

CMPE 281 -spring 2018 - Assignment- 1  
Sampath Lakkaraju  
011818781

Attached video files along.

## Simple Website:

For the application I have chosen a simple “Hello” web page hosted with Node.js and Express framework.

The EC2 in the AWS(Amazon Web Services) has been setup using the following steps:

- For the Amazon Machine Image, I selected the ubuntu server option.
- For the instance I selected the General purpose t2.micro instance with 1 CPU and 1 GiB (gigabytes) of memory.
- The default configuration has been used.
- A storage of 8 GiB SSD has been used. No tags were applied to the instance.
- The Security Group was configured to allow internet traffic by adding HTTP and HTTPS rules the group. Additionally, port 8080 and 9000 were also configured to be used. All the group sources were set to “Anywhere” to make the configuration simple.

With the above configurations were launched. Once launched a key pair file can be generated in the console which is used to connect to the server. The encrypted key file must be modified to read mode this can be done using the command:

```
chmod 600 sam11818781.pem
```

Once the file is in read mode, the server can be accessed using the terminal and the key file.

The command can be found in the connect property of the particular instance, under the standalone SSH client option.

My personal instance required the following command:

```
ssh -i "sam11818781.pem" ubuntu@ec2-13-57-227-94.us-west-1.compute.amazonaws.com
```

This will connect you to the created instance command line. From now on all the required configurations that has made into the instance can be performed from this window.

Since the Image chosen is a ubuntu, update apps and include required libssl. These can be performed using the below commands:

```
sudo apt-get update
```

```
sudo apt-get install libssl-dev g++ make
```

After the required updates are installed, install python:

```
sudo apt-get install python
```

Install node.js on the instance using the following commands:

```
wget https://nodejs.org/dist/v8.9.4/node-v8.9.4.tar.gz
tar -xvf node-v0.10.32.tar.gz
cd node-v0.10.32
./configure && make && sudo make install
```

Create a test file to check is the installation is correct.

```
cd ../
mkdir app1
cd app1
touch app1.js
vim app1.js
```

Insert the following code into the app1.js file

```
***** app1.js

var http = require('http');
var port = 9000;
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello there, I am Sampath Lakkaraju\n');
}).listen(port);
console.log('Listening on port', port);

***** app1.js
```

Exit the app.js file using esc key.

Run the app using the command

```
node app.js
```

Using the Public DNS of the instance from the Description tab in the instance details. Since we have used the port 9000 add “:9000” after the DNS.

Use the DNS & :9000 in any browser to view the site. Once checking the site the server can be stopped using the Cntrl +C command twice in the command line.

Now install the express in a new directory in the home/ubuntu directory using the command:

```
sudo npm install express
```

open a new file for the express script and fill in the file with express render code:

```
mkdir expapp2  
cd expapp2  
touch expapp2.js  
vim test2.js
```

```
***** expapp2.js  
  
var express = require('express');  
var app = express();  
app.get('/', function (req, res){  
  res.send('Hello there, I am Sampath Lakkaraju using Express\n');  
});  
app.get('/test', function (req, res){  
  res.send('Hello there, I am Sampath Lakkaraju using Express this is a  
test \n');  
});  
var port = 9000;  
app.listen(port);  
console.log('Listening on port', port);
```

```
***** expapp2.js
```

This code can be run using the same commands for the app1 but modify the file name:

```
node expapp2.js
```

The server starts running if there are no issues in the code.

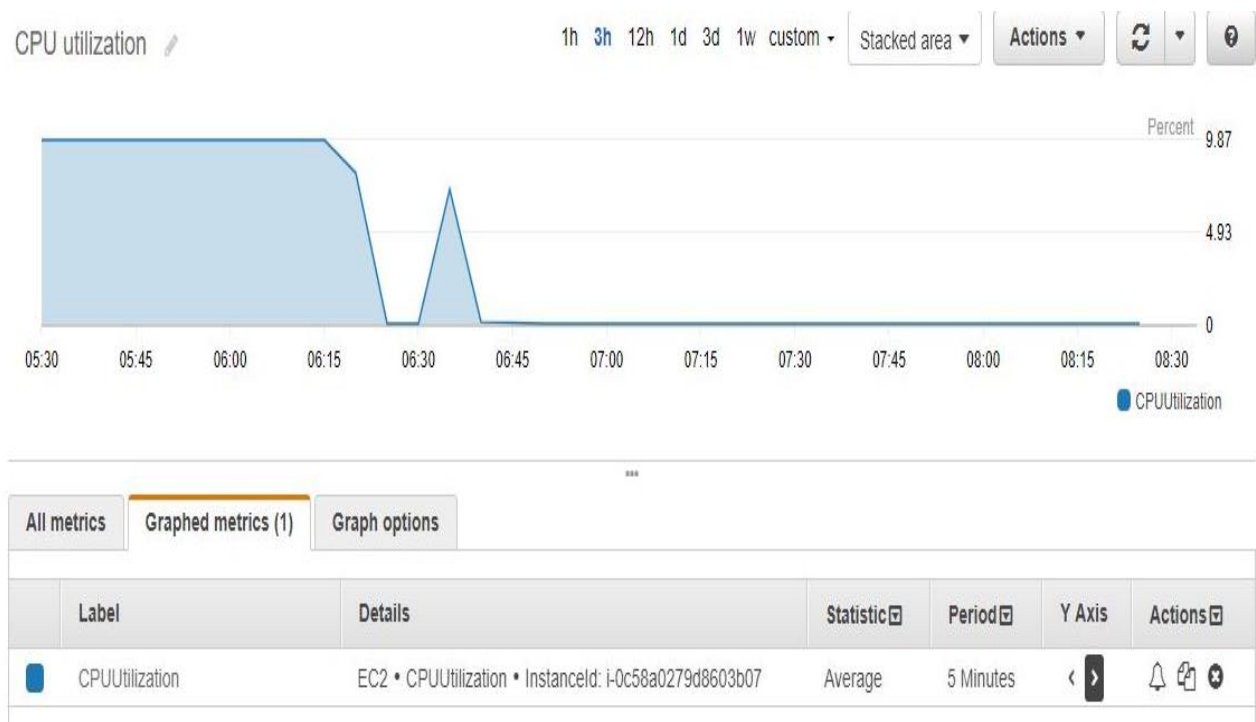
If you visit the DNS &":9000" you can check the website.

DNS\$:9000/test" will redirect you to a different page.

The Outputs have been displayed in the video.

Performance monitoring through Cloud Watch:

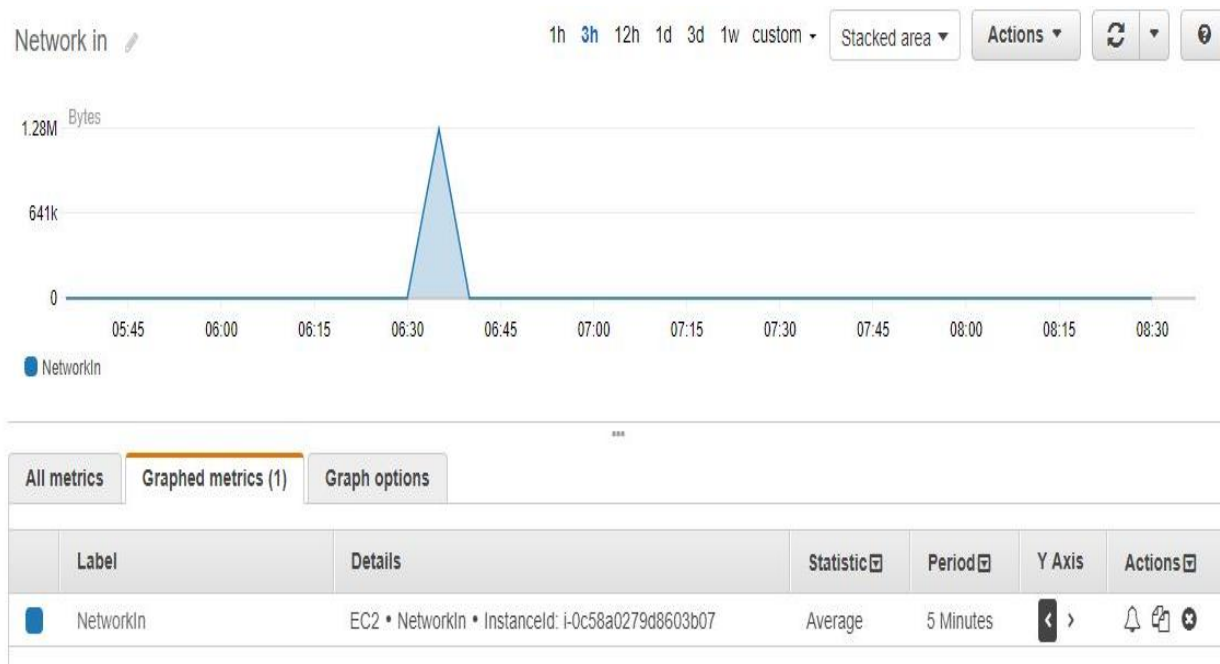
CPU Utilization:



Network out:



Network in:



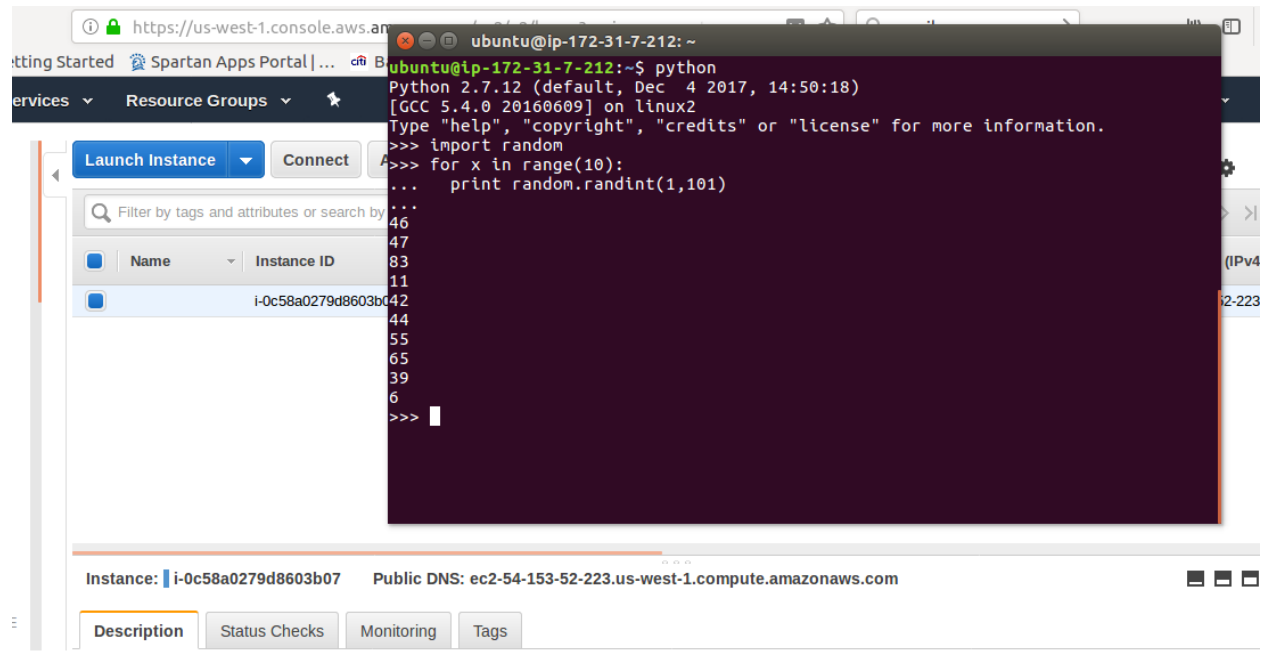
## Simple Python Script

For the Second application I have run a simple python script which generates random numbers.

Python and Pip were installed in the instance and the following script was run:

```
import random
for x in range(10):
    print random.randint(1,101)
```

the output:

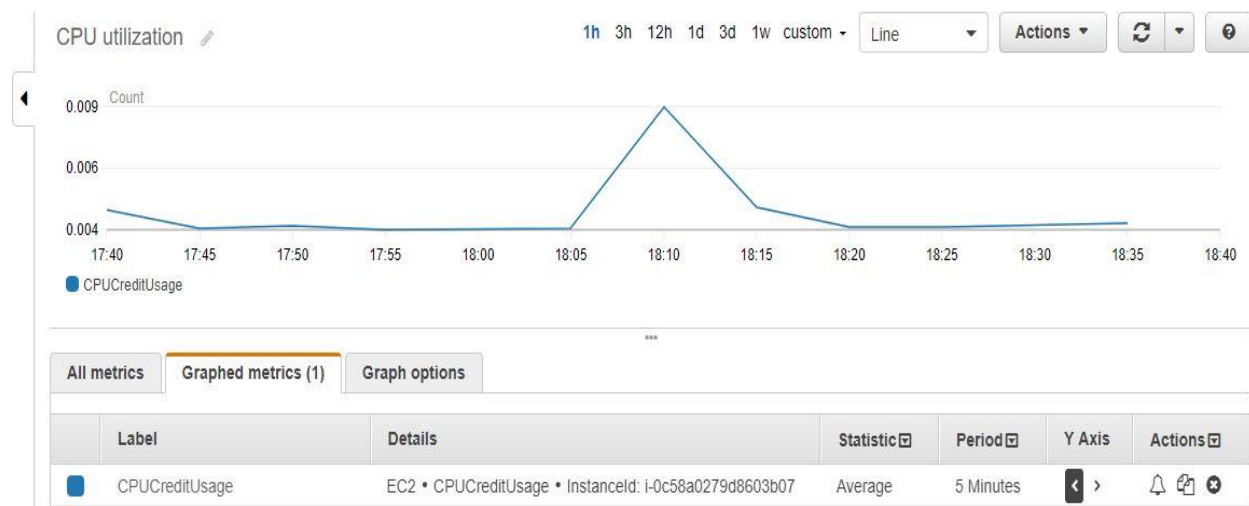


The screenshot shows the AWS Management Console for an EC2 instance named 'i-0c58a0279d8603b07'. A terminal window is open, showing the execution of a Python script. The script imports the 'random' module and uses a 'for' loop to generate 10 random numbers between 1 and 101. The output of the script is displayed in the terminal window.

```
ubuntu@ip-172-31-7-212: ~$ python
Python 2.7.12 (default, Dec 4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import random
>>> for x in range(10):
...     print random.randint(1,101)
...
46
47
83
11
42
44
55
65
39
6
>>>
```

Performance monitoring through Cloud Watch:

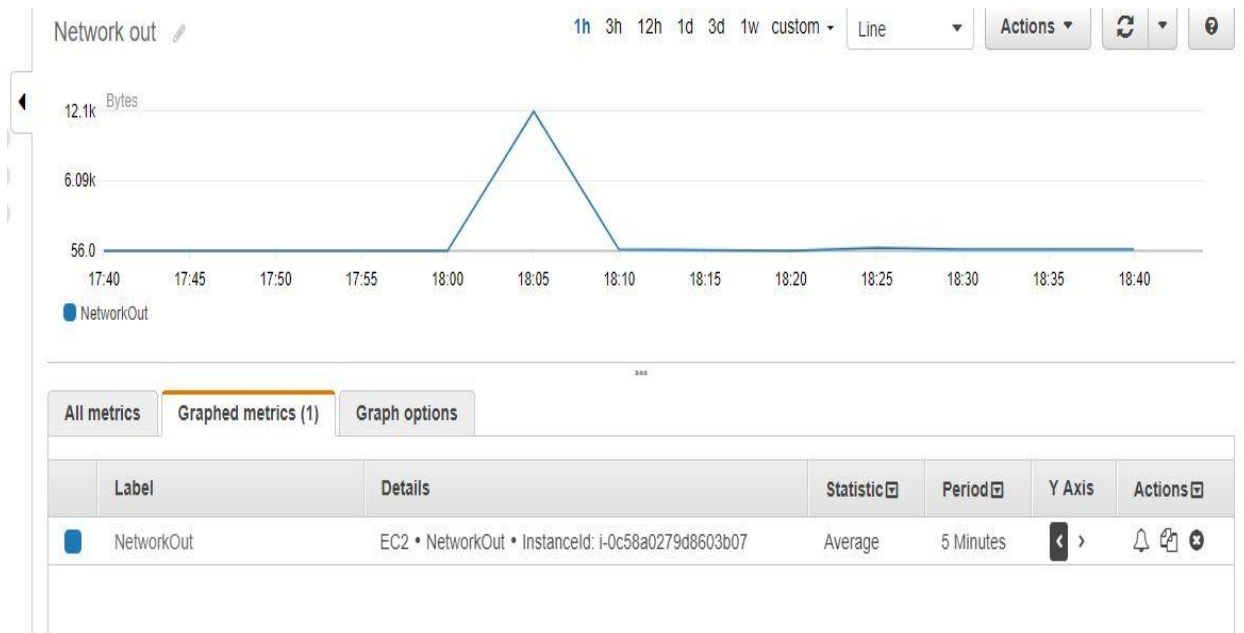
CPU Utilization:



Network in:



Network Out:



References:

1. <https://aws.amazon.com/ec2/getting-started/>
2. <https://www.youtube.com/watch?v=WxhFq64FQzA>
3. <https://www.pythoncentral.io/how-to-generate-a-random-number-in-python/>