

CLOUD COMPUTING TASK 3

Group 30 - Bejjagam Lokesh, Chikkala Naga Veera Sairam and Saif Ahmed

To achieve the 3rd task, we had to go through a certain step-by-step procedure. These are more thoroughly explained in the screen recording provided. The steps are as follows:

>> We start by creating 2 VM instances named VM1 and VM2.

The screenshot shows the AWS EC2 Instances page with the following details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
VM1	i-0b65b010779b5b46e	Terminated	t2.micro	-	No alarms	us-east-1c	-
VM2	i-09c04a1d5f13bd264	Terminated	t2.micro	-	No alarms	us-east-1c	-
VM1	i-0a2dcdf9396fdb26	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-54-197-88-2
VM2	i-0a970211b93c7fc87	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-54-163-23-1

A modal window titled "Select an instance" is open at the bottom, indicating the user needs to choose an instance to proceed.

>> Now create a private IP add to VM2 so that we can connect it to VM1.

>> To do that, we need to go to 'Manage IP addresses' and assign a new private IP.

The screenshot shows the "Manage IP addresses" page for instance i-0a970211b93c7fc87. The interface includes the following sections:

- EC2 > Instances > i-0a970211b93c7fc87 > Manage IP addresses**
- Manage IP addresses** (Info): A note states, "Assign or unassign IPv4 and IPv6 addresses to or from an instance's network interfaces." A callout box says, "To assign additional public IPv4 addresses to this instance, you must [allocate](#) Elastic IP addresses and associate them with the instance or its network interfaces."
- IPv4 addresses**: A table showing current assignments:

Private IP address	Public IP address	Action
172.31.81.174	54.163.23.17	Unassign
- Actions**: Buttons for "Assign new IP address" and "Undo".

>> We can always cross-check this IP implementation from the instances dashboard.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
VM1	i-0b65b010779b5b46e	Terminated	t2.micro	-	No alarms	us-east-1c	-
VM2	i-09c04a1d5f13bd264	Terminated	t2.micro	-	No alarms	us-east-1c	-
VM1	i-0a2dc6f9396fdb26	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-54-197-88-2
VM2	i-0a970211b93c7fc87	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-54-163-23-1

>> Now we will connect our VM1 through SSH Client using GitHub. after the connection is established it looks like this:

EC2 > Instances > i-0a2dc6f9396fdb26 > Connect to instance

Connect to instance [Info](#)

Connect to your instance i-0a2dc6f9396fdb26 (VM1) using any of these options

EC2 Instance Connect | Session Manager | **SSH client** | EC2 Serial Console

Instance ID: i-0a2dc6f9396fdb26 (VM1)

- Open an SSH client.
- Locate your private key file. The key used to launch this instance is ApacheKp.pem
- Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 ApacheKp.pem
- Connect to your instance using its Public DNS:
7-88-22.compute-1.amazonaws.com

Command copied

```
ec2-user@ip-172-31-91-172:~$ 
[okesh@DESKTOP-DTSB59I MINGW64 ~/Downloads]$ 
[okesh@DESKTOP-DTSB59I MINGW64 ~/Downloads]$ 
[okesh@DESKTOP-DTSB59I MINGW64 ~/Downloads]$ ssh -i "ApacheKp.pem" ec2-user@ec2-54-197-88-22.compute-1.amazonaws.com
The authenticity of host 'ec2-54-197-88-22.compute-1.amazonaws.com (54.197.88.22)' can't be established.
ED25519 key fingerprint is SHA256:URPVNSRJkhckKgXOxi9RLxtlyv/KmrDX80BraulSOo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
warning: Permanently added 'ec2-54-197-88-22.compute-1.amazonaws.com' (ED25519)
to the list of known hosts.

[okesh@DESKTOP-DTSB59I MINGW64 ~/Downloads]$ https://aws.amazon.com/amazon-linux-2/
[okesh@DESKTOP-DTSB59I MINGW64 ~/Downloads]$ 6 package(s) needed for security, out of 14 available
[okesh@DESKTOP-DTSB59I MINGW64 ~/Downloads]$ Run "sudo yum update" to apply all updates.
[okesh@DESKTOP-DTSB59I MINGW64 ~/Downloads]$ [ec2-user@ip-172-31-91-172 ~]$
```

ssh -i "ApacheKp.pem" ec2-user@ec2-54-197-88-22.compute-1.amazonaws.com

>> Similarly we will connect VM2 using Git as well but on another terminal:

The screenshot shows the AWS Management Console with the 'SSH client' tab selected. The instance ID is listed as i-0a970211b93c7fc87 (VM2). A step-by-step guide for connecting via SSH is provided:

- Open an SSH client.
- Locate your private key file. The key used to launch this instance is ApacheKp.pem.
- Run this command, if necessary, to ensure your key is not publicly viewable:


```
chmod 400 ApacheKp.pem
```
- Connect to your instance using its Public DNS:


```
ssh -i "ApacheKp.pem" ec2-user@ec2-54-163-23-17.compute-1.amazonaws.com
```

A note in the terminal window states: "Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name."

>> Then we need to install Apache HTTPD. and after installation, we need to start it.

```

root@ip-172-31-91-172:/home/ec2-user
Verifying : mailcap-2.1.41-2.amzn2.noarch                                         6/9
Verifying : generic-logos-httpd-18.0.0-4.amzn2.noarch                            7/9
Verifying : mod_http2-1.15.19-1.amzn2.0.1.x86_64                                8/9
Verifying : httpd-tools-2.4.51-1.amzn2.x86_64                                     9/9

Installed:
  httpd.x86_64 0:2.4.51-1.amzn2

Dependency Installed:
  apr.x86_64 0:1.7.0-9.amzn2
  apr-util.x86_64 0:1.6.1-5.amzn2.0.2
  apr-util-bdb.x86_64 0:1.6.1-5.amzn2.0.2
  generic-logos-httpd.noarch 0:18.0.0-4.amzn2
  httpd-filesystem.noarch 0:2.4.51-1.amzn2
  httpd-tools.x86_64 0:2.4.51-1.amzn2
  mailcap.noarch 0:2.1.41-2.amzn2
  mod_http2.x86_64 0:1.15.19-1.amzn2.0.1

Complete!
[root@ip-172-31-91-172 ec2-user]# sudo systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-91-172 ec2-user]# sudo systemctl start httpd
[root@ip-172-31-91-172 ec2-user]#

```

>> After the start it is possible for us to view our default apache website.

The screenshot shows a web browser displaying the Apache test page. The URL is 54.197.88.22. The page content is as follows:

If you are a member of the general public:
 The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

If you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

If you are the website administrator:
 You may now add content to the directory /var/www/html/. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file /etc/httpd/conf.d/welcome.conf.

You are free to use the image below on web sites powered by the Apache HTTP Server:

2.4

>> We need to do the same for VM2 as well right until we are able to view the apache website as done in the previous steps.

>> Our main objective in this task is to stress VM2 by running it and monitoring its performance using VM1.

>> To achieve this objective we need to connect both our VMS using our private IP. We create an Elastic IP add and attach it to VM@ ec2 instance.

Elastic IP addresses (1/1)						
<input type="button" value="Actions"/> <input type="button" value="Allocate Elastic IP address"/>						
<input checked="" type="checkbox"/>	Name	Allocated IPv4 add...	Type	Allocation ID	Reverse DNS record	
<input checked="" type="checkbox"/>	VM2	52.70.205.45	Public IP	eipalloc-0a53a1a04525e7efb	-	

>> Now we establish a connection between VM1 and vm2 using the private ipv4 ad of vm2. After the connection, we can see that both the ec2 instances are on the same private ipv4 ad.

```
ec2-user@ip-172-31-90-199:~ Connection to ec2-54-197-66-156.compute-1.amazonaws.com closed.
lokesha@DESKTOP-DTSB59I MINGW64 ~/Downloads
$ ssh -i "Myapachekp.pem" ec2-user@ec2-54-197-66-156.compute-1.amazonaws.com
Last login: Tue Dec 14 07:56:48 2021 from 223.230.96.204
      _\   _ ) 
     _\ ( _ /  Amazon Linux 2 AMI
      \_\|_|_|
https://aws.amazon.com/amazon-linux-2/
6 package(s) needed for security, out of 14 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-90-199 ~]$ nano
[ec2-user@ip-172-31-90-199 ~]$ ssh -i kp ec2-user@172.31.80.55
oooooooooooooooooooooooooooooooooooo
@       WARNING: UNPROTECTED PRIVATE KEY FILE!       @
oooooooooooooooooooooooooooooooooooo
Permissions 0664 for 'kp' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "kp": bad permissions
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-90-199 ~]$
```

The screenshot shows two terminal windows side-by-side. The left window is titled 'ec2-user@ip-172-31-80-55:' and displays the process of generating an RSA key pair and setting permissions for it. The right window is titled 'ec2-user@ip-172-31-80-55:' and shows the deployment of an Apache web server, including the creation of a self-signed certificate and the configuration of the Apache service.

```

ec2-user@ip-172-31-80-55:~$ ssh -i kp ec2-user@172.31.80.55
WARNING: UNPROTECTED PRIVATE KEY FILE!
Permissions 0664 for 'kp' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "kp": bad permissions
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-90-199 ~]$ chmod 600 kp
[ec2-user@ip-172-31-90-199 ~]$ ssh -i kp ec2-user@172.31.80.55
Last login: Tue Dec 14 07:56:51 2021 from 223.230.96.204
[ec2-user@ip-172-31-80-55 ~]$ | _\|_/_ Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
6 package(s) needed for security, out of 14 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-80-55 ~]$ | _\|_/_ Amazon Linux 2 AMI

Instance type: t2.micro
80%   6
90%   6
95%   7
98%   8
99%   8
100%  25 (longest request)
[ec2-user@ip-172-31-80-55 website]# exit
t2.micro
exit
[ec2-user@ip-172-31-80-55 ~]$ exit
logout
Connection to ec2-54-197-200-25.compute-1.amazonaws.com closed.

jokesh@DESKTOP-DTSB591 MINGW64 ~/Downloads
$ ssh -i "Myapachekp.pem" ec2-user@ec2-54-197-200-25.compute-1.amazonaws.com
Last login: Tue Dec 14 07:43:22 2021 from ip-172-31-90-199.ec2.internal
[ec2-user@ip-172-31-80-55 ~]$ | _\|_/_ Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
6 package(s) needed for security, out of 14 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-80-55 ~]$ | _\|_/_ Amazon Linux 2 AMI

```

>> After the connection is established, we use the ApacheBench stress test to test the VM. The command used to stress it was:

'ab -n 10000 -c 50 http://54.197.200.25/website/index.html'

>> after this command was run, we get the following performance results.

The screenshot shows the output of the ApacheBench (ab) command. It provides detailed performance metrics for the stress test, including the number of requests per second, the time taken for tests, and various connection times (Connect, Processing, Waiting, Total). It also includes a histogram of response times and a warning about processing times being outside the normal deviation.

```

Server Software:      Apache/2.4.51
Server Hostname:     54.197.200.25
Server Port:          80

Document Path:        /website/index.html
Document Length:     125 bytes

Concurrency Level:   50
Time taken for tests: 5.116 seconds
Complete requests:   10000
Failed requests:     0
Total transferred:   4070000 bytes
HTML transferred:    1250000 bytes
Requests per second: 1954.68 [#/sec] (mean)
Time per request:   25.580 [ms] (mean)
Time per request:   0.512 [ms] (mean, across all concurrent requests)
Transfer rate:       776.91 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    7   3.6    6   28
Processing:     2   18   3.8   18   38
Time per request: 0.541 [ms] (mean, across all concurrent requests)
Transfer rate:  734.40 [Kbytes/sec] received

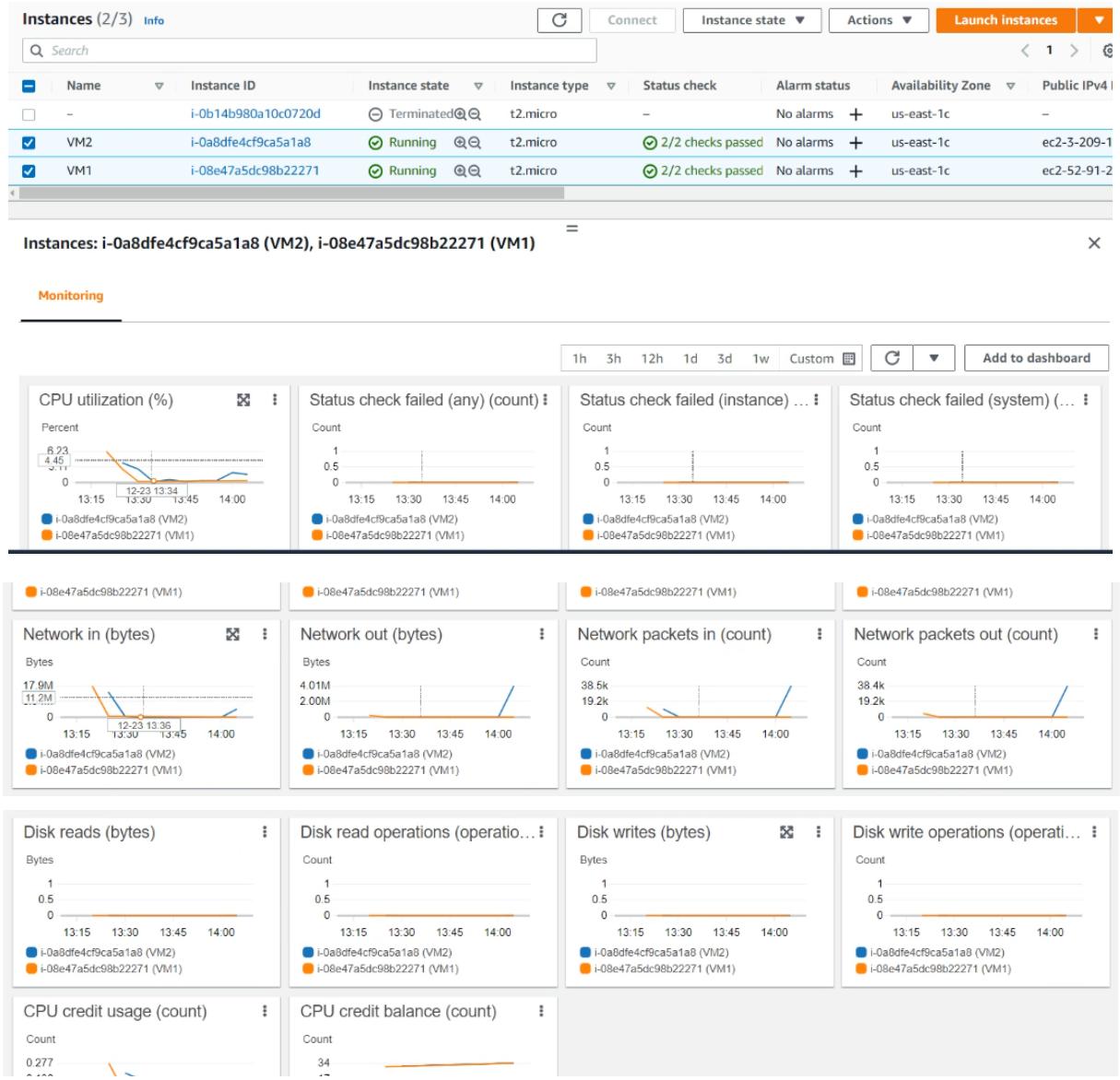
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        1    1   0.5    1   11
Processing:     1    2   0.5    1   11
Waiting:        0    1   0.5    1   11
Total:         1    3   0.8    3   21
WARNING: The median and mean for the processing time are not within a normal deviation
These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
 50%   3
 66%   3
 75%   3
 80%   3
 90%   3
 95%   4
 98%   4
 99%   5
100%  21 (longest request)
[ec2-user@ip-172-31-80-55 ~]$ |

```

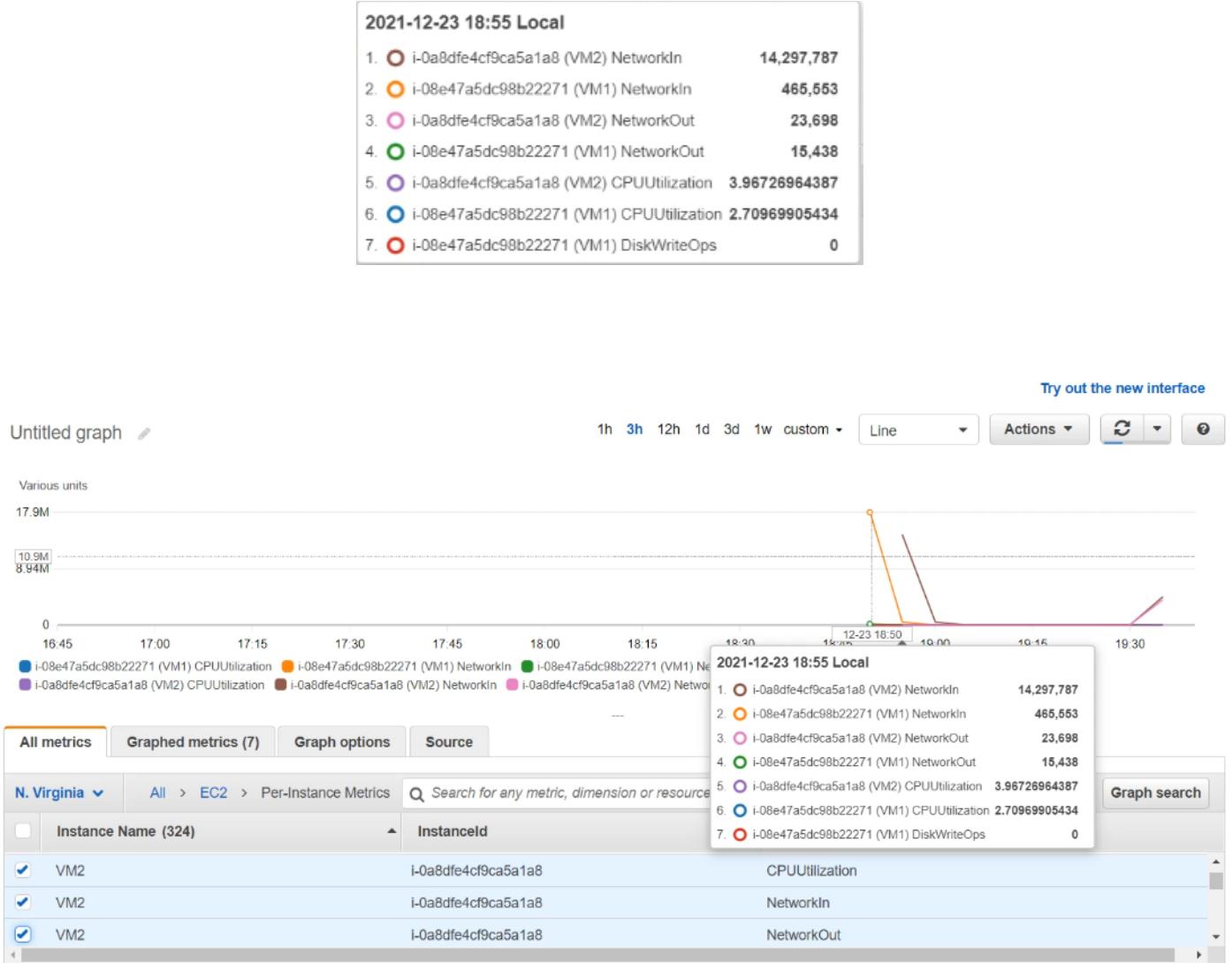
From here we redo a part of Task 3 based on the comments from our previous submission. It is worth noting that, new VM's were created to make this part due to the fact that the original/previous ones were terminated after completion. Hence the IP address will be different, but the main concept still applies.

>> Now after achieving the results from the ApacheBench stress test, we can monitor them through our EC2 instances. From there we can monitor all the stats for our VM's.

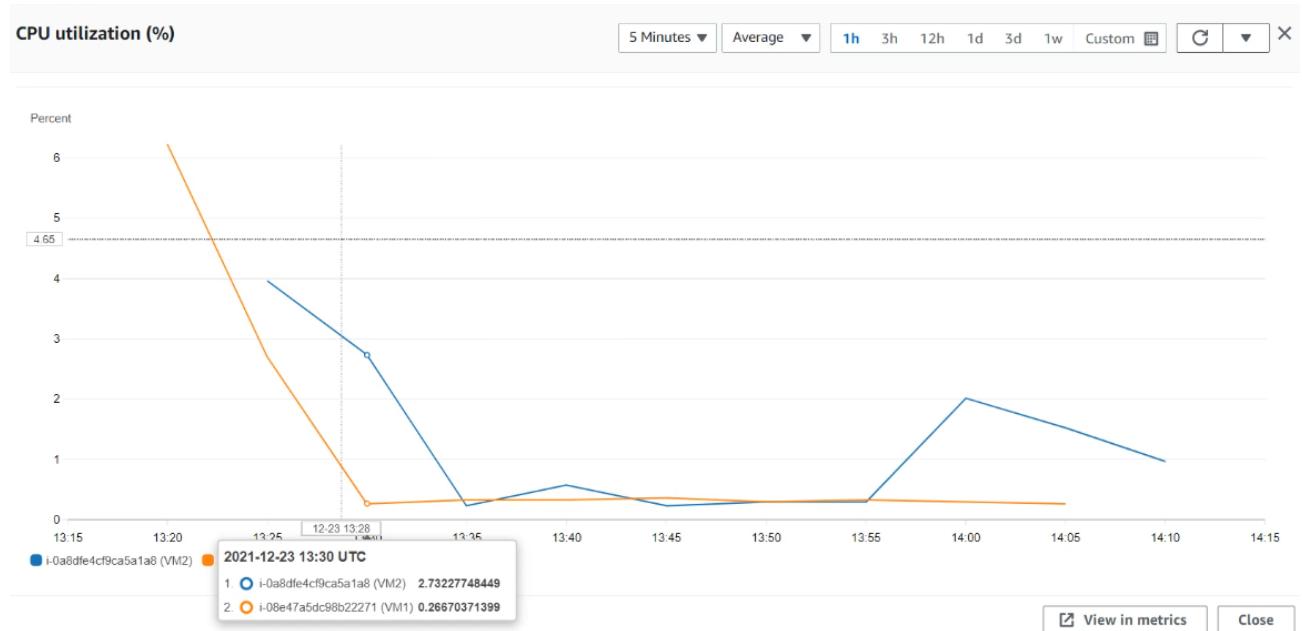


>> We should also note that this isn't the only viewing/monitoring option we have. To have a better look at the performance graphs, we can use CloudWatch.

>> CloudWatch allows us to compare the performance of any combination parameters between the VM's. The image below illustrates that. The performance graphs based on the following parameters would look like this:



>> We can also compare a specific parameter of such as CPU Utilization between VM1 and VM2 simultaneously.



In this CPU Utilization graph, the orange line is VM1 and the blue line is VM2.