



SCHOOL OF COMPUTER SCIENCES

UNIVERSITI SAINS MALAYSIA

CST333: DISTRIBUTED & GRID COMPUTING

Semester 1, Academic Session 2021/2022

Assignment 2: Highly Available Auto Scaling Web Server

NAME	MATRIC NO
Muhammad Faikal Izham bin Abdul Rahim	141969
Farhana Zulaikha binti Fadzli	143949

Lecturer: Prof. Madya Dr. Chan Huah Yong

Date of Submission: 31th January 2022

TABLE OF CONTENTS

1	INTRODUCTION	1
2	PROJECT ENVIRONMENT CONFIGURATION.....	2
2.1	Services used:.....	2
2.2	Project setting (including source code):	4
3	INPUT AND OUTPUT.....	15
4	WORK DIVISION	16
5	LESSONS LEARNED	17
6	CONCLUSION.....	17
7	REFERENCES.....	18

1 INTRODUCTION

Cloud computing has been on demand ever since the launch of Amazon Web Services (AWS) in 2006. With multiple benefits and uses offered to the users, AWS has been used by millions of people worldwide.

One of the uses of AWS is to deploy a highly available web server. A web server basically provides contents to its users using HTTP request and response via the internet [1].

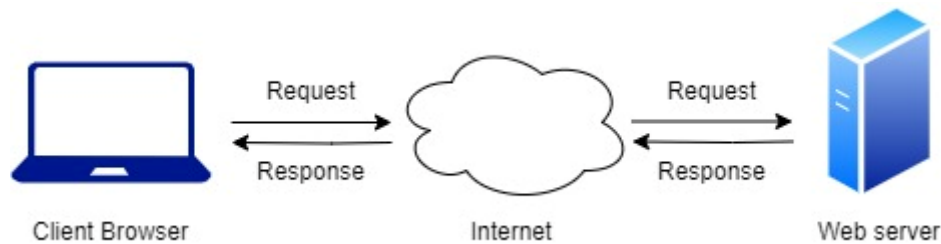


Figure 1.1. Web server request and response

A highly available web server means that any failure that the server goes through can be recovered from. For example, if one server goes down, it will automatically be replaced with another healthy server for traffic rerouting. In this assignment, AWS' Elastic Load Balancer (ELB) is used to distribute the traffic load and conduct regular health checks on each of the Elastic Cloud Compute (EC2) instances [2]. If any of the instances does not respond, then Auto Scaling Group will automatically create a new instance to replace the unhealthy instance [3]. Apache web server is used in this assignment to host the website.

Section 2, *Project Environment Configuration* contains the codes and configuration needed to create the highly available web server. This includes the explanation on all AWS services used for the web server creation. These processes are all done in AWS console.

Section 3, *Input and Output* presents the screenshots of the configuration result.

Section 4, *Lesson Learned* consists of the knowledge we acquired throughout the process of the project.

Section 5, *Conclusion* summarizes the project and our future plans.

2 PROJECT ENVIRONMENT CONFIGURATION

2.1 Services used:

- Elastic Cloud Compute (EC2)

Elastic Compute Cloud (EC2) provides scalable instance launcher for applications [4]. This service lets users run their resources on cloud, eliminating the need for hardware. Many instance types are available to be chosen from, such as memory size and communication types. These can be scaled according to user's needs. EC2 is used in this assignment for hosting the web server.

- Virtual Private Cloud (VPC)

Virtual Private Cloud (VPC) allows users to configure their own virtual network including Internet Protocol (IP) address, subnets and gateways [4]. User creates as many security layers as they can. In this assignment, the web is launched using the virtual network defined by VPC.

- Elastic Load Balancer (ELB)

Elastic Load Balancing (ELB) scales incoming traffic across various instances [4]. ELB increases fault tolerance by eliminating unhealthy EC2 instances and rerouting the traffic to a healthy instance. ELB is applied in this assignment to manage the traffic of the instances.

- Auto Scaling Group

Auto Scaling Group allows user to manage their application freely through scaling it according to their needs [4]. A cluster of EC2 instances will be defined by user. This assignment uses Auto Scaling to monitor the application and scale the EC2 for better performance.

Figure 2.1 shows how a web server that is highly available works [5].

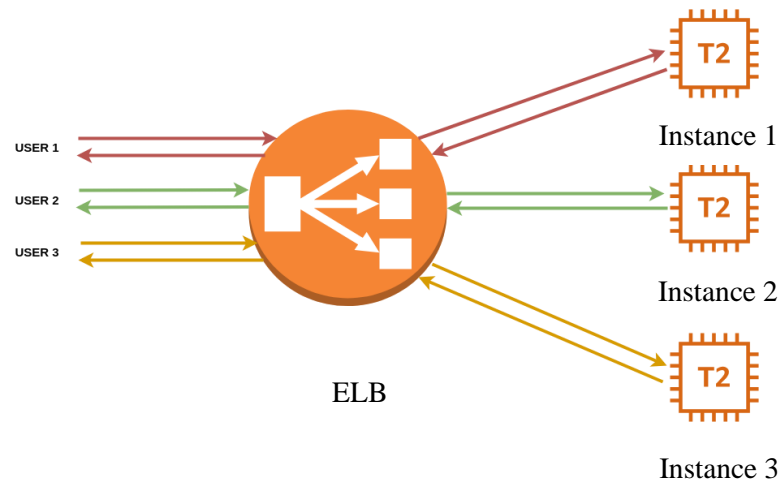


Figure 2.1. Highly available web server

Elastic Load Balancer will be the front server that will forward all the internet traffic to the instances of the application. ELB reroutes traffic to the instances, and the instances will send back a response to ELB. ELB will then send the response back to the user.

ELB will distribute the load across the available instances. From Figure 2.1, we can see that User 1 connects to Instance 1, while User 2 and User 3 connect to Instance 2 and Instance 3, respectively.

ELB will do regular health checks to each of the instances. If one instance is unhealthy or goes down, the load balancer will reroute the traffic to another healthy instance. Auto Scaling Group can be used to scale the instances up and down according to the user's needs. Figure 2.2 below shows when the user scales up the instances [6]. The load balancer will distribute all traffic to all six instances instead of the original three instances.

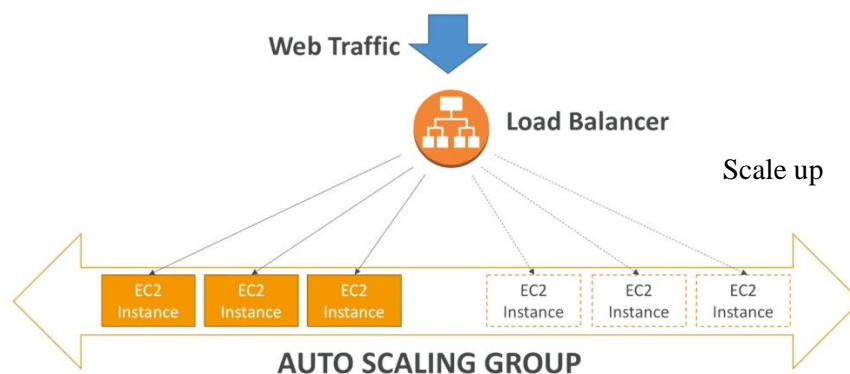


Figure 2.2. Auto Scaling Group

2.2 Project setting (including source code):

Running the web server requires EC2, VPC, ELB and Auto Scaling Group to be configured together. The detailed steps to launch a web server with high availability are explained as below:

1. Open the AWS console and choose **VPCs** from the list of resources given.

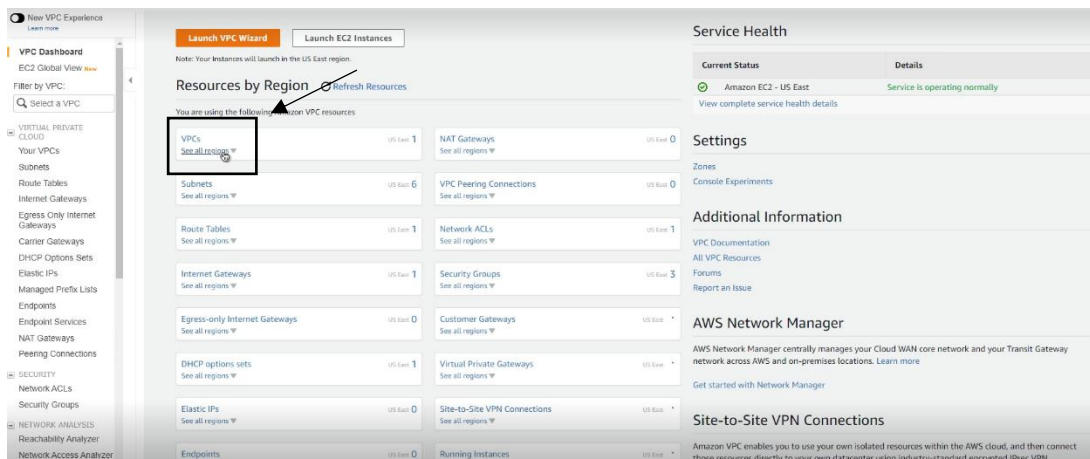


Figure 2.3. Project configuration screenshot 1

2. Select **Actions**, then **Create Default VPC**.

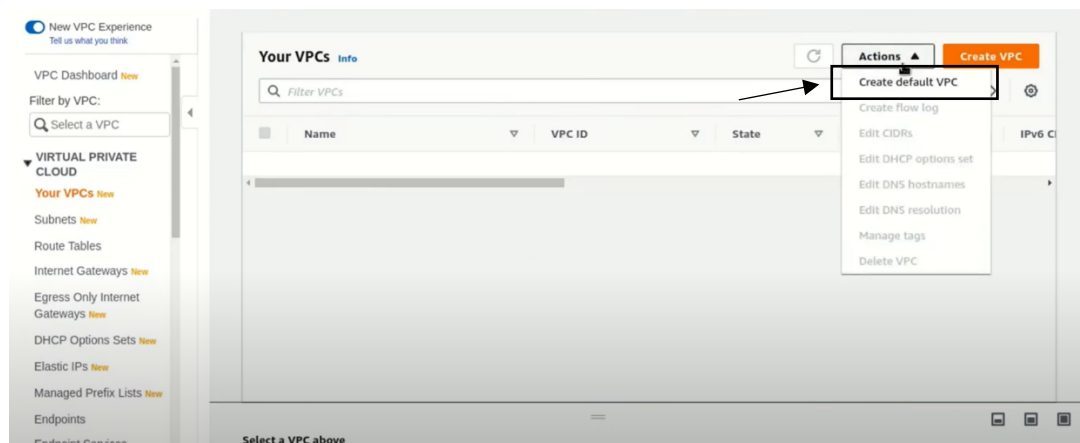


Figure 2.4. Project configuration screenshot 2

3. The default VPC created will have standard configurations and cannot be changed. Click **Create Default VPC**.
4. Navigate to EC2 console and choose **Load Balancers** from the navigation pane. Then, select **Create New Load Balancer**.

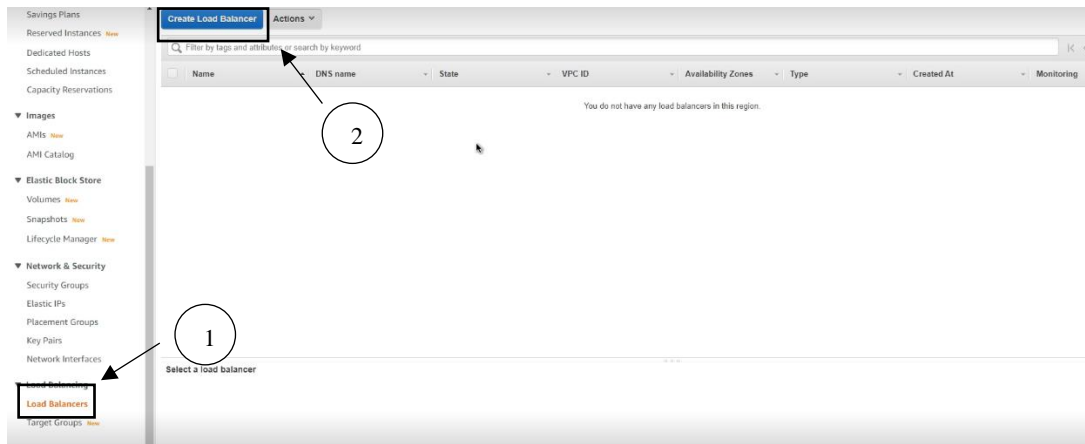


Figure 2.5. Project configuration screenshot 3

5. Select **Application Load Balancer** to create a new load balancer.

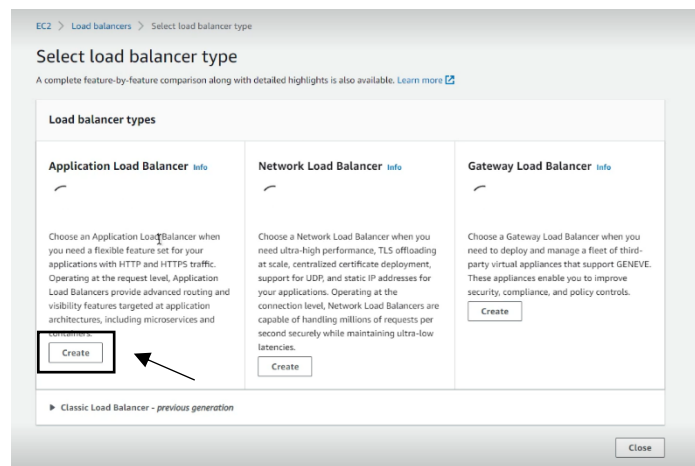


Figure 2.6. Project configuration screenshot 4

6. Use these configurations for the load balancer:

- Load balancer name: TestLoadBalancer (can be any name but must be unique)
- Scheme: Internet-facing
- IP address type: IPv4
- VPC: Default VPC
- Mappings: us-east-1a, us-east-1b
- Create a new Security Group for the load balancer. Choose this security group. The configurations for the new security group:
 - Security group name: Test_Security_Group_1
 - Description: Security group for load balancer
 - VPC: Default VPC
 - Inbound rules (The chosen inbound rule is to send and receive requested Web pages from a HTTP server):

Type	Protocol	Port range	Source
Custom TCP	TCP	80	Anywhere – IPv4

Table 2.1 Inbound rules for Test_Security_Group_1

- Outbound rules: All traffic
- Create a new target group for the load balancer. Use these configurations (everything else will be default):
 - Target type: Instances
 - Target group name: TestTargetGroup
 - Tags: Key – name, value – target-group
- Tags: Key – name, value – load-balancer

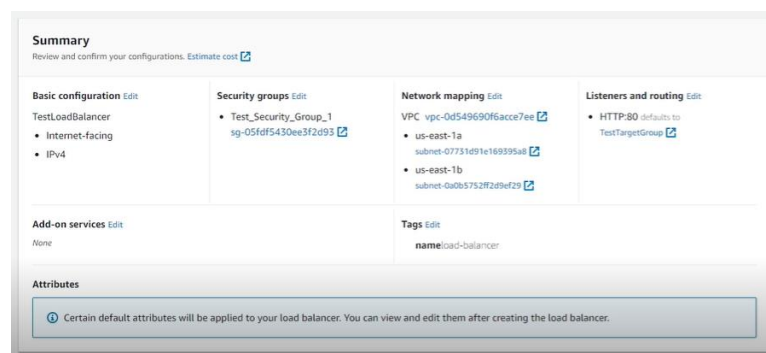


Figure 2.7. Project configuration screenshot 5

7. Load balancer has been successfully created.
8. Choose **Security Groups** from the navigation pane and click **Create New Security Group**.

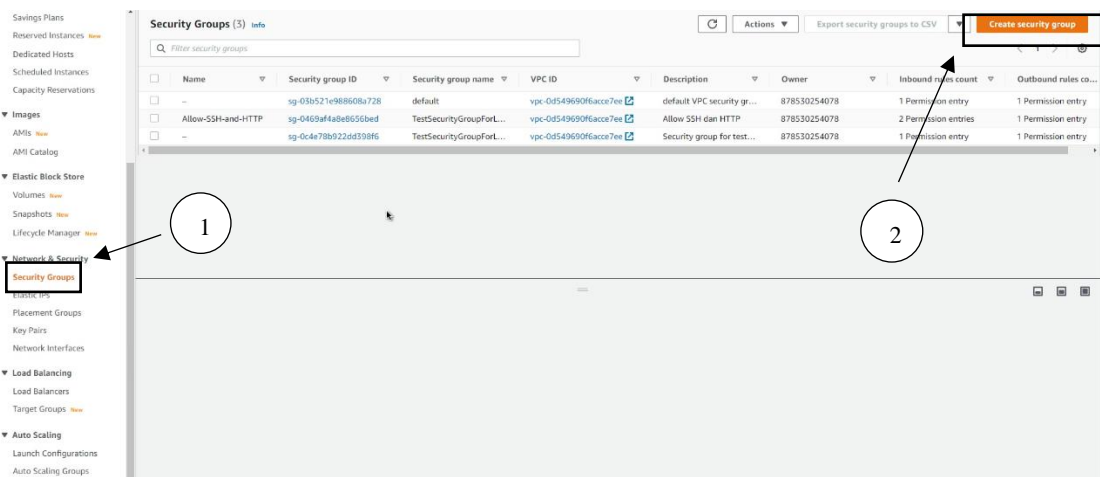


Figure 2.8. Project configuration screenshot 6

9. The configurations for the security group:
 - Security group name: Test_Security_Group_2

- Description: Allow SSH and HTTP
- VPC: Default VPC
- Inbound rules: (The chosen inbound rule is to allow HTTP and SSH):

Type	Protocol	Port range	Source
HTTP	TCP	80	Anywhere – IPv4
SSH	TCP	22	Anywhere – IPv4

Table 2.2 Inbound rules for Test_Security_Group_2

- Outbound rules: All traffic
- Tags: Key - name, Value – allow-ssh-http

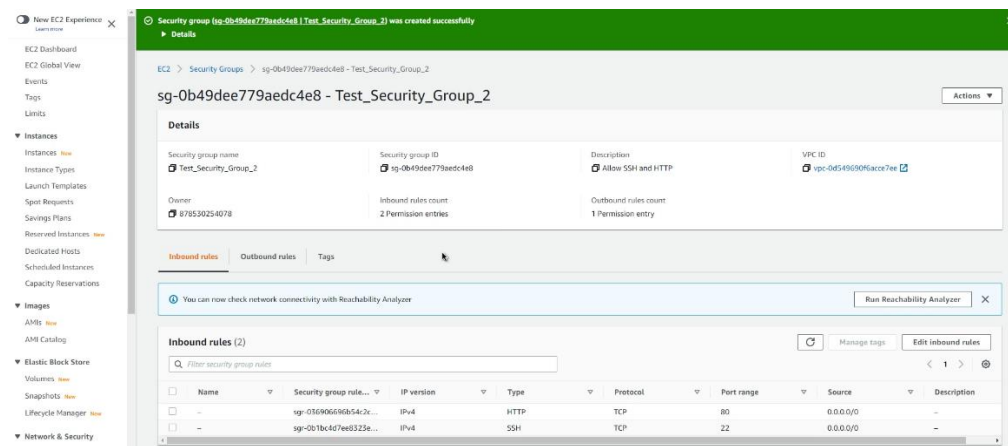


Figure 2.9. Project configuration screenshot 7

10. Choose **Auto Scaling Groups** from the navigation pane and click **Create Auto Scaling group**.

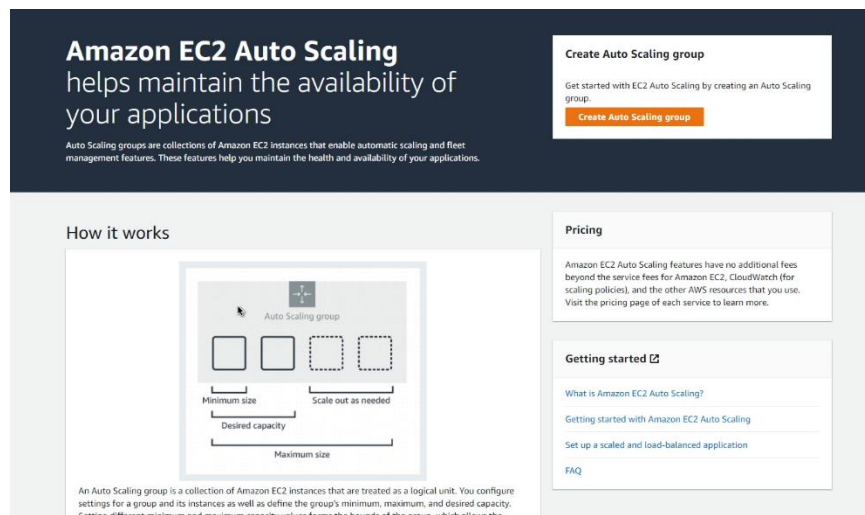


Figure 2.10. Project configuration screenshot 8

Auto Scaling group name: Test_Auto_Scaling_Group.

Create new launch template using these settings:

- Launch template name: TestLaunchTemplate
- Template version description: Launch template for app server
- Amazon Machine Image (AMI): Amazon Linux 2 AMI (HVM) - SSD Volume Type, Latest Kernel, Architecture, 64-bit (x86)
- Instance Type: T2.Micro, free tier eligible
- Key pair, create a new one with these settings:
 - Key pair name: key-pair-load-balancer
 - Key pair type: RSA
 - Private key file format: perm
- Network settings:
 - Security groups: Test_Security_Group_2 created from Step 9
- Storage (volume): Default
- Advanced details, add this command under user data and click Create launch template. This command or code will install Apache Web and start it during the start-up of our EC2 instances later.

```
#!/bin/bash
yum install httpd -y
systemctl start httpd
systemctl enable httpd
```

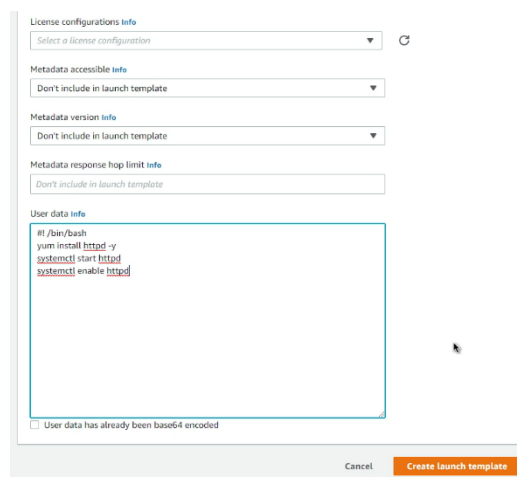


Figure 2.11. Project configuration screenshot 9

Use the created launch template for the group and click Next.

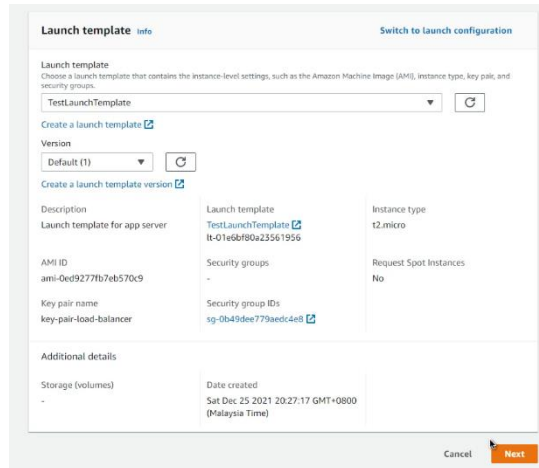


Figure 2.12. Project configuration screenshot 10

- Network:
 - VPC: Default VPC
 - Availability Zones and subnets: us-east-1a, us-east-1b
- Configure advanced options:
 - Load balancing: Attach to an existing load balancer
 - Target group: TestTargetGroup created from Step 6
 - Health checks: Default option
- Configure group size and scaling policies:
 - Group size:

Desired capacity	2
Minimum capacity	2
Maximum capacity	2

Table 2.3 Number of Instance desired for Auto Scaling

- Scaling policies: none
- Tags: Key – name, Value – auto-scaling-ec2

Click **Create Auto Scaling group**. The group will appear in a list like this:

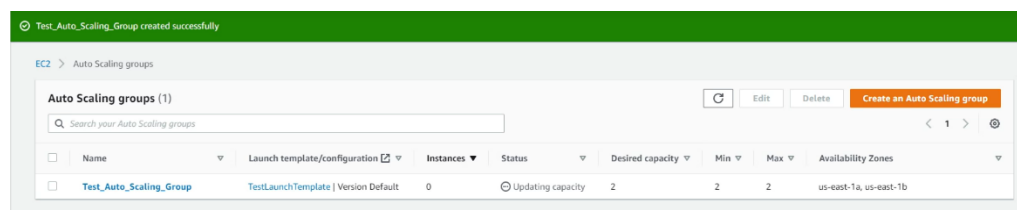


Figure 2.13. Project configuration screenshot 11

11. Testing can now be done. First, test whether the load balancer is working properly by choosing Load Balancers from the navigation pane.
12. Copy the DNS name link of the load balancer created from Step 5 and 6, TestLoadBalancer.

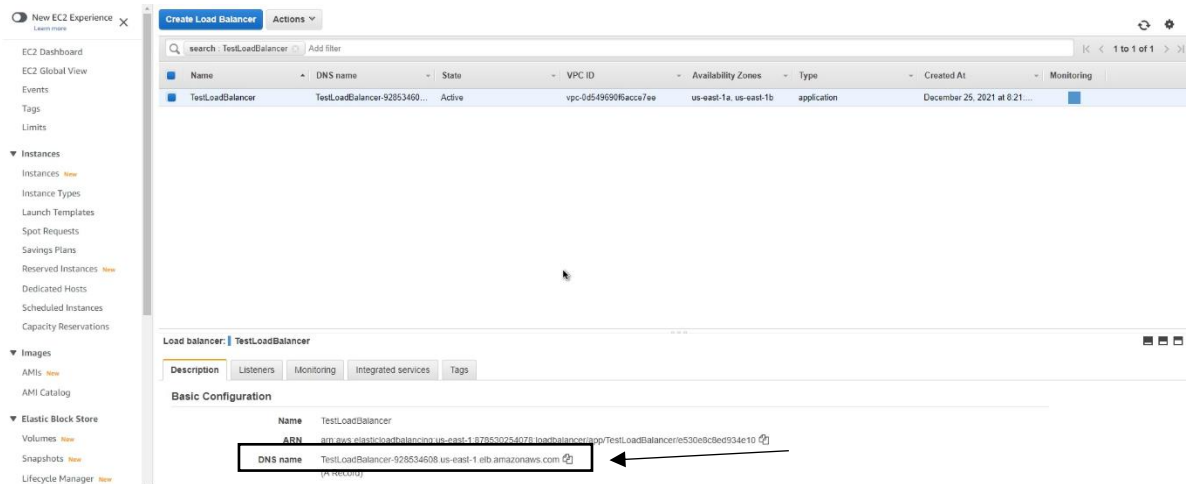


Figure 2.14. Project configuration screenshot 12

13. Open a new tab and paste the link on the search bar. An Apache test page will appear if the load balancer is working properly.

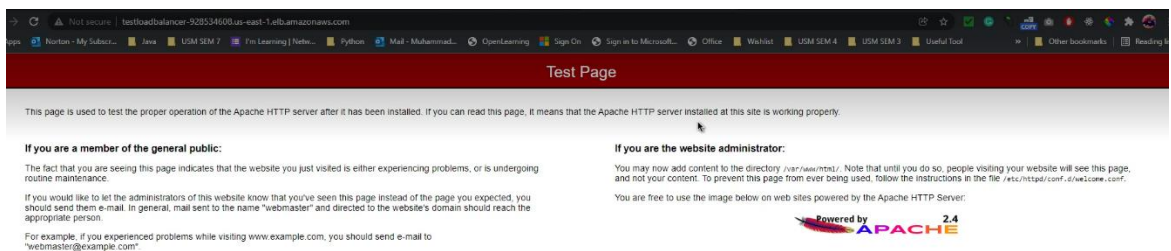


Figure 2.15. Project configuration screenshot 13

14. Select **Instances** from the navigation pane. The list should display two running EC2 instances from two different availability zones, us-east-1a and us-east-1b. Ensure that the Status check are both 2/2 checks passed.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4	Elastic IP
-	i-0f19e25ec4941db96	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-3-85-22-43.comput...	3.83.22.43	-
-	i-0e9d8597c2f19ae2	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-3-80-99-87.comput...	3.80.99.87	-

Figure 2.16. Project configuration screenshot 14

15. Connect to the first running instance from Availability Zone us-east-1a by clicking on the instance and select **Connect**.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input checked="" type="checkbox"/>	i-0f19e25ec4941db96	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-3-83-22-43.compute-1.amazonaws.com	3.83.22.43	-
<input type="checkbox"/>	i-0e9d8597c2f19ae2	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-3-80-99-87.compute-1.amazonaws.com	3.80.99.87	-

Figure 2.17. Project configuration screenshot 15

16. To publicly view the web server, the EC2 needs to be connected using the SSH key pair. Copy the command below:

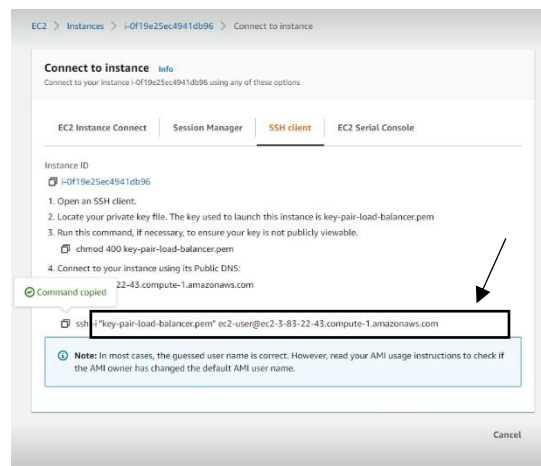


Figure 2.18. Project configuration screenshot 16

17. Open Windows command prompt and navigate to the location the key pair is located inside local folder. Paste the command copied from the previous step and press Enter.
18. After that, run this command:

This command will navigate to `/var/www/html/` directory and create a new file name `index.html`. The `index.html` file contains a line of code that displays “This is Apache Web Server 1” in the heading text.

```
cd /var/www/html/
sudo nano index.html
<h>This is Apache Web Server 1</h>
```

19. Repeat Step 15-18 with the second running EC2 instance from Availability Zone `us-east-1b` but replace the command with:

This command will navigate to `/var/www/html/` directory and create a new file name `index.html`. The `index.html` file contains a line of code that displays “This is Apache Web Server 2” in the heading text.

```
cd /var/www/html/
sudo nano index.html
<h>This is Apache Web Server 2</h>
```

20. Refresh the Apache test page from Step 13 and the page will look like an example of Figure 2.19 below. This means that the first EC2 instance of Availability Zone us-east-1a is running properly.

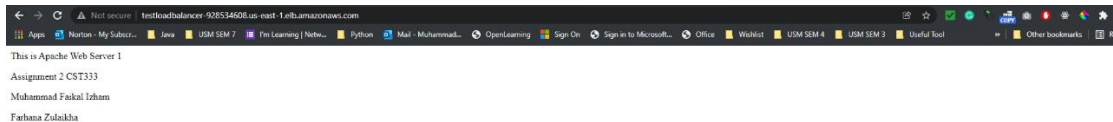


Figure 2.19. Project configuration screenshot 17

21. Refresh the page, again and again, to see the page alternatively switching to the second EC2 instance of Availability Zone us-east-1b.

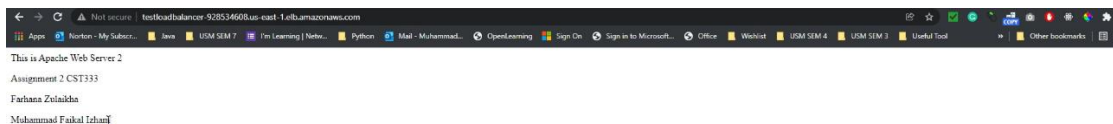


Figure 2.20. Project configuration screenshot 18

22. Test whether the Elastic Load Balancer will create a new instance when an unhealthy instance is detected or not. First, stop the first running EC2 instance of Availability Zone us-east-1a.

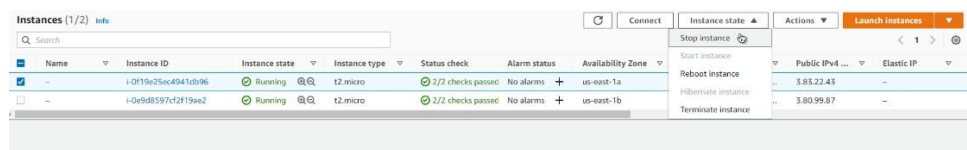


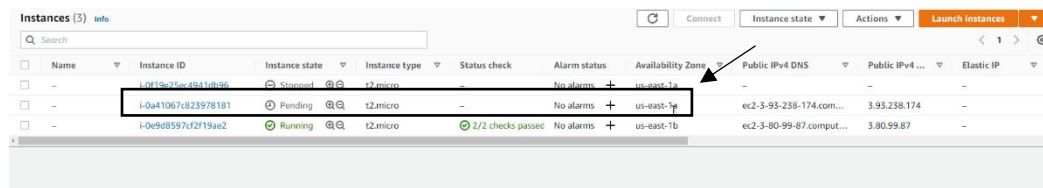
Figure 2.21. Project configuration screenshot 19

23. When the Instance state shows 'Stopped', the first instance can no longer be accessed publicly.



Figure 2.22. Project configuration screenshot 20

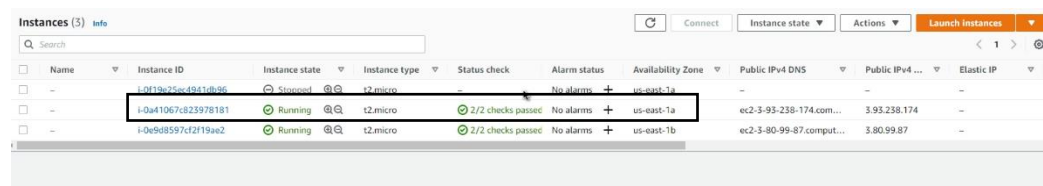
24. A new instance is created automatically to replace the terminated instance. When refreshing the page, a new instance with the same Availability Zone as the terminated instance and 'Pending' state will appear.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
-	i-0f19c75e4951d836	Stopped	t2.micro	-	No alarms	us-east-1a	-	-	-
-	i-0a41067c823978181	Pending	t2.micro	-	No alarms	us-east-1a	ec2-3-93-238-174.com...	3.93.238.174	-
-	i-0e9d8597cf2f19ae2	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-3-80-99-87.comput...	3.80.99.87	-

Figure 2.23. Project configuration screenshot 21

25. Refresh the instances until the new instance shows Instance state as 'Running' and Status check as '2/2 checks passed'.



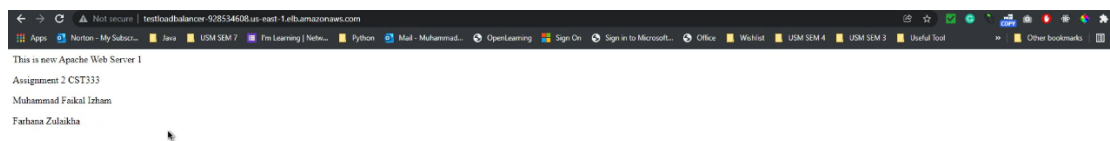
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
-	i-0f19c75e4951d836	Stopped	t2.micro	-	No alarms	us-east-1a	-	-	-
-	i-0a41067c823978181	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-3-93-238-174.com...	3.93.238.174	-
-	i-0e9d8597cf2f19ae2	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-3-80-99-87.comput...	3.80.99.87	-

Figure 2.24. Project configuration screenshot 22

26. Connect to the new EC2 instance using the step from 15-18 and replace the command with the new command:

```
cd /var/www/html/
sudo nano index.html
<h>This is new Apache Web Server 1</h>
```

27. Refreshing the Apache test page will also now show the new EC2 instance.



This is new Apache Web Server 1

Assignment 2 CST333

Muhammad Faikal Izham

Farhana Zulakha

Figure 2.25. Project configuration screenshot 23

We have also recorded the video on how we execute the above steps to achieve the final product which is the Highly Available Auto Scaling Web Server. The link for full video on how we executed for the above steps can be accessed through here:

Full 30 minutes Video (video starting from how we do the setting and configure AWS services used)

https://studentusm-my.sharepoint.com/:v:/g/personal/faikalizham_student_usm_my/ERJsl8OCnIBCImEiCBozoYBu-IGYV_2zt0a_PuzbTEmlQ?e=UQYeVx

5 Minutes video (video only shows the final output)

https://studentusm-my.sharepoint.com/:v:/g/personal/faikalizham_student_usm_my/EQWdng9SnUhIgYZ-1lpfjE8BZ8e_RM3q9tRYCg1FGCgT3A?e=dTzaSs

3 INPUT AND OUTPUT

1. Input – Apache Web Server 1

Comment: This command will navigate to /var/www/html/ directory and create a new file name index.html. The index.html file contains simple lines of code to indicate that this instance is hosting the web server 1.

```
cd /var/www/html/  
sudo nano index.html  
<h>This is Apache Web Server 1</h>  
<h>Assignment 2 CST333</h>  
<h>Muhammad Faikal Izham</h>  
<h>Farhana Zulaikha</h>
```

Output:

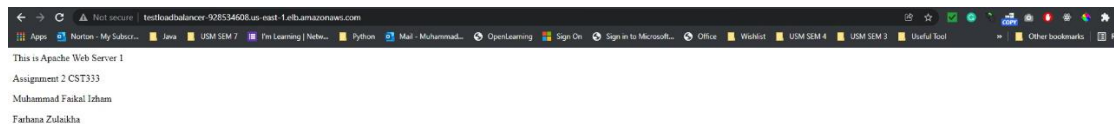


Figure 3.1. Output 1

2. Input – Apache Web Server 2

Comment: This command will navigate to /var/www/html/ directory and create a new file name index.html. The index.html file contains simple lines of code to indicate that this instance is hosting the web server 2.

```
cd /var/www/html/  
sudo nano index.html  
<h>This is Apache Web Server 2</h>  
<h>Assignment 2 CST333</h>  
<h>Farhana Zulaikha</h>  
<h>Muhammad Faikal Izham</h>
```

Output:

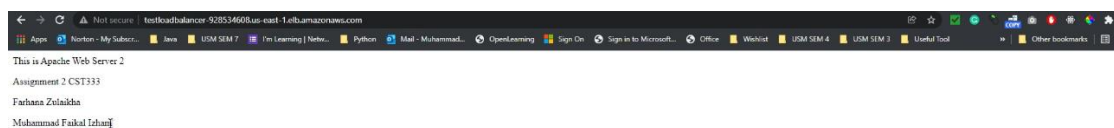


Figure 3.2. Output 2

4 WORK DIVISION

The clear division of our work for this assignment is depicted in Table 4.1 below:

Group member	Work division
Muhammad Faikal Izham bin Abdul Rahim	<ul style="list-style-type: none">• Do research on the AWS tutorial.• Creating and coding the web server.• Write the documentation.
Farhana Zulaikha binti Fadzli	<ul style="list-style-type: none">• Do research on AWS services.• Testing/debugging the web server.• Write the documentation.

Table 4.1 Table of Work Division

5 LESSONS LEARNED

Through this assignment, there are several points that we could take note of regarding the use of AWS to maintain a highly available web server. Firstly, AWS ELB can help protect a website from failure due to traffic rerouting. In case of the website having high traffic, ELB distributes the load evenly to avoid the website suddenly crashing. Health check is done by ELB to ensure that all instances are healthy. Auto Scaling Group is crucial in a way that it replaces the unhealthy instance with a new one, for ELB to redirect the user to.

We can also learn that having a highly available web server is important because of these reasons:

- Recovering from server failure automatically.
- Minimizing website downtime.
- Ensuring user can access the website anytime without interruptions.
- Increasing the performance of the website.

6 CONCLUSION

In conclusion, having a web server that is accessible to users most of the time can help optimize users' experience when using the website. We can achieve this in AWS using the combination EC2, VPC ELB, and Auto Scaling Group to deploy a highly available web server infrastructure.

For our future work, we hope that we can use Relational Database Service (RDS) as well when deploying the web server. For now, the web server created in this assignment is just a simple one that does not require any database setup and management. When we want to create a more complex web server infrastructure, we hope to use RDS for data storing as it is cost-effective, fast, and saves time [7].

7 REFERENCES

- [1] Techopedia, "Web Server," Techopedia Inc., 29 6 2017. [Online]. Available: <https://www.techopedia.com/definition/4928/web-server>. [Accessed 25 1 2022].
- [2] D. Carty, "Elastic Load Balancing (ELB)," SearchAWS, [Online]. Available: <https://searchaws.techtarget.com/definition/elastic-load-balancing>. [Accessed 25 1 2022].
- [3] AWS, "Auto Scaling groups," AWS, [Online]. Available: <https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroup.html>. [Accessed 25 1 2022].
- [4] J. Varia and S. Mathew, "Overview of Amazon Web Services," 1 2014. [Online]. Available: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/aws-overview.pdf#introduction>. [Accessed 10 1 2022].
- [5] S. Dashora, "Setting up Elastic Load Balancer for AWS EC2 Instances," Progressive Coder, 16 7 2019. [Online]. Available: <https://progressivecoder.com/setting-up-elastic-load-balancer-for-aws-ec2-instances/>. [Accessed 12 1 2022].
- [6] S. Maarek, *AWS Auto Scaling Groups Introduction*, Stephane Maarek, 2020.
- [7] "Amazon Relational Database Service (RDS)," AWS, [Online]. Available: <https://aws.amazon.com/rds/>. [Accessed 29 1 2022].