

COS20019 – Cloud Computing Architecture

Assignment 2: Developing a Highly Available Photo Album Website

Student ID: 104082552

Student Name: Nhat Minh Tran

Lab section: Tuesday 06:30PM

I. Introduction

In this assignment two, we used the foundation that we created in the previous assignment-assignment 1b, we have known how to create a basic web server and security groups, different subnets and run phpMyAdmin. This assignment, we are enhancing and extending the infrastructure. We make the web becomes highly available with auto scaling groups, Application Load Balancer, Lambda function. With all this being done, we will have more vision and knowledge of a scalability, reliability, security website that managers and company use.

II. Infrastructure deployment

1. VPC

VPC “MTranVPC” is created, and it is a fundamental networking service provided by AWS. I created the VPC with configuration match the architecture diagram. Here is a clearer look of VPC provided by resource map. Two public subnet and two private subnets are created within two AZs.

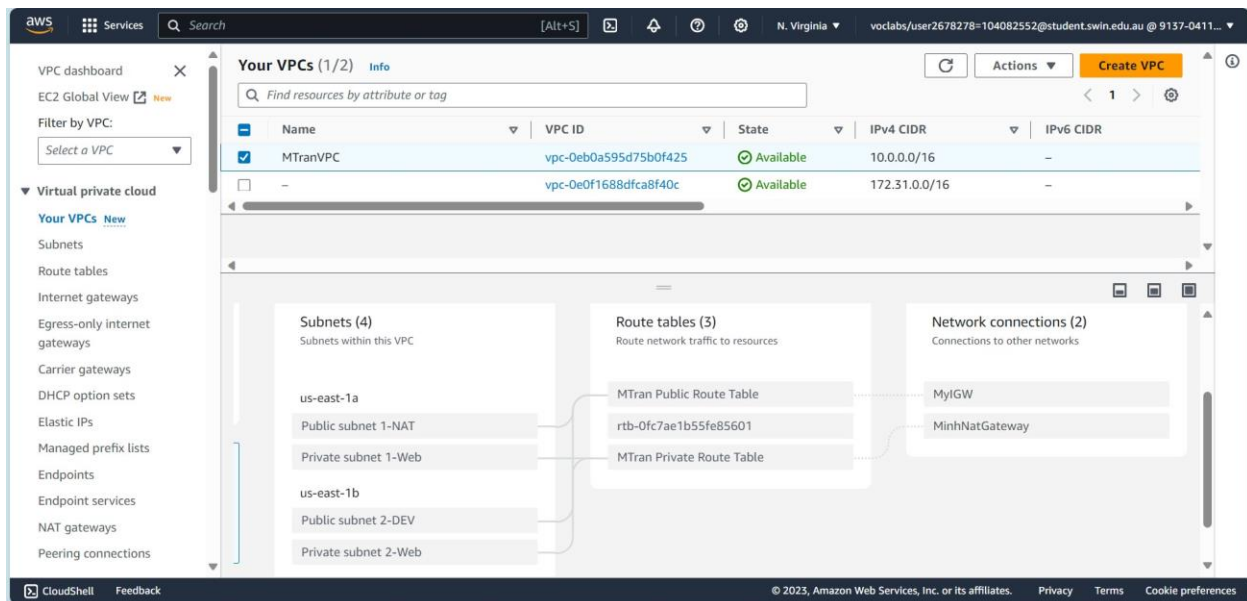


Figure 1: Resource Map

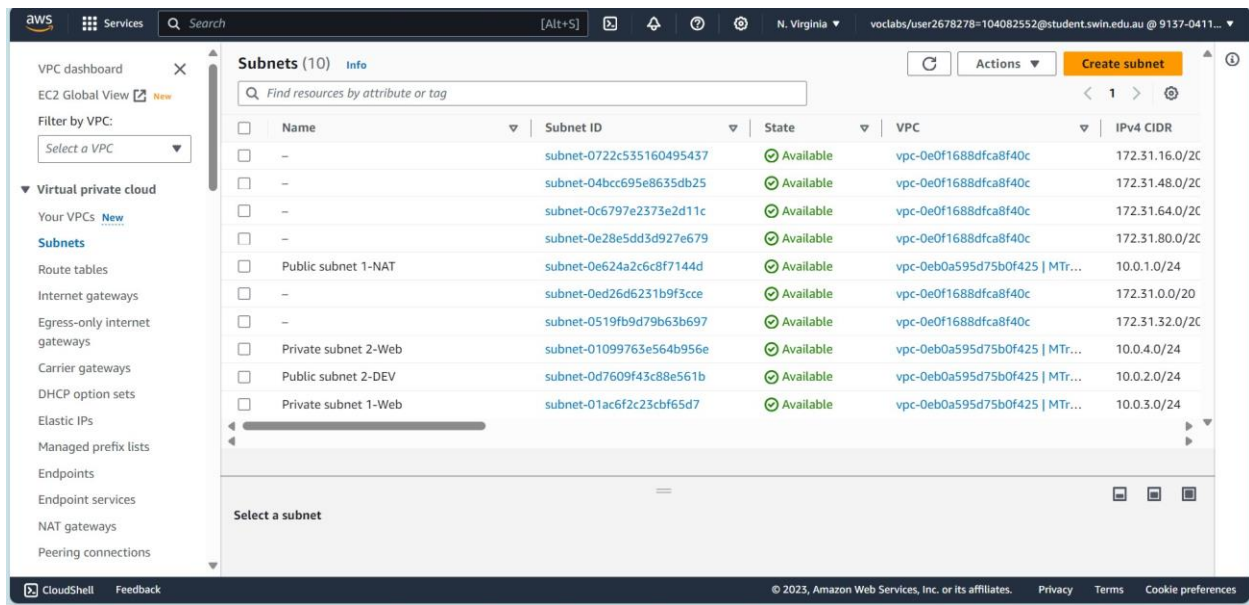


Figure 2: 4 subnets with 2AZ

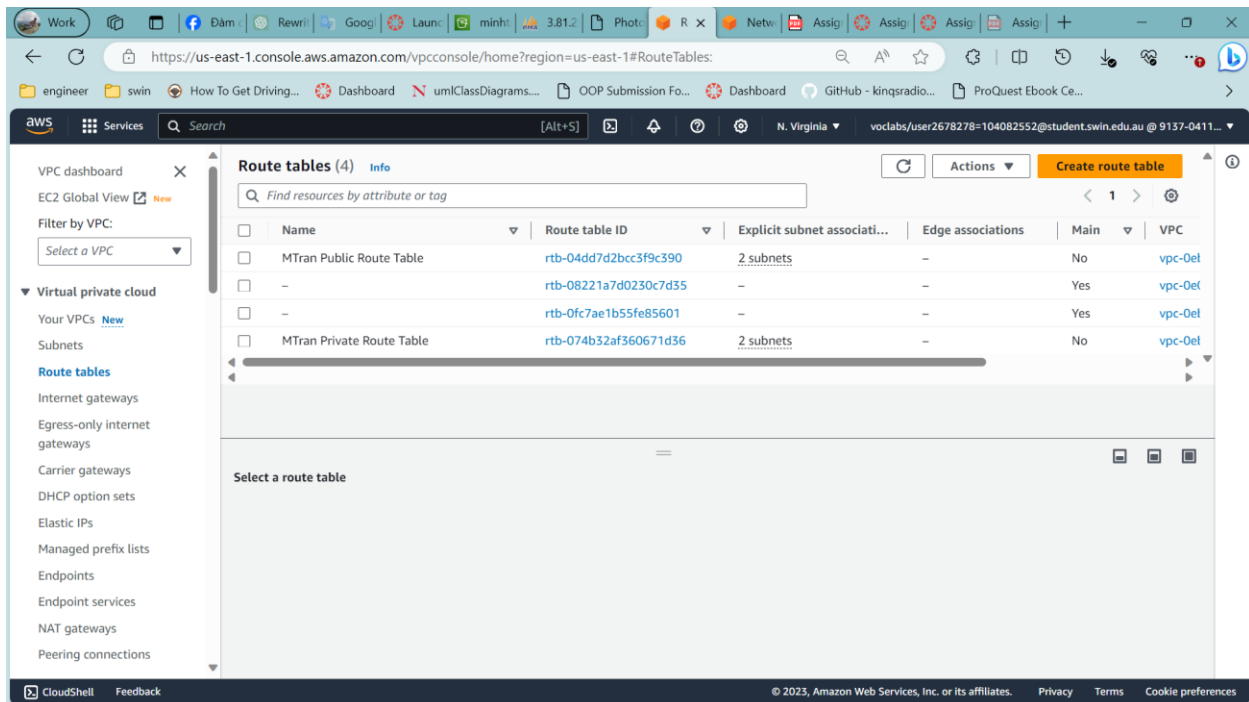


Figure 3: 2 Route Table associated with different subnets.

Moreover, the Public Subnet 2-DEV is routed to Internet Gateway to go to users. In the Public Subnet 1-Web, I created a NAT gateway for scalability and reliability instead of NAT instance. The NAT gateway allows resources within private subnets to access the internet for updating software or uploading things, but keep it hidden from inbound traffic from the internet. The two private subnets are configured to route the traffic through this NAT gateway.

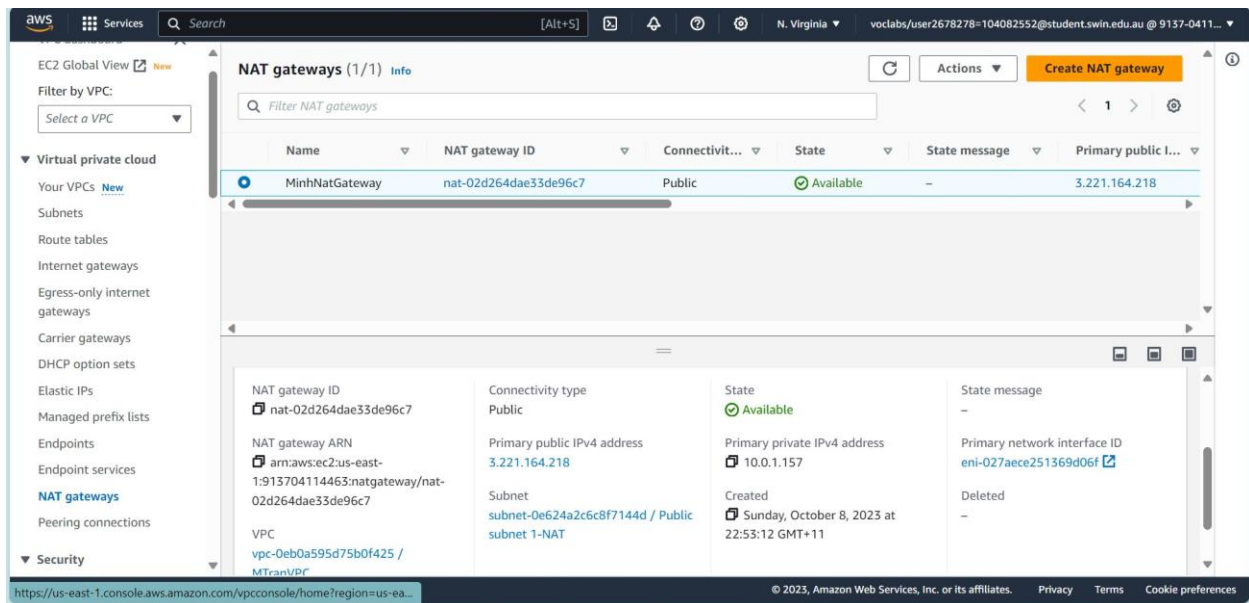


Figure 4: NAT gateway

2. S3 photo storage

S3 bucket is created to store images, which has been created in assignment 1b. However, the policy and permission of this will be different due to the Application Load Balancer (ALB) and the S3 is also staying outside of VPC.

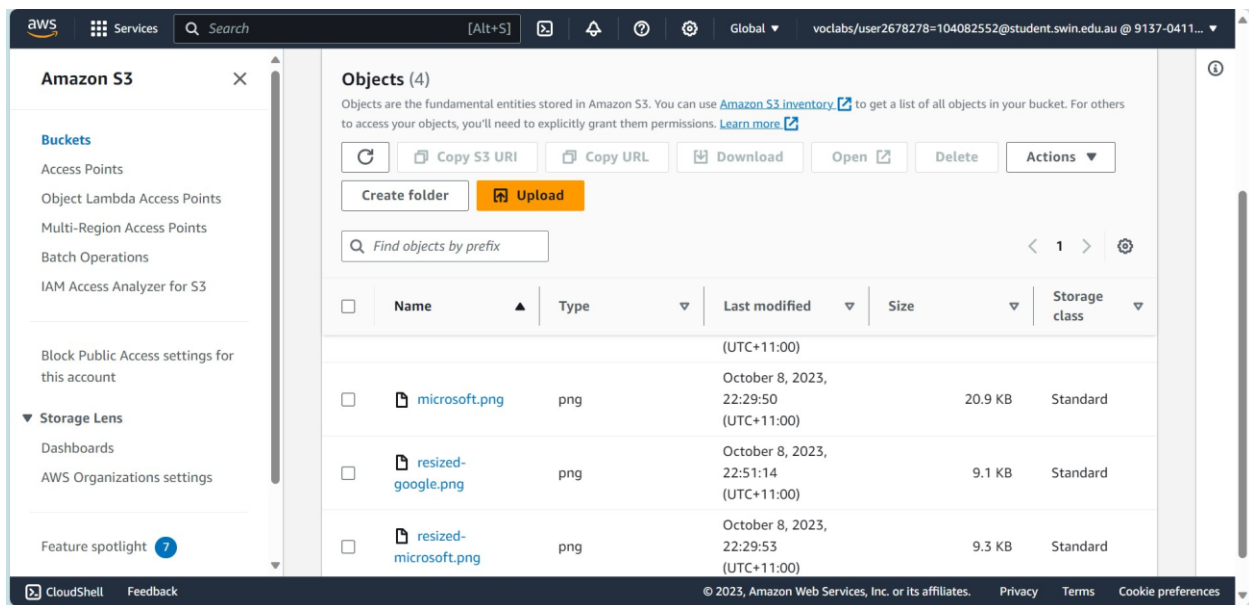


Figure 5: Minhtranbucket

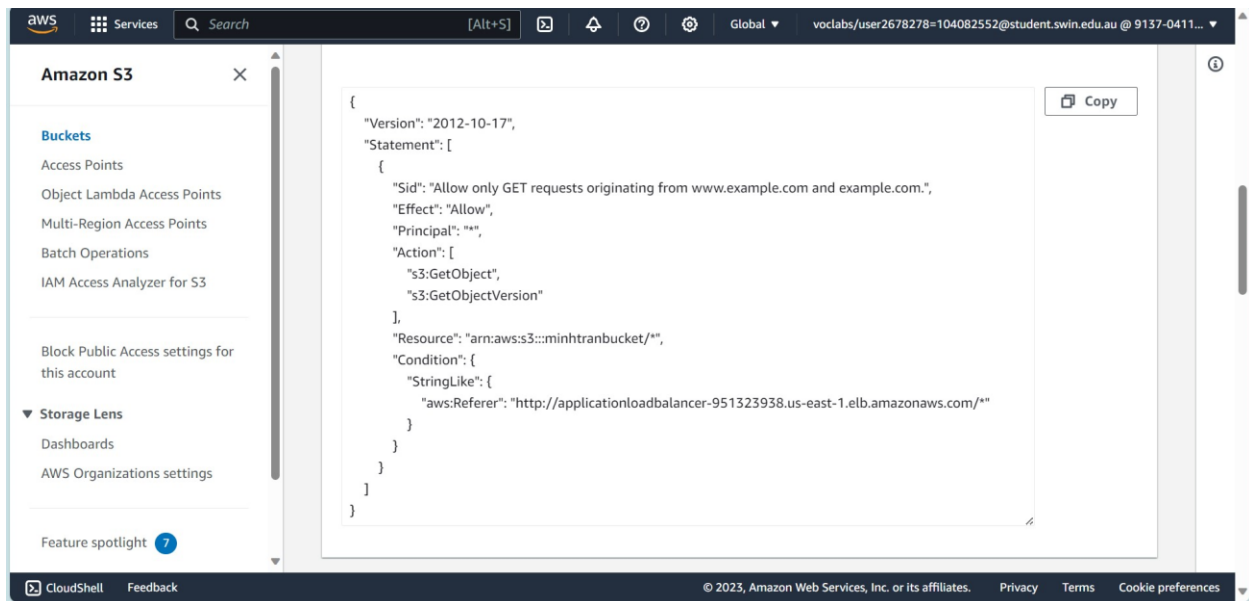


Figure 6: S3 Bucket policy

This IAM policy statement allows the GET requests to objects in my bucket but only if the Referer header in the HTTP matches the value. This policy is a way to restrict people getting to the website, ensuring that only requests from a particular domain are permitted to access the S3 objects. The permission also goes through ALB, enhancing the overall security of the website.



Figure 7: Access denied to S3 objects

3. Security Groups and Network ACL

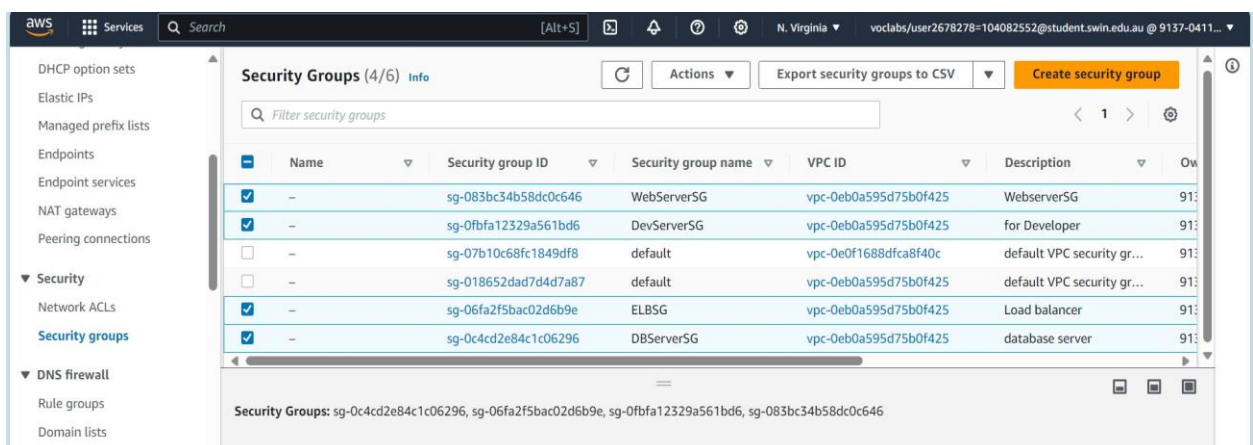
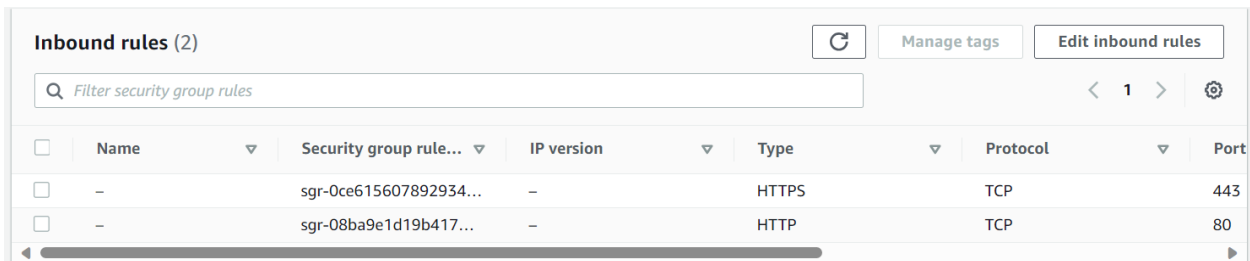


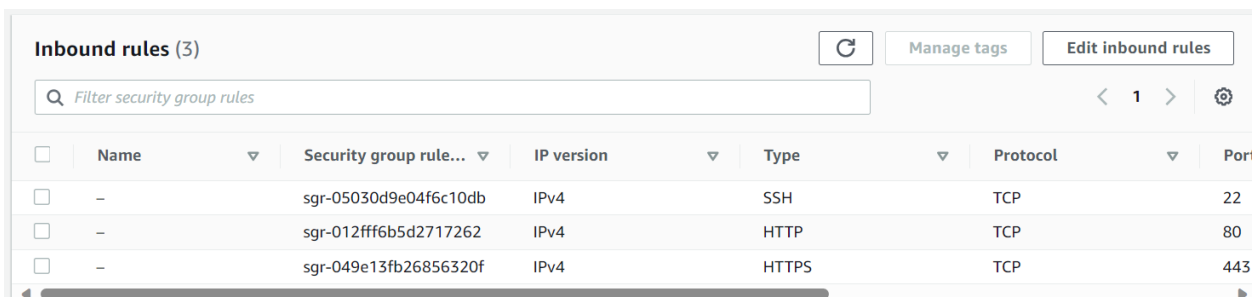
Figure 8: Security groups

To ensure the website secured and working great, I have security groups with different configurations on inbound and outbound rules.



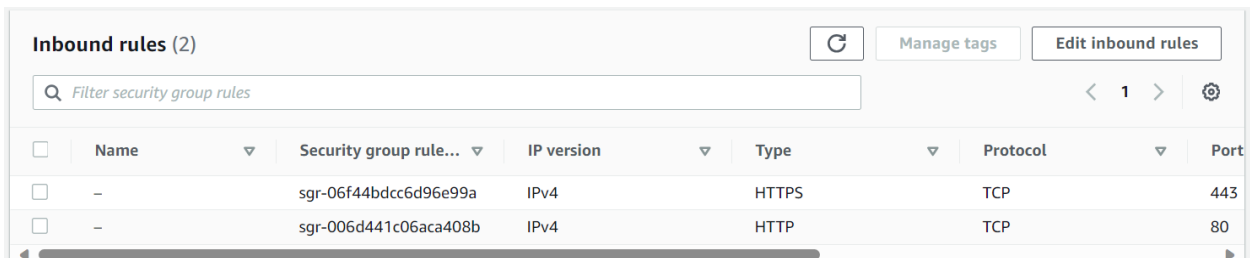
Inbound rules (2)							
Filter security group rules							
	Name	Security group rule...	IP version	Type	Protocol	Port	
<input type="checkbox"/>	-	sgr-0ce615607892934...	-	HTTPS	TCP	443	
<input type="checkbox"/>	-	sgr-08ba9e1d19b417...	-	HTTP	TCP	80	

Figure 9: Inbound WebServerSG



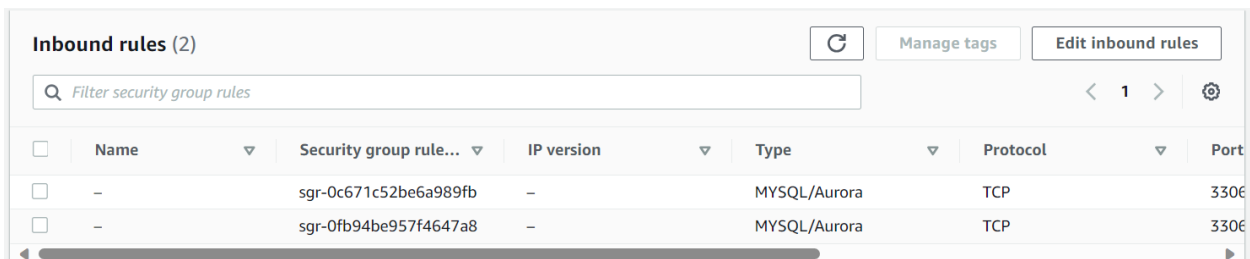
Inbound rules (3)							
Filter security group rules							
	Name	Security group rule...	IP version	Type	Protocol	Port	
<input type="checkbox"/>	-	sgr-05030d9e04f6c10db	IPv4	SSH	TCP	22	
<input type="checkbox"/>	-	sgr-012fff6b5d2717262	IPv4	HTTP	TCP	80	
<input type="checkbox"/>	-	sgr-049e13fb26856320f	IPv4	HTTPS	TCP	443	

Figure 10: Inbound DevServerSG



Inbound rules (2)							
Filter security group rules							
	Name	Security group rule...	IP version	Type	Protocol	Port	
<input type="checkbox"/>	-	sgr-06f44bdcc6d96e99a	IPv4	HTTPS	TCP	443	
<input type="checkbox"/>	-	sgr-006d441c06aca408b	IPv4	HTTP	TCP	80	

Figure 11: Inbound ELBSG

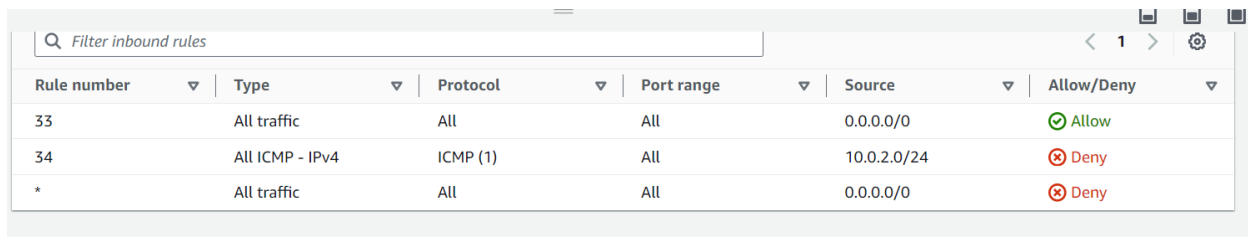


Inbound rules (2)							
Filter security group rules							
	Name	Security group rule...	IP version	Type	Protocol	Port	
<input type="checkbox"/>	-	sgr-0c671c52be6a989fb	-	MySQL/Aurora	TCP	3306	
<input type="checkbox"/>	-	sgr-0fb94be957f4647a8	-	MySQL/Aurora	TCP	3306	

Figure 12: Inbound DBServerSG

The outbound of these security groups remained unchanged. These rules maintain a controlled and secure environment, only allow specific interactions and restricted other unnecessary sources.

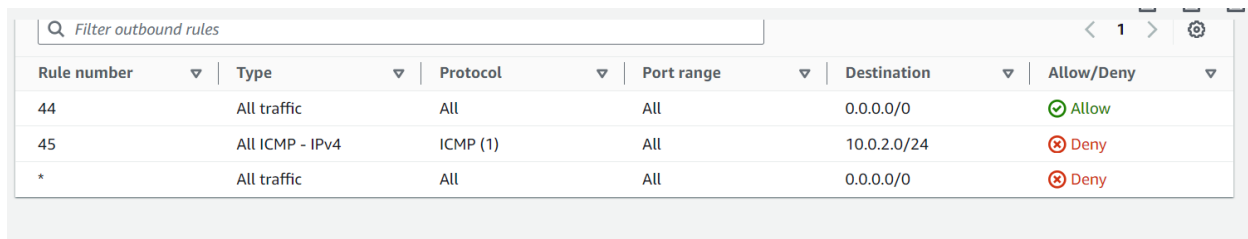
To make the website better and more security, advance configuration such as Network ACL are asked to be done and deployed. This NACL limits ICMP traffic to the corresponding subnets, blocking all the ICMP traffic to/from Dev server.



The screenshot shows the AWS Network ACL inbound rules configuration. A search bar at the top contains the text "Filter inbound rules". The table lists three rules:

Rule number	Type	Protocol	Port range	Source	Allow/Deny
33	All traffic	All	All	0.0.0.0/0	Allow
34	All ICMP - IPv4	ICMP (1)	All	10.0.2.0/24	Deny
*	All traffic	All	All	0.0.0.0/0	Deny

Figure 13: Inbound rules



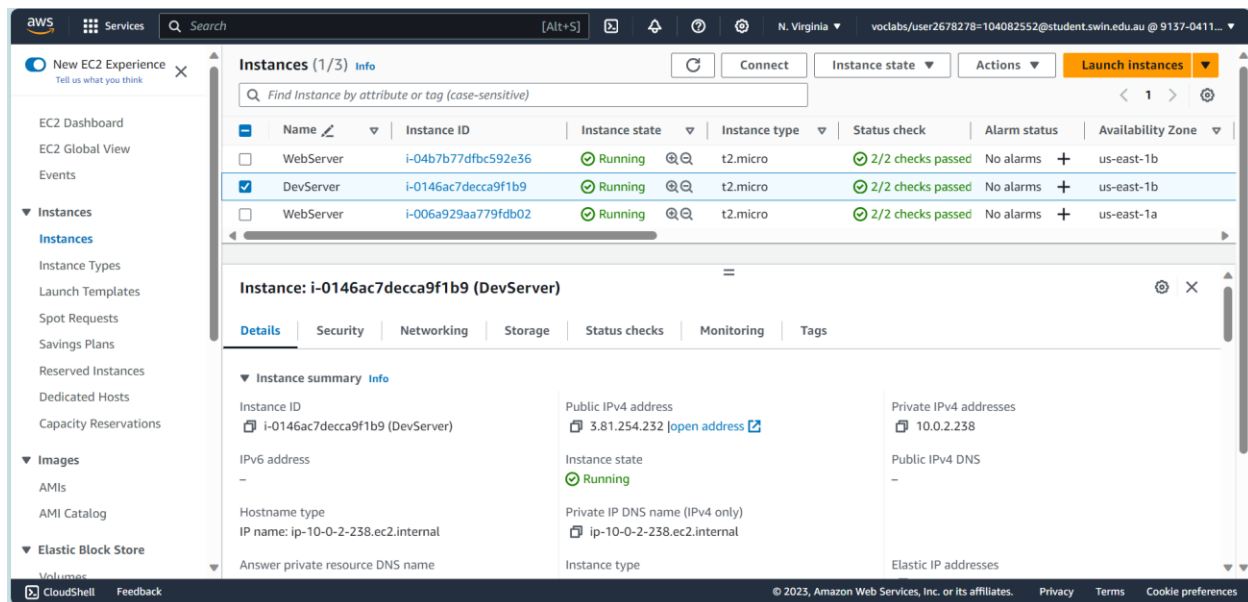
The screenshot shows the AWS Network ACL outbound rules configuration. A search bar at the top contains the text "Filter outbound rules". The table lists three rules:

Rule number	Type	Protocol	Port range	Destination	Allow/Deny
44	All traffic	All	All	0.0.0.0/0	Allow
45	All ICMP - IPv4	ICMP (1)	All	10.0.2.0/24	Deny
*	All traffic	All	All	0.0.0.0/0	Deny

Figure 14: Outbound rules

4. EC2 Instances

The EC2 instances provides a virtual server in cloud. With EC2 instance I can create web application, database, and more. For this assignment, I created one instance, the WebServer instance.



The screenshot shows the AWS Management Console. The left sidebar contains navigation links for EC2 Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, and Volumes. The main content area shows the "Instances (1/3)" page with a table of instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
WebServer	i-04b7b7dfbc592e36	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b
DevServer	i-0146ac7decca9f1b9	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b
WebServer	i-006a929aa779fdb02	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a

Below the table, the details for the "Instance: i-0146ac7decca9f1b9 (DevServer)" are shown. The "Details" tab is selected, displaying the following information:

- Instance ID:** i-0146ac7decca9f1b9 (DevServer)
- Public IPv4 address:** 3.81.254.232 [open address]
- Private IPv4 addresses:** 10.0.2.238
- Instance state:** Running
- Private IP DNS name (IPv4 only):** ip-10-0-2-238.ec2.internal
- Answer private resource DNS name:** ip-10-0-2-238.ec2.internal
- Instance type:** t2.micro
- Elastic IP addresses:** (None listed)

Figure 15: DevServer Instance

5. CreateThumbnail Lambda function

Following the task, I uploaded the code to Lambda to resize images, download and upload images to S3. Also, a test is run to check correction.

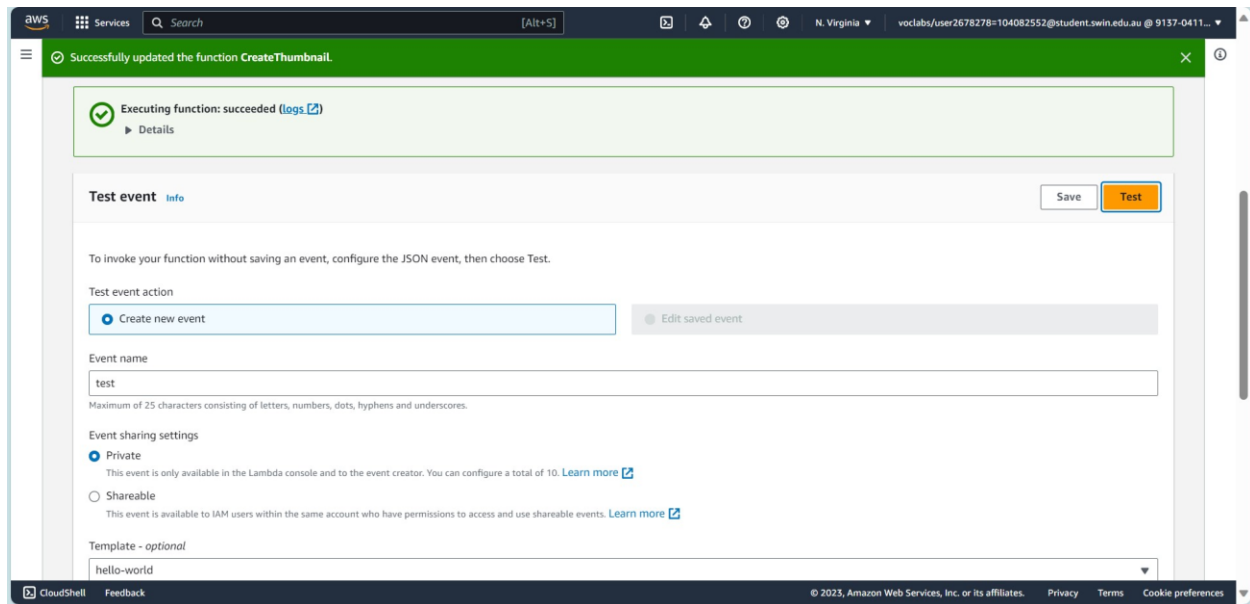


Figure 16: Function testing

6. RDS

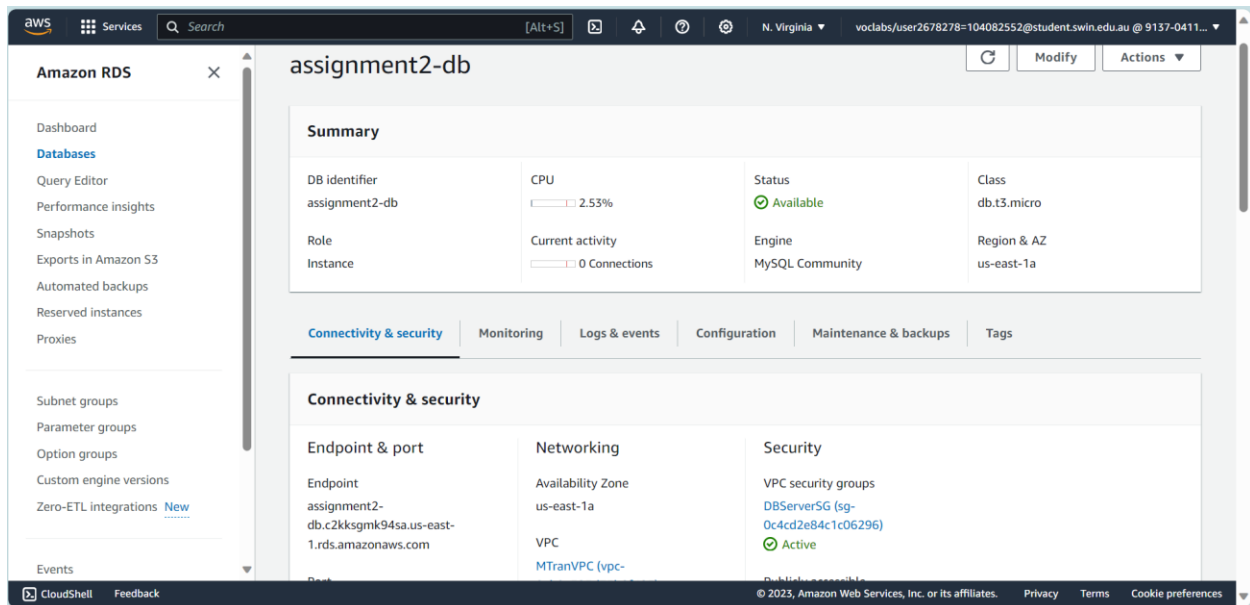


Figure 17: Assignment2-db

The Relational Database Service allows us to instantly launch a database. This RDS is created with MySQL server. It automatically replicates the database to a standby instance in different availability zone for failover protection.

7. Load balancing

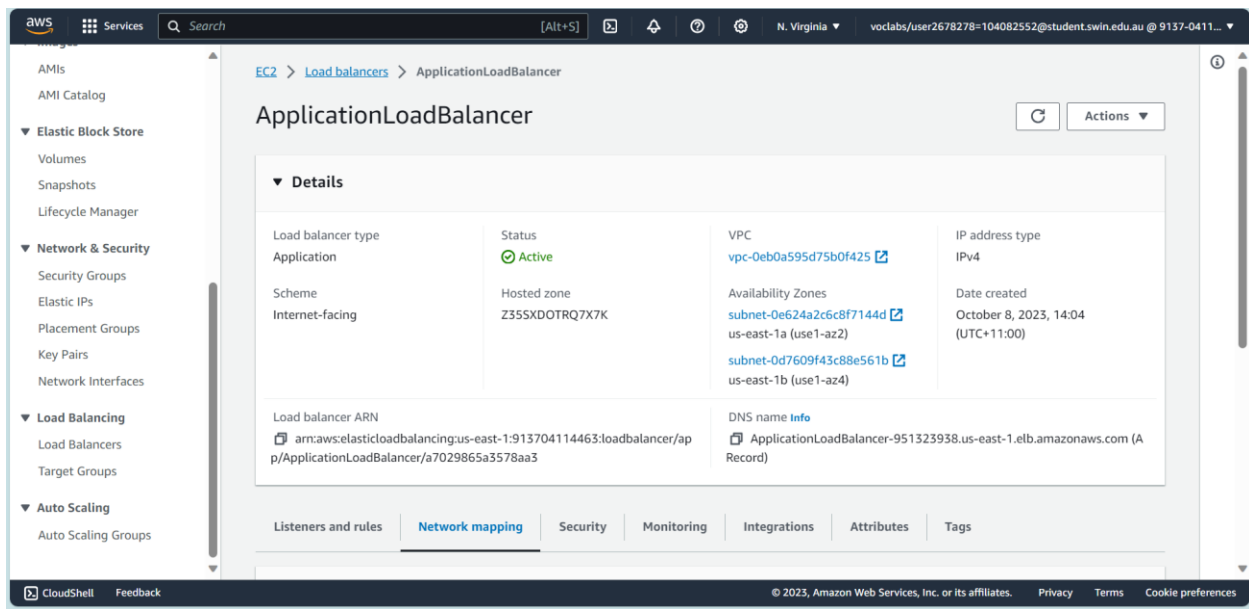


Figure 18: ApplicationLoadBalancer

The mission of Load Balancer is to ensure there is no single server or resource overwhelmed with traffic. It improves the fault tolerance, availability of a system. Also, Target Groups are created to run health checks on all instances.

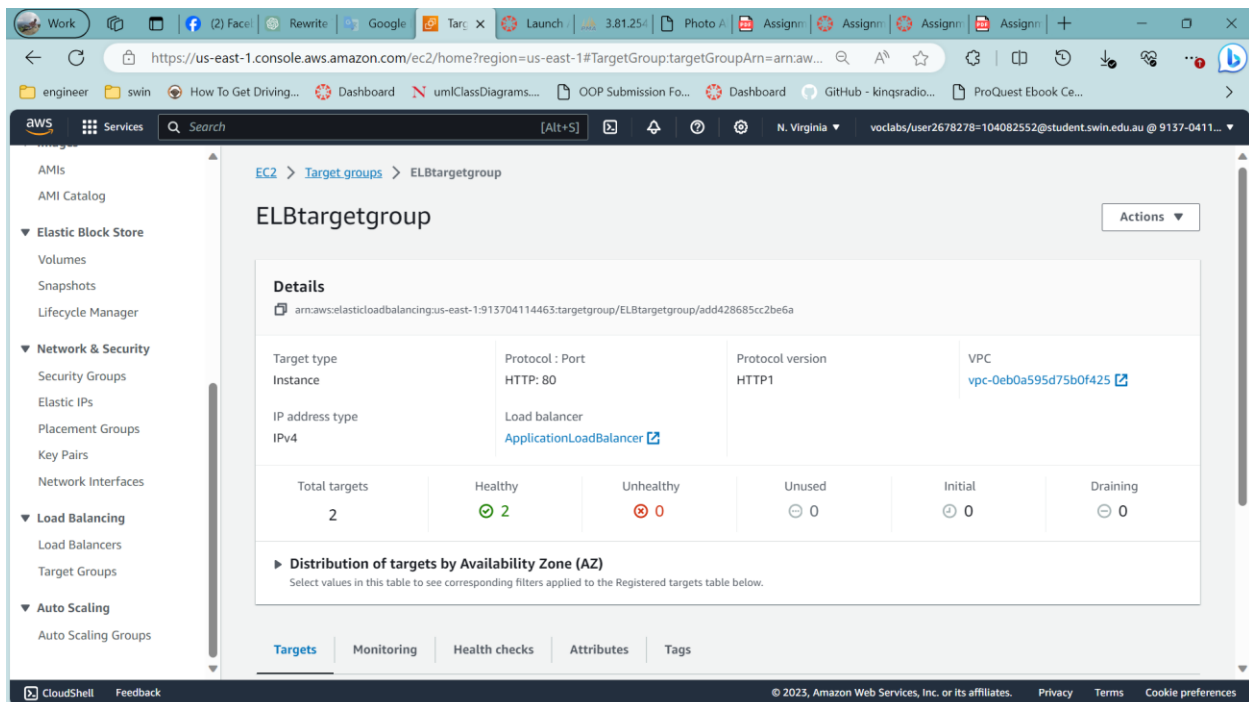


Figure 19: Target group

8. Auto Scaling Group

This feature will scale up the web server, making it highly available, and it has the ability to recover from failures. The ASG, Load Balancer and health checks all go together to create the environment.

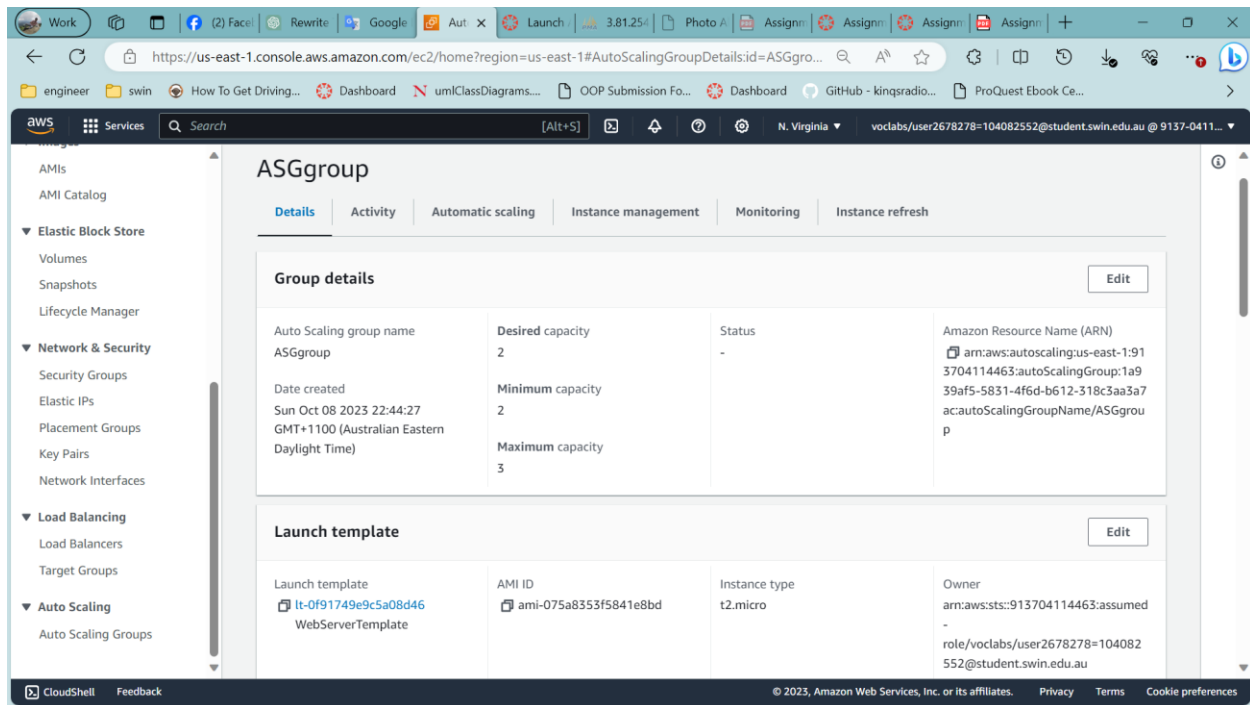


Figure 19: ASG group

9. Photo Album










/var/www/html/					
Name	Size	Changed	Rights	Owner	
 ..		10/8/2023 1:09:36 PM	rw-rw-r--	ec2-user	
 aws		10/8/2023 10:20:20 PM	rw-rw-r--	ec2-user	
 photoalbum		10/8/2023 10:18:16 PM	rw-rw-r--	ec2-user	
 phpmyadmin		10/8/2023 5:38:08 PM	rw-r--r--	ec2-user	
 aws.zip	5,376 KB	10/7/2023 6:09:43 AM	rw-rw-r--	ec2-user	
 phpinfo.php	1 KB	10/8/2023 1:09:44 PM	rw-r--r--	root	
 phpMyAdmin-5.2.1-e...	10,204 KB	2/8/2023 1:59:00 PM	rw-rw-r--	ec2-user	
0 B of 15.2 MB in 0 of 6					
 SETP-3  0:00:04					

Figure 20: WinSCP

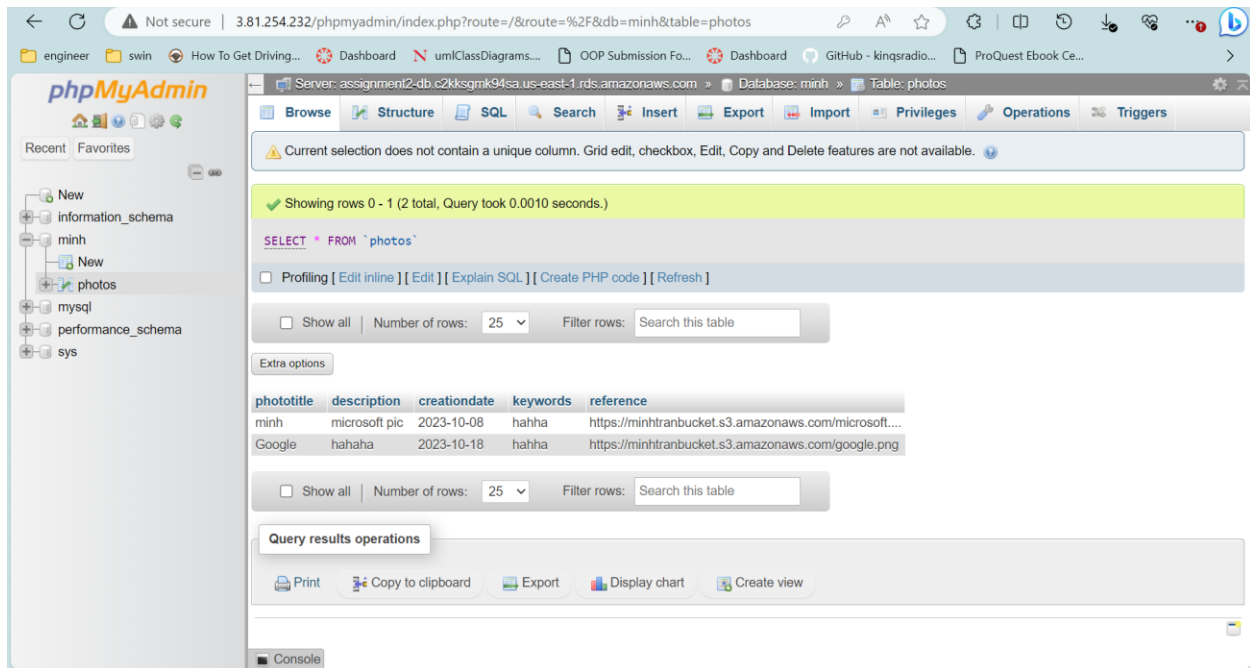


Figure 21: phpMyAdmin

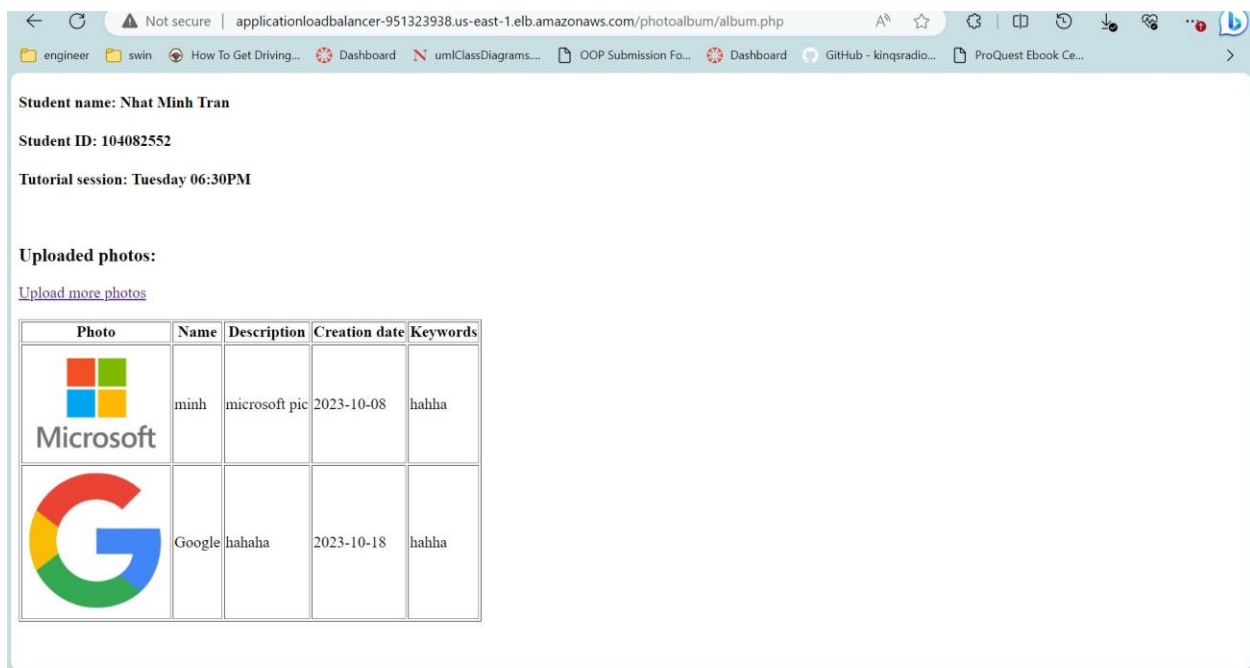


Figure 22: Album.php

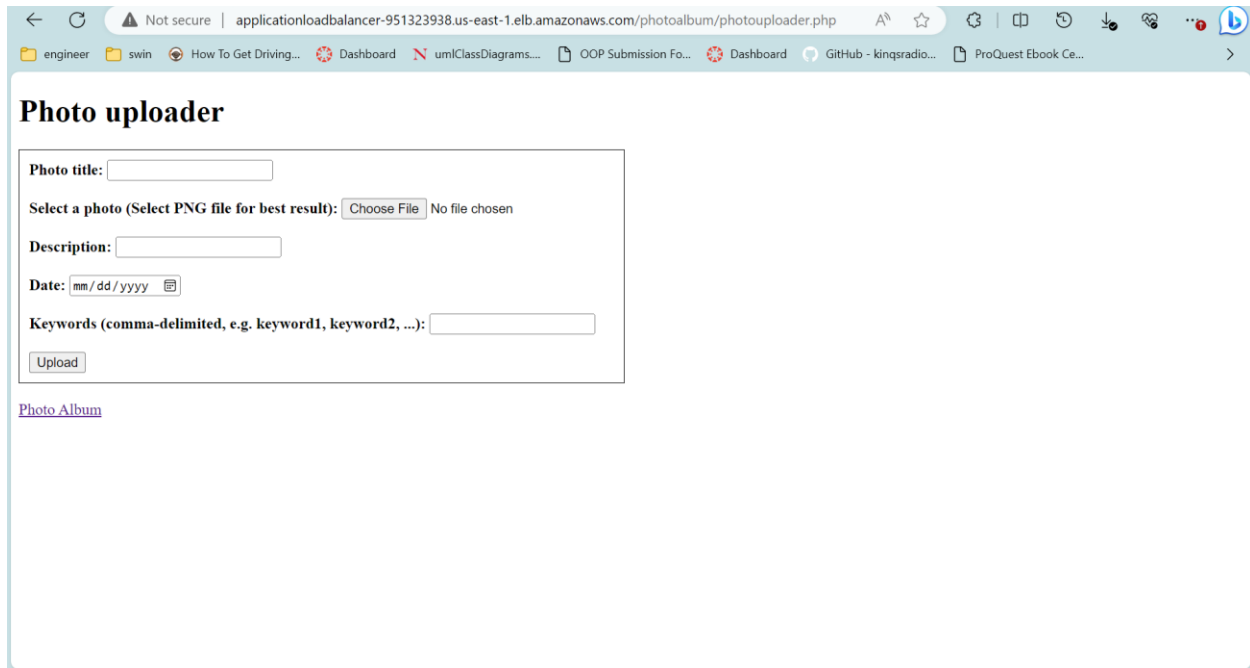


Figure 23: Photo uploader

The link is no more elastic Ip like assignment 1b, because we have created the load balancer, now we are accessing the photoalbum through load balancer link. It indicates that we have successfully create the highly available environment.

Link to ELB album.php: [Photo Album \(applicationloadbalancer-951323938.us-east-1.elb.amazonaws.com\)](http://applicationloadbalancer-951323938.us-east-1.elb.amazonaws.com)

Link to ELB photouploader.php: [Photo Album \(applicationloadbalancer-951323938.us-east-1.elb.amazonaws.com\)](http://applicationloadbalancer-951323938.us-east-1.elb.amazonaws.com)

