Store (Amazon EBS) volume is encrypted and the Lambda role isn't authorized to use the encryption key. For more information, see [Required AWS KMS key policy for use with encrypted volumes](#) and [Key policies in AWS KMS](#).

