COS20019 Assignment 2

Luu Tuan Hoang
104180391
Tutor class: 8.00am Saturday
Swinburne University of Technology
Hanoi, Vietnam
104180391@student.swin.edu.au

***Abstract*— This paper presents an extended AWS infrastructure with an emphasis on improving interactions between EC2, Lambda, and S3 services. The goals include developing a Lambda function, using custom AMIs, implementing auto scaling with launch configurations, utilizing elastic load balancers, enforcing access control using AWS NACLs, and enforcing access control through S3 bucket policies. By accomplishing these goals, the infrastructure shows enhanced security, scalability, and performance, allowing effective interactions between EC2, Lambda, and S3 services inside the AWS environment.**

***Index terms*—Cloud Computing, System Architecture, Cloud Service, Computing Power**

I. Introduction

The Photo Album website project involves using EC2 web servers to host the website and a variety of AWS services, including S3, RDS, and Lambda. Users can upload photographs to the website, explore photo albums, and have thumbnails that have been automatically resized created. To integrate the website with the S3 bucket, RDS database, and Lambda function, the whole source code and instructions have been made available. This project uses the power of AWS services to create a dynamic and user-friendly picture album experience with features like photo retrieval, uploading, and thumbnail creation.

II. Website Architecture

A. Infrastructure requirements

1) *Virtual Private Cloud (VPC):* VPC configured with 2 Availability Zones (AZs) both with public and private subnets. The public subnets are associated with a route table that directs traffic to an Internet Gateway (IGW). On the other hand, the private subnets are associated with a separate route table that routes traffic to NAT gateway (In my architecture, I use NAT gateway instead of NAT instance).

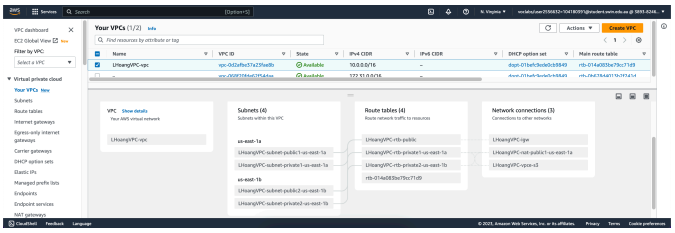


Figure 1: VPC Resource Map (NAT Gateway included)

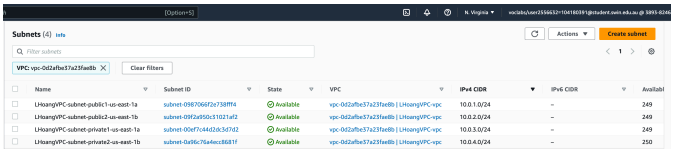


Figure 2: VPC Subnet CIDR Block

The VPC is ready for the photo album web server to be launched.

2) *Security group:* There are a total of four security groups in the architecture: ELBSG, WebServerSG, DBServerSG, DevServerSG. Since NAT gateway is used instead of NAT instance, NATServerSG is not necessary. Outbound rule of all security group set to default (All traffic, anywhere IPv4).

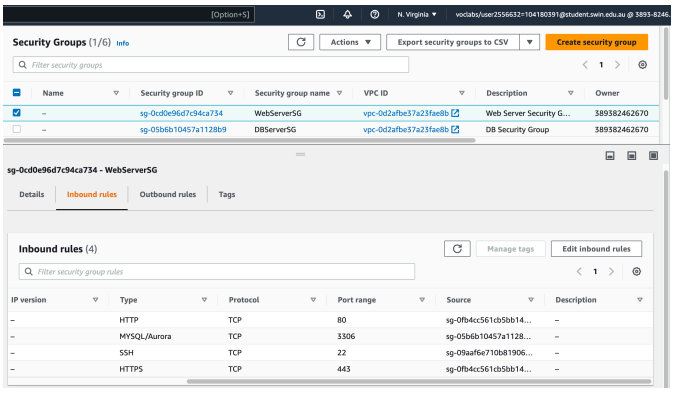


Figure 3: WebServerSG Security Group

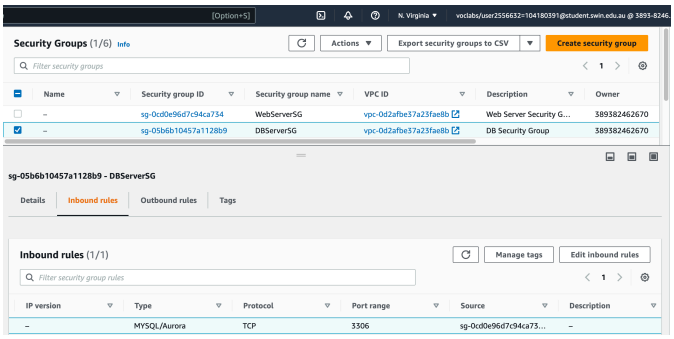


Figure 4: DBServerSG Security Group

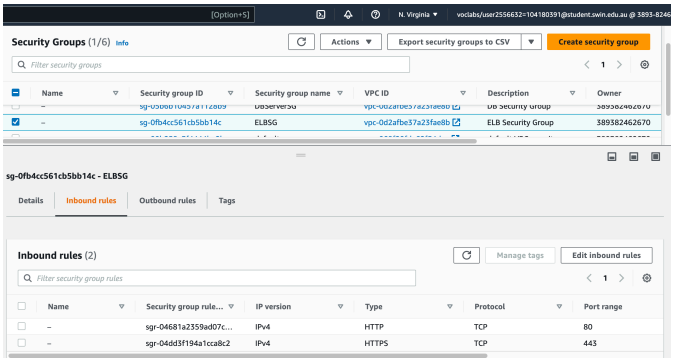


Figure 5: ELBSG Security Group

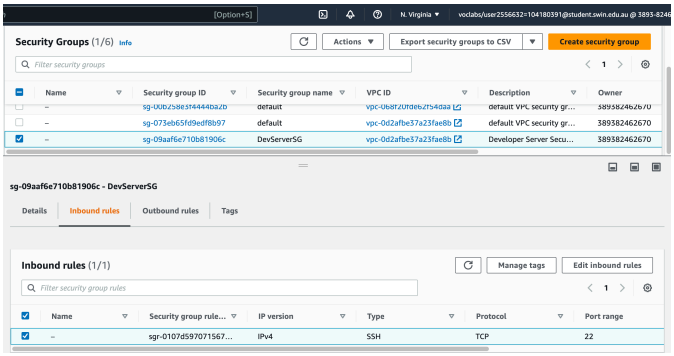


Figure 6: DevServerSG Security Group

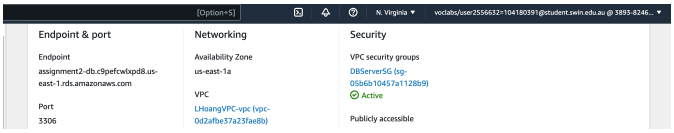


Figure 7: DBServerSG Security Group attached to RDS

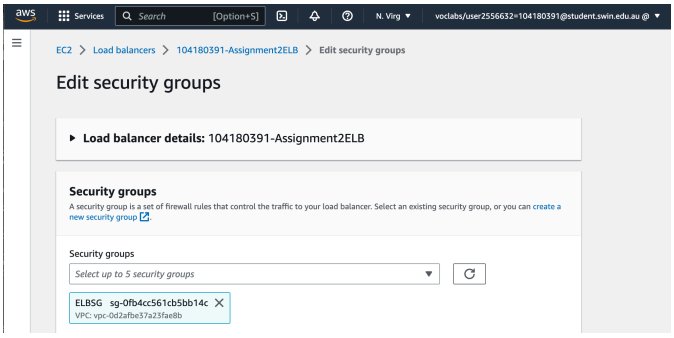


Figure 8: ELBSG Security Group attached to ELB

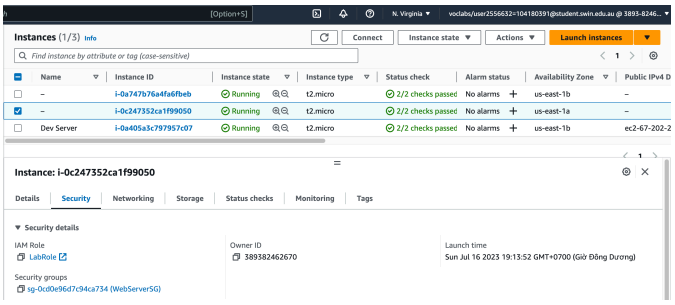


Figure 9: WebServerSG Security Group attached to Launch instance

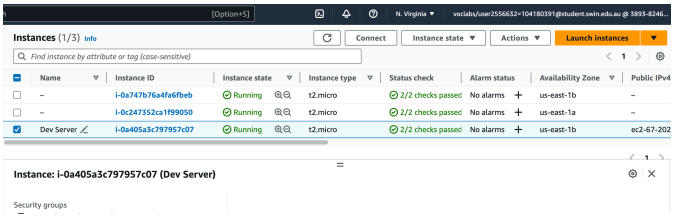


Figure 10: DevServerSG Security Group attached to Dev Server instance

Security group name	Protocols	Source
WebServerSG	HTTP (80), HTTPS (443)	ELBSG
	SSH (80)	DevServerSG
	MYSQL/Aurora (3306)	DBServerSG
DBServerSG	MYSQL/Aurora (3306)	WebServerSG
DevServerSG	SSH (80)	Anywhere-IPv4
ELBSG	HTTP (80), HTTPS (443)	Anywhere-IPv4

Figure 11: Inbound security rules summary

The architecture is now secured well with least privillege principal.

3) *Network ACL (NACL):* A Network Access Control List (NACL) called “PrivateSubnetsNACL” was created and implemented to improve the security of the web servers in the private subnets. This NACL’s function is to limit ICMP (Internet Control Message Protocol) traffic going to and coming from the Dev Server in both directions.

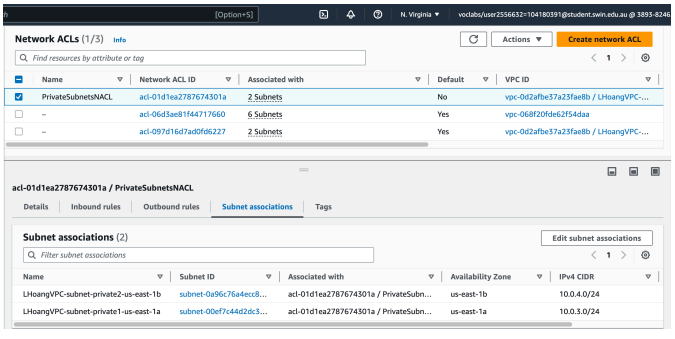


Figure 12: NACL associated with private subnets in the VPC

ICMP protocol from Public Subnet 2 (CIDR 10.0.2.0/24) which Dev Server resides in is blocked, both inbound and outbound. Rules are listed in Figure 13 and Figure 14.

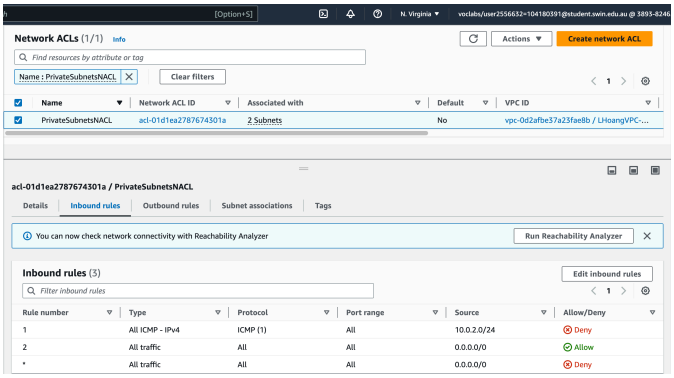


Figure 13: Inbound rules of *PrivateSubnetsNACL*

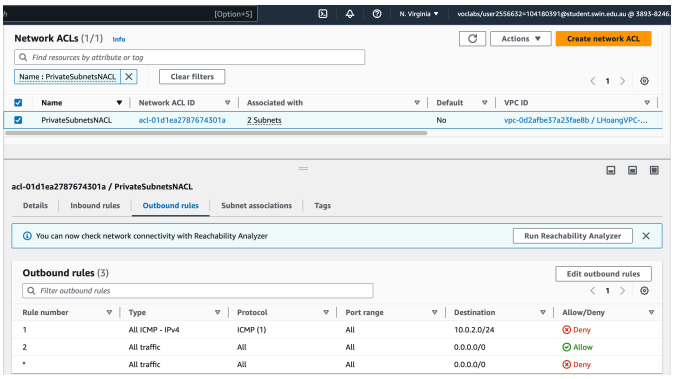


Figure 14: Outbound rules of *PrivateSubnetsNACL*

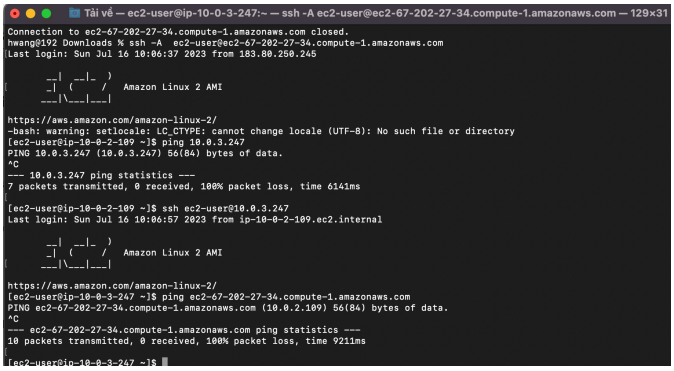


Figure 15: Testing the NACL

To test the NACL, make SSH connection to Dev Server via elastic IP that was allocated for Dev Server. Ping to private IP of EC2 instance from Web Server ASG. Next, make SSH connection from Dev Server to EC2 instance with private IP address and ping to Dev Server [1]. From Figure 15 we can clearly see that the NACL is preventing any communication via ICMP packets to and from the Dev Server.

4) *IAM Role:* The management console already has IAM roles with the necessary permissions called “LabRole” and “Labinstancerole” that can be used for this assignment.

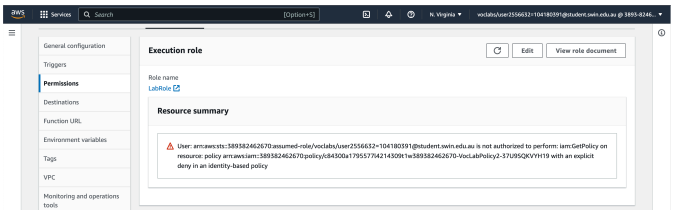


Figure 16: *CreateThumbnail* Lambda function execution role

The Lambda function was permitted with an IAM execution role to ensure the proper security and permissions. The Lambda function is granted access to and control over the objects in the specified S3 bucket by the IAM role named *LabRole*, which was already created with the least privilege principle.

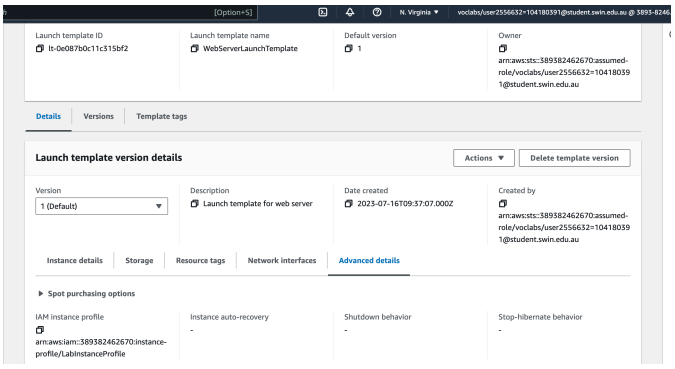


Figure 17: *CreateThumbnail* Launch template IAM role

Following the principle of least privilege, an IAM role named *LabInstanceProfile* was created to grant the web server the required permissions. The role was set up to give the designated S3 bucket’s designated Web Server the particular permissions needed to add objects to it and call the CreateThumbnail Lambda function.

5) *Auto Scaling Group (ASG):* To create ASG, we need to create a Dev Server instance first, which is used to develop the custom AMI for the web server and make a launch template later.

Dev Server Instance: Dev server is not receiving traffic from ELB, as it serves as the platform solely for developing the custom AMI required to run the PhotoAlbum website. The custom AMI encompasses all the necessary components such as the AWS PHP SDK, Apache web server, and website source code. Additionally, the Dev server can manage MySQL RDS instance using phpMyAdmin.

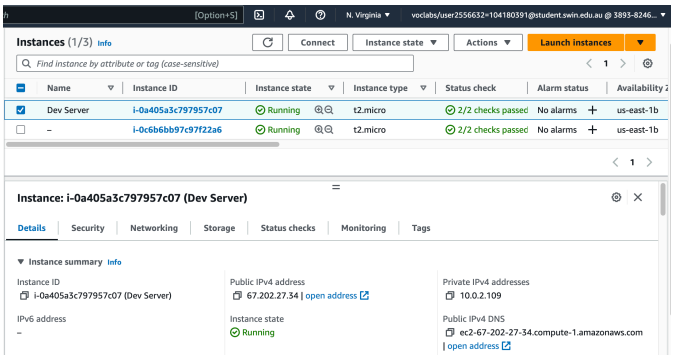


Figure 18: Dev Server instance resides in Public Subnet 2 (CIDR: 10.0.2.0/24) with an Elastic IP associated

Dev Server is configured with t2-micro as instance type using Amazon Linux 2 AMI (HVM), SSD Volume Type, with Apache Web Server installed using bash script in assignment 1a. IAM Role is LabRole, which is already created in AWS Learner Lab.

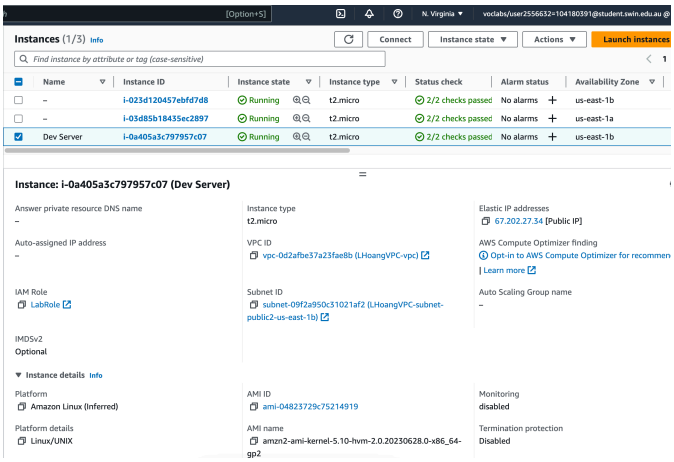


Figure 19: Configuration of Dev Server EC2 instance

The Dev Server is associated with an Elasitc IP to allow SSH connection to manage Dev Server directory.

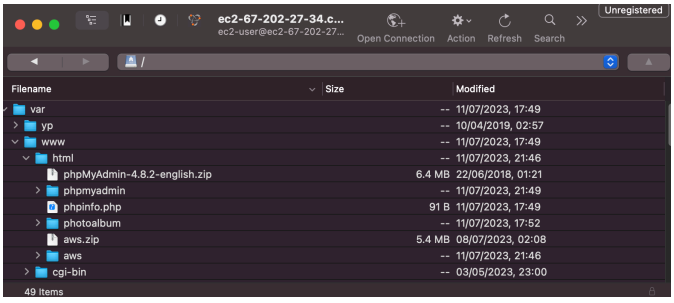


Figure 20: Dev Server Directory Structure with necessary SDKs and components included

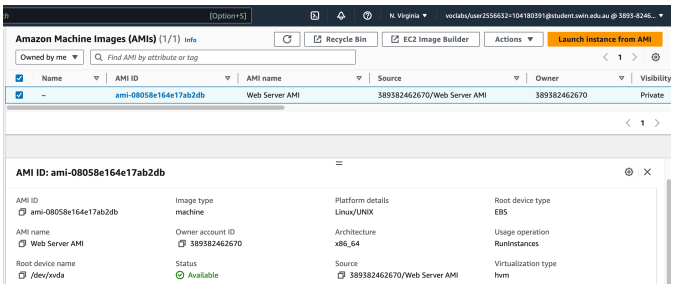


Figure 21: Dev Server image created, ready to use

Launch template: Use the created image from Figure 21 to create new launch template with instance type *t2.micro* and IAM role *LabInstanceProfile* attached.

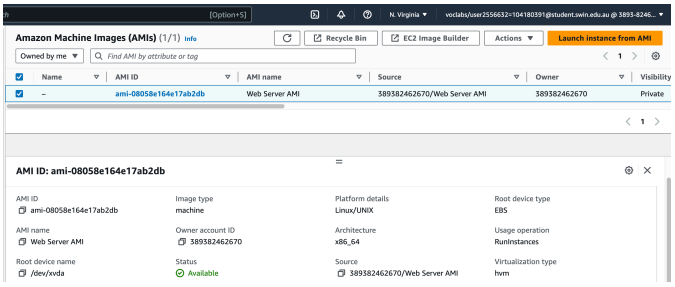


Figure 22: Launch template for ASG

Auto Scaling Group: The ASG is specifically configured to launch instances into the private subnets, maintaining a minimum of 2 instaces and a maximum of 3 instances, with 2 instances as the desired number. This ensures that the application has a minimum number of instances available at all times while also preventing the infrastructure from scaling beyond the defined maximum.

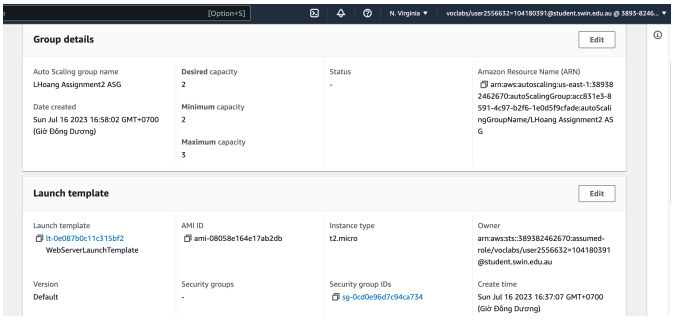


Figure 23: ASG basic configuration

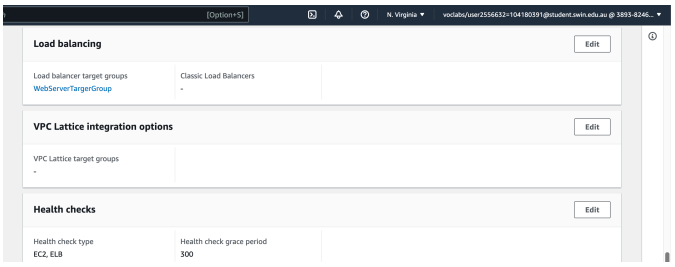


Figure 24: ASG health check configuration with target group attached

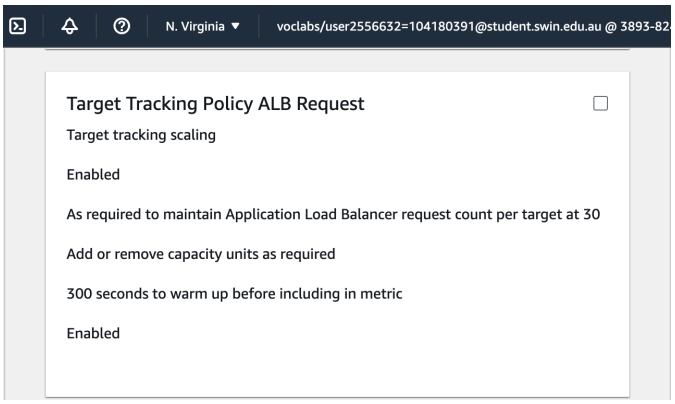


Figure 25: Target tracking policy based on application load balancer request

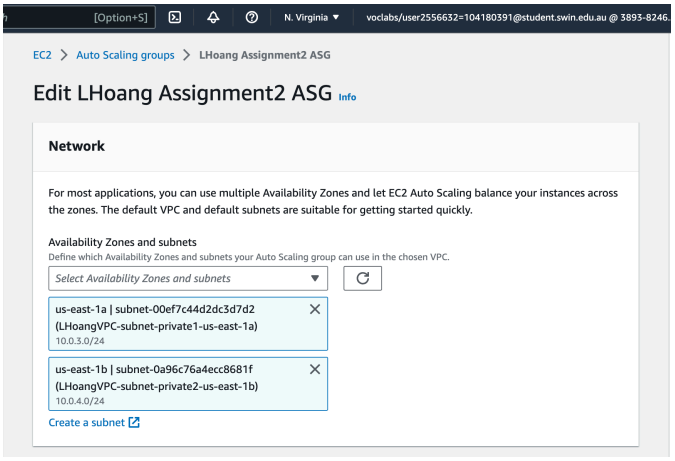


Figure 26: ASG network mapped to private subnets

To control the number of instances based on the number of requests received for each target in the ELB target group, a target tracking scaling policy was created (Figure 25). The policy is configured to maintain a target request count of 30. In order to maintain the desired request count per target, the auto-scaling group will automatically scale the number of instances up or down, ensuring optimal performance and effective resource utilization.

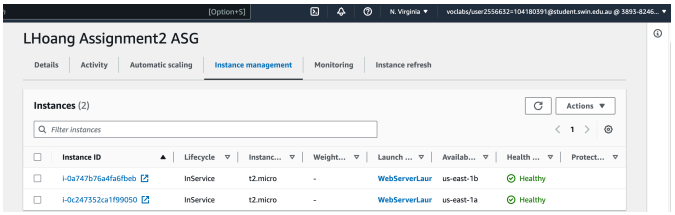


Figure 27: EC2 instances properly distributed accross two private subnets with healthy states

Now the web server can dynamically adapt its capacity based on the request load, keeping a constant number of requests per target and maximizing resource utilization.

6) *Elastic Load Balancing (ELB):* Firstly, create a new target group as load balancer need to routes requests to the targets in a target group and performs health checks on the targets.

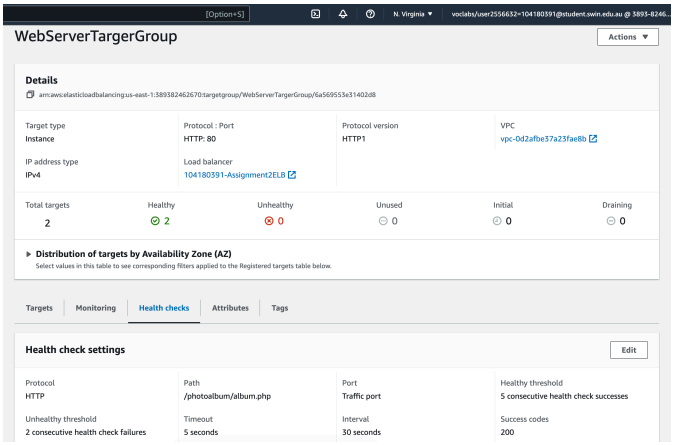


Figure 28: Target group configuration, with health check path set to `/photoalbum/album.php`

Create a new load balancer and attach it to the target group.

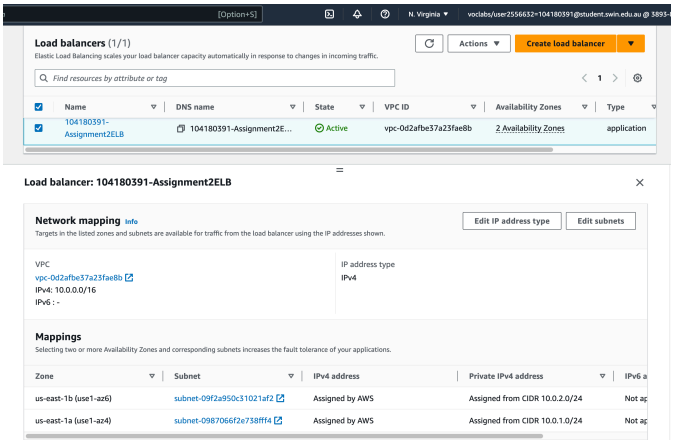


Figure 29: Application Load Balancer mapped to Public Subnet 1 and Public Subnet 2

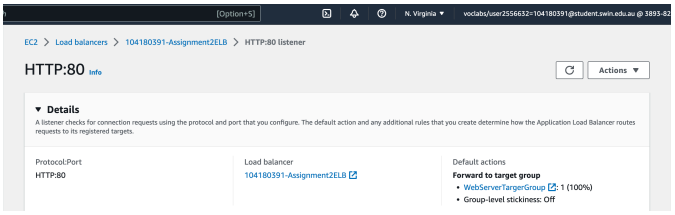


Figure 30: ELB Listener check rule which forward to targer group created in Figure 28

Now The ELB can distribute incoming HTTP and HTTPS traffic across multiple EC2 targets.

7) *Simple Storage Service (S3):* The S3 bucket has been created with almost the same configuration as the one used in assignment 1b, specifically for storing photos. To ensure proper accessibility of objects stored in this S3 bucket, appropriate permissions and policies have been applied. These measures guarantee that the necessary permissions are set

up correctly to allow access to the stored photos as intended [2].

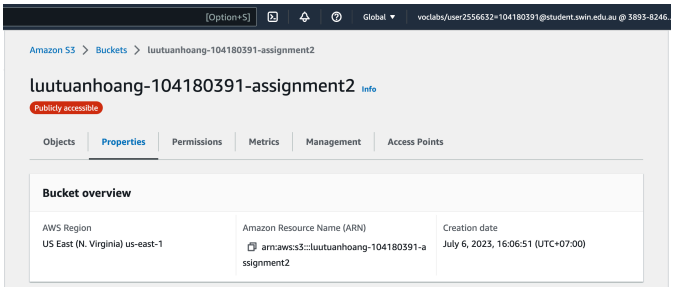


Figure 31: Properties of S3 bucket

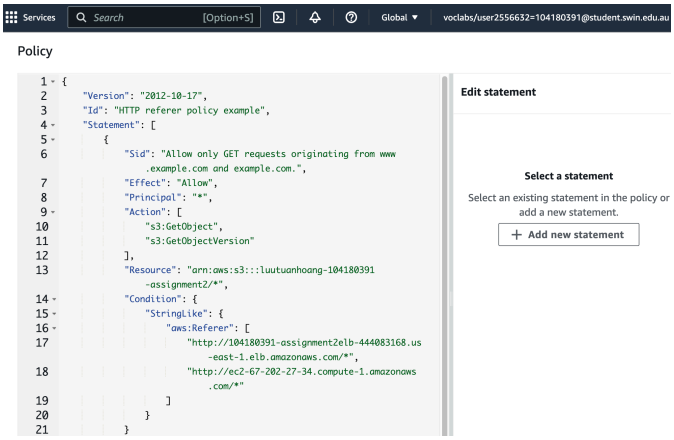


Figure 32: S3 policy to restrict access to a specific HTTP referer from Dev Server and Elastic Load Balancer

In summary, this policy restricts access to the S3 bucket to only those GET requests originating from the specified domains, ensuring a controlled and secure access policy for the bucket's objects.

8) *Lambda Fuction:* A Lambda function named "CreateThumbnail" was created using Python 3.7 as the runtime environment.

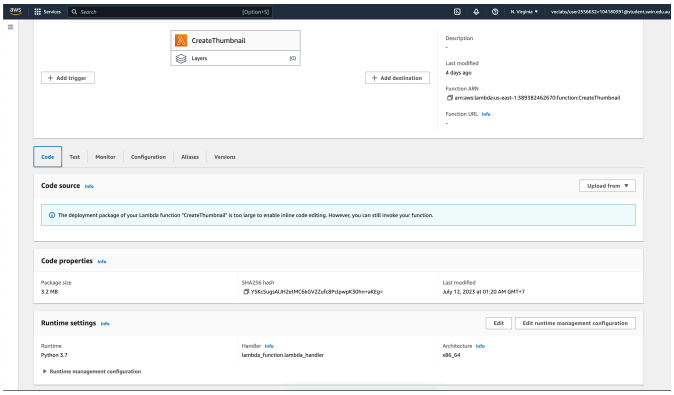


Figure 33: *CreateThumbnail* Lambda function configuration

The "lambda-deployment-package.zip" deployment package was uploaded. The required library and complete source code for resizing images and processing downloads and uploads using the S3 bucket are both included in this package.

9) *Relational Database Service (RDS):* The RDS instace use in this assignment is configured the same as the previous assignment.

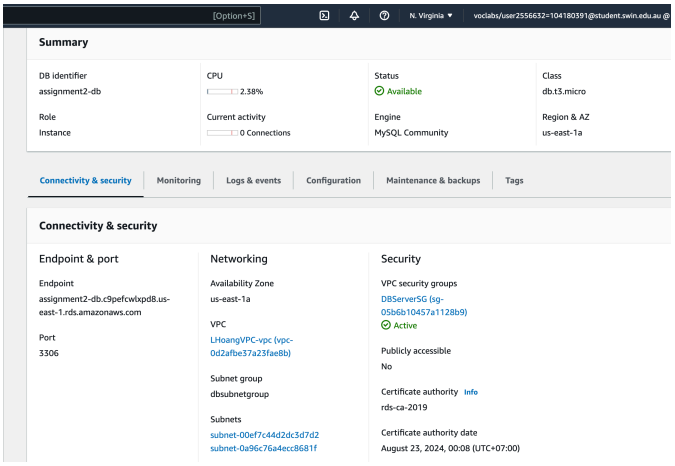


Figure 34: RDS instance

Configuration:

– Template: Free-tier

– Database engine: MySQL Community 8.0.28

– Public access set to No.

– Use DBServerSG for VPC security group

– AZ set to us-east-1a (According to the provided diagram).

– The RDS instance is associated with a subnet group *dbsubnetgroup* which consists of private subnets in both AZs.

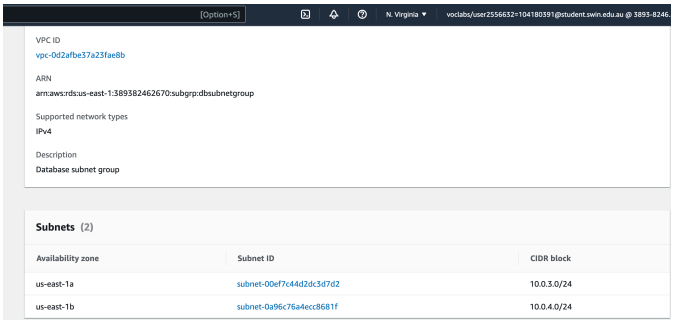


Figure 35: Subnet *dbsubnetgroup* with Private subnet 3 and Private subnet 4

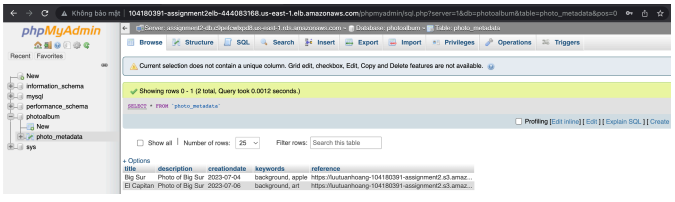


Figure 36: Data records of the database

B. *Functional requirements*

1) *Website accessibility:*

To access the PhotoAlbum website, use the following URL: http://104180391-assignment2elb-444083168.us-east-1.elb.amazonaws.com/photoalbum/album.php. It allows you to view and interact with the PhotoAlbum web application. Additionally, to upload photos and their associated metadata, utilize the PhotoUploader web page at http://104180391-assignment2elb-444083168.us-east-1.elb.amazonaws.com/photoalbum/photouploader.php. By using this page, multiple photos can be easily upload and input their corresponding metadata to enhance the functionality of the PhotoAlbum website.

Student name: Luu Tuan Hoang

Student ID: 104180391

Tutorial session: Saturday 08:00AM

Uploaded photos:

Upload more photos

Figure 37: Website accessible through ELB DNS

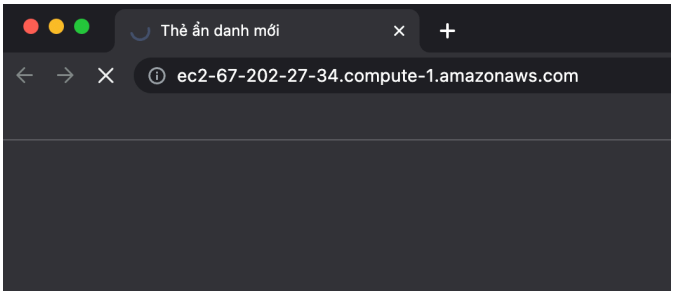


Figure 38: Website is not accessible through Dev Server ec2-67-202-27-34.compute-1.amazonaws.com/photoalbum/album.php, as security group has blocked HTTP

Therefore, the website is only accessible through ELB.

2) *Photo display function:*

Student name: Luu Tuan Hoang

Student ID: 104180391

Tutorial session: Saturday 08:00AM

Uploaded photos:

Upload more photos

Photo	Name	Description	Creation date	Keywords
	Big Sur	Photo of Big Sur	2023-07-04	background, apple

Figure 39: Photo display function

Photo display function is working properly

3) *Photo upload function:*

Photo uploader

Photo title: El Capitan

Select a photo (Select PNG file for best result):

Chọn tệp ElCapitan.jpeg

Description: Photo of Big Sur

Date: 06/07/2023

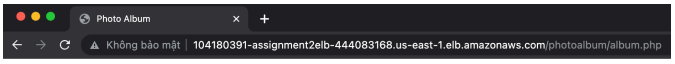
Keywords (comma-delimited, e.g. keyword1, keyword2, ...):

background, art

Upload

Photo Album

Figure 40: Uploading photo



Student name: Luu Tuan Hoang

Student ID: 104180391

Tutorial session: Saturday 08:00AM

Uploaded photos:

Upload more photos

Photo	Name	Description	Creation date	Keywords
	Big Sur	Photo of Big Sur	2023-07-04	background, apple
	El Capitan	Photo of Big Sur	2023-07-06	background, art

Figure 41: Photo uploaded

Photo uploading function is working properly.

4) *Resizing Lambda:*

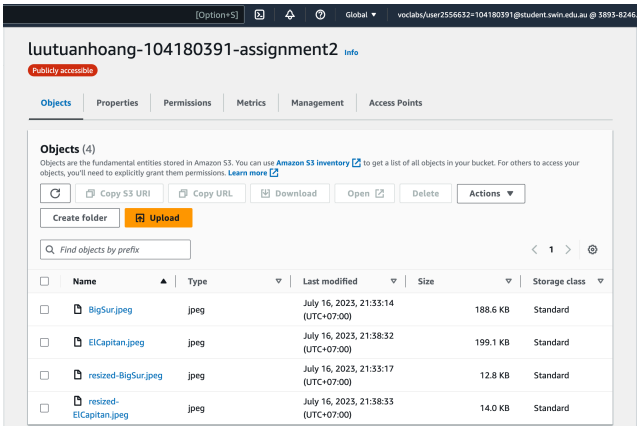


Figure 42: Photo uploaded

REFERENCES

[1] M. Pope, “Securely connect to linux instances running in a private amazon vpc,” Amazon Web Services, 2014. [Online]. Available: https://aws.amazon.com/blogs/security/securely-connect-to-linux-instances-running-in-a-private-amazon-vpc/

[2] Amazon Web Service, “Bucket policy examples - amazon simple storage service,” docs.aws.amazon.com, 2023. Accessed: Jul. 16, 2023. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html#example-bucket-policies-HTTP-HTTPS