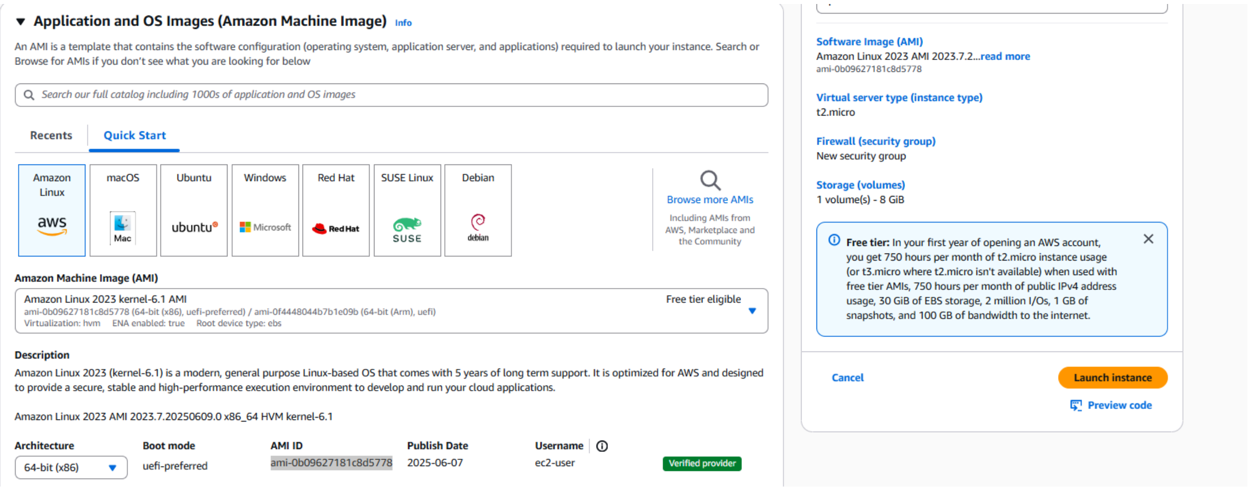
Use case

Launch and manage EC2 instances using Python scripts

1. Get the base AMI from AWS console (Here, using Amazon Linux AMI) with id ami-0b09627181c8d5778



Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI — Free tier eligible
ami-0b09627181c8d5778 (64-bit (x86), uefi-preferred) / ami-0f4448044b7b1e09b (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.7.20250609.0 x86_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Publish Date	Username	
64-bit (x86)	uefi-preferred	ami-0b09627181c8d5778	2025-06-07	ec2-user	Verified provider

Software Image (AMI)
Amazon Linux 2023 AMI 2023.7.2...read more
ami-0b09627181c8d5778

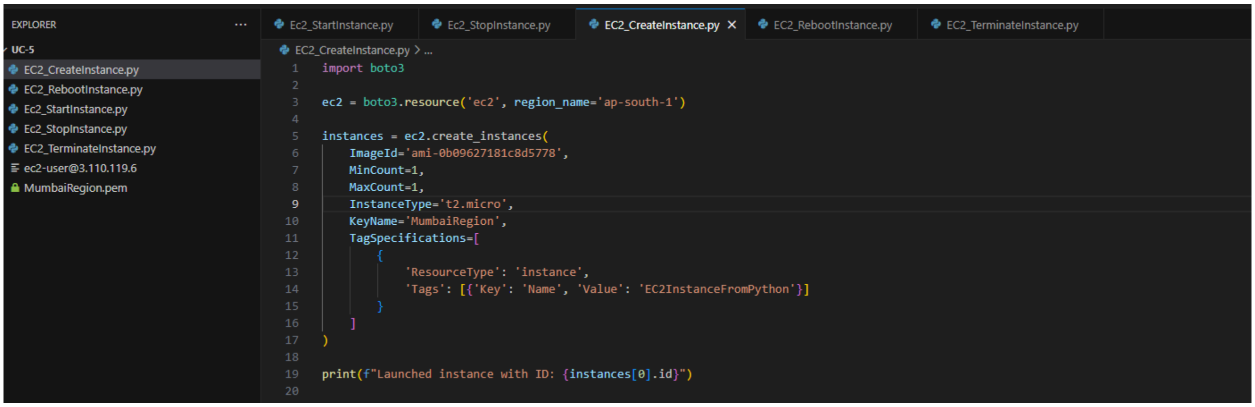
Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

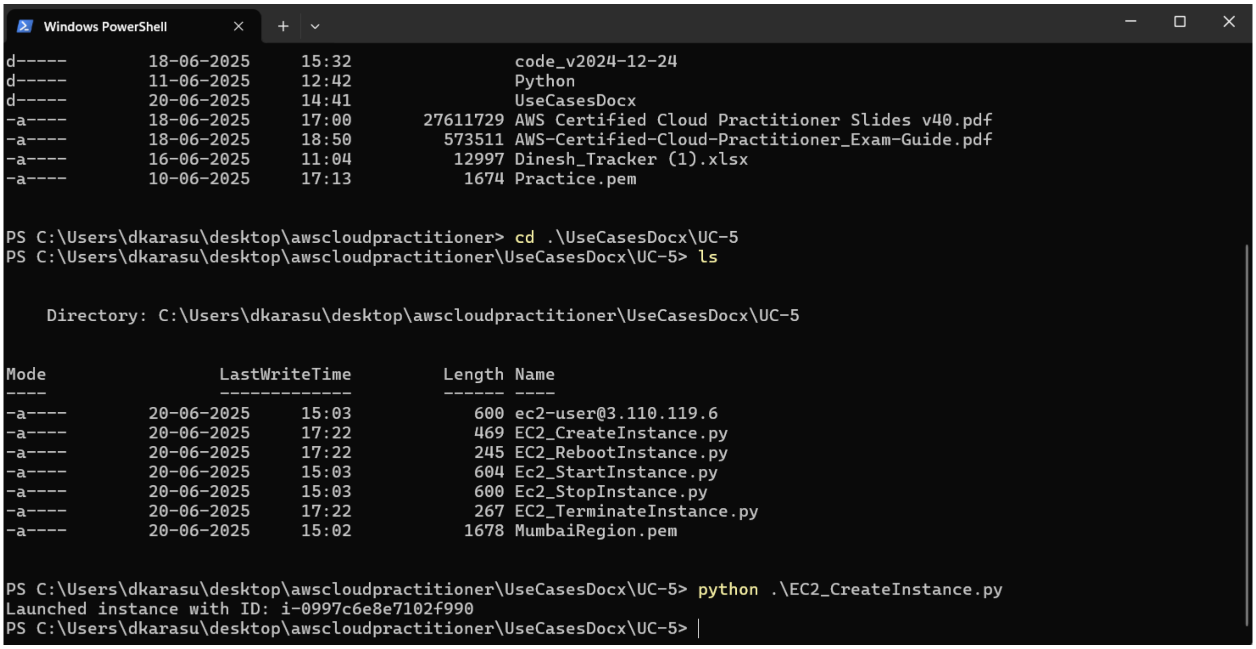
Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

2. Import boto3 module and use built action create_instances() by providing proper values to the arguments



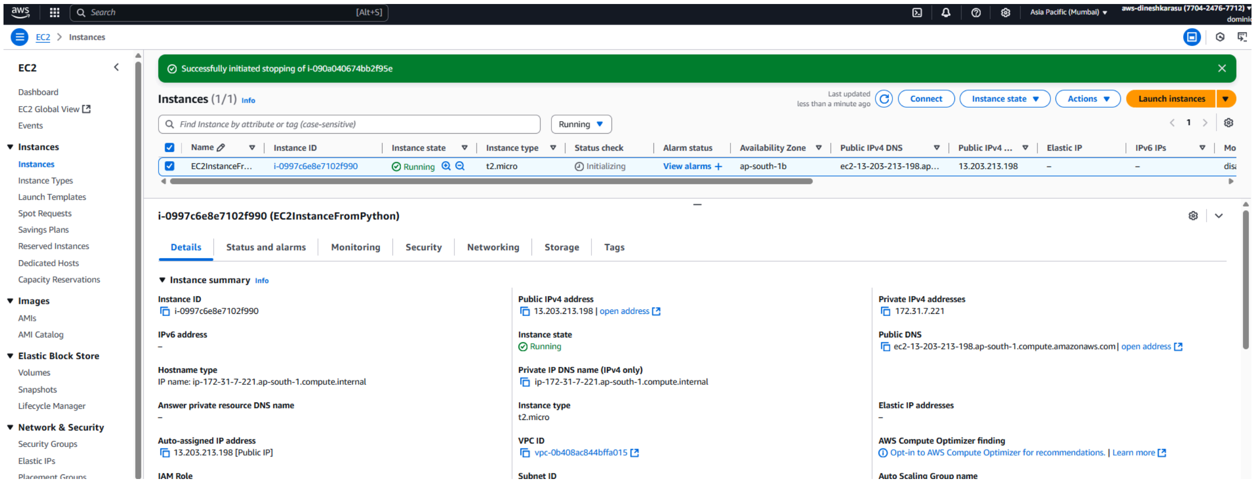
```python
import boto3

ec2 = boto3.resource('ec2', region_name='ap-south-1')

instances = ec2.create_instances(
    ImageId='ami-0b09627181c8d5778',
    MinCount=1,
    MaxCount=1,
    InstanceType='t2.micro',
    KeyName='MumbaiRegion',
    TagSpecifications=[
        {
            'ResourceType': 'instance',
            'Tags': [{'Key': 'Name', 'Value': 'EC2InstanceFromPython'}]
        }
    ]
)

print(f"Launched instance with ID: {instances[0].id}")

```

3. Launch the Instance with python code from AWS CLI as below



```
d-----        18-06-2025     15:32                code_v2024-12-24
d-----        11-06-2025     12:42                Python
d-----        20-06-2025     14:41                UseCasesDocx
-a----        18-06-2025     17:00       27611729 AWS Certified Cloud Practitioner Slides v40.pdf
-a----        18-06-2025     18:50         573511 AWS-Certified-Cloud-Practitioner_Exam-Guide.pdf
-a----        16-06-2025     11:04          12997 Dinesh_Tracker (1).xlsx
-a----        10-06-2025     17:13           1674 Practice.pem


PS C:\Users\dkarasu\desktop\awscloudpractitioner> cd .\UseCasesDocx\UC-5
PS C:\Users\dkarasu\desktop\awscloudpractitioner\UseCasesDocx\UC-5> ls


    Directory: C:\Users\dkarasu\desktop\awscloudpractitioner\UseCasesDocx\UC-5


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        20-06-2025     15:03            600 ec2-user@3.110.119.6
-a----        20-06-2025     17:22            469 EC2_CreateInstance.py
-a----        20-06-2025     17:22            245 EC2_RebootInstance.py
-a----        20-06-2025     15:03            604 Ec2_StartInstance.py
-a----        20-06-2025     15:03            600 Ec2_StopInstance.py
-a----        20-06-2025     17:22            267 EC2_TerminateInstance.py
-a----        20-06-2025     15:02           1678 MumbaiRegion.pem


PS C:\Users\dkarasu\desktop\awscloudpractitioner\UseCasesDocx\UC-5> python .\EC2_CreateInstance.py
Launched instance with ID: i-0997c6e8e7102f990
PS C:\Users\dkarasu\desktop\awscloudpractitioner\UseCasesDocx\UC-5> |
```

4. A new instance is created in the specified region, you can see the launched instance ID



Successfully initiated stopping of i-090a040674bb2f95e

Instances (1/1) Info

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

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Mo
☑	EC2InstanceFr…	i-0997c6e8e7102f990	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-13-203-213-198.ap…	13.203.213.198	–	–	disa

i-0997c6e8e7102f990 (EC2InstanceFromPython)

Instance summary Info

Instance ID
i-0997c6e8e7102f990

IPv6 address
–

Hostname type
IP name: ip-172-31-7-221.ap-south-1.compute.internal

Answer private resource DNS name
–

Auto-assigned IP address
13.203.213.198 [Public IP]

IAM Role

Public IPv4 address
13.203.213.198 | open address

Instance state
Running

Private IP DNS name (IPv4 only)
ip-172-31-7-221.ap-south-1.compute.internal

Instance type
t2.micro

VPC ID
vpc-0b408ac844bffa015

Subnet ID

Private IPv4 addresses
172.31.7.221

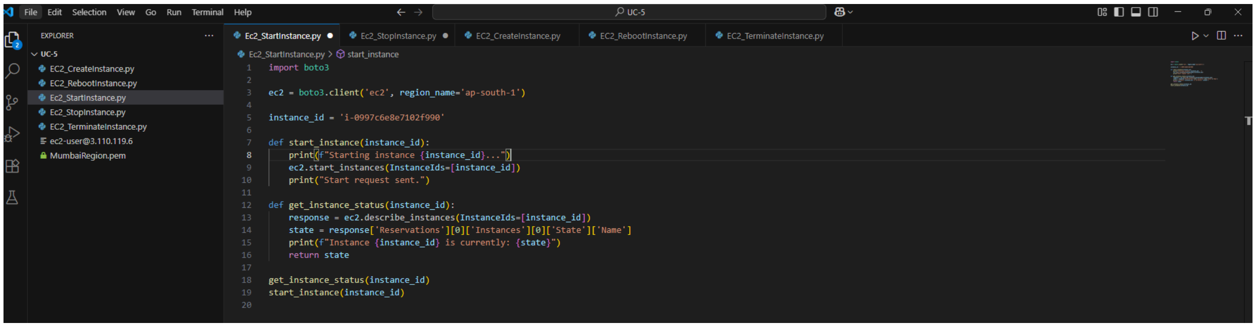
Public DNS
ec2-13-203-213-198.ap-south-1.compute.amazonaws.com | open address

Elastic IP addresses
–

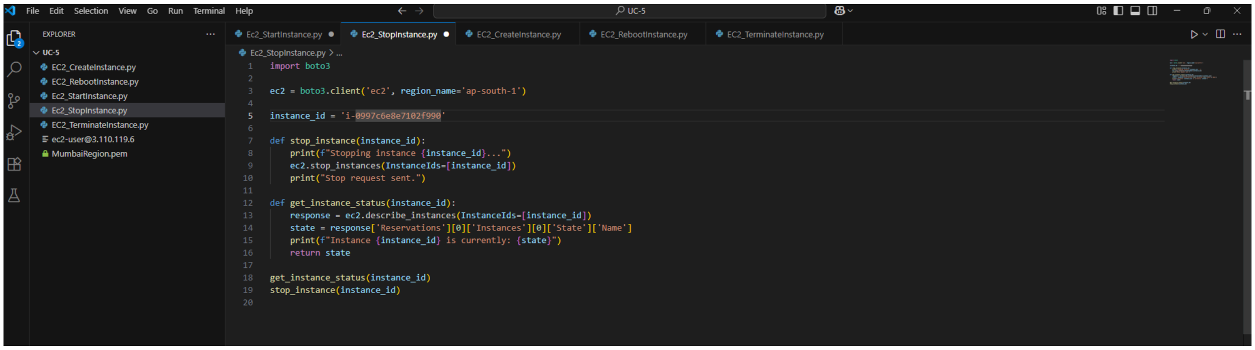
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Auto Scaling Group name

5. Similarly, We can reboot/Start/Stop/Terminate the instances by running the following code

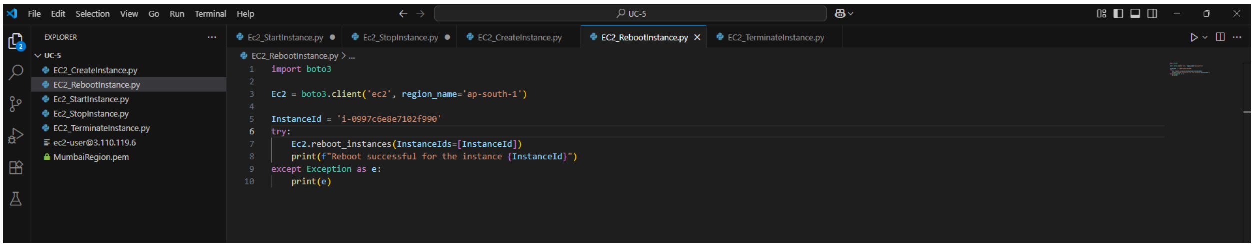


```python
import boto3

ec2 = boto3.client('ec2', region_name='ap-south-1')

instance_id = 'i-0997c6e8e7102f990'

def start_instance(instance_id):
    print(f"Starting instance {instance_id}...")
    ec2.start_instances(InstanceIds=[instance_id])
    print("Start request sent.")

def get_instance_status(instance_id):
    response = ec2.describe_instances(InstanceIds=[instance_id])
    state = response['Reservations'][0]['Instances'][0]['State']['Name']
    print(f"Instance {instance_id} is currently: {state}")
    return state

get_instance_status(instance_id)
start_instance(instance_id)
```

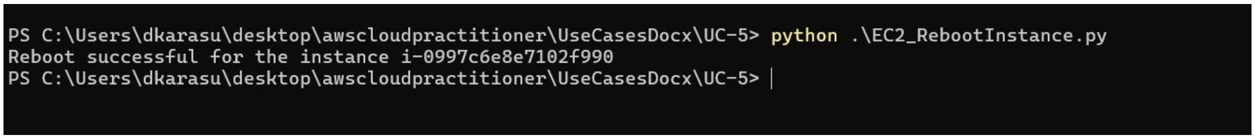


```python
import boto3

ec2 = boto3.client('ec2', region_name='ap-south-1')

instance_id = 'i-0997c6e8e7102f990'

def stop_instance(instance_id):
    print(f"Stopping instance {instance_id}...")
    ec2.stop_instances(InstanceIds=[instance_id])
    print("Stop request sent.")

def get_instance_status(instance_id):
    response = ec2.describe_instances(InstanceIds=[instance_id])
    state = response['Reservations'][0]['Instances'][0]['State']['Name']
    print(f"Instance {instance_id} is currently: {state}")
    return state

get_instance_status(instance_id)
stop_instance(instance_id)
```

Reboot Instance

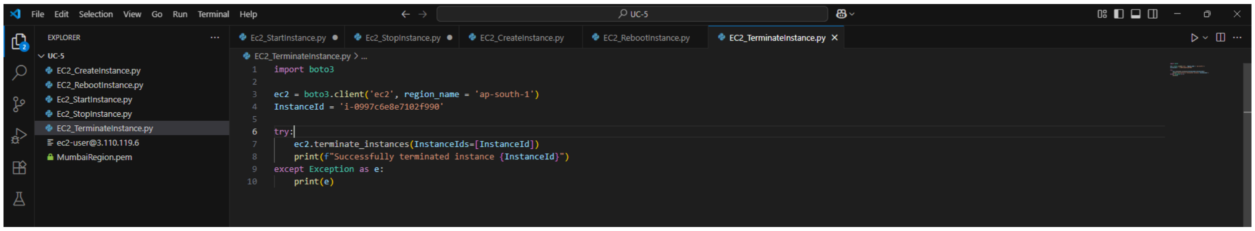


```python
import boto3

Ec2 = boto3.client('ec2', region_name='ap-south-1')

InstanceId = 'i-0997c6e8e7102f990'
try:
    Ec2.reboot_instances(InstanceIds=[InstanceId])
    print(f"Reboot successful for the instance {InstanceId}")
except Exception as e:
    print(e)
```

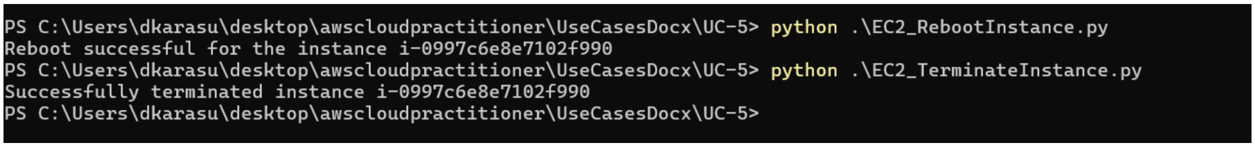


```
PS C:\Users\dkarasu\desktop\awscloudpractitioner\UseCasesDocx\UC-5> python .\EC2_RebootInstance.py
Reboot successful for the instance i-0997c6e8e7102f990
PS C:\Users\dkarasu\desktop\awscloudpractitioner\UseCasesDocx\UC-5>
```

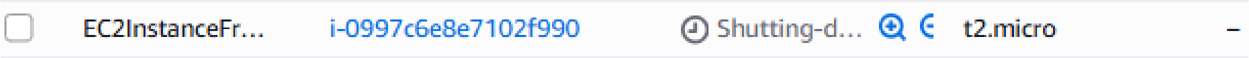
Terminate Instance



```python
import boto3

ec2 = boto3.client('ec2', region_name = 'ap-south-1')
InstanceId = 'i-0997c6e8e7102f990'

try:
    ec2.terminate_instances(InstanceIds=[InstanceId])
    print(f"Successfully terminated instance {InstanceId}")
except Exception as e:
    print(e)
```



```
PS C:\Users\dkarasu\desktop\awscloudpractitioner\UseCasesDocx\UC-5> python .\EC2_RebootInstance.py
Reboot successful for the instance i-0997c6e8e7102f990
PS C:\Users\dkarasu\desktop\awscloudpractitioner\UseCasesDocx\UC-5> python .\EC2_TerminateInstance.py
Successfully terminated instance i-0997c6e8e7102f990
PS C:\Users\dkarasu\desktop\awscloudpractitioner\UseCasesDocx\UC-5>
```



□	EC2InstanceFr…	i-0997c6e8e7102f990	Shutting-d…	t2.micro	–