M7024E Laboratory 3: Programming Cloud Services - Compute Services

Ameer Hamza, Wania Khan

December 24, 2021

Lab Report

Exercise 1

1. 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.

We used Boto3 as AWS SDK for python by creating a boto3 client object which provides an interface to control Amazon computing resources such as EC2 web service that provides a computing capacity in the AWS cloud. A low-level client is created using EC2.client class which represents amazon elastic compute cloud. To access the Amazon computing resources and perform different functions, we first validated AWS access key ID and secret access key for authentication of user requests. [1]

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

Specific libraries (e.g. boto3, logging etc) for accessing particular packages and performing certain operations are initialized in the start of the program. A function specifically for listing regions names is defined where a boto3 client object is created and within that an EC2 client object is created for accessing AWS EC2 service. A for-loop is used to return the names of all the regions where EC2 instances are created using describe-regions() function. On running this program, a list of regions are returned or user can enter an instance ID for specific EC2 instance to get its region name.

3. Create a program that runs an instance from the list of regions.

A function to run an instance is created where a list of regions of EC2 instances is provided to the user. Based on the region selection, an instance is started into running state using start-instances() - a client service method.

4. Create a program to retrieve the status of your running instance(s).

A function – running-instances() is created which allows retrieval of status of running instances. First, all the instances with running states are filtered out in a list using instances.filter() method. A for-loop is traversed over the obtained list to retrieve and print the information based on the specified parameters such as, instance-id, instance-type, instance-state etc.

5. Create a program to stop an instance(s) that you started.

A separate function named stop-instance() is created where user is asked to enter an instance-ID. Based on the entered ID which as an argument is passed to the stop-instances() service client method for stopping the instance from running state.

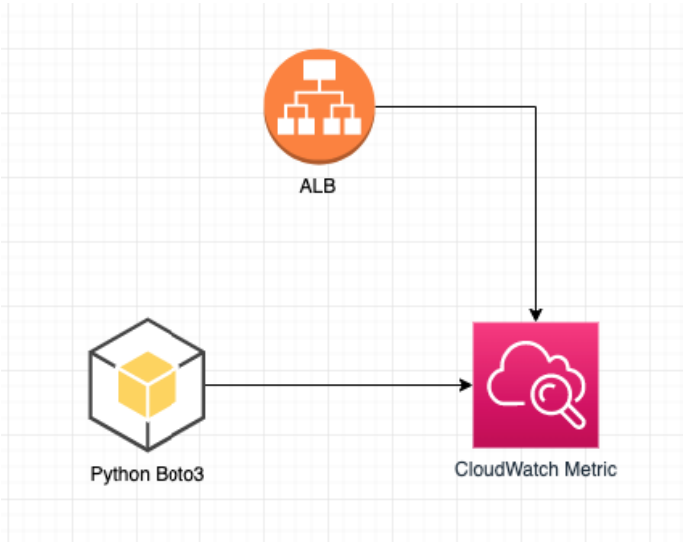

Figure 1: Dependencies Diagram [2]

Program Interface

```
Select an option:
1. List Status of all instances
2. Create a new EC2 instance
3. Start an instance
4. Stop an instance
5. Delete an instance
6. List regions
7. Get Metrics
8. Exit


Enter your choice [1-8]: 6


Listing regions
eu-north-1
ap-south-1
eu-west-3
eu-west-2
eu-west-1
ap-northeast-3
ap-northeast-2
ap-northeast-1
sa-east-1
ca-central-1
ap-southeast-1
ap-southeast-2
eu-central-1
us-east-1
us-east-2
us-west-1
us-west-2
```

Figure 2: Command Line Menu

We developed an easy to use user interface in Python to create and manage EC2 instance and objects. Initially a CLI menu is created, however we plan on making a web based app as well.

Exercise 2

1. Creating the CloudWatch clients.

To monitor resources of our EC2 instance, Boto3 allow us to create a low-level client of Amazon CloudWatch to use services such as collecting and tracking metrics. In our program, we have created a cloud-watch object from CloudWatch.Client class to monitor the CPU usage and disk reads and writes of our Amazon EC2 instance. [3]

2. Write a Java program to monitor the status of your EC2 instances.

(a) For example, CPU utilization, disk I/O, etc.

```
Enter the instance ID: i-0c0b94218b28eb80c
Please select a metric

Select an option:
1 - EBSIOBalance%
2 - EBSByteBalance%
3 - EBSReadOps
4 - EBSReadBytes
5 - EBSWriteOps
6 - EBSWriteBytes
7 - CPUUtilization
8 - NetworkPacketsIn
9 - NetworkPacketsOut
10 - MetadataNoToken
11 - NetworkIn
12 - NetworkOut
13 - CPUCreditBalance
14 - CPUCreditUsage
15 - CPUSurplusCreditBalance
16 - CPUSurplusCreditsCharged
17 - StatusCheckFailed_System
18 - StatusCheckFailed_Instance
19 - StatusCheckFailed
```

Figure 3: Cloud Watch Menu

User has the option to select of the metrics from the above list to get the metrics data. Once the choice is made, a graph is displayed as shown:

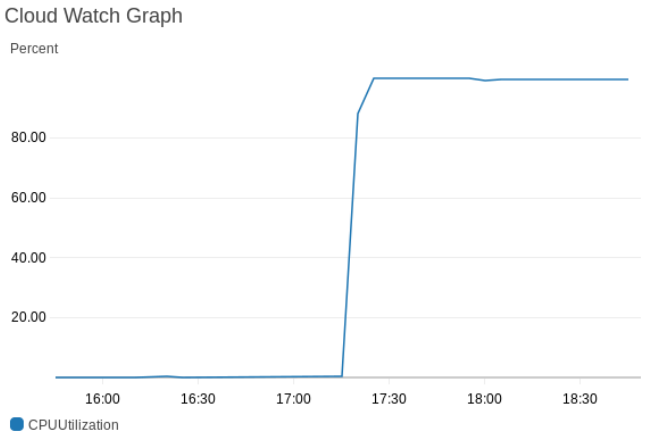

Figure 4: CPU Utilization Graph

Figure 4 shows the CPU Utilization graph for the particular instance over a 3 Hour period.

Web Interface

To create and manage EC2 instances we created a Web Application using Flask which allows a user to:

1. Create an instance
2. List the region names
3. Get a list of all the created instances
4. Start, Stop or Delete an instance
5. See CloudWatch metrics for an instance

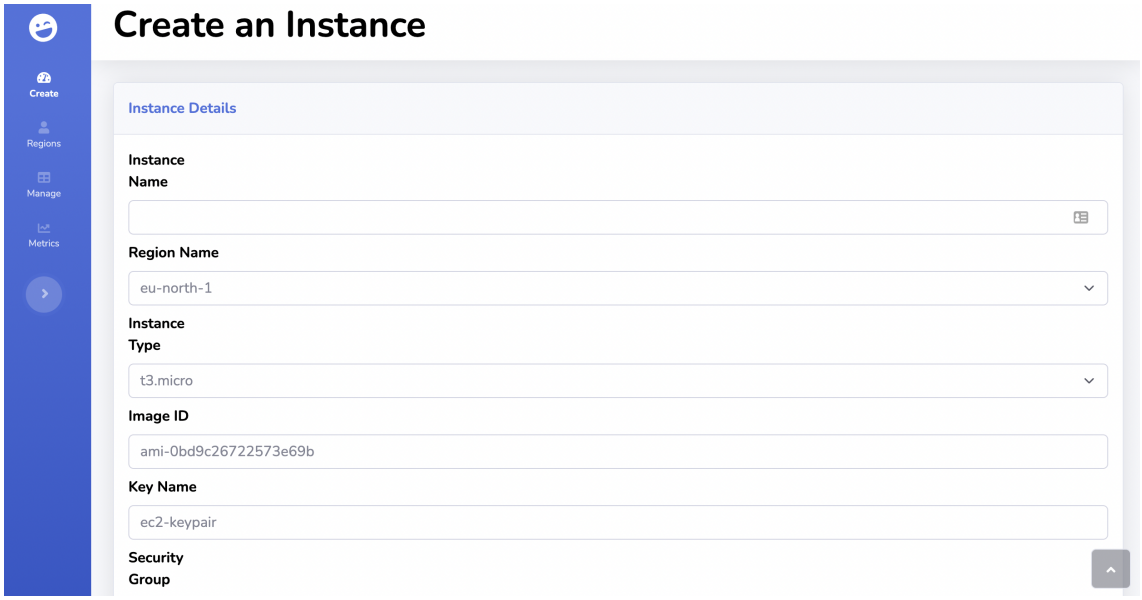

Figure 5: Create an new EC2 Instance

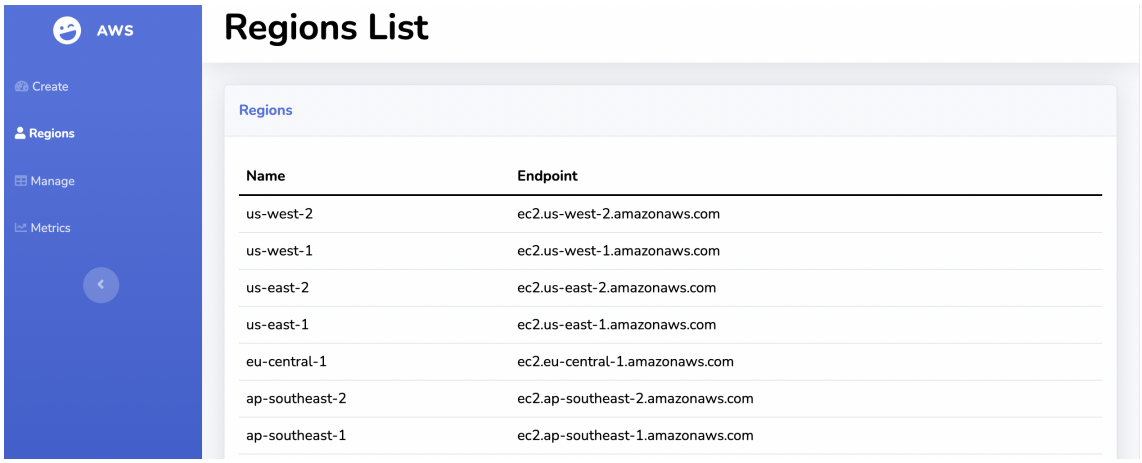

Figure 6: List of available regions

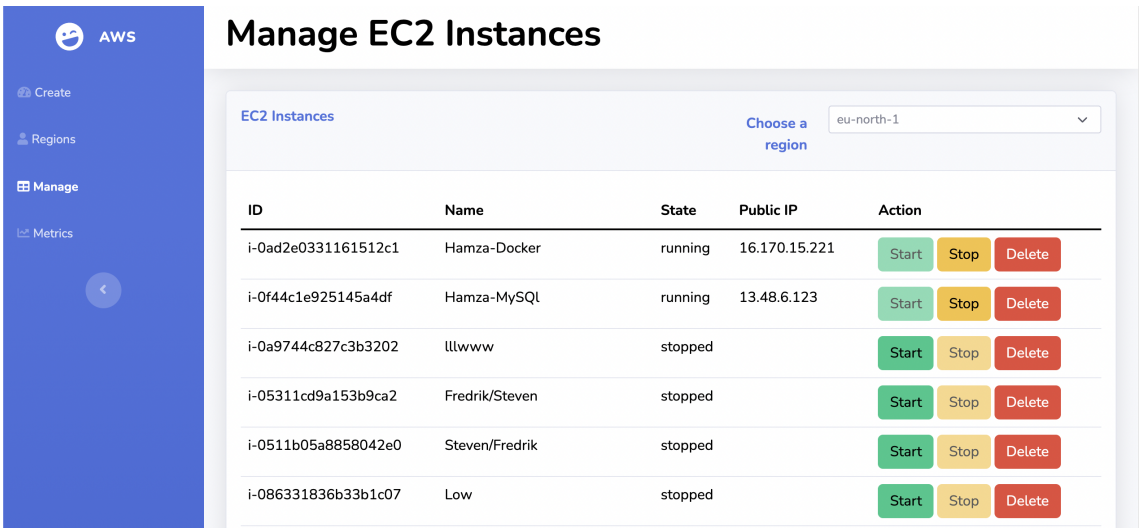

Figure 7: Manage existing instance

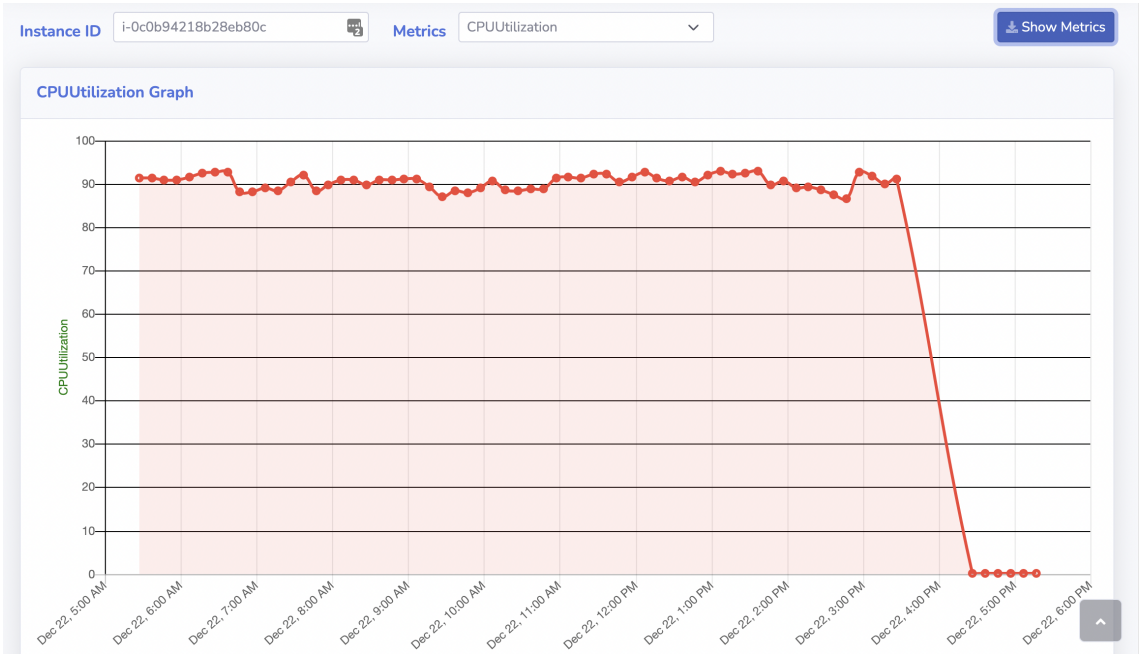

Figure 8: CPU Utilization CloudWatch Metrics

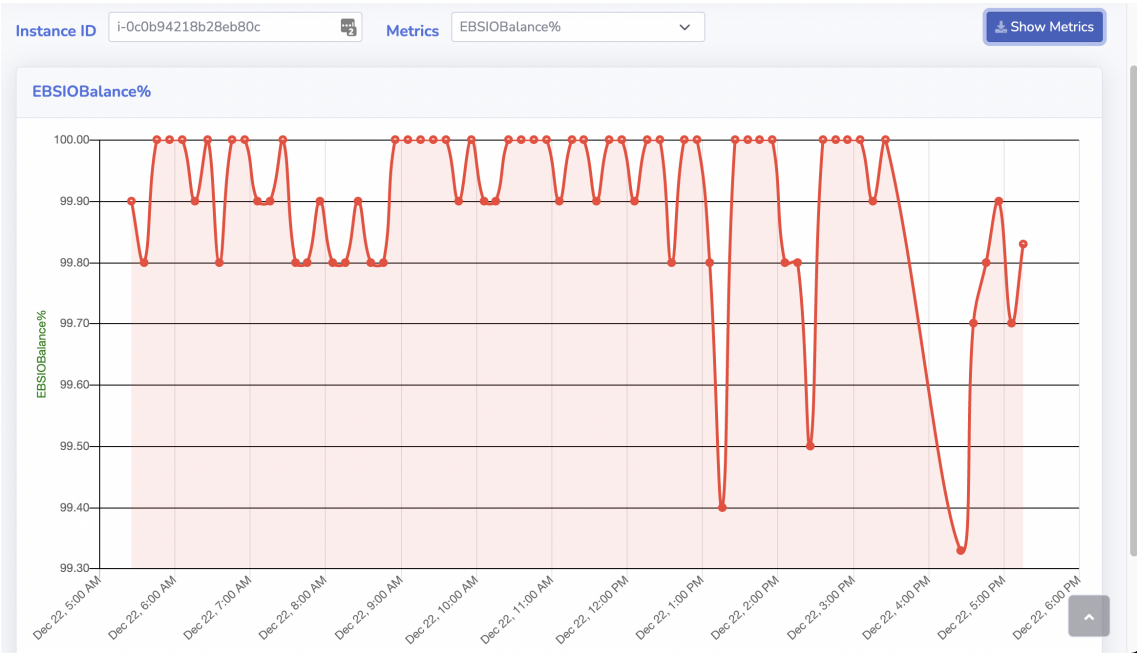

Figure 9: EBSIOBalance CloudWatch Metrics

References

[1] "Createinstance - amazon ec2 web service." [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html

[2] "Dependencies diagram." [Online]. Available: https://giuseppeborgese.medium.com/read-aws-cloudwatch-metrics-from-boto3-d245673dff79

[3] "Monitorinstance - amazon cloudwatch." [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/cloudwatch.html