Using Lambda Function to Start & Stop the EC2 Instance Using Cloud watch Events.

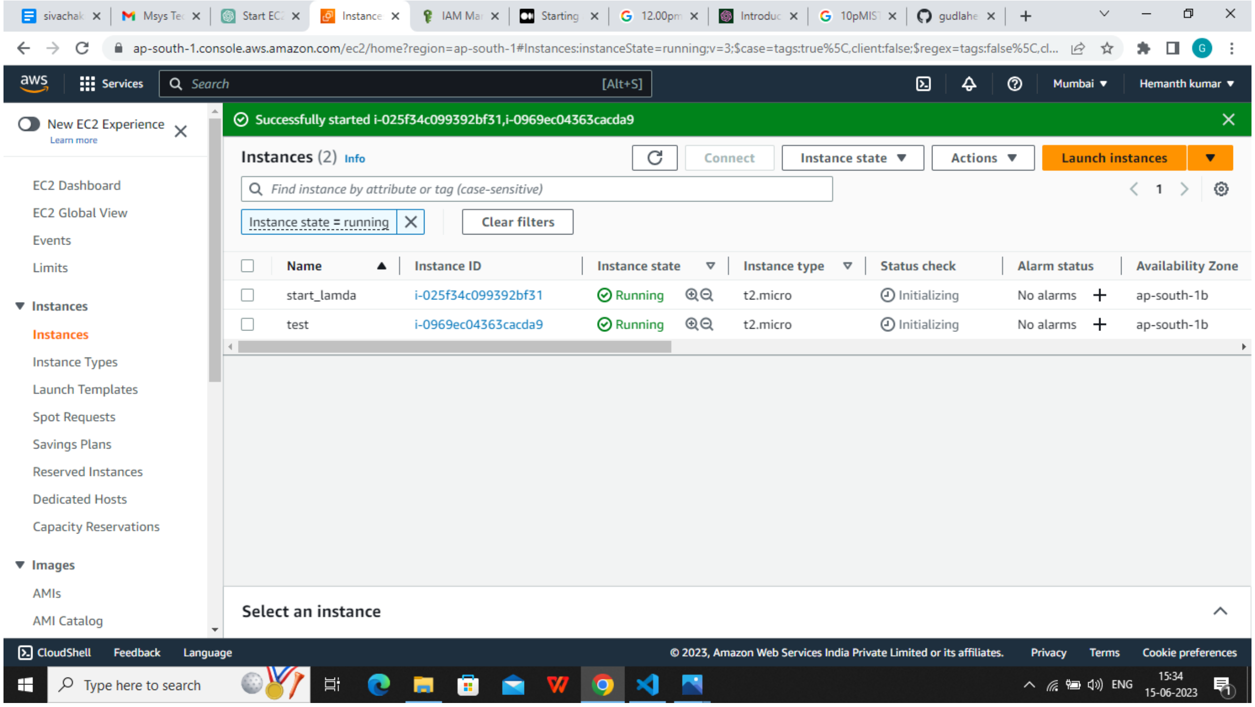
QUESTION

Using Lambda Function to Start & Stop the EC2 Instance Using Cloud watch Events.

- Create a function to start an ec2 instance at 10 AM IST if the instance has tag ec2_start = "true".
- Create a function to stop an ec2 instance at 10 PM IST if the instance has tag ec2_stop = "true".
- Outcome is ec2 instance should start or stop automatically based on cron job schedule.
- Keep the code stuff into the git repository. - Document all the execution steps.

STEP1: create Ec2 Instance

- Create or launch Ec2 Instance by using AWS console



Successfully started i-025f34c099392bf31,i-0969ec04363cacda9

Instances (2)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
start_lamda	i-025f34c099392bf31	Running	t2.micro	Initializing	No alarms	ap-south-1b
test	i-0969ec04363cacda9	Running	t2.micro	Initializing	No alarms	ap-south-1b

STEP2: create IAM policy

IAM policy:

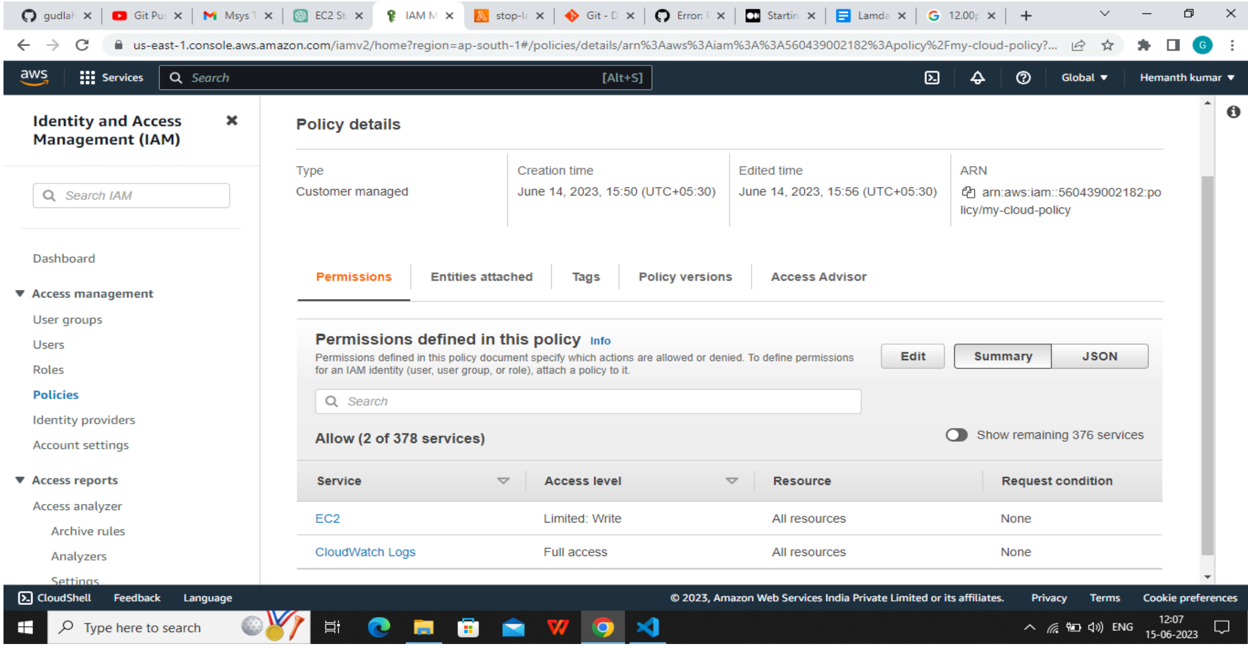
IAM policies in AWS are used to define permissions and access control for users, groups, and roles. Here's an example of an IAM policy expressed in two lines:

1. IAM Policy Name: "ExamplePolicy"
2. **Permissions:** Allow "s3:GetObject" on "arn:aws:s3:::example-bucket/*"

The IAM policy name ExamplePolicy.grants the "s3:GetObject" permission

- Go to IAM in your AWS account and create an IAM policy and role. Give the permissions to Ec2 instance and Cloudwatch which will attach to your lambda function.

□



us-east-1.console.aws.amazon.com/iamv2/home?region=ap-south-1#/policies/details/arn%3Aaws%3Aiam%3A%3A560439002182%3Apolicy%2Fmy-cloud-policy?...

Policy details

Type: Customer managed

Creation time: June 14, 2023, 15:50 (UTC+05:30)

Edited time: June 14, 2023, 15:56 (UTC+05:30)

ARN: arn:aws:iam::560439002182:policy/my-cloud-policy

Permissions defined in this policy

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.

Allow (2 of 378 services)

Show remaining 376 services

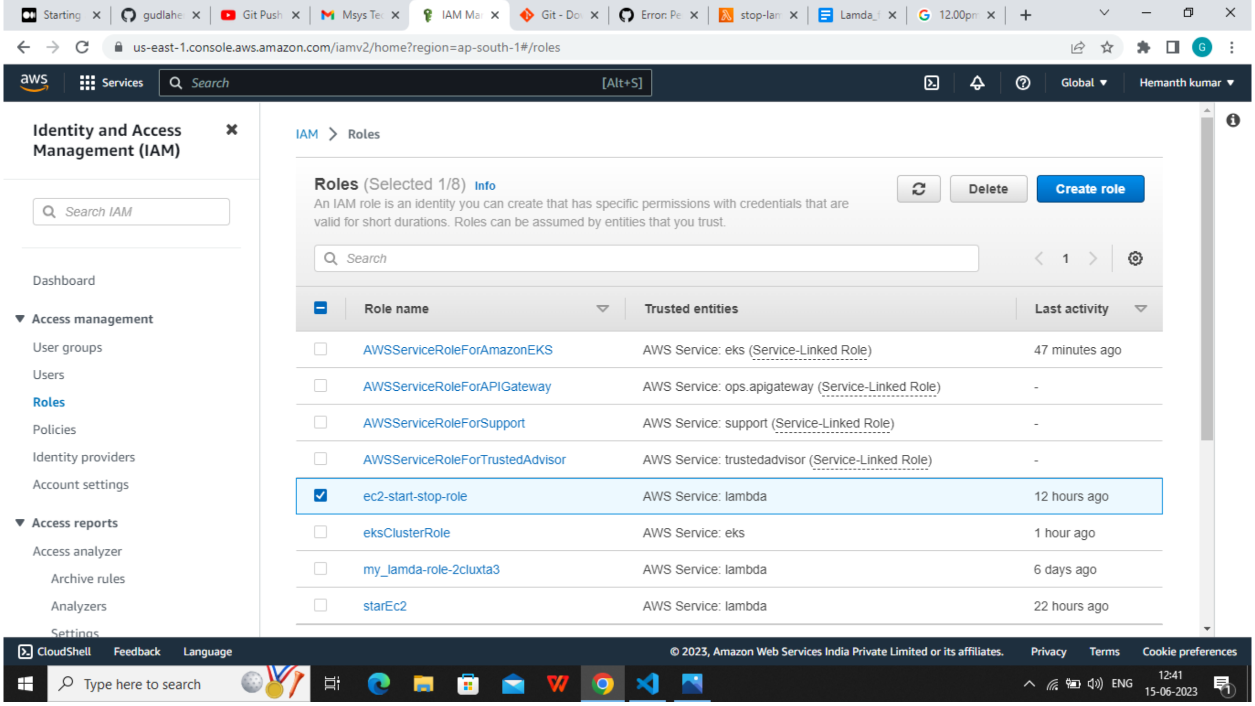
Service	Access level	Resource	Request condition
EC2	Limited: Write	All resources	None
CloudWatch Logs	Full access	All resources	None

© 2023, Amazon Web Services India Private Limited or its affiliates.

- I have created policies and give permissions to Ec2 and cloud watch type

STEP3: create IAM role

- We're now going to create a role and attach the policy we created in Step 2 to this role..

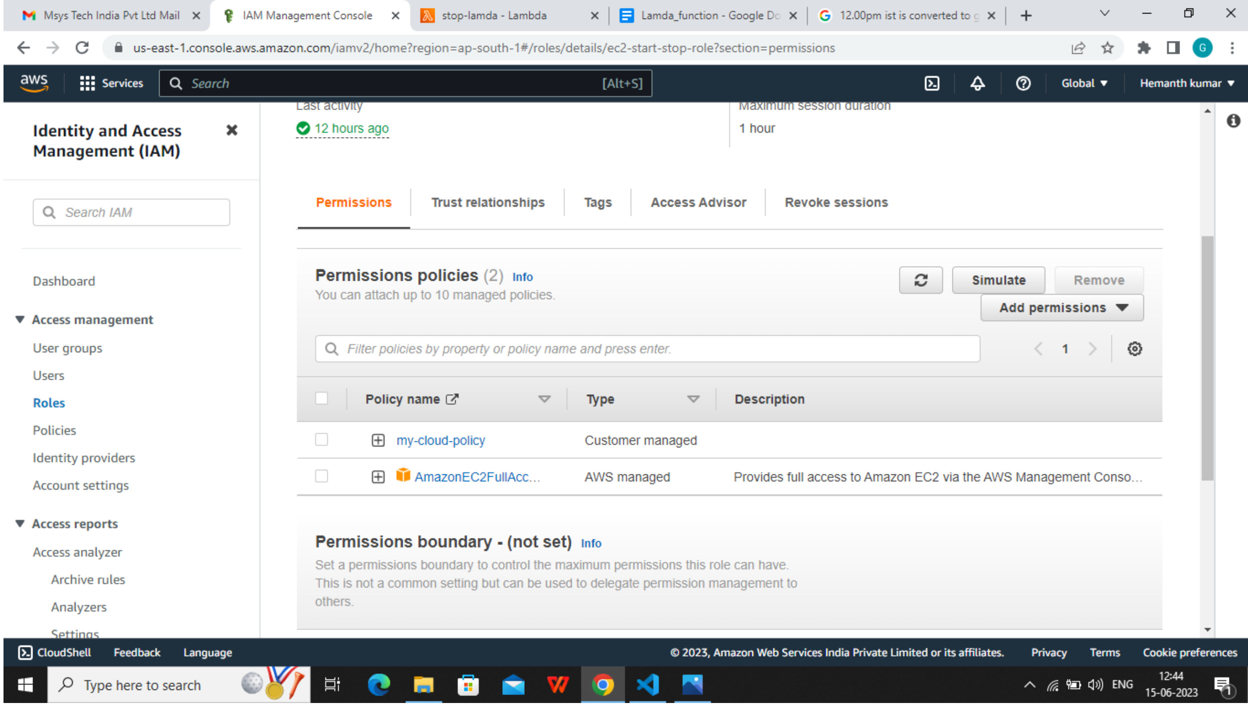


Roles (Selected 1/8) Info

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.

	Role name	Trusted entities	Last activity
□	AWSServiceRoleForAmazonEKS	AWS Service: eks (Service-Linked Role)	47 minutes ago
□	AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Service-Linked Role)	-
□	AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
□	AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
☑	ec2-start-stop-role	AWS Service: lambda	12 hours ago
□	eksClusterRole	AWS Service: eks	1 hour ago
□	my_lamda-role-2cluxta3	AWS Service: lambda	6 days ago
□	starEc2	AWS Service: lambda	22 hours ago

- I have created new role my role name is ec2_start_stop role



Last activity: 12 hours ago

Maximum session duration: 1 hour

Permissions policies (2) Info

You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter.

	Policy name	Type	Description
□	my-cloud-policy	Customer managed	
□	AmazonEC2FullAcc...	AWS managed	Provides full access to Amazon EC2 via the AWS Management Conso...

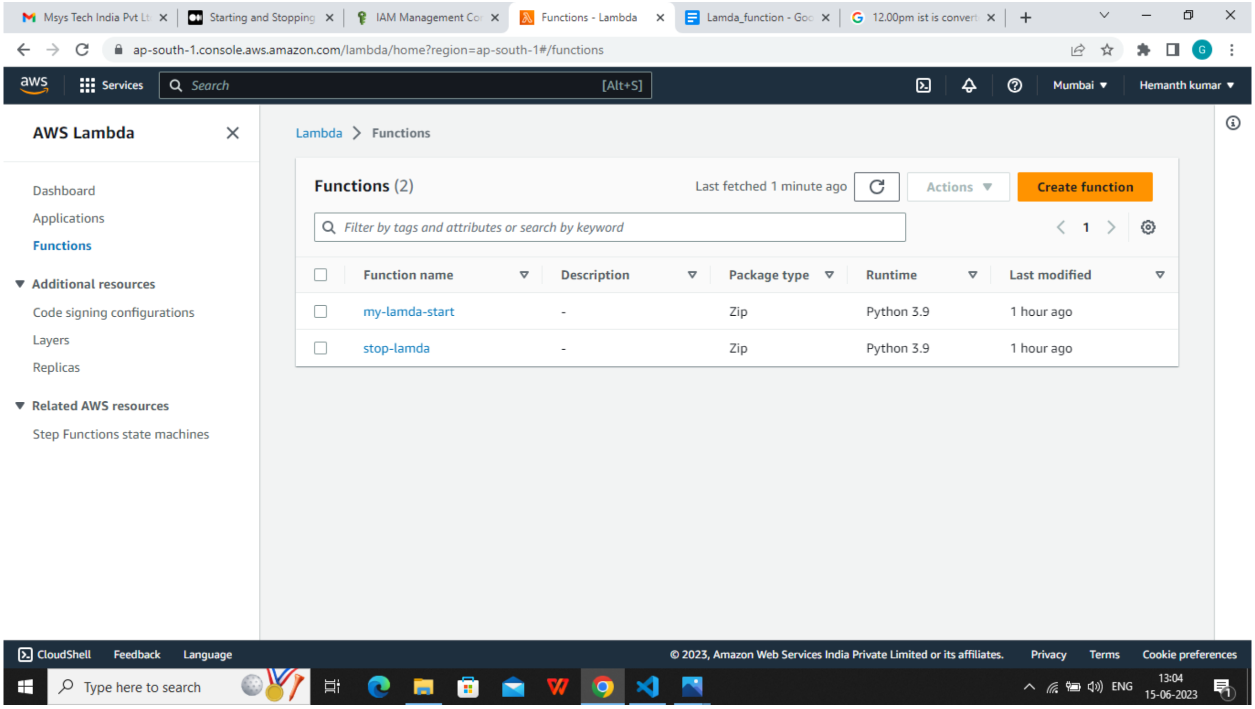
Permissions boundary - (not set) Info

Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting but can be used to delegate permission management to others.

- The above figure is to attach policy the policy name is my_cloud_policy
- Give AmazonEc2FullAccess to amazon role

STEP4: creating lambda functions

- Go to the Lambda function to your AWS console. Create two lambda functions .
- One is for Ec2 start and another one is Ec2 stop



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions

Functions (2) Last fetched 1 minute ago

Function name	Description	Package type	Runtime	Last modified
my-lamda-start	-	Zip	Python 3.9	1 hour ago
stop-lamda	-	Zip	Python 3.9	1 hour ago

- Go to lambda click on create function. In the above diagram two lambda functions are created. One is my_lambda_start another one is stop_lambda
- Go to my_lambda_start function write a code for Ec2 start

➔ **Create a function to start an ec2 instance at 10 AM IST if the instance has tag ec2_start = "true".**

Ec2_start_code:

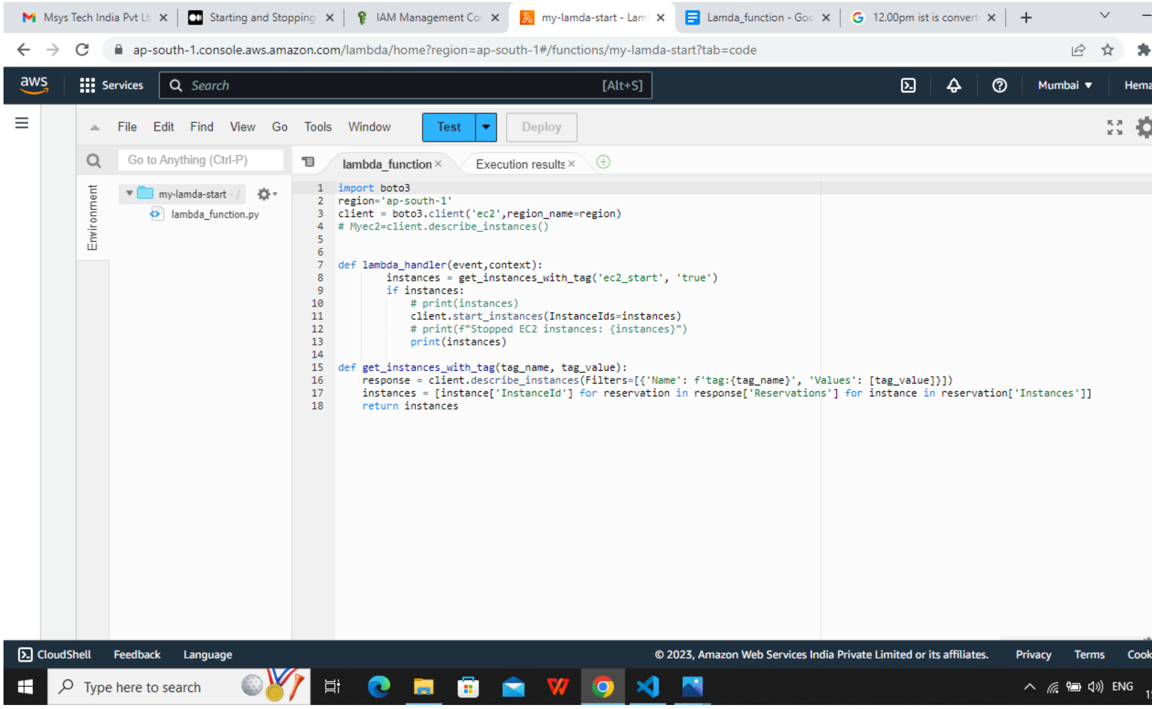
```
import boto3
region='ap-south-1'
```

```
client = boto3.client('ec2',region_name=region)
# Myec2=client.describe_instances()


def lambda_handler(event,context):
        instances = get_instances_with_tag('ec2_start', 'true')
        if instances:
            # print(instances)
            client.start_instances(InstanceIds=instances)
            # print(f"Stopped EC2 instances: {instances}")
            print(instances)

def get_instances_with_tag(tag_name, tag_value):
    response = client.describe_instances(Filters=[{'Name': f'tag:{tag_name}', 'Values': [tag_value]}])
    instances = [instance['InstanceId'] for reservation in response['Reservations'] for instance in reservation['Instances']]
    return instances

```

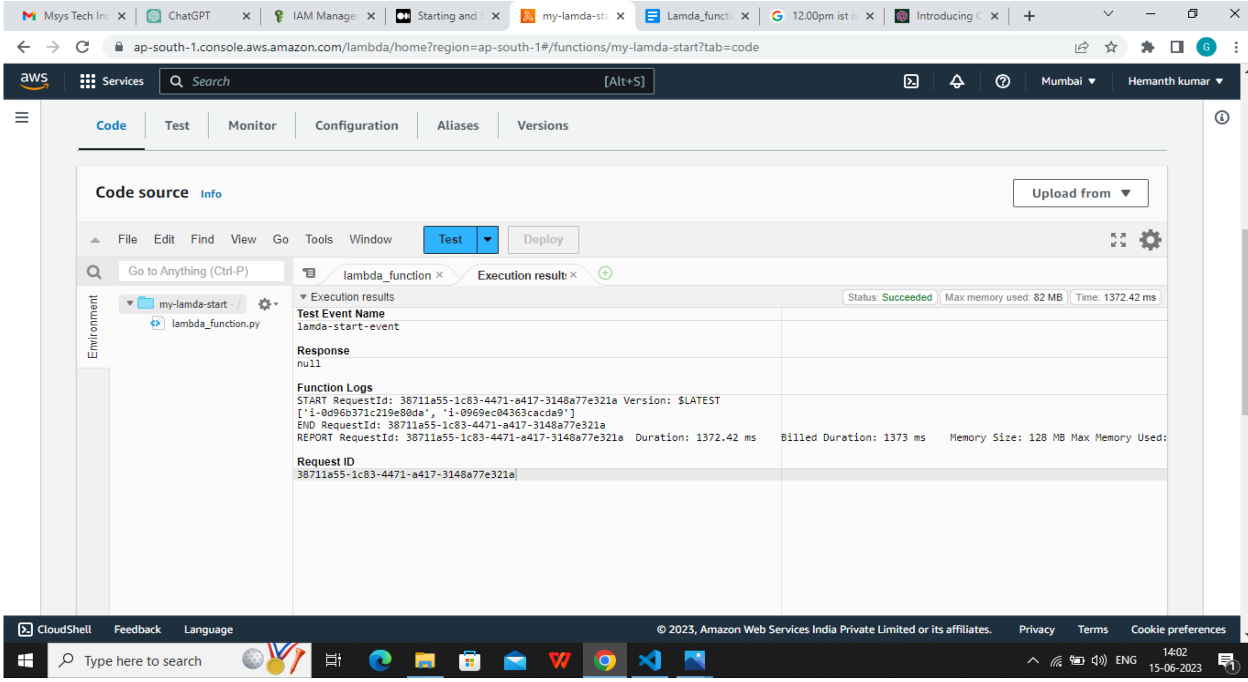


```
import boto3
region='ap-south-1'
client = boto3.client('ec2',region_name=region)
# Myec2=client.describe_instances()


def lambda_handler(event,context):
        instances = get_instances_with_tag('ec2_start', 'true')
        if instances:
           # print(instances)
           client.start_instances(InstanceIds=instances)
           # print(f"Stopped EC2 instances: {instances}")
           print(instances)

def get_instances_with_tag(tag_name, tag_value):
    response = client.describe_instances(Filters=[{'Name': f'tag:{tag_name}', 'Values': [tag_value]}])
    instances = [instance['InstanceId'] for reservation in response['Reservations'] for instance in reservation['Instances']]
    return instances
```

- The code is written in python boto3. So create a file by using **“.py”** .write your python code in your lambda function.
- After completed deploy and test the code.



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/my-lamda-start?tab=code

Execution results — Status: Succeeded | Max memory used: 82 MB | Time: 1372.42 ms

Test Event Name

```
lamda-start-event
```

Response

```
null
```

Function Logs

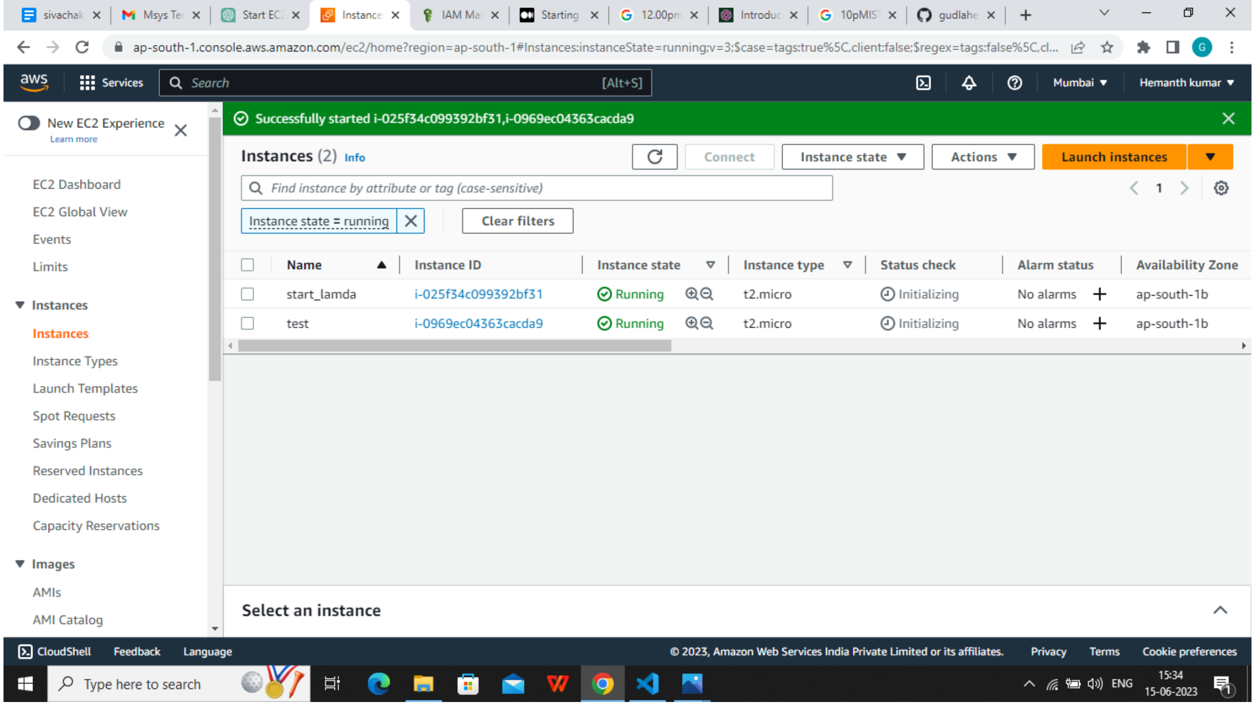
```
START RequestId: 38711a55-1c83-4471-a417-3148a77e321a Version: $LATEST
['i-0d96b371c219e80da', 'i-0969ec04363cacda9']
END RequestId: 38711a55-1c83-4471-a417-3148a77e321a
REPORT RequestId: 38711a55-1c83-4471-a417-3148a77e321a  Duration: 1372.42 ms    Billed Duration: 1373 ms    Memory Size: 128 MB Max Memory Used:
```

Request ID

```
38711a55-1c83-4471-a417-3148a77e321a
```

© 2023, Amazon Web Services India Private Limited or its affiliates.

- By observing the above figure I have created two instances . I have given tag names to my Ec2 instance. My output was successfully executed.so automatically start my ec2 instance.



Successfully started i-025f34c099392bf31,i-0969ec04363cacda9

Instances (2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
start_lamda	i-025f34c099392bf31	Running	t2.micro	Initializing	No alarms	ap-south-1b
test	i-0969ec04363cacda9	Running	t2.micro	Initializing	No alarms	ap-south-1b

➔ Create a function to stop an ec2 instance at 10 PM IST if the instance has tag ec2_stop = "true"

- By using this code the code was stop automatically

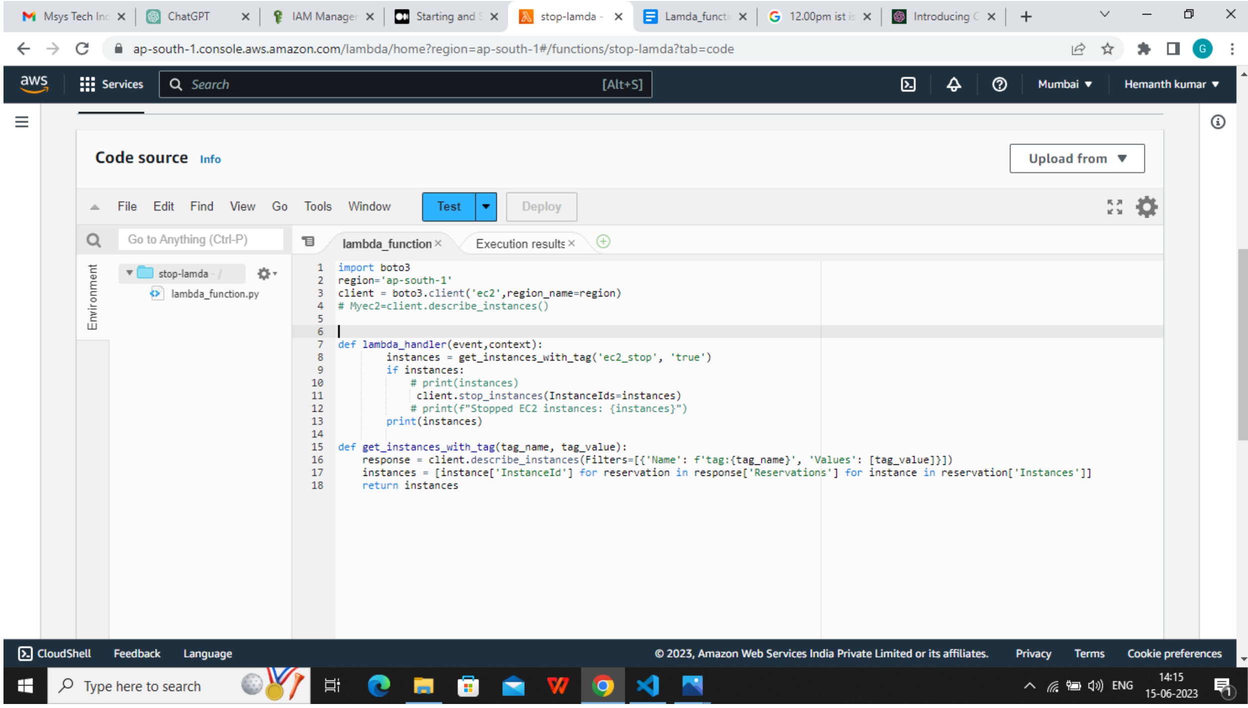
```
import boto3
region='ap-south-1'
client = boto3.client('ec2',region_name=region)
# Myec2=client.describe_instances()



def lambda_handler(event,context):
        instances = get_instances_with_tag('ec2_stop', 'true')
        if instances:
            # print(instances)
             client.stop_instances(InstanceIds=instances)
            # print(f"Stopped EC2 instances: {instances}")
        print(instances)
```

```
def get_instances_with_tag(tag_name, tag_value):
   response = client.describe_instances(Filters=[{'Name': f'tag:{tag_name}', 'Values': [tag_value]}])
   instances = [instance['InstanceId'] for reservation in response['Reservations'] for instance in reservation['Instances']]
   return instances
```

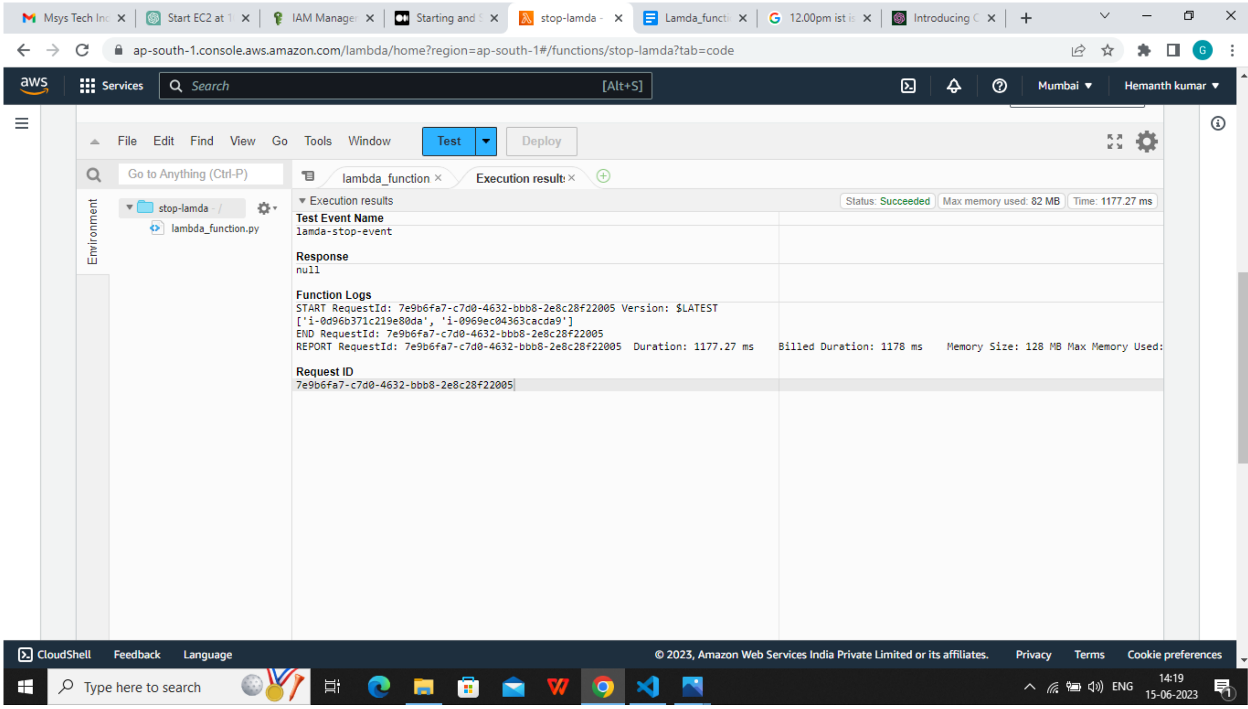
- Write this code to your AWS stop lambda function.
- Deploy and run your code your ec2 instance stops automatically.



```
import boto3
region='ap-south-1'
client = boto3.client('ec2',region_name=region)
# Myec2=client.describe_instances()


def lambda_handler(event,context):
        instances = get_instances_with_tag('ec2_stop', 'true')
        if instances:
            # print(instances)
             client.stop_instances(InstanceIds=instances)
            # print(f"Stopped EC2 instances: {instances}")
        print(instances)

def get_instances_with_tag(tag_name, tag_value):
    response = client.describe_instances(Filters=[{'Name': f'tag:{tag_name}', 'Values': [tag_value]}])
    instances = [instance['InstanceId'] for reservation in response['Reservations'] for instance in reservation['Instances']]
    return instances
```



Execution results

Status: Succeeded | Max memory used: 82 MB | Time: 1177.27 ms

Test Event Name

```
lamda-stop-event
```

Response

```
null
```

Function Logs

```
START RequestId: 7e9b6fa7-c7d0-4632-bbb8-2e8c28f22005 Version: $LATEST
['i-0d96b371c219e80da', 'i-0969ec04363cacda9']
END RequestId: 7e9b6fa7-c7d0-4632-bbb8-2e8c28f22005
REPORT RequestId: 7e9b6fa7-c7d0-4632-bbb8-2e8c28f22005  Duration: 1177.27 ms    Billed Duration: 1178 ms    Memory Size: 128 MB Max Memory Used:
```

Request ID

```
7e9b6fa7-c7d0-4632-bbb8-2e8c28f22005
```

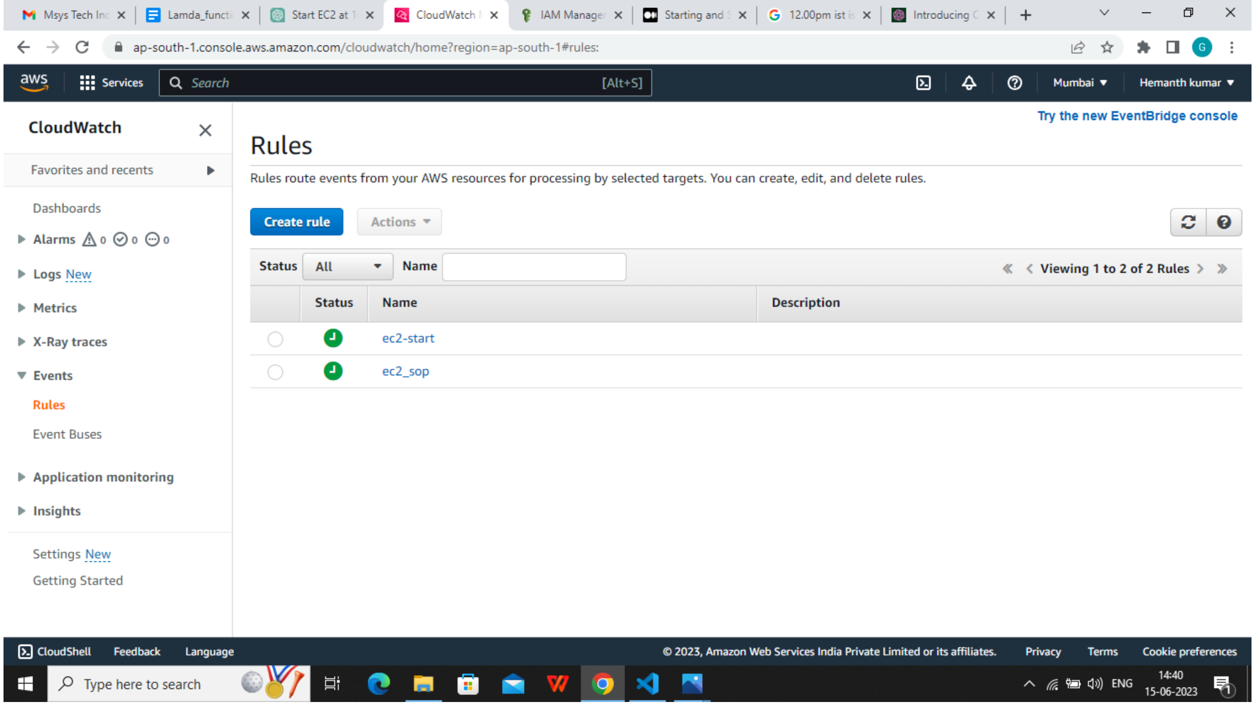
- Above my code was executed successfully. After running my code automatically stop Ec2 instance.

<u>STEP5:</u> AWS cloud watch

<u>AWS cloud watch:</u>

AWS cloud watch is an amazon web service collects and tracks matrics,logs, and events from your aws resources and applications enabling monitoring, troubleshooting and decision making.

- Go to cloud watch in AWS console management. Create two rules one is for starting Ec2 and another one is for stopping Ec2.
- Below figure I have created two lambda functions

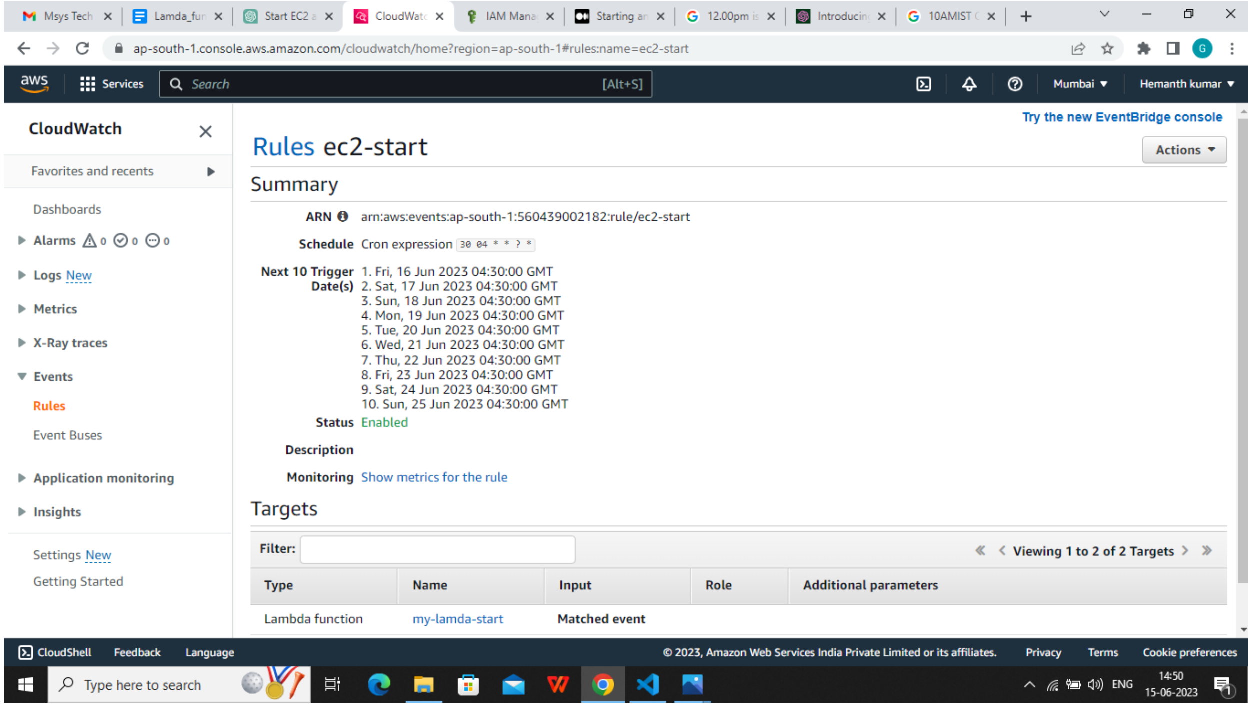


Rules

Rules route events from your AWS resources for processing by selected targets. You can create, edit, and delete rules.

	Status	Name	Description
○	●	ec2-start	
○	●	ec2_sop	

- Let's go to your Ec2 start lambda. Select cron job expressions.
- Cronjob follows GMT to time .Your scheduled time is 10AM IST .Convert 10AM IST to GMT. In GMT my time was 4.40AM scheduled 4.30AM to my cron job.



Rules ec2-start

Summary

ARN arn:aws:events:ap-south-1:560439002182:rule/ec2-start

Schedule Cron expression `30 04 * * ? *`

Next 10 Trigger Date(s)
1. Fri, 16 Jun 2023 04:30:00 GMT
2. Sat, 17 Jun 2023 04:30:00 GMT
3. Sun, 18 Jun 2023 04:30:00 GMT
4. Mon, 19 Jun 2023 04:30:00 GMT
5. Tue, 20 Jun 2023 04:30:00 GMT
6. Wed, 21 Jun 2023 04:30:00 GMT
7. Thu, 22 Jun 2023 04:30:00 GMT
8. Fri, 23 Jun 2023 04:30:00 GMT
9. Sat, 24 Jun 2023 04:30:00 GMT
10. Sun, 25 Jun 2023 04:30:00 GMT

Status Enabled

Description

Monitoring Show metrics for the rule

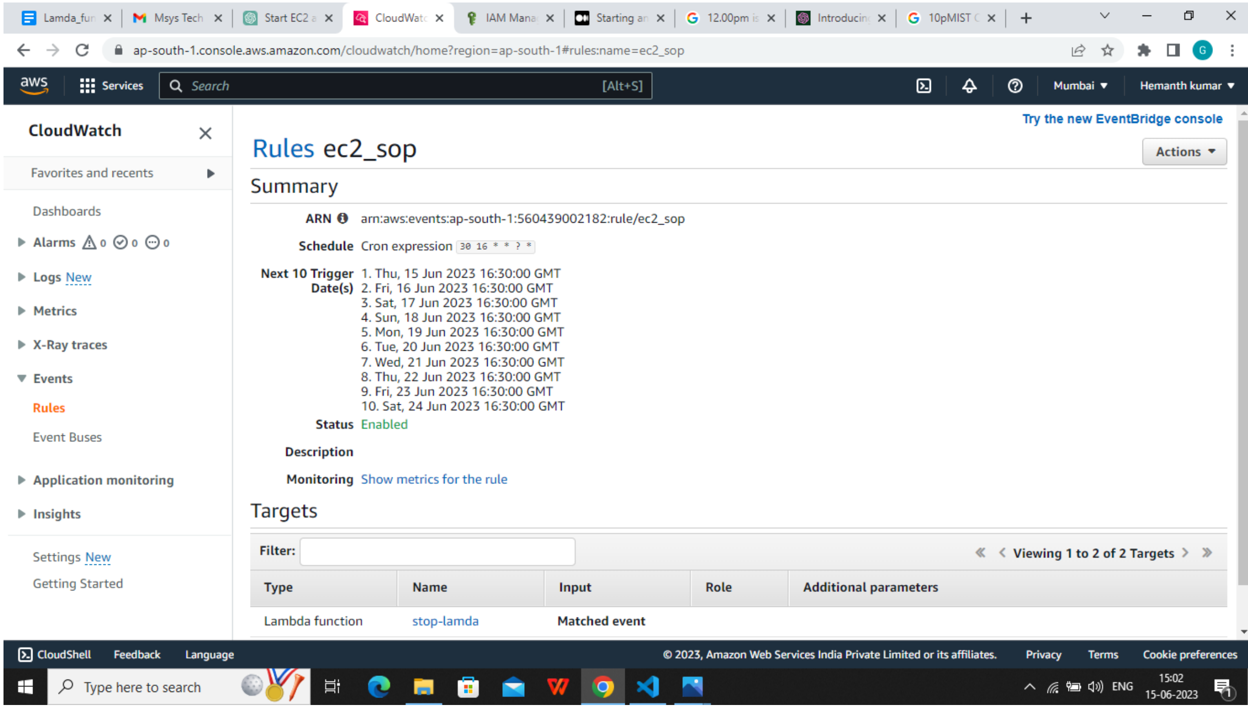
Targets

Filter: Viewing 1 to 2 of 2 Targets

Type	Name	Input	Role	Additional parameters
Lambda function	my-lamda-start	Matched event		

- By using cron job schedule my Ec2 instance was start automatically at 10AM

  <u>Stop Ec2 by using Ec2 cloud watch</u>
- Let's go to your Ec2_stop lambda function. Go to your cron job schedule.
- Schedule time



Rules ec2_sop

Summary

ARN arn:aws:events:ap-south-1:560439002182:rule/ec2_sop

Schedule Cron expression `30 16 * * ? *`

Next 10 Trigger Date(s)
1. Thu, 15 Jun 2023 16:30:00 GMT
2. Fri, 16 Jun 2023 16:30:00 GMT
3. Sat, 17 Jun 2023 16:30:00 GMT
4. Sun, 18 Jun 2023 16:30:00 GMT
5. Mon, 19 Jun 2023 16:30:00 GMT
6. Tue, 20 Jun 2023 16:30:00 GMT
7. Wed, 21 Jun 2023 16:30:00 GMT
8. Thu, 22 Jun 2023 16:30:00 GMT
9. Fri, 23 Jun 2023 16:30:00 GMT
10. Sat, 24 Jun 2023 16:30:00 GMT

Status Enabled

Description

Monitoring Show metrics for the rule

Targets

Filter:

Viewing 1 to 2 of 2 Targets

Type	Name	Input	Role	Additional parameters
Lambda function	stop-lamda	Matched event		

- In the above figure I have scheduled EC2 to stop at 10PM by using cron job schedule.

What is a cron job ?

A cron job is a time-based job scheduler in Unix-like operating systems. It allows users to schedule and automate the execution of commands or scripts at specified intervals, referred to as cron expressions.

```
* * * * * command
| | | | |
| | | | ----- Day of the week (0 - 7) (Sunday = 0 or 7)
| | | ------- Month (1 - 12)
| | --------- Day of the month (1 - 31)
| ----------- Hour (0 - 23)
------------- Minute (0 - 59)
```

Git:

My code successfully pushed to my git repository by using these commands

- git init
- git add .
- git status

- git commit -m “message”
- git remote add origin “https://github.com/gudlahemanth/lamda_start_stop.git”
- git remote -v
- git branch
- git push -v origin master