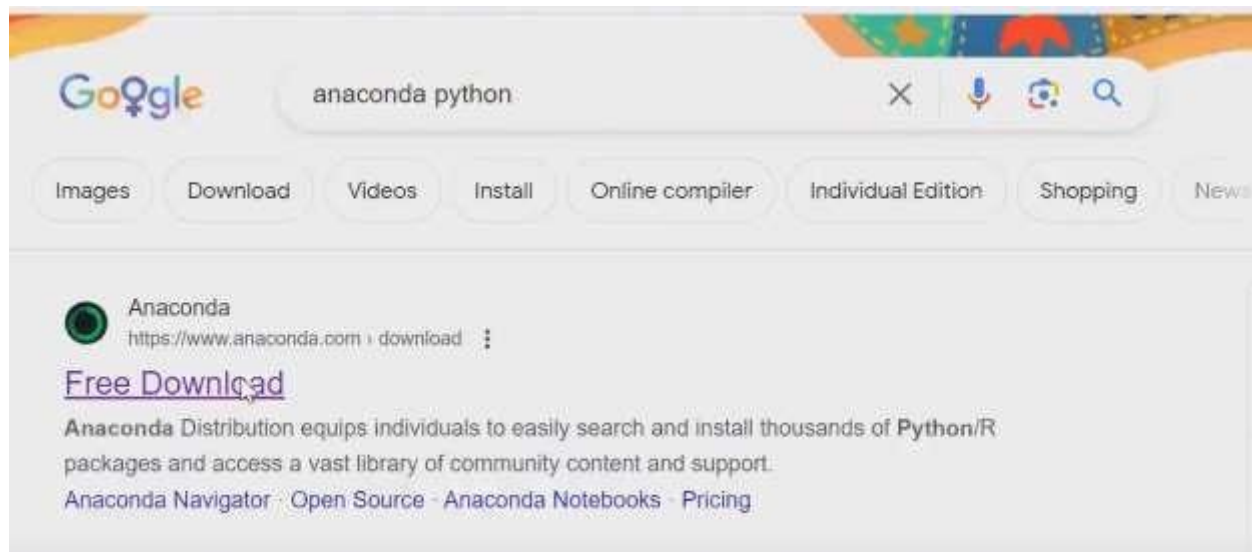


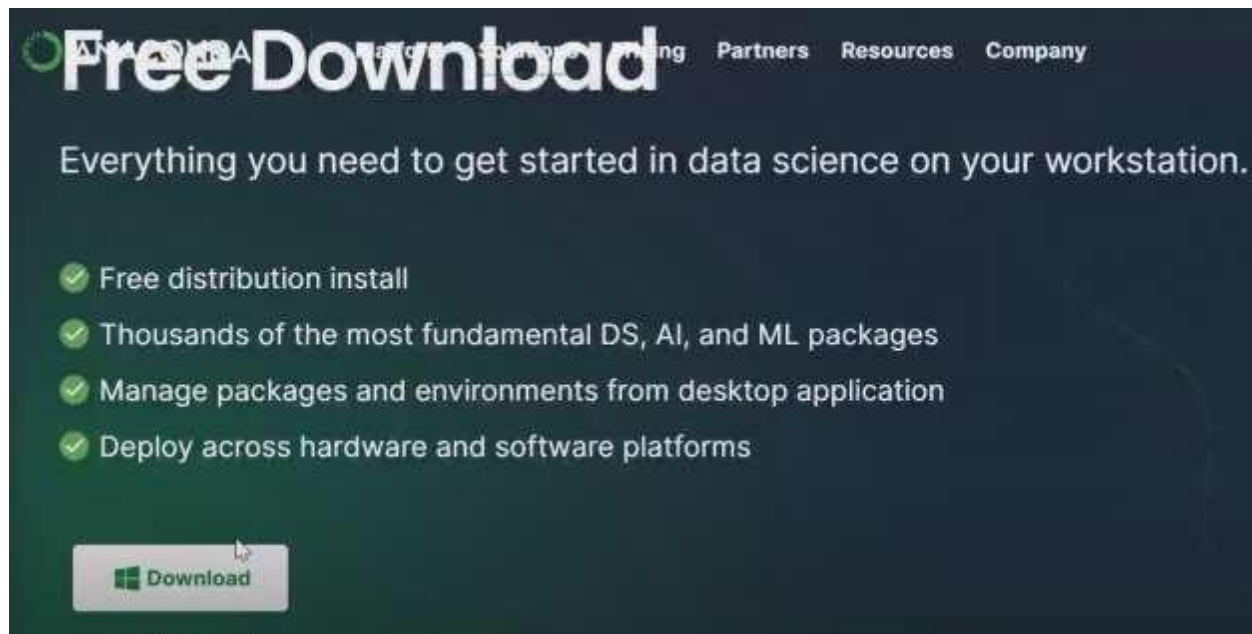


AWS Training Session N0. 5

Summary [08-03-2024]

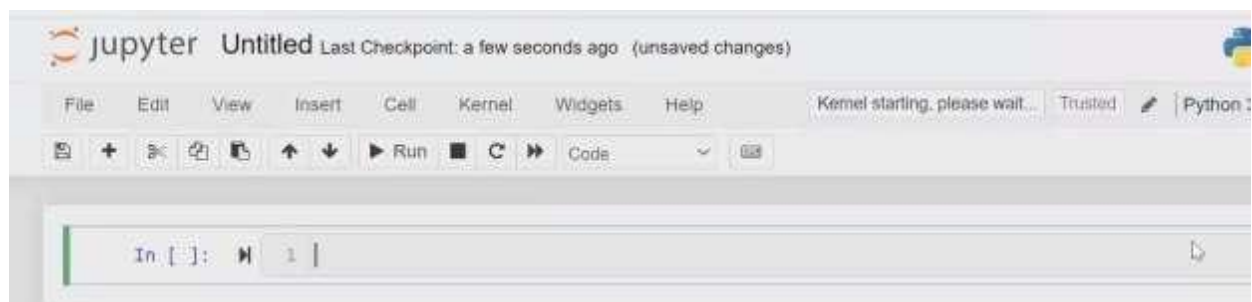
- **Demonstration** – How to launch an ec2 instance (or use some other services like lambda etc) without logging into the account or without login and password.
- There are three ways to interact with the cloud
 - WebUI – it is a graphic interface, simple to use but gives a manual way.
 - CLI – command line interface
 - API – mostly used in the corporate world, it is an automated way. Automation works on the programs(code).
- The code can be written by some developer or by Generative AI
- Generative AI + Automation(DevOps) = **GenAIOps**
- A programming language (python) interpreter is needed to write and execute the code. Install Python from a program called Anaconda. Anaconda is one kind of distributor of the Python.





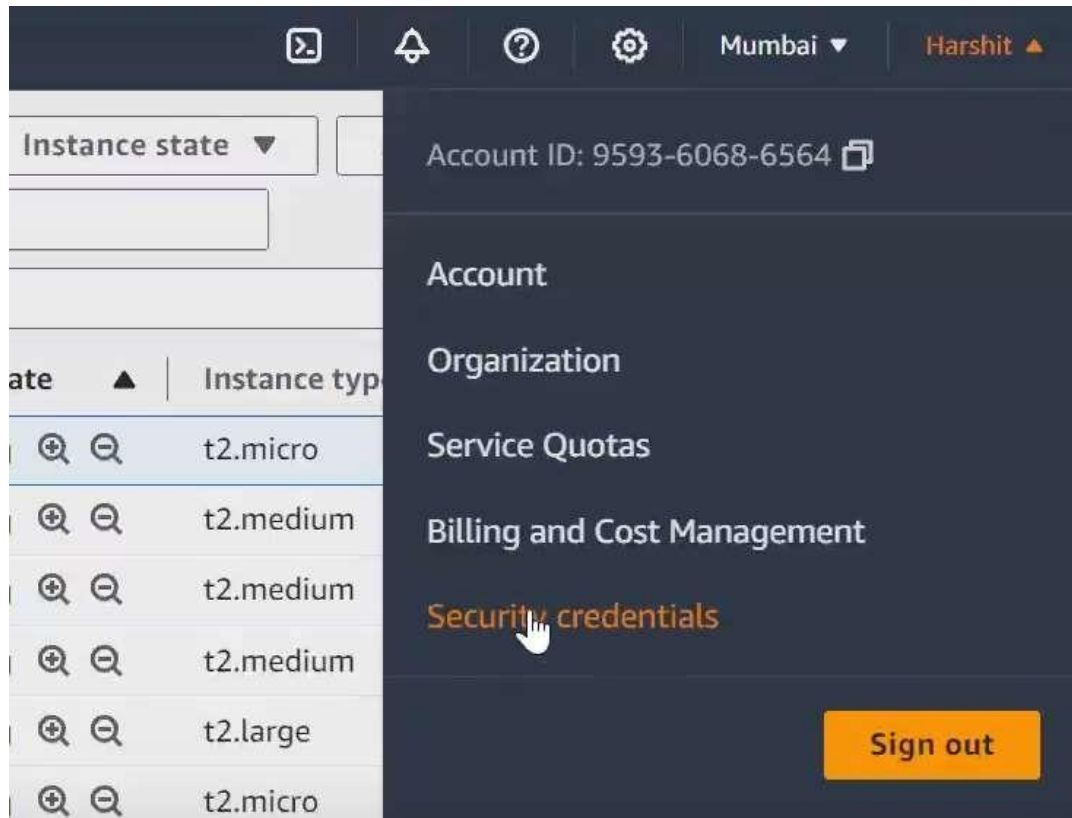
- Python has one IDE called **Jupyter**. So launch the Jupyter.
- When we install Anaconda, this IDE is also installed.
- Open your command prompt
 - **jupyter notebook** – command to launch Jupyter

```
C:\Users\Vimal Daga\Documents\aws_training_2024>jupyter notebook
```



- For human beings to log in, the authentication credentials are **username and password**.
- For the code to login, the authentication credentials are some **keys or tokens**.
- Here we are going inside through the code so we need keys to authentication for the same AWS account. So go on your AWS account.

- Click on your 'profile name' then click on 'Security credentials'



- Click on 'create access key'



- Accept the agreement and click on the 'create access key'

Alternatives to root user access keys Info



Root user access keys are not recommended

We don't recommend that you create root user access keys. Because you can't specify the root user in a permissions policy, you can't limit its permissions, which is a best practice.

Instead, use alternatives such as an IAM role or a user in IAM Identity Center, which provide temporary rather than long-term credentials. [Learn More](#)

If your use case requires an access key, create an IAM user with an access key and apply least privilege permissions for that user. [Learn More](#)

Continue to create access key?

☒ I understand creating a root access key is not a best practice, but I still want to create one.

Cancel

Create access key

- Two keys are generated.
 1. **Access key** – works as username
 2. **Secret key** – works as a password

(recommended – as the keys are generated copy these keys and keep them saved with you)

Retrieve access key Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key

 AKIA56XS2LHSAO3D2RSA

Secret access key

 [Show](#)

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

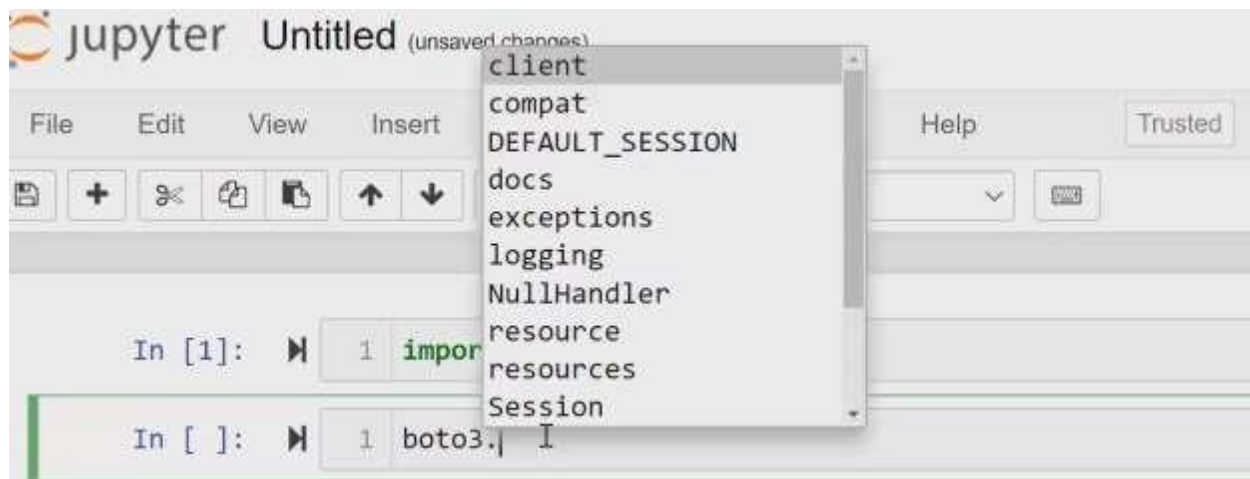
- **Boto3** – Library or SDK is used to make Python understand how to interact with AWS.
- **pip install boto3** – command to install boto3. Pip command is provided by Anaconda itself.

```
C:\Users\Vimal Daga>pip install boto3
```

- The **'import'** keyword is used to load the library in the code. Inside the code these libraries are called modules.

```
In [1]: 1 import boto3
```

- Modules have a lot of functionalities to see these, write the library name, dot(.), and click the double tab.



- The variable **'myec2'** is referring all the features of the service ec2.
- Pass the key credentials as the parameters along with the service name (EC2) and region name to the resource function.

```

]: 1 myec2 = boto3.resource(
    2     service_name="ec2",
    3     aws_access_key_id="AKIA56XS2LHSAO3D2R5A",
    4     aws_secret_access_key="4kC1OKSRUENQg27JdWu1Mr6HxXmUvZBSp",
    5     region_name="ap-south-1"
    6 )

```

- To see the help menu put the cursor inside the parentheses of the function and press **shift + tab**.

```

1 myec2.create_instances( "amazon linux", "t2.micro")

```

you don't specify this address, we choose one from the IPv4 range of your sub...

* Not all instance types support IPv6 addresses. For more information, see 'Instance types' <<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html>>.

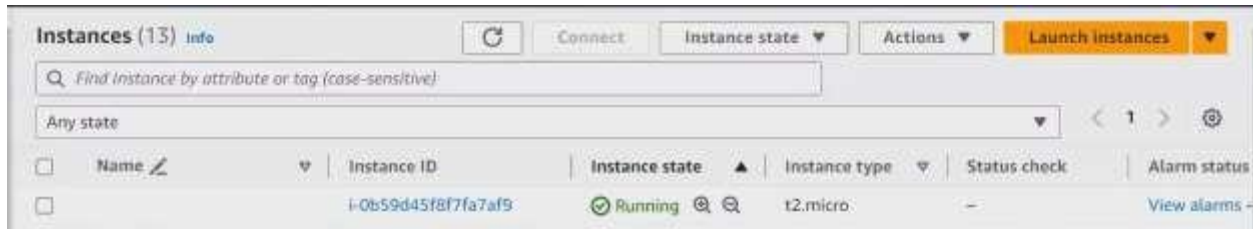
- **Create_instances()** – method to launch the EC2 instance. The minimum required parameters to launch the instance are as follows.
- **ImageId** – Every OS is identified by a unique image ID.
- **InstanceType** – what configuration is required in an instance.
- **MinCount/MaxCount** – number of instances you want to launch.
- Store the output in the variable 'myos'

```

1 myos = myec2.create_instances(
2     ImageId="ami-0ba259e664698cbfc",
3     InstanceType="t2.micro",
4     MinCount=1,
5     MaxCount=1
6 )

```

Out[5]: [ec2.Instance(id='i-0b59d45f8f7fa7af9')]



- Myos shows the array of launched instance names, here the instance name position is zero.

```
1 myos
I
[0]: [ec2.Instance(id='i-00486b4dd55ab2383')]
```

- So, we will use this with position number which makes it specific.

```
1 myos[0]
]: ec2.Instance(id='i-00486b4dd55ab2383')
```

- We can retrieve the information about the instance using the variable 'myos'.

```
1 myos[0].id
2]: 'i-00486b4dd55ab2383'
```

- **Resource()** method does not have a way to delete the instance
- **Client()** method does have the functionality to delete the method. We can also use the client method to launch the instance
- Client() method gives you more facilities at a low level than resource(). You can use them accordingly.

```

: 1 myec22.terminate_instances(
2     InstanceIds=['i-00486b4dd55ab2383']
3 )

[15]: {'TerminatingInstances': [{'CurrentState': {'Code': 32,
        'Name': 'shutting-down'},
        'InstanceId': 'i-00486b4dd55ab2383',
        'PreviousState': {'Code': 16, 'Name': 'running'}}],
        'ResponseMetadata': {'RequestId': '49f54a2c-7d3e-4f97-802e-6b831096e657',
        'HTTPStatusCode': 200,
        'HTTPHeaders': {'x-amzn-requestid': '49f54a2c-7d3e-4f97-802e-6b831096e657'}}}

```

- Now to run this code automatically so that your team member can launch the instance without knowing the credentials for this we will put this code in the AWS Lambda.
- API gateway has the capability to integrate with services like lambda.
- Go to the lambda service



- Click on 'create function'



- Give the function name

Basic information

Function name
Enter a name that describes the purpose of your function.

ec2launchfunc1

Use only letters, numbers, hyphens, or underscores with no spaces.

- Choose the runtime as Python and click on ‘create function’

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 3.12

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

► Advanced settings

Cancel Create function

- First import the boto3 library (import boto3) and put all the code as it is inside the **lambda_handler**.

```
lambda_function x Environment Var x +
1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     # TODO implement
6     myec2 = boto3.resource(
7         service_name="ec2",
8         aws_access_key_id="AKIA56XS2LHSA03D2R5A",
9         aws_secret_access_key="dkC1OKSRUENQg27JdWu1Mr6HxXmUvZBSprVFd0GG",
10        region_name="ap-south-1"
11    )
12
13    myos = myec2.create_instances(
14        ImageId="ami-0ba259e664698cbfc",
15        InstanceType="t2.micro",
16        MinCount=1,
17        MaxCount=1
18    )
19
20    print(myos[0].id)
21
22    return {
23        'statusCode': 200,
24        'body': json.dumps('Hello EC2 Launched ...! ' + myos[0].id)
25    }
```

- Deploy the code by clicking on ‘Deploy’ and test the code



- Give some ‘event name’ and click on ‘Save’

To invoke your function without saving an event, configure the JSON event, then choose Test.

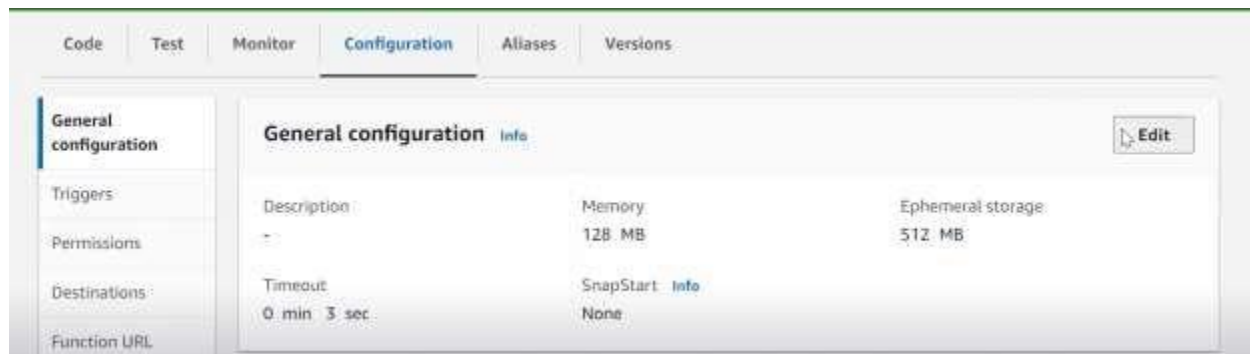
Test event action

☒ Create new event ☐ Edit saved event

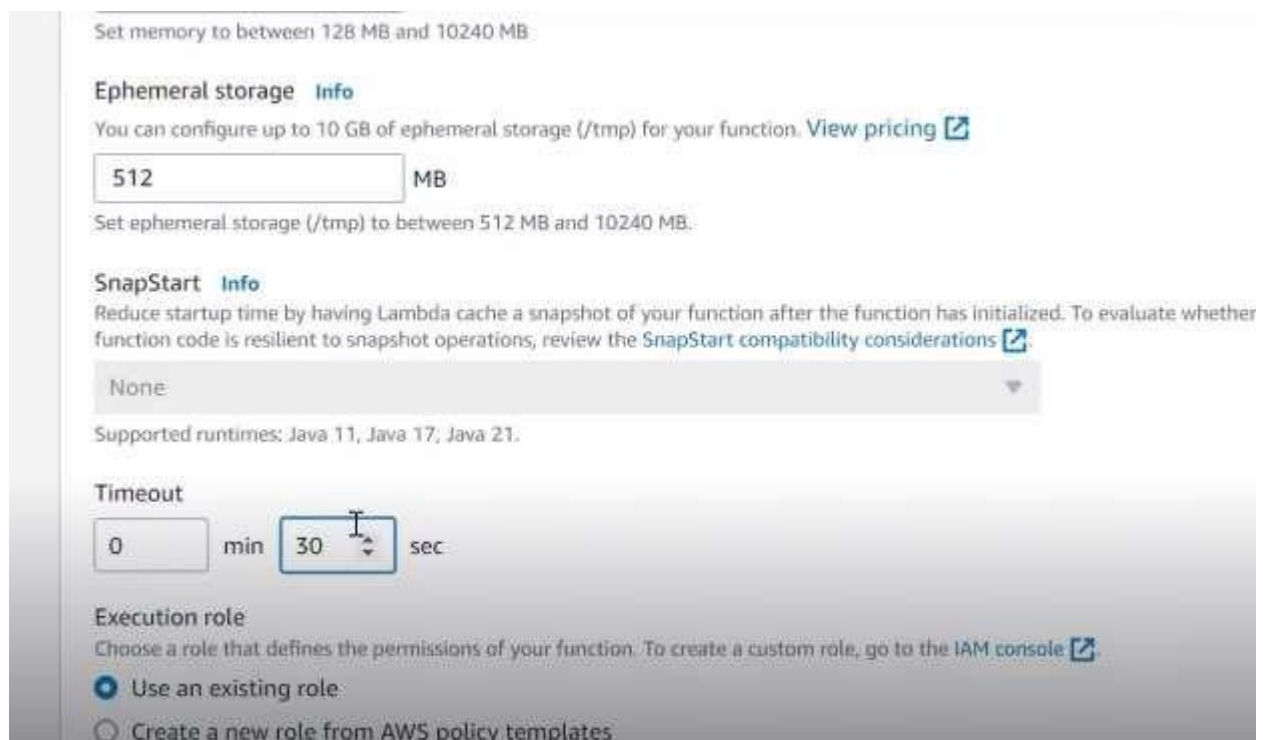
Event name

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

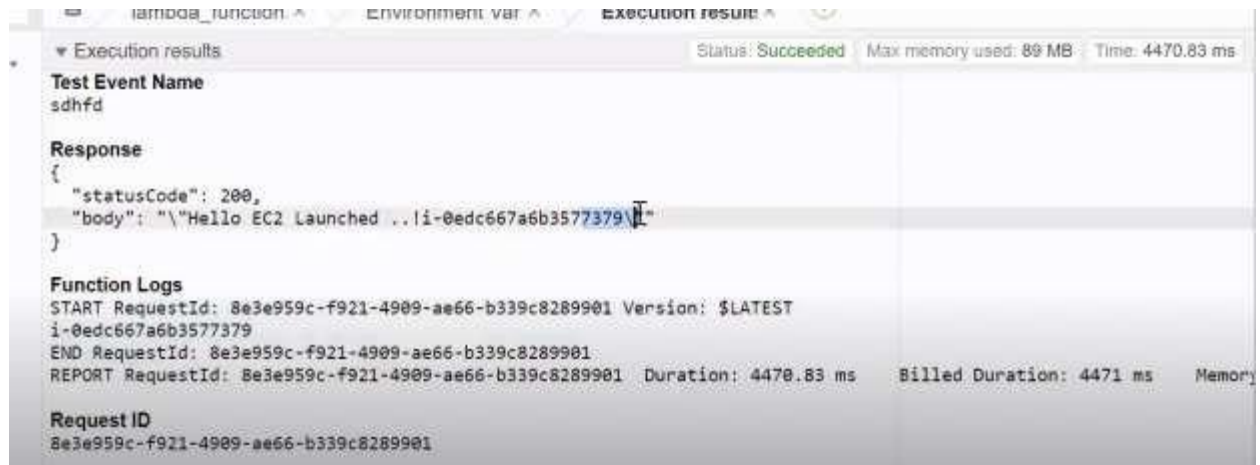
- If the execution time of the lambda function is more than 3 seconds it will be failed.
 - So, if your code is taking more time go to the 'Configuration' setting of lambda.



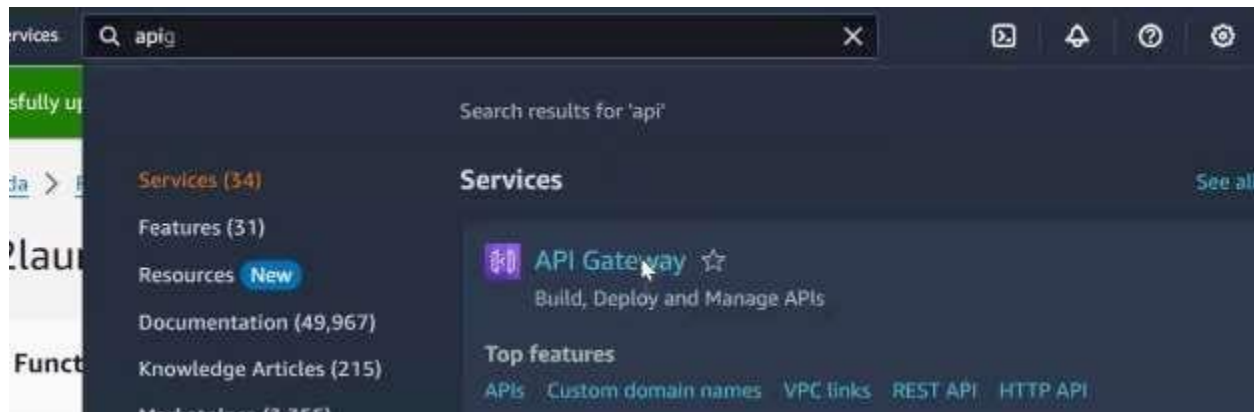
- And increase the timeout by 3.



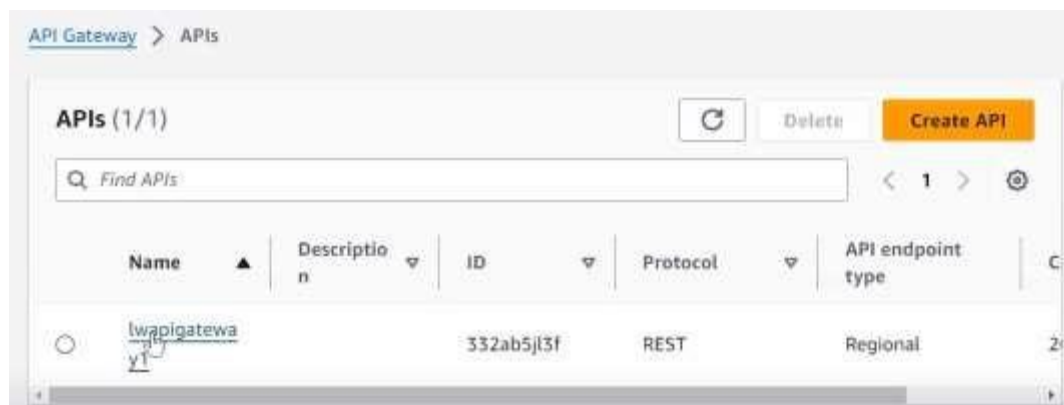
- Code is successfully executed.



- Now, go to API Gateway



- Click on created API or create API if there is not.



- Give the resource name and click 'create resource'

Create resource

Resource details

☒ Proxy resource [Info](#)
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path:

Resource name:

☐ CORS (Cross Origin Resource Sharing) [Info](#)
Create an OPTIONS method that allows all origins, all methods, and several common headers.

Cancel Create resource

- New resource is created

Resources

Create resource

/

/mail

/oslaunch

/search

GET

GET

Resource details

Delete Update documentation

Enable CORS

Path
/oslaunch

Resource ID
6yowic

- Click on 'create method'

/oslaunch

/search

GET

Methods (0) Delete Create method

Method type ▲ Integration type ▼ Authorization ▼ API key ▼

No methods


No methods defined.


- Choose method as 'GET' and service as Lambda function.


Method details

Method type
GET

Integration type

☒ **Lambda function**
Integrate your API with a Lambda function.


☐ **HTTP**
Integrate with an existing HTTP endpoint.


☐ **Mock**
Generate a response based on API Gateway mappings and transformations.


- Choose the lambda function name that you want to integrate. Click on 'create method'

Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.

ap-south-1

Choose a Lambda function or enter its ARN

- arn:aws:lambda:ap-south-1:959360686564:function:lwsearch
- arn:aws:lambda:ap-south-1:959360686564:function:lwfunc1
- arn:aws:lambda:ap-south-1:959360686564:function:lwmail
- arn:aws:lambda:ap-south-1:959360686564:function:ec2launchfunc1

☒ Grant API Gateway policy yourself, I will manage it

☒ Default timeout
The default timeout is 29 seconds.

function's resource

- One URL is created

Create resource

/

/mail

GET

/oslaunch

GET

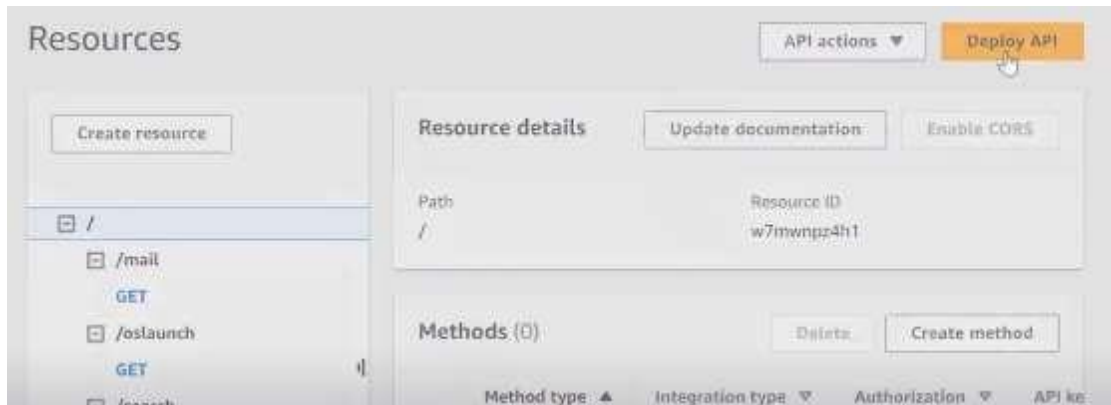
Update documentation

ARN

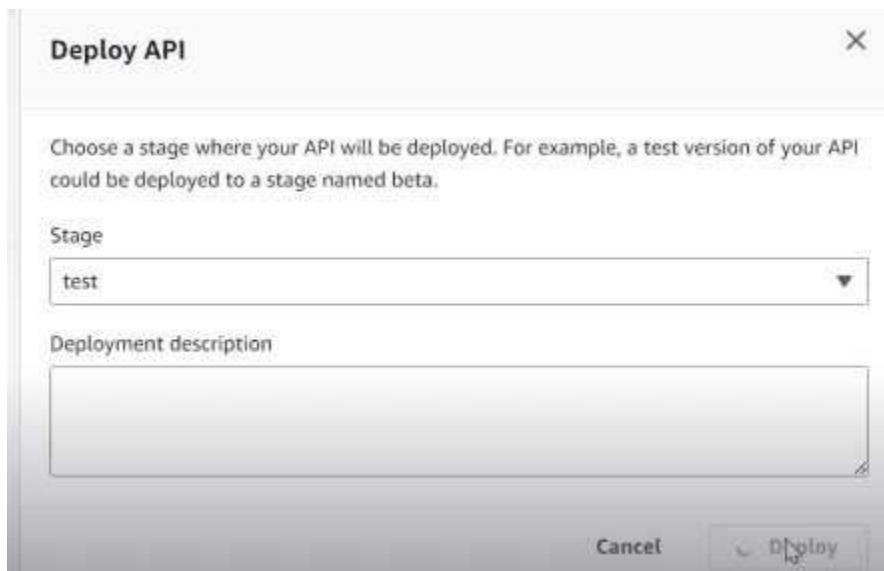
arn:aws:execute-api:ap-south-1:959360686564:332ab5jl3f/*/*launch

→ Me req

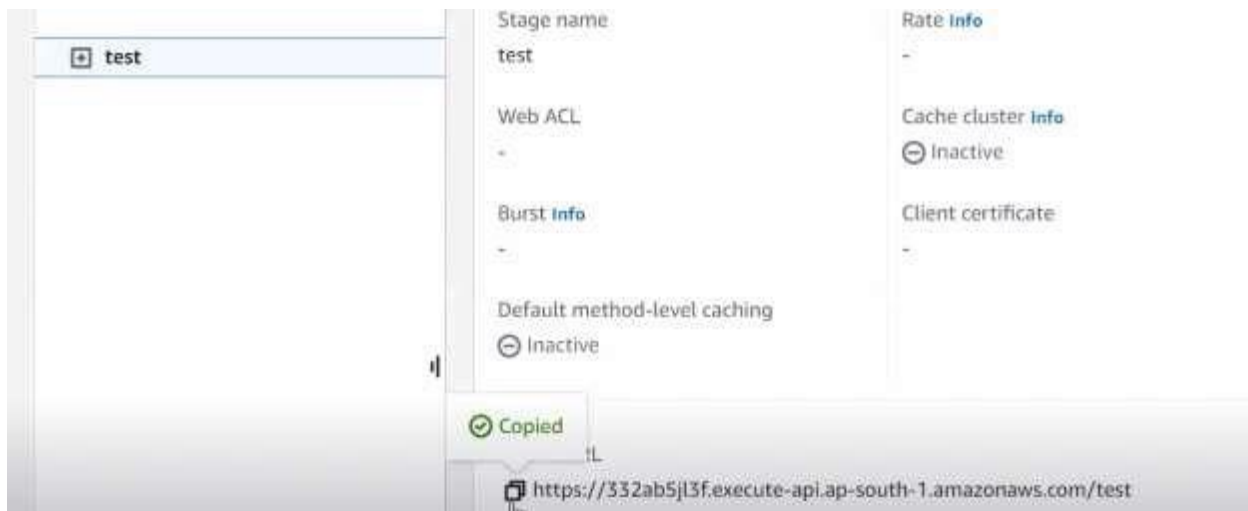
- Click on 'Deploy API' to redeploy the API



- Click on 'Deploy'



- Copy the 'Invoke URL'



- Give this URL to your team member.
- One thing the URL user needs to do is to add /(endpoint_name) at the last as follows.



➤ Instance is launched.



- Anyone else is capable of using the service inside your account without knowing the credentials, the concept is known as **Self-Serving**.