

## Assignment 1 - Part B

### Creating and deploying Photo Album website onto a simple AWS infrastructure

Student name: Hai Nam Ngo

Student ID: 103488515

Tutorial session: Friday 04:30PM

URL of Album Page: <http://44.194.3.227/cos20019/photoalbum/album.php>

#### I. Introduction

For the assignment, I will be focusing on a set of key objectives using VPC, RDS, S3 and EC2. Firstly, I will create a secure Virtual Private Cloud (VPC) with subnets, routing tables, and security groups. Following that, I will ensure controlled access to and from the VPC via an Internet Gateway. Additionally, I will modify the provided PHP code to create a website that stores metadata information about photos uploaded to Amazon S3 in a MySQL database managed by Amazon RDS. This website will offer users the capability to search for and display photos using metadata. Subsequently, I will deploy and test the PHP website on an Apache web server running on an EC2 virtual machine instance. Lastly, I will add an extra layer of security by implementing a Network ACL to the public subnet responsible for hosting the web server.

#### II. Infrastructure deployment

The main target is to create an infrastructure looks like this diagram, with web instance attached to public subnet 2, test instance in private subnet 2, RDS instance in private subnet 1, all of them will be attached with the suitable Security Groups, connect with Internet Gateway through web instance. S3 bucket will also be connected to that IGW.

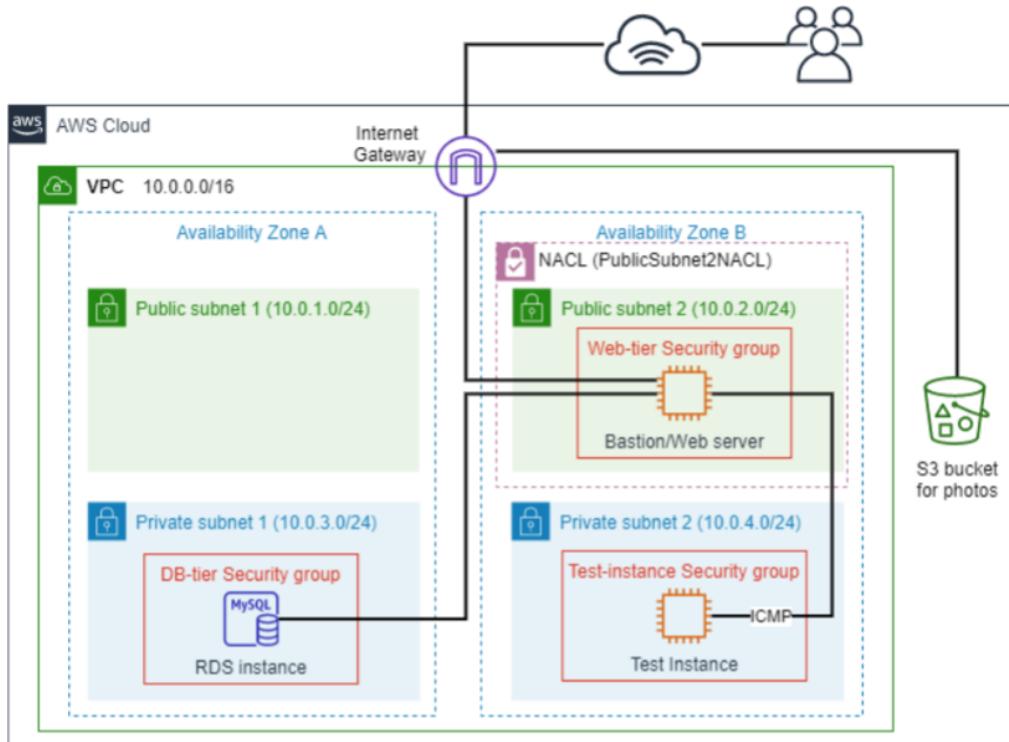


Figure 1 The structure and services in this assignment

## 1.1 – VPC

This step is creating only VPC named “HNgoVPC” based on my name, with the right IPv4 CIDR and keep all other default setting.

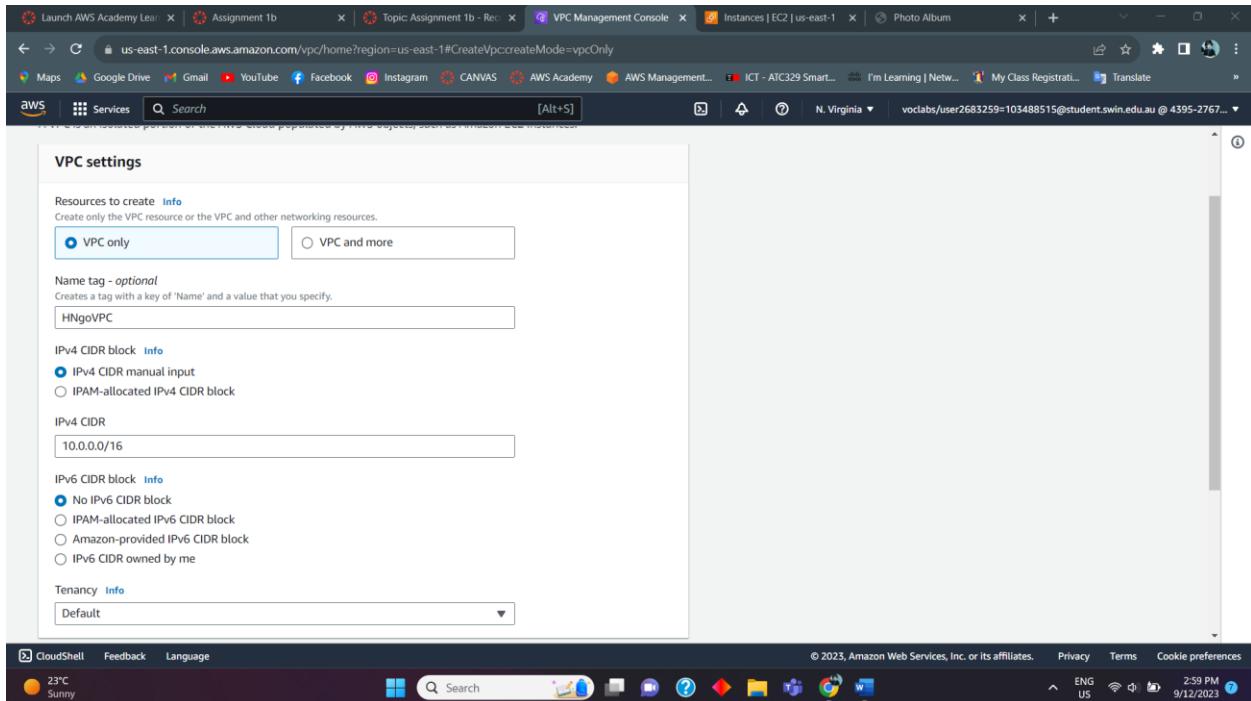


Figure 2 Create VPC (choose "VPC Only" option)

Next step is creating an internet gateway named “HNgo\_IGW” and get ready to attach it with the VPC above.

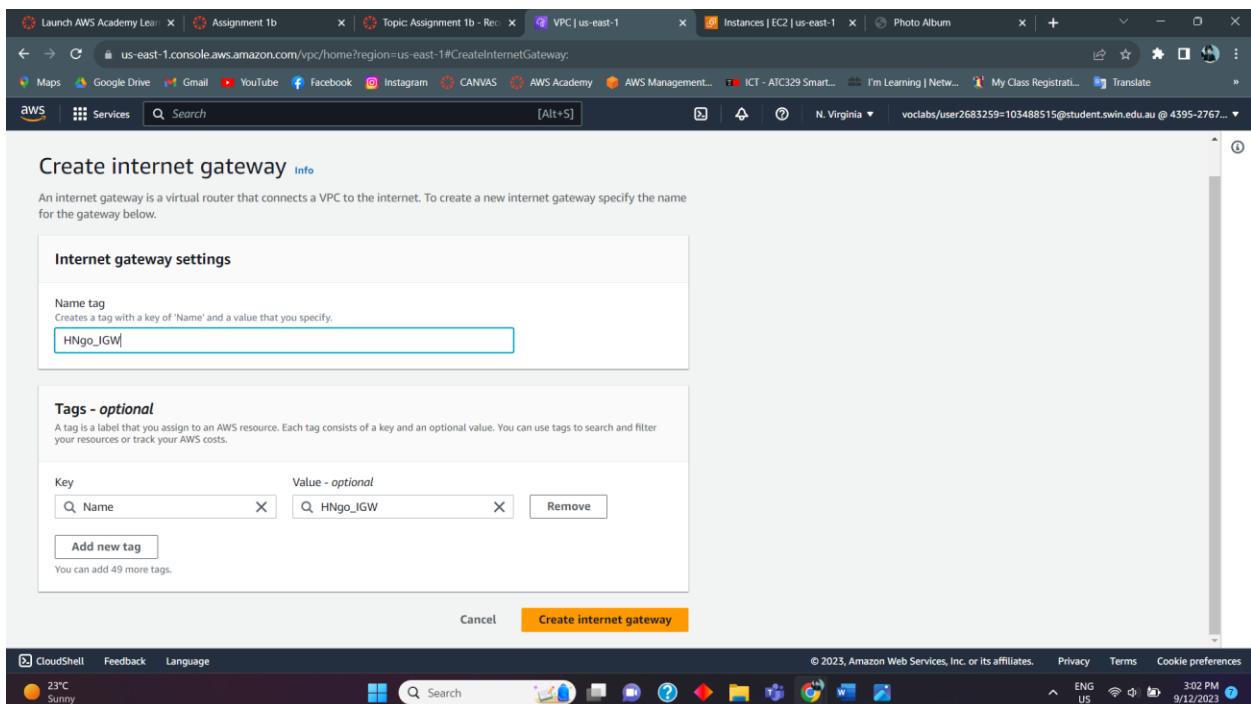


Figure 3 Create Internet gateway in VPC section.

After it is created, click on the “Action” button, and choose to attach, down here is the result. In the VPC ID section, the IGW is already attached to the HNgoVPC.

The screenshot shows the AWS VPC Internet Gateways page. There are two entries in the table:

Name	Internet gateway ID	State	VPC ID
igw-076b6bf45765d088a	Attached	vpc-012b4a48fea101a5d	
<b>HNgo_IGW</b>	Attached	vpc-0b9f7e487a7e5ea0f   HNgoVPC	

The modal below provides more details for the selected gateway (HNgo\_IGW):

Internet gateway ID	State	VPC ID	Owner
igw-0c97280273049a4ea	Attached	vpc-0b9f7e487a7e5ea0f   HNgoVPC	439527679533

Figure 4 Attach IGW to HNgoVPC

Next step is Creating Route Tables, in this case, the route table will be attached to the HNgoVPC, and add the internet route in Public Route Table, so after this step, there will be 2 routes available in the route section, one is for the internet, and one for local.

The screenshot shows the AWS Route Tables page. There are three entries in the table:

Route table ID	Explicit subnet associations	Main	VPC	Owner
rtb-031620bbef515bc24	-	No	vpc-0b9f7e487a7e5ea0f   HNgoVPC	439527679533
rtb-05fa43a8e5ca523f5	-	Yes	vpc-012b4a48fea101a5d	439527679533
<b>rtb-08d01fec1a6fd7e67</b>	2 subnets	-	vpc-0b9f7e487a7e5ea0f   HNgoVPC	439527679533

The modal below shows the routes for the selected route table (rtb-08d01fec1a6fd7e67):

Destination	Target	Status	Propagated
0.0.0.0/0	igw-0c97280273049a4ea	Active	No
10.0.0.0/16	local	Active	No

Figure 5 Connect route table with internet route.

Next step is Creating Subnets in right AZs and associate the right Route Table to them. For the assignment, there will be 4 subnets with 2 Availability Zones, they are created correctly with the right IPv4 CIDR.

The screenshot shows the AWS VPC Subnets page. On the left, a sidebar lists various VPC-related services and resources. The main area displays a table of subnets with the following data:

Name	Subnet ID	State	VPC	IPv4 CIDR
Public subnet 1	subnet-01c135315bb663a24	Available	vpc-0b9f7e487a7e5ea0f   HNg...	10.0.1.0/24
-	subnet-0d7aa739146847d69	Available	vpc-012b4a48fea101a5d	172.31.0.0/20
-	subnet-0c0377238d020c81	Available	vpc-012b4a48fea101a5d	172.31.48.0/20
-	subnet-0f2dfc2ea619d7c46	Available	vpc-012b4a48fea101a5d	172.31.32.0/20
Private subnet 2	subnet-03b6d232463f7a185	Available	vpc-0b9f7e487a7e5ea0f   HNg...	10.0.4.0/24
-	subnet-04425df54f5ec5d46	Available	vpc-012b4a48fea101a5d	172.31.80.0/20
Private subnet 1	subnet-0b406fd96c4133a4	Available	vpc-0b9f7e487a7e5ea0f   HNg...	10.0.3.0/24
-	subnet-0cef27643b2461fb	Available	vpc-012b4a48fea101a5d	172.31.16.0/20
Public subnet 2	subnet-d7582428884d44f0	Available	vpc-0b9f7e487a7e5ea0f   HNg...	10.0.2.0/24
-	subnet-0e890f0187e76257f	Available	vpc-012b4a48fea101a5d	172.31.64.0/20

Figure 6 Successfully creating subnets.

Then, associate public subnets with a public route table that routes to the Internet Gateway.

The screenshot shows the AWS Route Tables page. The main area displays a table of route tables with the following data:

Route table ID	Explicit subnet associations	Main	VPC
rtb-031620bbef515bc24	-	No	vpc-0b9f7e487a7e5ea0f   HNg... 439527...
rtb-05fa43a8e5ca523f5	-	Yes	vpc-012b4a48fea101a5d 439527...
rtb-08d01fec1a6fd7e67	2 subnets	Yes	vpc-0b9f7e487a7e5ea0f   HNg... 439527...

Below the table, the details for the route table 'rtb-08d01fec1a6fd7e67' are shown, specifically the 'Subnet associations' tab. It lists the explicit subnet associations:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
Public subnet 1	subnet-01c135315bb663a24	10.0.1.0/24	-
Public subnet 2	subnet-d7582428884d44f0	10.0.2.0/24	-

Figure 7 Make public subnets link with the public route table successfully.

## 1.2 Create Security Groups

Three security groups will be created and each of them will be attached to the correct EC2 and RDS.

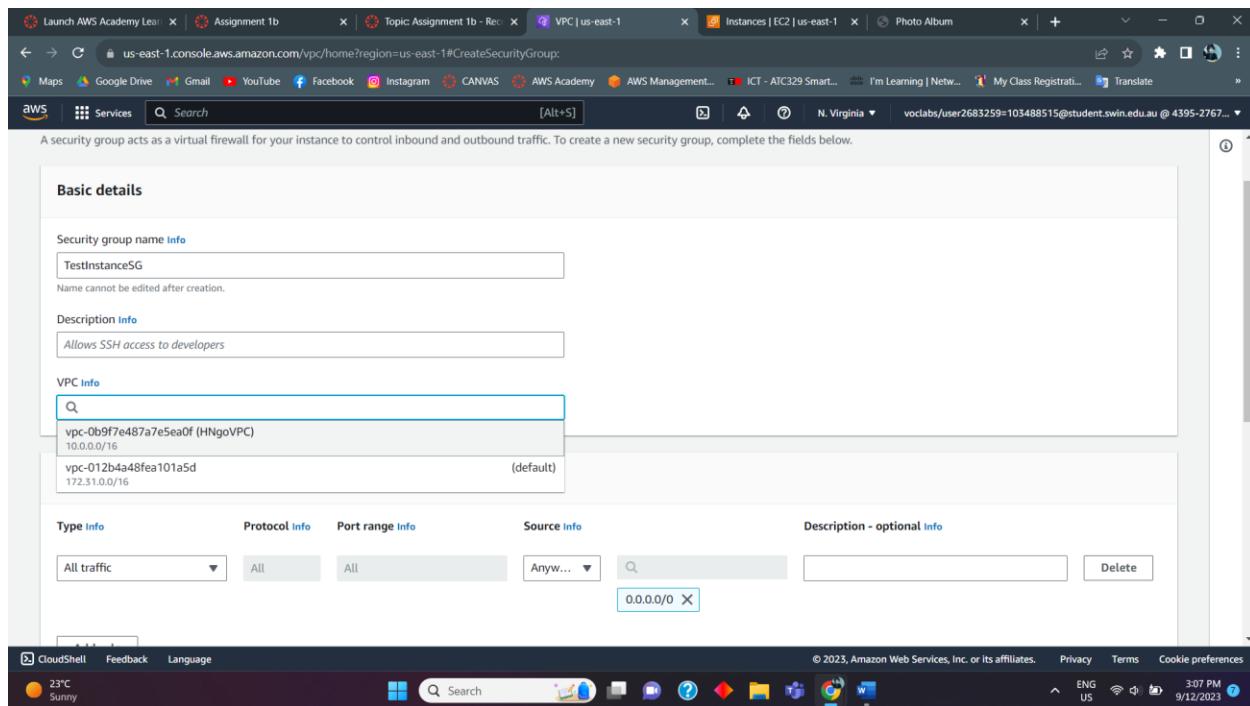


Figure 8 Creating Test Instance SG and include the inbound rule the same as the requirement.

After creating three SG, they will be displayed (highlight section) and ready to be attached.

Security group name	VPC ID	Description	Owner	Inbound rules count	Outbound rules count
rds-ec2-1	vpc-0b9f7e487a7e5ea0f	Security group attached to rds-ec2-1	439527679533	1 Permission entry	0 Permission entries
ec2-rds-1	vpc-0b9f7e487a7e5ea0f	Security group attached to ec2-rds-1	439527679533	0 Permission entries	1 Permission entry
WebServer-SG	vpc-012b4a48fea101a5d	launch-wizard-1 created	439527679533	3 Permission entries	1 Permission entry
default	vpc-0b9f7e487a7e5ea0f	default VPC security group	439527679533	1 Permission entry	1 Permission entry
rds-ec2-2	vpc-0b9f7e487a7e5ea0f	Security group attached to rds-ec2-2	439527679533	1 Permission entry	0 Permission entries
default	vpc-012b4a48fea101a5d	default VPC security group	439527679533	1 Permission entry	1 Permission entry
TestInstanceSG	vpc-0b9f7e487a7e5ea0f	TestInstanceSG	439527679533	1 Permission entry	1 Permission entry
WebServerSG	vpc-0b9f7e487a7e5ea0f	WebServerSG	439527679533	3 Permission entries	1 Permission entry
DBServerSG	vpc-0b9f7e487a7e5ea0f	DBServerSG	439527679533	1 Permission entry	1 Permission entry
ec2-rds-2	vpc-0b9f7e487a7e5ea0f	Security group attached to ec2-rds-2	439527679533	0 Permission entries	1 Permission entry

Figure 9 Successfully creating three SG.

### 1.3. EC2 virtual machine

Public DNS will change every time the webserver instance restarts. To avoid this behavior and to ensure the Webserver URL remains persistent, Elastic IP Address will be added to this instance by allocating an Elastic IP Address.

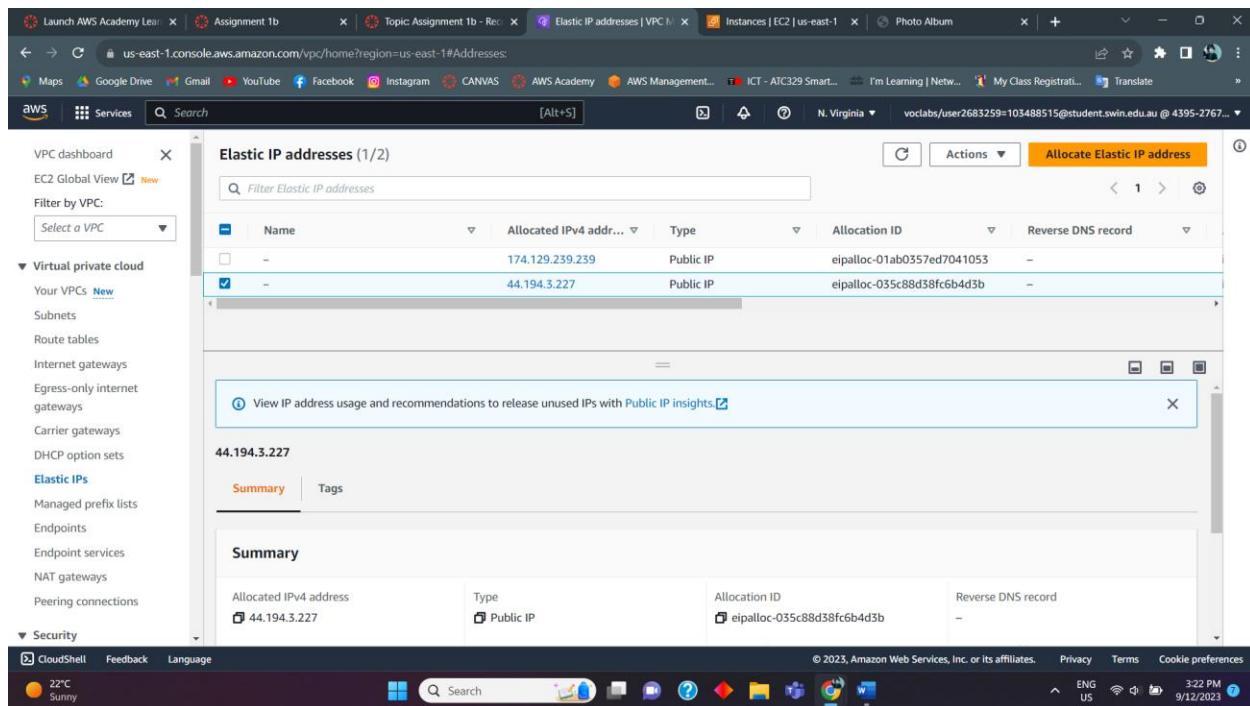


Figure 10 Create an Elastic Public IP address.

After that, Bastion/Web Server instance and test instance is created based on the requirement of the assignment, connect with the right security group, and Elastic IP Address is successfully associated with the instance.

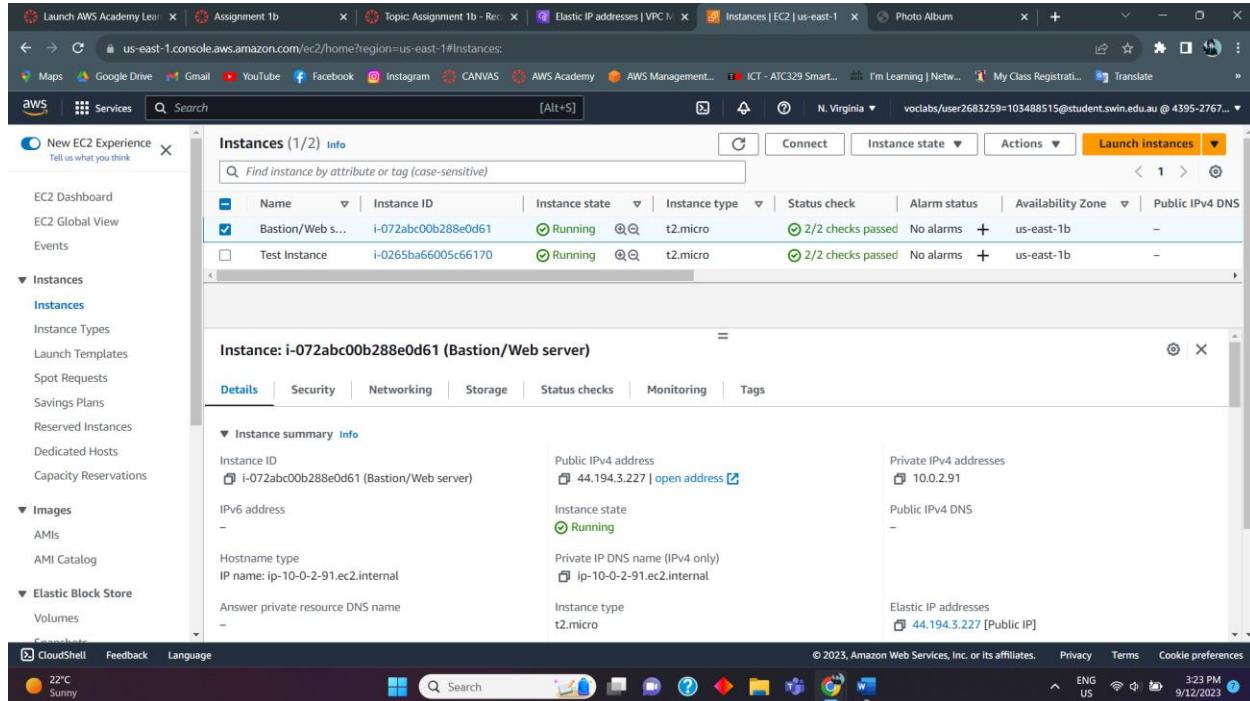


Figure 11 Create the two necessary instances.

## 1.4. RDS database instance

This step will create an RDS database, put it in the private subnet 1 and connect it with the Bastion/Web Server Instance. Because I have done the part of connecting RDS with EC2, so the security group will also have the rds-ec2-2 group (I delete the first rds-ec2-1 by accident so I need to connect them again).

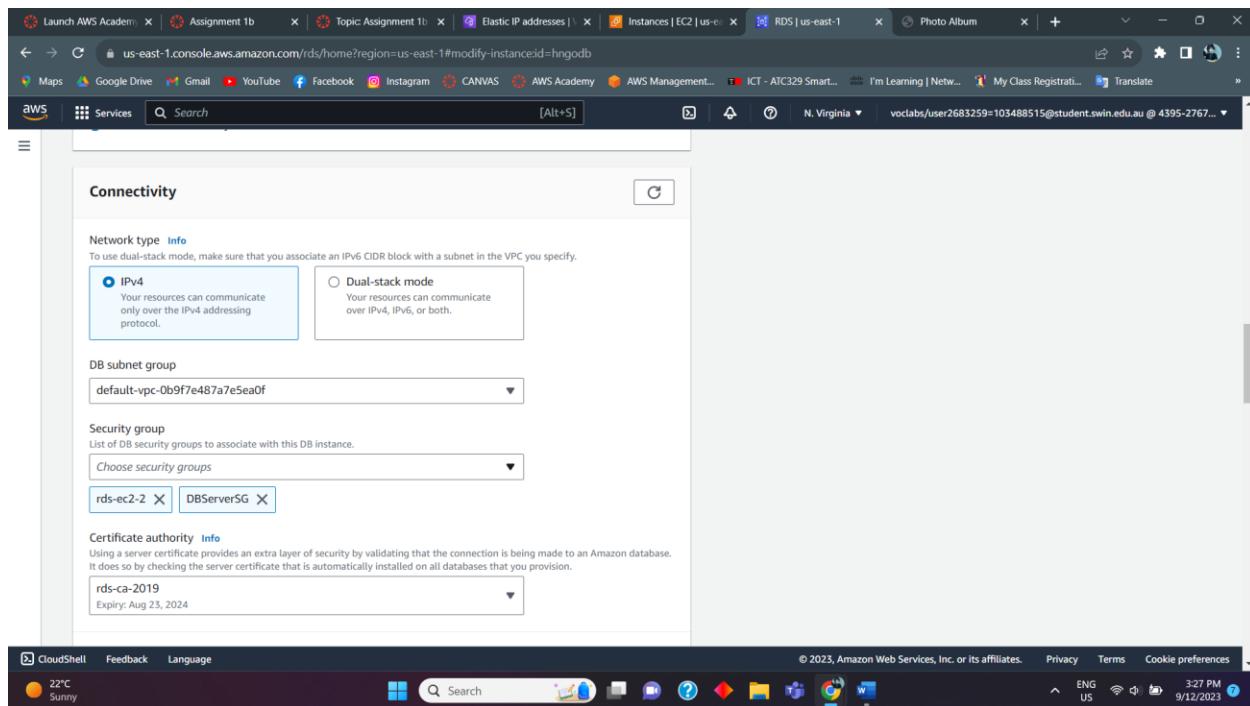


Figure 12 Modify it with the right security group.

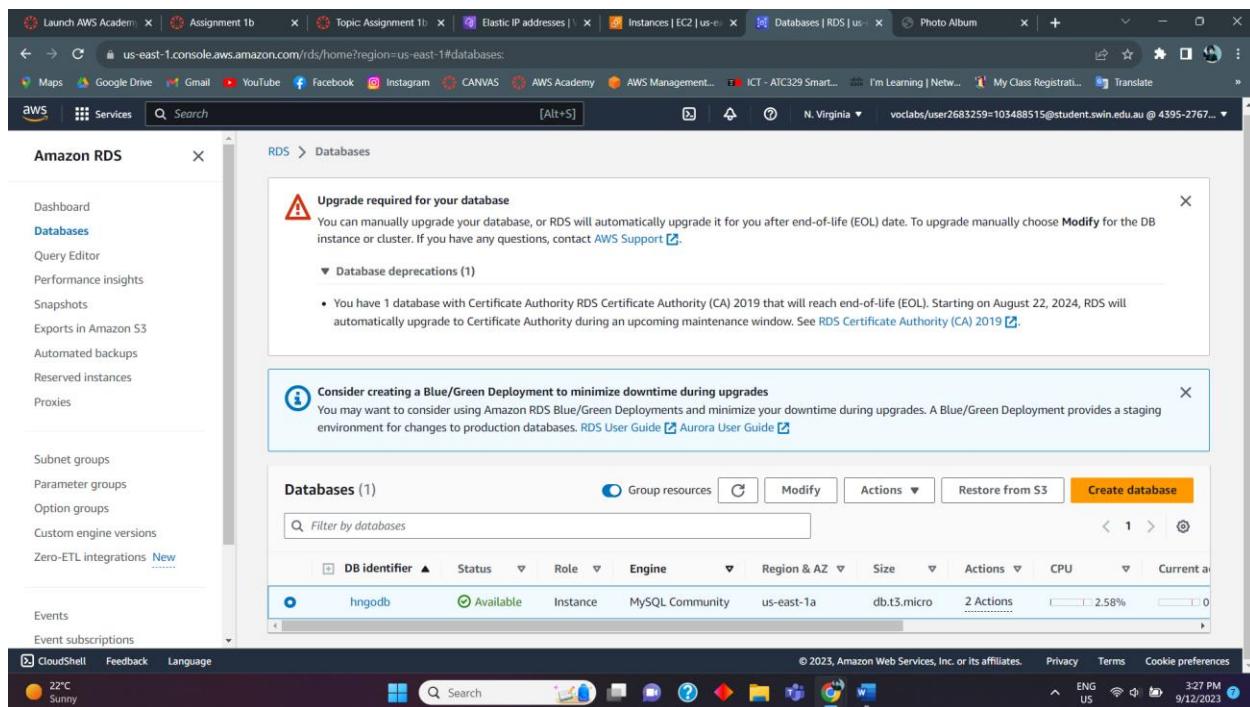


Figure 13 Create RDS named "hngodb".

After creating RDS, create and configure the right Database Table on RDS, using Web Server instance. To do that, PuTTY is required. The private key is downloaded from the learner lab. The host name will be the Elastic IP Address of the Bastion/Web Server Instance.

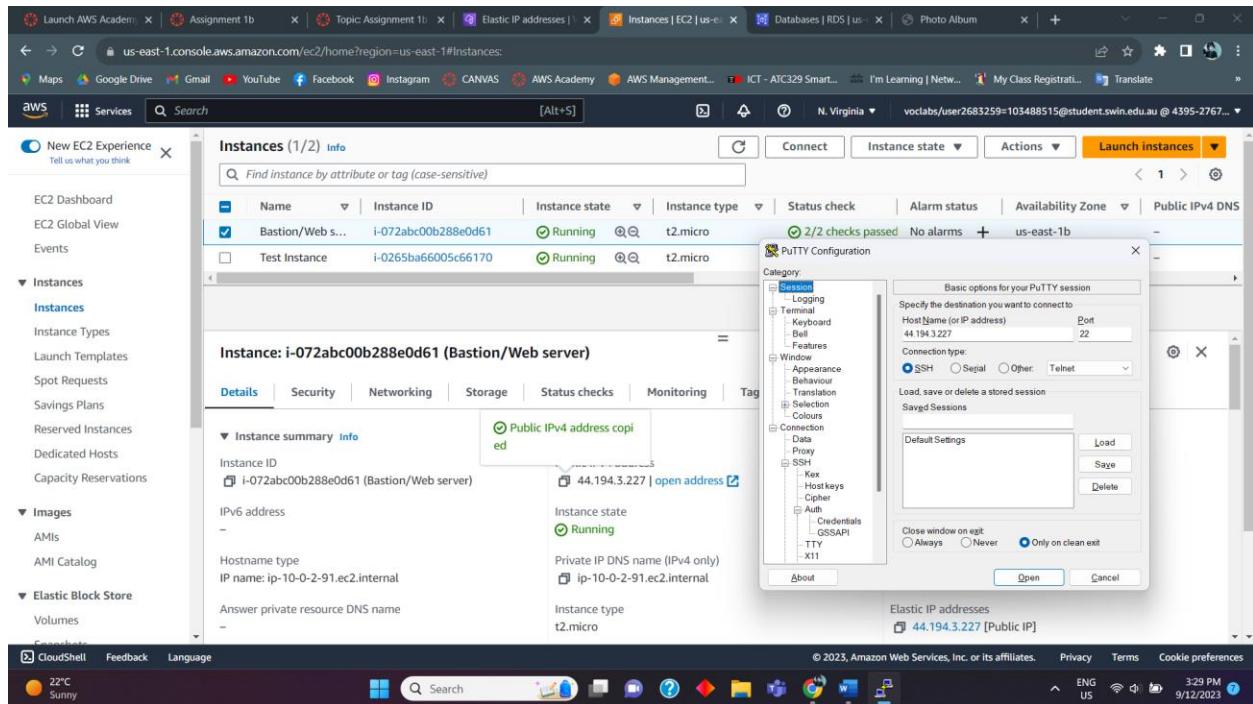


Figure 14 Access to Web Server Instance using PuTTY.

After successfully logging in to the terminal, the next step is to download the phpMyAdmin file into the EC2, because I have already done this, the result will be displayed down below.

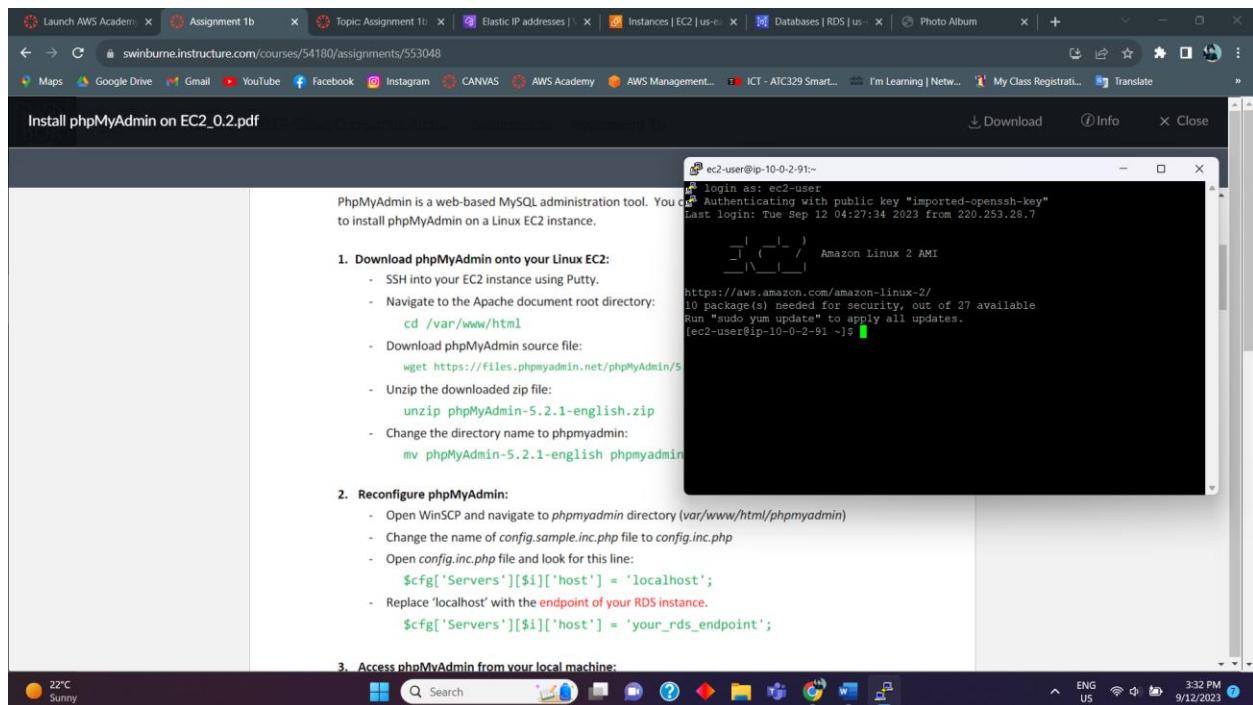


Figure 15 Log into Web Instance Linux

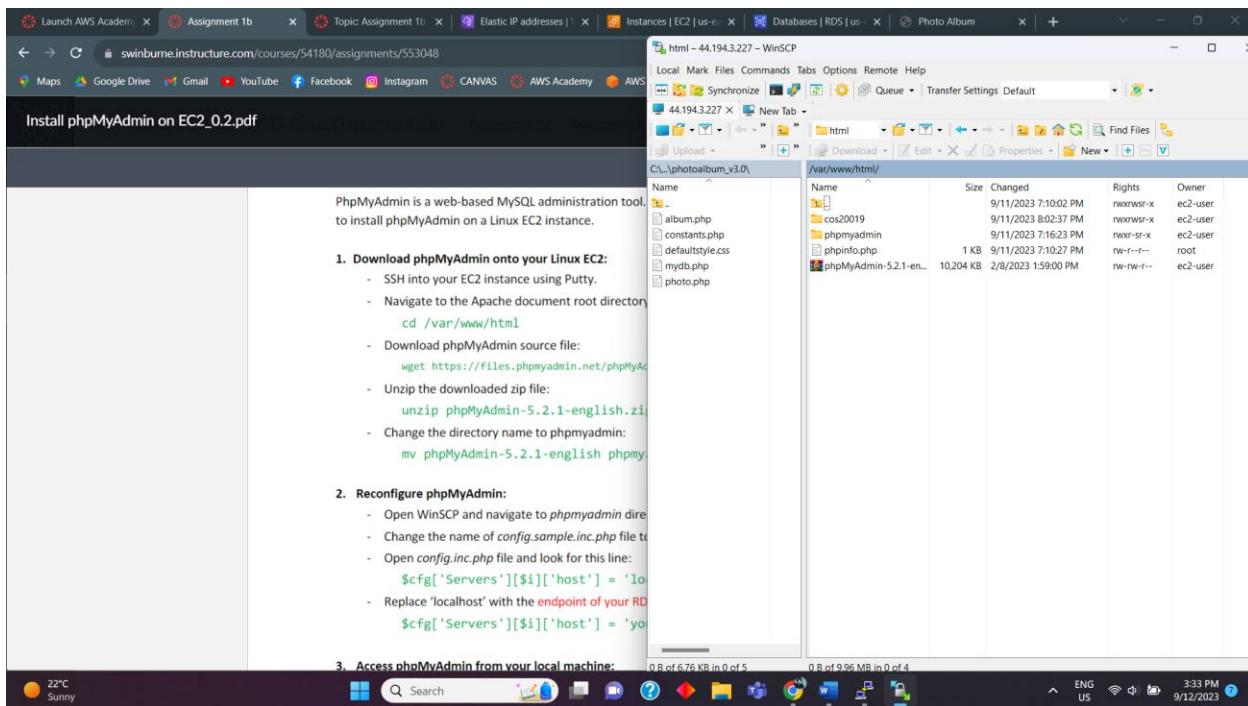


Figure 16 PhpMyAdmin file is downloaded and unzip successfully.

After that, the config.inc.php must be modified by replacing the host server with the link to the hngodb RDS.

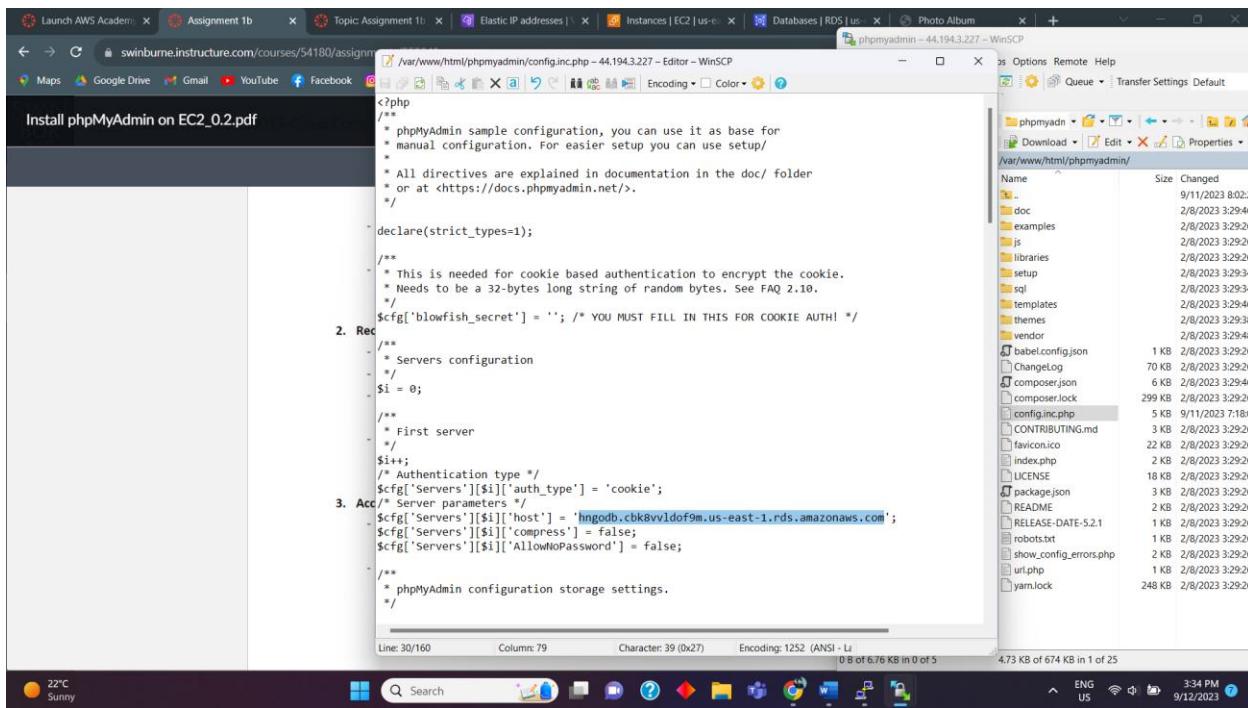


Figure 17 Modify the host server link.

For the next step, I went to the phpMyAdmin website and typed in the username and password to log in. The password and username have been written inside the constant file.

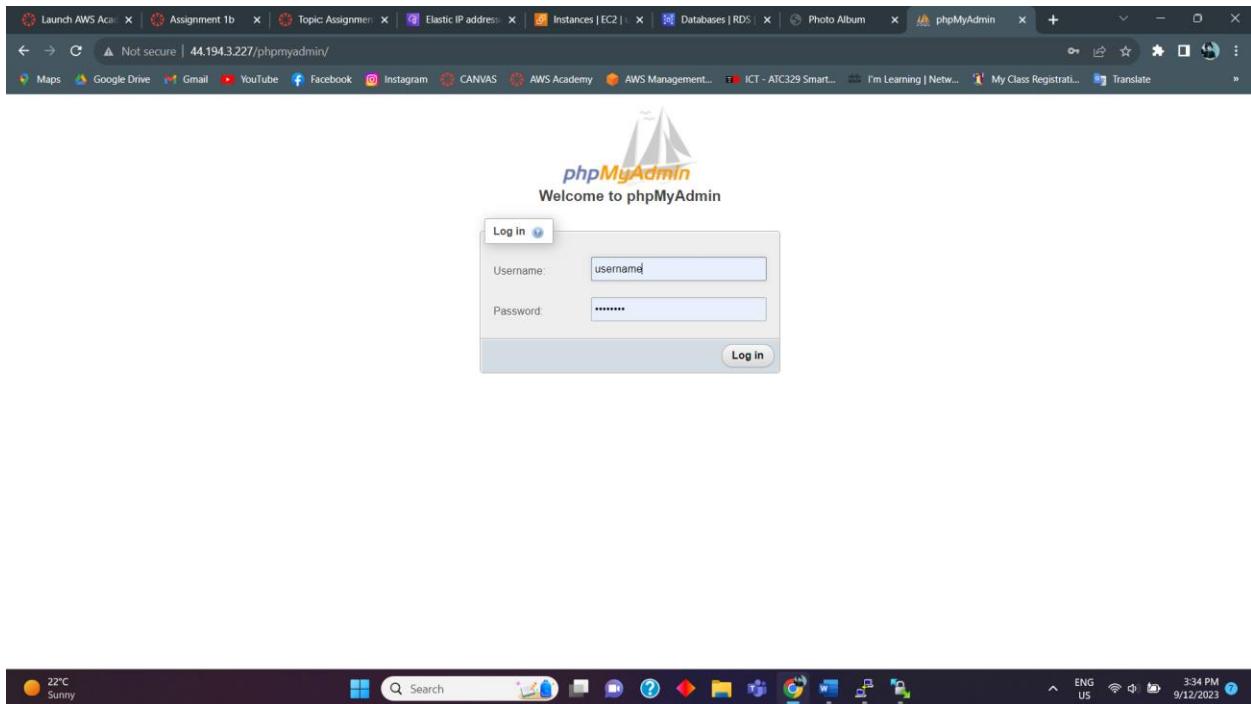


Figure 18 Log into phpMyAdmin

The next step is creating a photo database, where will keep the table “photos”.

Figure 19 Create a database "photos" and table called "photos" as well.

According to the requirement, the table has been created with 5 columns: photo title, description, creation date, keywords, and reference, four varchar (255) columns and one date column.

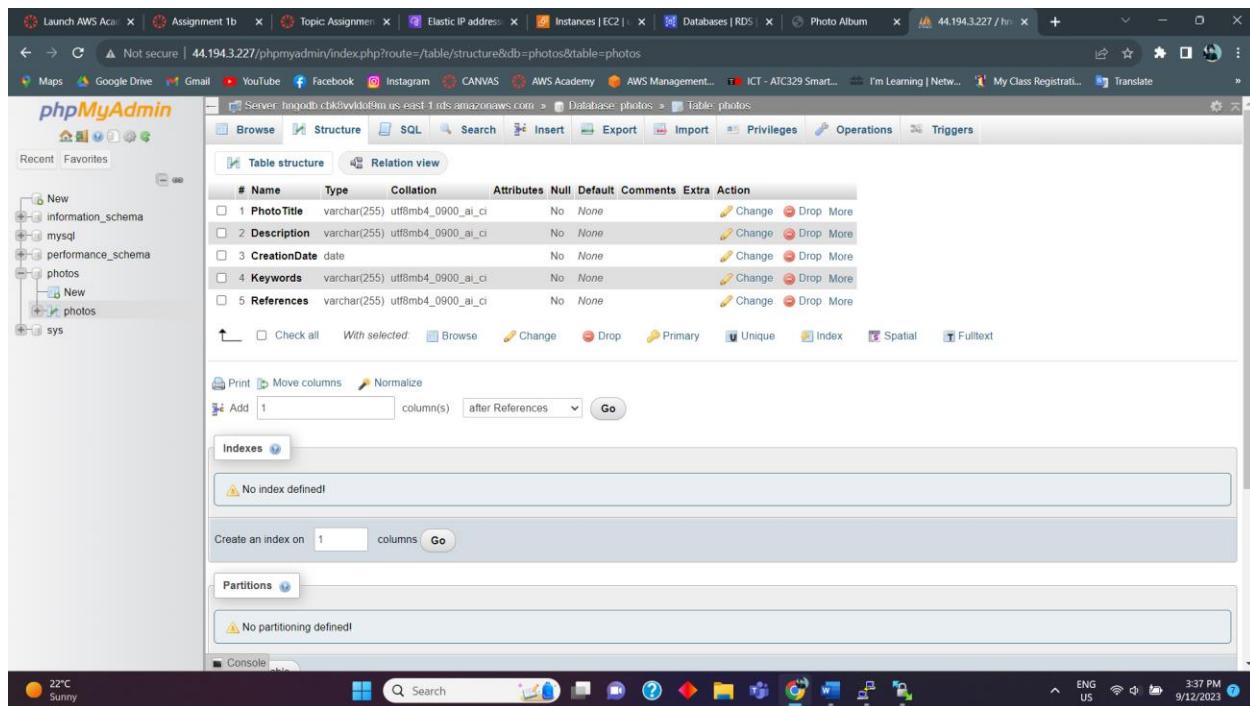


Figure 20 Table with 5 columns has been created.

## 1.5. Network ACL

To enhance the server's security, the next step is creating and implementing a Network ACL named "PublicSubnet2NACL" for Public Subnet 2. This ACL should strictly adhere to the principle of least privilege, allowing only essential traffic. Specifically, it should permit SSH (port 22) traffic from any source for Webserver instance access, allow ICMP traffic exclusively from the Test instance's subnet, and enable other required traffic to ensure full functionality of the Photo Album website for users everywhere.

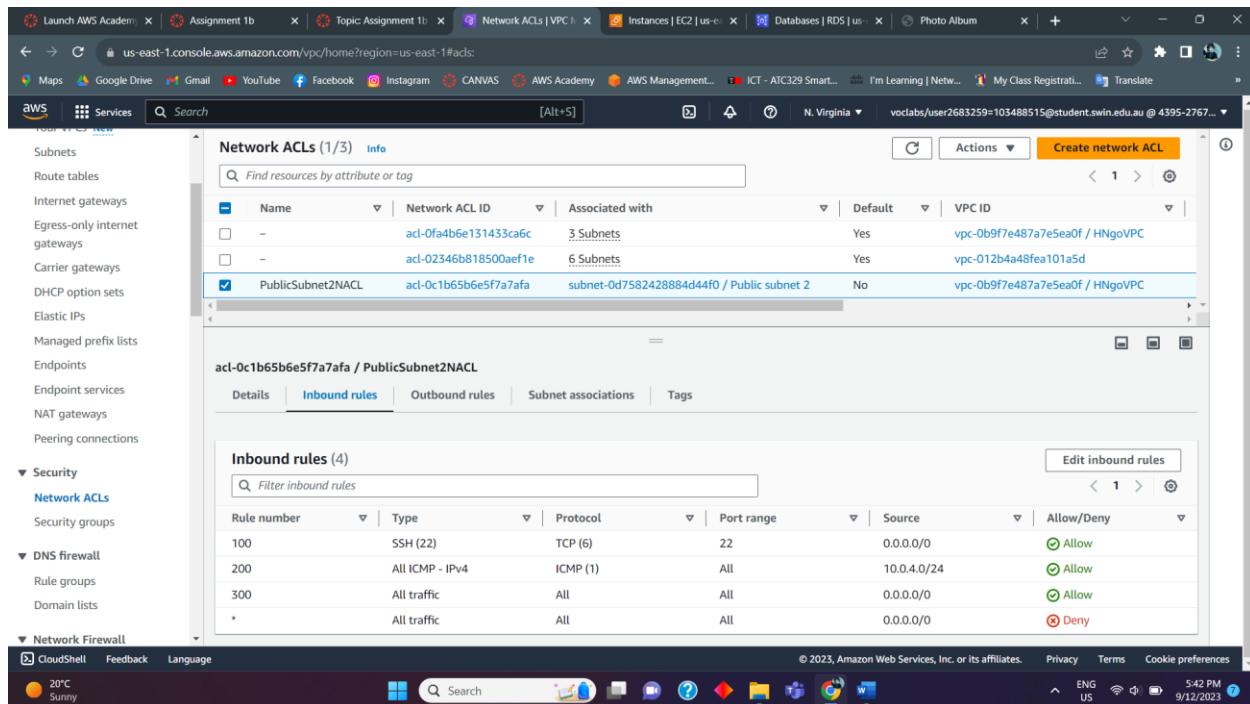


Figure 21 PublicSubnet2NACL is created with inbound and outbound rules, and connect it with public subnet 2

### III. Functional requirements of Photo Album website

#### 2.1. Photo storage

For this step, it is required to create an S3 Bucket with appropriate access policy.

The screenshot shows the AWS S3 Management console. On the left, the navigation pane is visible with sections like Buckets, Storage Lens, and Feature spotlight. The main content area displays an 'Account snapshot' with a link to 'View Storage Lens dashboard'. Below it, a table lists the single bucket 'bucketforphoto' with details: Name (bucketforphoto), AWS Region (US East (N. Virginia) us-east-1), Access (Public), and Creation date (September 11, 2023, 19:37:32 (UTC+10:00)). At the top right of the table, there are buttons for Copy ARN, Empty, Delete, and Create bucket. A search bar at the top of the table allows filtering by bucket name. The bottom of the screen shows the AWS navigation bar with CloudShell, Feedback, Language, and weather information (22°C, Sunny).

Figure 22 S3 bucket called "bucketforphoto" is created.

All objects (photos) in this S3 bucket must become publicly available. The policy below enables public access to all available objects in this S3 bucket.

The screenshot shows the AWS S3 Management console. The left sidebar includes Buckets, Storage Lens, and Feature spotlight. The main area shows the 'Bucket policy' section for the 'bucketforphoto' bucket. It displays a JSON policy document:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::bucketforphoto/*"
    }
  ]
}
```

Below the policy, there are 'Edit' and 'Delete' buttons, and a 'Copy' button. The bottom of the screen shows the AWS navigation bar with CloudShell, Feedback, Language, and weather information (22°C, Sunny).

Figure 23 Policy for public access object in S3

#### 2.2. Photo meta-data in RDS Database

Four logo photos of four universities have been uploaded in s3 bucket and get ready to transfer to phpMyAdmin.

The screenshot shows the AWS S3 console interface. On the left, a sidebar lists various AWS services like Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and IAM Access Analyzer for S3. The main area is titled 'bucketforphoto' and shows a table of objects. The table has columns for Name, Type, Last modified, Size, and Storage class. The objects listed are:

Name	Type	Last modified	Size	Storage class
monashlogo.png	png	September 11, 2023, 20:21:04 (UTC+10:00)	18.2 KB	Standard
rmitlogo.png	png	September 11, 2023, 20:21:05 (UTC+10:00)	57.4 KB	Standard
swinburnelogo.jpg	jpg	September 11, 2023, 20:34:56 (UTC+10:00)	48.8 KB	Standard
unimelblogo.png	png	September 11, 2023, 20:21:08 (UTC+10:00)	1.8 MB	Standard

Figure 24 Upload four photos to S3 "bucketforphoto"

For the next step, the information of each photo must be inserted correctly, and the reference should be pre-signed URL of the photo in the S3 bucket, so the system will know where to get the photo and display it correctly to the website.

The screenshot shows the phpMyAdmin interface. The database selected is 'photos'. The 'photos' table is displayed with the following data:

PhotoTitle	Description	CreationDate	Keywords	References
Swinburne Logo	Logo of Swinburne Uni	2023-09-11	logo, university	<a href="https://bucketforphoto.s3.amazonaws.com/swinburnelogo.jpg">https://bucketforphoto.s3.amazonaws.com/swinburnelogo.jpg</a>
Monash Logo	Logo of Monash	2023-09-11	monash, logo, uni	<a href="https://bucketforphoto.s3.amazonaws.com/monashlogo.jpg">https://bucketforphoto.s3.amazonaws.com/monashlogo.jpg</a>
UniMelb logo	Logo of UniMelb	2023-09-11	UniMelb, logo	<a href="https://bucketforphoto.s3.amazonaws.com/unimelblogo.jpg">https://bucketforphoto.s3.amazonaws.com/unimelblogo.jpg</a>
RMIT Logo	Logo of RMIT	2023-09-11	Logo	<a href="https://bucketforphoto.s3.amazonaws.com/rmitlogo.jpg">https://bucketforphoto.s3.amazonaws.com/rmitlogo.jpg</a>

Figure 25 Successfully upload four photos to phpMyAdmin.

## 2.3. Photo Album website functionality

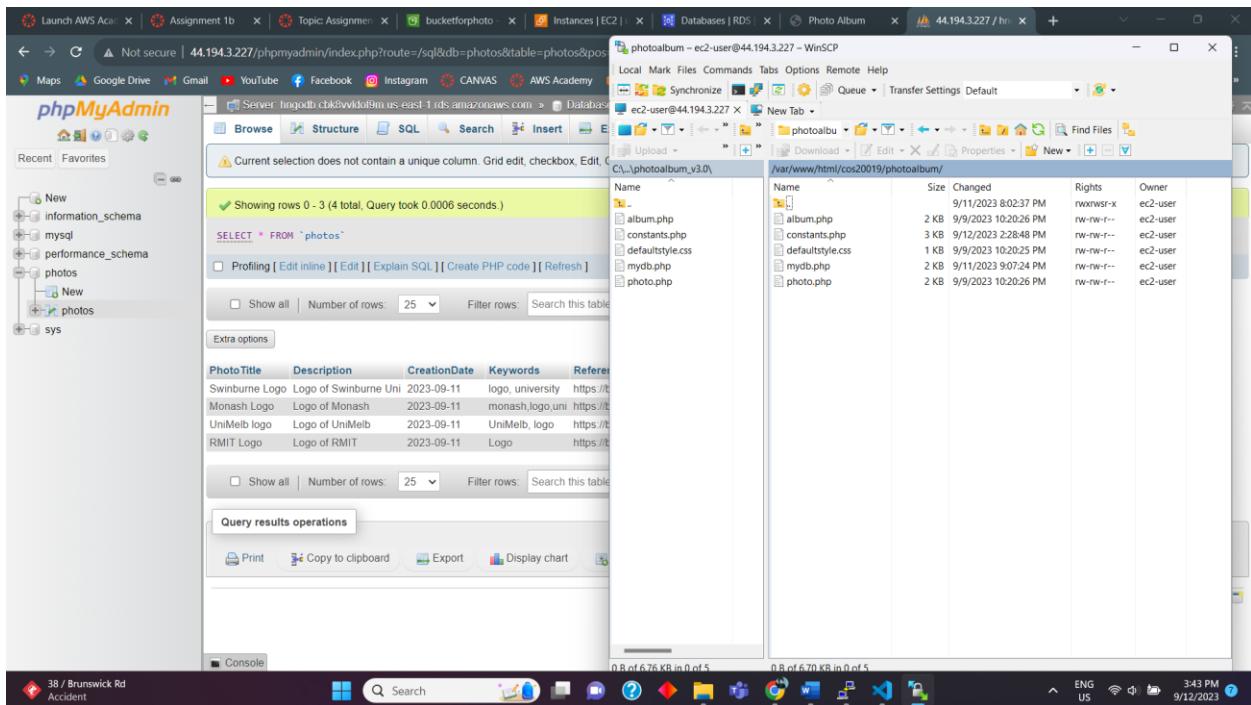
The website must be able to list all the photos (stored in the S3 bucket) along with their meta-data (stored in the database). In this case, I have Modified the constants.php file in the provided code using available information from the S3 bucket and RDS database.

```
File Edit Selection View Go Run Terminal Help ⏪ ⏫ Search
constants.php •
C:\Users\ngoha\OneDrive\Desktop\COS20019-Cloud Computing Architecture\Sandbox\Asm1b> photoalbum_v3.0> photoalbum_v3.0> constants.php

34
35 // [ACTION REQUIRED] your full name
36 define('STUDENT_NAME', 'Hai Nam Ngo');
37 // [ACTION REQUIRED] your Student ID
38 define('STUDENT_ID', '103488515');
39 // [ACTION REQUIRED] your tutorial session
40 define('TUTORIAL_SESSION', 'Friday 04:30PM');
41
42 // [ACTION REQUIRED] name of the s3 bucket that stores images
43 define('BUCKET_NAME', 'bucketforphoto');
44 // [ACTION REQUIRED] region of the above bucket
45 define('REGION', 'us-east-1');
46 // no need to update this const
47 define('S3_BASE_URL', 'https://'.BUCKET_NAME.'.s3.amazonaws.com/');
48
49 // [ACTION REQUIRED] name of the database that stores photo meta-data (note that this is not the DB identifier of the RDS instance)
50 define('DB_NAME', 'photos');
51 // [ACTION REQUIRED] endpoint of RDS instance
52 define('DB_ENDPOINT', 'hngodb.cbkavv1dof9m.us-east-1.rds.amazonaws.com');
53 // [ACTION REQUIRED] username of your RDS instance
54 define('DB_USERNAME', 'username');
55 // [ACTION REQUIRED] password of your RDS instance
56 define('DB_PWD', 'password');
57
58 // [ACTION REQUIRED] name of the DB table that stores photo's meta-data
59 define('DB_PHOTO_TABLE_NAME', 'photos');
60 // The table above has 5 columns:
61 // [ACTION REQUIRED] name of the column in the above table that stores photo's titles
62 define('DB_PHOTO_TITLE_COL_NAME', 'PhotoTitle');
63 // [ACTION REQUIRED] name of the column in the above table that stores photo's descriptions
64 define('DB_PHOTO_DESCRIPTION_COL_NAME', 'Description');
65 // [ACTION REQUIRED] name of the column in the above table that stores photo's creation dates
66 define('DB_PHOTO_CREATEDATE_COL_NAME', 'creationDate');
67 // [ACTION REQUIRED] name of the column in the above table that stores photo's keywords
68 define('DB_PHOTO_KEYWORDS_COL_NAME', 'Keywords');
69 // [ACTION REQUIRED] name of the column in the above table that stores photo's links in S3
70 define('DB_PHOTO_S3REFERENCE_COL_NAME', 'References');
```

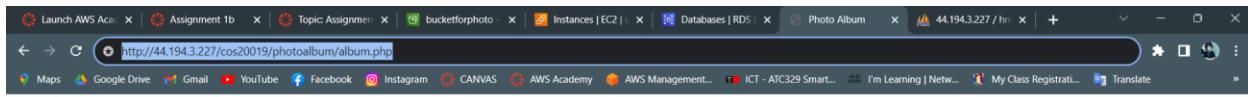
*Figure 26 Modify the given file with correct information.*

Next step is Using WinSCP to upload five sources code for the website.



*Figure 27 Upload source code files to the server*

After all these steps, the website should successfully display four photos with additional information, along with my student information. (The link to access is already given on the first page).



Student name: Hai Nam Ngo

Student ID: 103488515

Tutorial session: Friday 04:30PM

Uploaded photos:

Photo	Name	Description	Creation date	Keywords
	Swinburne Logo	Logo of Swinburne Uni	2023-09-11	logo, university
	Monash Logo	Logo of Monash	2023-09-11	monash,logo,uni
	UniMelb logo	Logo of UniMelb	2023-09-11	UniMelb, logo
	RMIT Logo	Logo of RMIT	2023-09-11	Logo



Figure 28 Final result of album website

#### IV. Testing

This part is created to test if it is possible to ping Bastion/ Web Server Instance from Test Instance.

First, we can put the private key into the Pageant, so when we access Test Instance, it is unnecessary to give the private key again. When we have accessed Test Instance EC2 Linux, we can use “ping” command to ping the Bastion/Web Server Instance. We need to ping the private IPv4 address to make it work. In this case, the ping command will be “ping 10.0.2.91”.

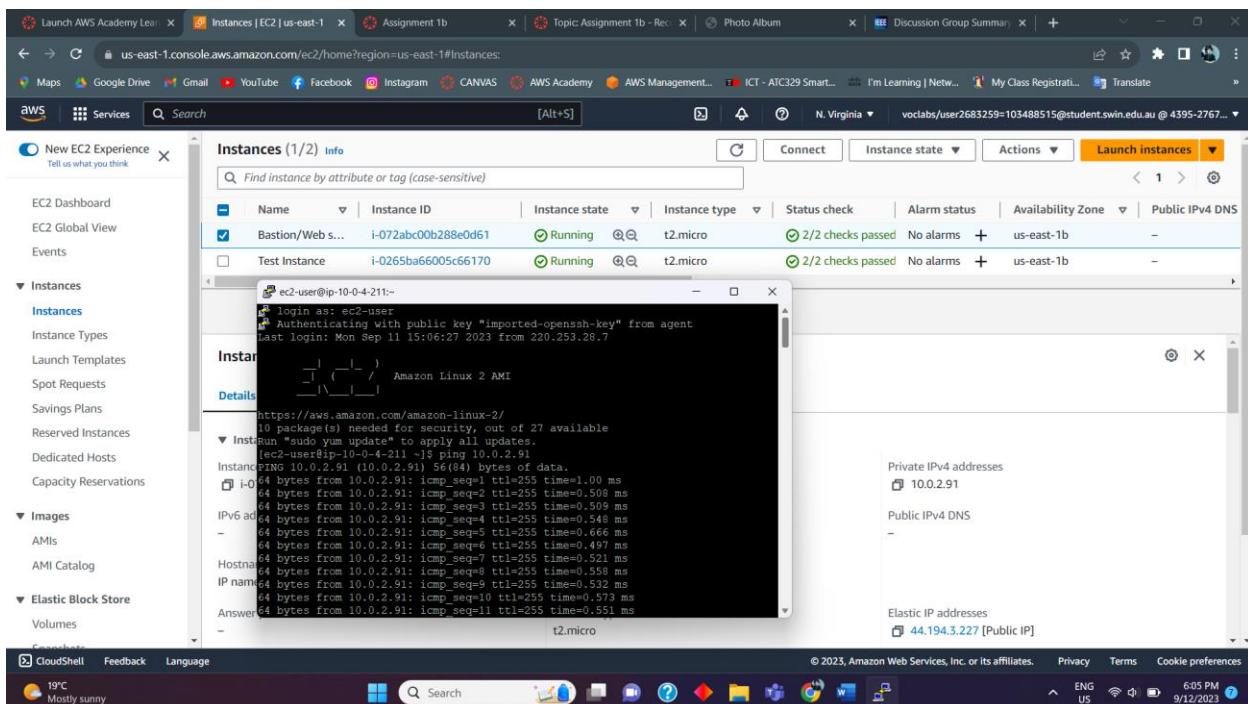


Figure 29 Test Instance successfully ping Bastion/Web Server Instance