Assignment 2 - Developing a highly available Photo Album website

Student name: Hai Nam Ngo

Student ID: 103488515

Tutorial session: Friday 04:30PM

URL of Album Page: http://hngo-lb-609917414.us-east-1.elb.amazonaws.com/photoalbum/album.php

I. INTRODUCTION

In this assignment, an AWS infrastructure was built based on the requirement. This entails incorporating LabRole and LabInstanceProfile IAM roles to facilitate secure interactions among EC2, Lambda, and S3 services. I will also implement restricted S3 access, craft a Lambda function, and create a custom AMI. Using this AMI, a new launch template and Auto Scaling Group will be developed, spanning multiple Availability Zones with scaling policies in place. Additionally, I will establish an Elastic Load Balancer and enforce stringent access and traffic controls via AWS NACLs to ensure secure and streamlined operations.

II. VPC

The VPC is like what has been created in Assignment 1b, the main VPC name is "HNgoVPC" with 2 availability zones and two subnets for each, that means four subnets in total as we can see below, correct configuration.

The public subnets will then connect to a public route table of the same VPC, which then connects to the Internet Gateway, making them become valid "public subnet".

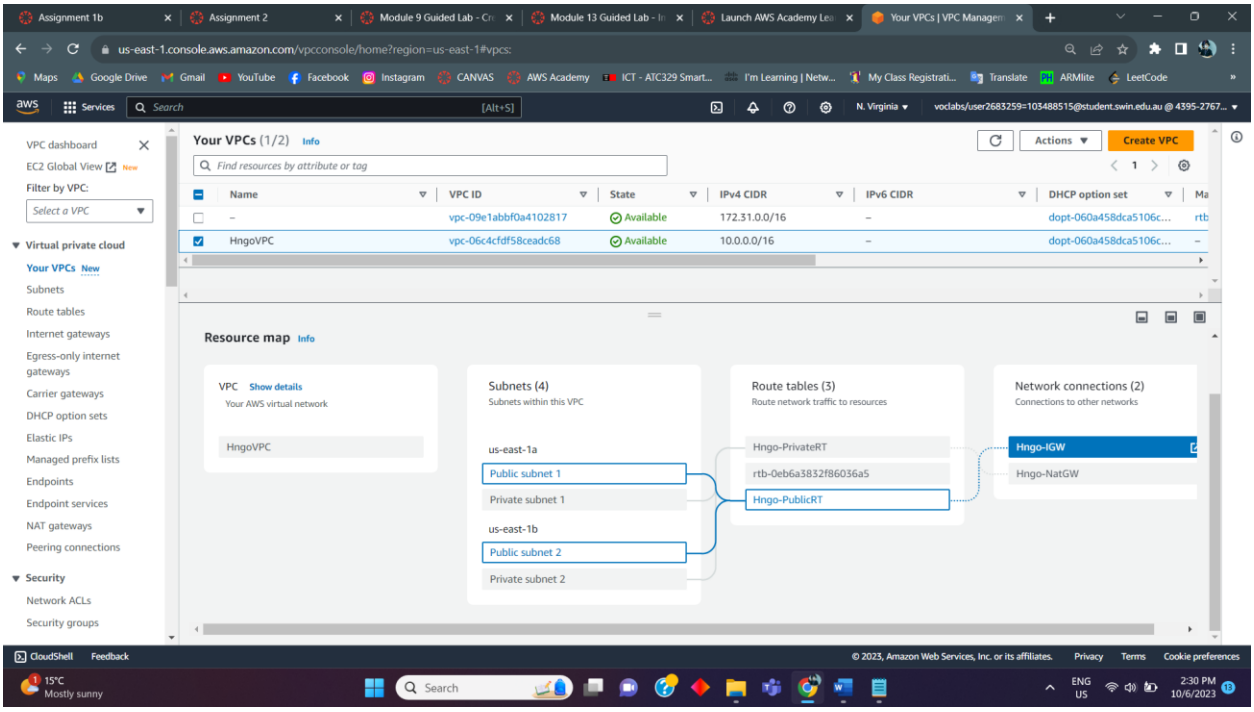


Figure 1 Resource map of HNgoVPC with subnets, route table, internet gateway and NAT gateway

On the other hand, private subnets will connect to a private route table, which is connected to a NAT Gateway. The way I created this Gateway will be illustrated in the next section.

III. NETWORK ADDRESS TRANSLATION (NAT)

The NAT Gateway service in AWS Console was chosen instead of creating a NAT Instance manually since NAT Gateway is easier to create. The configuration is making the Gateway available in the HNgoVPC and connect it with the subnets that I illustrate above.

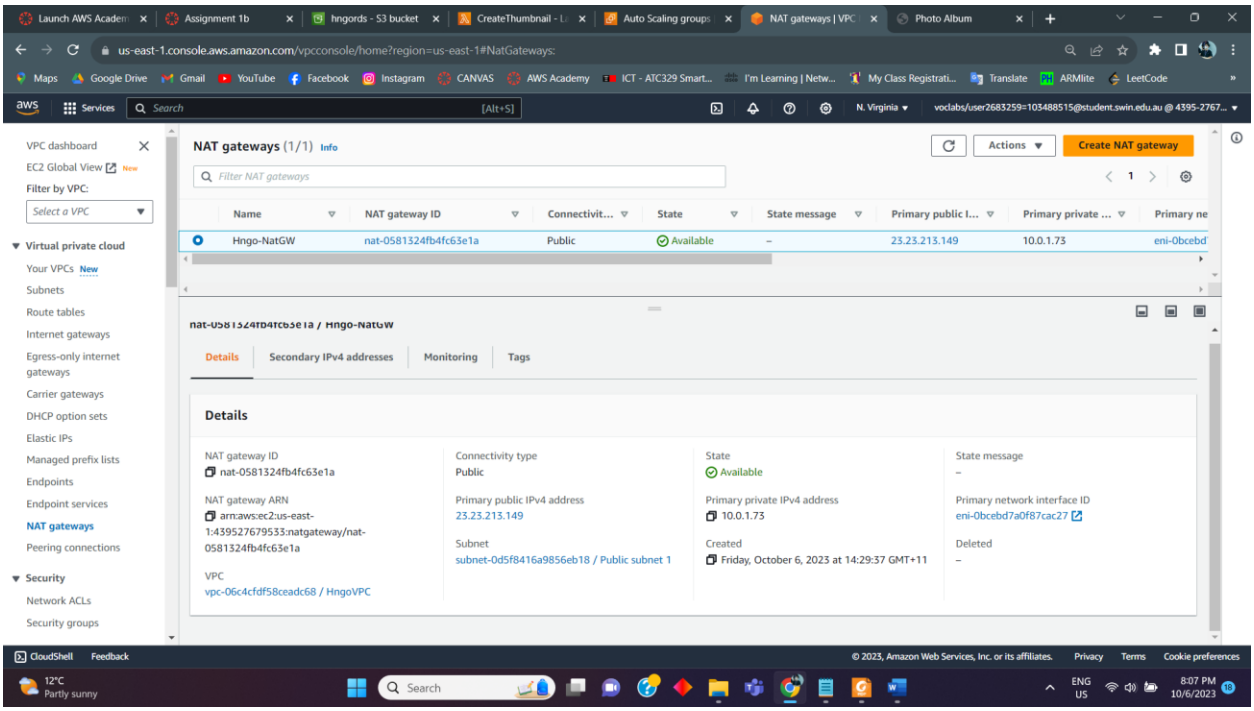


Figure 2 Creating Nat Gateway correctly using AWS Service

IV. EC2 INSTANCE

The EC2 Instance that needs to be created will be DevServer Instance, which is also allocated elastic IP address.

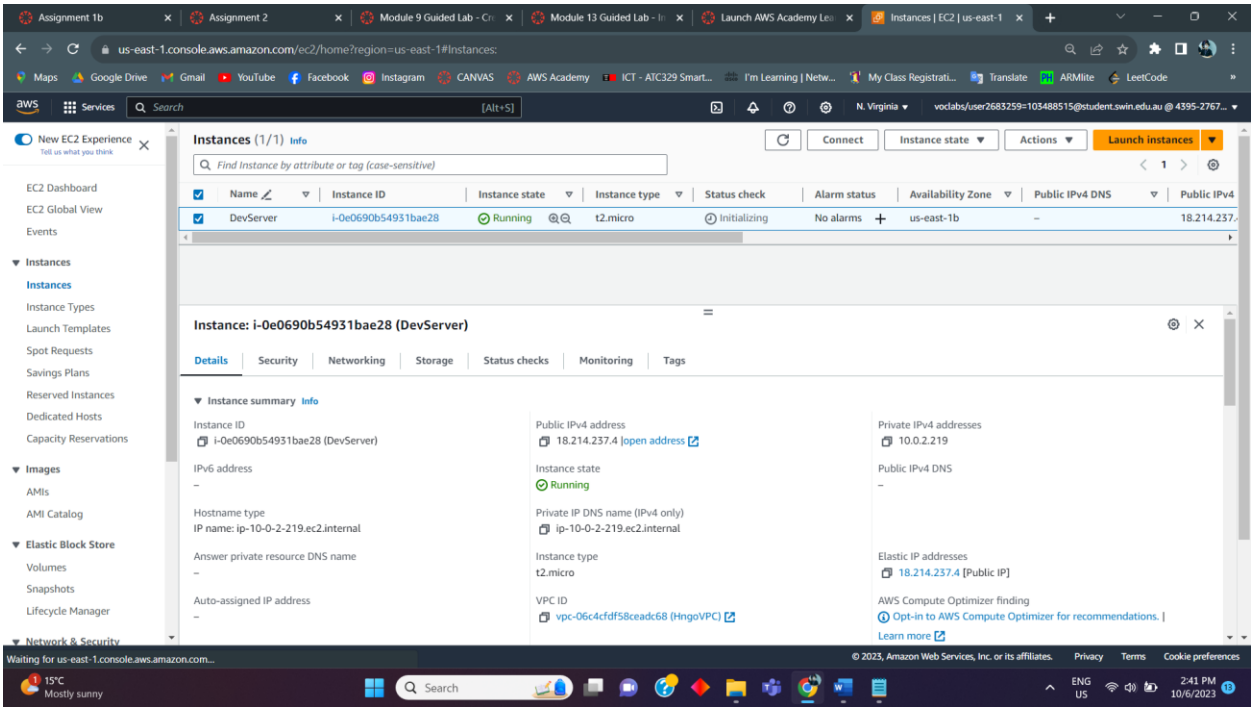


Figure 3 DevServer Instance is created correctly

It will be available in public subnet 2, have IAM Role “LabInstanceProfile” attached to it, and DevServerSG is also attached to this instance, however, the specification of the rule will be illustrated in the security group section.

V. DATABASE WITH RDS

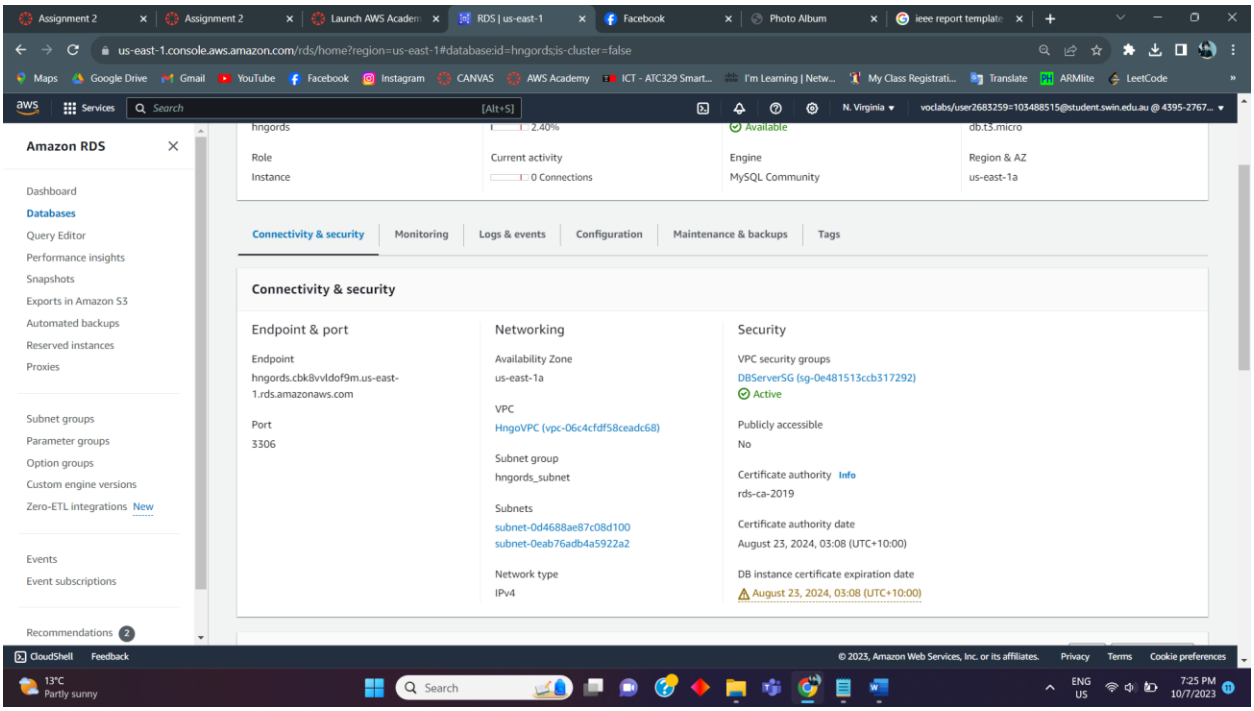


Figure 4 hngords is created correctly

The hngords database is created with HNgoVPC, the subnet group that was created above, and DBServerSG (which will be specified in the security group section).

Next step, DevServer Instance will be used to connect to putty, which will allow us to download aws file and phpMyAdmin file, which are all the information and resources needed to build the website, combined with this database.

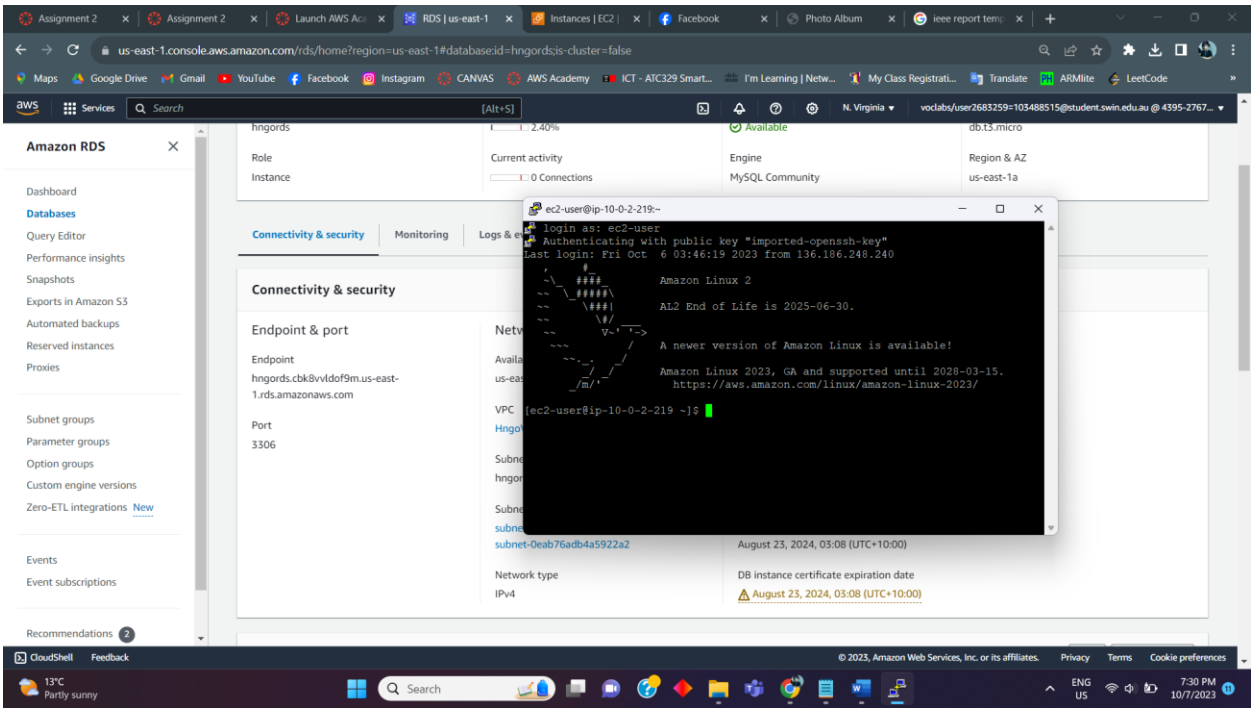


Figure 5 Access puty.exe by using DevServer Instance

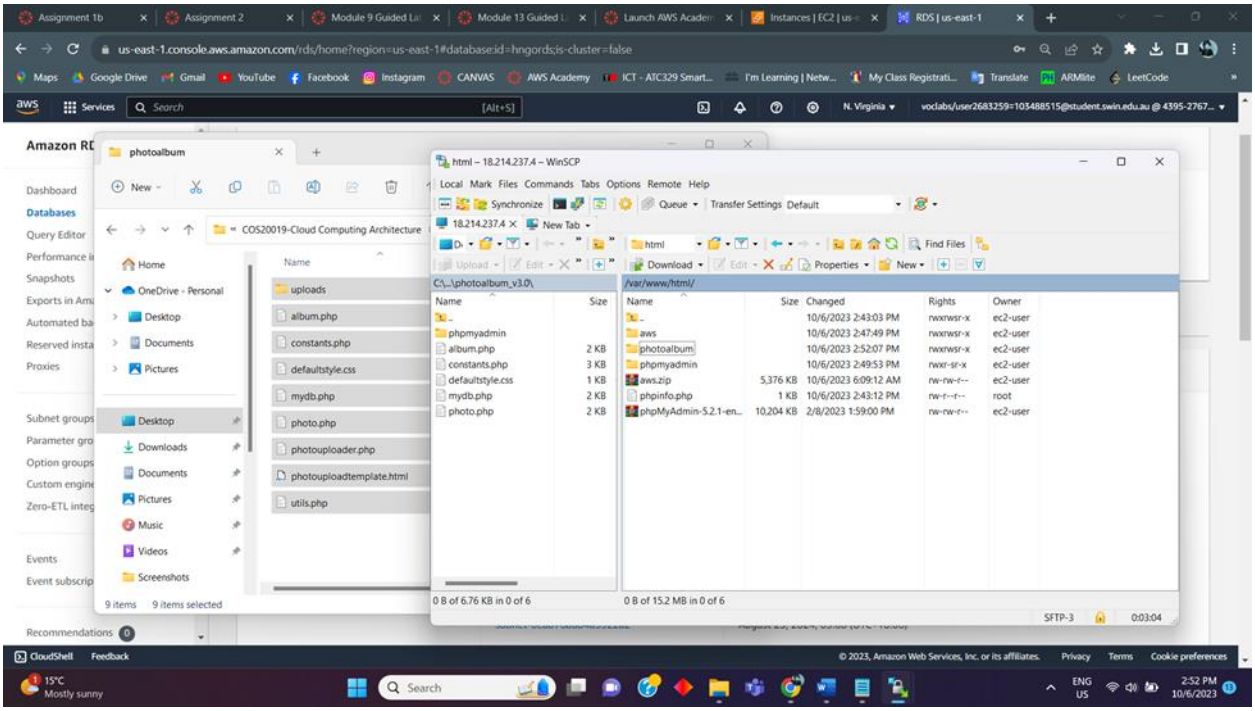


Figure 6 Access to WinSCP

Another file which is called “photoalbum” will be created to contain every file for the album website.

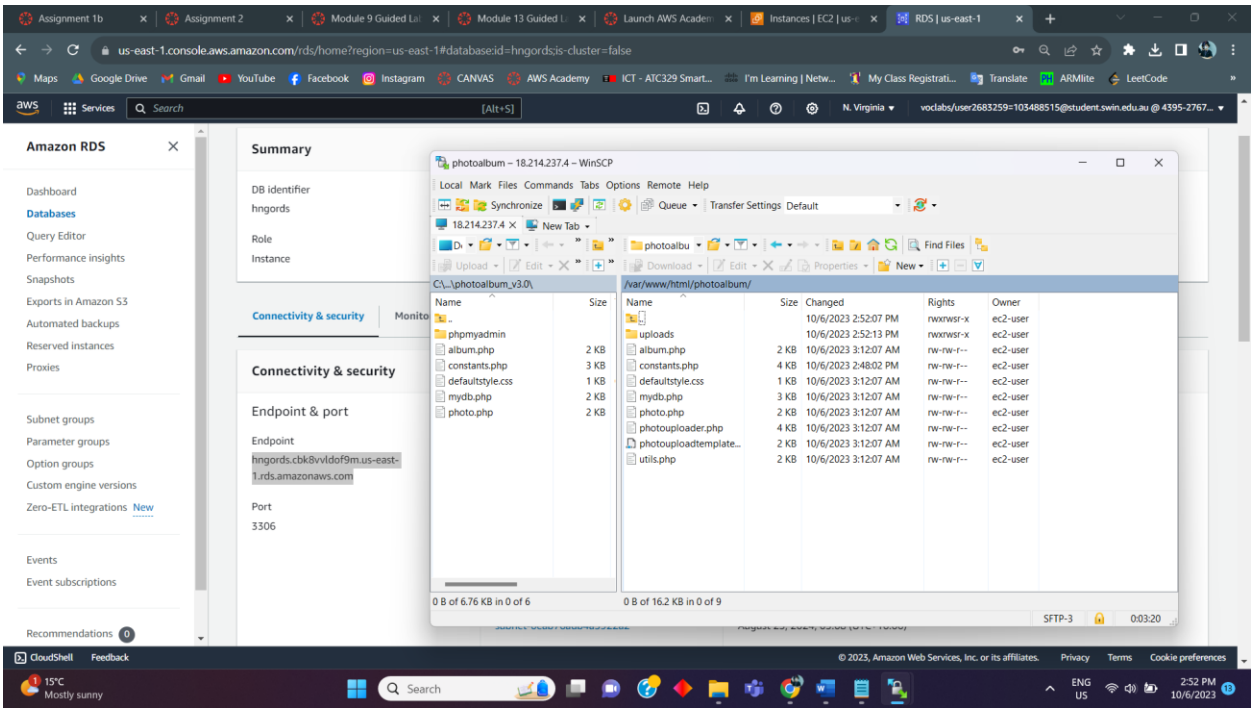


Figure 7 Files in the "photoalbum"

Next, using WinSCP, we access the resources that were downloaded earlier. We will modify the “config.sample.inc.php” to “config.inc.php”, and change the host server to the endpoint of RDS

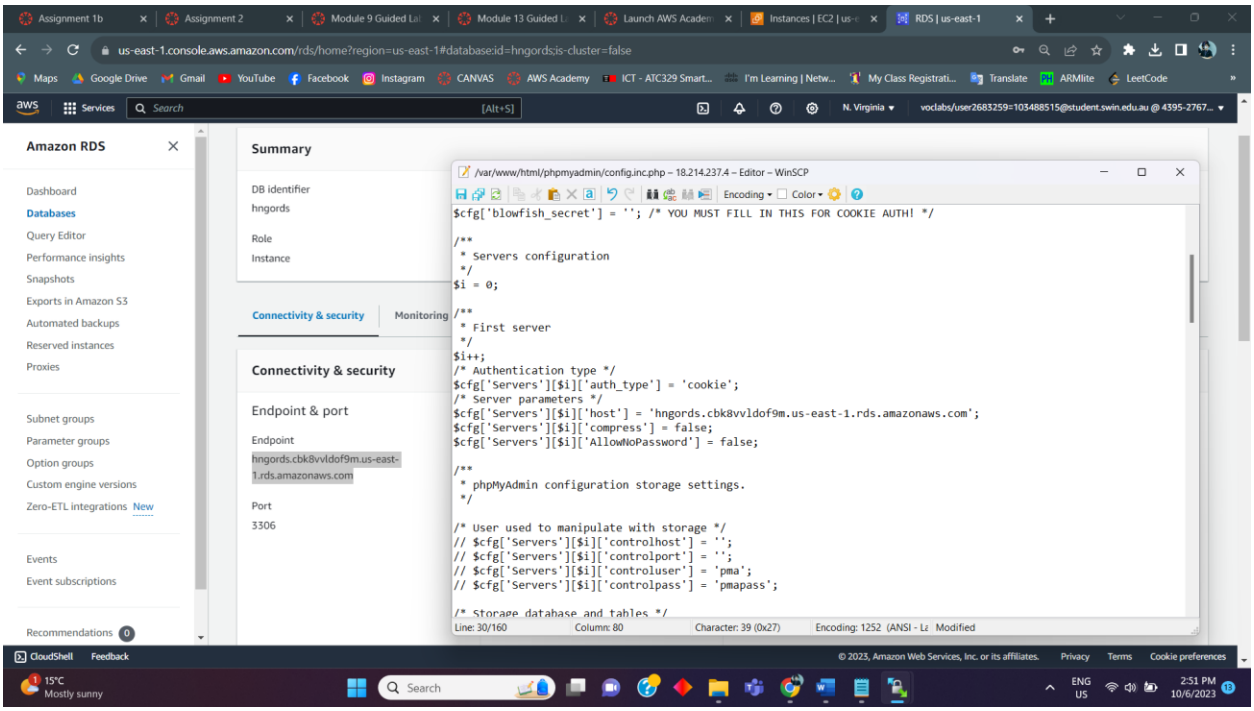


Figure 8 Modify the information to the endpoint of RDS

Finally, is accessing the phpMyAdmin to create a database and a table in that database, which contain all information needed for the photo album website.

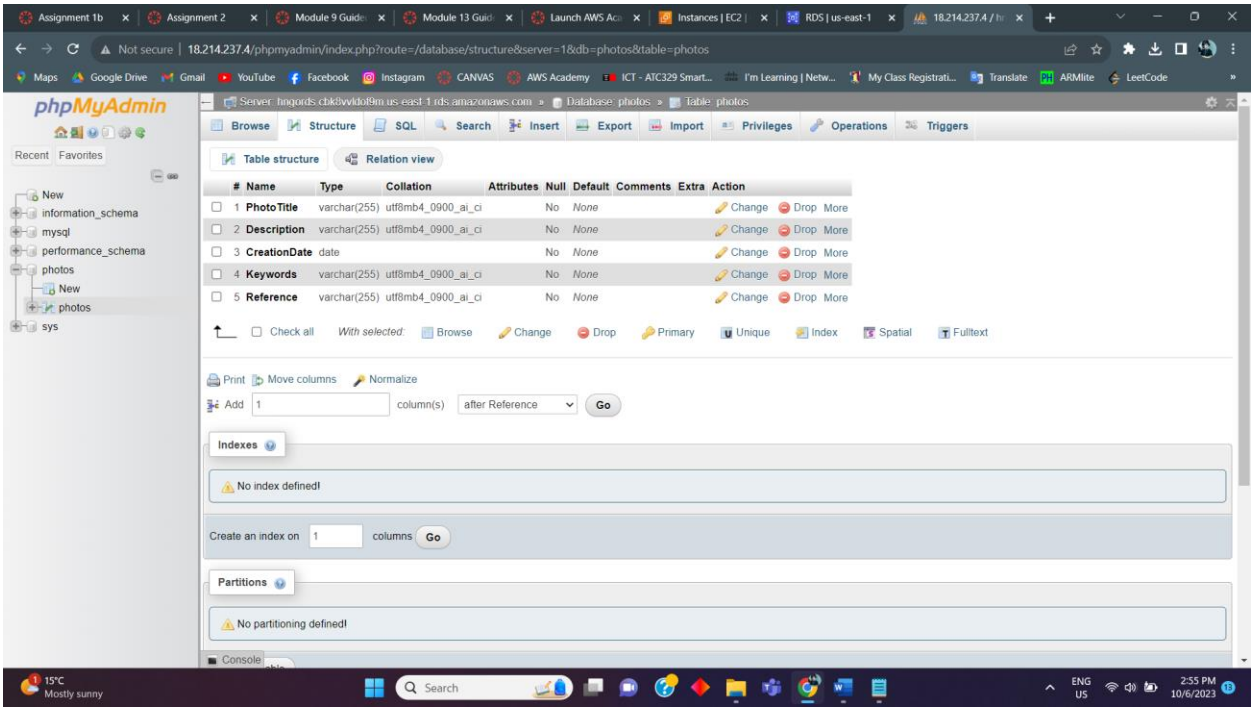


Figure 9 "photos" table inside the "photos" database

VI. LOAD BALANCING

I created a target group and a load balancer to support the auto scaling to create a highly available environment. Target group will be named “Hngo-TG” and it will be linked with the load balancer later.

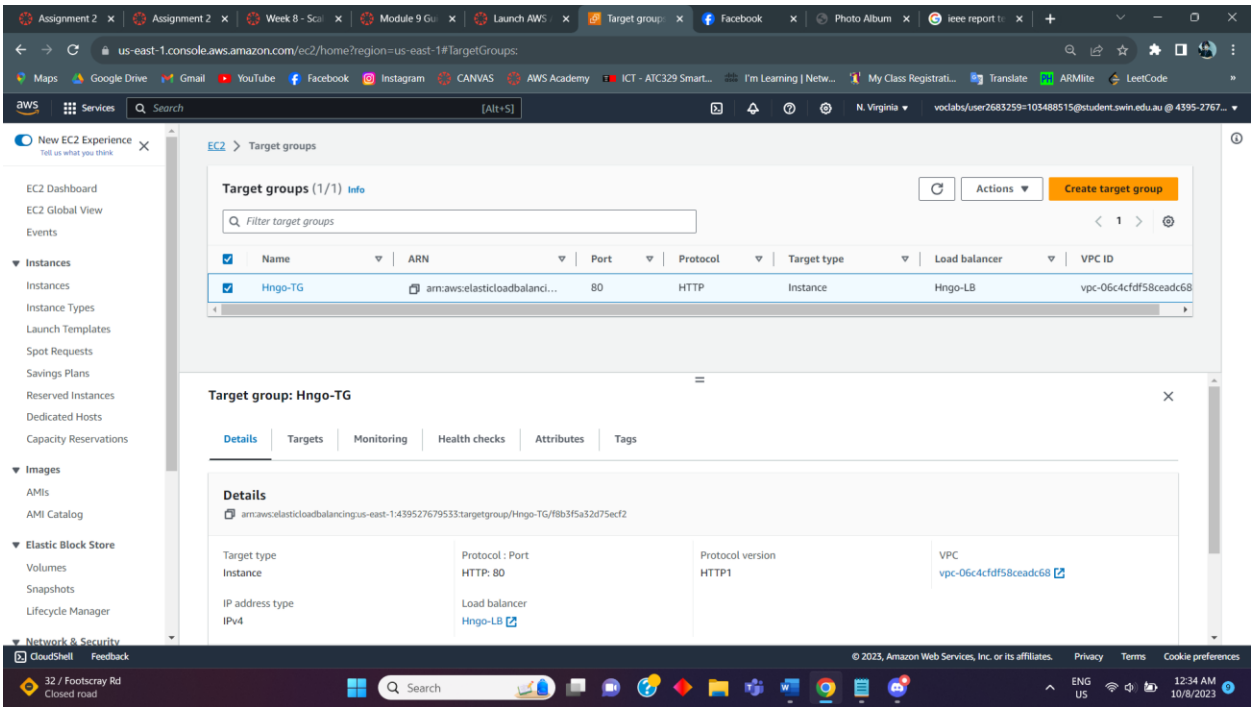


Figure 10 Hngo-TG (Target group) is created successfully

Next, the load balancer mentioned earlier will be created right after that. This load balancer will be placed in public subnets in HNgoVPC. And the most important thing is the health check path must be correctly configured as "/photoalbum/album.php".

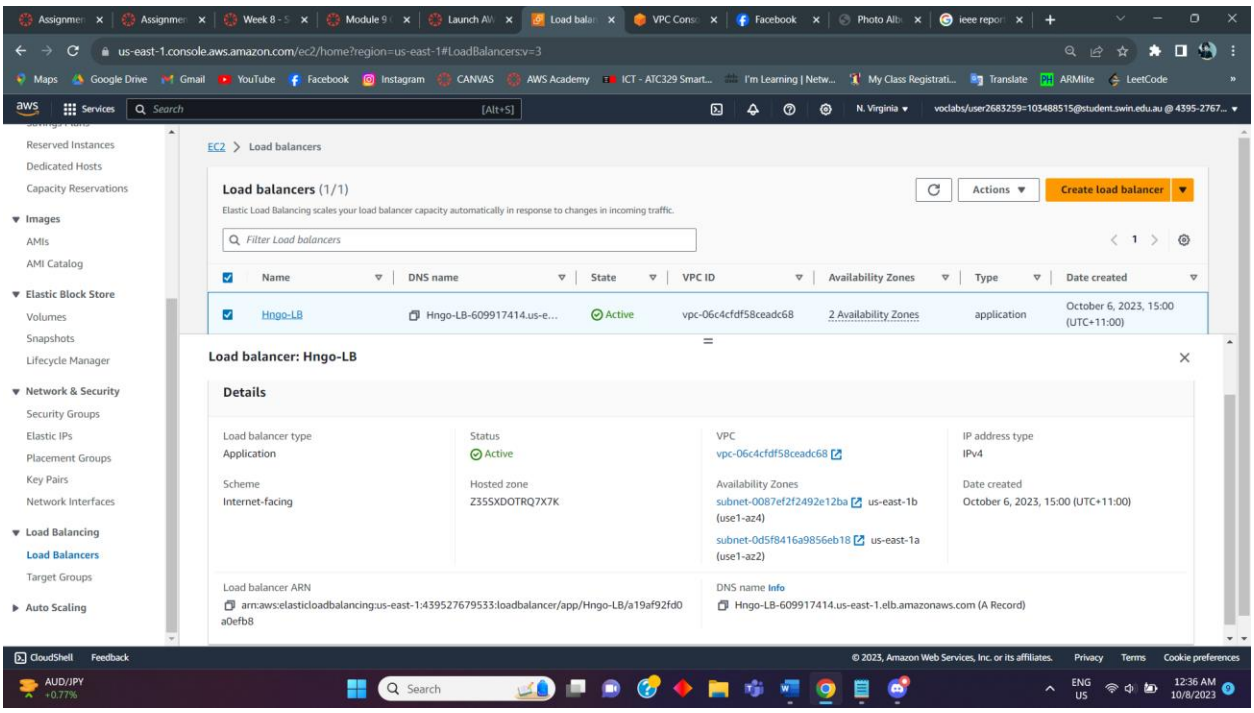


Figure 11 Hngo-LB (Load balancer) created successfully

As mentioned above, the load balancer will forward to target group Hngo-TG.

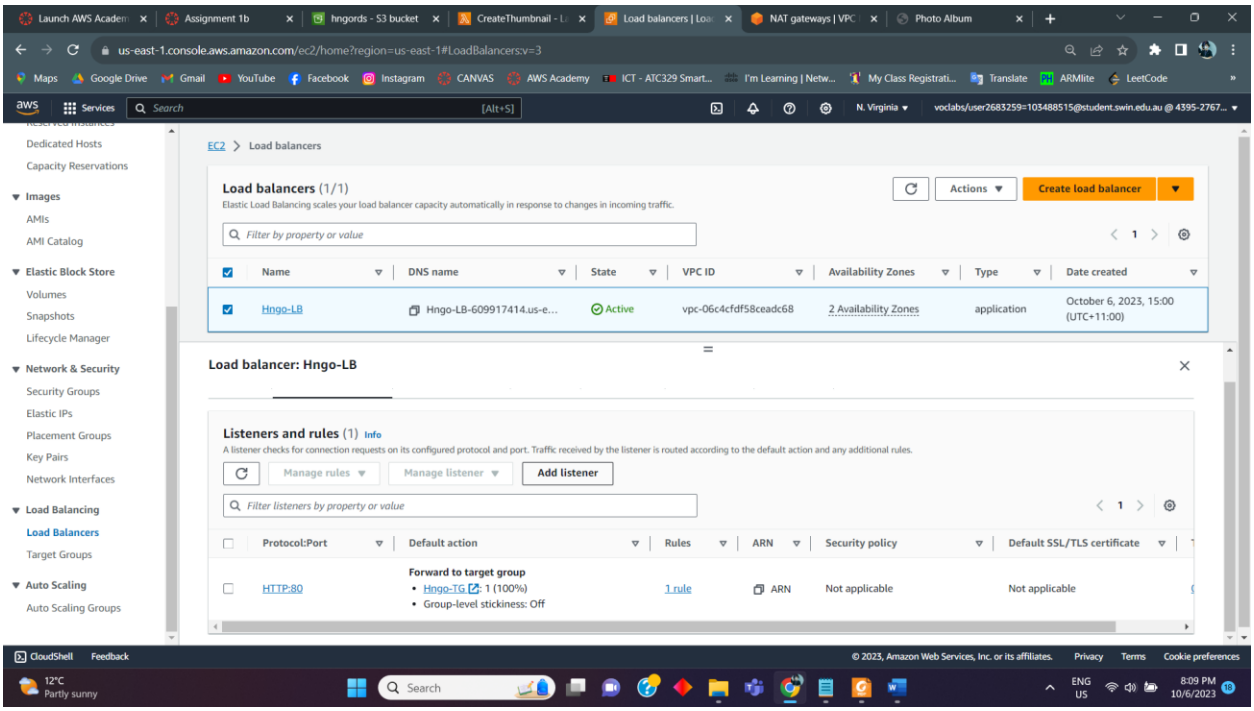
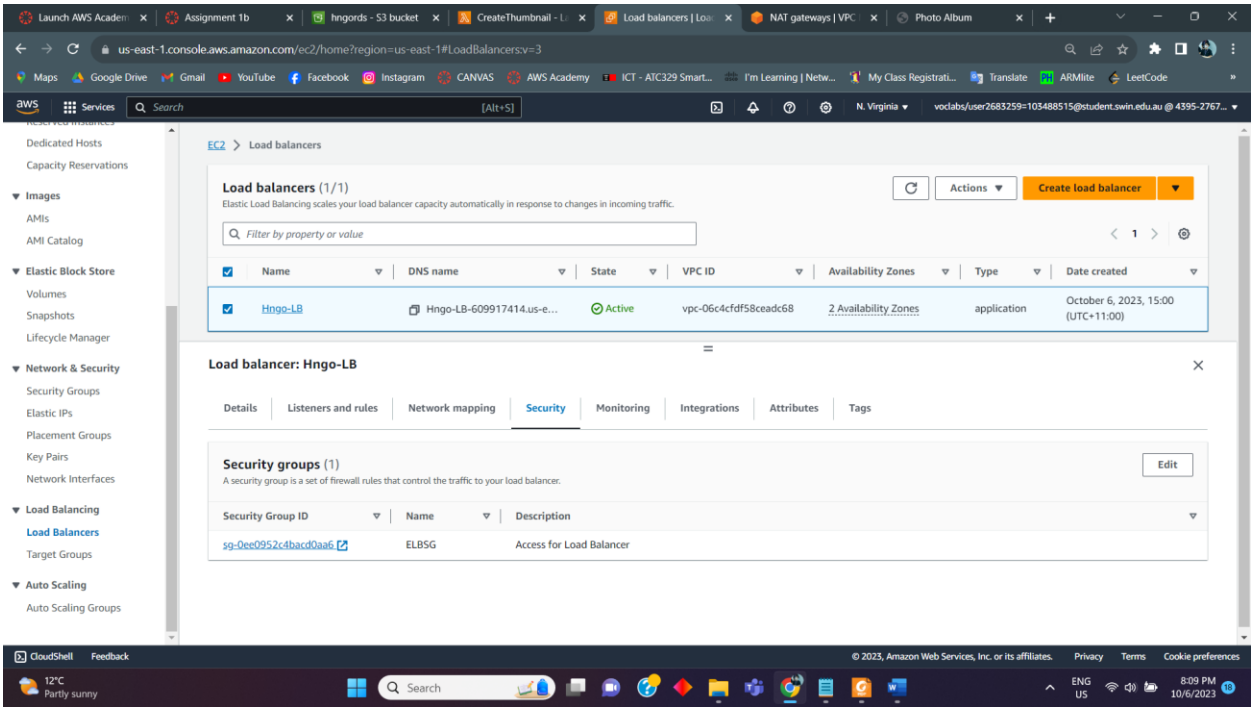


Figure 12 Hngo-TG linked with Hngo-LB

The ELBSG (Security Group for Load Balancer) is also applied correctly. Details for the configuration will be illustrated in the security group section.



VII. AUTO SCALING

In this stage, we need to create an AMI first to use for the auto scaling group.

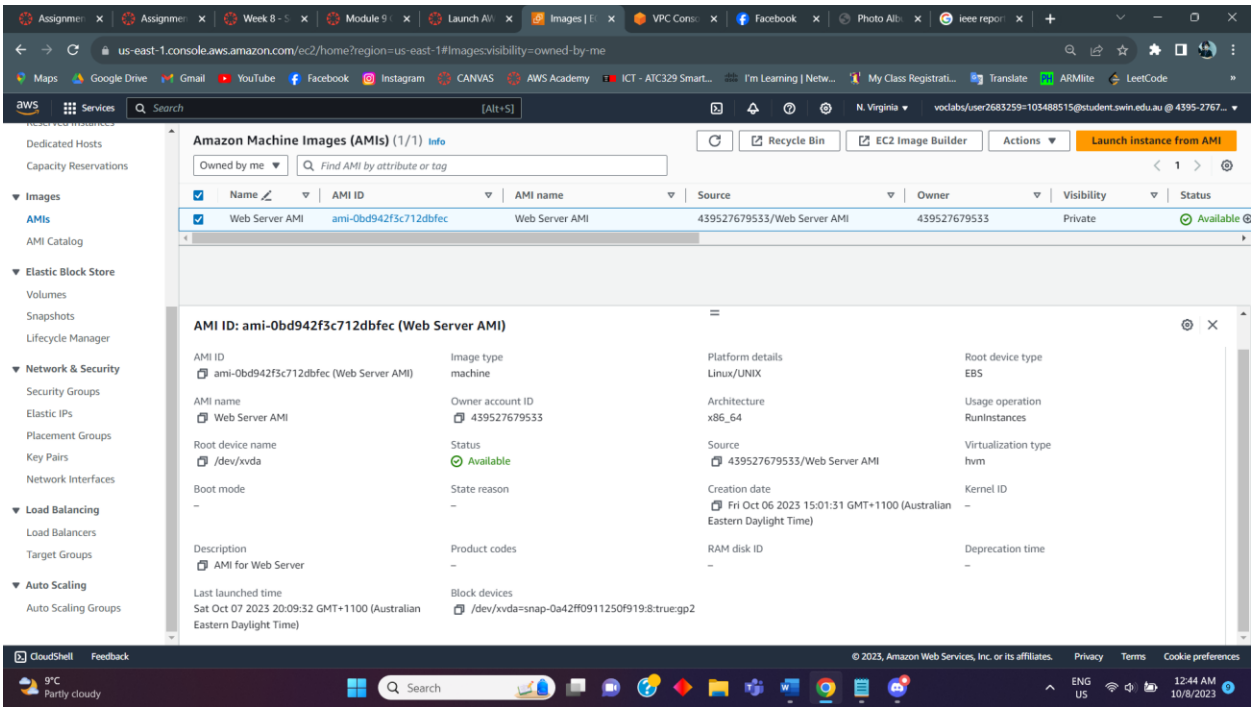


Figure 13 Web Server AMI is created by using DevServer Instance

After that, we will come to create the main part, auto scaling group with the requirement rules.

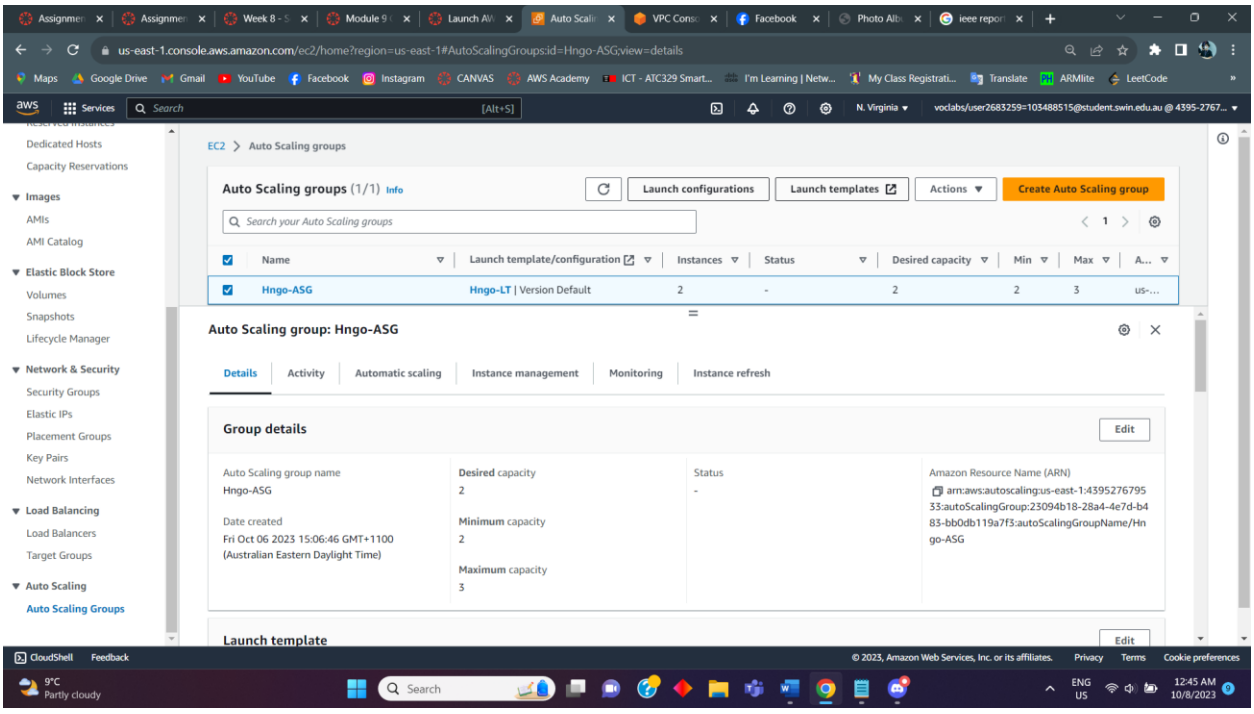


Figure 14 The minimum number of servers is 2. The maximum number of servers is 3

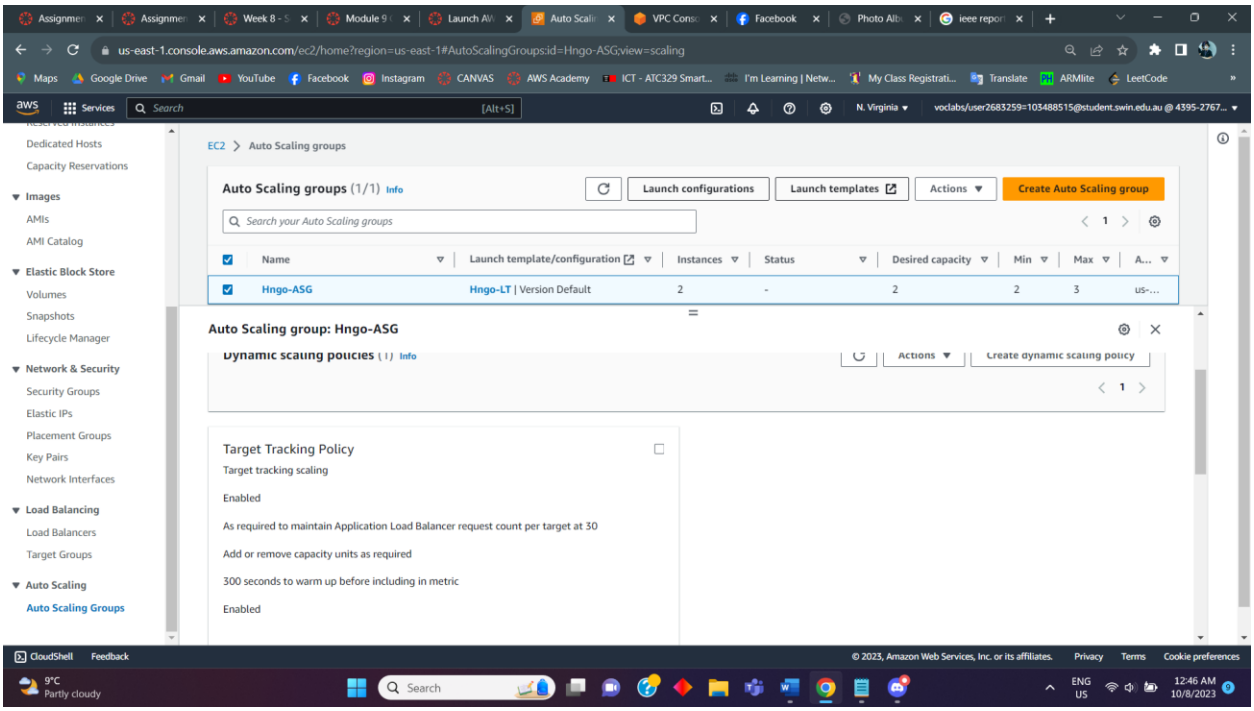


Figure 15 Target tracking scaling policy to keep the request count per target of the ELB

The Auto Scaling Group is placed in private subnet 1 and private subnet 2.

VIII. S3 PHOTO STORAGE

In this stage, I have created a s3 bucket also named “hngords”, with a bucket policy that restricts access to a specific HTTP referrer, which is my album website.

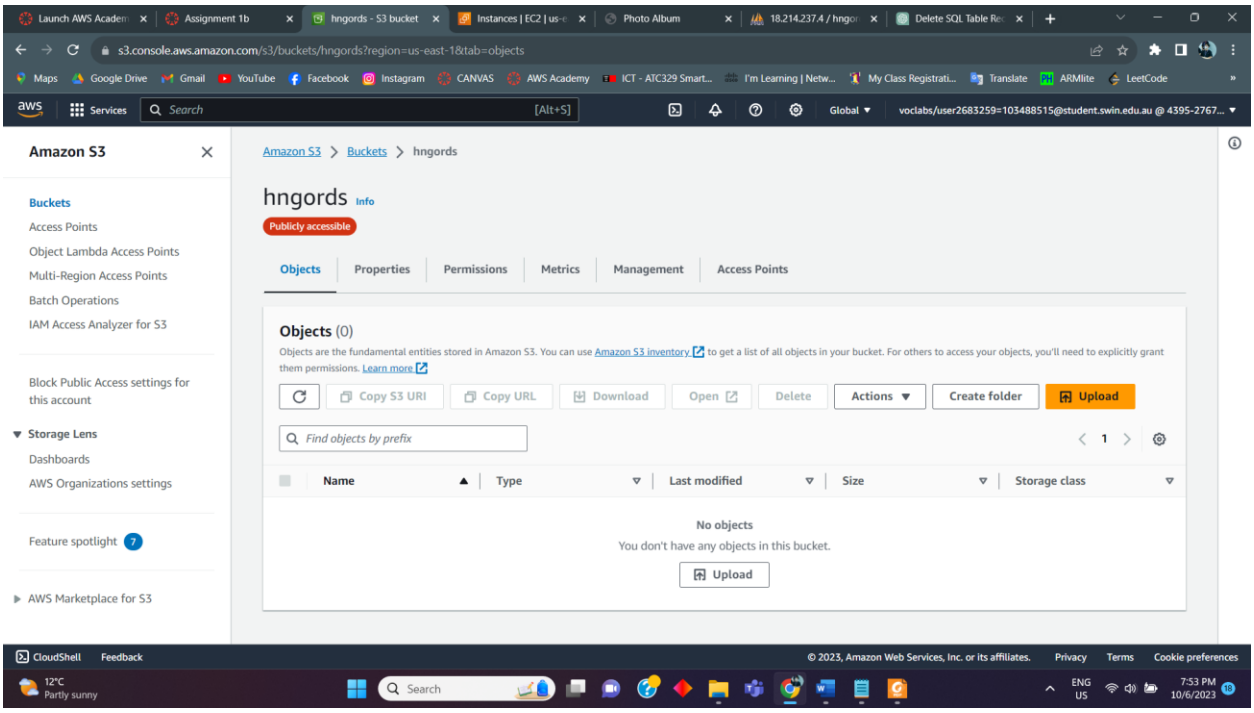


Figure 16 S3 bucket is created successfully

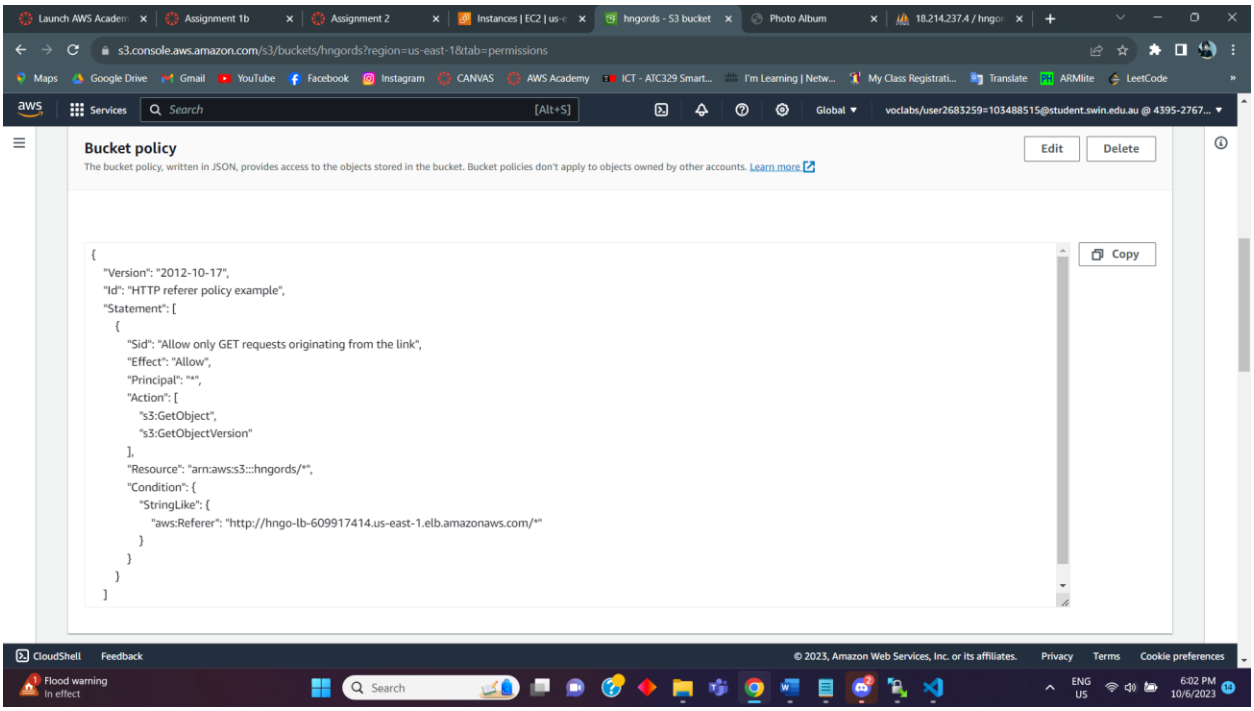


Figure 17 S3 Bucket Policy details

IX. CREATETHUMBNAIL LAMBDA FUNCTION

In this stage, I have created a Lambda function called “CreateThumbnail”. A zip file was uploaded to this lambda function.

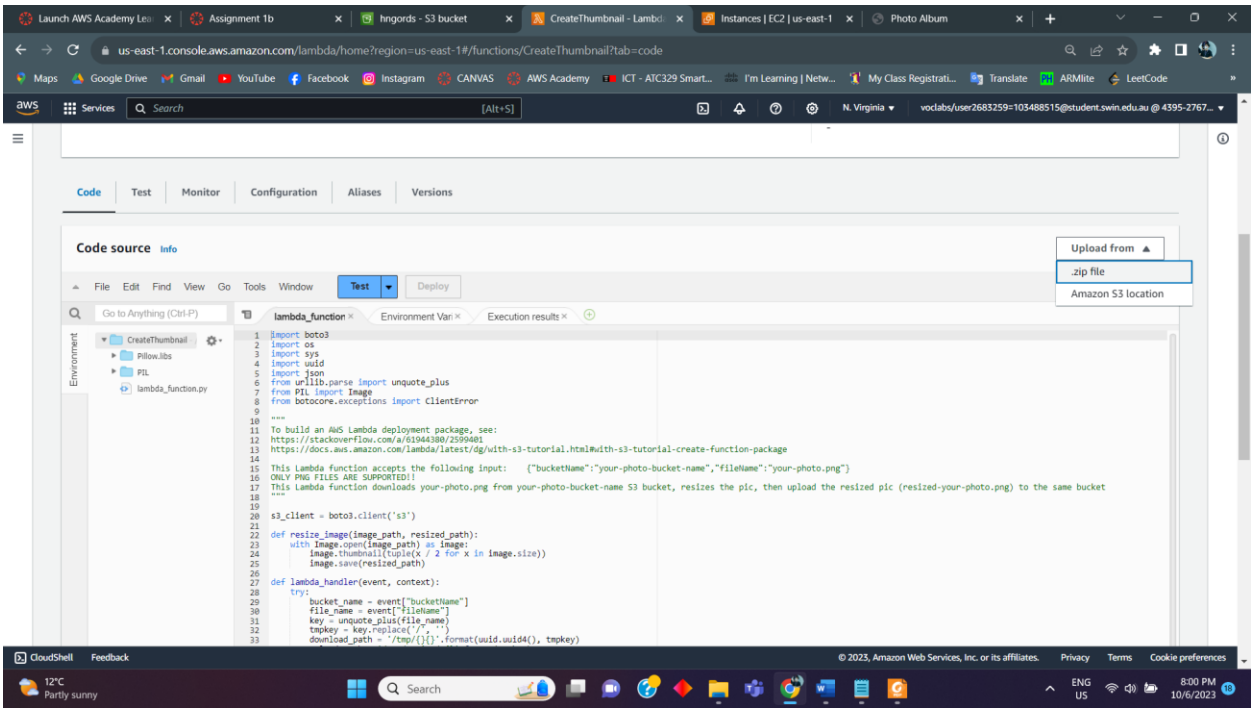


Figure 18 Source code is uploaded successfully

A test function will also be created to test the function in the testing section below.

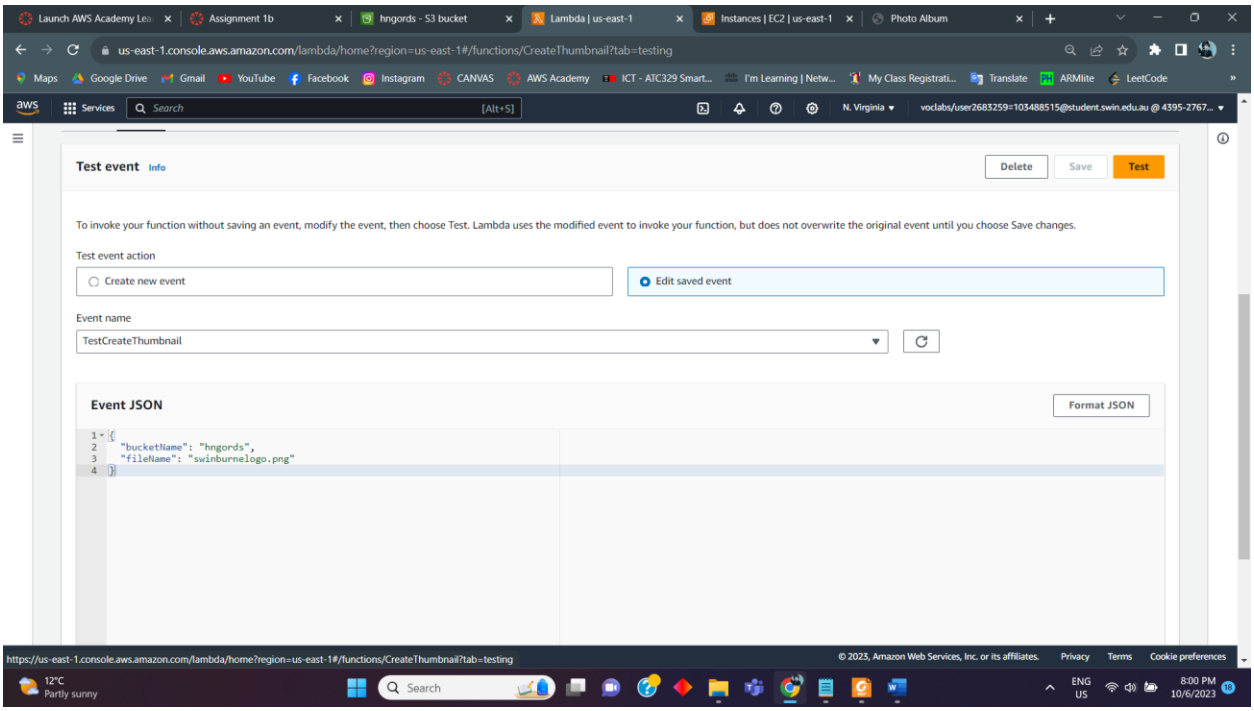


Figure 19 Testing function for s3 and a filename siwnburnelogo.png (please click the test button to see details)

X. SECURITY GROUP

At this stage, the infrastructure has almost been completed, the thing that needs to be tightened will be security group. There are four security groups in total since I used NAT Gateway, so NAT Security group is not necessary. ELBSG, WebServerSG, DBServerSG must follow the least-privilege principle

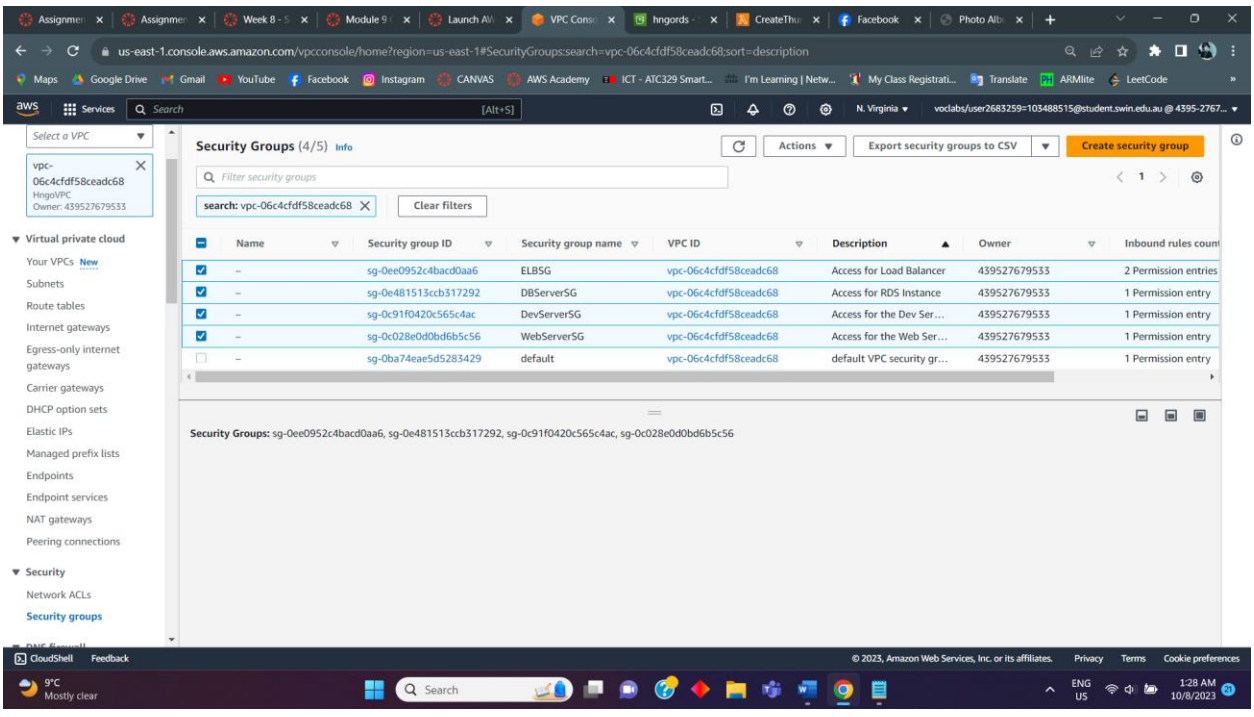


Figure 20 4 Security Groups in HNgoVPC

ELBSG allows HTTP and HTTPS from anywhere. WebServer will take HTTP traffic from ELB, and DBServer will take SQL traffic from the WebServer, making it become a 3-tier security. The DevServerSG allows all traffic from anywhere.

XI. NACL

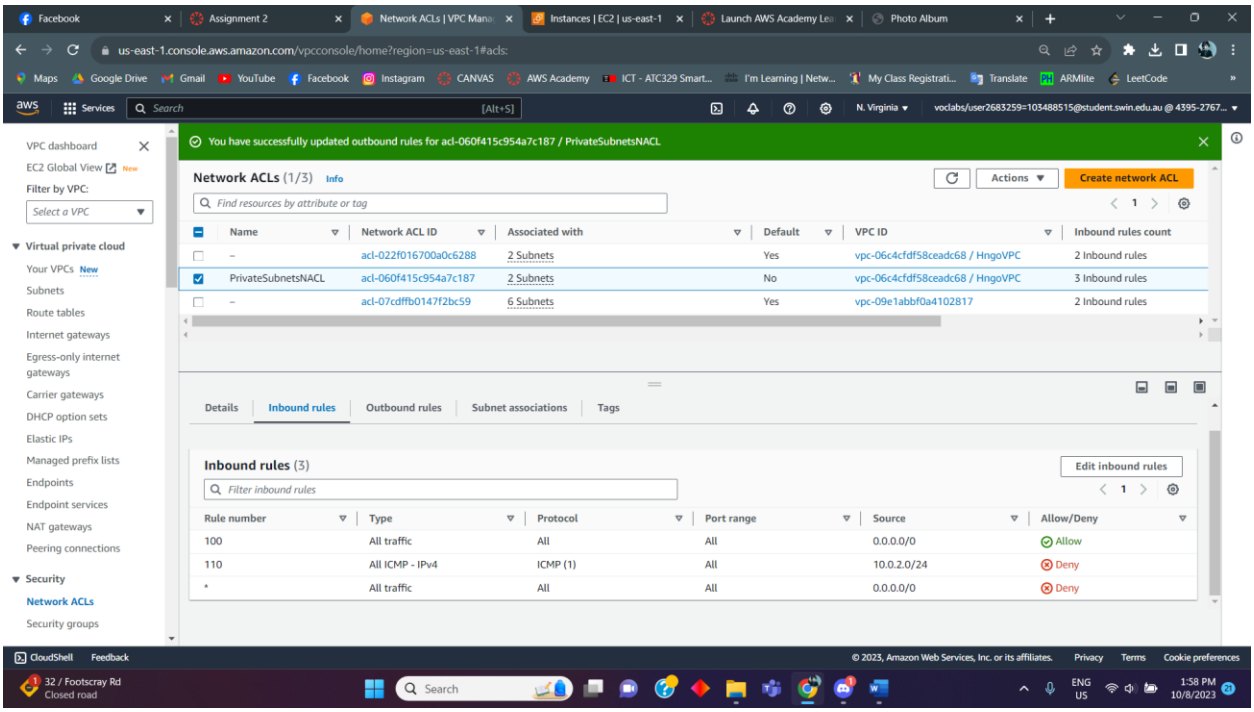


Figure 21 NACL in private subnet 1 and private subnet 2, allow everything except ICMP from/to DevServer

XII. TESTING

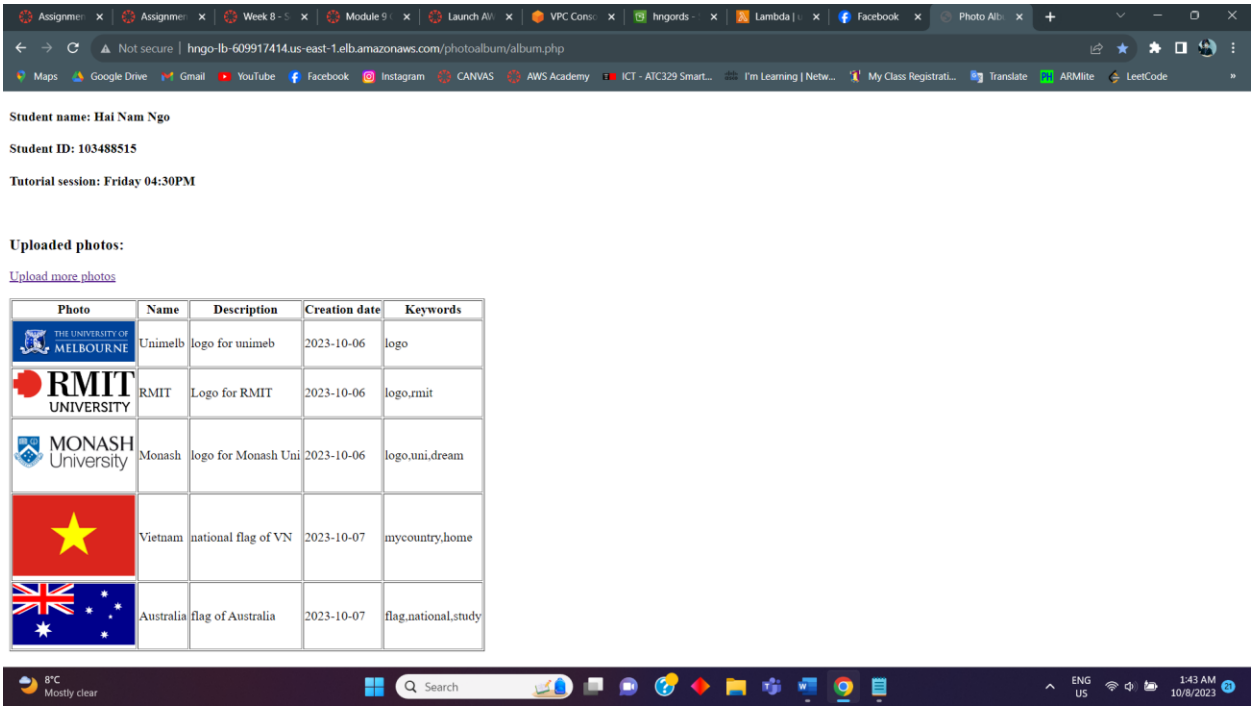


Figure 22 The PhotoAlbum website should be accessible

The swinburnelogo.png file is uploaded directly to S3 bucket to test the lambda function, for more details please come to Lambda Function to specify.

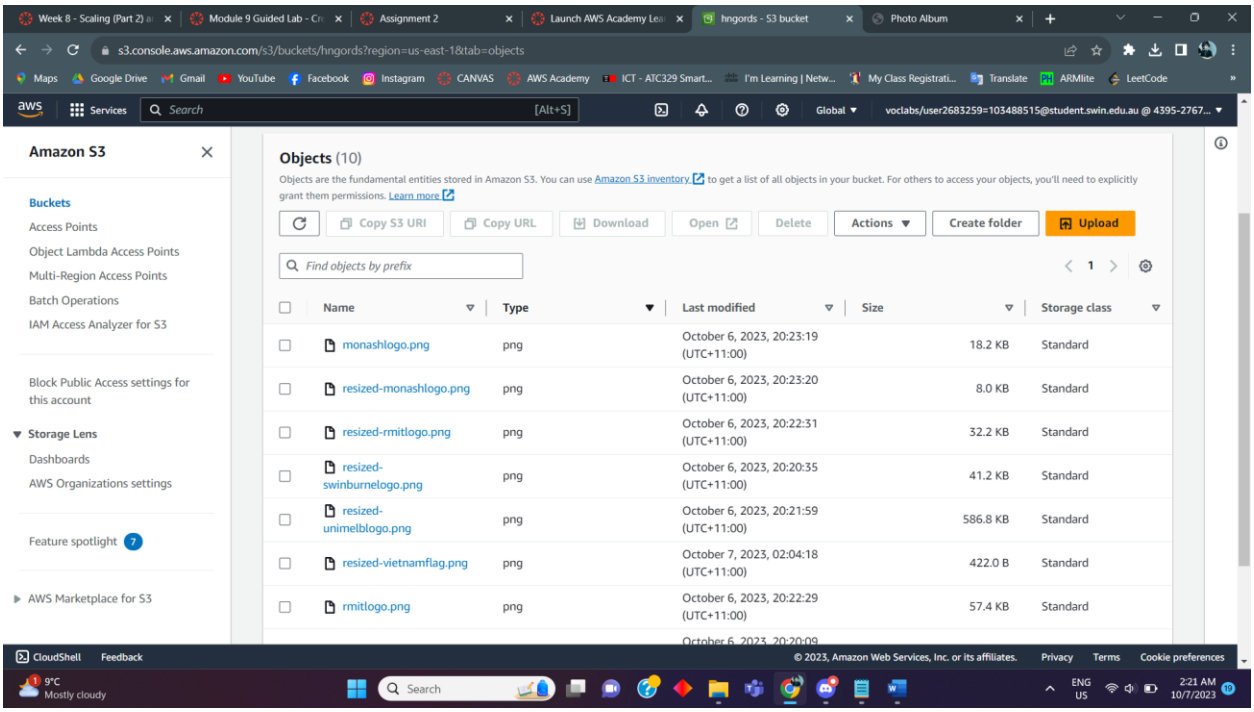


Figure 23 The S3 bucket to see if photos are uploaded and their resized versions are created

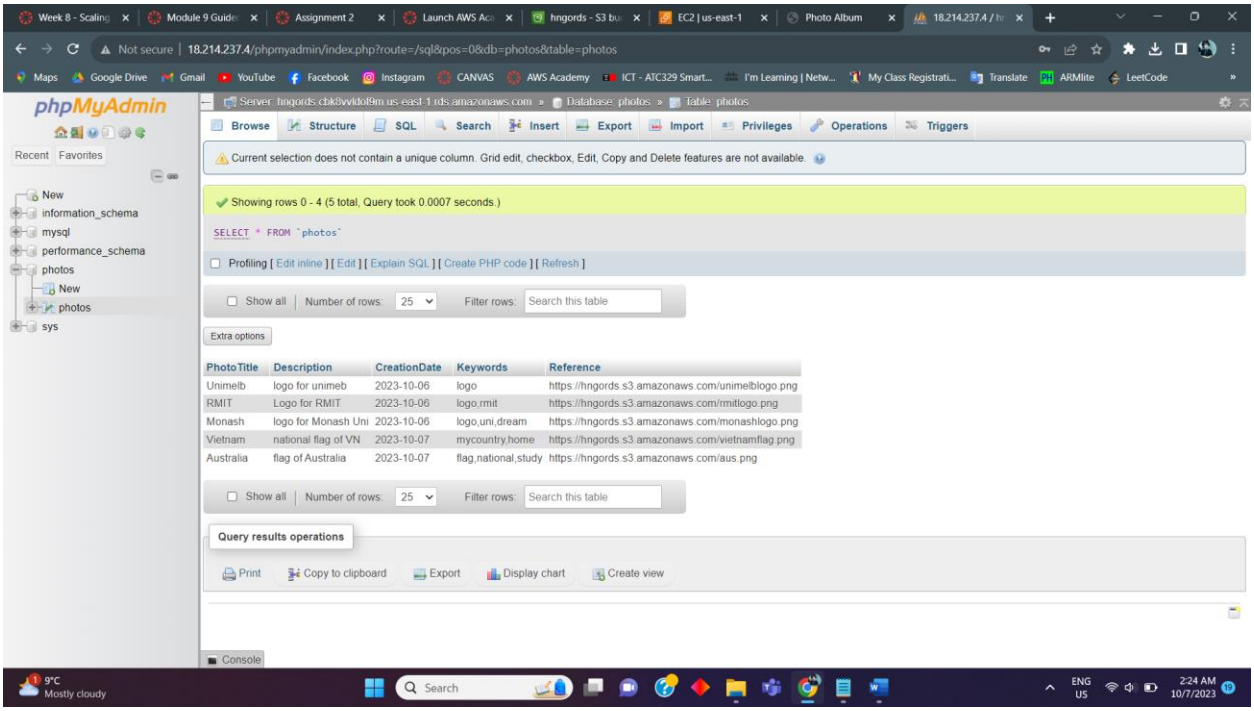


Figure 24 Check the database to see if their meta-data is recorded

The PhotoAlbum website is accessible through the load balancer only, cannot access to the same way as assignment 1b: http://18.214.237.4/photoalbum/album.php (not accessible).

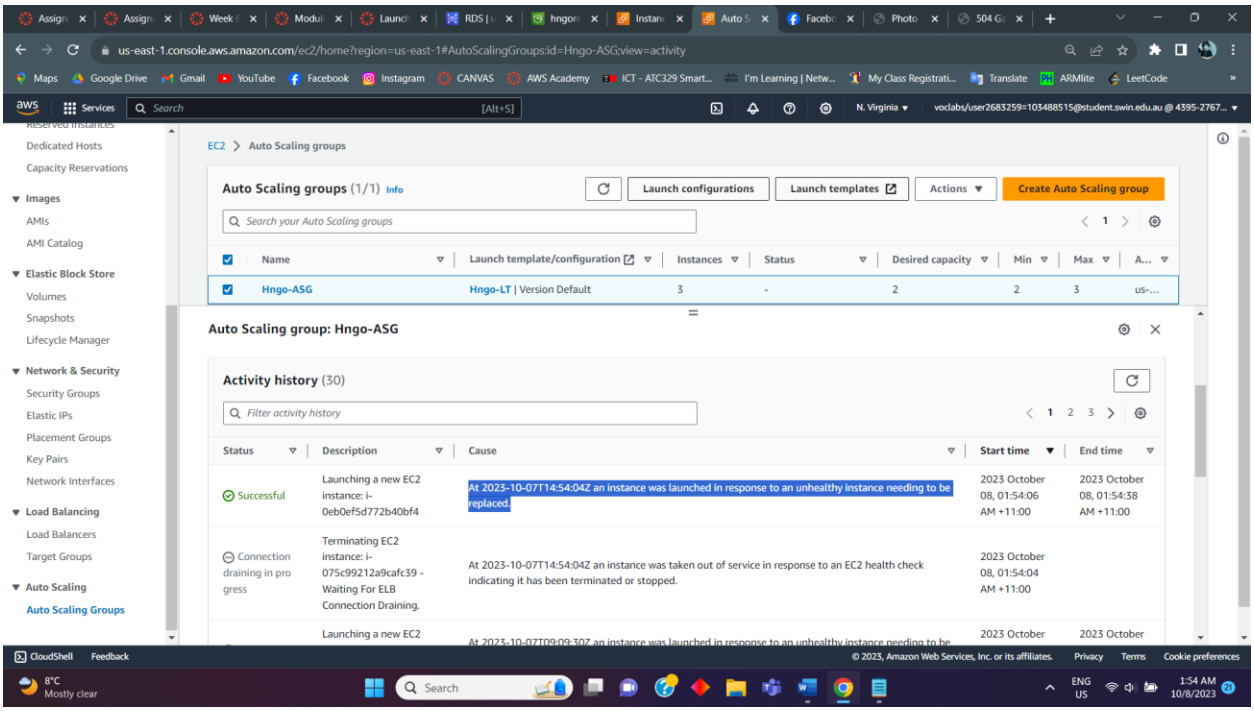


Figure 25 ASG launched a new WebServers Instance after user terminated a WebServers Instance earlier

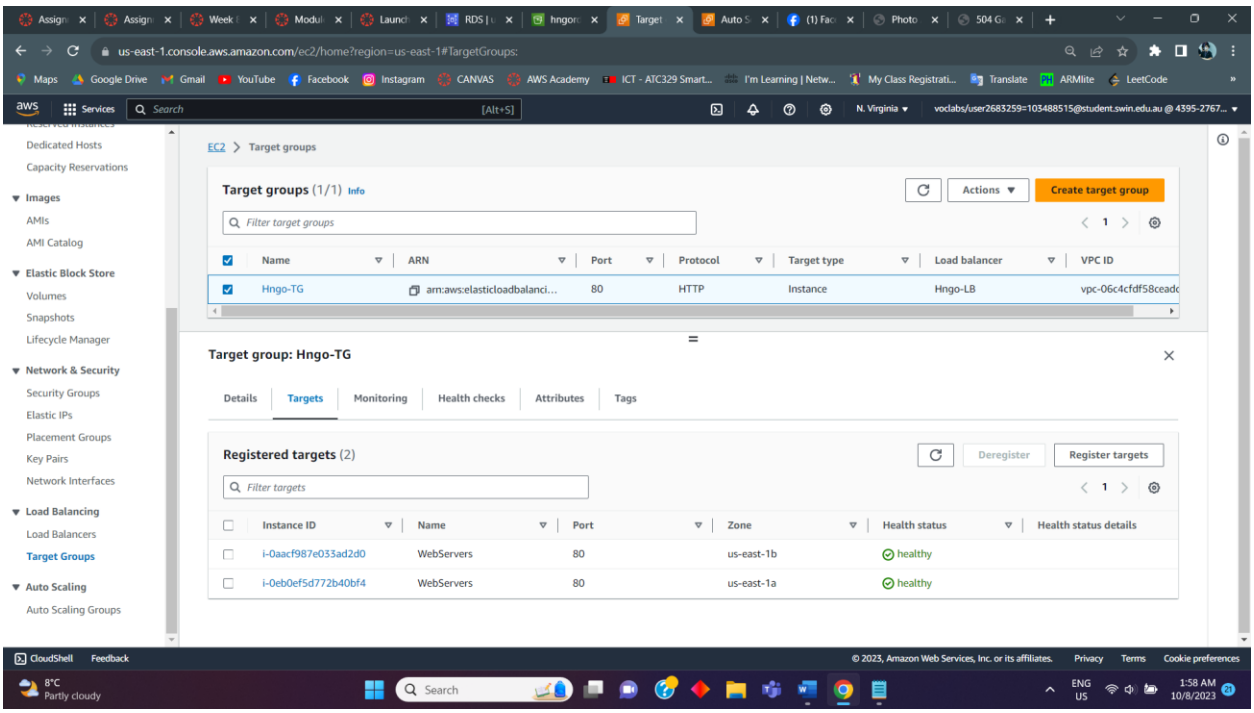


Figure 26 All EC2 targets are healthy

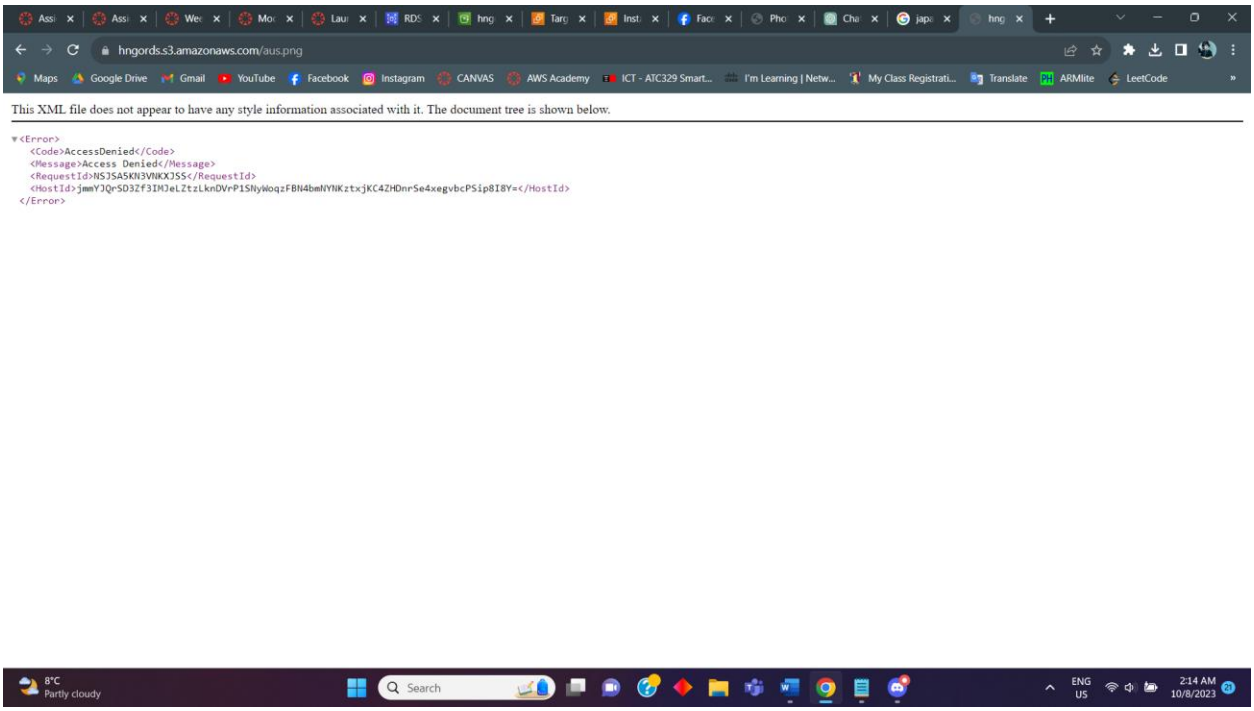


Figure 27 Test direct access to S3 photos, which should not be publicly accessible

Access putty.exe by using DevServer and I ping one of the WebServers Instance to see if they can ping each other or not, and the result shows that it is not possible to ping between those instances, which means that the rule is working.

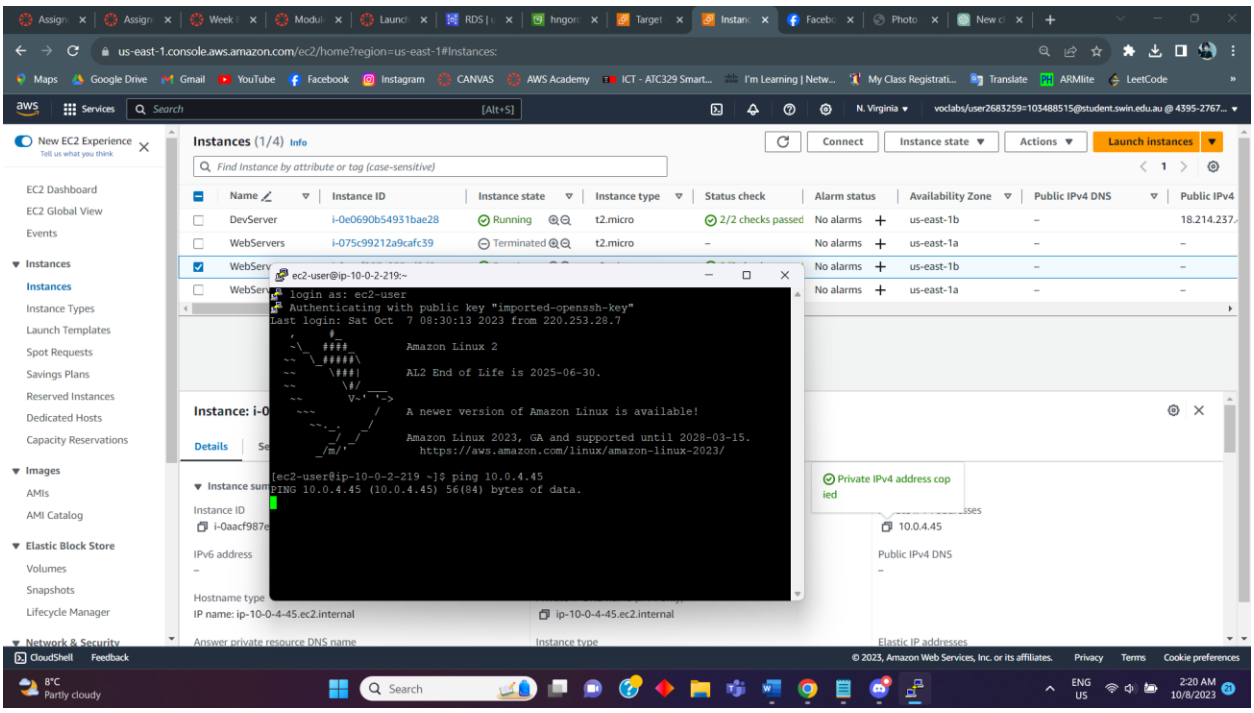


Figure 28 Test the Network ACL bidirectional functionality by sending ICMP traffic between the WebServers and DevServer.