LULEÅ UNIVERSITY OF TECHNOLOGY

M7024E Laboratory 3: Programming Cloud Services - Compute Services

submitted by

Muiz Olalekan Raheem, Md Asif Mahmod Tusher Siddique

December 2, 2022

submitted to

Dr. Karan Mitra

1 Objectives

The objective of this lab is to:

• Setup a programming environment for building (using programming tools and languages) Cloud services for a major Cloud provider, for example, Amazon Web services (AWS);

• Develop Cloud services for managing the EC2 compute services using the APIs provided by the AWS.

• Develop Cloud services for monitoring the EC2 compute services using the APIs provided by the AWS.

2 Exercises

2.1 Setting up the programming environment:

2.2 Exercise a

In this exercise, you will learn how to use the compute service provided by AWS.

1. Identify ways of creating the Amazon EC2 service clients. Look at the lecture slides and the Java Developer Guide.

(a) Explain in detail how the Amazon EC2 service clients are created by providing details of the packages and classes involved. Create a diagram of the dependencies involved.

Answer:

Making an Amazon EC2 service client is a simple process.

- Configure the AWS account's authentication credentials using the CLI command "aws configure."

- Put boto3 and other necessary dependencies in place.

- Start writing code.

To develop EC2 service clients, we used the Python SDK. There are two Python packages in the SDK:

Botocore: The library that offers the low-level features that are shared by the AWS CLI and the Python SDK.

Boto3: The actual Python SDK implementation.

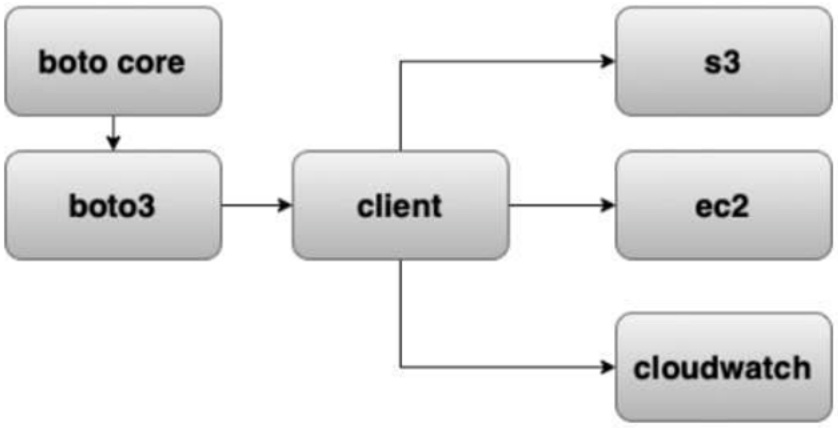
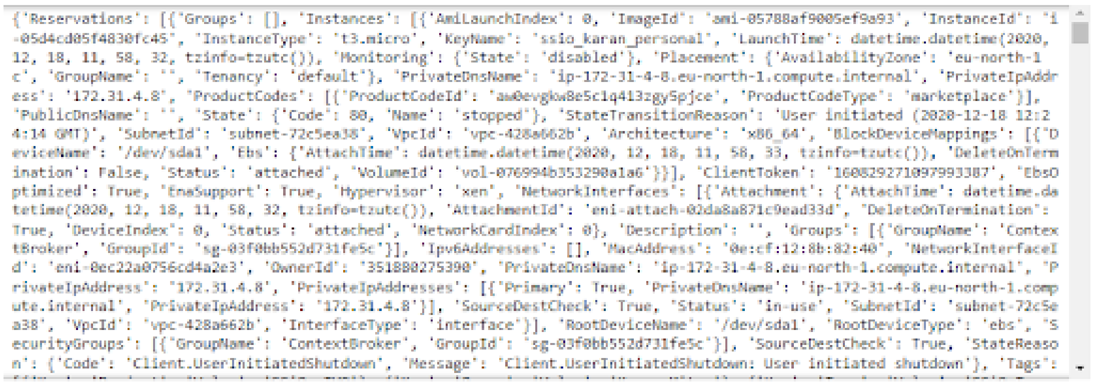


Figure 1: Class diagram of the used classes and methods for EC2 management [1]

2. Create a Java program to manage your EC2 instance. Start with listing region names and their endpoints.

```
def get_all_regions(service_name='ec2'):

    s = Session()
    regions = s.get_available_regions(service_name)
    print(regions)
    return regions

def get_regions(service_name='ec2'):
    ec2 = boto3.client(service_name)
    response = ec2.describe_instances()
    zones = []
    for i in range(len(response['Reservations'])):
        zones.append(response['Reservations'][i]['Instances'][0]['Placement']
        ['AvailabilityZone'])
    print(zones)
    return zones
get_regions()
```

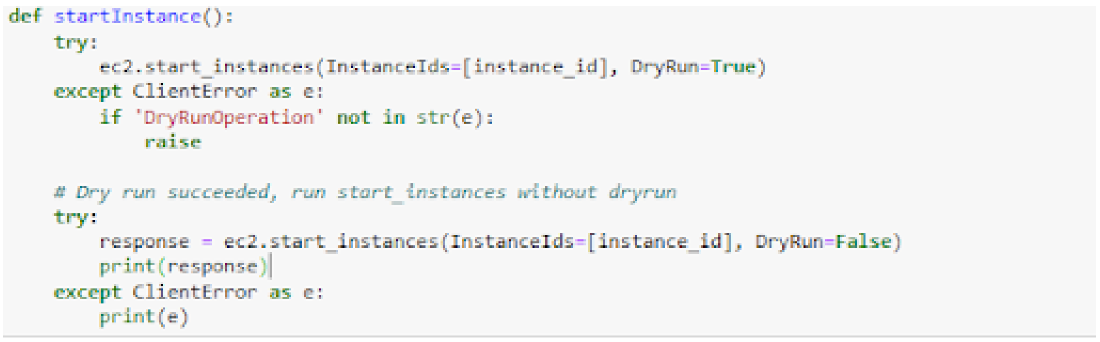


```
{'Reservations': [{'Groups': [], 'Instances': [{'AmiLaunchIndex': 0, 'ImageId': 'ami-05788af9005ef9a93', 'InstanceId': 'i
-05d4cd05f4830fc45', 'InstanceType': 't3.micro', 'KeyName': 'ssio_karan_personal', 'LaunchTime': datetime.datetime(2020,
12, 18, 11, 58, 32, tzinfo=tzutc()), 'Monitoring': {'State': 'disabled'}, 'Placement': {'AvailabilityZone': 'eu-north-1
c', 'GroupName': '', 'Tenancy': 'default'}, 'PrivateDnsName': 'ip-172-31-4-8.eu-north-1.compute.internal', 'PrivateIpAddr
ess': '172.31.4.8', 'ProductCodes': [{'ProductCodeId': 'aw0evgkw8e5c1q413zgy5pjce', 'ProductCodeType': 'marketplace'}],
'PublicDnsName': '', 'State': {'Code': 80, 'Name': 'stopped'}, 'StateTransitionReason': 'User initiated (2020-12-18 12:2
4:14 GMT)', 'SubnetId': 'subnet-72c5ea38', 'VpcId': 'vpc-428a662b', 'Architecture': 'x86_64', 'BlockDeviceMappings': [{'D
eviceName': '/dev/sda1', 'Ebs': {'AttachTime': datetime.datetime(2020, 12, 18, 11, 58, 33, tzinfo=tzutc()), 'DeleteOnTerm
ination': False, 'Status': 'attached', 'VolumeId': 'vol-076994b353290a1a6'}}], 'ClientToken': '160829271097993387', 'EbsO
ptimized': True, 'EnaSupport': True, 'Hypervisor': 'xen', 'NetworkInterfaces': [{'Attachment': {'AttachTime': datetime.da
tetime(2020, 12, 18, 11, 58, 32, tzinfo=tzutc()), 'AttachmentId': 'eni-attach-02da8a871c9ead33d', 'DeleteOnTermination':
True, 'DeviceIndex': 0, 'Status': 'attached', 'NetworkCardIndex': 0}, 'Description': '', 'Groups': [{'GroupName': 'Contex
tBroker', 'GroupId': 'sg-03f0bb552d731fe5c'}], 'Ipv6Addresses': [], 'MacAddress': '0e:cf:12:8b:82:40', 'NetworkInterfaceI
d': 'eni-0ec22a0756cd4a2e3', 'OwnerId': '351880275390', 'PrivateDnsName': 'ip-172-31-4-8.eu-north-1.compute.internal', 'P
rivateIpAddress': '172.31.4.8', 'PrivateIpAddresses': [{'Primary': True, 'PrivateDnsName': 'ip-172-31-4-8.eu-north-1.comp
ute.internal', 'PrivateIpAddress': '172.31.4.8'}], 'SourceDestCheck': True, 'Status': 'in-use', 'SubnetId': 'subnet-72c5e
a38', 'VpcId': 'vpc-428a662b', 'InterfaceType': 'interface'}], 'RootDeviceName': '/dev/sda1', 'RootDeviceType': 'ebs', 'S
ecurityGroups': [{'GroupName': 'ContextBroker', 'GroupId': 'sg-03f0bb552d731fe5c'}], 'SourceDestCheck': True, 'StateReaso
n': {'Code': 'Client.UserInitiatedShutdown', 'Message': 'Client.UserInitiatedShutdown: User initiated shutdown'}, 'Tags':
```

3. Write a method to run an instance from the list of regions from the previous step.

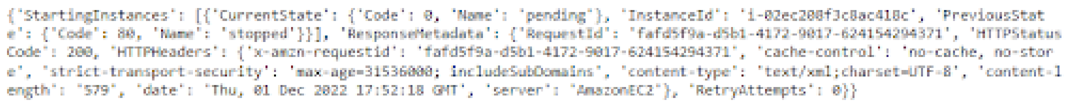
(a) Use ¡keypair, security groups, number of instances, etc¿ as parameters to start an instance.

Starting an instance using its ID as a parameter



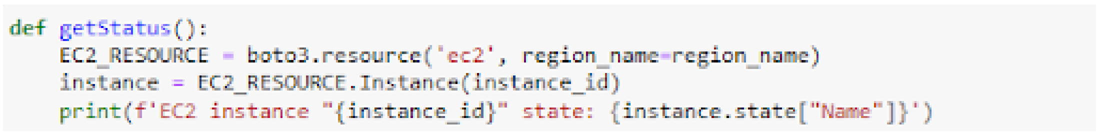
```
def startInstance():
    try:
        ec2.start_instances(InstanceIds=[instance_id], DryRun=True)
    except ClientError as e:
        if 'DryRunOperation' not in str(e):
            raise

    # Dry run succeeded, run start_instances without dryrun
    try:
        response = ec2.start_instances(InstanceIds=[instance_id], DryRun=False)
        print(response)
    except ClientError as e:
        print(e)
```

output:

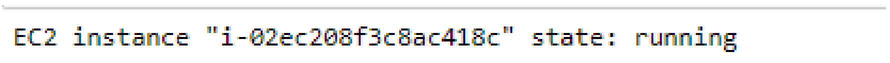


```
{'StartingInstances': [{'CurrentState': {'Code': 0, 'Name': 'pending'}, 'InstanceId': 'i-02ec208f3c8ac418c', 'PreviousStat
e': {'Code': 80, 'Name': 'stopped'}}], 'ResponseMetadata': {'RequestId': 'fafd5f9a-d5b1-4172-9017-624154294371', 'HTTPStatus
Code': 200, 'HTTPHeaders': {'x-amzn-requestid': 'fafd5f9a-d5b1-4172-9017-624154294371', 'cache-control': 'no-cache, no-stor
e', 'strict-transport-security': 'max-age=31536000; includeSubDomains', 'content-type': 'text/xml;charset=UTF-8', 'content-l
ength': '579', 'date': 'Thu, 01 Dec 2022 17:52:18 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}
```

4. Write a method to retrieve the status of your running instance(s).

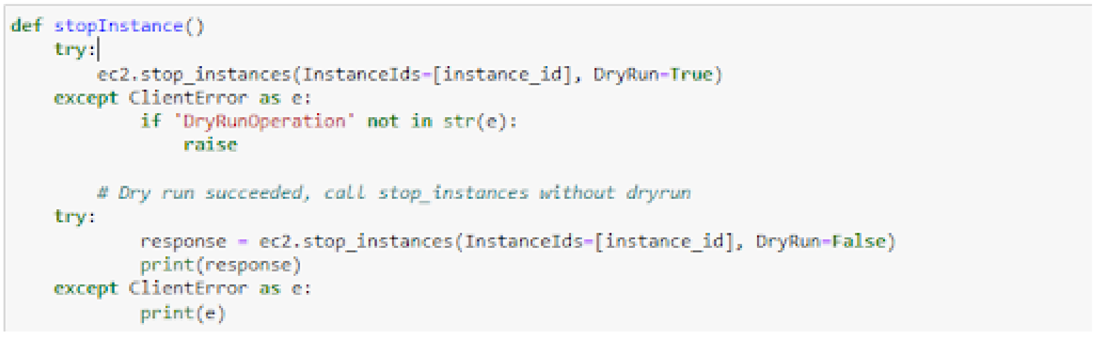


```
def getStatus():
    EC2_RESOURCE = boto3.resource('ec2', region_name=region_name)
    instance = EC2_RESOURCE.Instance(instance_id)
    print(f'EC2 instance "{instance_id}" state: {instance.state["Name"]}')
```

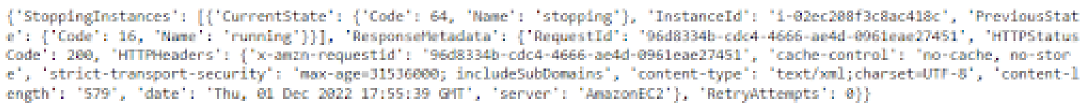


```
EC2 instance "i-02ec208f3c8ac418c" state: running
```

5. Write a method to stop an instance(s) that you started.



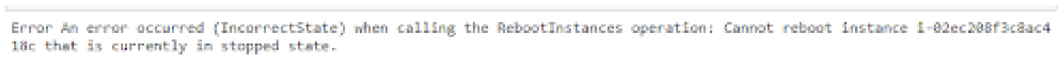
```
def stopInstance()
    try:
        ec2.stop_instances(InstanceIds=[instance_id], DryRun=True)
    except ClientError as e:
            if 'DryRunOperation' not in str(e):
                raise

        # Dry run succeeded, call stop_instances without dryrun
    try:
            response = ec2.stop_instances(InstanceIds=[instance_id], DryRun=False)
            print(response)
    except ClientError as e:
            print(e)
```



```
{'StoppingInstances': [{'CurrentState': {'Code': 64, 'Name': 'stopping'}, 'InstanceId': 'i-02ec208f3c8ac418c', 'PreviousStat
e': {'Code': 16, 'Name': 'running'}}], 'ResponseMetadata': {'RequestId': '96d8334b-cdc4-4666-ae4d-0961eae27451', 'HTTPStatus
Code': 200, 'HTTPHeaders': {'x-amzn-requestid': '96d8334b-cdc4-4666-ae4d-0961eae27451', 'cache-control': 'no-cache, no-stor
e', 'strict-transport-security': 'max-age=31536000; includeSubDomains', 'content-type': 'text/xml;charset=UTF-8', 'content-l
ength': '579', 'date': 'Thu, 01 Dec 2022 17:55:39 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}
```

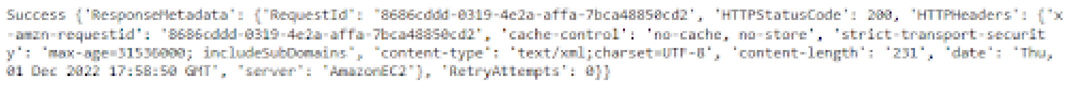
Rebooting Instances:

I. In a stopped state:



```
Error An error occurred (IncorrectState) when calling the RebootInstances operation: Cannot reboot instance i-02ec208f3c8ac4
18c that is currently in stopped state.
```

II. In a running/active state



```
Success {'ResponseMetadata': {'RequestId': '8686cddd-0319-4e2a-affa-7bca48850cd2', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x
-amzn-requestid': '8686cddd-0319-4e2a-affa-7bca48850cd2', 'cache-control': 'no-cache, no-store', 'strict-transport-securit
y': 'max-age=31536000; includeSubDomains', 'content-type': 'text/xml;charset=UTF-8', 'content-length': '231', 'date': 'Thu,
01 Dec 2022 17:58:50 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}
```

2.3 Exercise b

In this exercise, you will learn how to use the monitoring service provided by AWS.

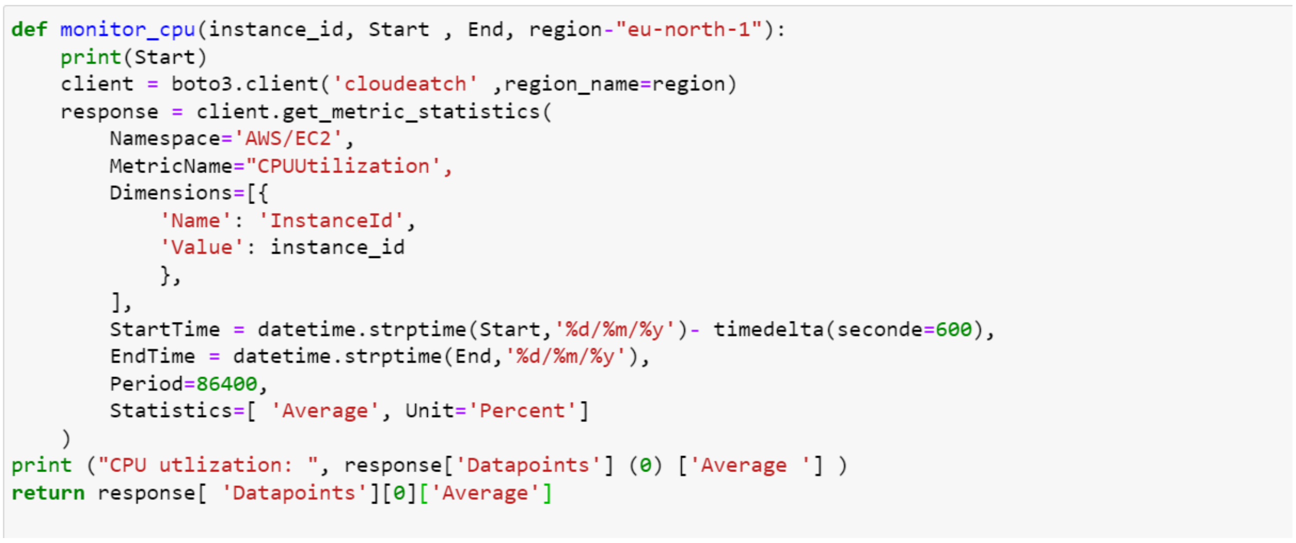
1. Identify ways of creating the CloudWatch clients

Answer:

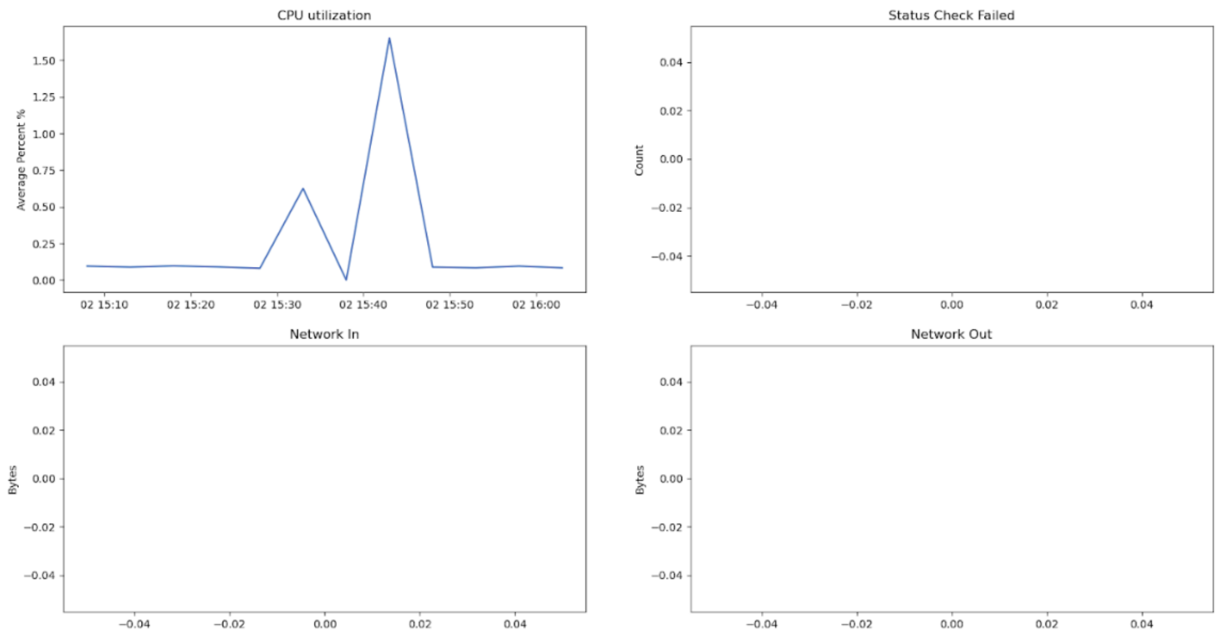
A number of SDKs can be used to build CloudWatch clients. In addition, the CloudWatch agent package can be downloaded through an S3 download URL and installed on the servers. However, to create our CloudWatch client, we used the Python SDK (boto3). Boto3's CloudWatch.Client class can be used to gather and monitor metrics, or the variables we wish to watch for our resources and applications.

2. Write a Java program to monitor the status of your EC2 instances (a) You should use as many metrics as possible. For example, CPU utilization, disk I/O, etc.

Answer:



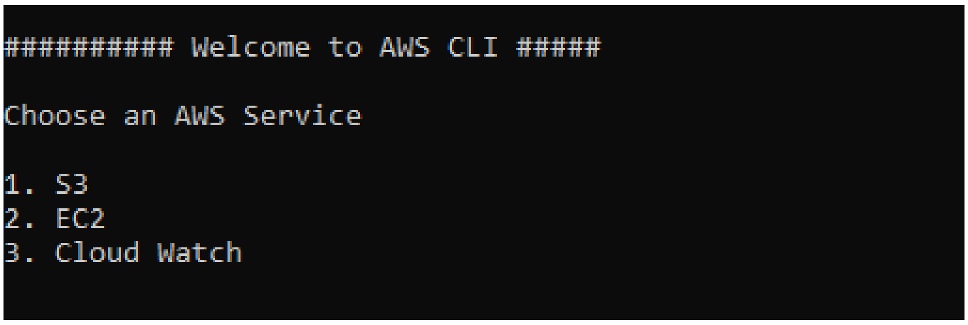
```
def monitor_cpu(instance_id, Start , End, region-"eu-north-1"):
    print(Start)
    client = boto3.client('cloudeatch' ,region_name=region)
    response = client.get_metric_statistics(
        Namespace='AWS/EC2',
        MetricName="CPUUtilization',
        Dimensions=[{
            'Name': 'InstanceId',
            'Value': instance_id
            },
        ],
        StartTime = datetime.strptime(Start,'%d/%m/%y')- timedelta(seconde=600),
        EndTime = datetime.strptime(End,'%d/%m/%y'),
        Period=86400,
        Statistics=[ 'Average', Unit='Percent']
    )
print ("CPU utlization: ", response['Datapoints'] (0) ['Average '] )
return response[ 'Datapoints'][0]['Average']
```



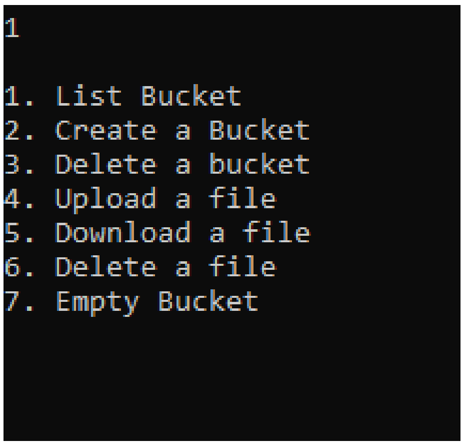
2.4 Exercise c

Be more creative! Add more methods of your choice, build a CLI or a UI. You may want to Integrate the S3 component (from the previous lab) with EC2 component in the UI. Ask the tutor/Karan for more ideas!

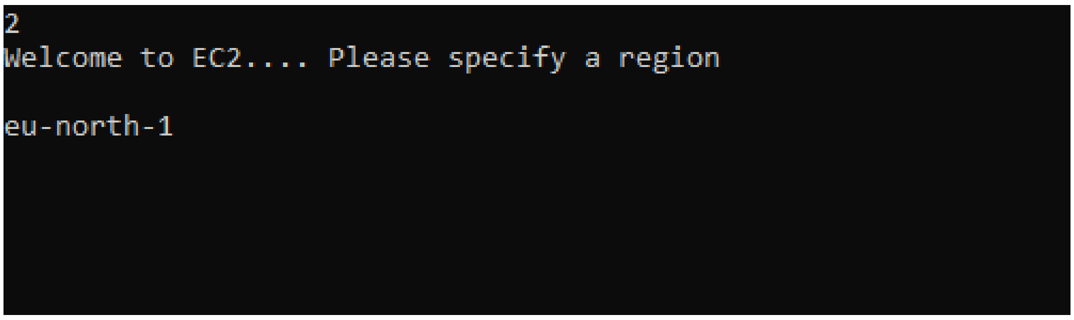
Answer:



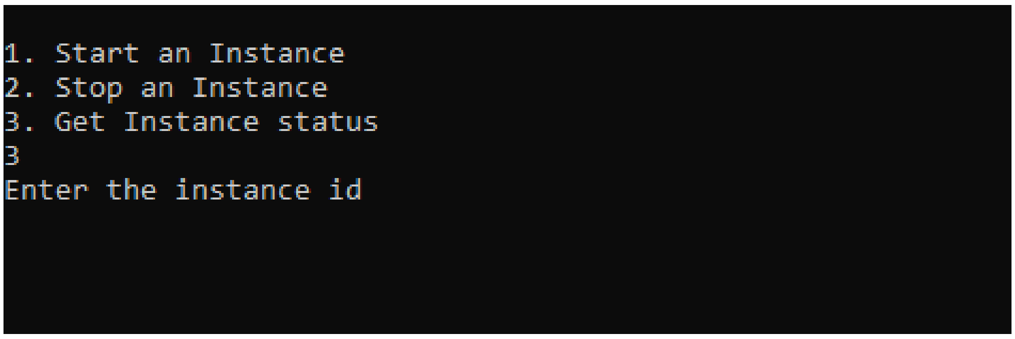
```
########## Welcome to AWS CLI #####

Choose an AWS Service

1. S3
2. EC2
3. Cloud Watch
```



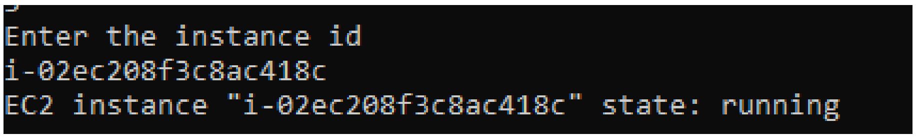
```
1

1. List Bucket
2. Create a Bucket
3. Delete a bucket
4. Upload a file
5. Download a file
6. Delete a file
7. Empty Bucket
```



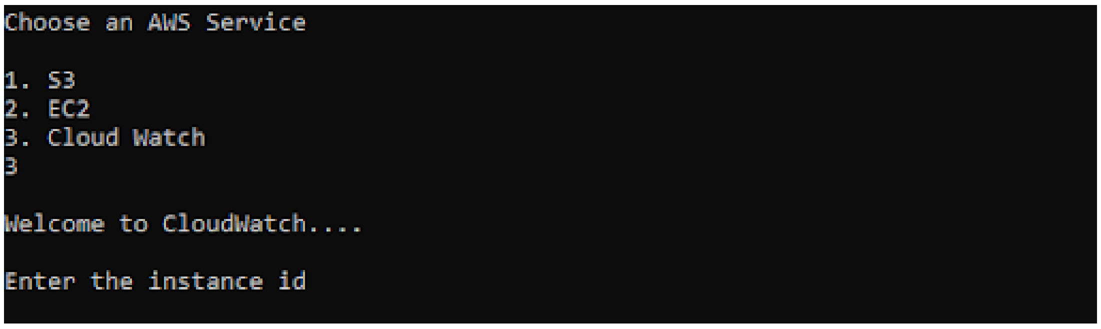
```
2
Welcome to EC2.... Please specify a region

eu-north-1
```



```
1. Start an Instance
2. Stop an Instance
3. Get Instance status
3
Enter the instance id
```



```
Enter the instance id
i-02ec208f3c8ac418c
EC2 instance "i-02ec208f3c8ac418c" state: running
```



```
Choose an AWS Service

1. S3
2. EC2
3. Cloud Watch
3

Welcome to CloudWatch....

Enter the instance id
```