

Developing a highly available Photo Album website

Trac Duc Anh Luong
Lab Section: Saturday 10:00 AM
Swinburne University of Technology
103488117@student.swin.edu.au

Abstract— This assignment serves as an extended iteration of Assignment 1b, expanding upon creating and deploying the Photo Album website on a streamlined AWS infrastructure. This task focuses on the interaction among AWS services, emphasizing VPC, EC2, RDS, Lambda, and S3. Following the creation and configuration of these services, the objective is to deploy a Photo Album website that ensures high availability, scalability, and optimal performance.

I. INTRODUCTION

The Photo Album website acts as a platform accessible via a public URL, facilitating users' photo uploads along with comprehensive meta-data such as the photo title, description, creation date, and keywords. After uploading, users can view their photos in a gallery with associated information. The website will be hosted on EC2 web servers linked to an RDS-hosted MySQL database. Photo storage will take place in an S3 bucket reinforced by enhanced policies. Upon upload, a Lambda function will resize the photo to create a smaller thumbnail, which will also be stored in S3. Upon completion, the Photo Album project will offer functionalities encompassing photo upload, display, and thumbnail creation.

II. INFRASTRUCTURE REQUIREMENTS

A. Virtual Private Cloud (VPC)

The VPC has 2 availability zones, each with a private and public subnet with suitable CIDR ranges. The public subnets are associated with a route table that directs traffic to an internet gateway (IGW). The private subnets are associated with a private route table that directs traffic to a NAT gateway (public subnet 1). I implemented a NAT gateway instead of a NAT instance for this assignment.

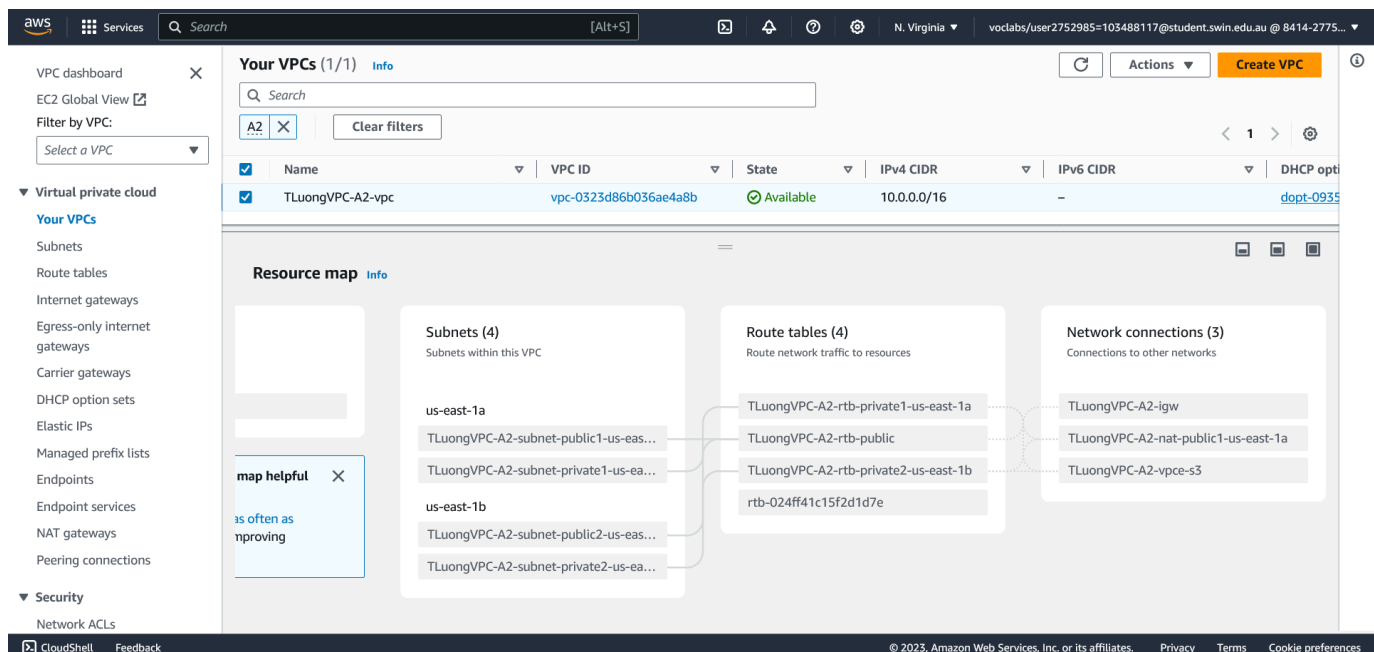


Figure 1: VPC Resource Map with NAT Gateway

The screenshot shows the AWS VPC Subnets console. The left sidebar lists navigation options: VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud, Your VPCs, Subnets, Route tables, and Internet gateways. The main content area is titled 'Subnets (4) Info'. It includes a search bar and a filter dropdown set to 'VPC: vpc-0323d86b036ae4a8b'. Below this is a table of subnets.

<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR
<input type="checkbox"/>	TLuongVPC-A2-subnet-public1-us-east-...	subnet-022670880cc6ed4ea	Available	vpc-0323d86b036ae4a8b TLu...	10.0.1.0/24	-
<input type="checkbox"/>	TLuongVPC-A2-subnet-public2-us-east-...	subnet-0c4678cf329f8310a	Available	vpc-0323d86b036ae4a8b TLu...	10.0.2.0/24	-
<input type="checkbox"/>	TLuongVPC-A2-subnet-private1-us-east-...	subnet-052118f8bec1fcfab	Available	vpc-0323d86b036ae4a8b TLu...	10.0.3.0/24	-
<input type="checkbox"/>	TLuongVPC-A2-subnet-private2-us-east-...	subnet-0ee044280919bab5b	Available	vpc-0323d86b036ae4a8b TLu...	10.0.4.0/24	-

Figure 2: VPC Subnet CIDR blocks

B. Security Group (SG)

Four security groups are DevServerSG, WebServerSG, DBServerSG, and ELBSG, each associated with a tier shown in the architecture diagram. Since I implemented a NAT gateway, the NATServerSG is not needed. The screenshots below show the inbound rules of each SG, while the outbound rules of all SGs are set to All traffic from Anywhere-IPv4.

The screenshot shows the AWS Security Groups console. The left sidebar lists navigation options: VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud, Your VPCs, Subnets, Route tables, and Internet gateways. The main content area is titled 'Security Groups (4) Info'. It includes a search bar and a filter dropdown set to 'VPC ID: vpc-0323d86b036ae4a8b'. Below this is a table of security groups.

Security group name	VPC ID	Description	Owner	Inbound rules count	Outbound rules count
DevServerSG-A2	vpc-0323d86b036ae4a8b	for the Dev server	841427755345	1 Permission entry	1 Permission entry
WebServerSG-A2	vpc-0323d86b036ae4a8b	for all the web servers in private subnets	841427755345	4 Permission entries	1 Permission entry
DBServerSG-A2	vpc-0323d86b036ae4a8b	for the RDS instance	841427755345	2 Permission entries	1 Permission entry
ELBSG-A2	vpc-0323d86b036ae4a8b	for the ELB created	841427755345	2 Permission entries	1 Permission entry

Figure 3: All Security Groups

The screenshot shows the AWS Security Groups console with the 'DevServerSG-A2' security group selected. The left sidebar lists navigation options: VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud, Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections, Security, and Network ACLs. The main content area is titled 'Security Groups (1/4) Info'. It includes a search bar and a filter dropdown set to 'VPC ID: vpc-0323d86b036ae4a8b'. Below this is a table of security groups. The 'Inbound rules' tab is selected, showing a table of inbound rules.

<input checked="" type="checkbox"/>	Name	Security group ID	Security group name	VPC ID	Description	Owner
<input checked="" type="checkbox"/>	-	sg-02d5b86ecef68d77f	DevServerSG-A2	vpc-0323d86b036ae4a8b	for the Dev server	841427755345
<input type="checkbox"/>	-	sg-0330ce6fa05cb4725	WebServerSG-A2	vpc-0323d86b036ae4a8b	for all the web servers in private subnets	841427755345
<input type="checkbox"/>	-	sg-07ba9409e16c27cde	DBServerSG-A2	vpc-0323d86b036ae4a8b	for the RDS instance	841427755345
<input type="checkbox"/>	-	sg-00a6c92f17b0e1961	ELBSG-A2	vpc-0323d86b036ae4a8b	for the ELB created	841427755345

The 'Inbound rules (1/1)' table is shown below:

IP version	Type	Protocol	Port range	Source	Description
IPv4	All traffic	All	All	0.0.0.0/0	-

Figure 4: DevServerSG Security Group

DevServerSG does not have to follow the least-privilege principle. Therefore, we will leave the inbound rule of this SG to All traffic and source to Anywhere-IPv4. With this inbound rule, we can access the instance using SSH to configure phpMyAdmin and access the phpMyAdmin Panel through the public DNS and create our database and the photo meta-data table.

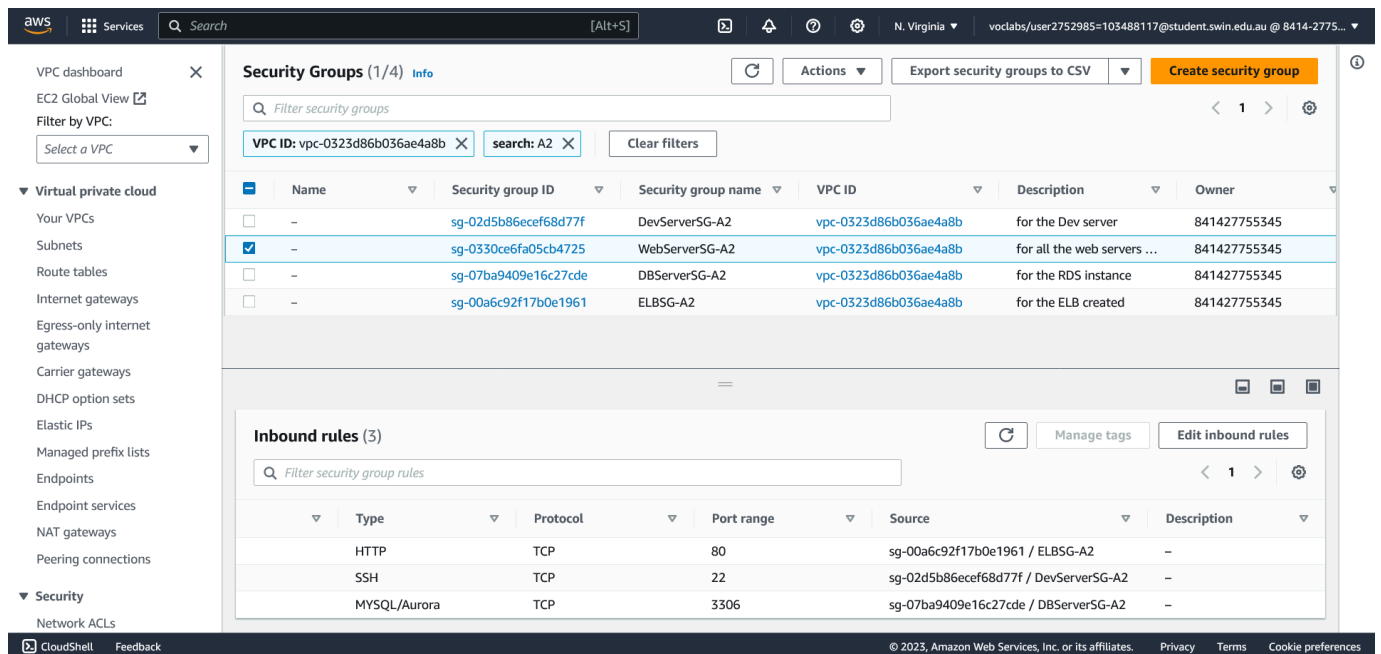


Figure 5: WebServerSG Security Group

The WebServerSG has 3 protocols, as can be seen in Figure 5.

- HTTP from the ELB to access the Photo Album website.
- SSH from the DevServerSG to test the NACL bidirectional functionality by sending ICMP traffic (ping) between these groups.
- MYSQL/Aurora to read data from the RDS.

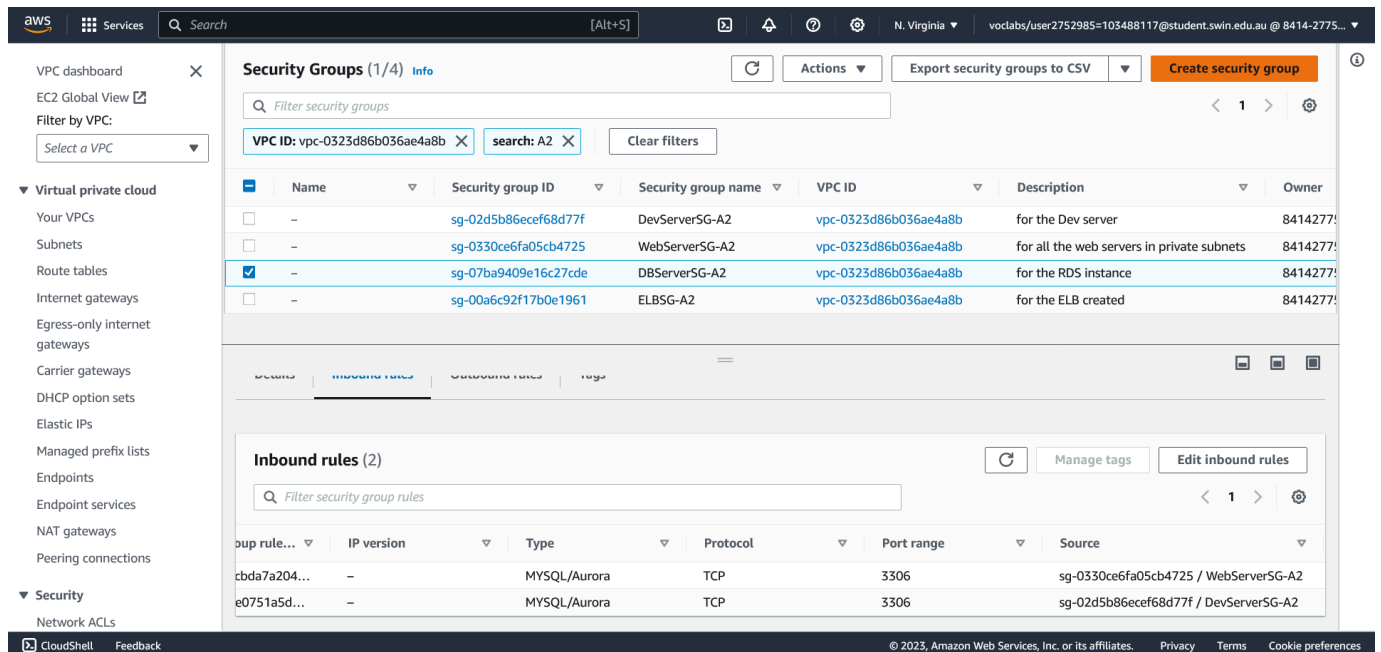


Figure 6: DBServerSG Security Group

DBServerSG will open MYSQL/Aurora traffic from DevServerSG and WebServerSG as we write data to the database from both Dev and Web instances.

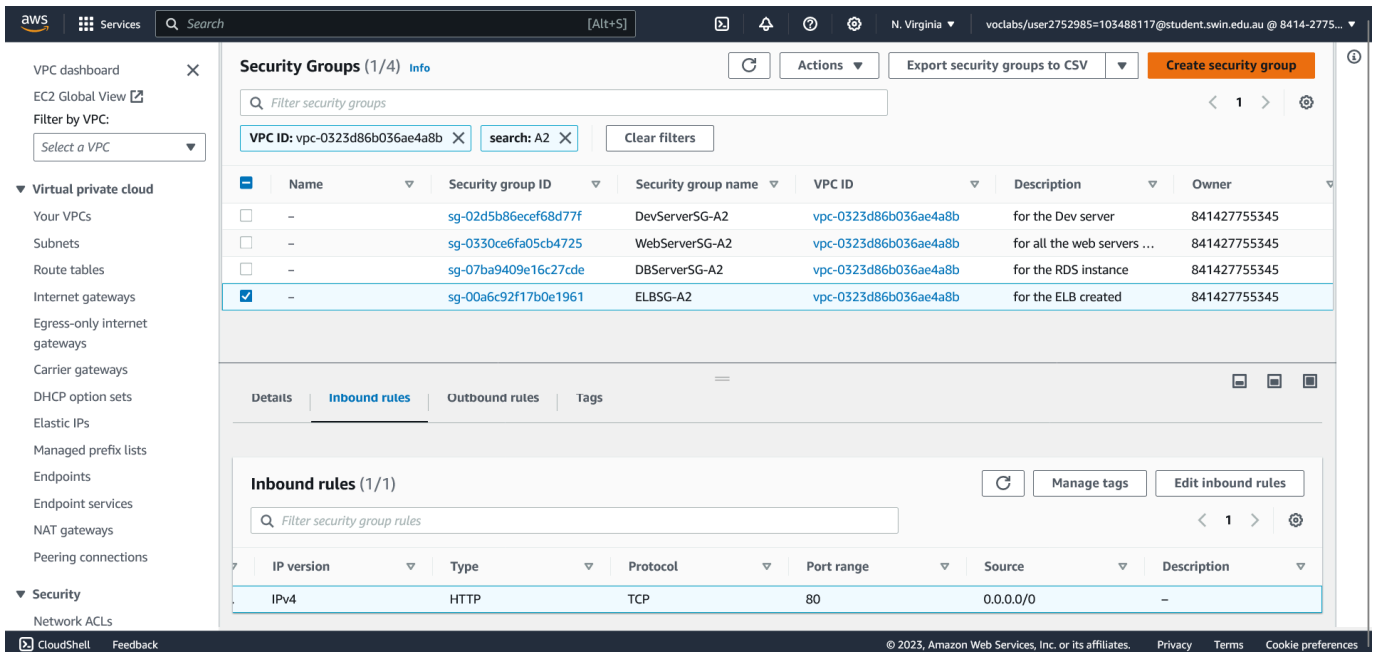


Figure 7: ELBSG Security Group

The ELBSG will have HTTP traffic from Anywhere-IPv4 so that the website can be accessible anywhere from the internet.

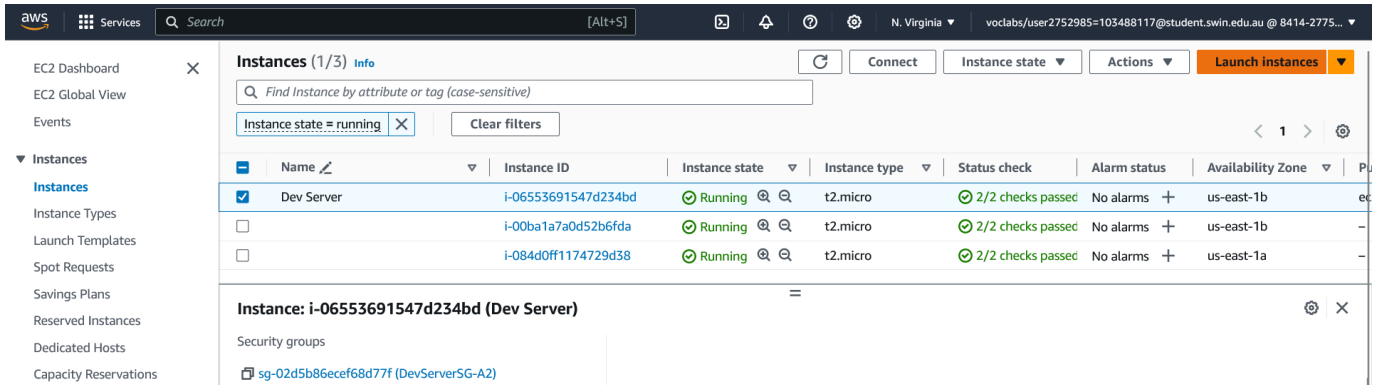


Figure 8: DevServerSG Security Group attached to Dev Server EC2 instance

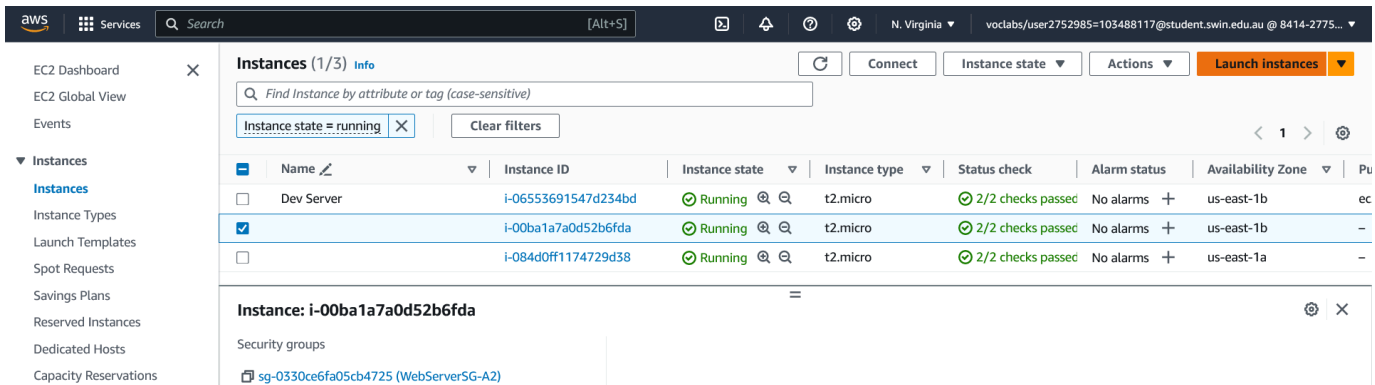


Figure 9: WebServerSG Security Group attached to Web Server EC2 launch instances

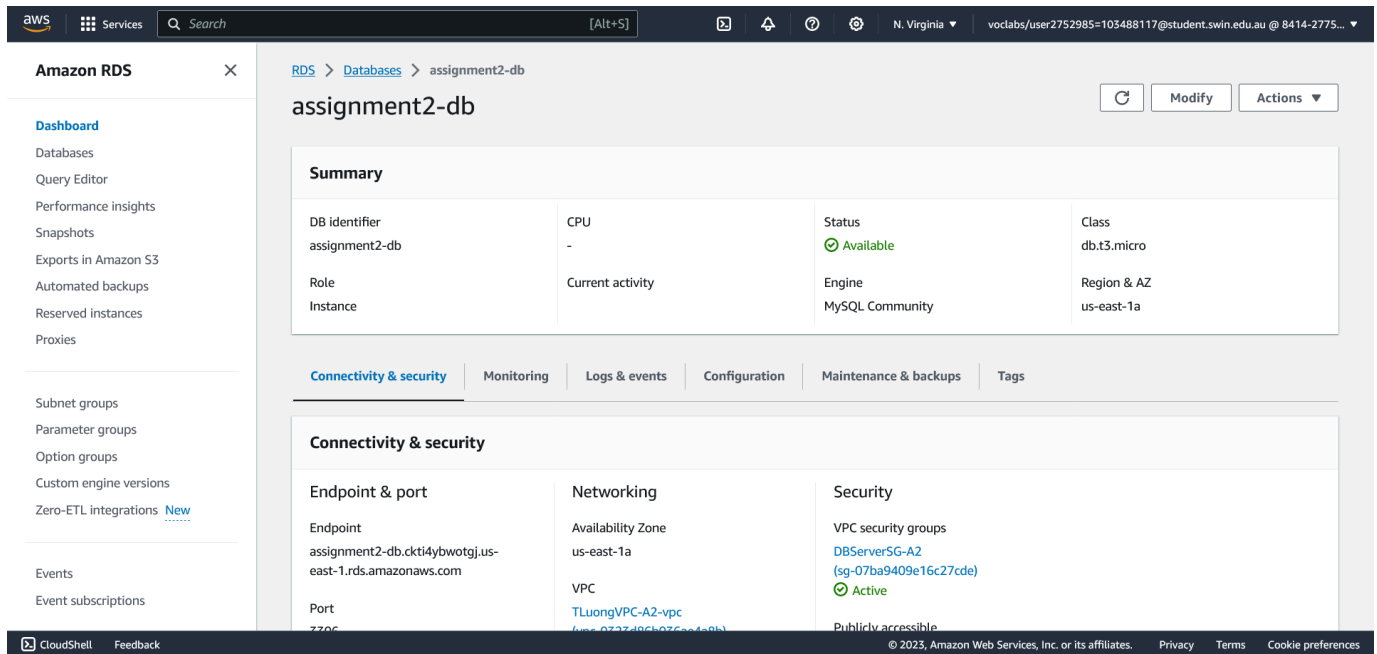


Figure 10: DBServerSG Security Group attached to RDS instance

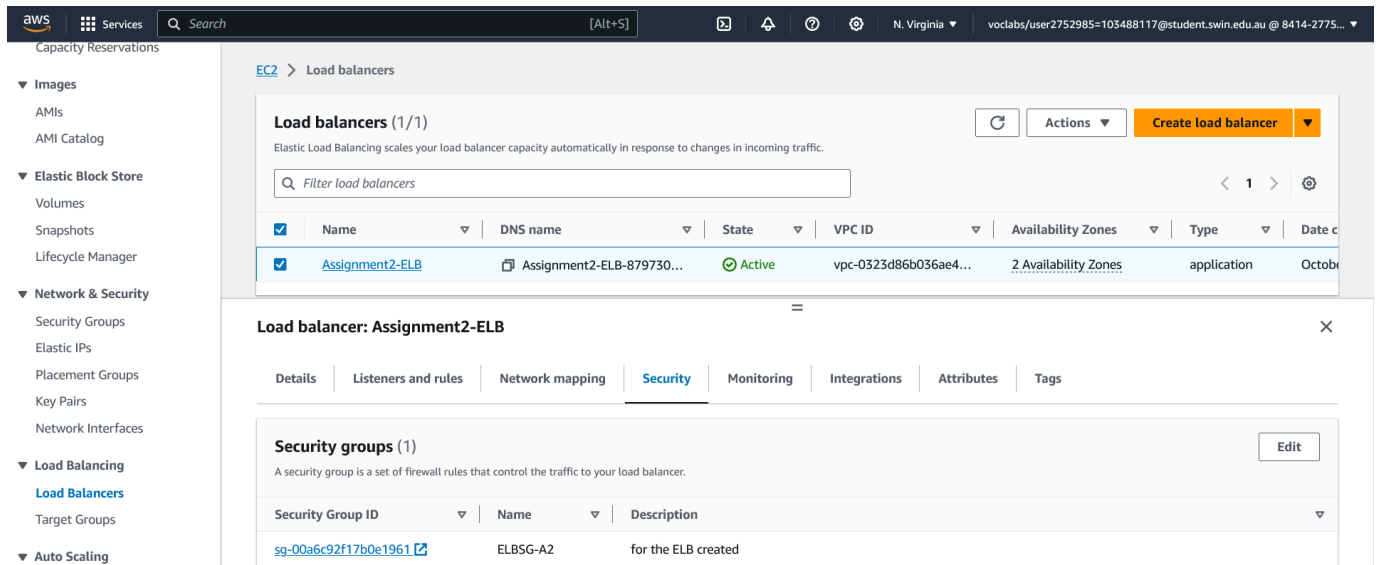


Figure 11: ELBSG Security Group attached to Elastic Load Balancer

Security Group Name	Type	Port range	Source
DevServerSG	All	All	Anywhere-IPv4
WebServerSG	HTTP	80	ELBSG
	SSH	22	DevServerSG
	MYSQL/Aurora	3306	DBServerSG
DBServerSG	MYSQL/Aurora	3306	DevServerSG
	MYSQL/Aurora	3306	WebServerSG
ELBSG	HTTP	80	Anywhere-IPv4

Figure 12: Security Groups' inbound rules summary

Now the SGs have been correctly attached and followed the least-privilege principle.

C. Network ACL (NACL)

To add a layer of security to the Web Servers, I have created a Network ACL named *Assignment2-NACL*.

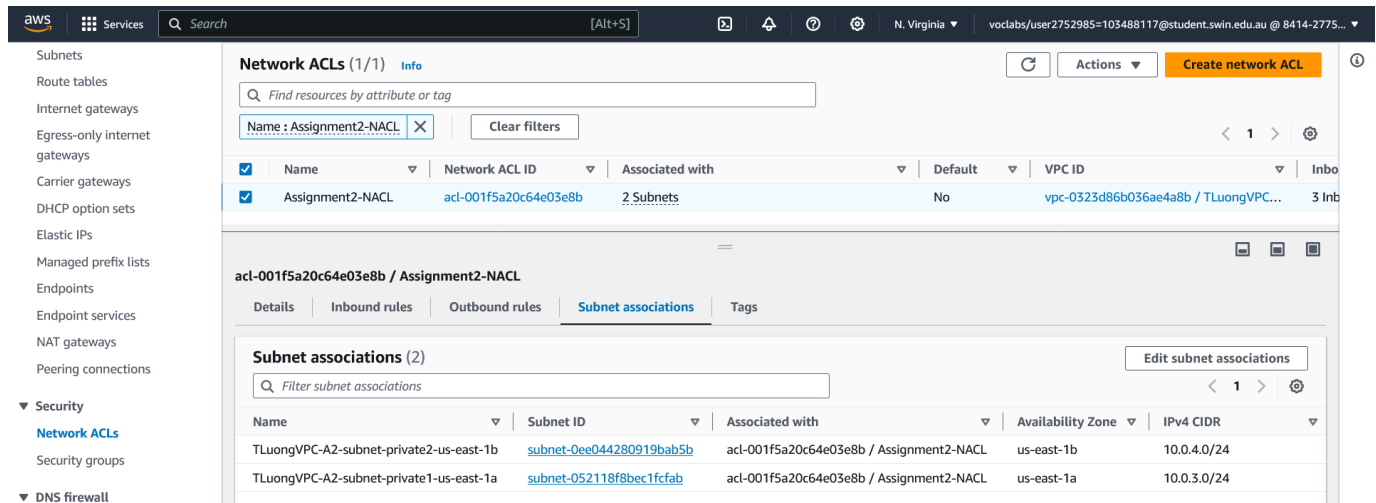


Figure 13: NACL subnet association with 2 private subnets, 10.0.3.0/24 and 10.0.4.0/24

Following the assignment's architecture diagram, the NACL subnet association is placed in private subnets 10.0.3.0/24 and 10.0.4.0/24.

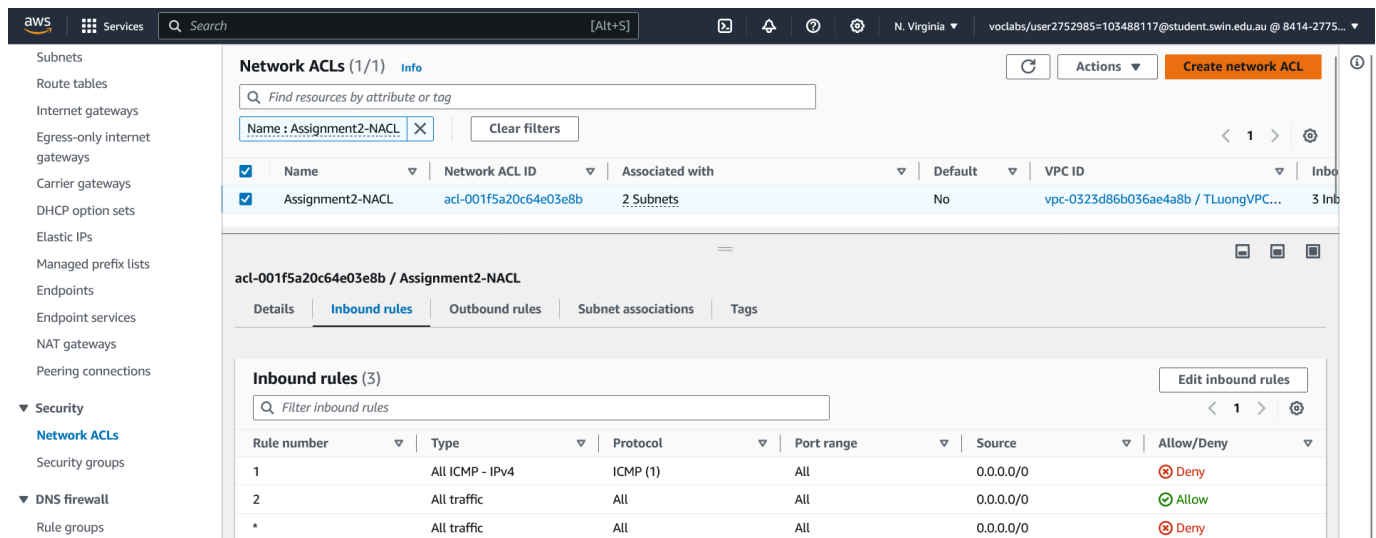


Figure 14: NACL inbound rules

Both inbound and outbound rules of the NACL will block All ICMP - IPv4 from Anywhere, making the Dev Server that resides in Public Subnet 2 (10.0.2.0/24) unable to send any ICMP traffic to our Web Servers.

A screenshot of a terminal window titled "ec2-user@ip-10-0-4-206:~". The terminal shows several commands being executed:

1. A login attempt as "ec2-user" which authenticates successfully using a public key named "imported-openssh-key".
2. The user runs a command that displays system information for Amazon Linux 2, including the end of life date (2025-06-30) and a notice about a newer version being available.
3. The user pings the IP address 10.0.4.206, receiving a response from 10.0.4.206 with 56(84) bytes of data.
4. The user attempts to ping 10.0.4.206 again, but receives 0 packets received, indicating 100% packet loss.
5. The user runs an SSH command to connect to ip-10-0-2-129.ec2.internal using a private key LTDA-linux.pem.
6. The terminal shows the output of the SSH connection, including the last login time and the same system information as before.
7. The user pings the IP address 10.0.2.129, receiving a response from 10.0.2.129 with 56(84) bytes of data.
8. The user attempts to ping 10.0.2.129 again, but receives 0 packets received, indicating 100% packet loss.
The terminal window has standard Ubuntu window controls at the top right.

We first logged in to Dev Server (local IP: 10.0.2.129) through SSH using PuTTY. Second, we tried to ping a Web Server (local IP: 10.0.4.206). Third, we logged in to the Web Server from the Dev Server and tried to ping the Dev Server. As shown in Figure 16, our attempts did not transmit any packets. Therefore, our NACL has been configured correctly.

D. IAM

For this assignment, the management console already has an IAM role named “LabRole” or “Labinstacerole” with the required permissions. We will assign this role to 3 components:

- The Lambda function
- The Dev Server instance
- The Launch template

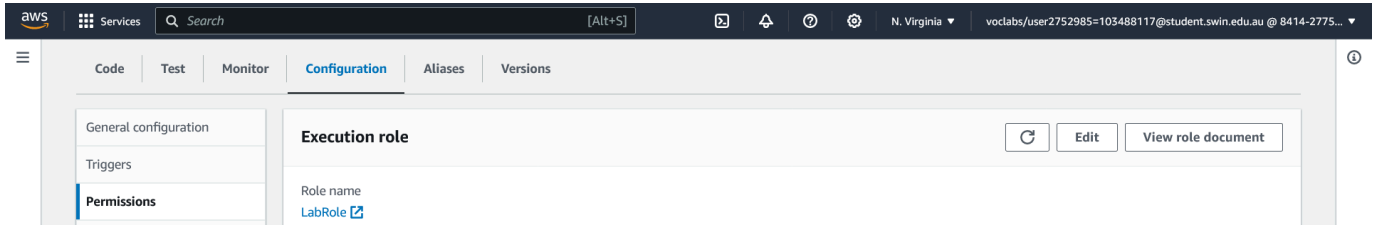


Figure 17: LabRole assigned to Lambda function

The execution role of the Lambda function is set to LabRole, allowing it to gain access control over the objects in our S3 bucket that was already specified, following the least-privilege principle. The Lambda function will get the photo uploaded to S3, create a smaller resized thumbnail and upload that thumbnail back to the bucket.

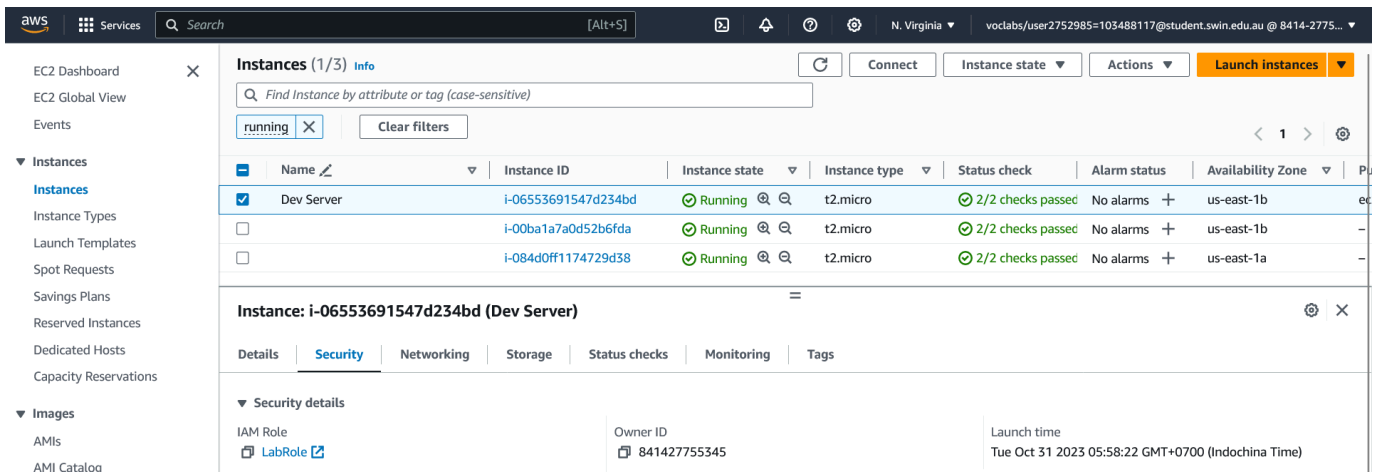


Figure 18: LabRole assigned to Dev Server instance

The Dev Server instance is also granted the IAM Role LabRole, which calls the Lambda function and uploads the photo to the S3 bucket.

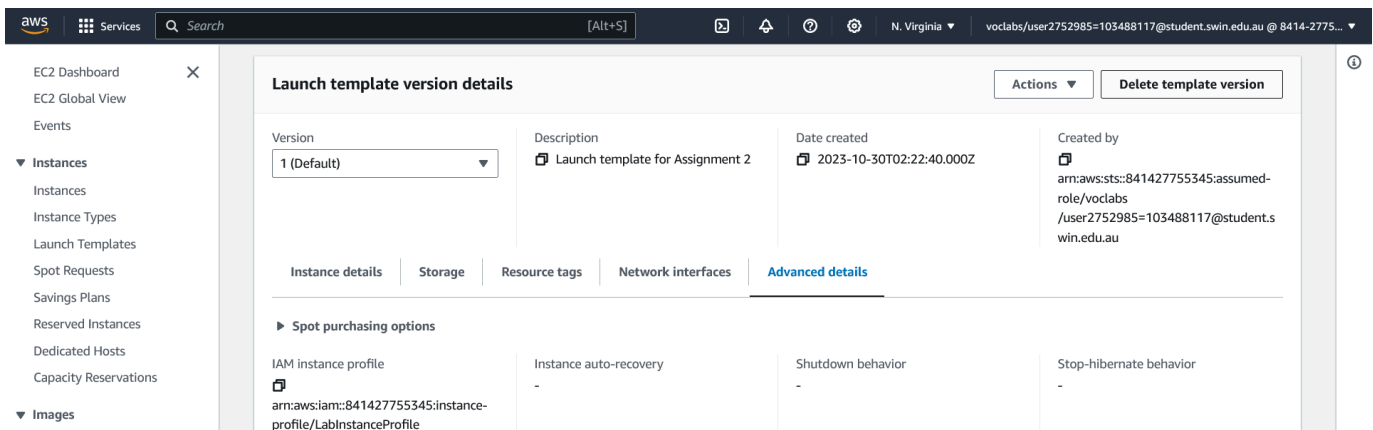


Figure 19: LabRole assigned to Launch template

Name: Trac Duc Anh Luong - ID: 103488117

The Web Server instances also need the same permissions as the Dev Server, from photo uploading, retrieving, and calling the Lambda function. The LabRole is also assigned to the Launch template, which will later be used in our Auto Scaling Group.

E. Auto Scaling Group (ASG)

To create a maintainable and scalable architecture, we will need to create an ASG and configure it as per the requirements of this assignment. Before creating the ASG, we need a fully functional Dev Server.

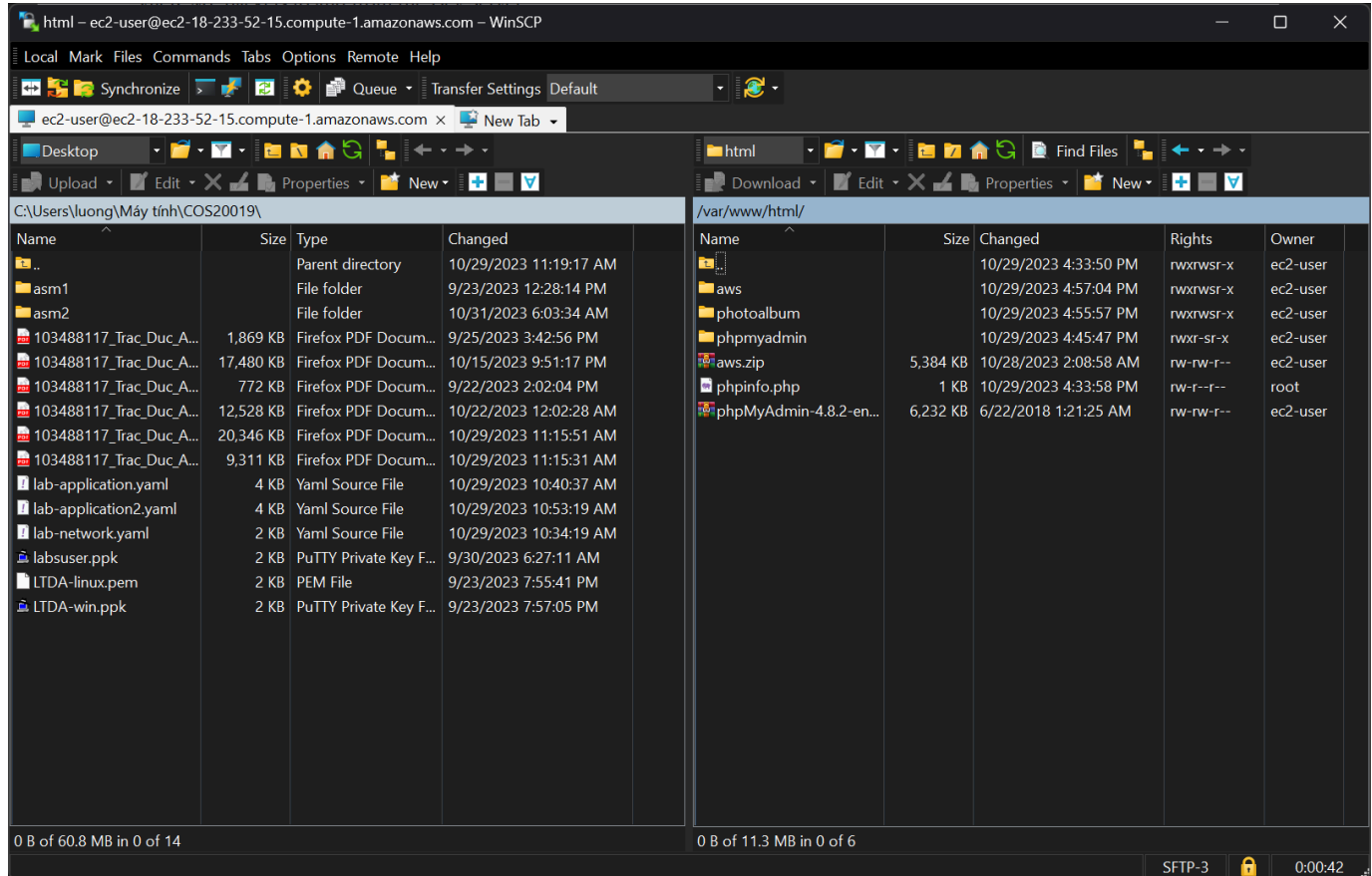
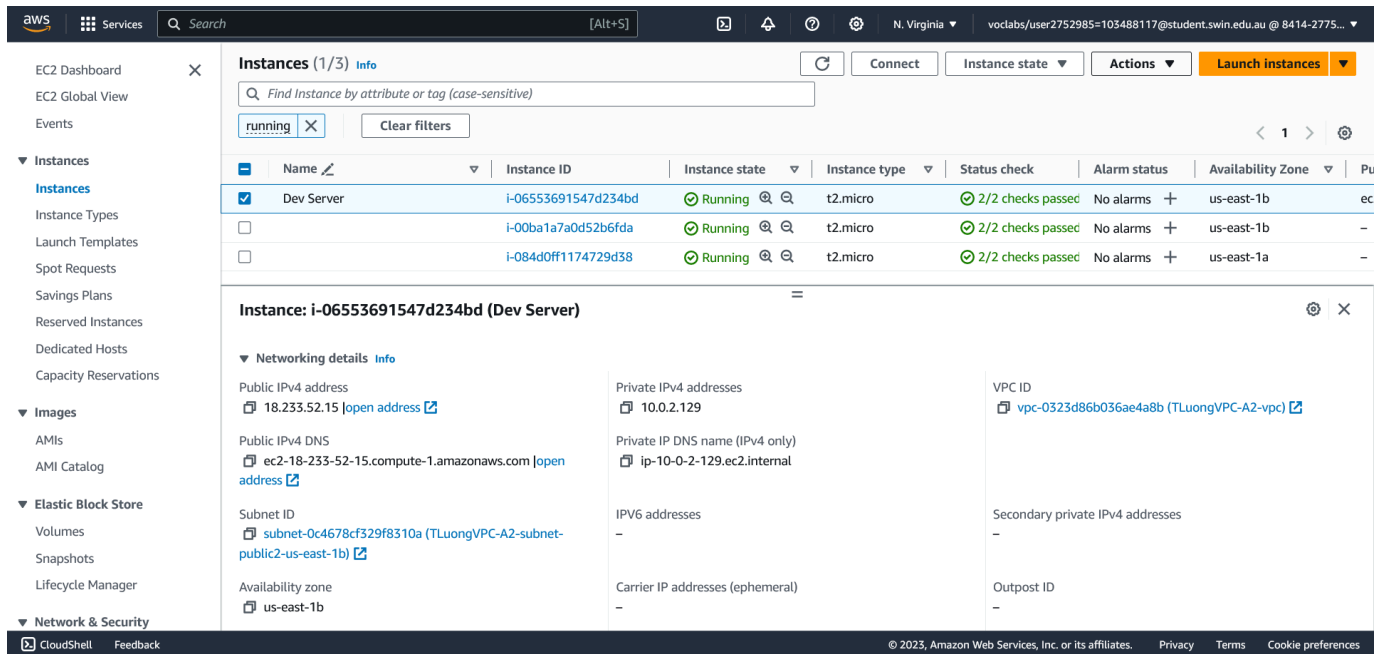


Figure 20: Dev Server directory structure with AWS SDK, Photo Album, and phpMyAdmin

Upon creation, the Dev Server will have Apache and PHP installed using the EC2 setup script from the previous assignment. After that, we will install phpMyAdmin, and change the host from *localhost* to the *RDS endpoint* in the *config.inc.php* file. We must also install the AWS SDK as instructed in the *constants.php* file. The Photo Album source code will be uploaded to the Dev Server using WinSCP (or FileZilla) through an SSH connection.



Instances (1/3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Dev Server	i-06553691547d234bd	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b
	i-00ba1a7a0d52b6fda	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b
	i-084d0ff1174729d38	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a

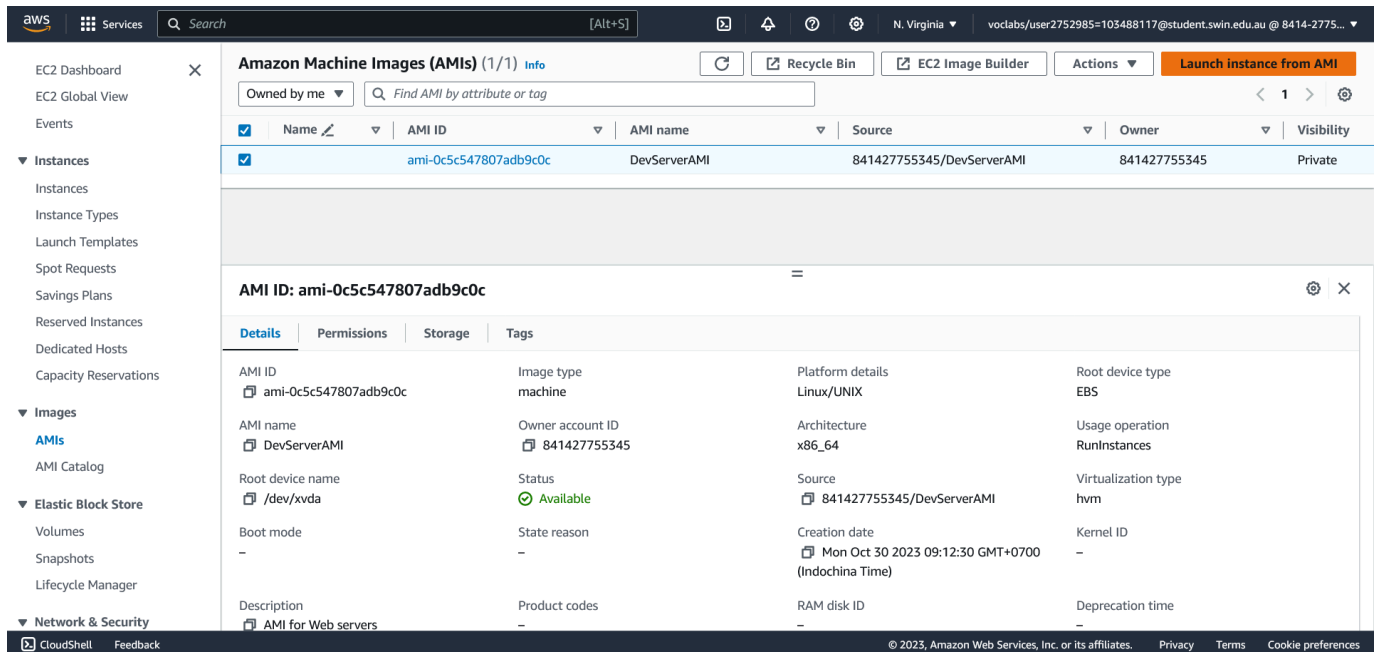
Instance: i-06553691547d234bd (Dev Server)

Networking details Info

Public IPv4 address	Private IPv4 addresses	VPC ID
18.233.52.15 open address	10.0.2.129	vpc-0323d86b036ae4a8b (TLuongVPC-A2-vpc) open
Public IPv4 DNS	Private IP DNS name (IPv4 only)	
ec2-18-233-52-15.compute-1.amazonaws.com open address	ip-10-0-2-129.ec2.internal	
Subnet ID	IPv6 addresses	Secondary private IPv4 addresses
subnet-0c4678cf329f8310a (TLuongVPC-A2-subnet-public2-us-east-1b) open	-	-
Availability zone	Carrier IP addresses (ephemeral)	Outpost ID
us-east-1b	-	-

Figure 21: Dev Server located in Public Subnet 2 (10.0.2.0/24) with an Elastic IP address associated

From the architecture diagram, the Dev Server instance must be placed in Public Subnet 2 (10.0.2.0/24), with an Elastic IP Address, to access the instance from our local computer.



Amazon Machine Images (AMIs) (1/1) Info

Name	AMI ID	AMI name	Source	Owner	Visibility
DevServerAMI	ami-0c5c547807adb9c0c	DevServerAMI	841427755345/DevServerAMI	841427755345	Private

AMI ID: ami-0c5c547807adb9c0c

Details	Permissions	Storage	Tags
AMI ID ami-0c5c547807adb9c0c	Image type machine	Platform details Linux/UNIX	Root device type EBS
AMI name DevServerAMI	Owner account ID 841427755345	Architecture x86_64	Usage operation RunInstances
Root device name /dev/xvda	Status Available	Source 841427755345/DevServerAMI	Virtualization type hvm
Boot mode -	State reason -	Creation date Mon Oct 30 2023 09:12:30 GMT+0700 (Indochina Time)	Kernel ID -
Description AMI for Web servers	Product codes -	RAM disk ID -	Deprecation time -

Figure 22: Dev Server AMI was created for the launch template

We will create an AMI image from the Dev Server to configure our Launch template next.

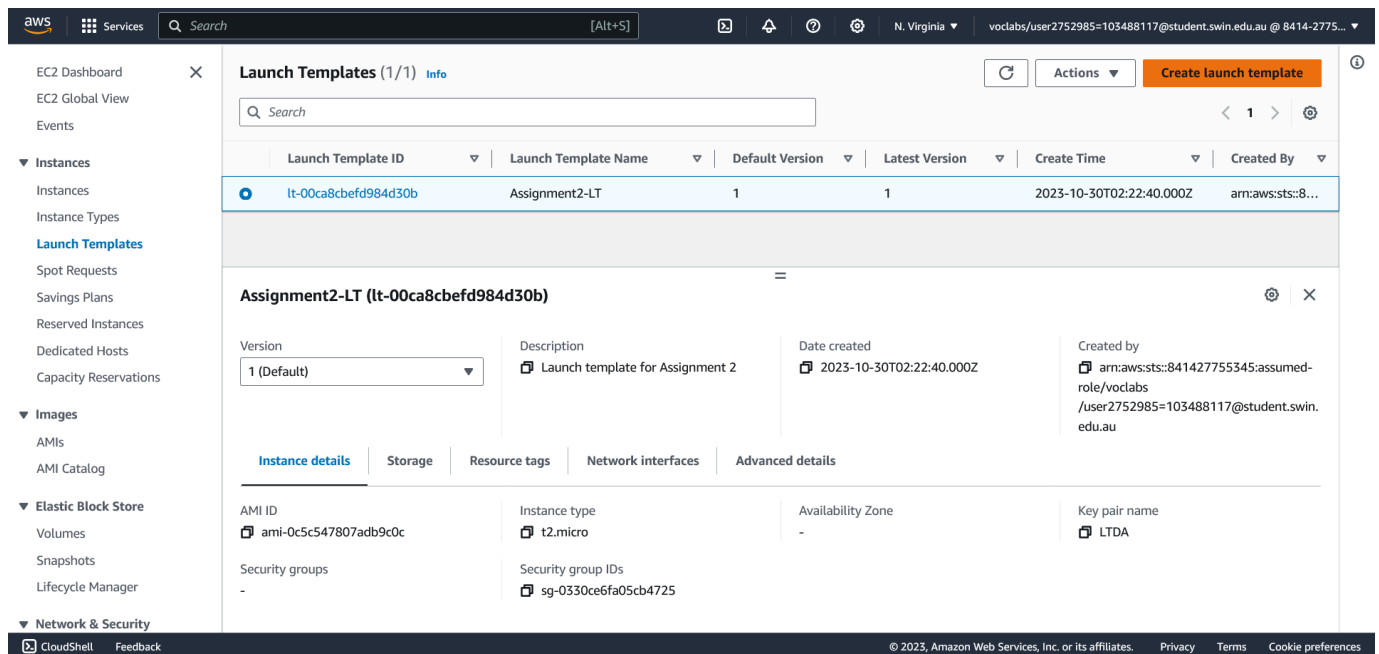


Figure 23: Launch template configured for ASG

We set the instance type for the Launch template to *t2.micro* with the WebServerSG security group and the IAM LabRole assigned to it.

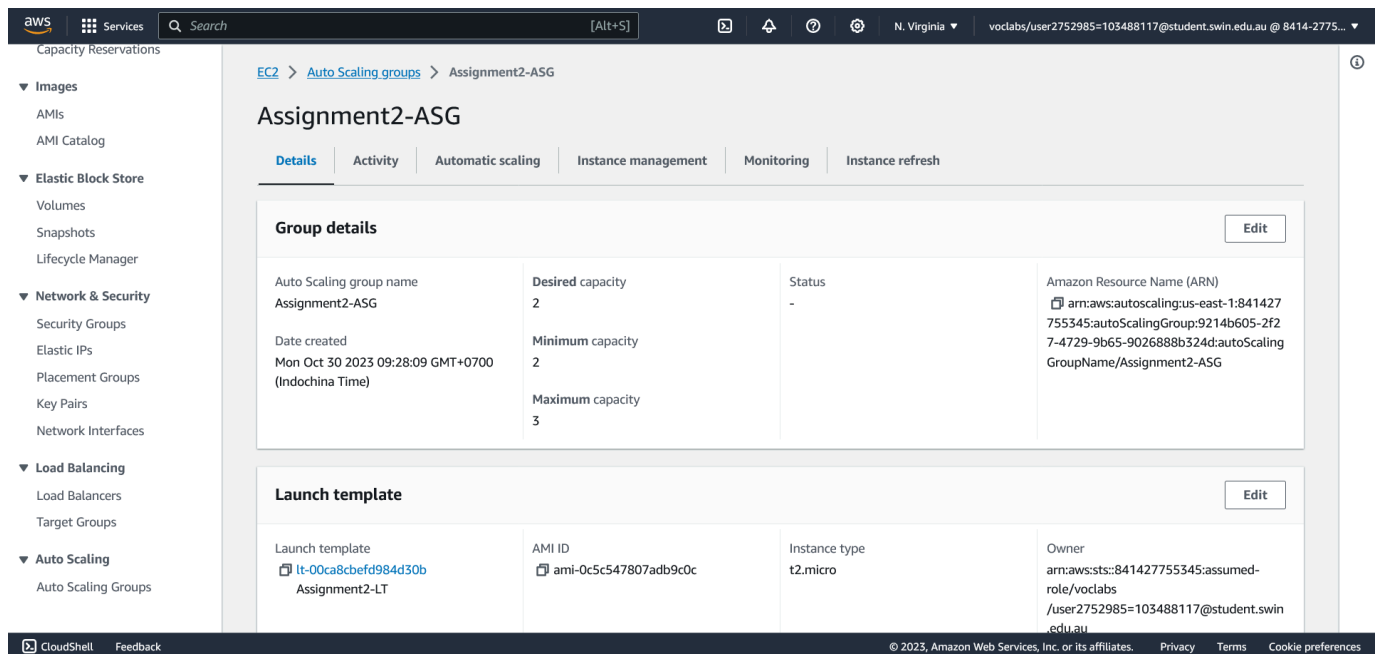


Figure 24: Configuration details for ASG

The ASG scaling policy is set to a minimum capacity of 2 and a maximum capacity of 3. The desired capacity is set to 2 in this scenario. The policy will always keep 2 EC2 instances available while also stopping the website from scaling more than the defined maximum of 3.

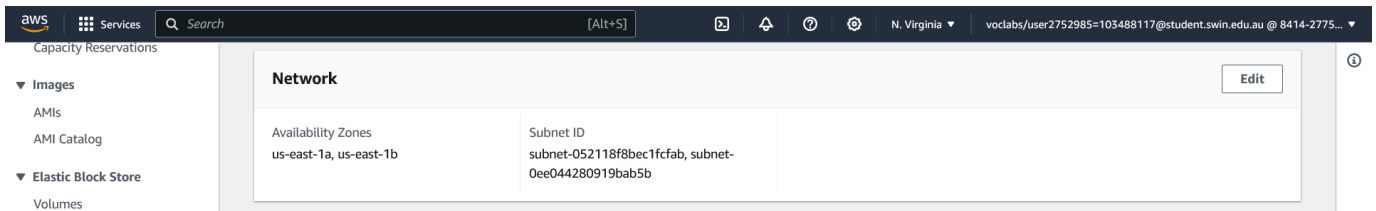


Figure 25: ASG network configuration with 2 private subnets, 10.0.3.0/24 and 10.0.4.0/24

From the architecture diagram, the auto-scaling group will be placed in 2 private subnets, 10.0.3.0/24 and 10.0.4.0/24, where our launch instances will reside.

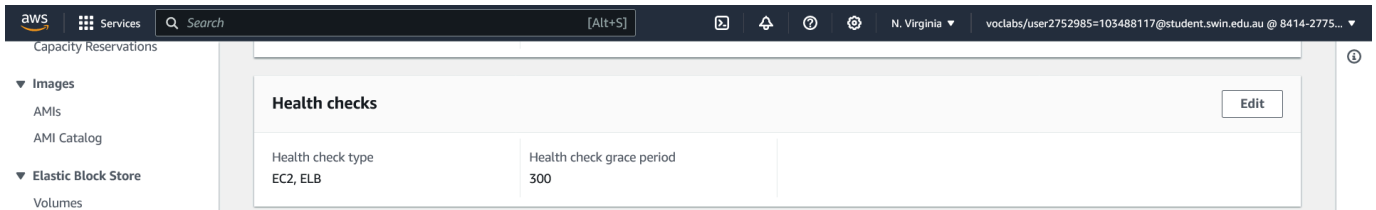


Figure 26: ASG health check type

The health check for our ASG will be set to EC2 and ELB, checking our Web Servers and the ELB up-front.

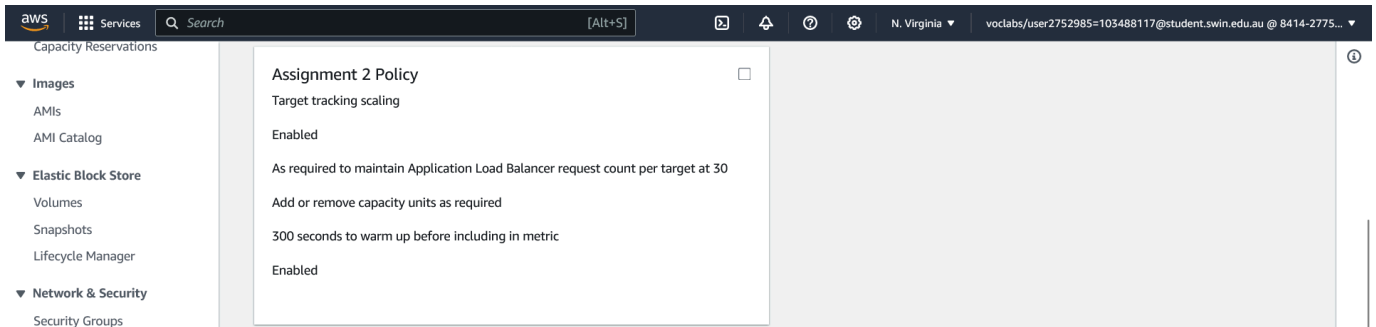


Figure 27: ASG target tracking policy

We will create a target tracking policy to keep the request count per target of my ELB target group at 30 for my ASG.

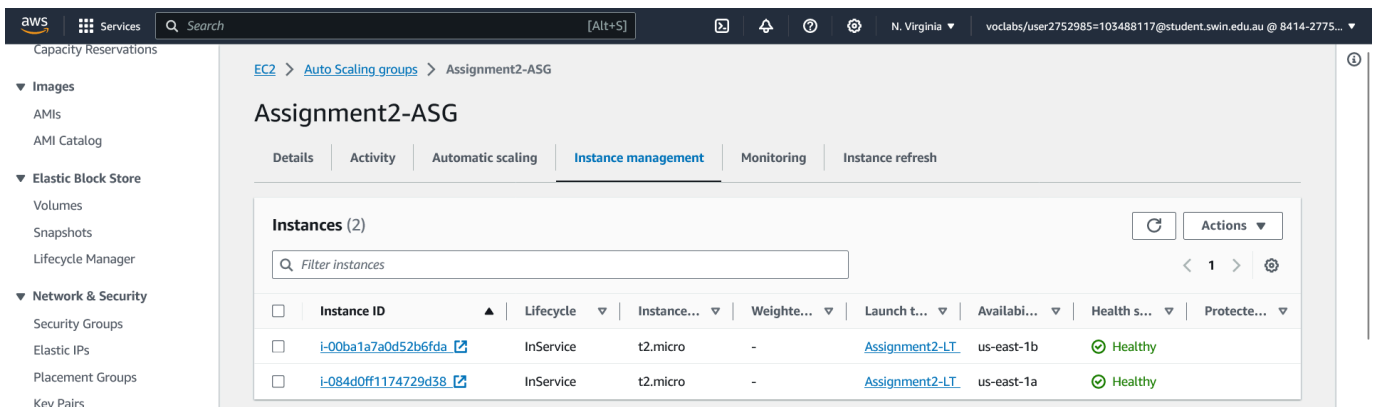


Figure 28: EC2 instances launched across 2 private subnets with healthy health checks

As shown in Figure 28, the Web Server instances are launched across the 2 private subnets with the health status at healthy.

F. Elastic Load Balancing (ELB)

The ELB will be placed in the public subnets to receive HTTP traffic from the internet and direct those traffic requests to healthy Web Server instances.

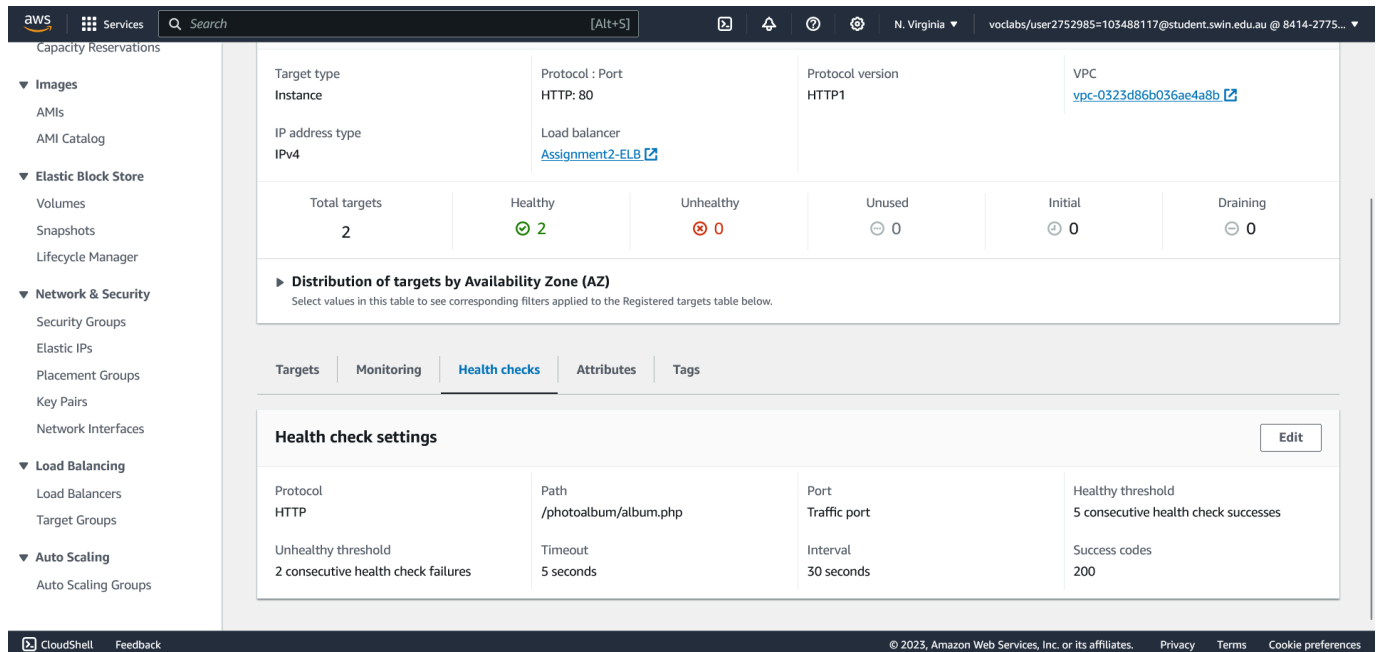


Figure 29: The health check path of the target group is set to `/photoalbum/album.php`

To create an ELB, we first need to create a target group and set the health check path to `/photoalbum/album.php` as it is our website's directory.

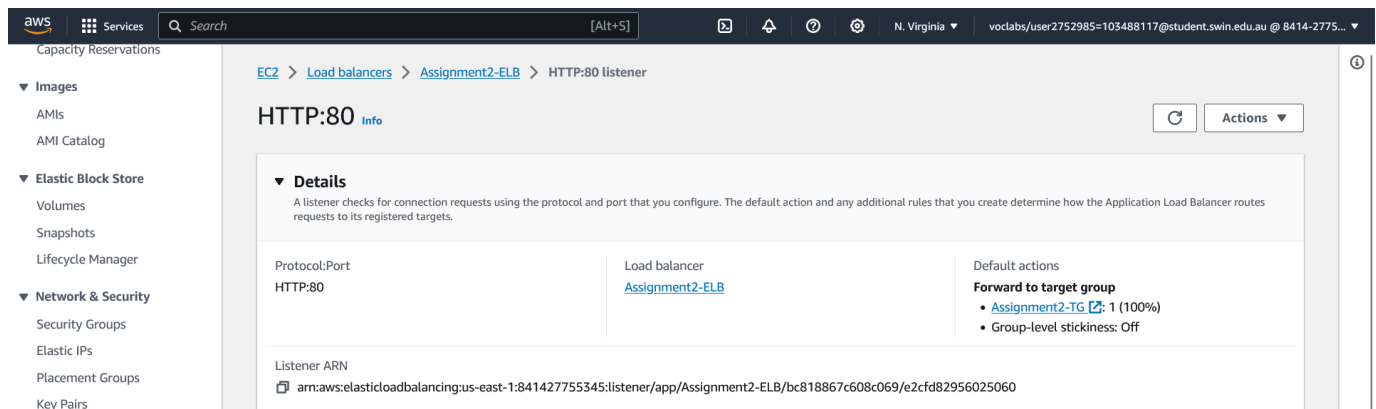


Figure 30: ELB listener forwarding HTTP protocol in port 80 to our target group

The ELB can redirect incoming HTTP requests across EC2 Web instances with the HTTP listener.

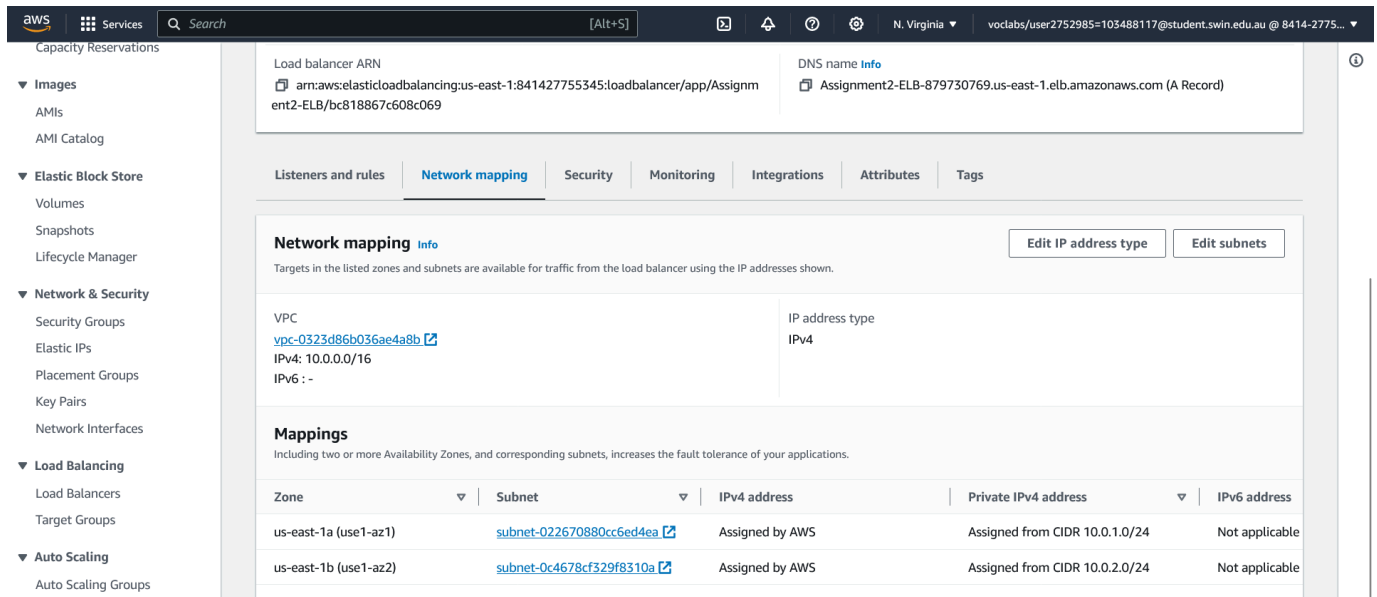


Figure 31: ELB network mappings to 2 public subnets, 10.0.1.0/24 and 10.0.2.0/24

G. Simple Storage Service (S3)

We need to create an S3 bucket with the necessary properties and policies to store the uploaded photos and resized thumbnails.

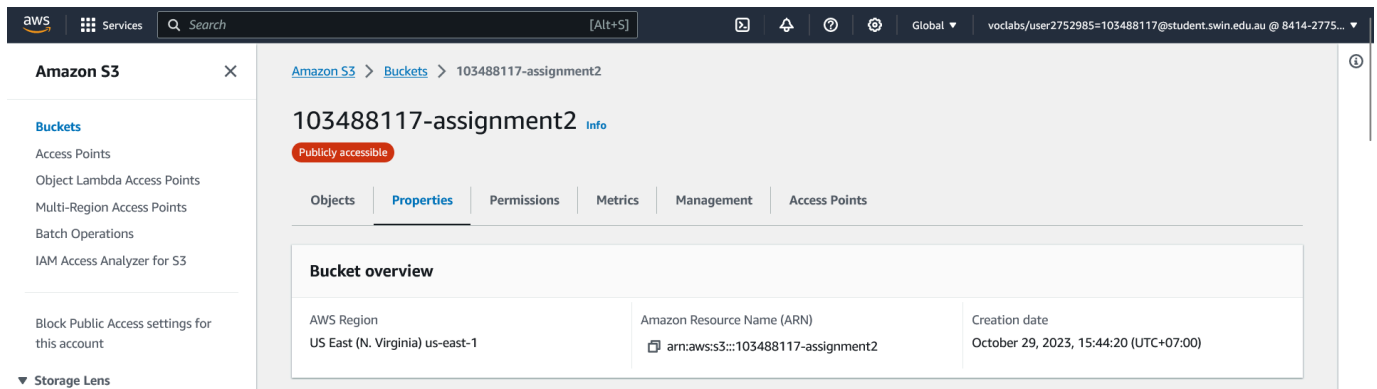


Figure 32: S3 bucket properties

The bucket first needs to be placed in the correct region.

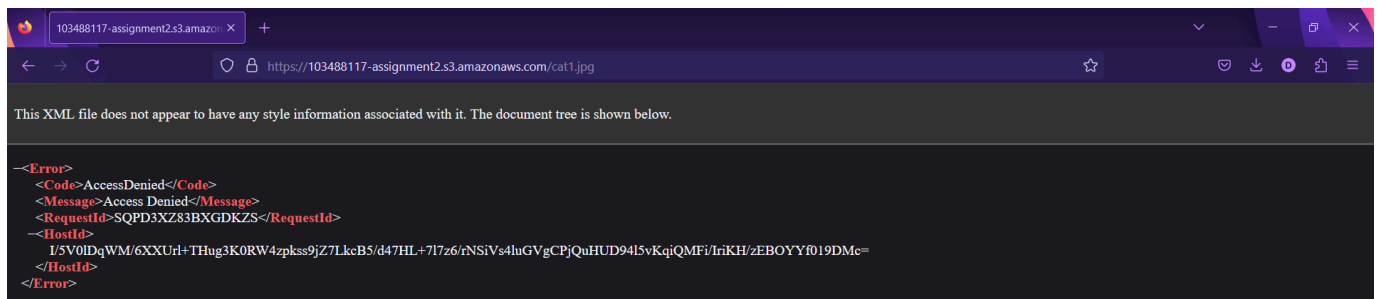


Figure 33: Objects in the S3 bucket are not directly accessible

If we try directly accessing the object through the computer's browser, it will return "Access Denied".

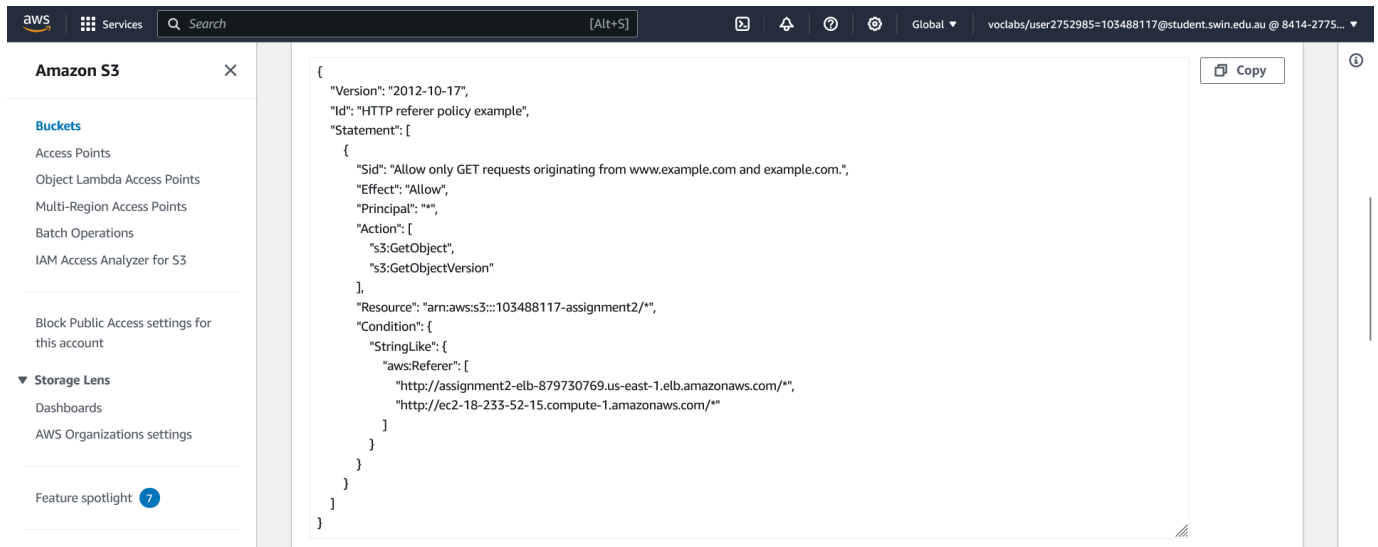
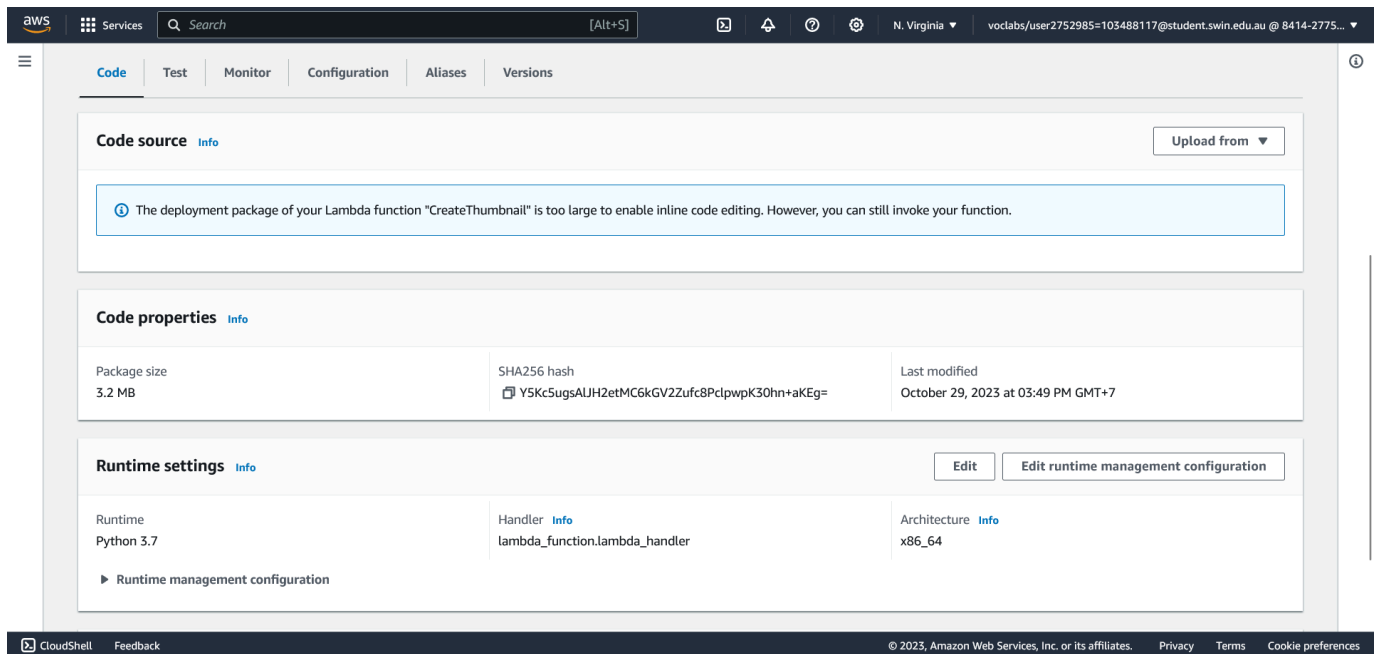


Figure 34: S3 bucket policy URL restriction

For the bucket policy, we restrict access to specific referrers, in this case, the DNS address of the ELB as well as the Dev Server instance. Only from these 2 URLs can we send GET requests to the bucket and get the objects in it.

H. *Lambda*

The Lambda function works by downloading the photo in the S3 bucket, both specified in the payload, resizing it, and uploading the resized image to the same S3 bucket.

Figure 35: Lambda *CreateThumbnail* function after uploading the zip file with Python 3.7 runtime

We must create a Lambda function named *CreateThumbnail* with Python 3.7 runtime environment. Then, we upload the *lambda-deployment-package.zip* file that was provided on Canvas as the deployment package.

I. Relational Database Service (RDS)

We will use the same RDS database configuration as Assignment 1b for this assignment.

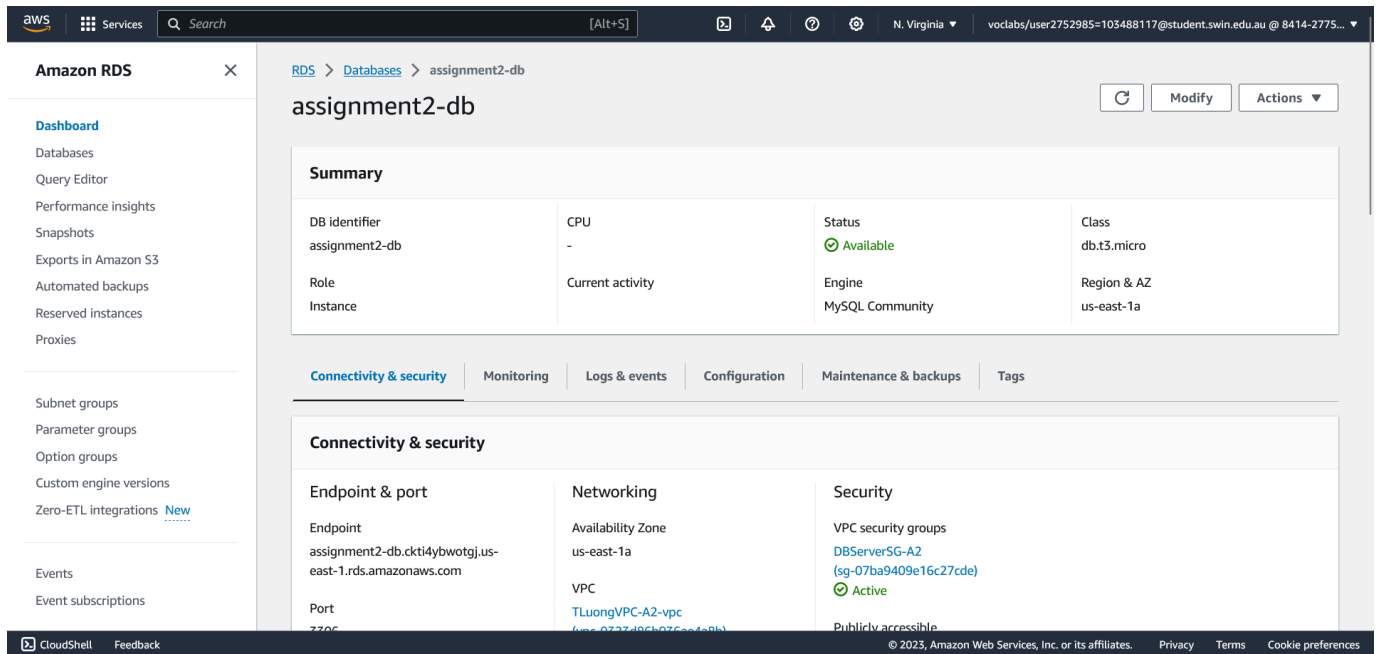


Figure 36: RDS configuration

RDS configuration:

- Engine type: MySQL
- Engine version: 8.0.28
- Templates: Free tier
- DB subnet group: dbsubnetgroup-a2
- Public access: No
- VPC SG: DBServerSG
- AZ: us-east-1a (as specified in the architecture diagram)

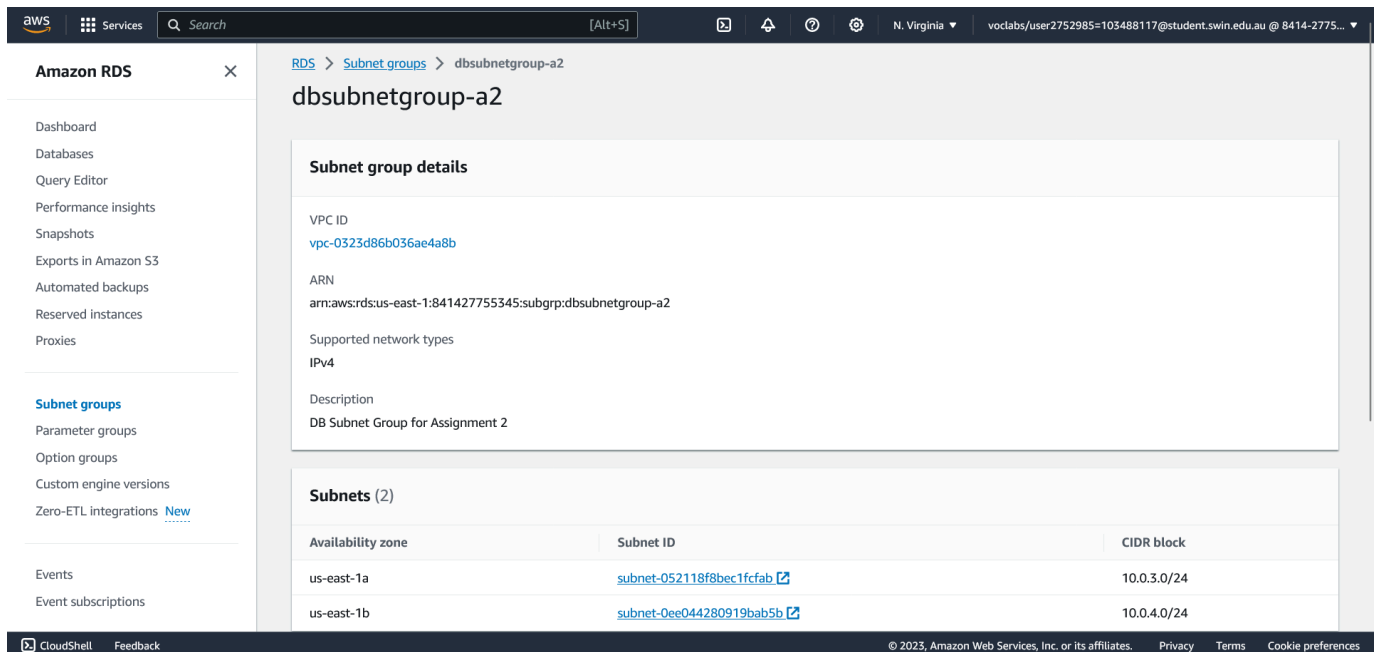


Figure 37: Subnet Group for RDS with 2 private subnets, 10.0.3.0/24 and 10.0.4.0/24

Name: Trac Duc Anh Luong - ID: 103488117

Before creating the database, we must create a subnet group named *dbsubnetgroup-a2* in private subnets 3 and 4 and then assign the group in the database creation wizard.

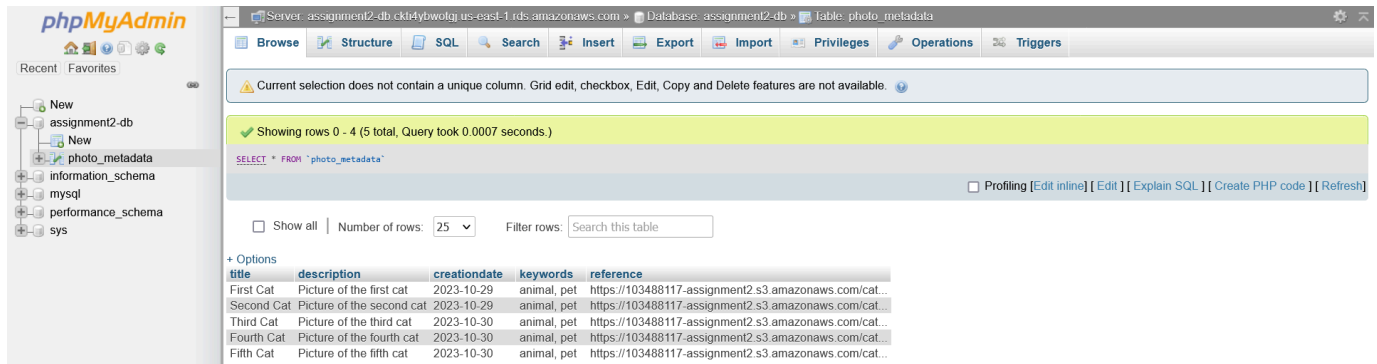


Figure 38: Inserted records viewed from phpMyAdmin

III. FUNCTIONAL REQUIREMENTS

A. Website accessible via ELB

The Photo Album website lets you view the photos in the S3 bucket and their associated meta-data in the RDS database. You can access it through the ELB's public DNS: <http://assignment2-elb-879730769.us-east-1.elb.amazonaws.com/photoalbum/album.php>

On the other hand, the Photo Uploader website is where you can choose to upload your photos and their meta-data to get them displayed in the album, which you can access through this URL: <http://assignment2-elb-879730769.us-east-1.elb.amazonaws.com/photoalbum/photouploader.php>

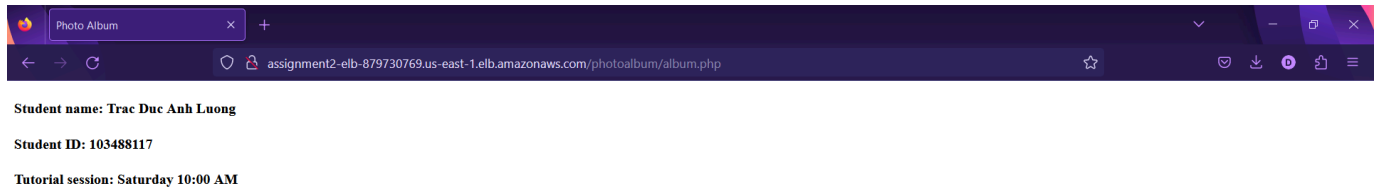


Figure 39: Website accessed through ELB public DNS

B. Photos and their meta-data displayed

Student name: Trac Duc Anh Luong

Student ID: 103488117

Tutorial session: Saturday 10:00 AM

Uploaded photos:

[Upload more photos](#)

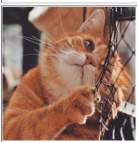

Photo	Name	Description	Creation date	Keywords
	First Cat	Picture of the first cat	2023-10-29	animal, pet
	Second Cat	Picture of the second cat	2023-10-29	animal, pet

Figure 40: album.php displaying the photos and their meta-data

C. Photos and their meta-data uploaded

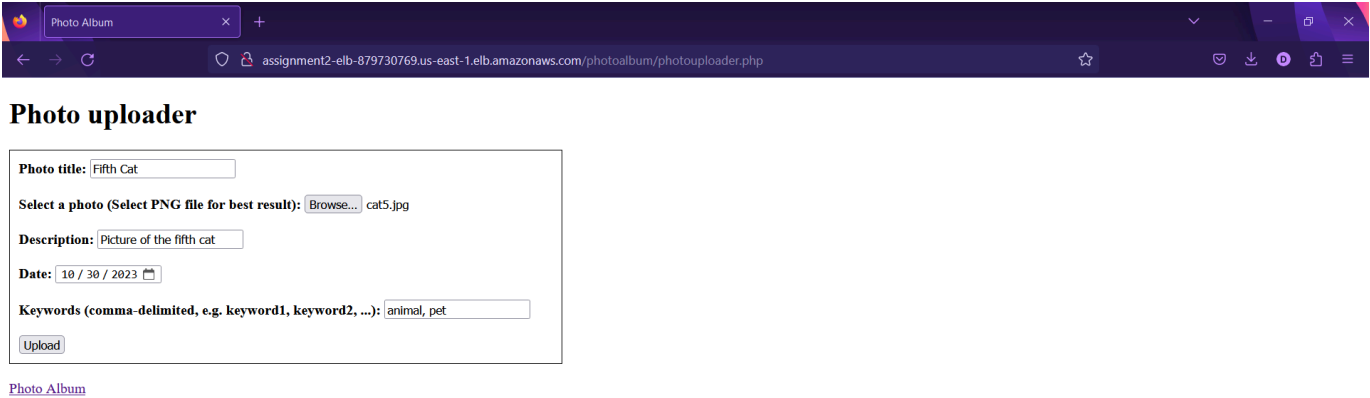


Figure 41: photouploader.php uploading the photos and their meta-data to S3 and RDS

D. Photos resized by the Lambda function

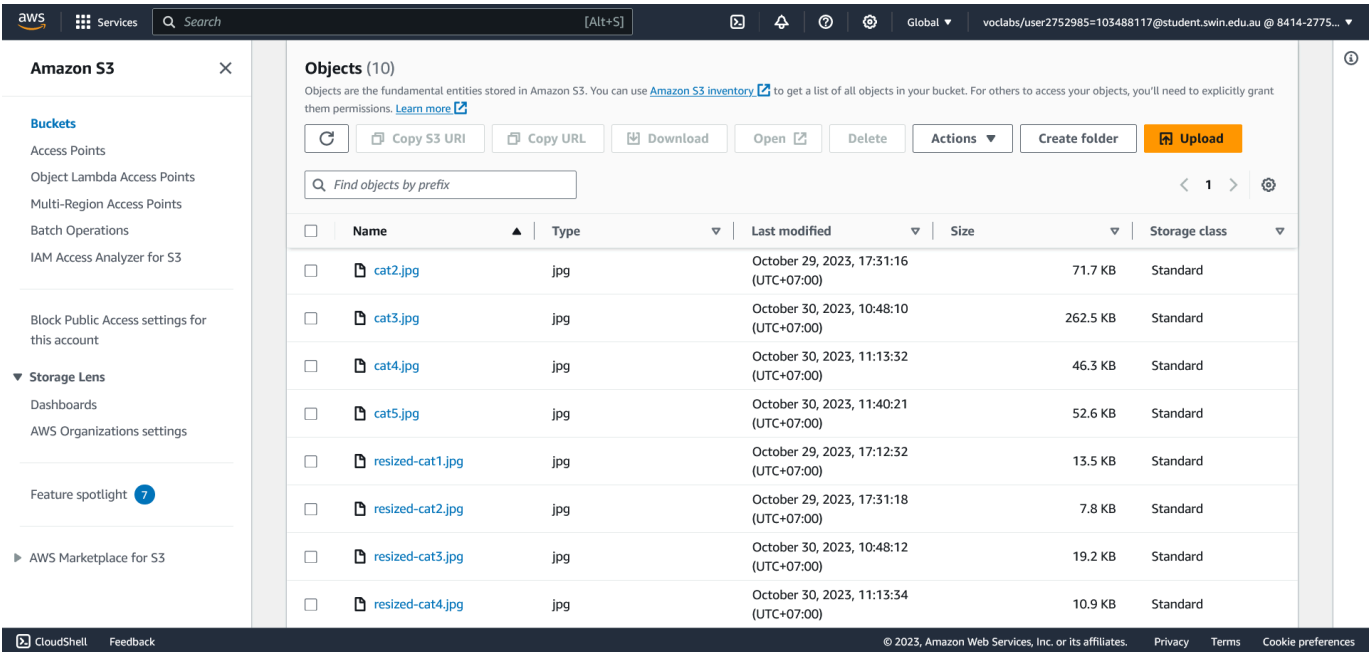


Figure 42: Photos uploaded to S3 and their generated resized thumbnails