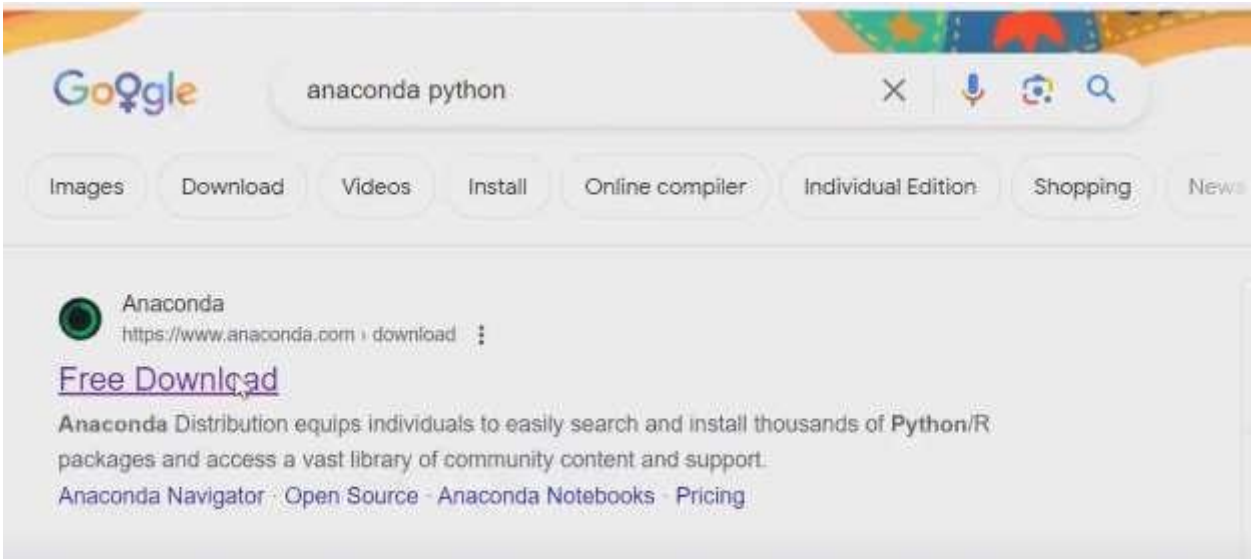
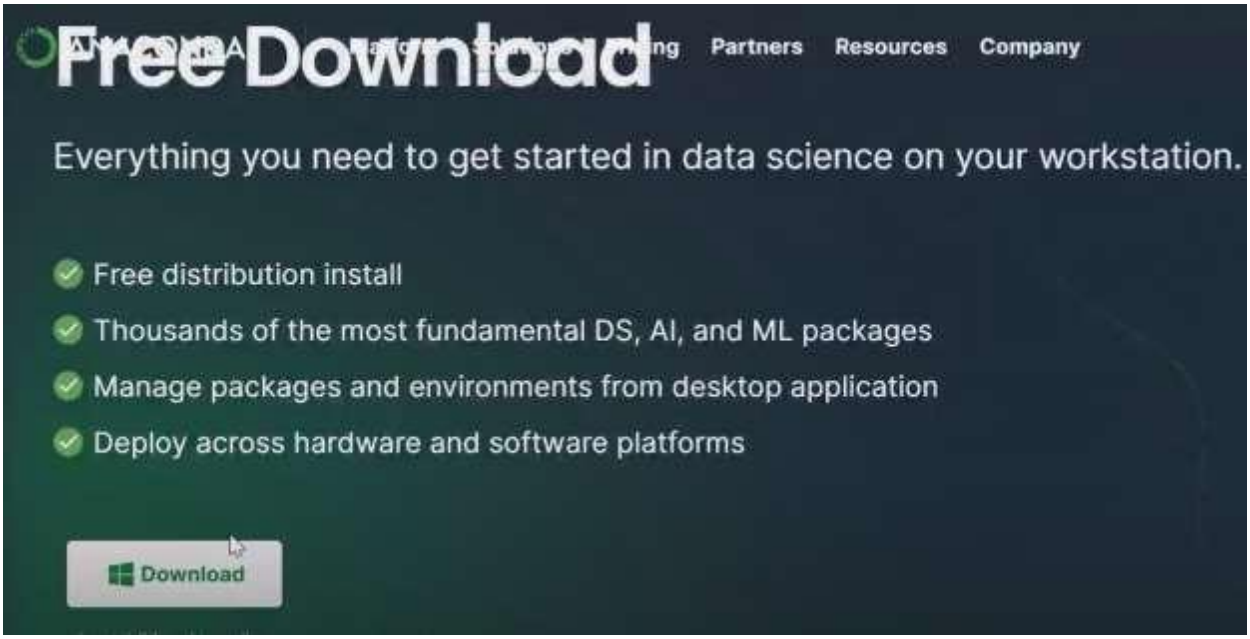
AWS Training Session N0. 5

Summary [08-03-2024]

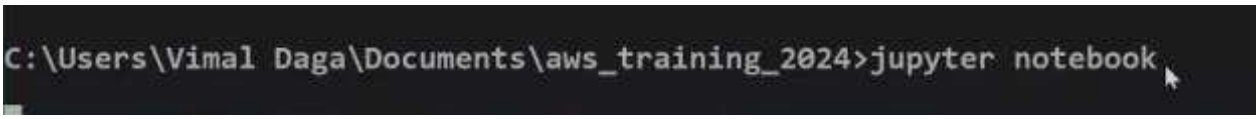
- **Demonstration** – How to launch an ec2 instance (or use some other services like lambda etc) without logging into the account or without login and password.
- There are three ways to interact with the cloud
  - WebUI – it is a graphic interface, simple to use but gives a manual way.
  - CLI – command 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**

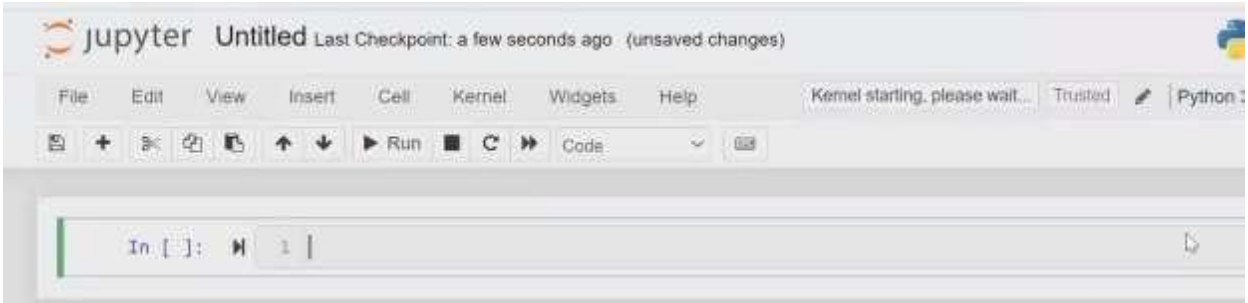
- A programming language (python) interpreter is needed to write and execute the code. Install Python from a program called Anaconda. Anaconda is one kind of distributor of the Python.





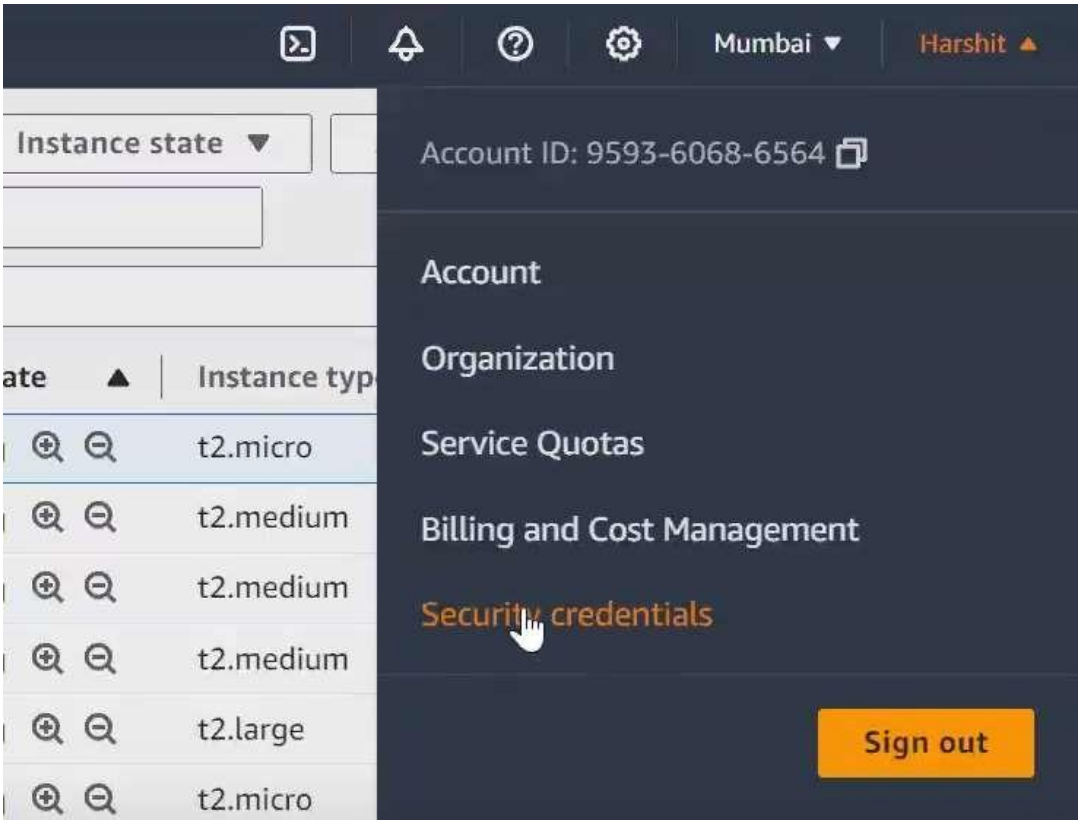
- Python has one IDE called **Jupyter**. So launch the Jupyter.
- When we install Anaconda, this IDE is also installed.
- Open your command prompt
    - **jupyter notebook** – command to launch Jupyter





- 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’

- 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)

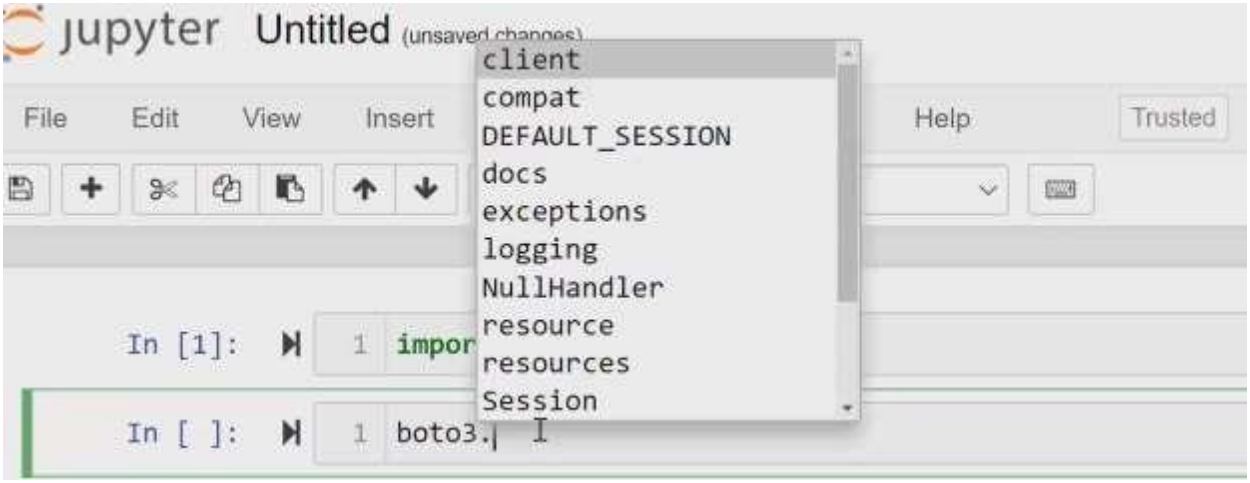
- **Boto3** – Library or SDK is used to make Python understand how to interact with AWS.
- **pip install boto3** – command to install boto3. Pip command is provided by Anaconda itself.

```
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.

```
]:
myec2 = boto3.resource(
                service_name="ec2",
                aws_access_key_id="AKIA56XS2LHSAO3D2R5A",
                aws_secret_access_key="dkC1OKSRUENQg27JdWulMr6HxXmUvZB5p
                region_name="ap-south-1"
        )
```

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

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

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

* Not all instance types support IPv6 addresses. For more information, see `Insta
nce types <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.htm
l>`__.
```

- **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’

```
myos = myec2.create_instances(
    ImageId="ami-0ba259e664698cbfc",
    InstanceType="t2.micro",
    MinCount=1,
    MaxCount=1
)
```

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

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

```
1 myos
[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
'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.

```
myec22.terminate_instances(
    InstanceIds=['i-00486b4dd55ab2383']
)
```

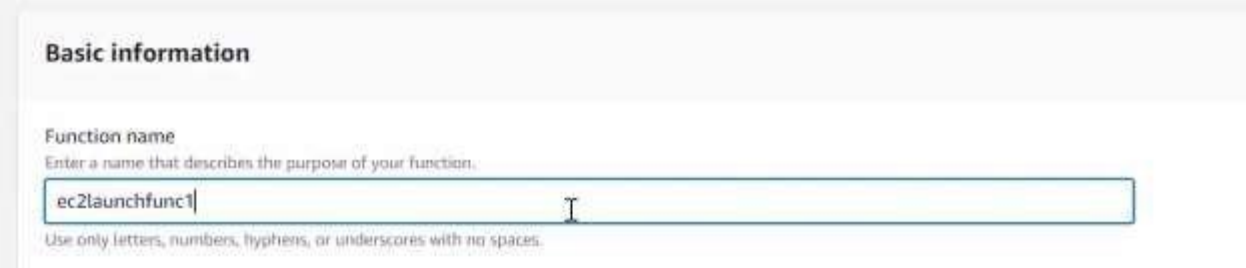
```
[15]: {'TerminatingInstances': [{'CurrentState': {'Code': 32,
        'Name': 'shutting-down'},
       'InstanceId': 'i-00486b4dd55ab2383',
       'PreviousState': {'Code': 16, 'Name': 'running'}}],
      'ResponseMetadata': {'RequestId': '49f54a2c-7d3e-4f97-802e-6b831096e657',
       'HTTPStatusCode': 200,
       'HTTPHeaders': {'x-amzn-requestid': '49f54a2c-7d3e-4f97-802e-6b831096e65
```

- 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.

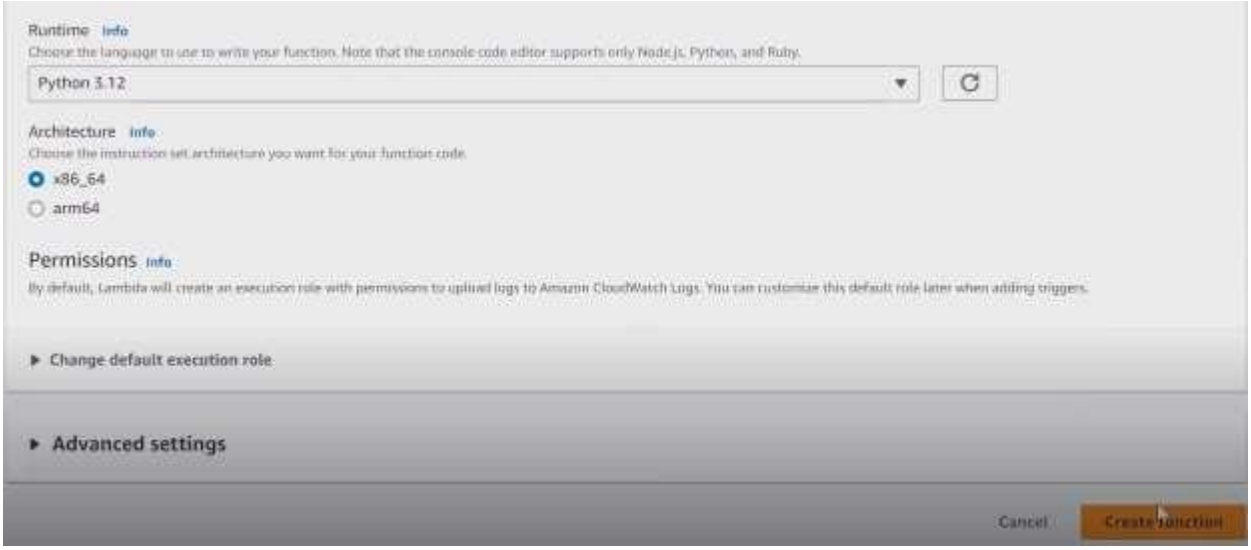
- Go to the lambda service

- Click on ‘create function’

- Give the function name



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



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

lambda_function × Environment Var ×

```
import json
import boto3

def lambda_handler(event, context):
    # TODO implement
    myec2 = boto3.resource(
                service_name="ec2",
                aws_access_key_id="AKIA56XS2LHSAO3D2R5A",
                aws_secret_access_key="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",
                region_name="ap-south-1"
        )

    myos = myec2.create_instances(
     ImageId="ami-0ba259e664698cbfc",
        InstanceType="t2.micro",
        MinCount=1,
        MaxCount=1
    )
```

```
print(myos[0].id)

return {
    'statusCode': 200,
    'body': json.dumps('Hello EC2 Launched ..!' +  myos[0].id)
}
```

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

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P) lambda_function × Environment Var ×

ec2launchfunc1

lambda_function.py

```
import json
import boto3

def lambda_handler(event, context):
```

- 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

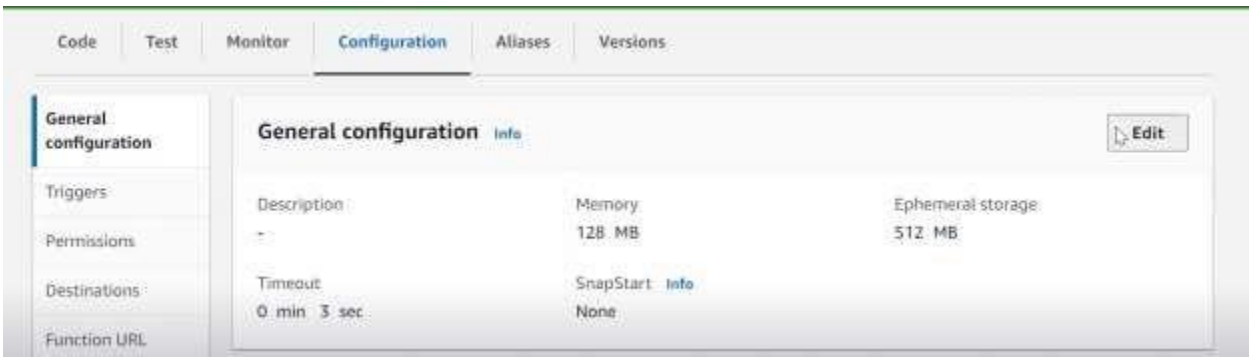
Edit saved event

Event name

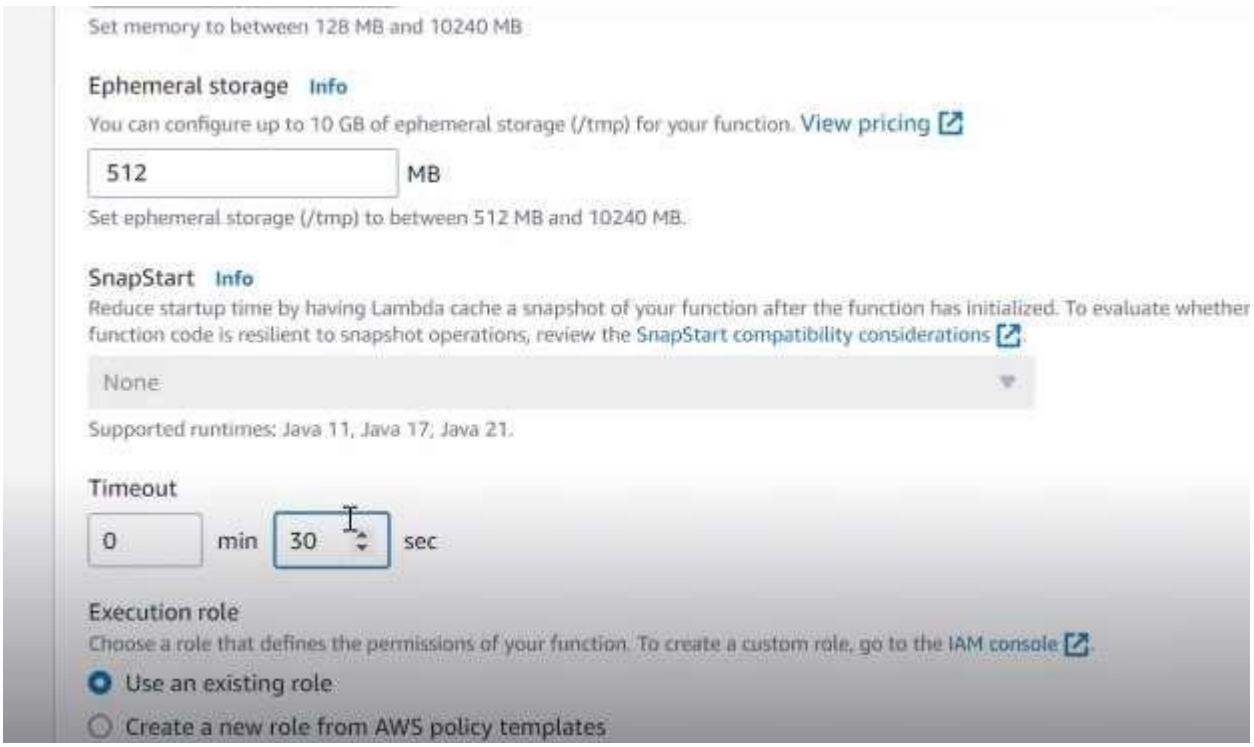
sdhfd

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

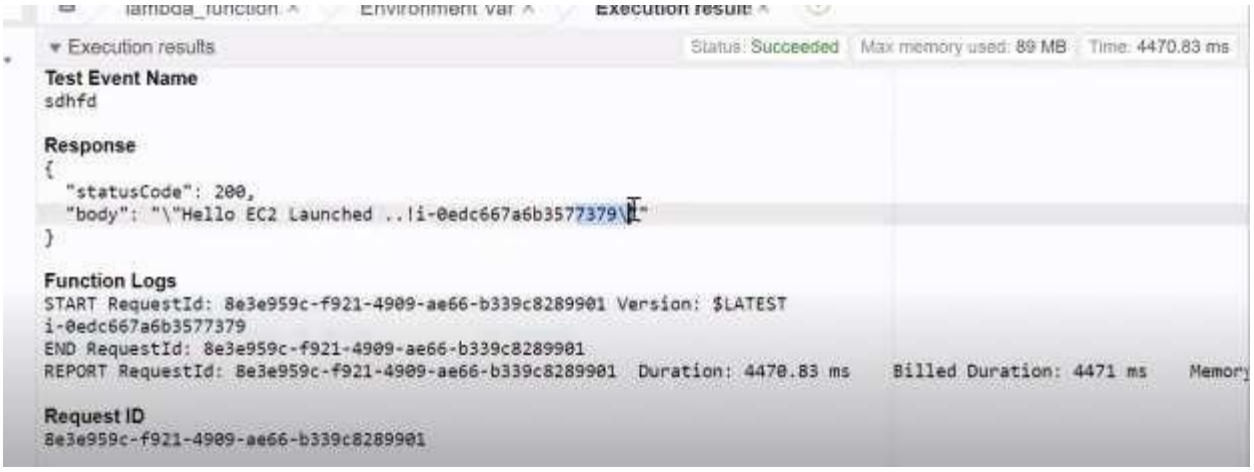
- If the execution time of the lambda function is more than 3 seconds it will be failed.
    - So, if your code is taking more time go to the ‘Configuration’ setting of lambda.



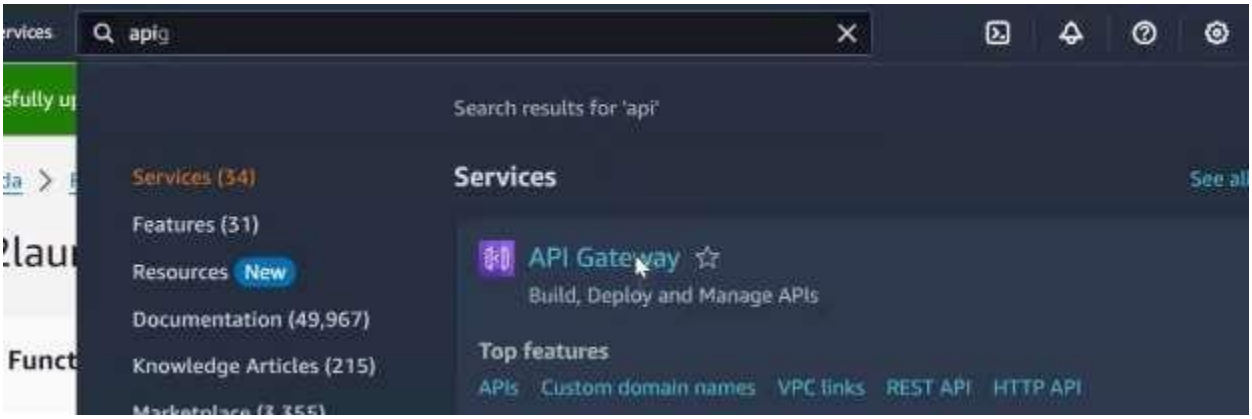
- And increase the timeout by 3.



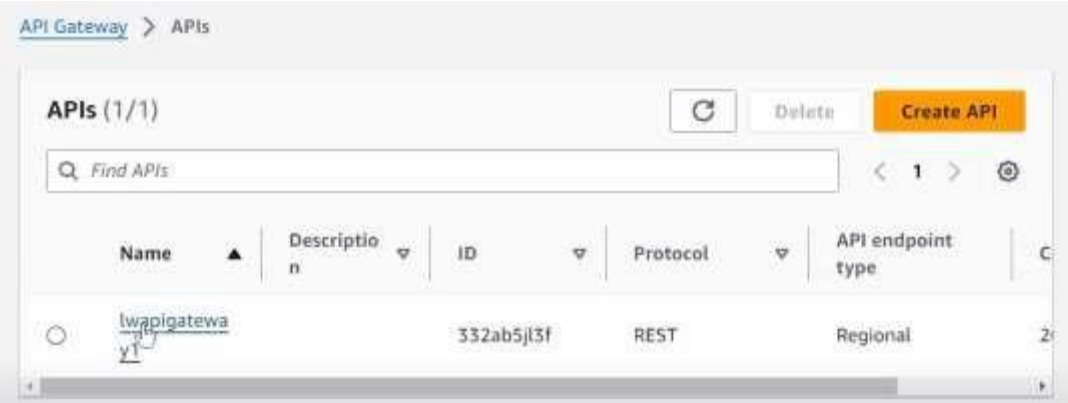
- Code is successfully executed.



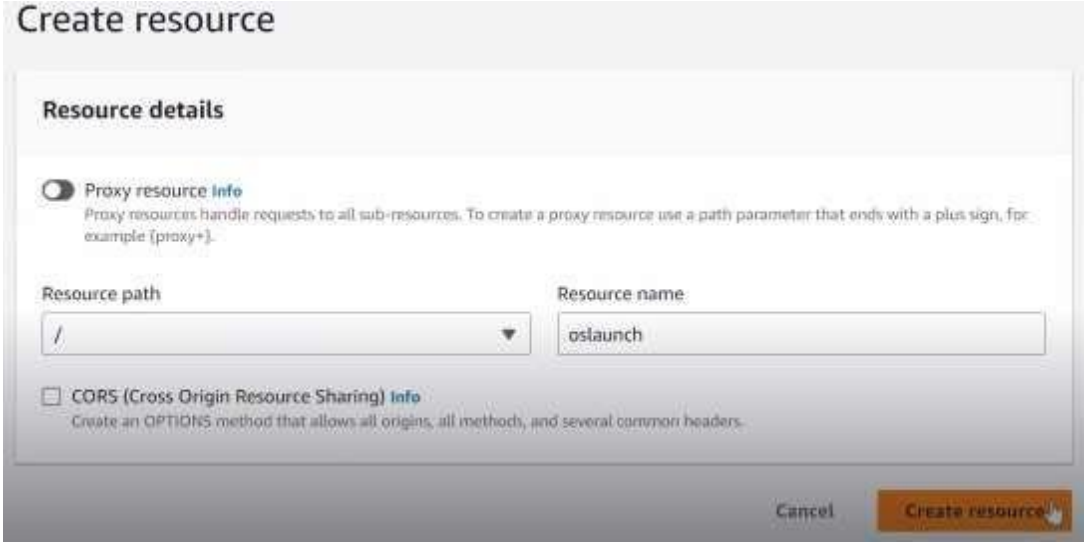
- Now, go to API Gateway



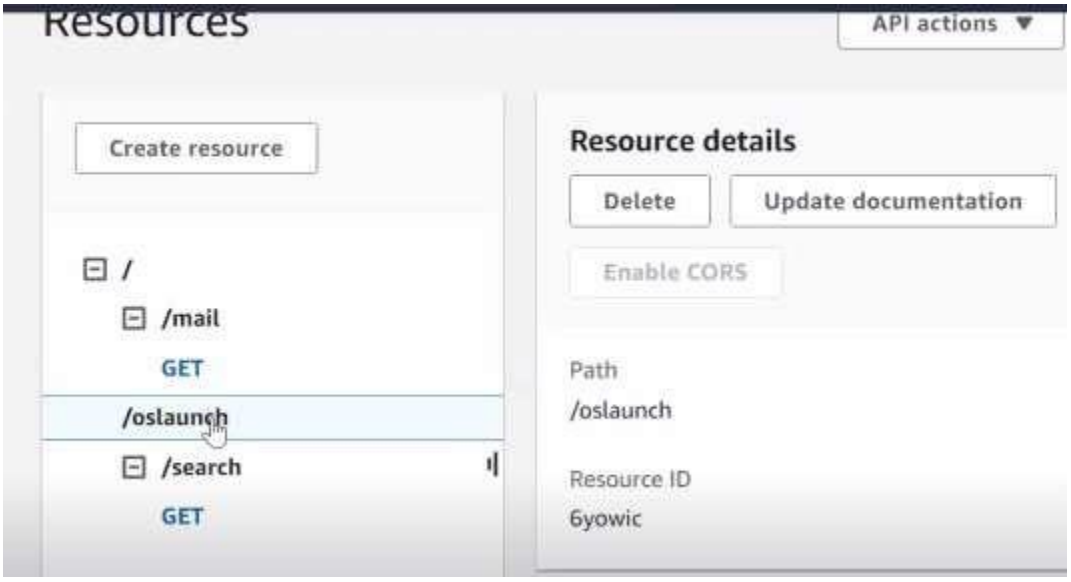
- Click on created API or create API if there is not.



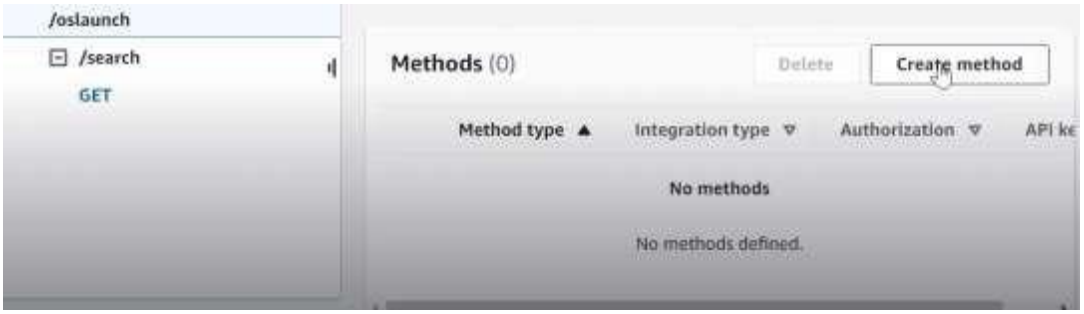
- Give the resource name and click ‘create resource’



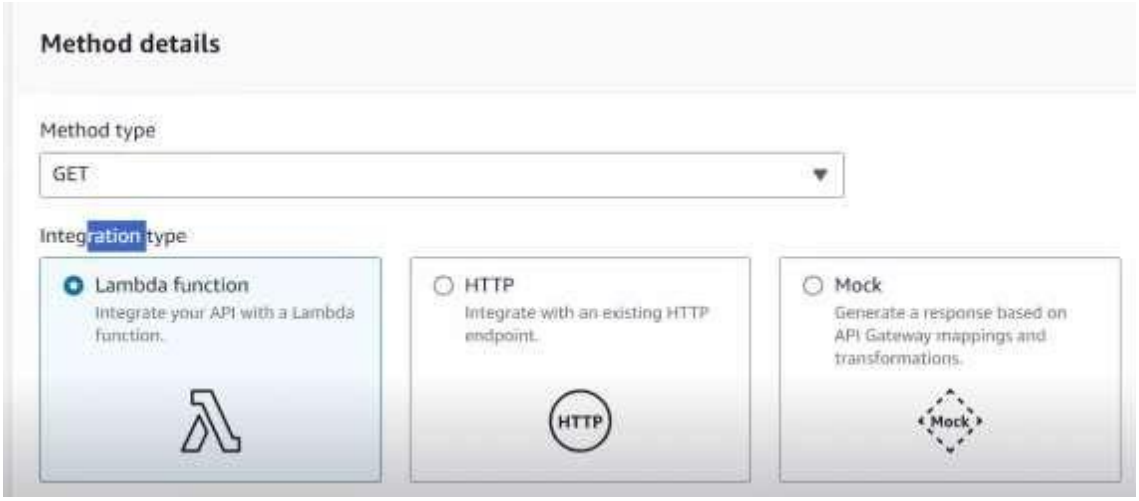
- New resource is created



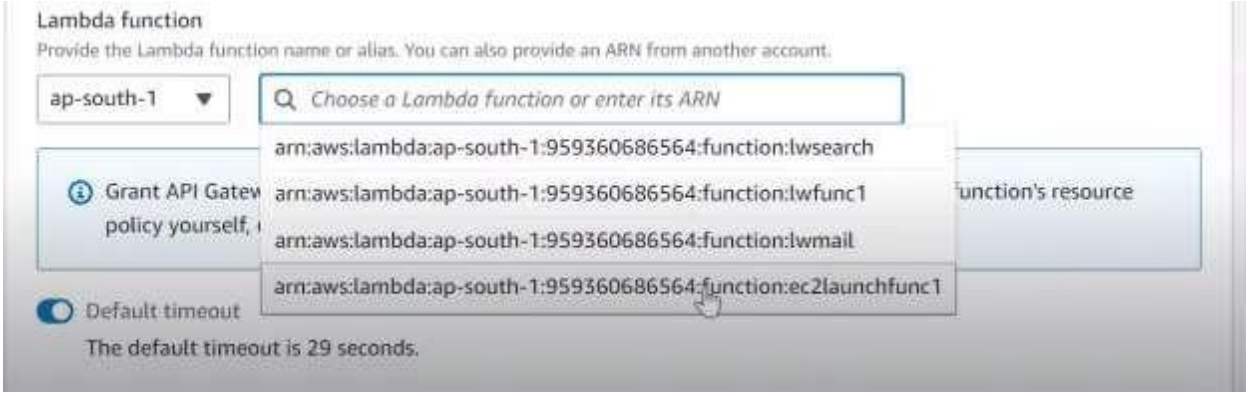
- Click on ‘create method’



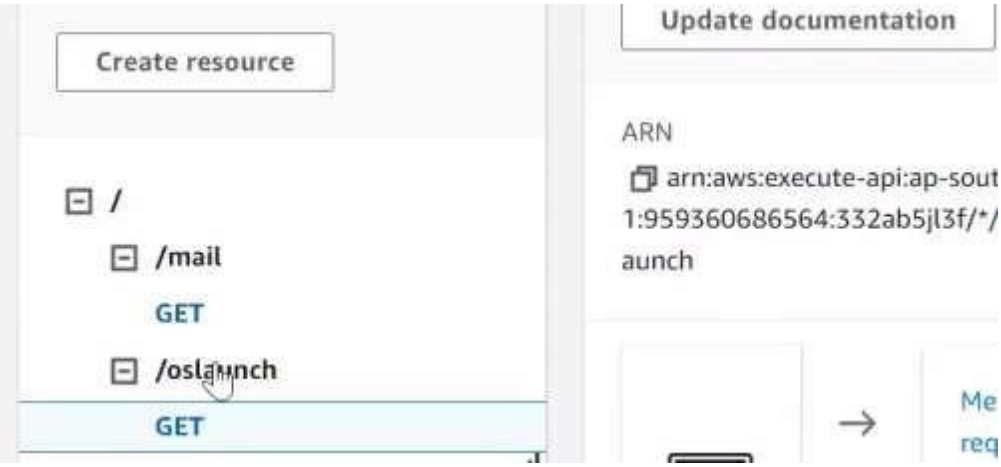
- Choose method as ‘GET’ and service as Lambda function.



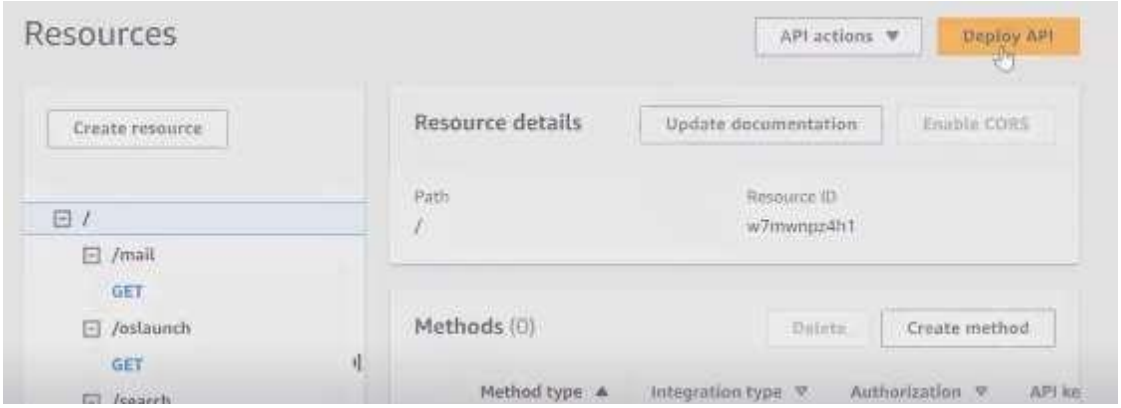
- Choose the lambda function name that you want to integrate. Click on ‘create method’



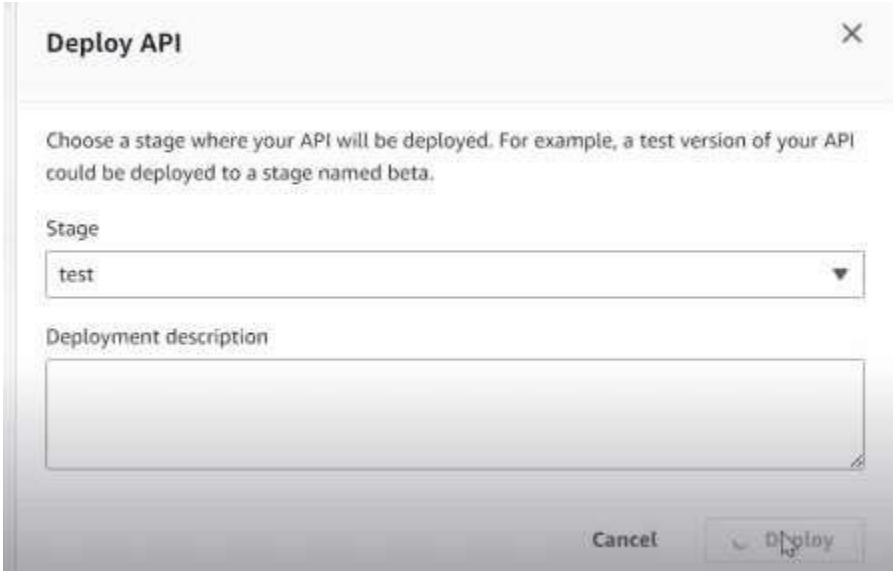
- One URL is created



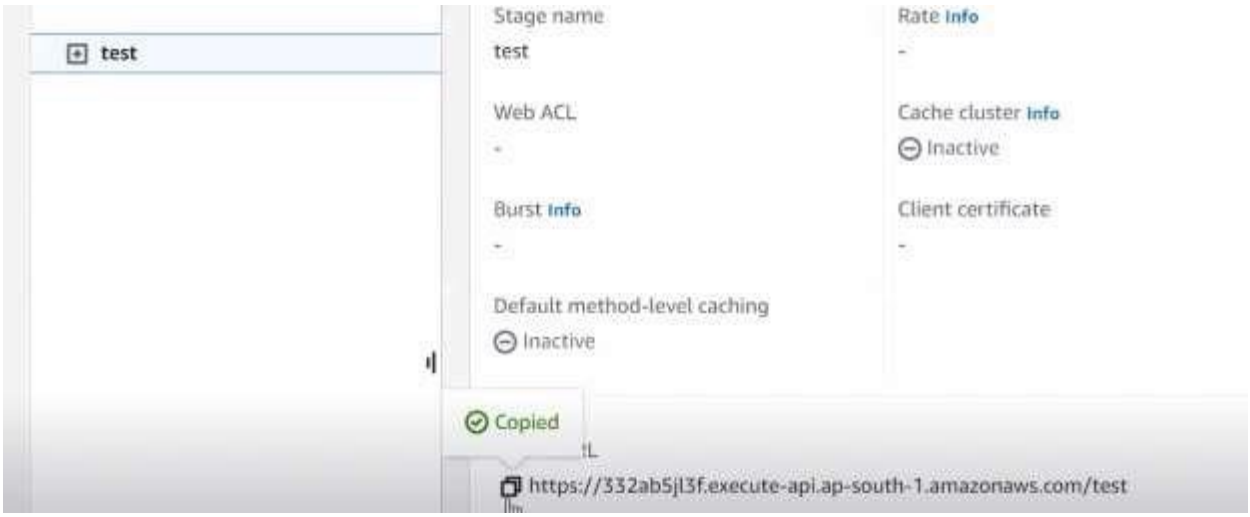
- Click on ‘Deploy API’ to redeploy the API



- 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.**