Automating EC2 Start/Stop with AWS Lambda & EventBridge

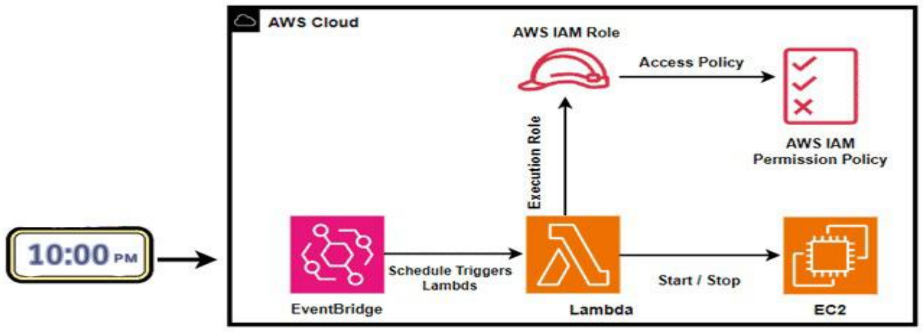
This project helps optimize cloud Costs by shutting down instances during non-working hours and restarting them when needed. The system ensures that EC2 instances are stopped automatically during idle periods (like nights or weekends) and started again during active hours.

Key Steps:

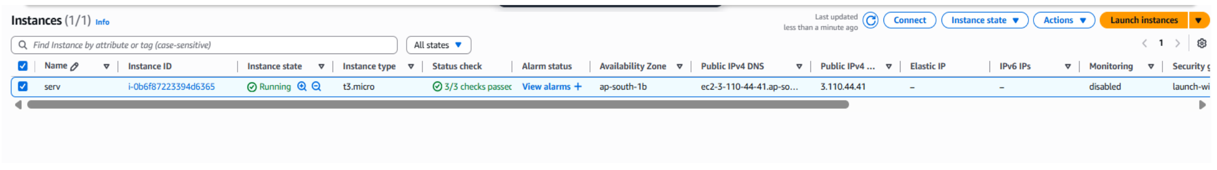
1. **Provisioned an EC2 Instance** – Created an EC2 instance to demonstrate the automation.
2. **Configured IAM Role & Policies** – Assigned an IAM role with least-privilege permissions to allow Lambda to start/stop EC2 securely.
3. **Configured Lambda Function** – Wrote a Python-based Lambda function to handle EC2 start/stop logic.
4. **Integrated with EventBridge Scheduler** – Set up cron-based **EventBridge** rules to trigger Lambda at specific times (e.g., stop at 10 PM, start at 8 AM).
5. **Verify the Lambda Function & EventBridge Rule** – Tested the workflow to confirm EC2 stops/starts as expected and verified EventBridge rule execution.

Cost Optimization:

- Reduce EC2 running hours by up to 70–80% during inactive times.
- Development or testing environments that don't need 24/7 uptime.
- Improved operational efficiency with hands-free instance management.



1. Provisioned an EC2 Instance



Instances (1/1) Info

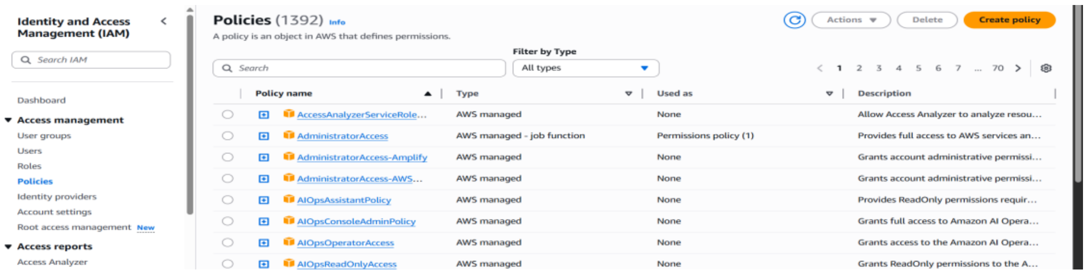
Last updated less than a minute ago | Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security g
serv	i-0b6f87223394d6365	Running	t3.micro	3/3 checks passec	View alarms +	ap-south-1b	ec2-3-110-44-41.ap-so...	3.110.44.41	–	–	disabled	launch-wi

2. Configured IAM Role & Policies.

a.) Click on the Policies from the Left Side and click on **Create Policy.**



Identity and Access Management (IAM)

Search IAM

Dashboard
- Access management
  - User groups
  - Users
  - Roles
  - Policies
  - Identity providers
  - Account settings
  - Root access management New
- Access reports
  - Access Analyzer

Policies (1392) Info

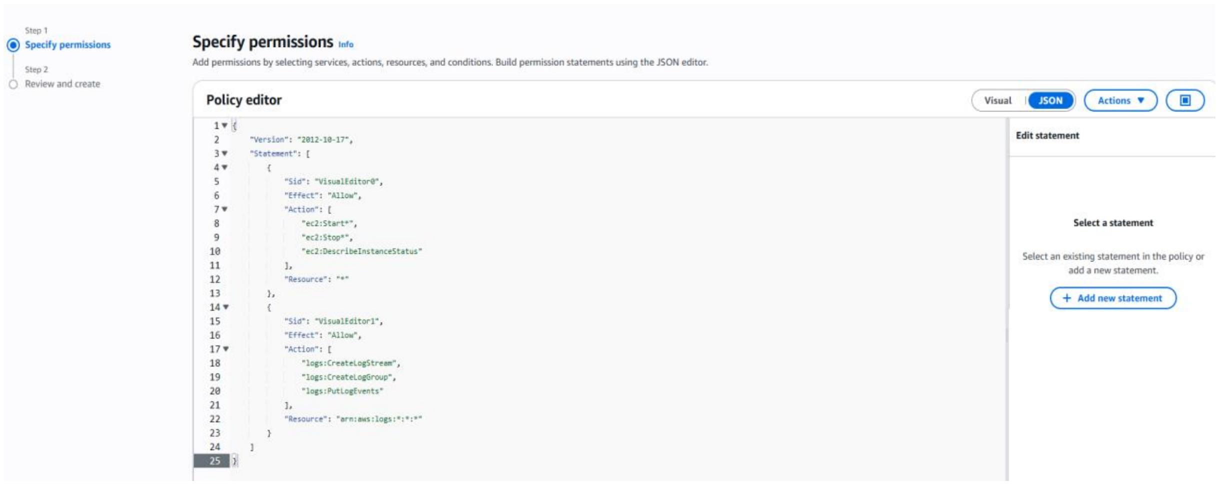
A policy is an object in AWS that defines permissions.

Actions | Delete | Create policy

Search | Filter by Type: All types | < 1 2 3 4 5 6 7 ... 70 >

Policy name	Type	Used as	Description
AccessAnalyzerServiceRole...	AWS managed	None	Allow Access Analyzer to analyze resou...
AdministratorAccess	AWS managed - job function	Permissions policy (1)	Provides full access to AWS services an...
AdministratorAccess-Amplify	AWS managed	None	Grants account administrative permissi...
AdministratorAccess-AWS...	AWS managed	None	Grants account administrative permissi...
AIOpsAssistantPolicy	AWS managed	None	Provides ReadOnly permissions requir...
AIOpsConsoleAdminPolicy	AWS managed	None	Grants full access to Amazon AI Opera...
AIOpsOperatorAccess	AWS managed	None	Grants access to the Amazon AI Opera...
AIOpsReadOnlyAccess	AWS managed	None	Grants ReadOnly permissions to the A...

b.) Replace the existing Policy with the below policy.



Specify permissions Info

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ec2:Start*",
                "ec2:Stop*",
                "ec2:DescribeInstanceStatus"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
```

Select a statement

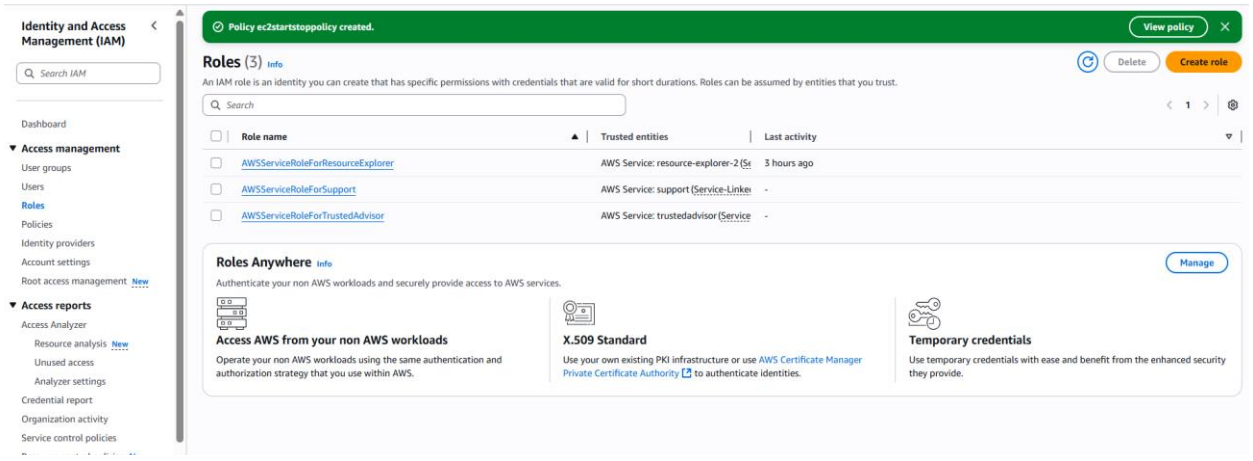
Select an existing statement in the policy or add a new statement.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "ec2:Start*",
        "ec2:Stop*",
        "ec2:DescribeInstanceStatus"
      ],
      "Resource": "*"
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:CreateLogGroup",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

- ***Github Repo:-*** https://github.com/xrootms/Automating-EC2-Start-Stop-with-Lambda-.git

c.) Policy Name as “**ec2startstoppolicy** “and click on Create policy.
Policy ec2startstoppolicy created.

d.) Now Click on the **Roles**



Identity and Access Management (IAM)

Search IAM

Dashboard
▼ Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
▼ Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Credential report
Organization activity
Service control policies

Policy ec2startstoppolicy created. View policy

Roles (3) 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.

Search

Role name	Trusted entities	Last activity
AWSServiceRoleForResourceExplorer	AWS Service: resource-explorer-2 (Se	3 hours ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linke	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-

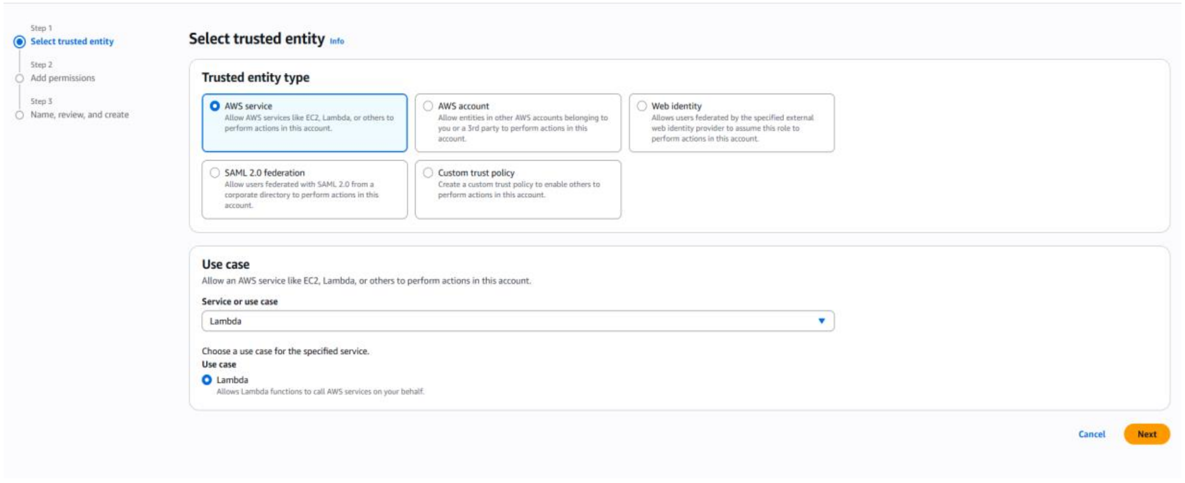
Roles Anywhere Info
Manage
Authenticate your non AWS workloads and securely provide access to AWS services.

Access AWS from your non AWS workloads
Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.

X.509 Standard
Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities.

Temporary credentials
Use temporary credentials with ease and benefit from the enhanced security they provide.

e.) Select **AWS Service** as the trusted entity type



Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create

Select trusted entity Info

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.

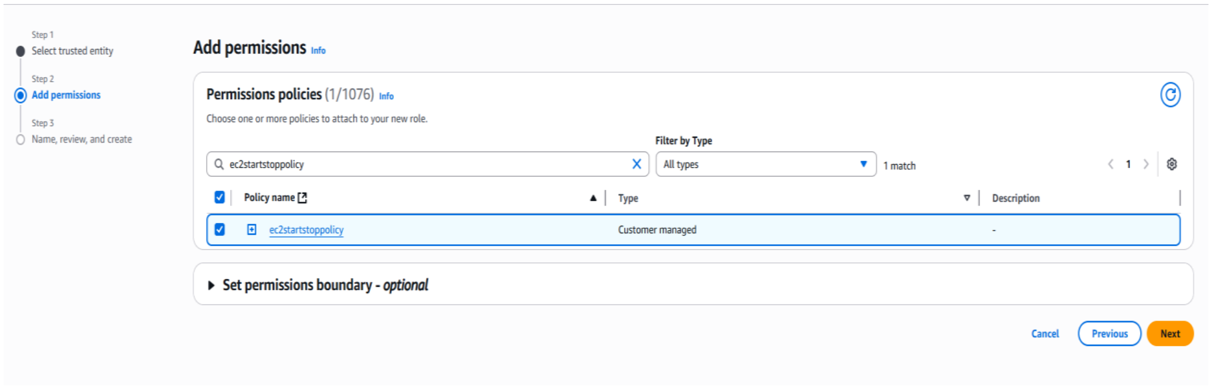
Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

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

f.) Search for the **ec2startstoppolicy** we had created and click Next



Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create

Add permissions Info

Permissions policies (1/1076) Info
Choose one or more policies to attach to your new role.

Filter by Type

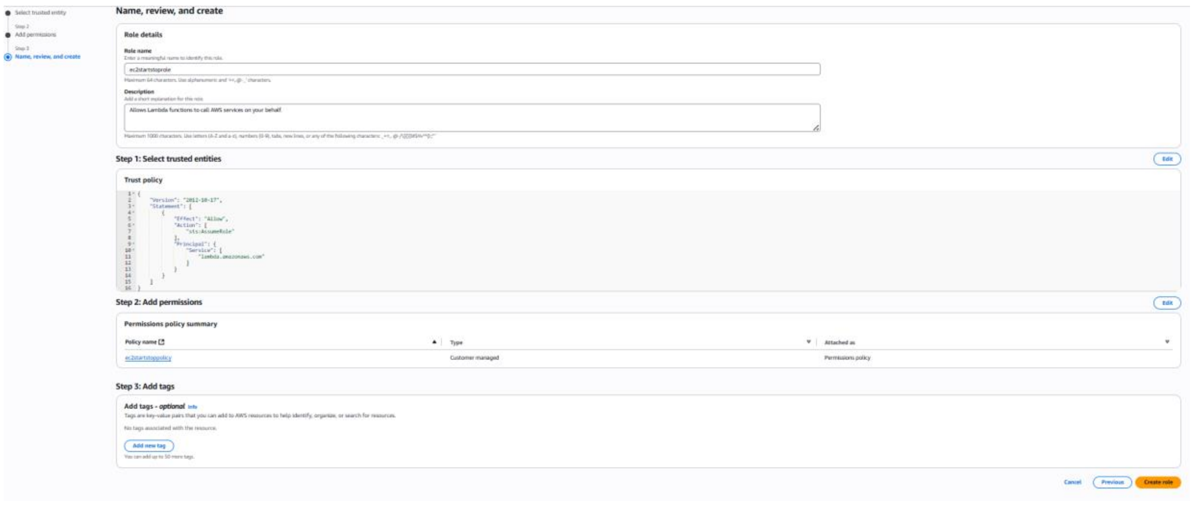
ec2startstoppolicy | All types | 1 match

✓	Policy name	Type	Description
✓	ec2startstoppolicy	Customer managed	-

▶ Set permissions boundary - *optional*

Cancel Previous Next

g.) Provide the name as **ec2startstoprole** and click on **Create Role.**



Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create

Name, review, and create

Role details

Role name: ec2startstoprole

Description: Allows Lambda functions to call AWS services on your behalf.

Step 1: Select trusted entities

Trust policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "lambda.amazonaws.com"
                ]
            }
        }
    ]
}
```

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
ec2startstoppolicy	Customer managed	Permissions policy

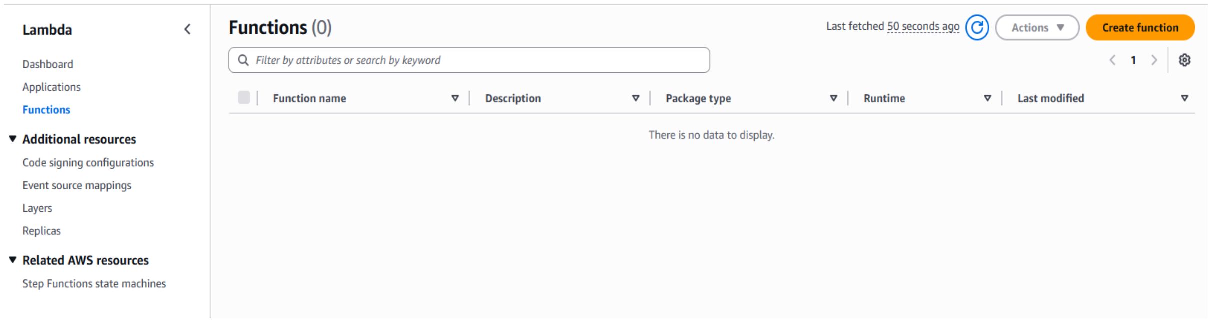
Step 3: Add tags

Add tags - *optional*
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.
You can add up to 50 more tags.

h.**)** **ec2startstoprole** is created successfully.

3. Configure the Lambda Function

a.) Search for **Lambda** and Click on **Create a function**.



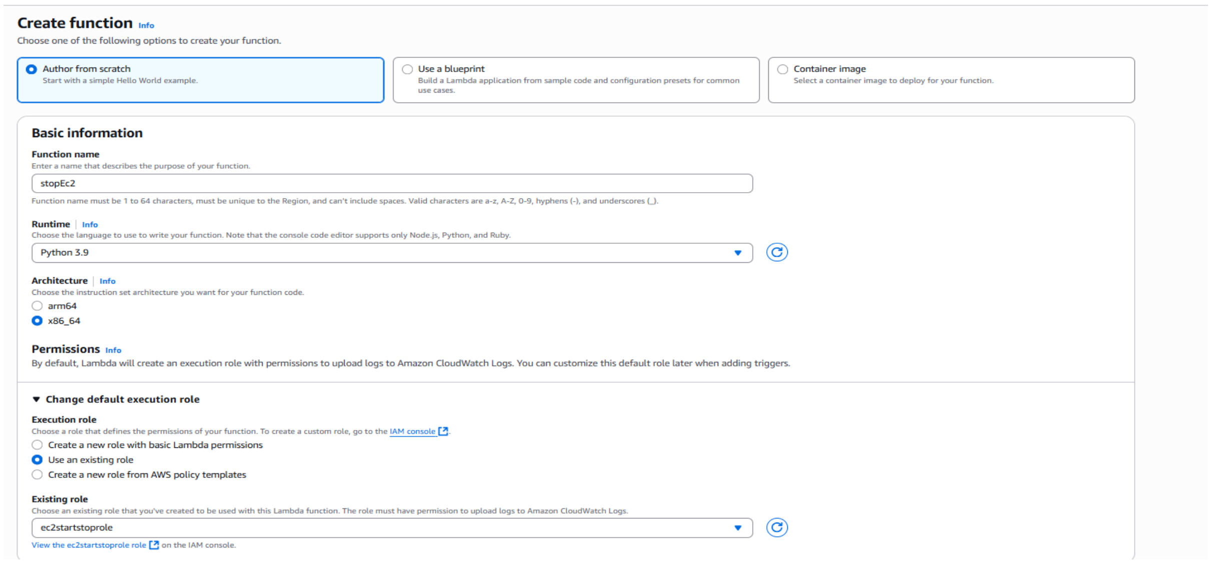
Functions (0)

Last fetched 50 seconds ago

Function name	Description	Package type	Runtime	Last modified

There is no data to display.

b.) Select the **Author from scratch** Name as **LambdaAutomation** and Select the Runtime as **Python 3.9.** select the Use an **existing role** chooses the role we created and then click on **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.
stopEc2
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 3.9

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.

▼ 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.
ec2startstoprole
View the ec2startstoprole role on the IAM console.

c.) Here, we have successfully created the **Lambda Function**. Under Code, Replace the existing code with the below Python code and click on deploy to save it.

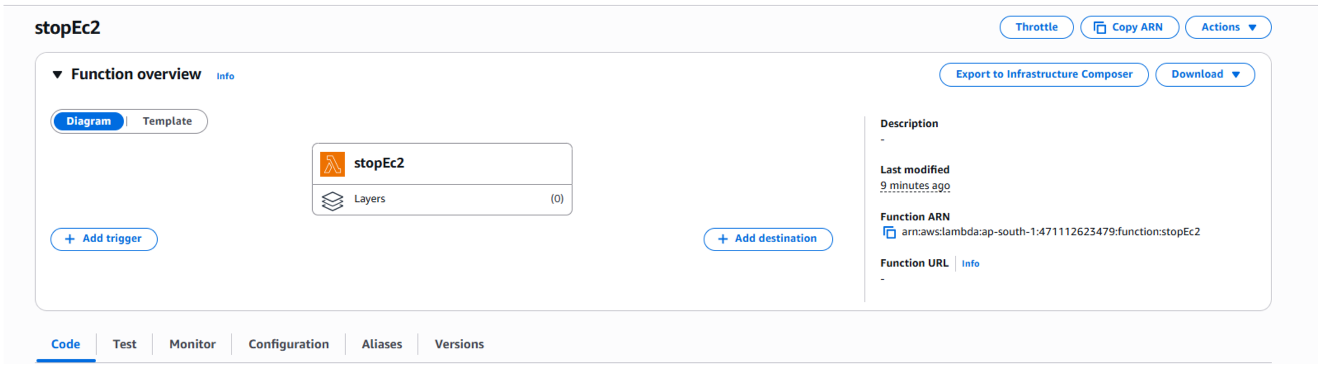
```
import boto3

region = 'us-east-1'

instances = ['i-001e2d04e37ccde3b']

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

def lambda_handler(event, context):

   print('Stopping instances')

   ec2.stop_instances(InstanceIds=instances)
```

- *(Note: update the **region = 'your region'** & **instances= ['your instance id']**)*
- ***GitHub Repo:-*** https://github.com/xrootms/Automating-EC2-Start-Stop-with-Lambda-.git

Now we have successfully deployed the Python code in the Lambda Function.

4. Integrated with EventBridge Scheduler

a.) Navigate to Lambda Console and click on **Add Trigger**.



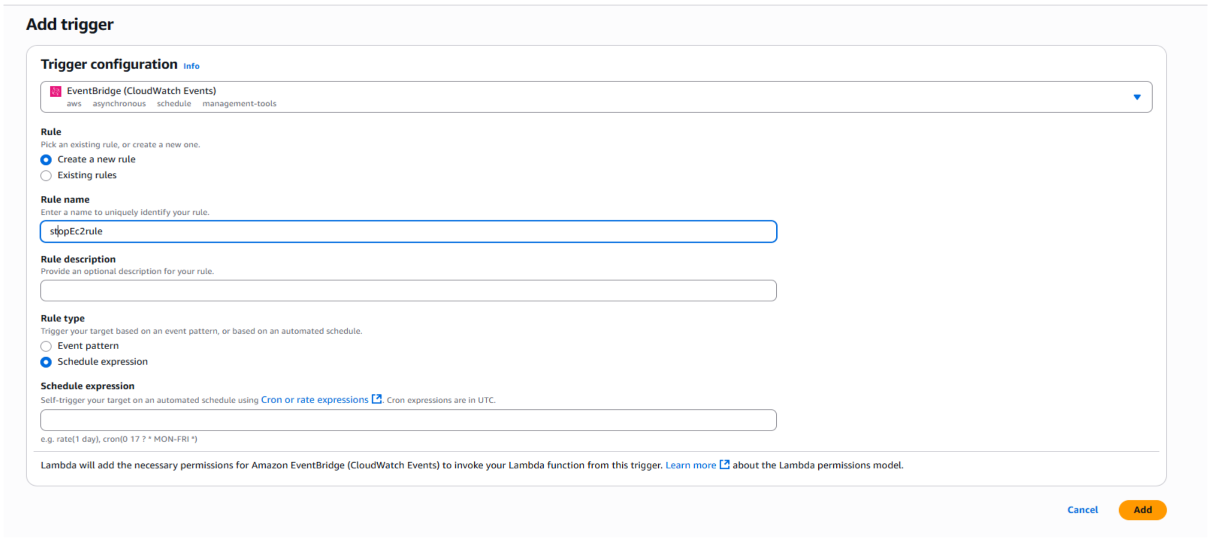
Description
-

Last modified
9 minutes ago

Function ARN
arn:aws:lambda:ap-south-1:471112623479:function:stopEc2

Function URL Info
-

b.) Select the **EventBridge (CloudWatch Events), Select Create a new rule and Select the Rule type as Schedule expression.**



EventBridge (CloudWatch Events)
aws asynchronous schedule management-tools

Rule
Pick an existing rule, or create a new one.
[✓ Create a new rule] Existing rules

Rule name
Enter a name to uniquely identify your rule.
stopEc2rule

Rule description
Provide an optional description for your rule.

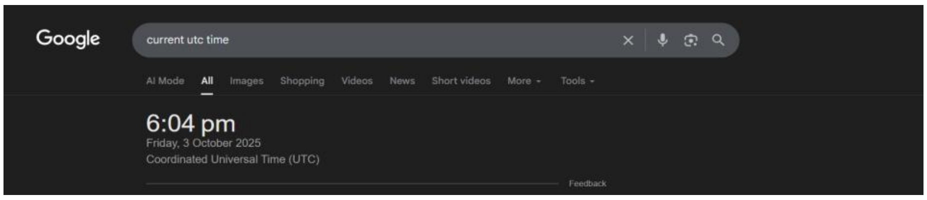
Rule type
Trigger your target based on an event pattern, or based on an automated schedule.
Event pattern [✓ Schedule expression]

Schedule expression
Self-trigger your target on an automated schedule using Cron or rate expressions. Cron expressions are in UTC.

e.g. rate(1 day), cron(0 17 ? * MON-FRI *)

Lambda will add the necessary permissions for Amazon EventBridge (CloudWatch Events) to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

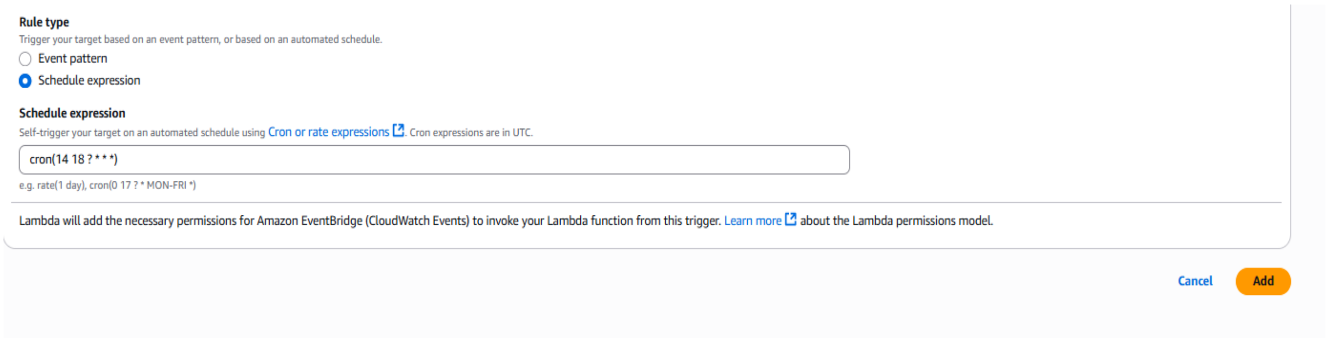
c.) Now in the New Google Tab, search for the **UTC Time Right Now.**



6:04 pm
Friday, 3 October 2025
Coordinated Universal Time (UTC)

(Note: Adjust the UTC time based on your local time zone.)

d.) we will write the Expression as ***cron(14 18 ? * * *)*** for the instance to be stopped after **10 minutes.**



Rule type
Trigger your target based on an event pattern, or based on an automated schedule.
( ) Event pattern
(●) Schedule expression

Schedule expression
Self-trigger your target on an automated schedule using Cron or rate expressions. Cron expressions are in UTC.

cron(14 18 ? * * *)

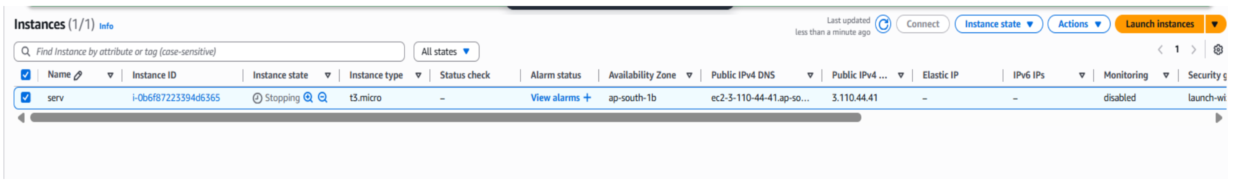
e.g. rate(1 day), cron(0 17 ? * MON-FRI *)

Lambda will add the necessary permissions for Amazon EventBridge (CloudWatch Events) to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

e.) We have successfully added the Trigger.

5. Verify the Lambda Function & EventBridge Rule

1. Now it's **18:14** UTC, refresh the EC2 Console and you can EC2 Instance has Stopped Automatically.


Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

Instances (1/1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security g
serv	i-0b6f87223394d6365	Stopping	t3.micro	–	View alarms +	ap-south-1b	ec2-3-110-44-41.ap-so...	3.110.44.41	–	–	disabled	launch-wi

Thus we have successfully automated the process of Stopping the AWS EC2 Instance at the Specific Time.

Author
Saif-uddin md