Swinburne University of Technology

COS20019 – Cloud Computing Architecture

ASSIGNMENT

1B

Name: Nguyen Linh Dan

Student ID: 103488557

TABLE OF CONTENT

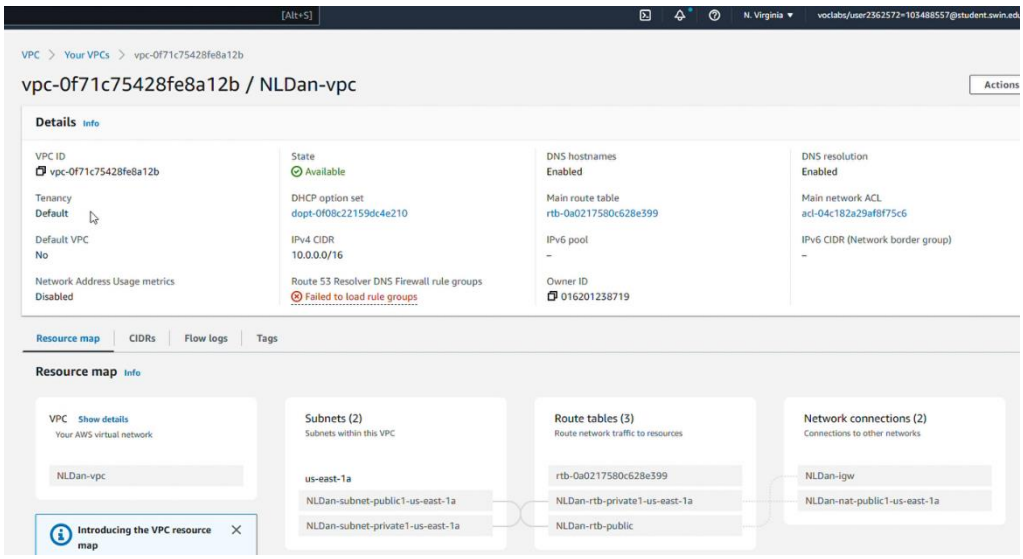
The target of Assignment 1B will be to deploy a simple PHP website on AWS, which requires basic VPC setup, PHP programming language, and database management (MySQL) skills. This report is the summary of all stages that I have done to deploy the Photo Album website on AWS.

1. INFRASTRUCTURE DEPLOYMENT

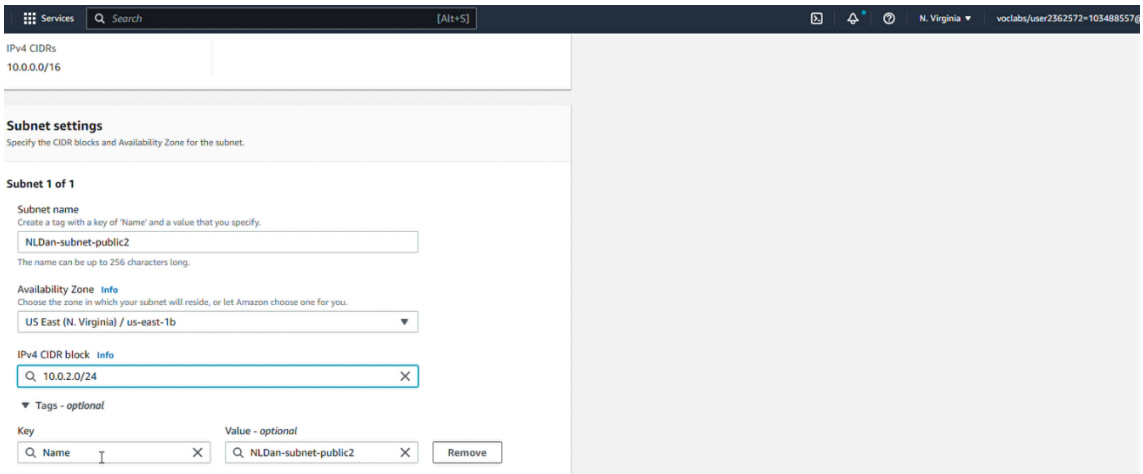
1.1 VPC SETUP

The first step of this assignment is to create VPC and configure 4 subnets (2 public subnets and 2 private subnets). I followed the instructions to use my name for the VPC name, which could make it easier for the instructor to check and confirm this is my work.

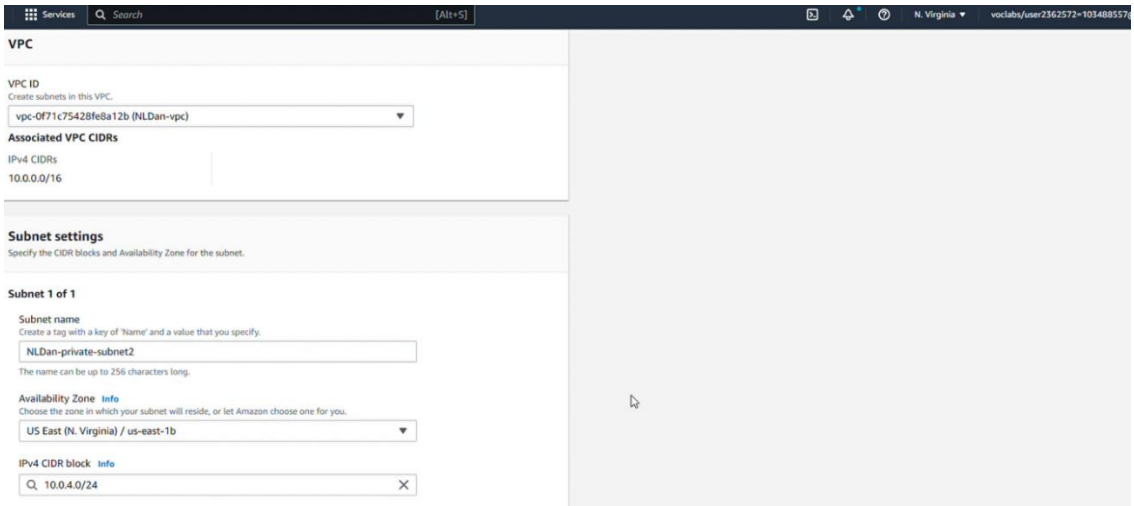
In the Create VPC page, I created one public subnet (10.0.1.0/24) and one private subnet (10.0.3.0/24), these subnets are from region us-east-1a. I will create two more subnets in the next step.



When I finished creating the VPC, I created one more public subnet which uses the **us-east-1b** zone instead of 1a. The IPv4 CIDR block of this subnet was 10.0.2.0/24.

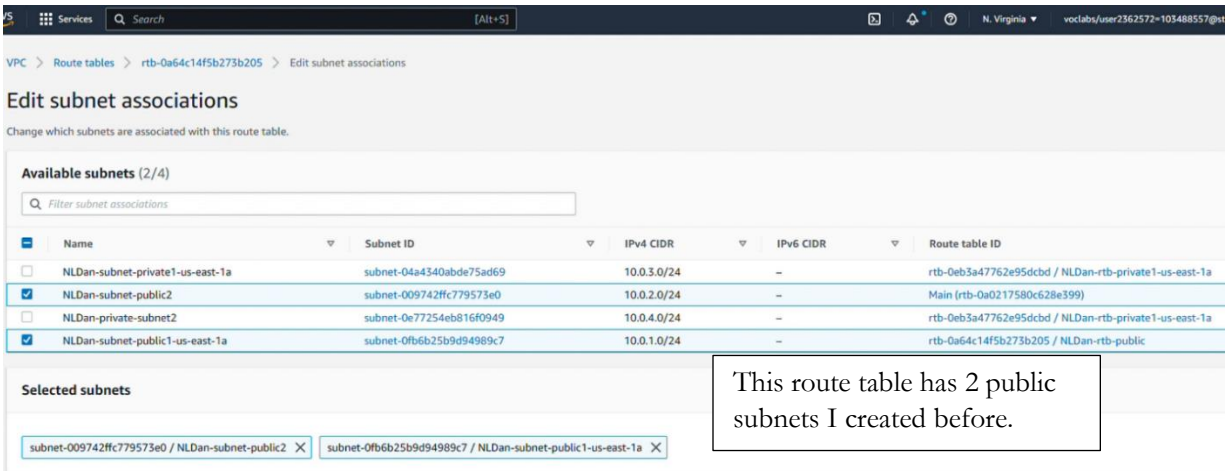


I also created another private subnet in the Availability Zone **us-east-1b** with IPv4 CIDR 10.0.4.0. At this time, I was having 4 subnets to prepare for the subnet associations on the Route Tables page.

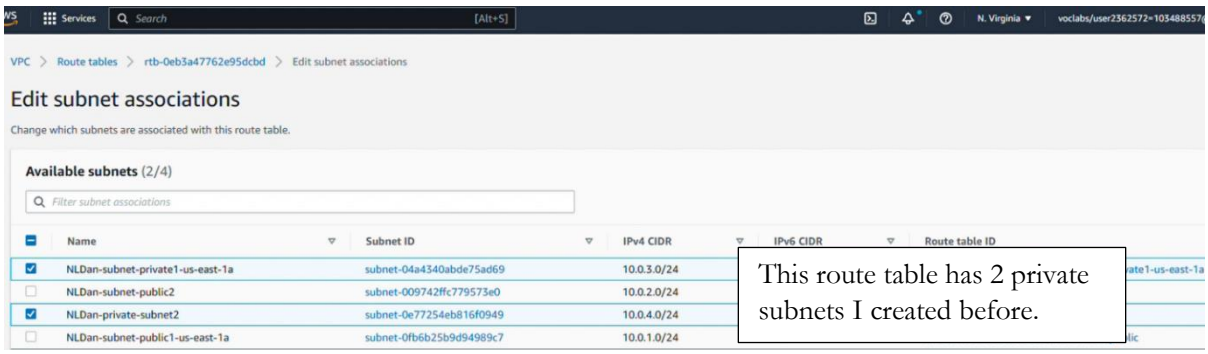


Next, I moved to configure the subnets on the **Route Tables** page. A route table is a collection of rules which are used to determine where your subnet's traffic will be directed to.

First, I configured the public route table which routes to the **Internet Gateway (IGW).**



The remaining 2 private subnets will be put into the route table which routes to the **NAT gateway**. This gateway allows private subnets to connect to the services outside VPC.

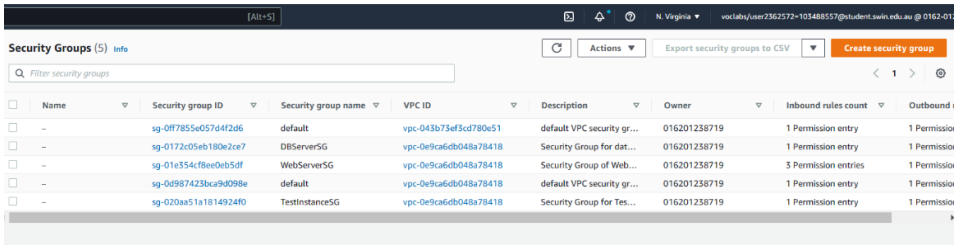


1.2 SECURITY GROUP

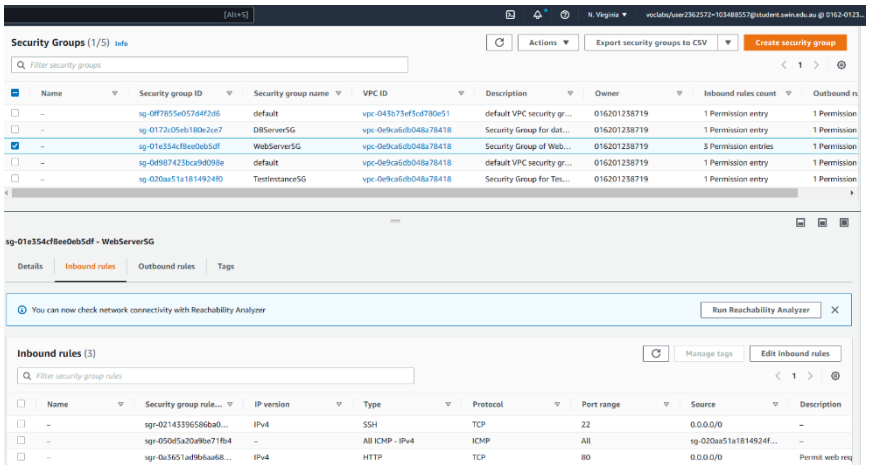
In the second part of this assignment, I created 3 security groups to control the access and functionality of each attached instance.

Below is an overview of 3 security groups and which instance/object is attached.

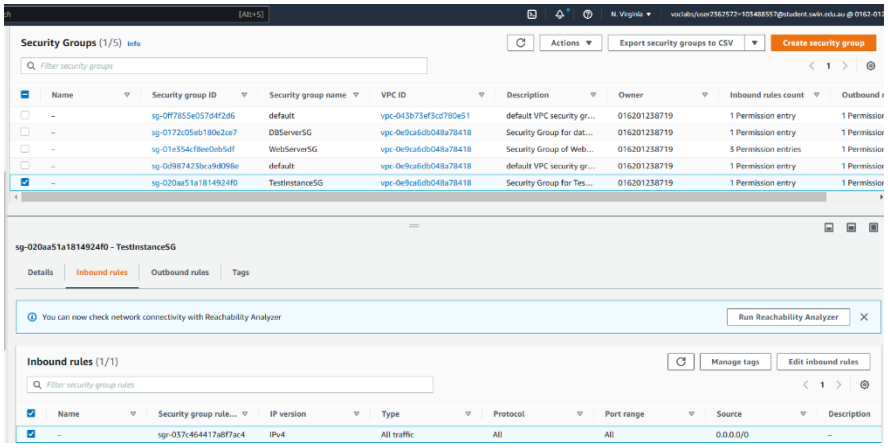
WebServerSG was attached to the **WebServer** instance, **TestInstanceSG** was attached to the **TestInstance**, and **DBServerSG** was attached to the **RDS** database. Please move to the next part if you want to read more about the launching process of these instances and the database.



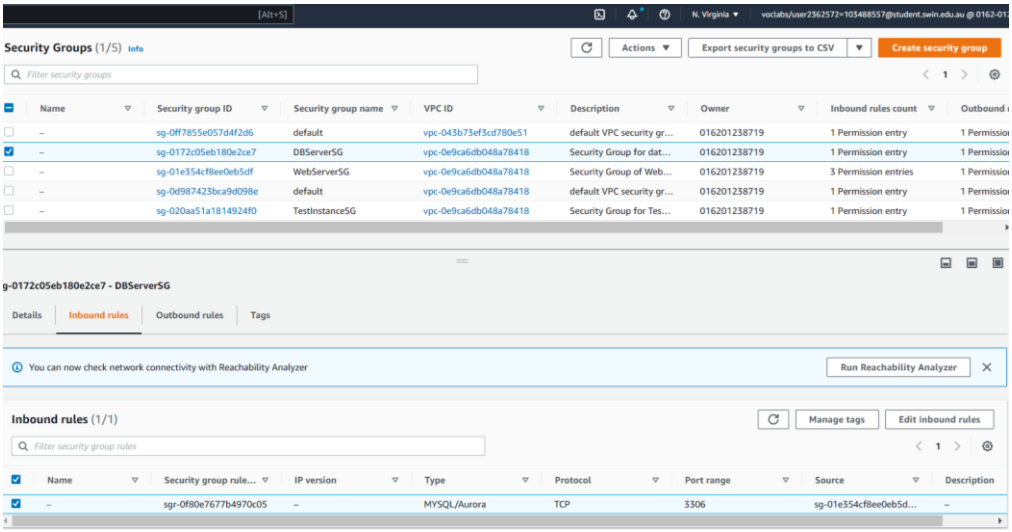
The **WebServerSG** had 3 inbound rules: SSH to allow the SSH login via PuTTY and WinSCP, ICMP to allow the TestInstance to ping the IP address, and HTTP to allow the instance access to the Internet.



TestInstanceSG is the easiest one to configure because it was only used for demonstration purposes, so it allows all traffic from anywhere.



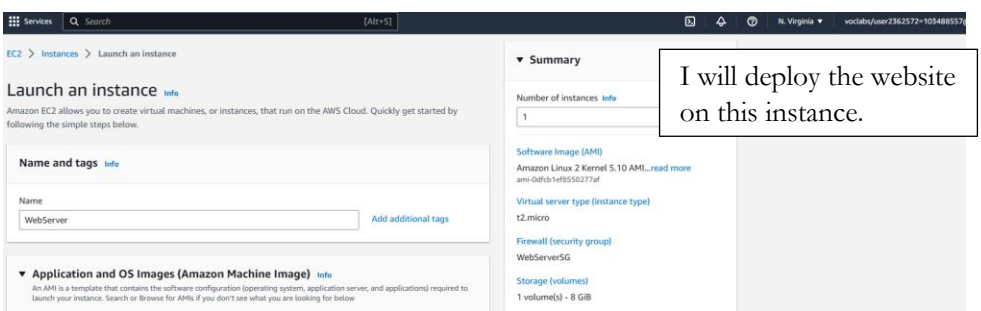
The last Security group was attached to an RDS database. The below security configurations would permit inbound traffic on port 3306 from any EC2 instances associated with **WebServerSG** (I set the source to WebServerSG). I will show how I attached this security group to my database in part 1.4.



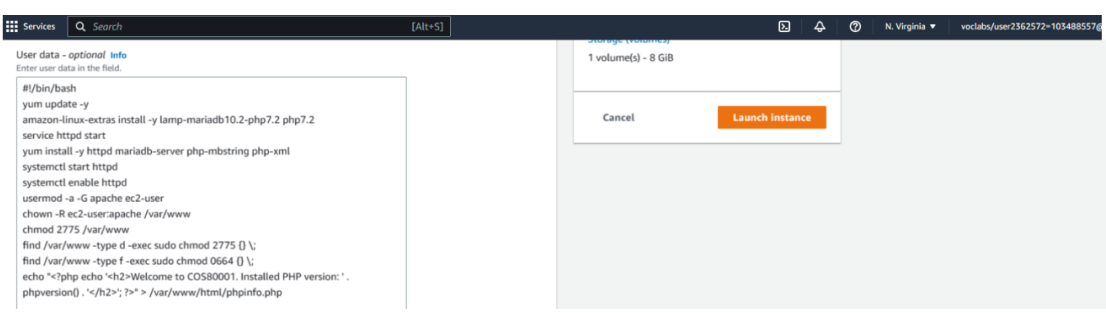
1.3 EC2 VIRTUAL MACHINE

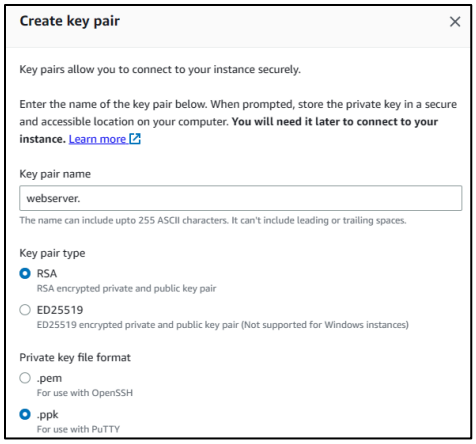
In this part, I will show how I launched two EC2 instances to use in this assignment (one works as the Web Server, and another one will be used for testing the ping command only). I will only explain some necessary parts of the launching process. If I do not mention any parts, that means I kept the default settings for that part.

WebServer Instance



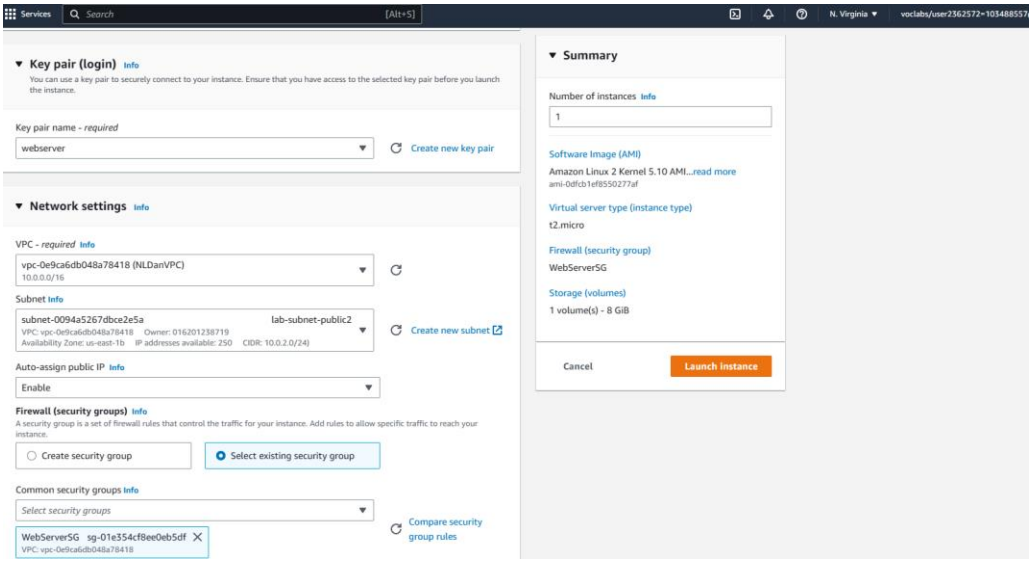
I used the user data script from Assignment 1A to install the Web Server and PHP.





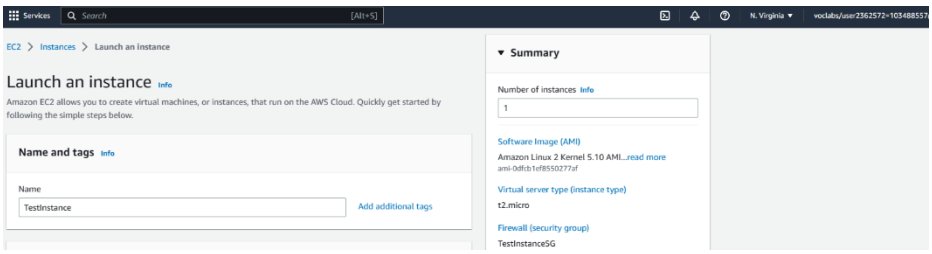
I did not use the default vockey key pair in this assignment, I created my own key pair to make it easier to manage. I created a key pair named 'webserver' and downloaded it with. ppk extension because it will need to use for authorization when using SSH via PuTTY.

I edited quite a lot of things in the **Network Settings** session as I needed to select my VPC and the correct Security group. I will use the second public subnet, which is stored in the us-east-1b zone. I also turned on the auto-assigned public IP (this option should only be enabled when you are creating a public instance, do not enable it when launching a private instance). I deleted the default security group and added the **WebServerSG** I created in the previous step.

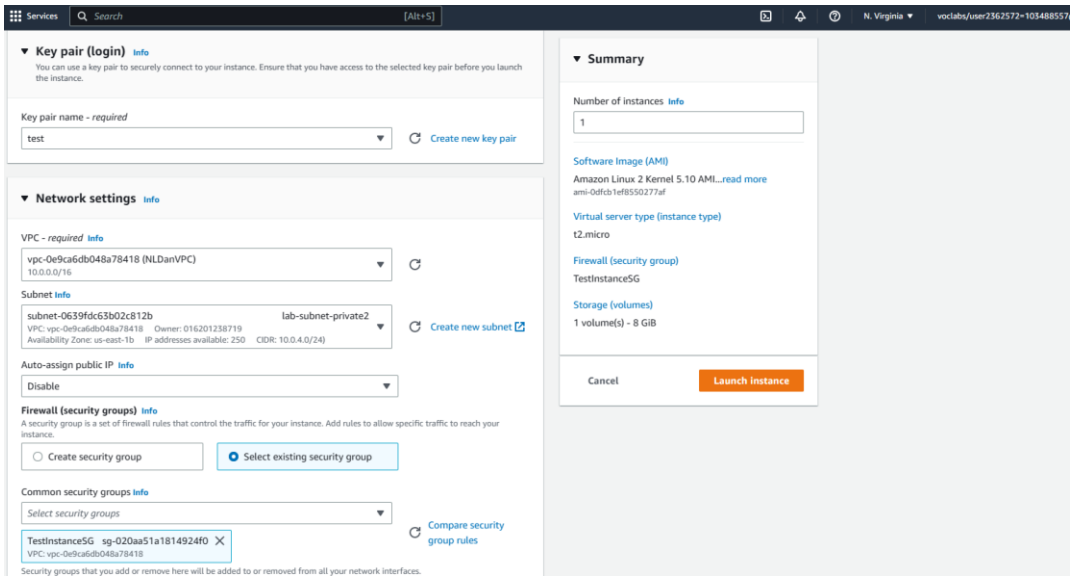


Test Instance

This is a private instance; it will not contribute to the deployment of the PHP website in this assignment. I only used it to test the ping command.



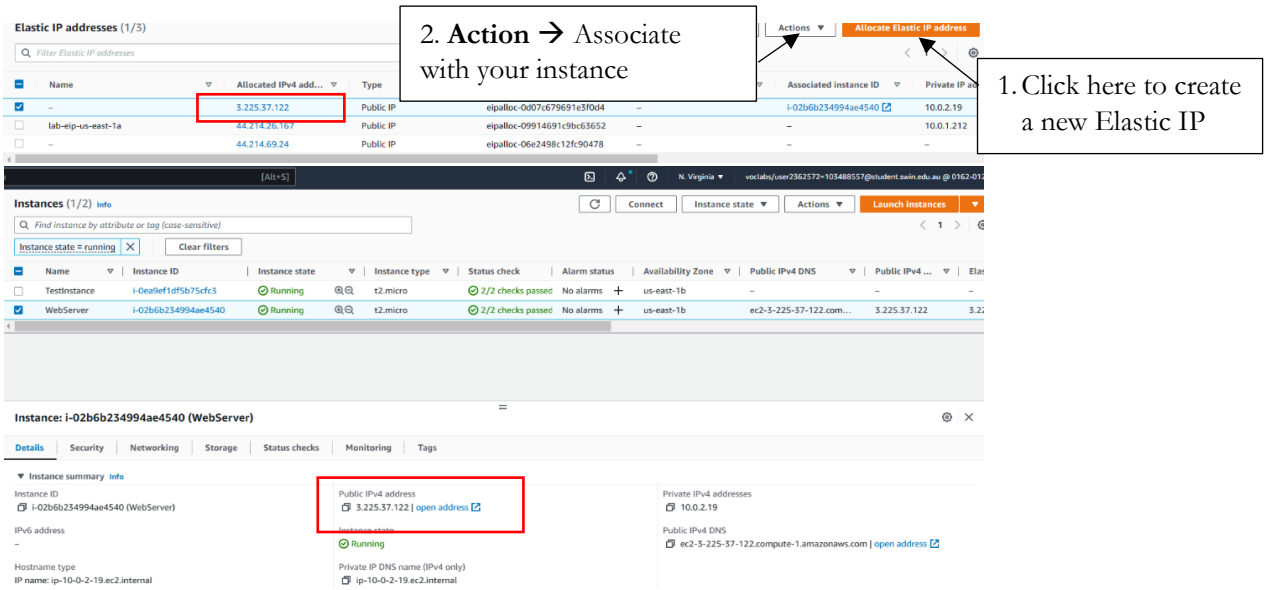
Similar to the WebServer instance, I created a key pair called **test** for this instance. It would work in the private subnet in the **us-east-1b** zone. I attached **TestInstanceSG**, which allows all traffic to this instance. Be careful, I must disable the Auto-assign public IP function to prevent it from becoming a public instance.



I had two ready EC2 instances, I could immediately use after it had finished initializing.

Elastic IP

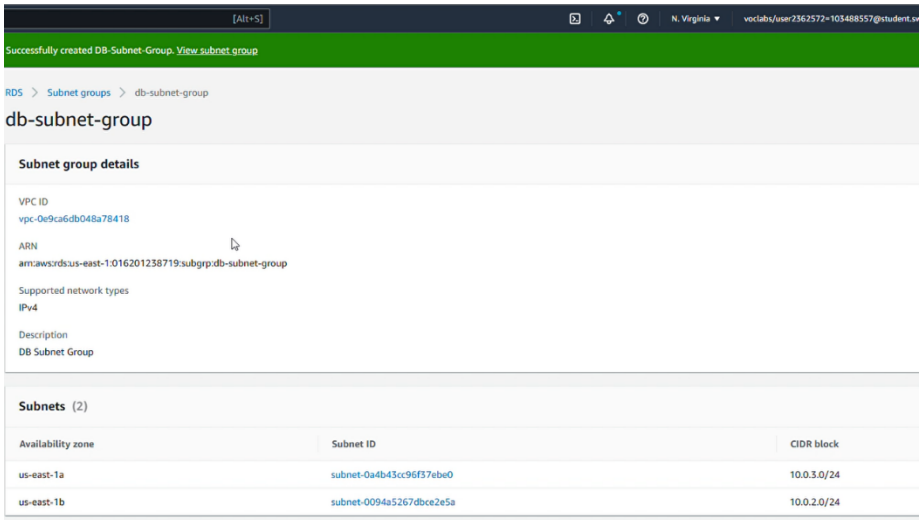
In all previous labs, I noticed that the Public IP address of an instance is dynamic, it will change whenever you refresh the page. Therefore, I needed to create an Elastic IP for it to ensure that the Public IP address is static and unchanged over time. An elastic IP is important if you need to send the URL to someone. If the IP address changes, other people cannot access your website.



1.4 RDS DATABASE INSTANCE

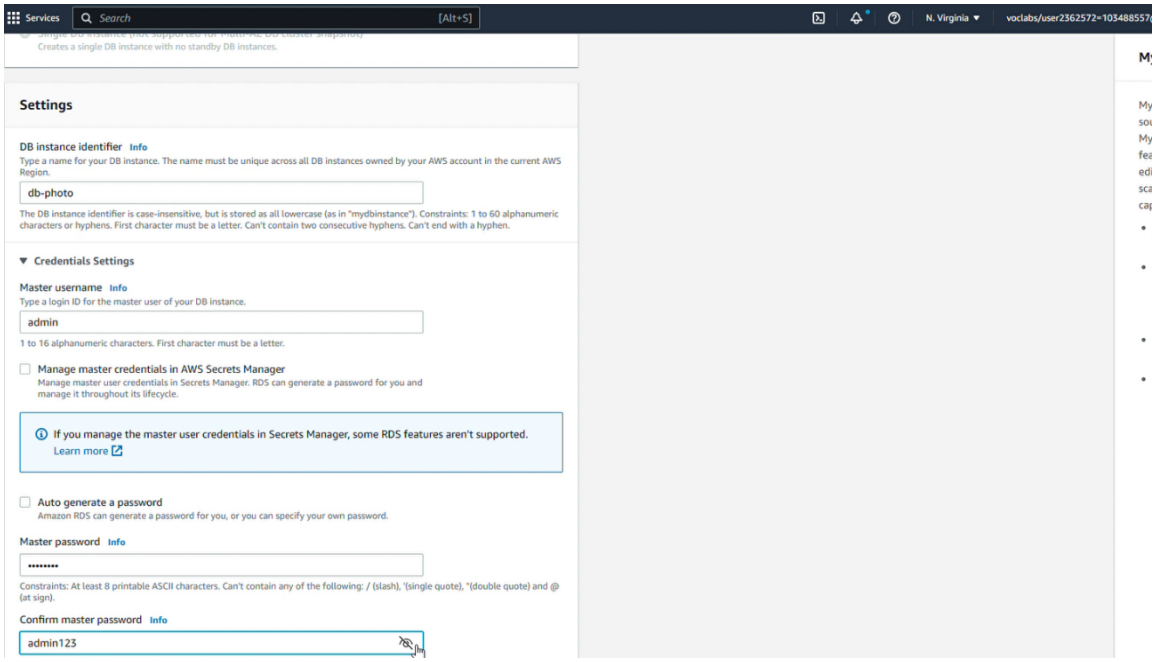
Create a subnet group for RDS

In a DB subnet group, you can specify which VPC and subnets you want to use for your database. The subnet group contains all subnets that you want to use in the RDS. A DB subnet group should contain the subnets from at least 2 Availability Zones.

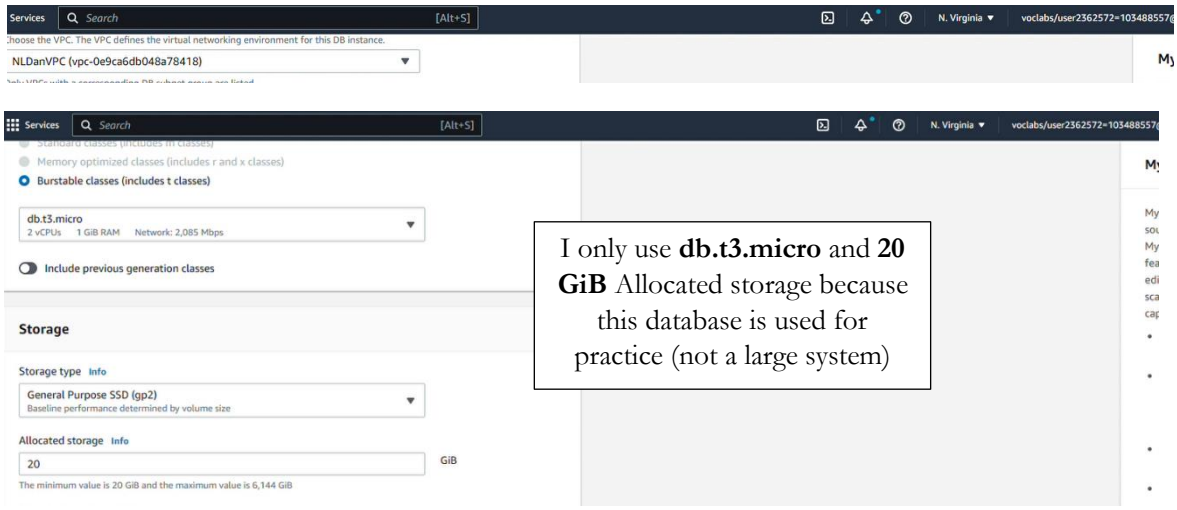


Database setup

I chose **MySQL version 8.0.25** for my database. Below is the **Credentials Settings** session on the database launching page, I defined the username and password for my database. This information is vital, and you must remember it to access your database in the next step.

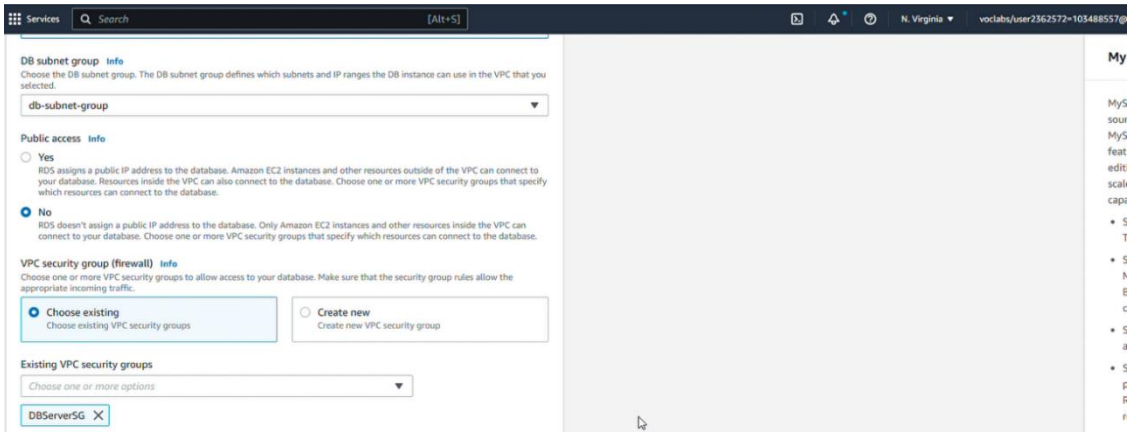


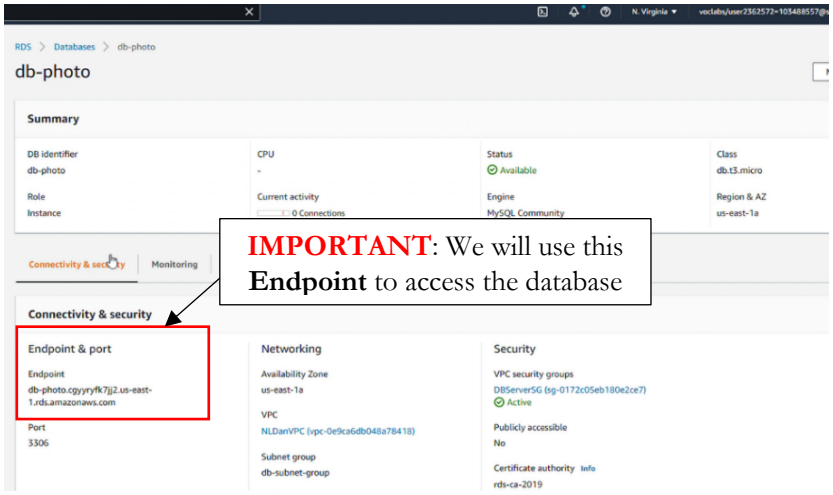
I used the VPC I created in step 1.1 to define the virtual networking environment for my RDS instance.



Next, I configure the **Networking Settings** of this database:

- I used the Subnet Group I created in the previous step
- I did not allow public access to this database
- The Security Group of this database is **DBServerSG**

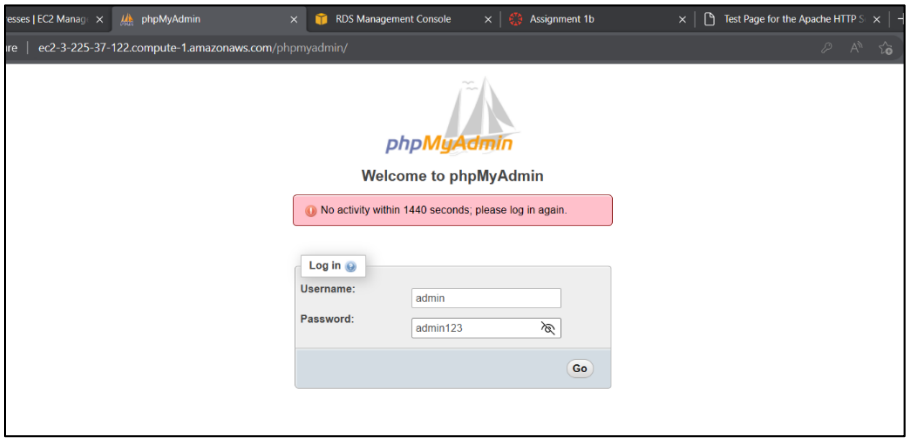




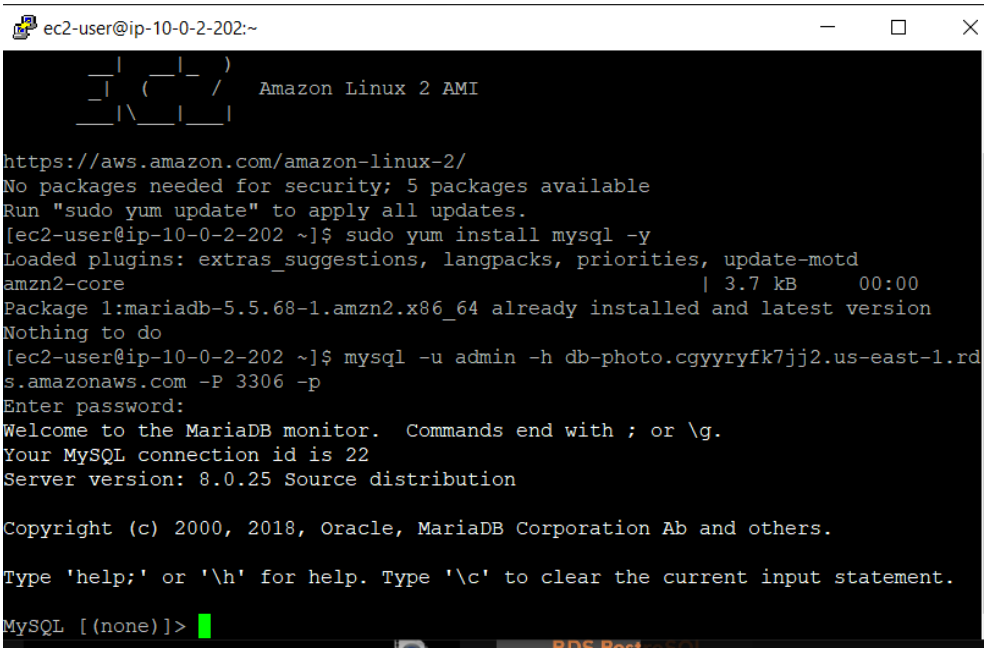
I created the RDS successfully

Install phpMyAdmin

I followed the instructions provided on Canvas to install **phpMyAdmin** which is a web-based tool to manage the database.



Install MySQL using the command and access my database



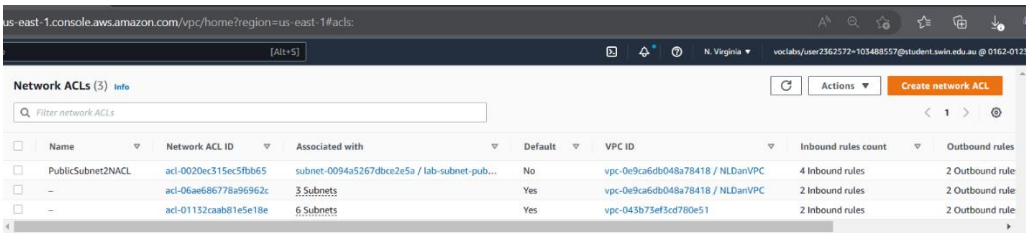
In this part, I only show the installation process of all the necessary tools. Please move to **Part 2** if you would like to check how I used the SQL statement to create a new table and add some metadata for my photos.

1.5 NETWORK ACL

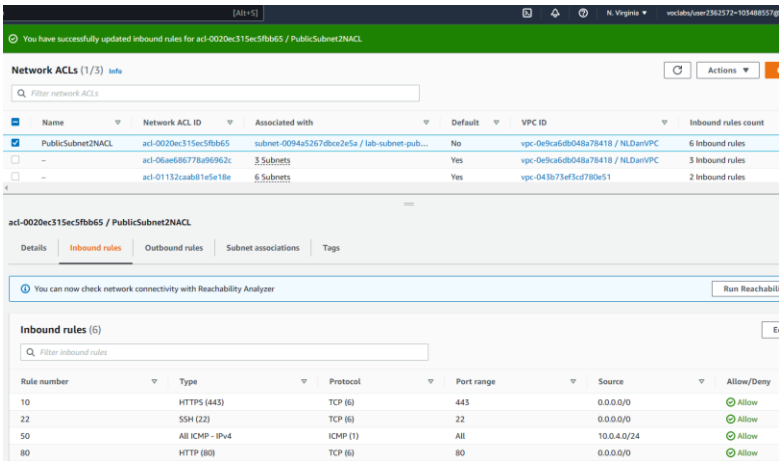
I configured an ACL to add one more security layer for the WebServer instance. Security Groups and ACLs are quite similar to each other. However, the Security Group is attached to the instances, while the ACL is attached to the subnets.

Overview

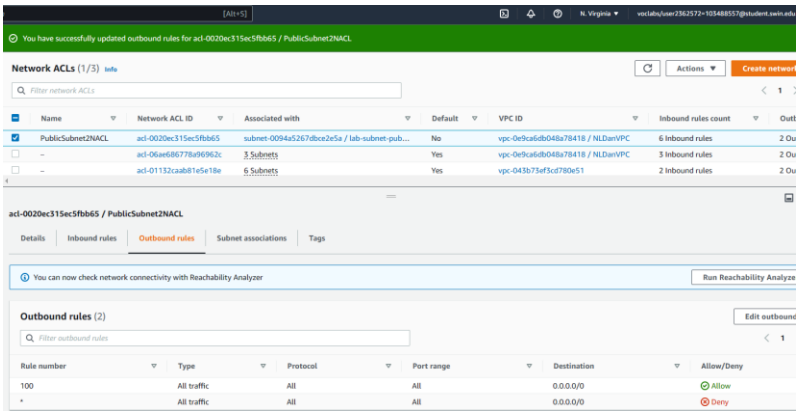
The **PublicSubnet2NACL** will be attached to the public subnet in Availability Zone us-east-1b (WebServer). It will only allow the ICMP traffic from TestInstance. SSH and other rules will also be defined to ensure the website is available anywhere.



Inbound rules



Outbound rules



1.6 HOW TO PING THE WEBSERVER FROM THE TEST INSTANCE

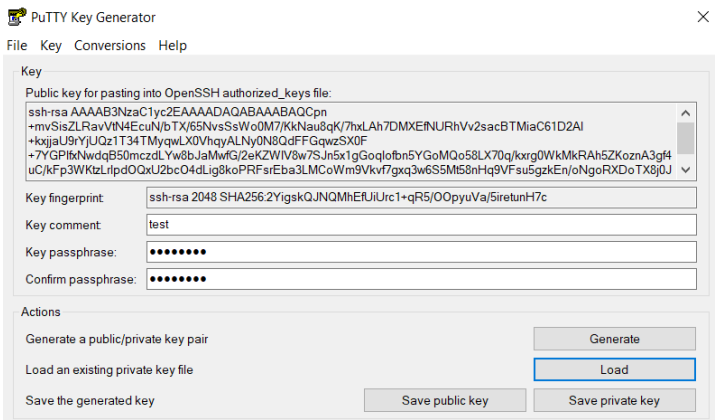
Notice that Test Instance is stored in the private subnet, so it does not have a public IP address. Therefore, we cannot login directly to this instance via PuTTY, I did the following steps to log in to the Test Instance:

Step 1: SSH to the WebServer via PuTTY

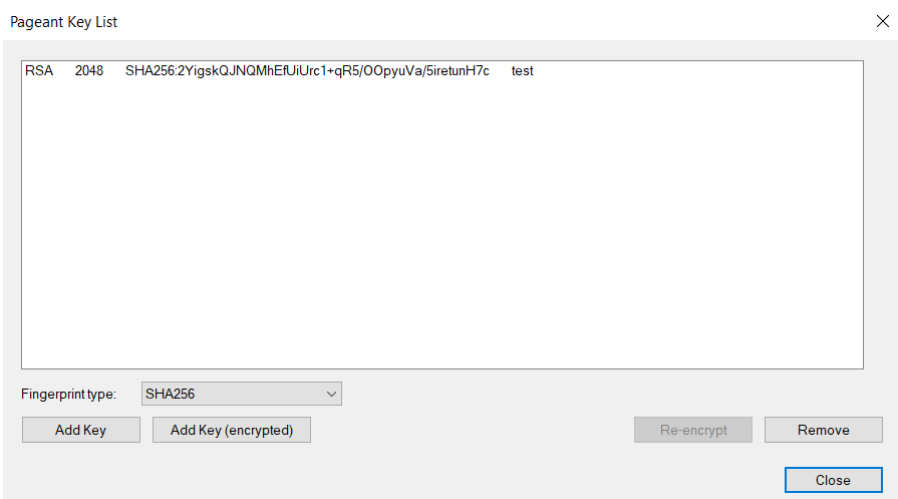
Step 2: SSH to the Test Instance via the command: **ssh ec2-user@private-IP-of-test-instance**

Because the Test Instance also needs a key to authorize access, we can put its key into PuTTY using the following steps:

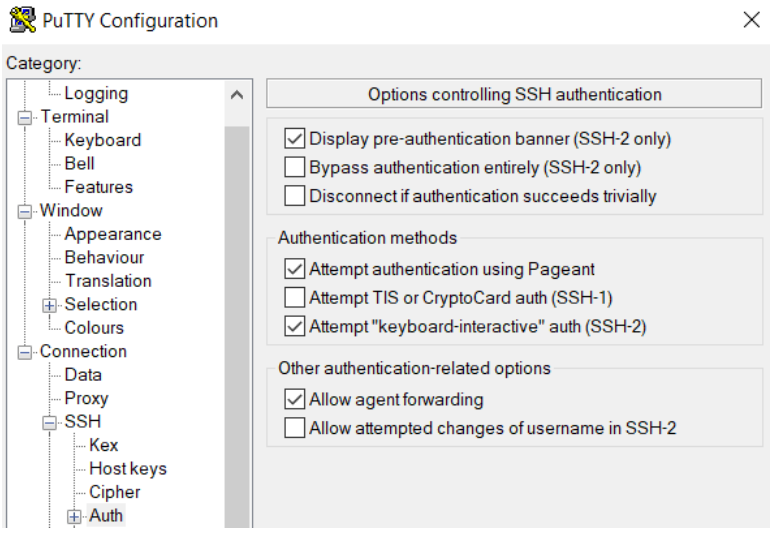
Step 1: Open **PuTTYgen**, load the Test Instance key, and set a new password for it.



Step 2: Open **Pageant** and add the Test Instance key.



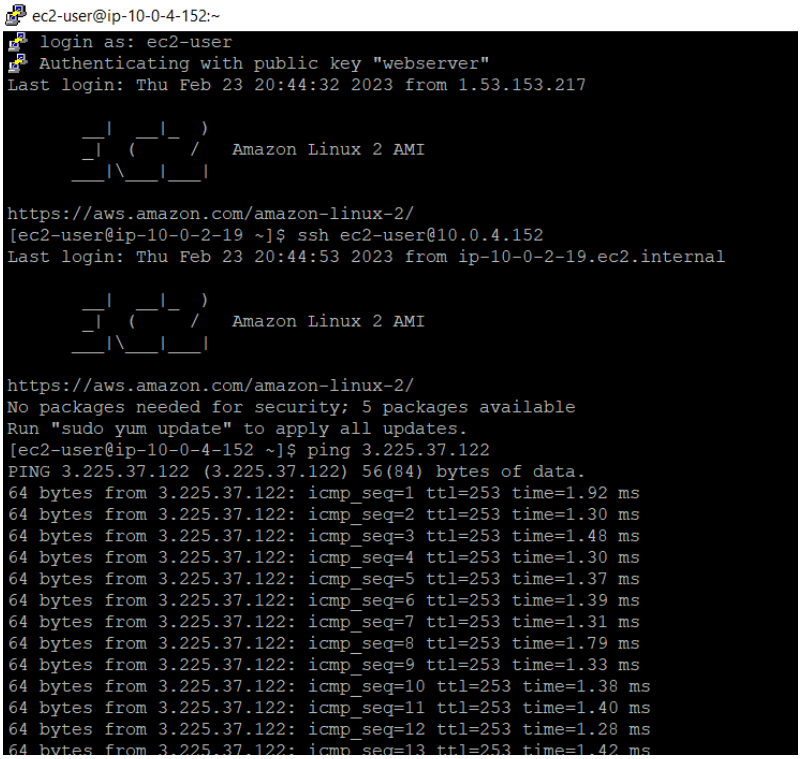
<u>Step 3:</u> Open PuTTY, go to the Auth session → check **Allow agent forwarding**



<u>Step 4:</u> Enter the IP address of the WebServer in the Host Name field, add the authentication key, and normally log in to the WebServer.

Now, we are ready to **ping the WebServer from the Test Instance**

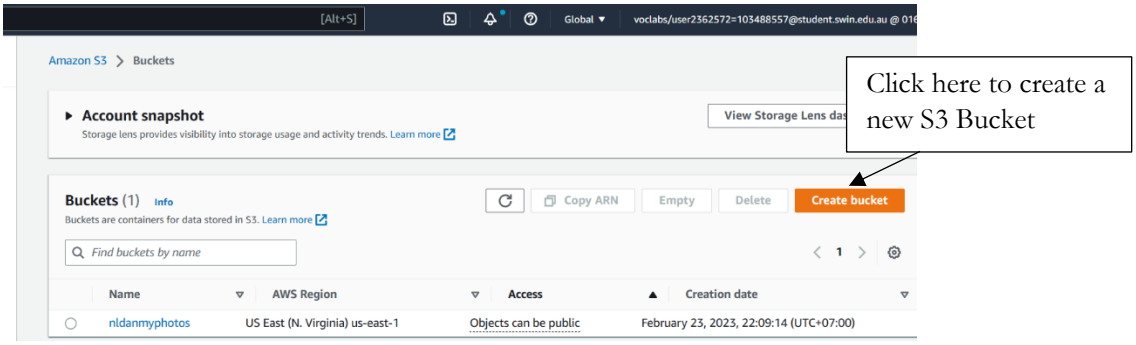
- **Login to the WebServer** (login as: ec2-user)
- **ssh ec2-user@10.0.4.152** (10.0.4.152 is the private IP address of the Test Instance)
- **Ping 3.225.37.122** (3.225.37.122 is the public IP address of the WebServer)



2. PHOTO ALBUM WEBSITE

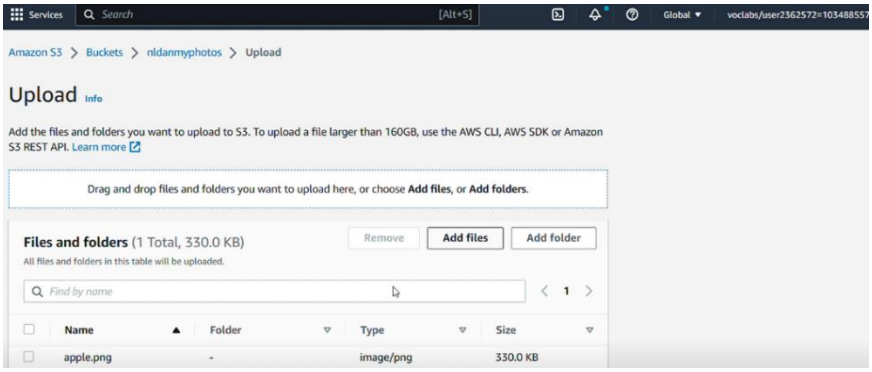
2.1 PHOTO STORAGE

Create a new S3 Bucket

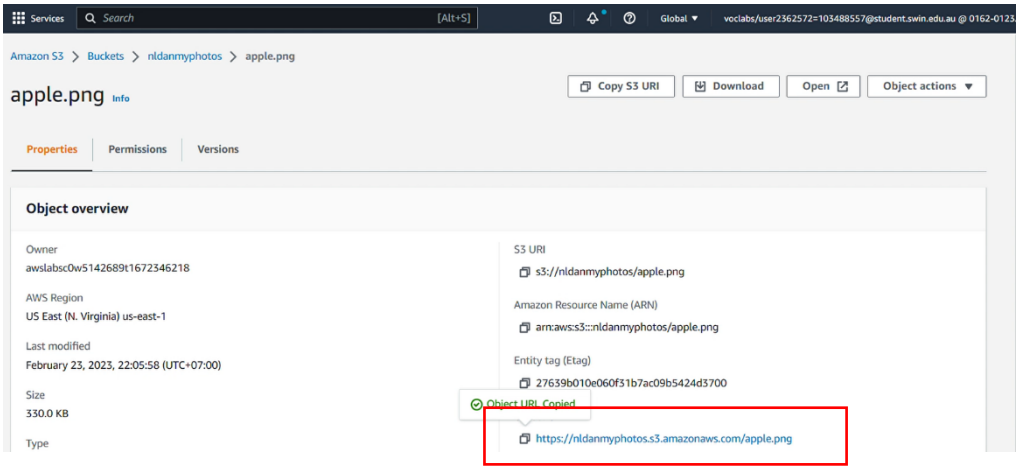


Upload a photo to the S3 Bucket

I uploaded a photo of a red apple to my Bucket. The photos stored here will appear on the Photo Album website with its metadata when I deployed the website successfully.



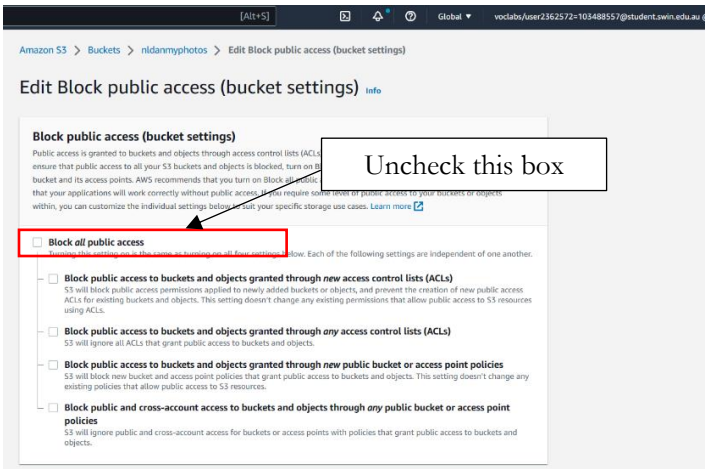
When I clicked on the **.png** file name I had just uploaded, I can see its properties. The highlighted link below will be inserted into the column **'reference'** (**'photos'** table) in my database.



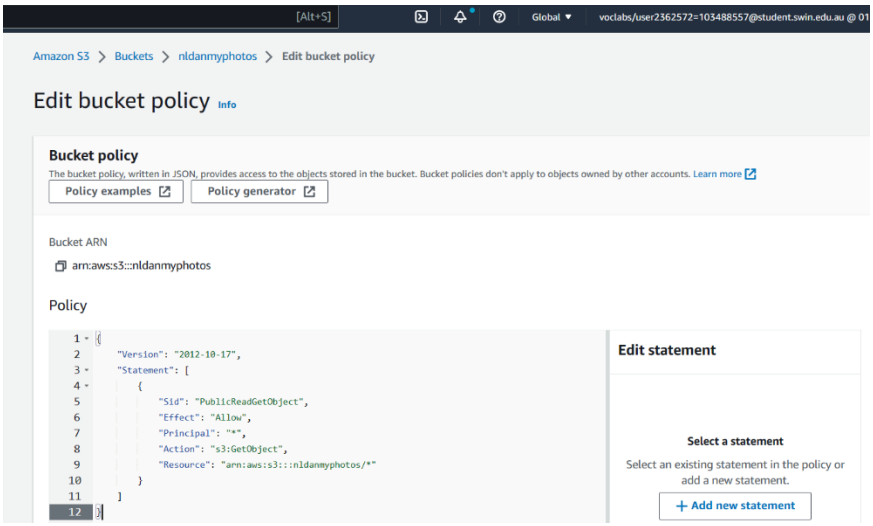
Allow public access to all objects in the S3 Bucket

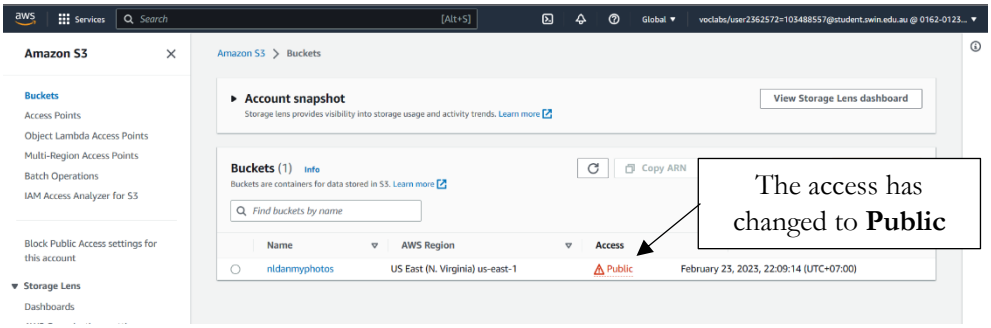
I made some changes in the **Permissions** of the Bucket. I clicked on my **Bucket → Permissions**.

First of all, I turned off the **Block all public access** to allow public access from anywhere.



Next, I edited the **Bucket policy**. The below script is used to allow all public access to the resources in the Bucket. I found this script on the AWS Documentation page.





2.2 PHOTO METADATA IN RDS DATABASE

Access to the database

I installed MySQL in step 1.3, and now I can access my database and work with it through the command: **mysql -u [username] -h [endpoint] -P 3306 -p**

```
ec2-user@ip-10-0-2-19:~
login as: ec2-user
Authenticating with public key "webserver"
Last login: Fri Feb 24 05:11:52 2023 from 1.53.153.217

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-2-19 ~]$ mysql -u admin -h db-photo.cgyyryfk7jj2.us-east-1.rds
.amazonaws.com -P 3306 -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 354
Server version: 8.0.25 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Create a new table

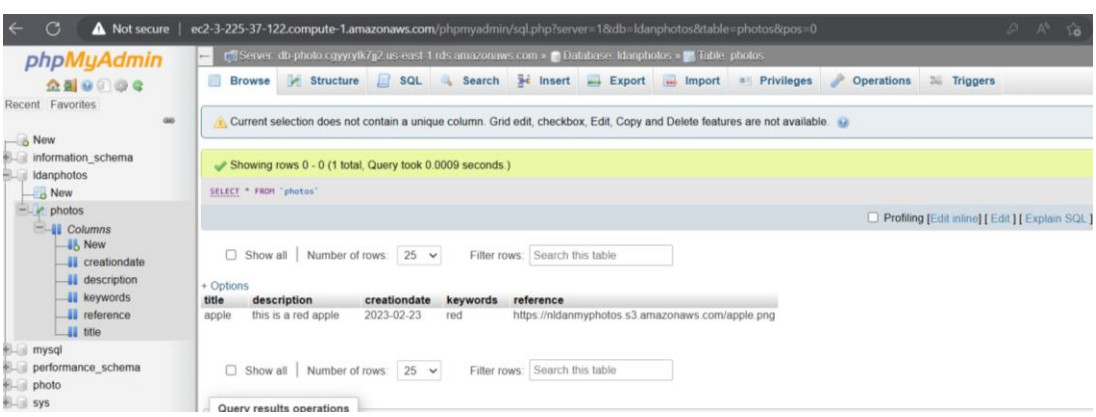
The SQL statements to create a new table, view tables, or insert new rows looks like you are working with a normal SQL database (with UI).

```
MySQL [ldanphotos]> create table photos (title varchar(255),description varchar2
(255), creationdate date, keywords varchar(255), reference varchar(255));
```

Insert a new row (the metadata of the photo I uploaded to S3 Bucket)

```
MySQL [ldanphotos]> insert into photos (title, description, creationdate, keywords, reference) values ('apple','this is a red apple','20
om/apple.png');
Query OK, 1 row affected (0.00 sec)

MySQL [ldanphotos]> select * from photos;
+-------+---------------------+--------------+----------+-----------------------------------------------+
| title | description         | creationdate | keywords | reference                                     |
+-------+---------------------+--------------+----------+-----------------------------------------------+
| apple | this is a red apple | 2023-02-23   | red      | https://nldanmyphotos.s3.amazonaws.com/apple.png |
+-------+---------------------+--------------+----------+-----------------------------------------------+
1 row in set (0.01 sec)
```

View my photos table via phpMyAdmin



2.3 PHOTO ALBUM WEBSITE FUNCTIONALITY

Modify the constant.php file

All necessary functional PHP files are provided to us on Canvas. I was only required to edit some information in the **constant.php** file to make it consistent with the resources name I created on AWS. Students will have different names for their database and their S3 Bucket, so we must modify the data ourselves.

Some data fields I edited:

- My personal information (Name, Student ID)
- Name of my S3 Bucket and its link
- Name of my RDS database and its endpoint
- Name of my metadata table

***Note:** I created the same column name specified in the **constant.php** file, so I did not make any changes to the column names.

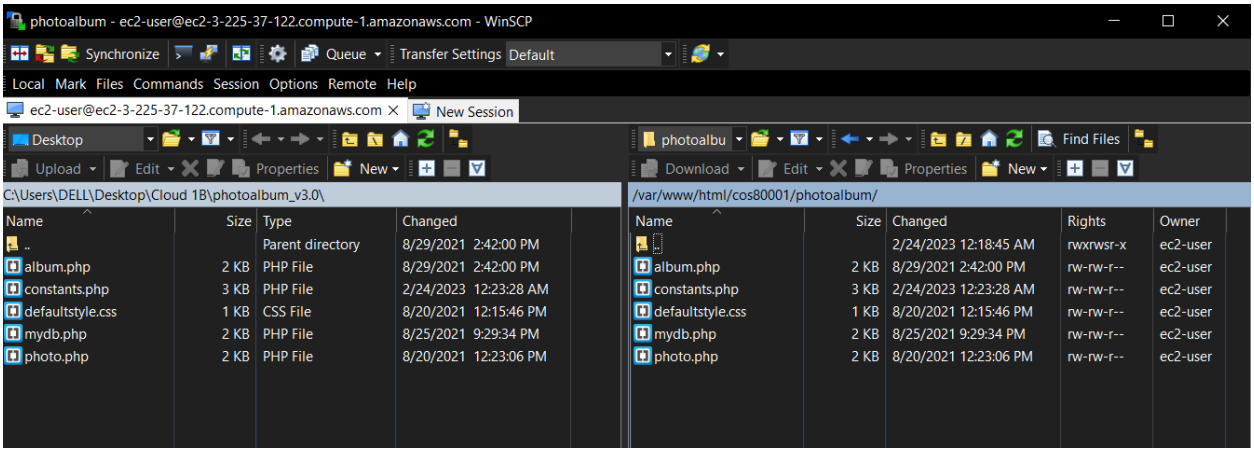
```
// [ACTION REQUIRED] your full name
define('STUDENT_NAME', 'Nguyen Linh Dan');
// [ACTION REQUIRED] your Student ID
define('STUDENT_ID', '103488557');
// [ACTION REQUIRED] your tutorial session
define('TUTORIAL_SESSION', 'Saturday 09:15AM');

// [ACTION REQUIRED] name of the S3 bucket that stores images
define('BUCKET_NAME', 'nldanmyphotos');
// [ACTION REQUIRED] region of the above bucket
define('REGION', 'us-east-1');
// no need to update this const
define('S3_BASE_URL','https://'.BUCKET_NAME.'.s3.amazonaws.com/');

// [ACTION REQUIRED] name of the database that stores photo meta-data (note
this is not the DB identifier of the RDS instance)
define('DB_NAME', 'ldanphotos');
// [ACTION REQUIRED] endpoint of RDS instance
define('DB_ENDPOINT', 'db-photo.cgyyryfk7jj2.us-east-1.rds.amazonaws.com');
// [ACTION REQUIRED] username of your RDS instance
define('DB_USERNAME', 'admin');
// [ACTION REQUIRED] password of your RDS instance
define('DB_PWD', 'admin123');
```

```
// [ACTION REQUIRED] name of the DB table that stores photo's meta-data
define('DB_PHOTO_TABLE_NAME', 'photos');
// The table above has 5 columns:
// [ACTION REQUIRED] name of the column in the above table that stores photo's
titles
define('DB_PHOTO_TITLE_COL_NAME', 'title');
// [ACTION REQUIRED] name of the column in the above table that stores photo's
descriptions
define('DB_PHOTO_DESCRIPTION_COL_NAME', 'description');
// [ACTION REQUIRED] name of the column in the above table that stores photo's
creation dates
define('DB_PHOTO_CREATIONDATE_COL_NAME', 'creationdate');
// [ACTION REQUIRED] name of the column in the above table that stores photo's
keywords
define('DB_PHOTO_KEYWORDS_COL_NAME', 'keywords');
// [ACTION REQUIRED] name of the column in the above table that stores photo's
links in S3
define('DB_PHOTO_S3REFERENCE_COL_NAME', 'reference');
```

Upload PHP files to WinSCP

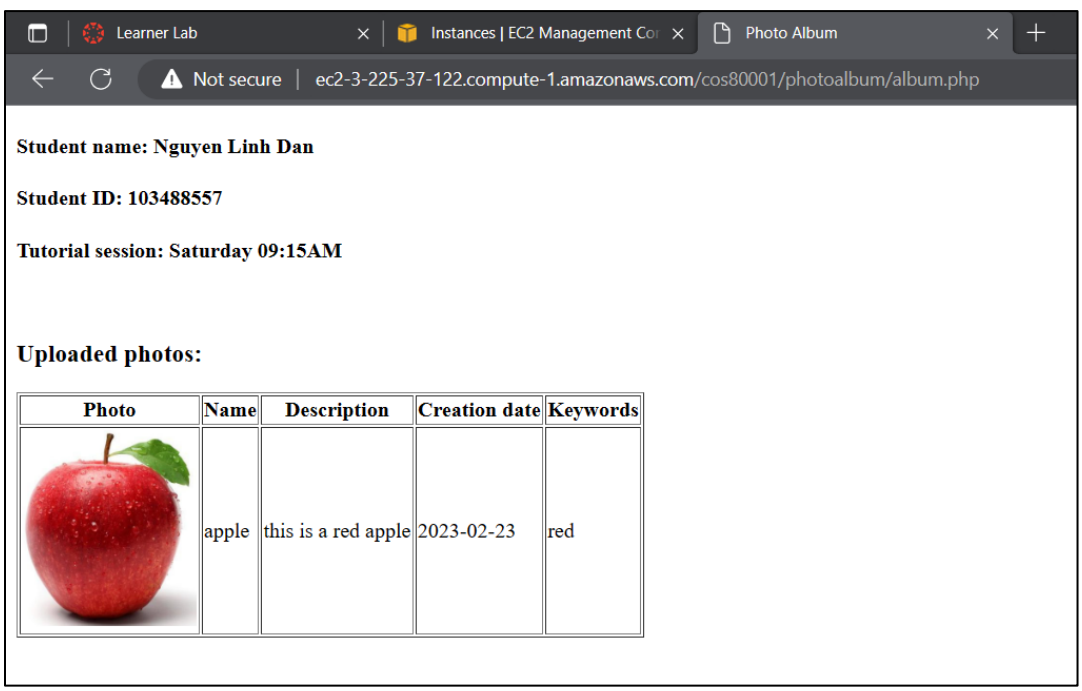
Similar to Assignment 1A, I navigated to **/var/www/html/** directory and created some folders following the instructions.

All my PHP file was copied and pasted into the directory **/var/www/html/cos80001/photoalbum.**



View the website

I successfully deployed my website on AWS. You can see the website by navigating to **album.php**.

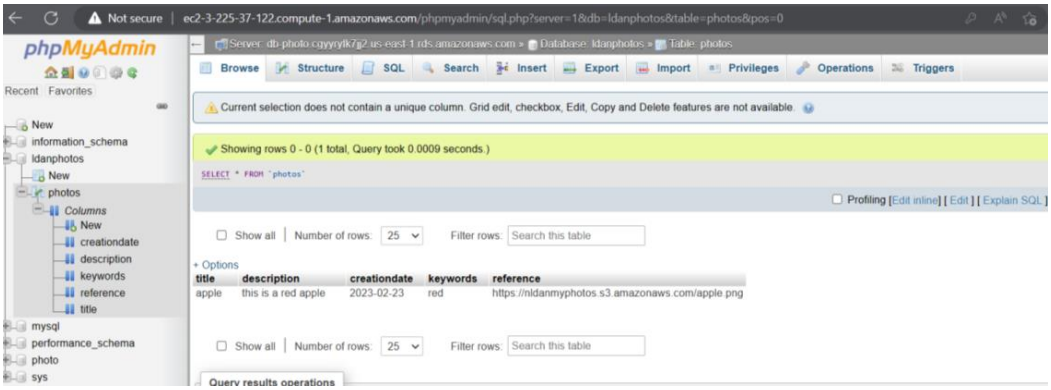


3. URL & ADDITIONAL SCREENSHOTS

URL: http://ec2-3-225-37-122.compute-1.amazonaws.com/cos80001/photoalbum/album.php

Additional Screenshots

Data records in the database




```
MySQL [ldanphotos]> select * from photos;
+-------+--------------------+--------------+----------+-----------------------------------------------+
| title | description        | creationdate | keywords | reference                                     |
+-------+--------------------+--------------+----------+-----------------------------------------------+
| apple | this is a red apple | 2023-02-23   | red      | https://nldanmyphotos.s3.amazonaws.com/apple.png |
+-------+--------------------+--------------+----------+-----------------------------------------------+
1 row in set (0.01 sec)
```

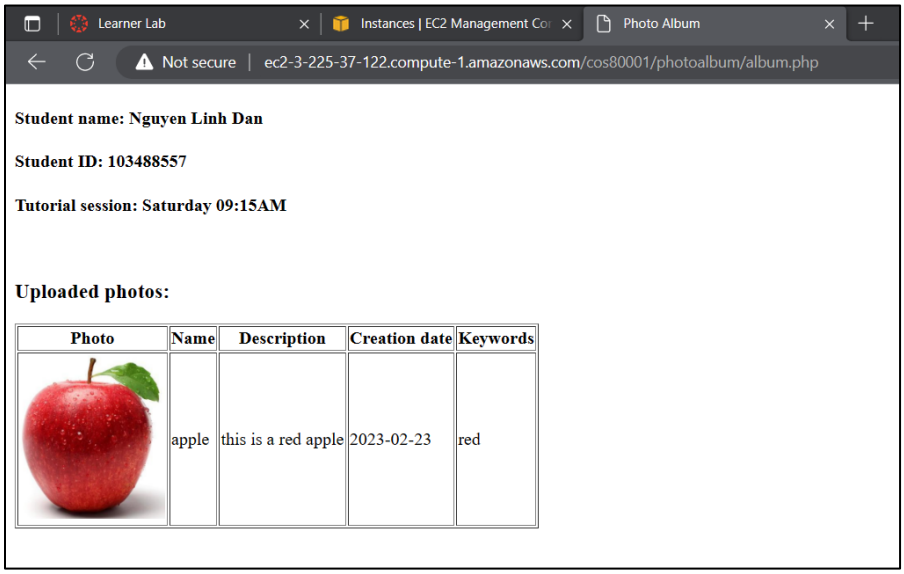
Ping the WebServer from Test Instance

ec2-user@ip-10-0-4-152:~

```
login as: ec2-user
Authenticating with public key "webserver"
Last login: Thu Feb 23 20:44:32 2023 from 1.53.153.217

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-2-19 ~]$ ssh ec2-user@10.0.4.152
Last login: Thu Feb 23 20:44:53 2023 from ip-10-0-2-19.ec2.internal

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
No packages needed for security; 5 packages available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-0-4-152 ~]$ ping 3.225.37.122
PING 3.225.37.122 (3.225.37.122) 56(84) bytes of data.
64 bytes from 3.225.37.122: icmp_seq=1 ttl=253 time=1.92 ms
64 bytes from 3.225.37.122: icmp_seq=2 ttl=253 time=1.30 ms
64 bytes from 3.225.37.122: icmp_seq=3 ttl=253 time=1.48 ms
64 bytes from 3.225.37.122: icmp_seq=4 ttl=253 time=1.30 ms
64 bytes from 3.225.37.122: icmp_seq=5 ttl=253 time=1.37 ms
64 bytes from 3.225.37.122: icmp_seq=6 ttl=253 time=1.39 ms
64 bytes from 3.225.37.122: icmp_seq=7 ttl=253 time=1.31 ms
64 bytes from 3.225.37.122: icmp_seq=8 ttl=253 time=1.79 ms
64 bytes from 3.225.37.122: icmp_seq=9 ttl=253 time=1.33 ms
64 bytes from 3.225.37.122: icmp_seq=10 ttl=253 time=1.38 ms
```

Final Result

On Desktop



On Mobile Phone