COS20019

Cloud Computing Architechture

Assignement 1 – Part B

Creating and Deploying Photo Album website onto a simple AWS Infrastructure

Truong Ngoc Gia Hieu
105565520

Due date: **by end of Week 6**

Prerequisite requirements:

- **Submitted** Assignment 1A. Please note that Assignment 1B submissions will **not** be marked if the student **has not submitted Assignment 1A**. Contact your instructor immediately if this requirement is not met in your grades.

Preparation:

- ACF Labs 2, 3, 4 and 5.
- Know how to set up and manage a MySQL database on Cloud
- General understanding of PHP programming language.
- Know how to set up and manage a Web accessible S3 bucket

All supporting materials mentioned in this document can be found in the corresponding assignment page on Canvas.

The PHP source code has been provided for this assignment. However, you will need to understand how this code works to be able to modify the missing parts. Each student is supposed to add their own specific information in this code; hence, you must not copy someone else's code.

Objectives

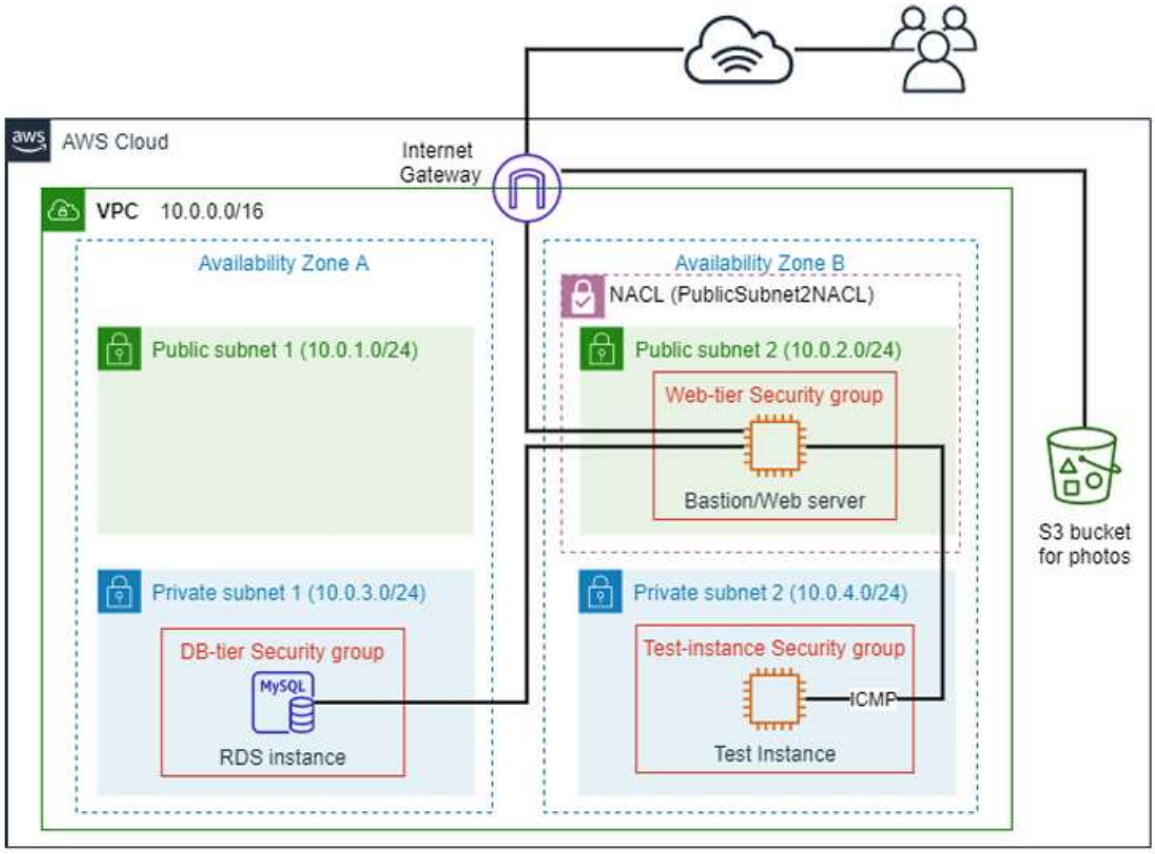
This assignment has the following objectives:

1. Create a secure Virtual Private Cloud (VPC) with subnets, routing tables and security groups.
2. Control access to and from your VPC via an Internet Gateway.
3. Modify the provided PHP code to create a website that stores meta-data information about photos uploaded to S3 in a MySQL database managed by Amazon RDS. The website should enable the user to search for and display photos using meta-data.
4. Deploy and test your PHP web site on an Apache web server running on an EC2 virtual machine instance.
5. Add an additional layer of security by applying a Network ACL to the public subnet that hosts your web server.

Part 1. Infrastructure deployment

You will set up a VPC with the structure and services as illustrated in the diagram below.

NOTE: Do not use the default VPC. All services should be in your custom VPC. Below are the detailed requirements for each service.



Step 1.1: Generating the VPC

Step 1.1.1: First and foremost, search and select the **VPC.**

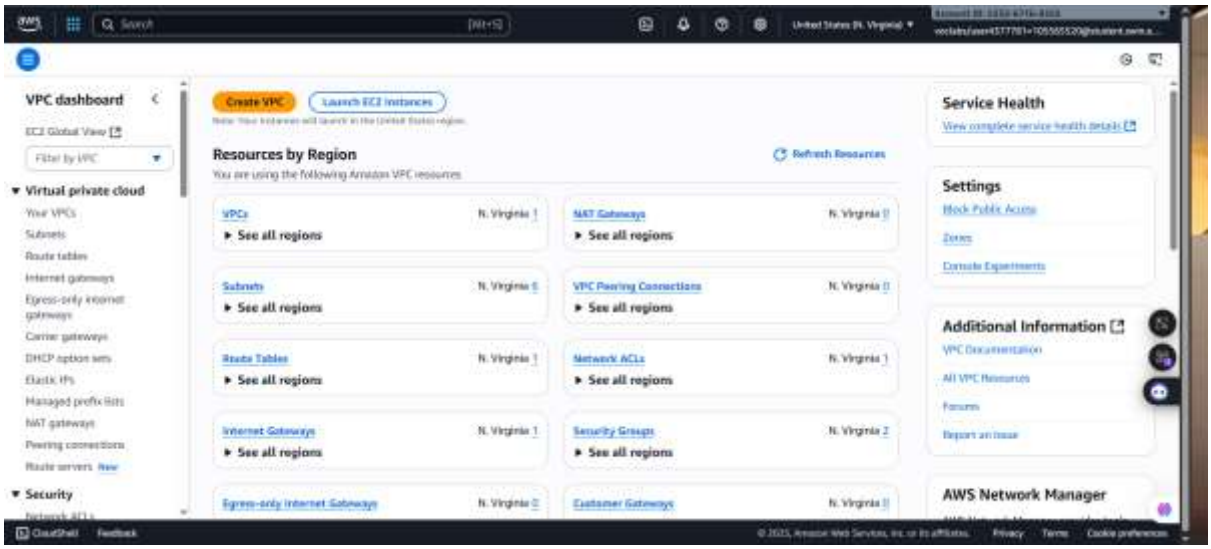


Figure 1: VPC Homepage

Step 1.1.2: Then, choose the **Create VPC** button. In the **VPC settings**, configure following settings:

- **Resources to create**: VPC and more
- **Auto-generate:** GHieuVPC
- **IPv4 CIDR block**: 10.0.0.0/16

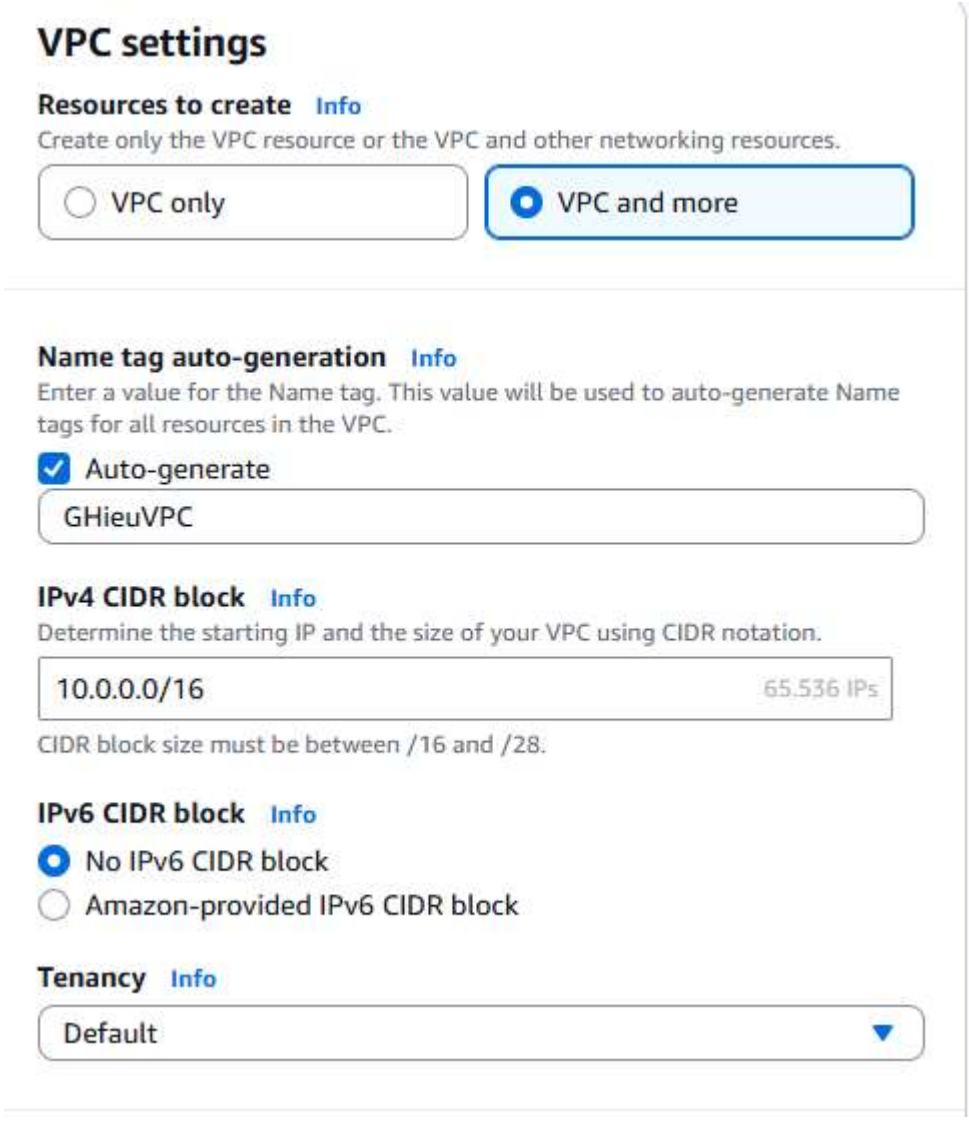


Figure 2: VPC basic configuration

- **Number of Availability Zones:** 2
  + First availability zone: use1-az4 (us-east-1a)
  + Second availability zone: use1-az6 (us-east-1b)

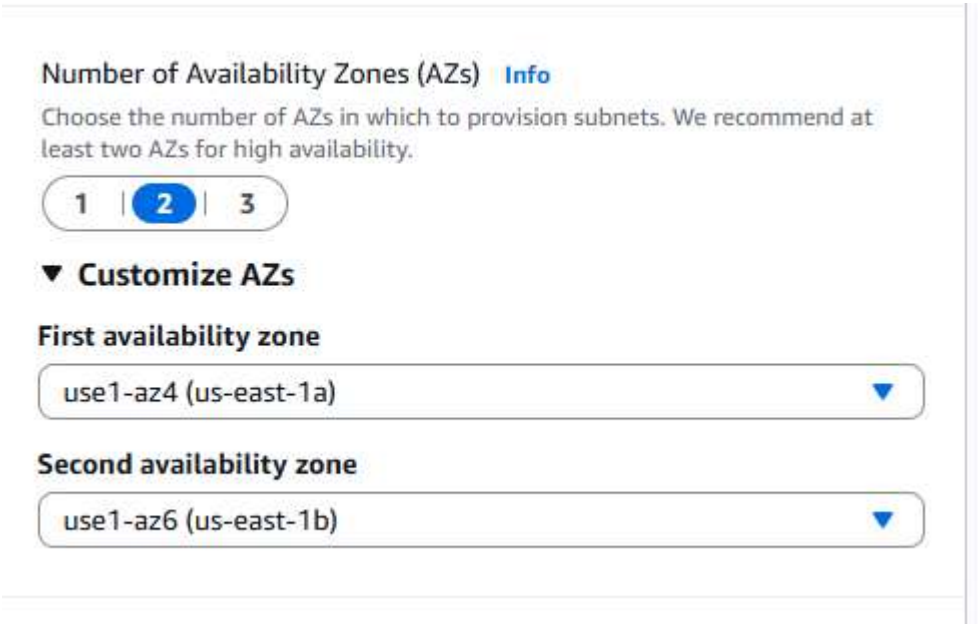


Figure 3: Availability zone configuration

- **Number of public subnets:** 2
  + **Public subnet CIDR block in us-east-1a:** 10.0.1.0/24
  + **Public subnet CIDR block in us-east-1b:** 10.0.2.0/24
- **Number of private subnets:** 2
  + **Private subnet CIDR block in us-east-1a:** 10.0.3.0/24
  + **Private subnet CIDR block in us-east-1b:** 10.0.4.0/24

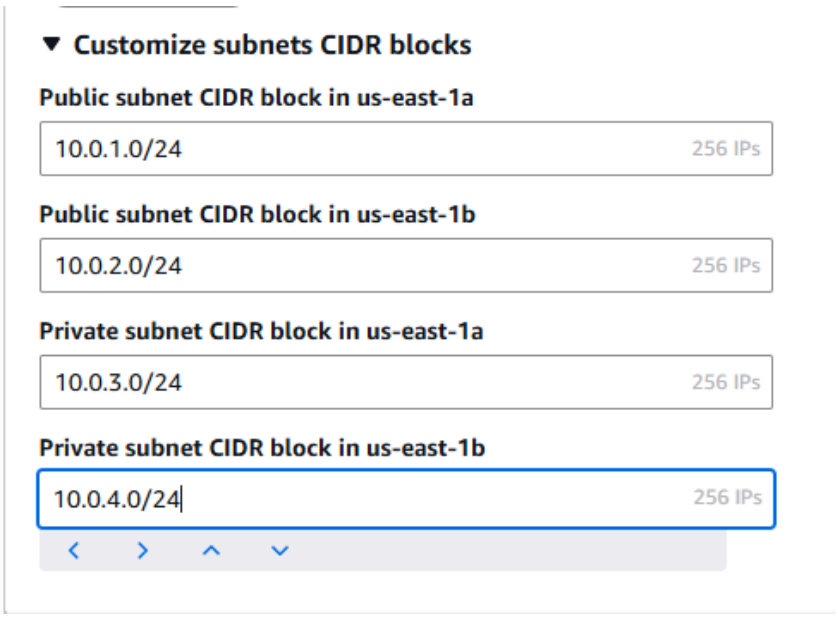


Figure 4: IP address configuration

Figure 5: Preview check

Step 1.1.3: After checking all configurations are correct, click button **Create VPC** to generate own VPC name "**GHieuVPC**"

Figure 6: "GHieuVPC" created sucessfully

Step 1.2: Creating Security Groups

Create the following security groups, each is associated with each tier shown in the architecture diagram:

Security group name	Protocols	Source
TestInstanceSG	All traffic	Anywhere
WebServerSG	HTTP (80), SSH (22)	Anywhere
	ICMP	TestInstanceSG
DBServerSG	MySQL (3306)	WebServerSG

Step 1.2.1: In the left navigation pane, choose **Security Groups**, and select the "**Create security group**" to create "**TestInstanceSG".**

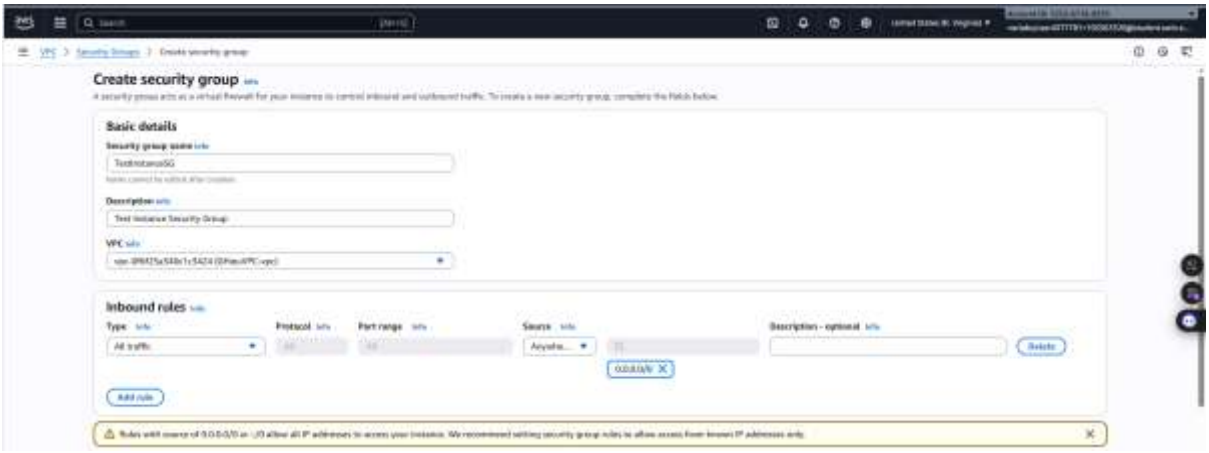


Figure 7: TestInstanceSG configuration

Step 1.2.2: By making sure all settings for the **TestInstanceSG** are correct, click the button **Create security group** to generate.

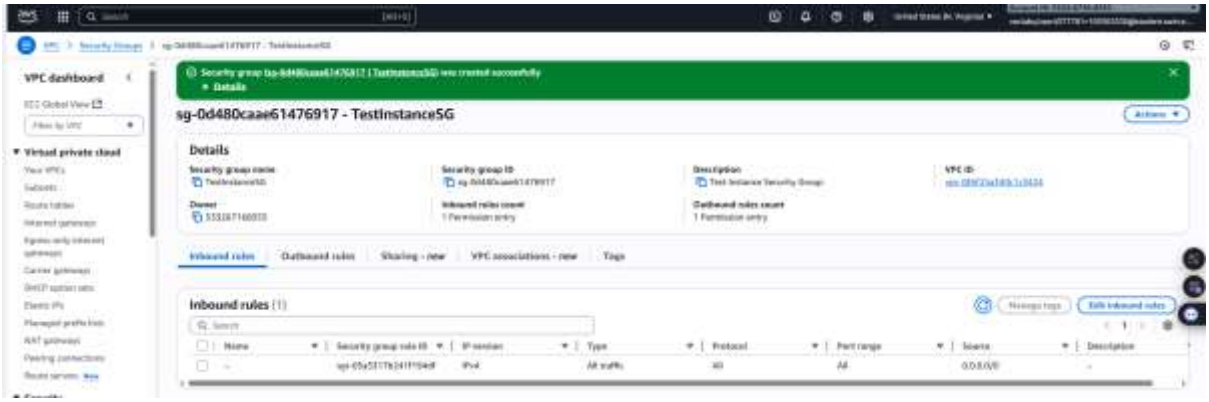


Figure 8: TestInstanceSG created successfully

Step 1.2.3: Similarly to **WebServerSG**

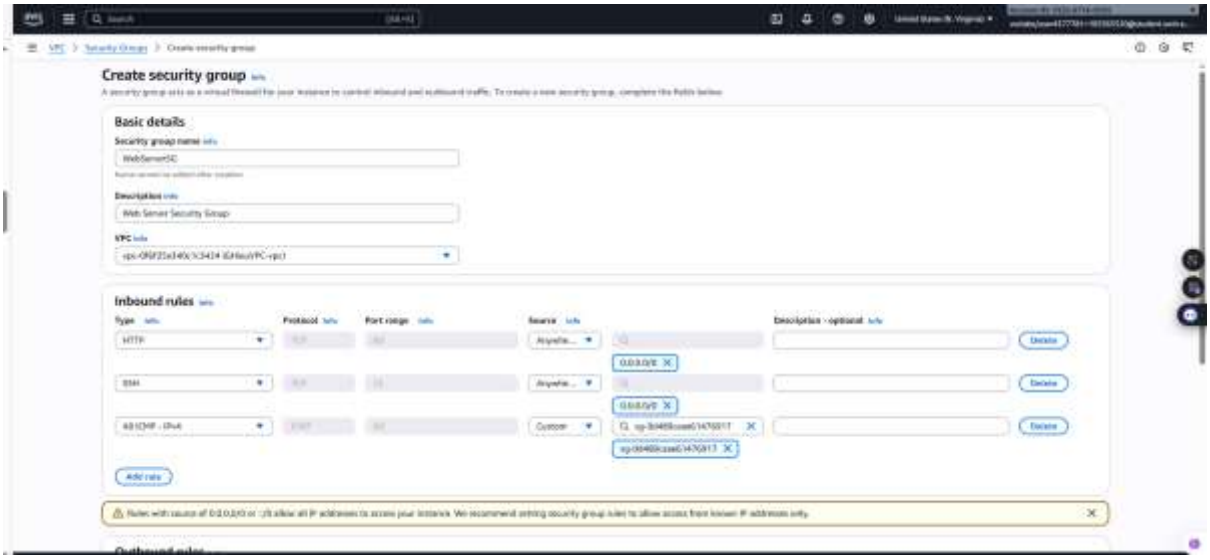


Figure 9: WebServerSG configuration

Step 1.2.4: Then, click the button **Create security group** to generate

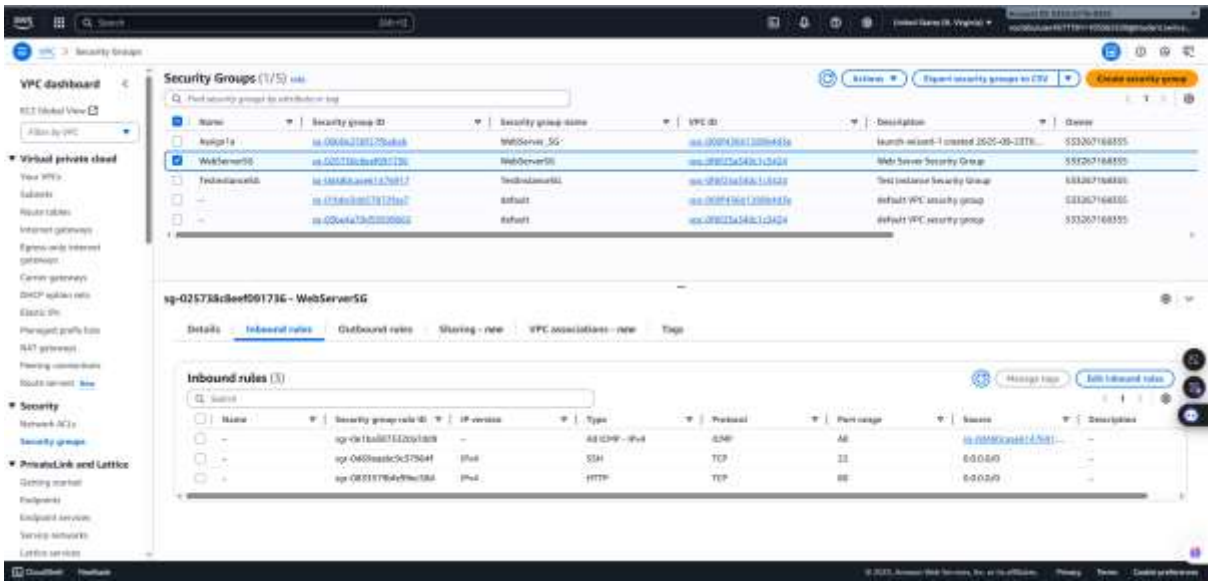


Figure 10: WebServerSG created successfully

Step 1.2.5: Finally, created the **DBServerSG**

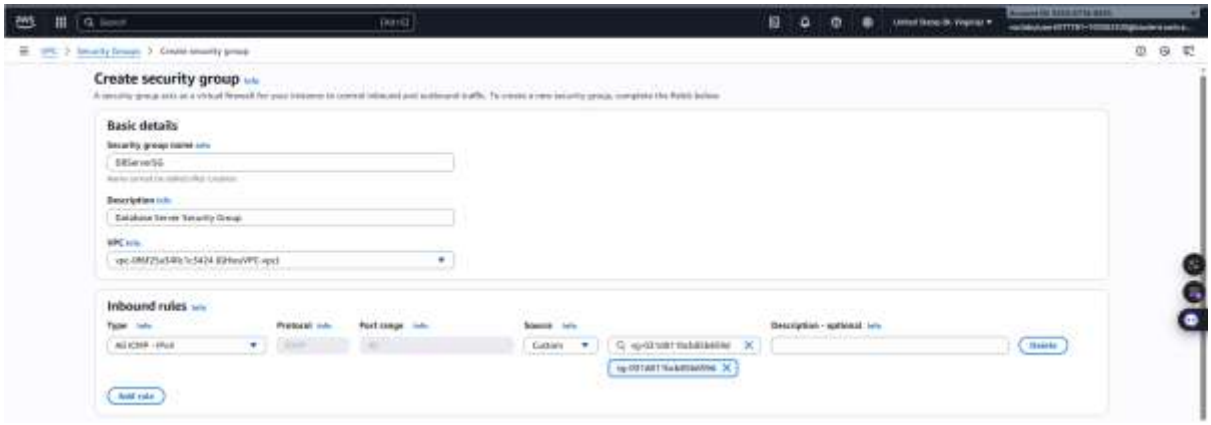


Figure 11: DBServerSG configuration

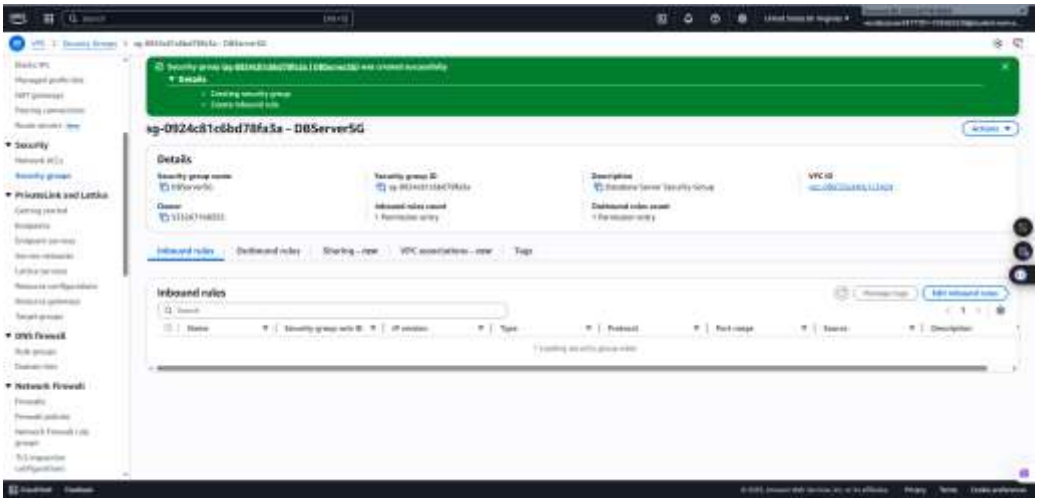


Figure 12: DBServerSG created successfully

Step 1.3: Create Instances

Step 1.3.1: Create **Bastion/Web server instance** and configure following:

+ **Name**: Bastion/Web server instance
+ **AMI**: Amazon Linux 2023
+ **Instance type**: t2.micro
+ **Key Pair**: Assignment1a
+ Network Settings:
  * **VPC**: GHieuVPC
  * **Subnet**: Public Subnet 2
  * **Select existing security group:** WebServerSG
+ **User Data**: Paste from Install_PHP_AWS_0.2

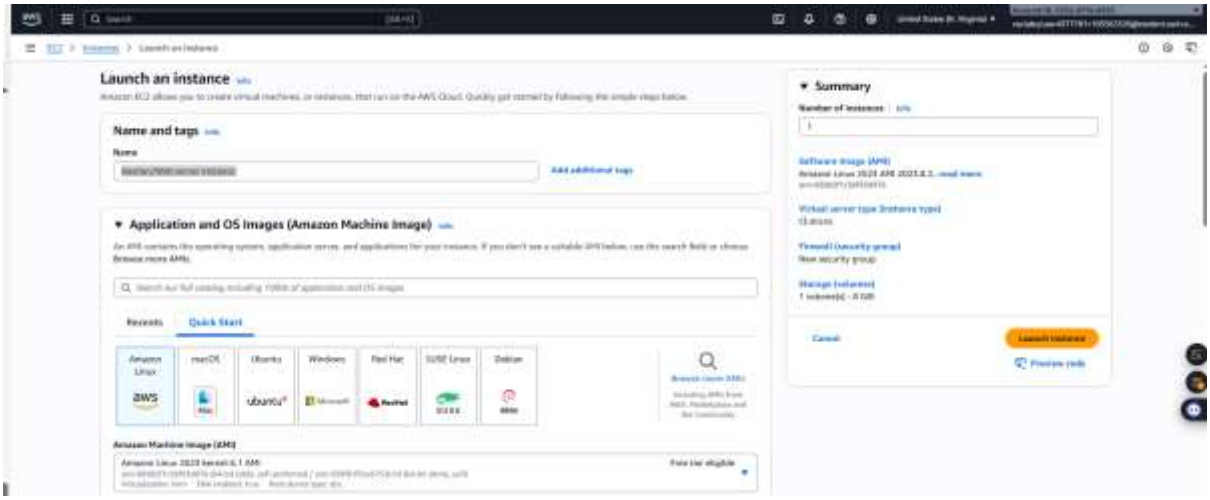


Figure 13: Name and AMI configuration

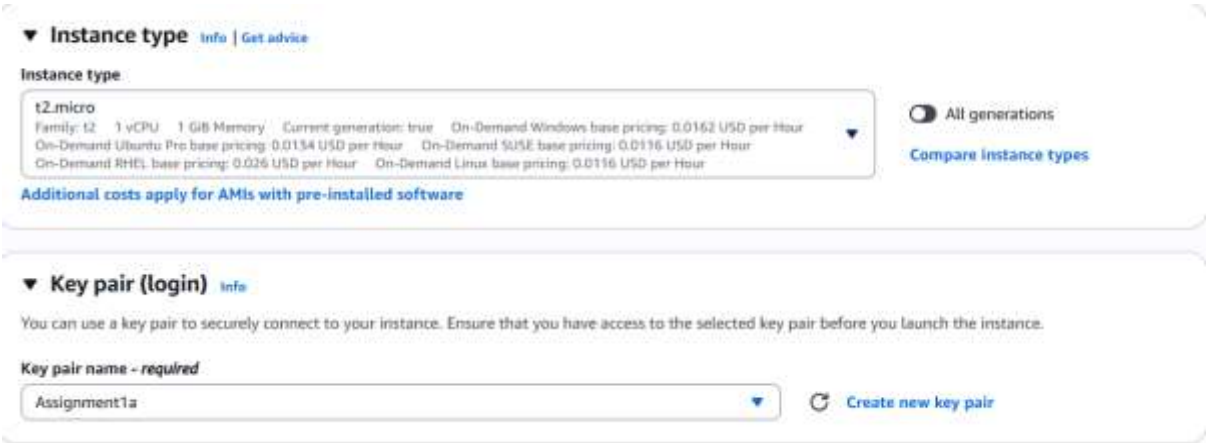


Figure 14: Instance and key pair configuration

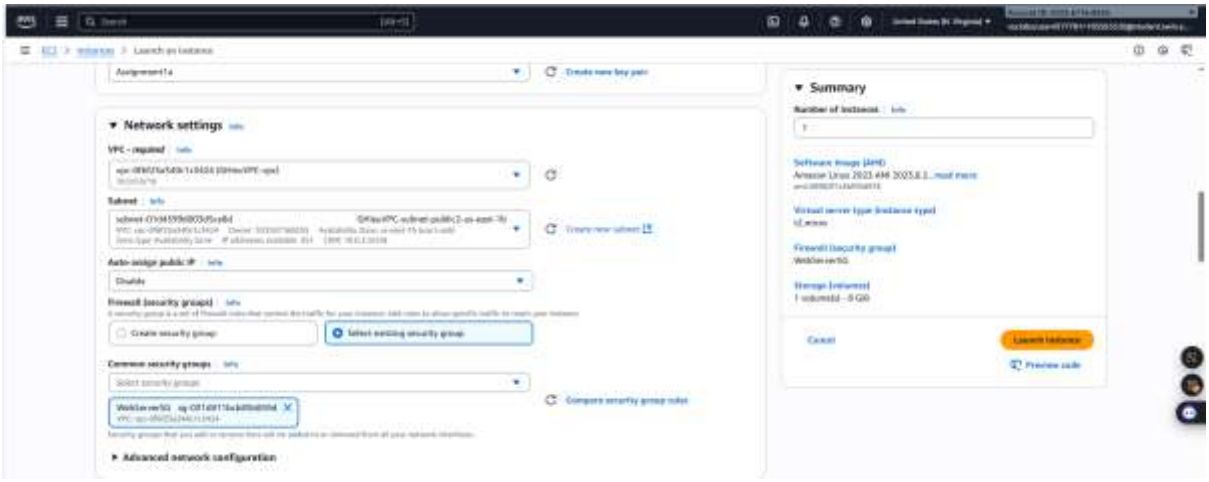


Figure 15: Network settings configuration

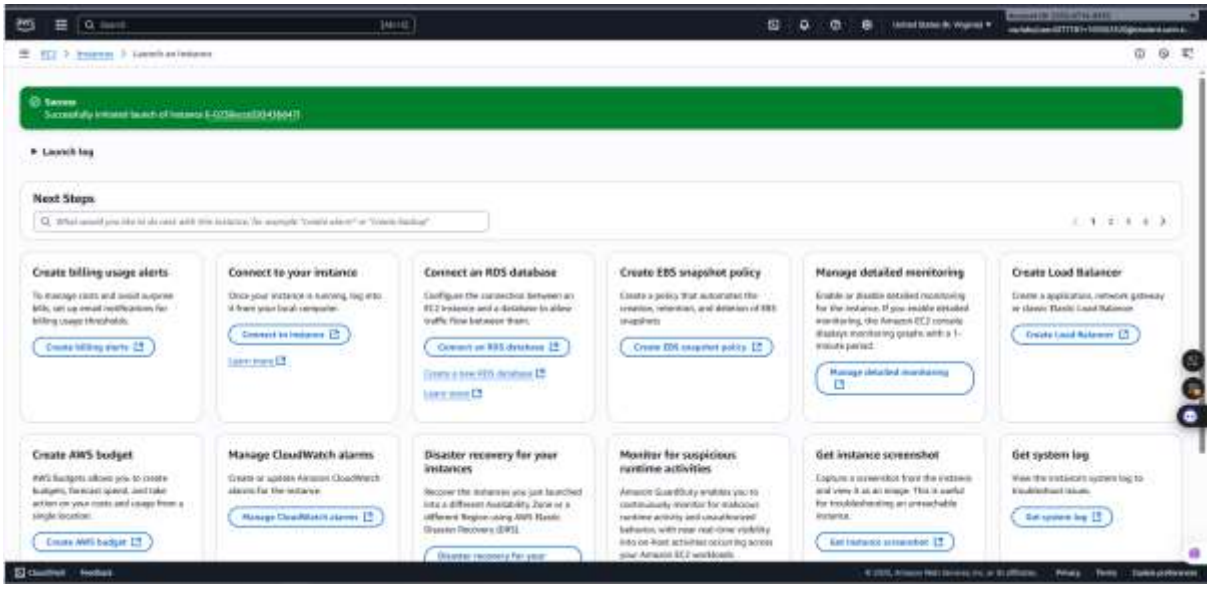


Figure 16: Instance created successfully

Step 1.3.2: Next, choose the **Elastic IPs**, and choose **Allocate Elastic IP address**

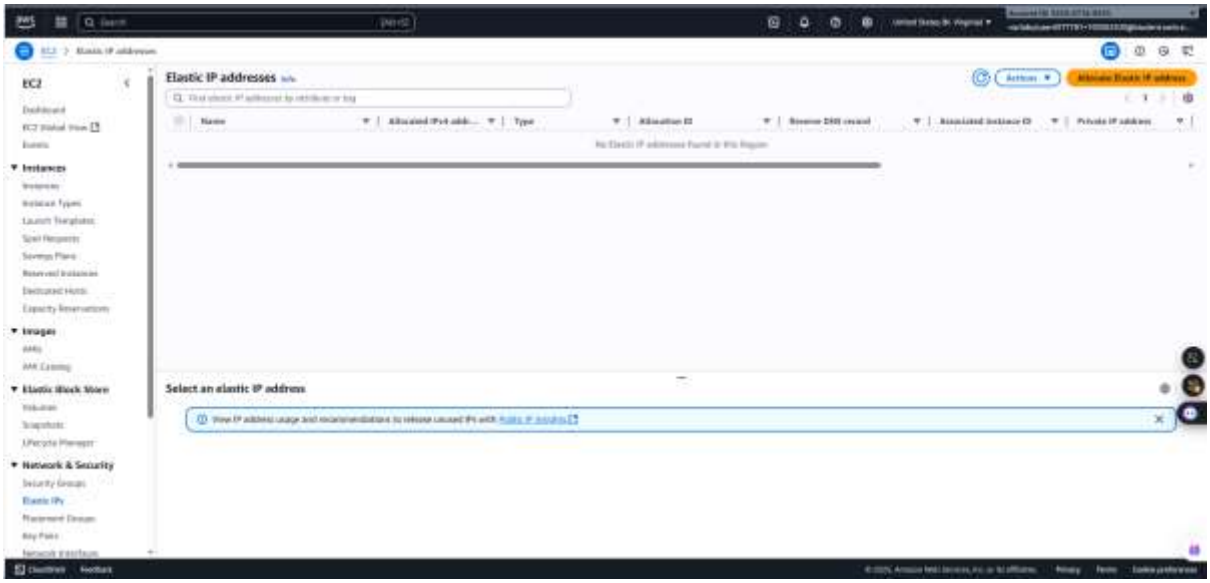


Figure 17: Elastic IPs window

Step 1.3.3: After clicking the button, choose **Allocate**

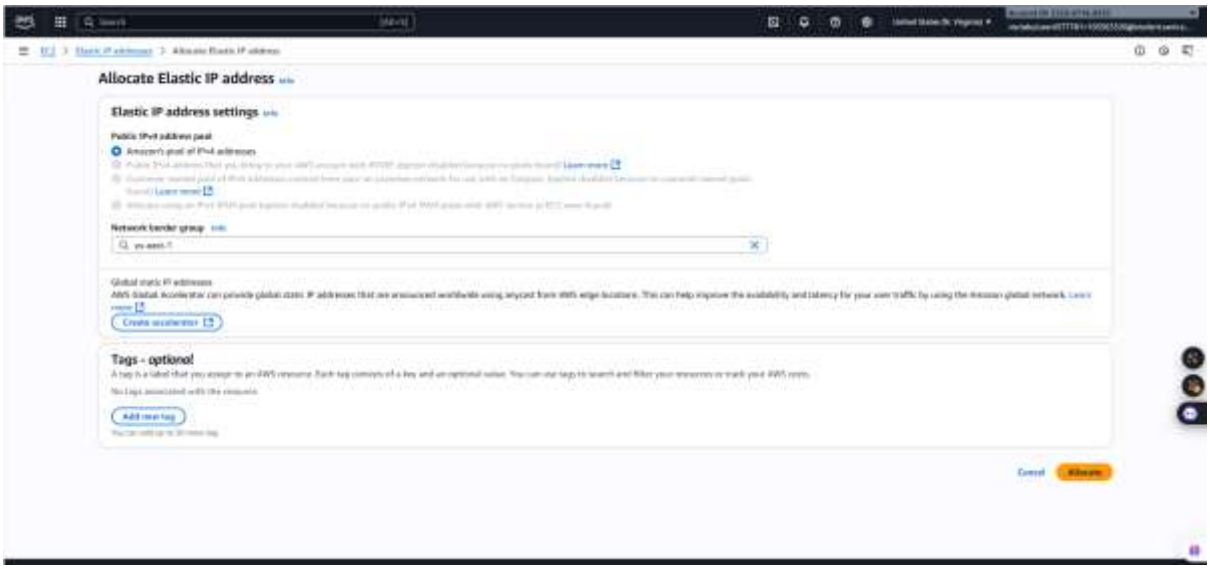


Figure 18: Allocate configuration

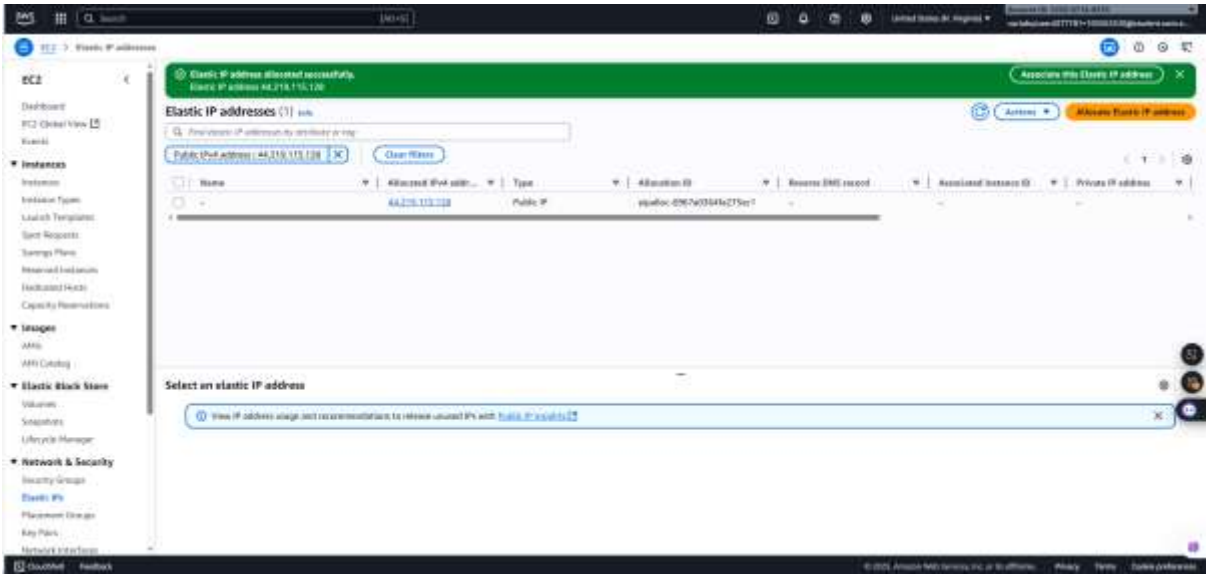


Figure 19: Allocate created successfully

Step 1.3.4: Then, tick the box, choose **Actions => Associate Elastic IP address** and select the **Bastion/Web server instance** in the **Instance**

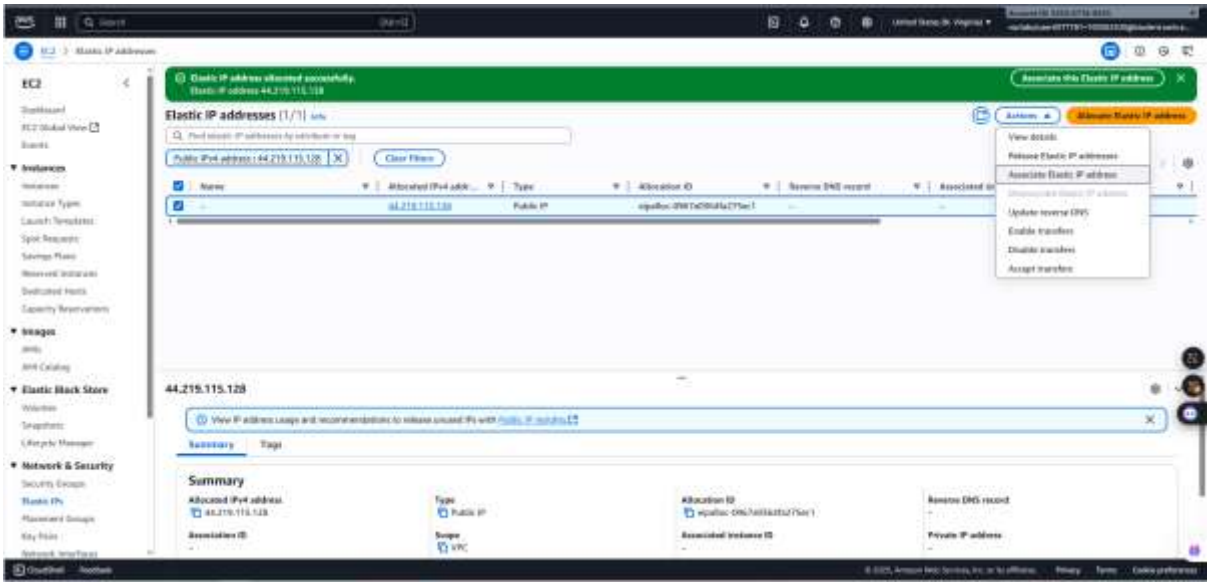


Figure 20: Action selection

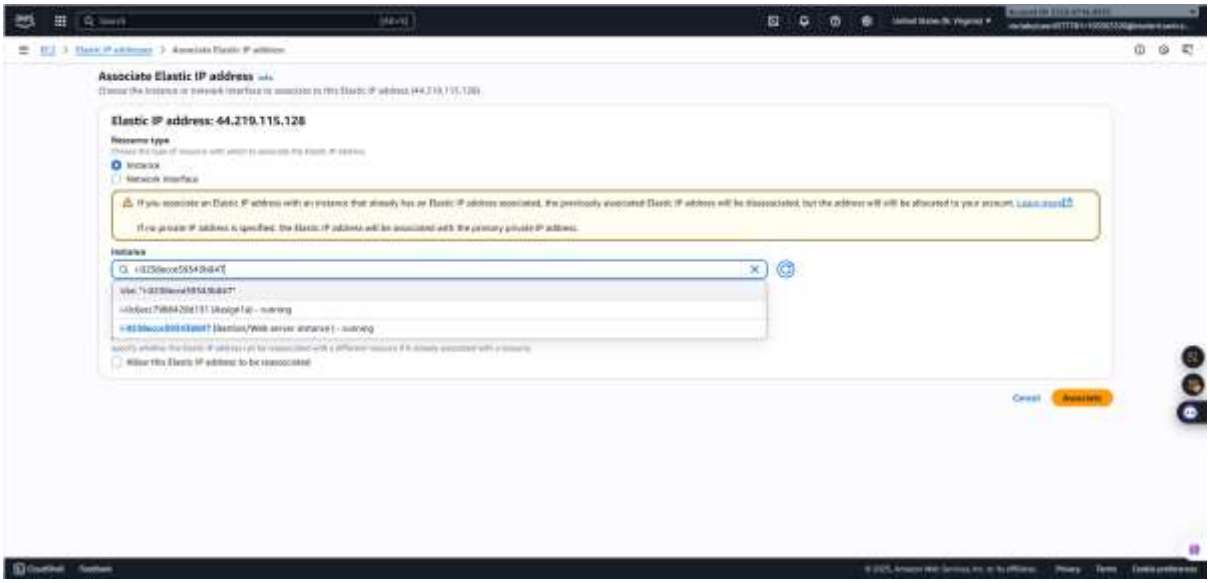


Figure 21: Allocation configuration

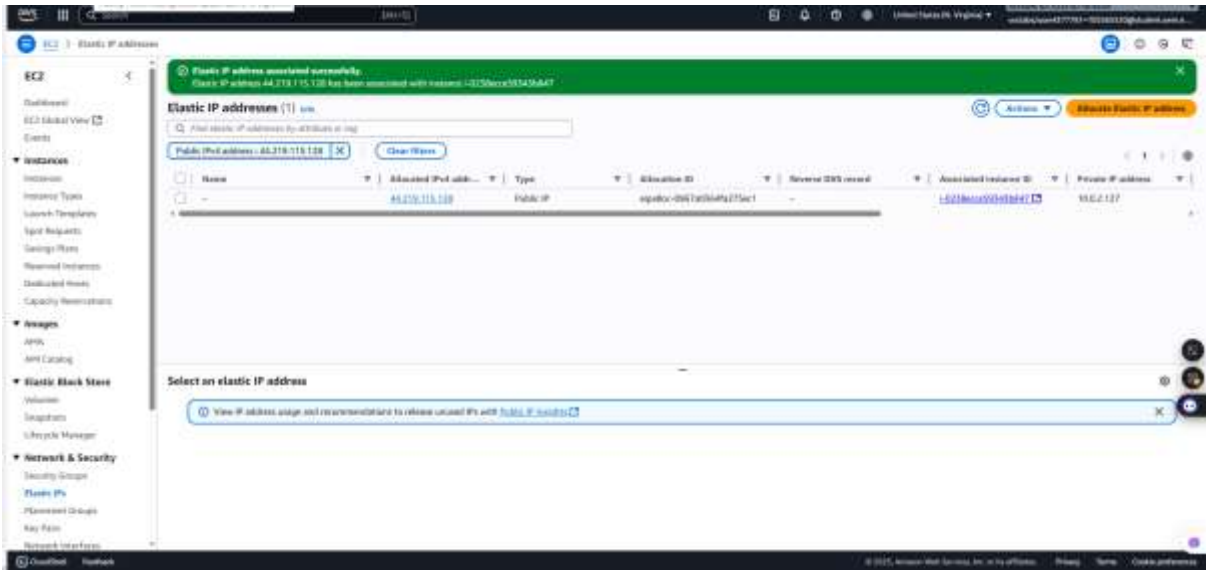


Figure 22: Allocation created successfully

Step 1.3.5: Create **Test Instance**

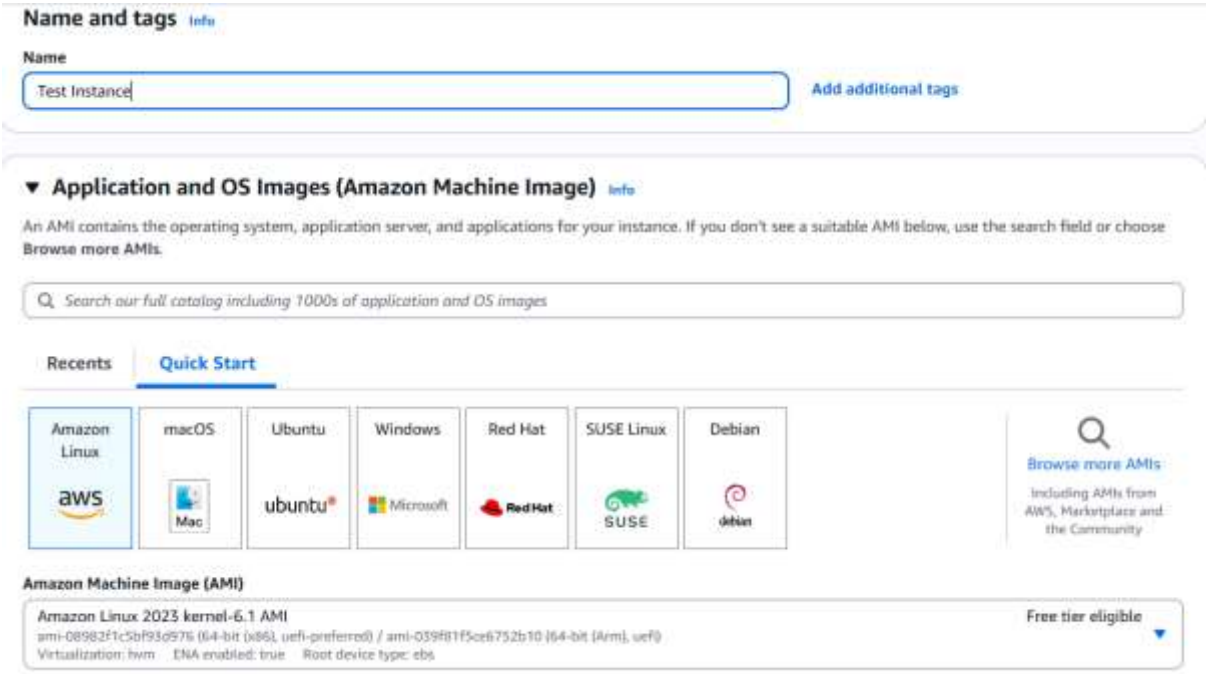


Figure 23: Test Instance's Name and AMI Configuration

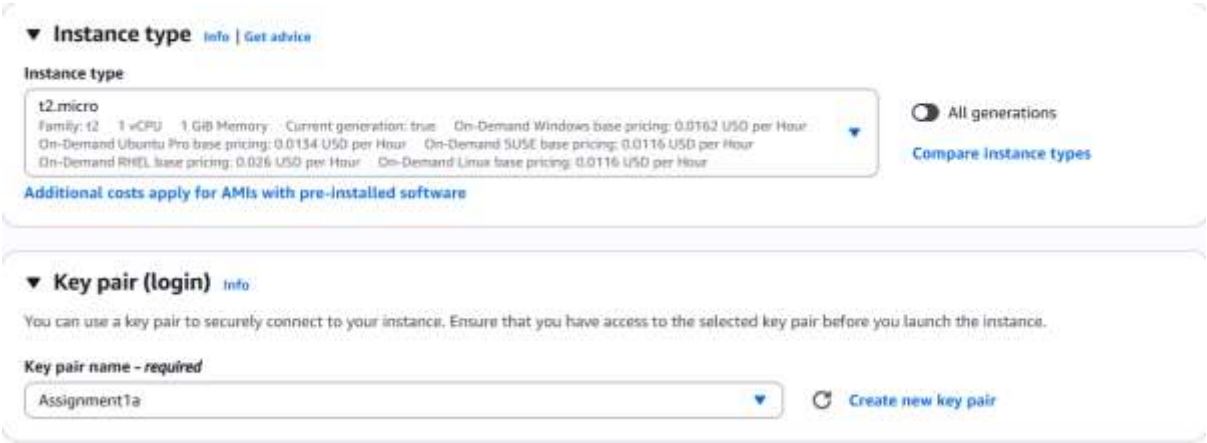


Figure 24: Test Instance's type and Key pair configuration

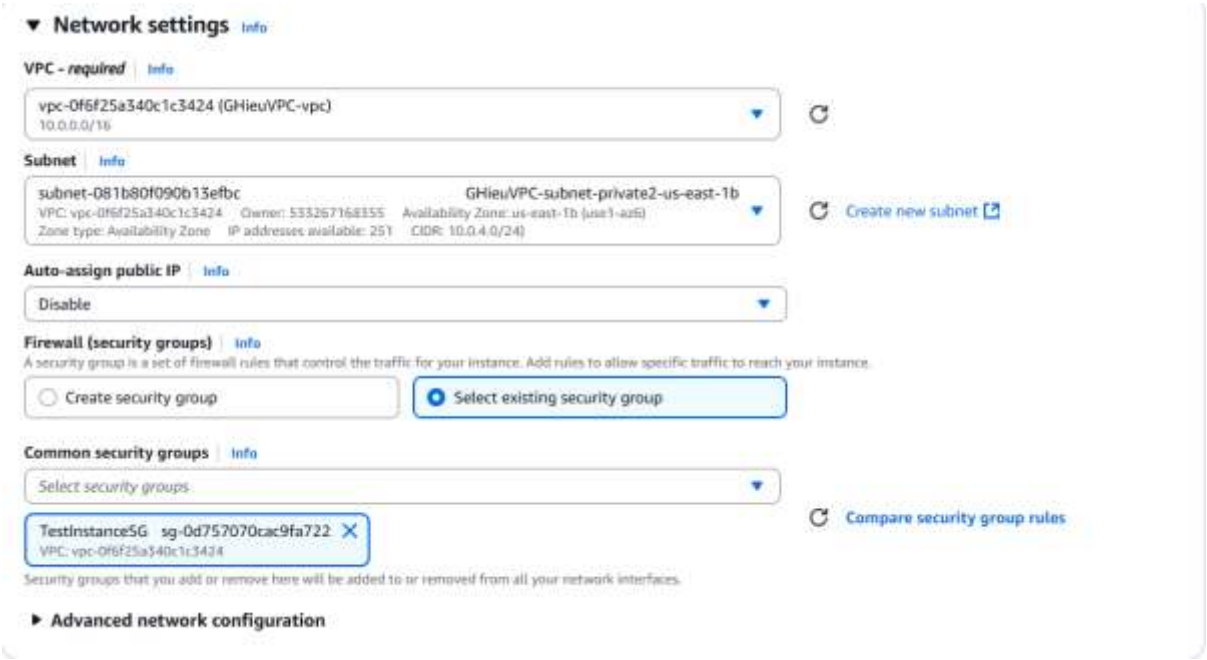


Figure 25: Test Instance's network settings configuration

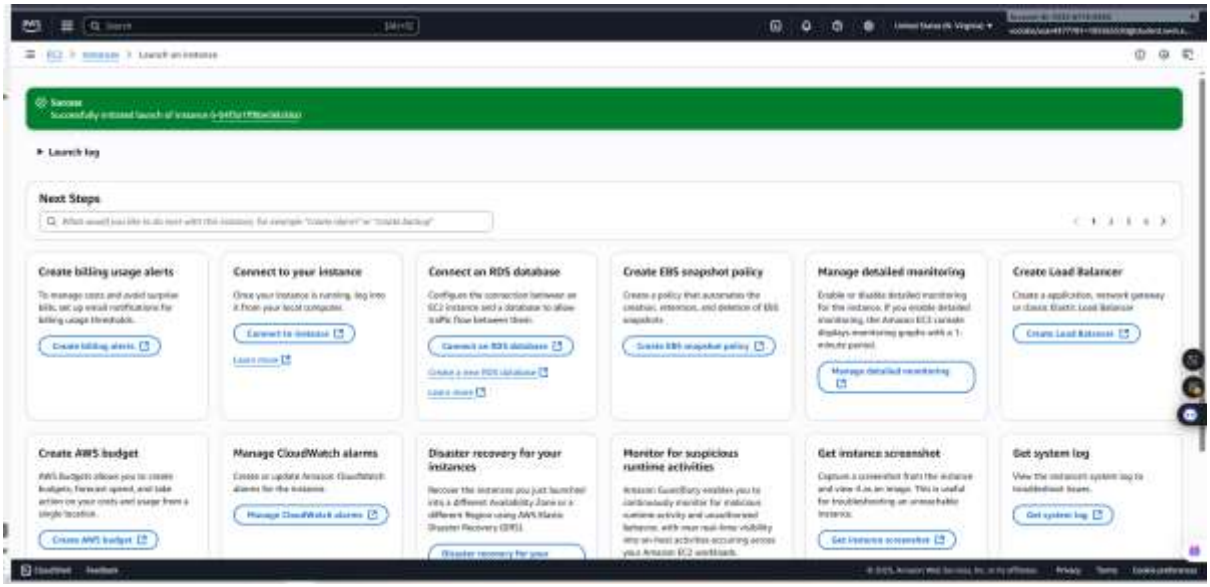


Figure 26: Test Instance's created successfully

Step 1.3.6: Choose Instances to view all **Instances**

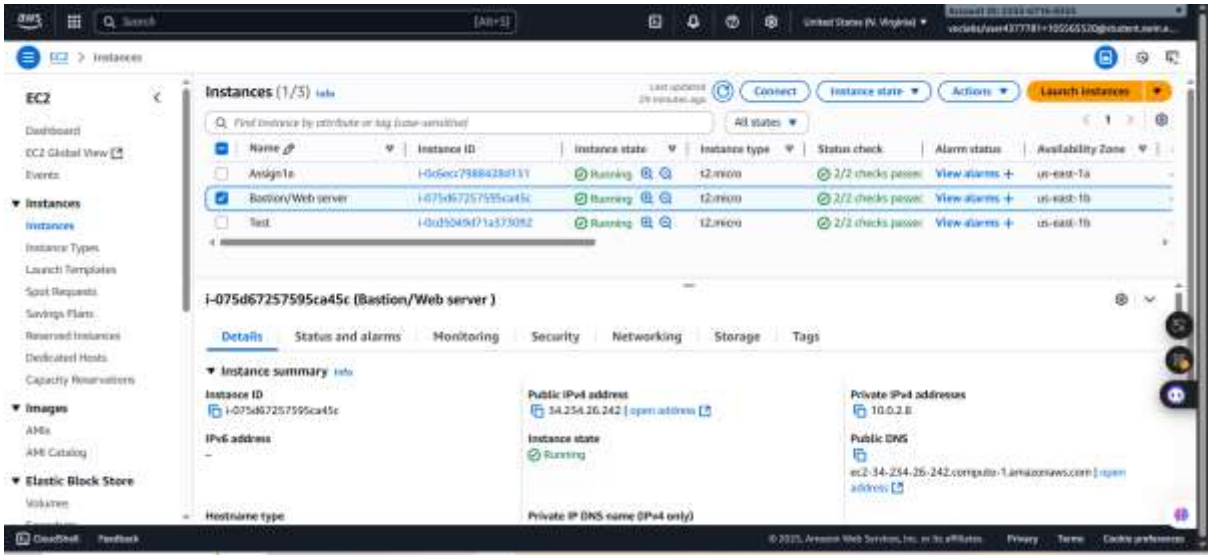


Figure 27: Instance board

Step 1.3.7: In the next step, open the **Putty**, copy the **public DNS link** of the **Bastion/Web server instance** and upload the key pair "**Assignment1a.ppk**" to test ping from **Bastion//Web server to Test instance**

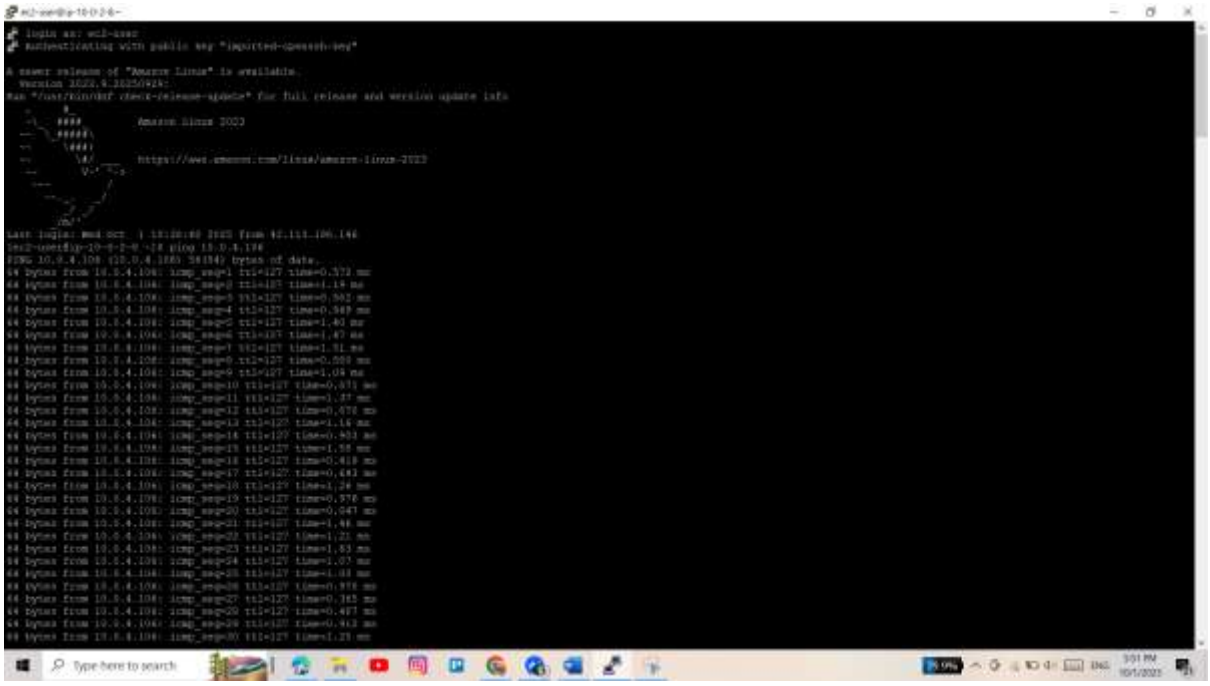


Figure 28: Test ping from Bastion to Test instance successfully

Step 1.3.8: Then, test ping ftom Test to Bastion instance constrast

```
[ec2-user@ip-10-0-2-8 ~]$ ssh -i ./Assignment1a.pem ec2-user@10.0.4.106
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-4-106 ~]$ ping 10.0.2.8
PING 10.0.2.8 (10.0.2.8) 56(84) bytes of data.
64 bytes from 10.0.2.8: icmp_seq=1 ttl=127 time=0.650 ms
64 bytes from 10.0.2.8: icmp_seq=2 ttl=127 time=0.607 ms
64 bytes from 10.0.2.8: icmp_seq=3 ttl=127 time=0.639 ms
64 bytes from 10.0.2.8: icmp_seq=4 ttl=127 time=1.06 ms
64 bytes from 10.0.2.8: icmp_seq=5 ttl=127 time=0.805 ms
64 bytes from 10.0.2.8: icmp_seq=6 ttl=127 time=0.982 ms
64 bytes from 10.0.2.8: icmp_seq=7 ttl=127 time=1.04 ms
64 bytes from 10.0.2.8: icmp_seq=8 ttl=127 time=0.563 ms
64 bytes from 10.0.2.8: icmp_seq=9 ttl=127 time=0.554 ms
64 bytes from 10.0.2.8: icmp_seq=10 ttl=127 time=1.26 ms
64 bytes from 10.0.2.8: icmp_seq=11 ttl=127 time=0.745 ms
64 bytes from 10.0.2.8: icmp_seq=12 ttl=127 time=1.43 ms
64 bytes from 10.0.2.8: icmp_seq=13 ttl=127 time=1.02 ms
64 bytes from 10.0.2.8: icmp_seq=14 ttl=127 time=0.664 ms
64 bytes from 10.0.2.8: icmp_seq=15 ttl=127 time=0.747 ms
^C
--- 10.0.2.8 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14418ms
rtt min/avg/max/mdev = 0.554/0.850/1.427/0.257 ms
[ec2-user@ip-10-0-4-106 ~]$
```

Figure 29: Test ping from Test to Bastiont instance successfully

Step 1.4: RDS database instance

Step 1.4.1: Search and select the **Aurora and RDS** service

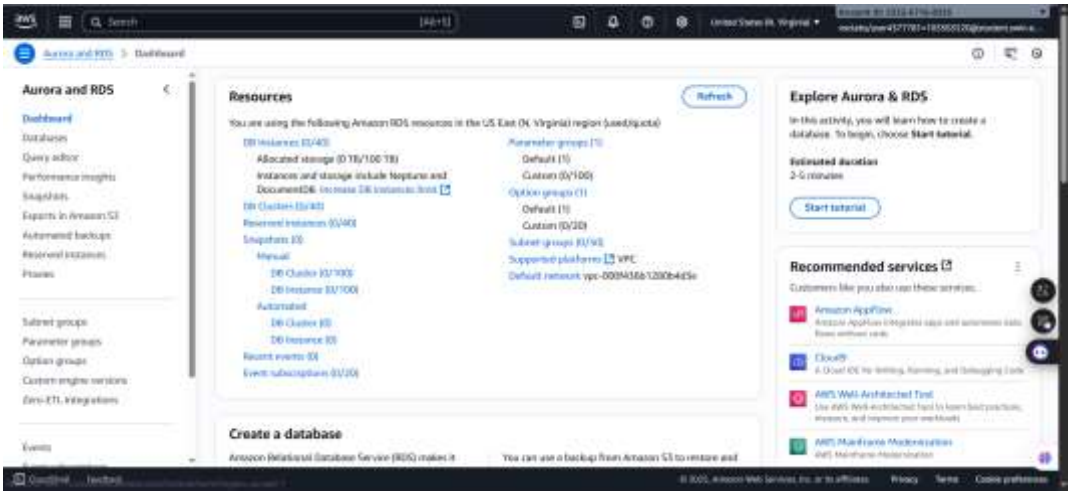


Figure 30: Aurora and RDS mainpage

Step 1.4.2: In the left navigation, choose the **Subnet groups** and press button **Creare DB subnet group**

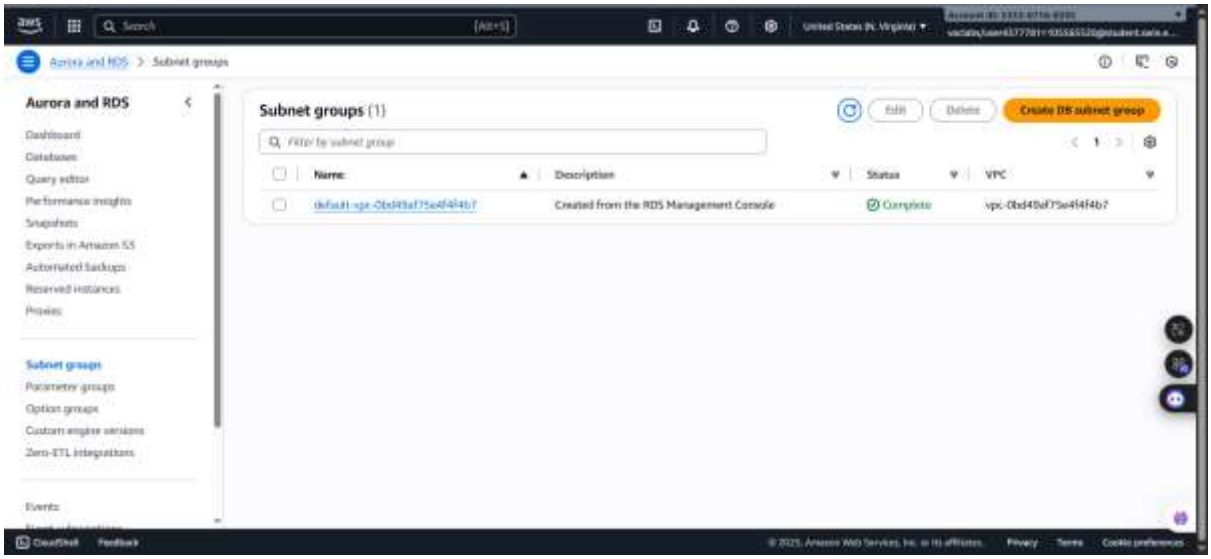


Figure 31: Create DB Subnet group subpage

Step 1.4.3: Then, configure:

+ **Name:** DBSubnetGroup
+ **Description:** Private subnet group
+ **VPC:** THieuVPC
+ **Add subnets:**
  * **Availability Zones:** us-east-1a and us-east-1b
  * **Subnets:** 10.0.3.0/24 and 10.0.4.0/24

Figure 32: Subnet Group details configuration

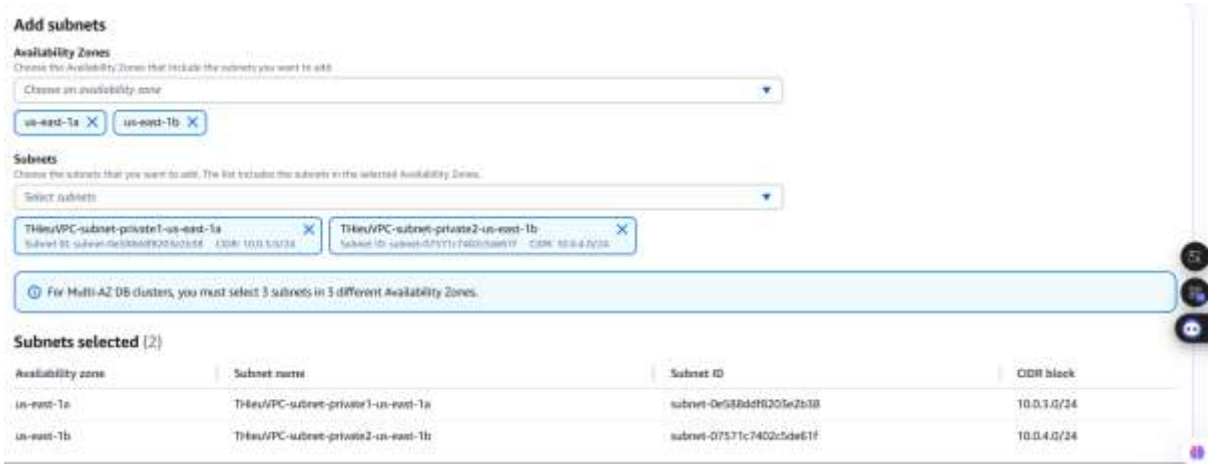


Figure 33: Add subnets configuration

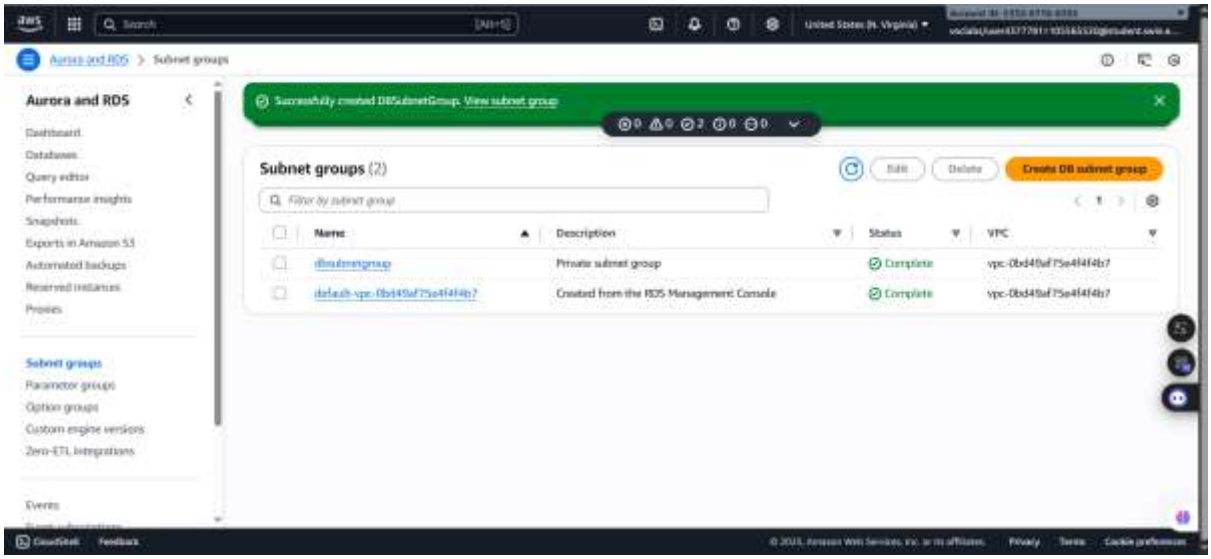


Figure 34: DBSubnetGroup created successfully

Step 1.4.4: Choose the **Databases** in the left navigation pane and select the **Create database button** with following configure:

+ **Public access**: No
+ **Choose a database creation method:** Standard create
+ **Engine type:** MySQL
+ **Engine version:** MySQL 8.0.39
+ **Templates:** Free Tier (Sandbox)
+ **DB instance identifier:** Assignment1b-database
+ **Password:** password
+ **Confirm master password:** password
+ **Compute resource:** *Don't connect to an EC2 compute resource*
+ **Network Type:** IPv4
+ **Virtual private cloud (VPC):** THieuVPC
+ **DB subnet group:** DBSubnetGroup

+ **Availability Zone:** us-east-1a

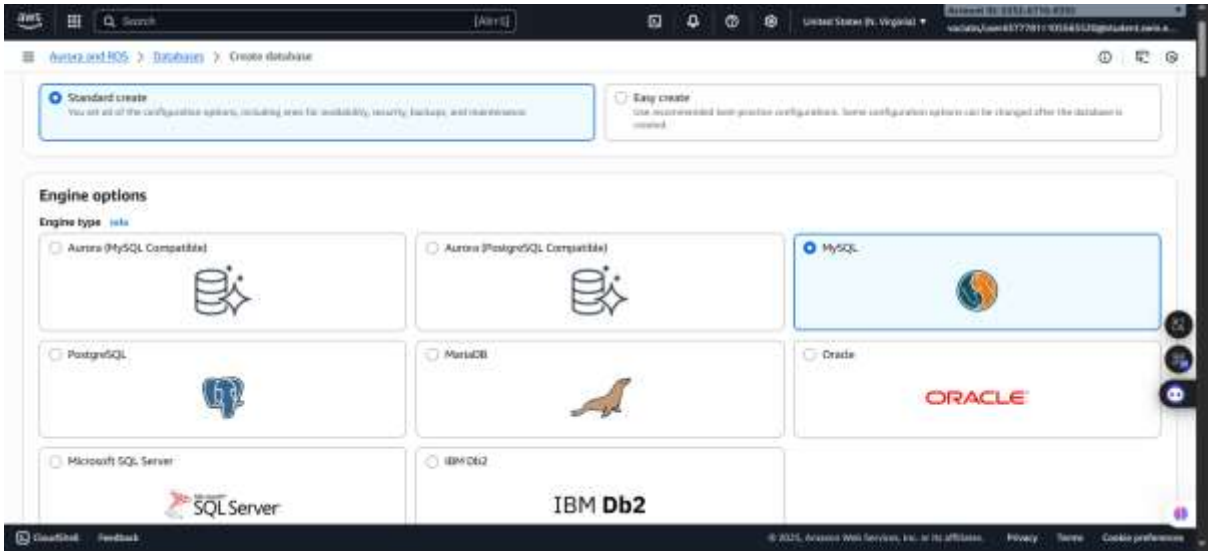


Figure 35: Database method and engine configuration

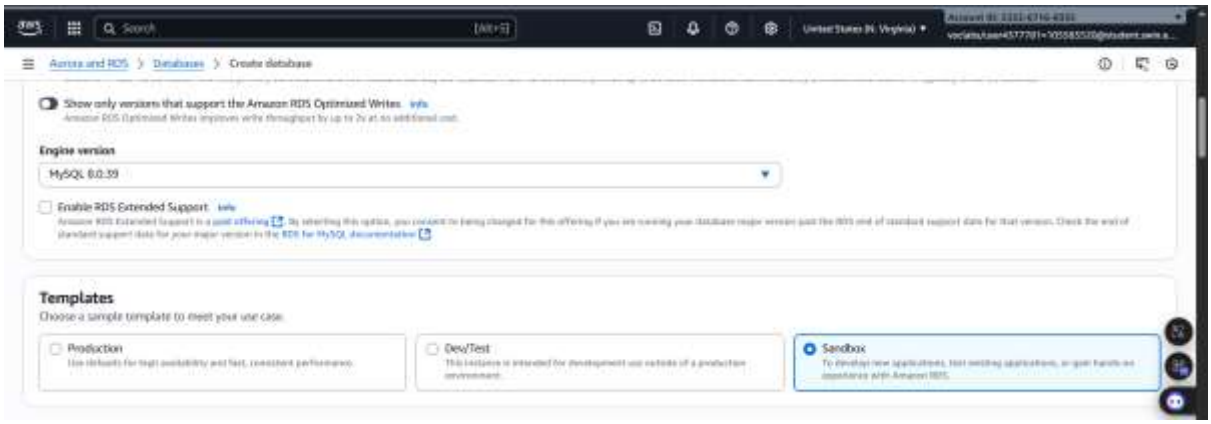


Figure 36: Engine version and Templates configuration

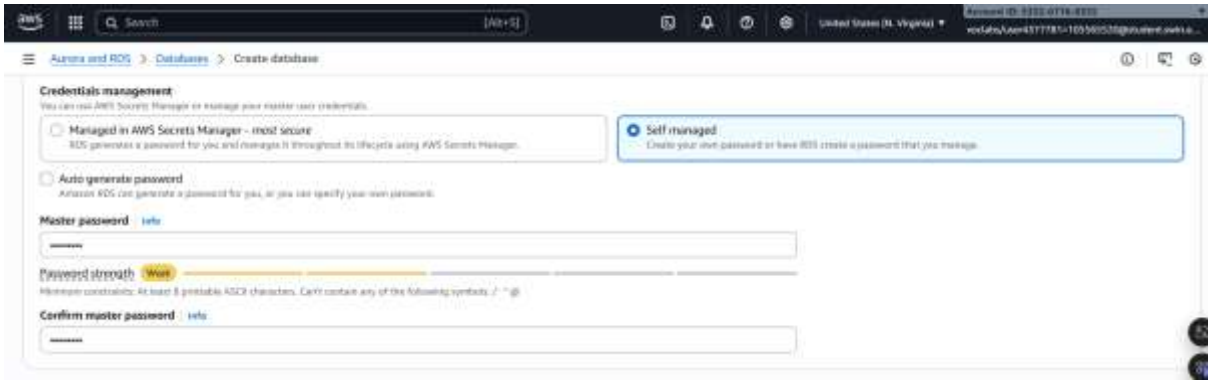


Figure 37: Hostname and password configuration

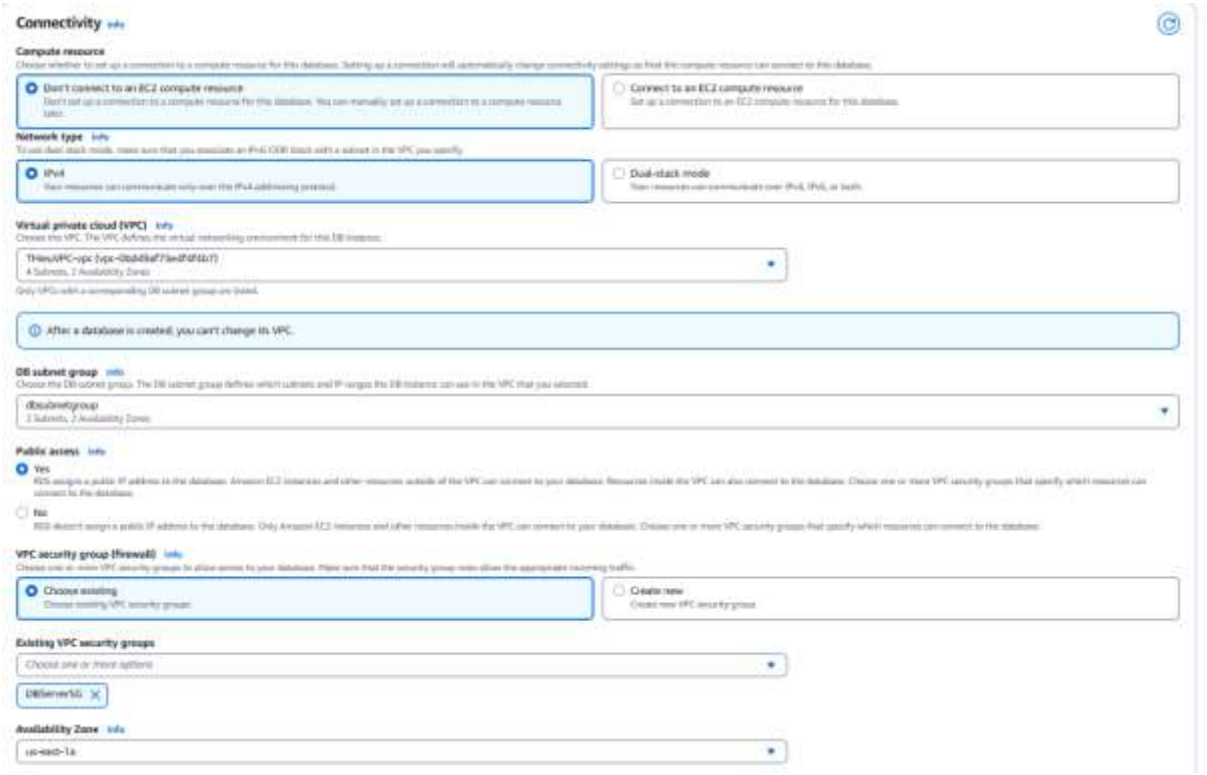


Figure 38: Connectivity configuration

Step 1.4.5: Open the Putty.exe, paste the **punlic DNS link of the Bastion/Web server instanc**e and upload the private key pair

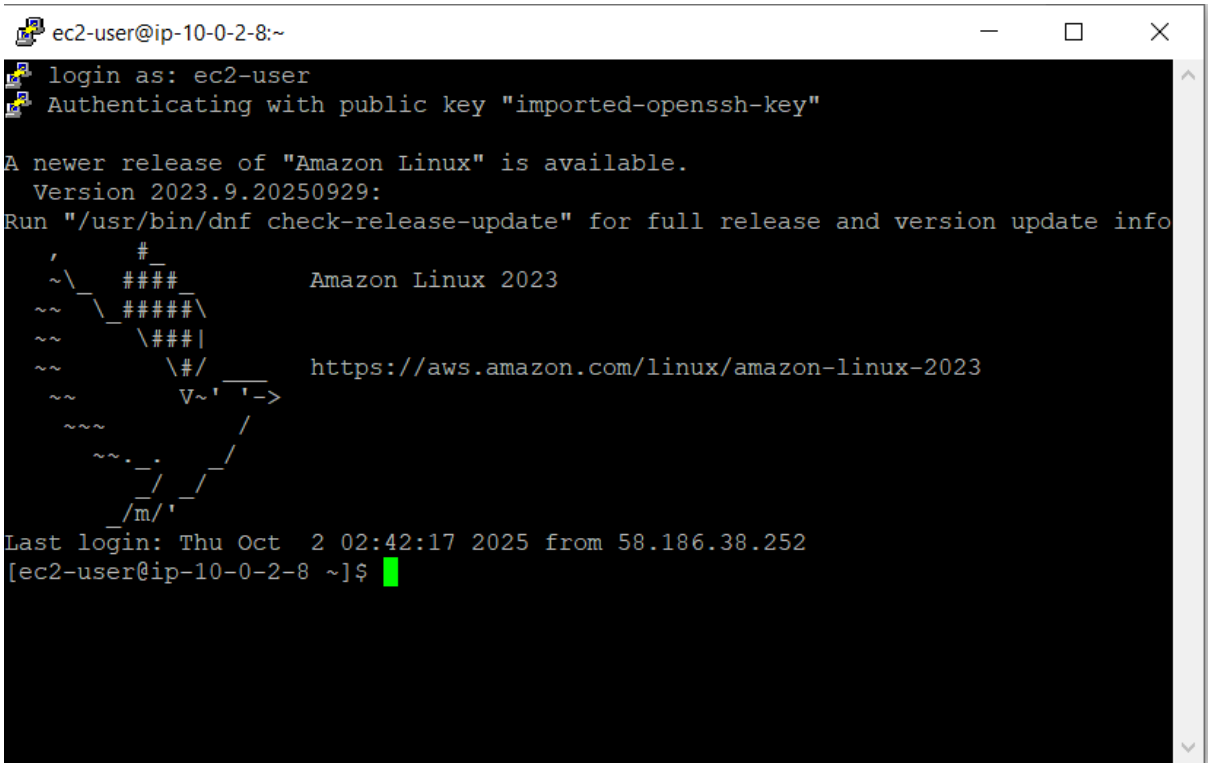


Figure 39: Login successfully

Step 1.4.6: Navigate to the Apache document root directory:

cd /var/www/html

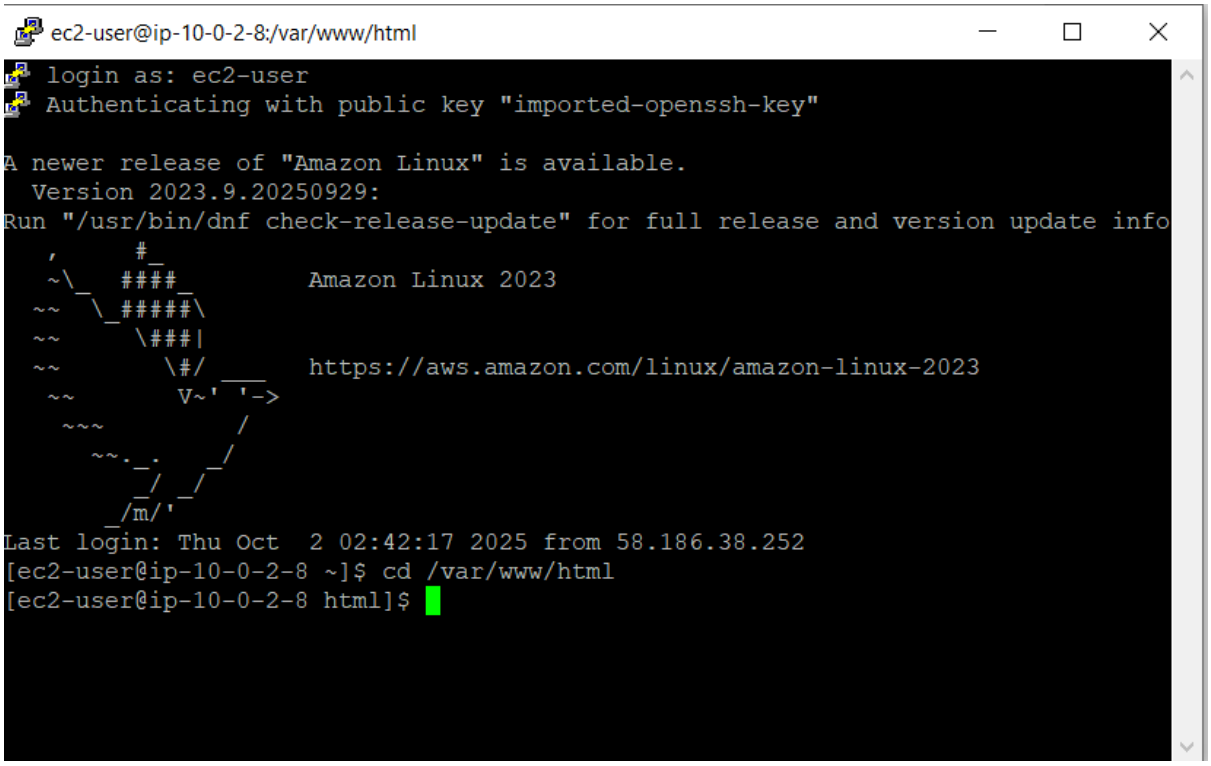


Figure 40: Changed directoty successfully

Step 1.4.7: Download phpMyAdmin source file:

wget https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-english.zip

Figure 41: Download phpMyAdmin sources successfully

Step 1.4.8: Unzip the downloaded zip file:

unzip phpMyAdmin-5.2.1-english.zip

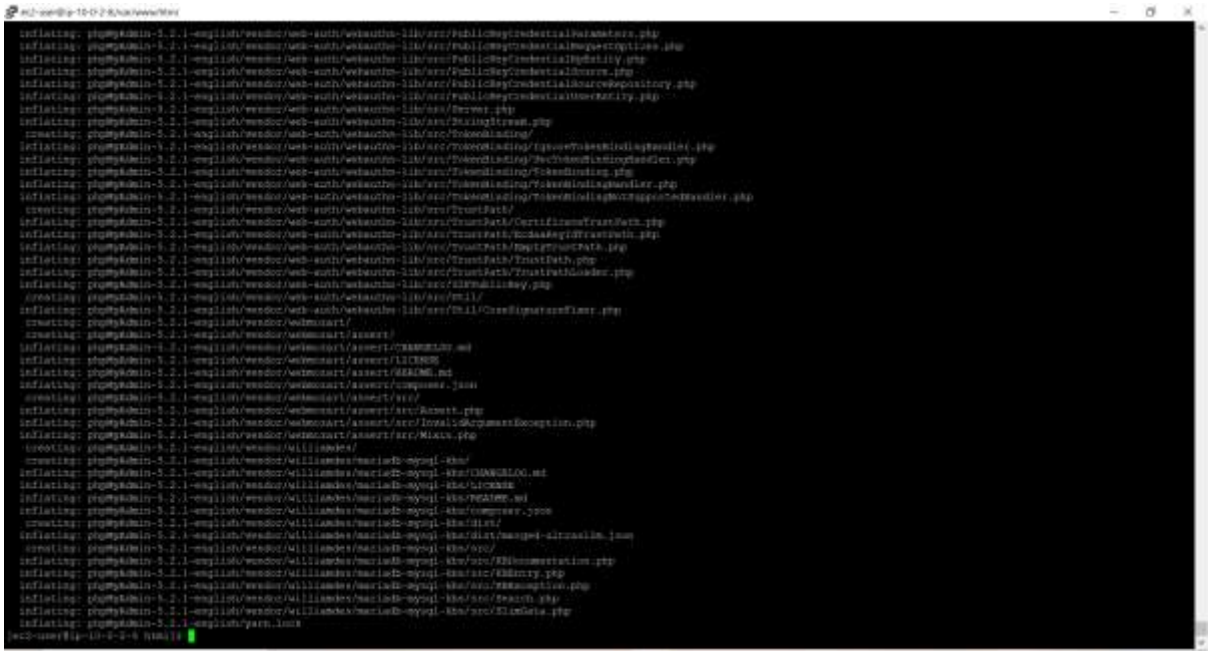


Figure 42: Unzip successfully

***Step 1.4.9*:** Change the directory name to phpmyadmin:

```
mv phpMyAdmin-5.2.1-english phpmyadmin
```

```
[ec2-user@ip-10-0-2-8 html]$ mv phpMyAdmin-5.2.1-english phpmyadmin
mv: cannot move 'phpMyAdmin-5.2.1-english' to 'phpmyadmin/phpMyAdmin-5.2.1-english': File exists
[ec2-user@ip-10-0-2-8 html]$ mv phpMyAdmin-5.2.1-english phpmyadmin
mv: cannot move 'phpMyAdmin-5.2.1-english' to 'phpmyadmin/phpMyAdmin-5.2.1-english': File exists
[ec2-user@ip-10-0-2-8 html]$
```

Figure 42: Change successfully

Step 1.4.10: Open WinSCP, navigate to phpmyadmin directory (var/www/html/phpmyadmin), and change the name of config.sample.inc.php file to config.inc.php

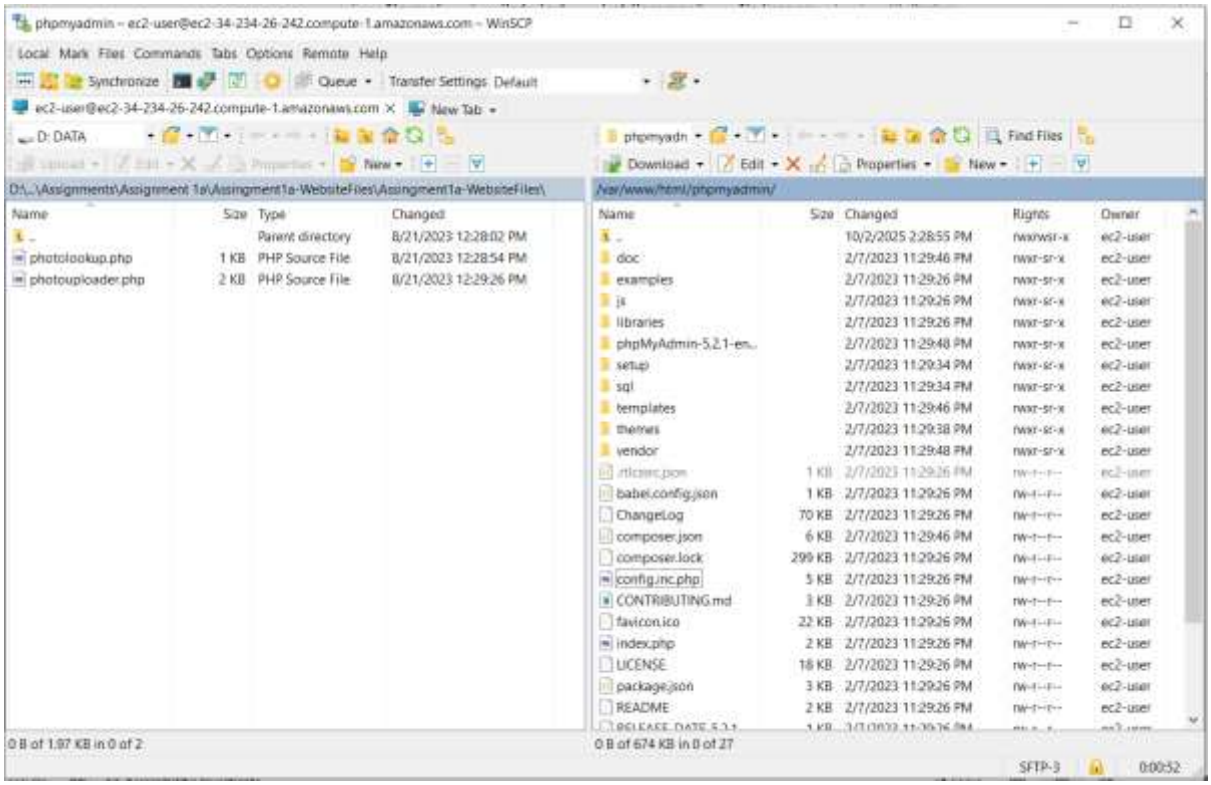


Figure 43: NamedChange successfully

Step 1.4.11: Open config.inc.php file and look for this line:

```
$cfg['Servers'][$i]['host'] = 'localhost';
```

Replace ‘localhost’ with the endpoint of your RDS instance.

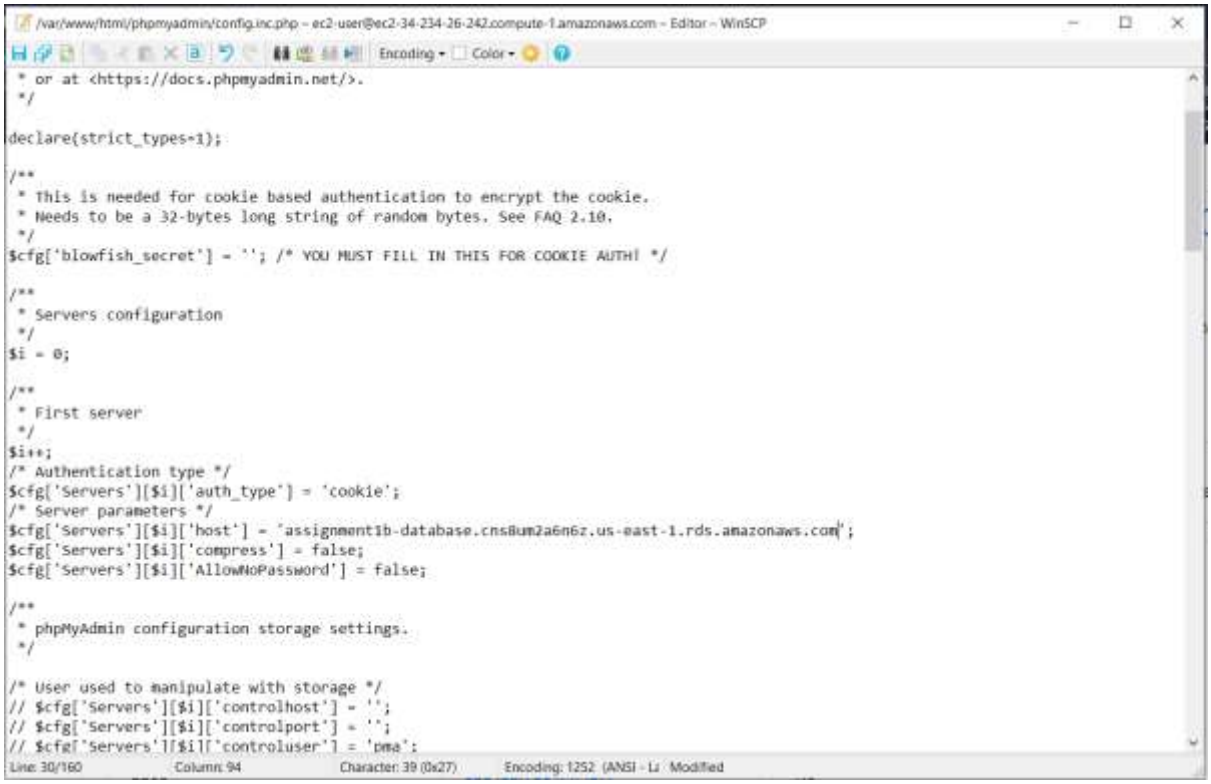


Figure 44: Changed successfully

Step 1.4.12: Paste the link below to a new tab:
http://ec2-34-234-26-242.compute-1.amazonaws.com/phpmyadmin/

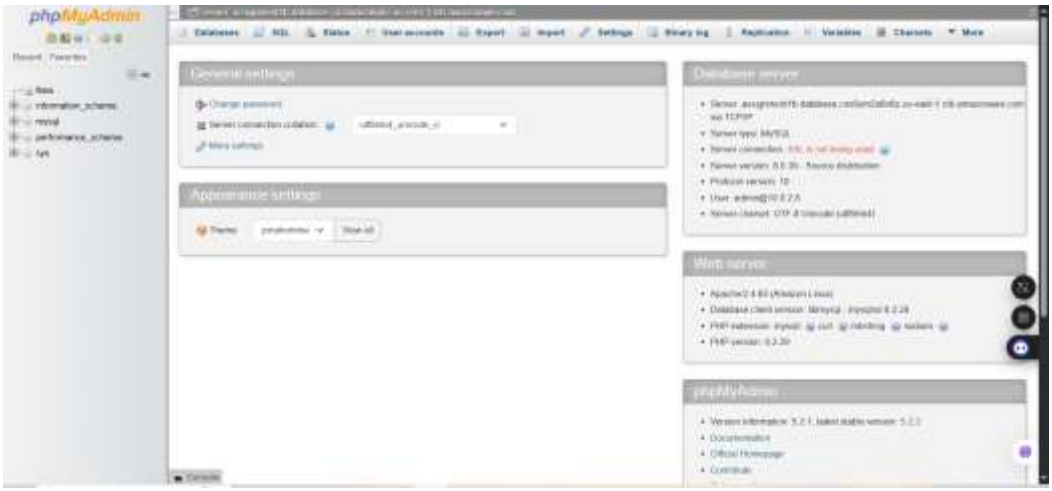


Figure 45: Access successfully

Step 1.4.13: Create new table with following requirements:

+ Photo title (varchar(255) type)
+ Description (varchar(255) type)
+ Creation date (date type)
+ Keywords (varchar(255) type)
+ Reference to the photo object in S3 (varchar(255) type)

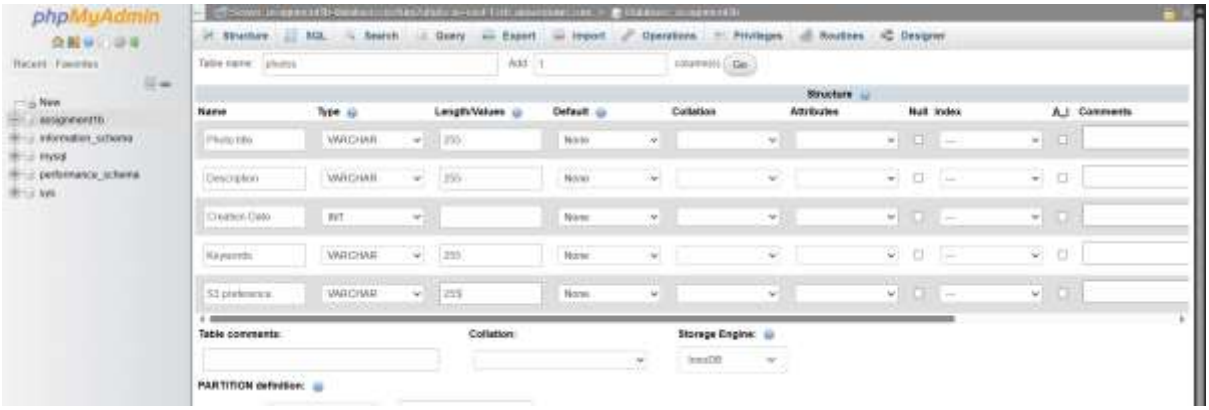


Figure 46: Table configuration

Figure 47: Table created successfully

Step 1.5: Network ACL

Step 1.5.1: Search and select the **VPC**, choose **Network ACLs** and select button **Create network ACL**

+ **Name:** PublicSubnet2NACL
+ **VPC:** THieu-VPC

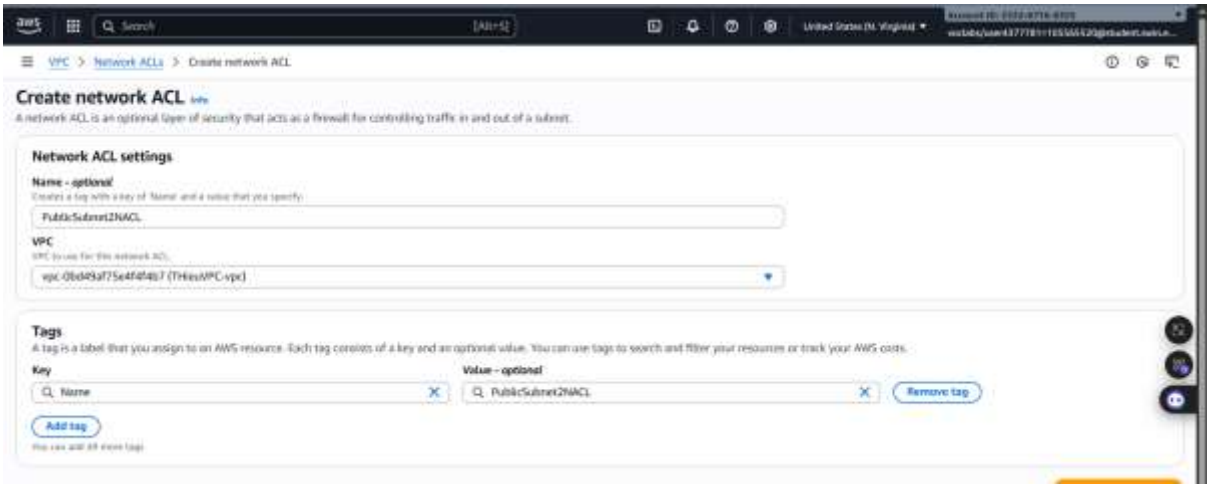


Figure 48: Basic configuration

Step 1.5.2: After created successfully, click **Edit Inbound rules** with following configurations:

+ must ALLOW SSH(22) 1traffic from anywhere so that you can access the WebServer instance.

+ must ALLOW ICMP traffic only from the subnet that contains the Test instance.

+ must ALLOW other necessary traffic so that the Photo Album website is fully functional for users from anywhere.

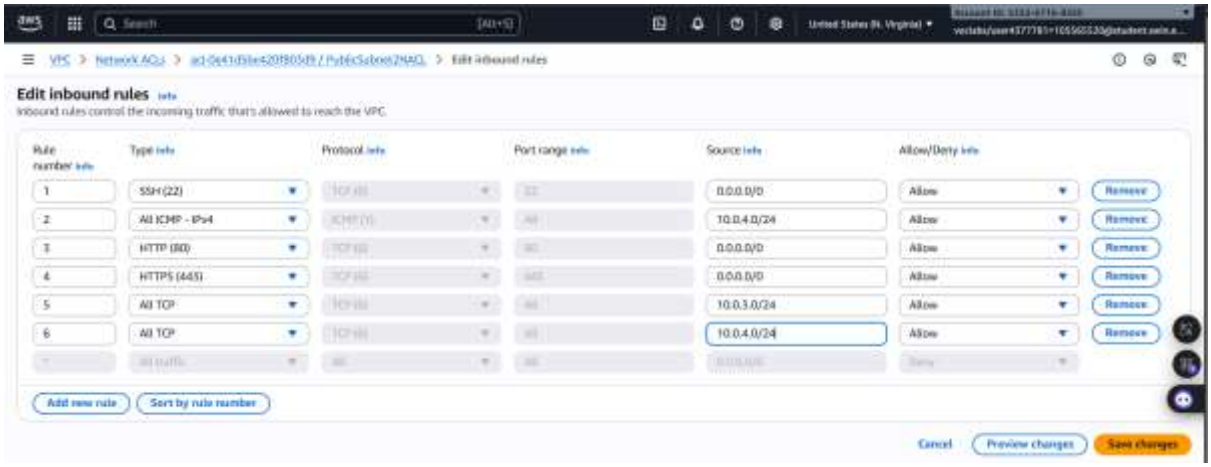


Figure 49: Inbound configuration

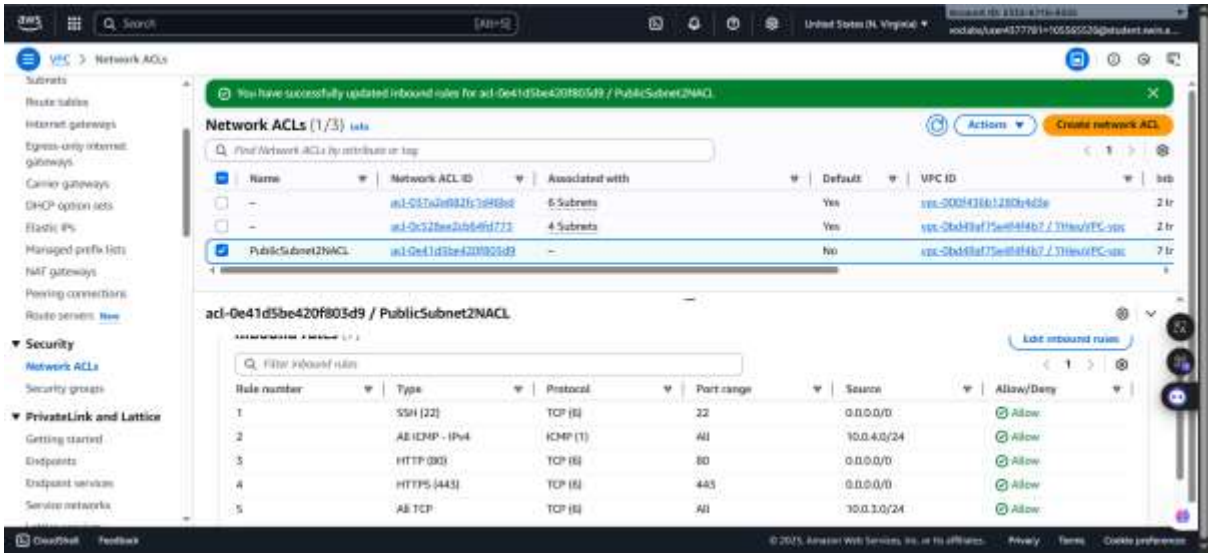


Figure 50: Inbound rule set successfully

Step 1.5.3: Then, edit the outbound rules

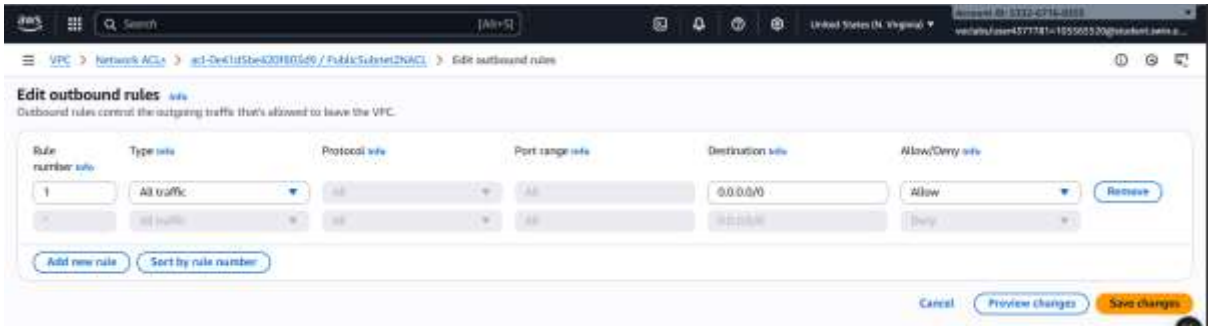


Figure 51: Outbound rule configuration

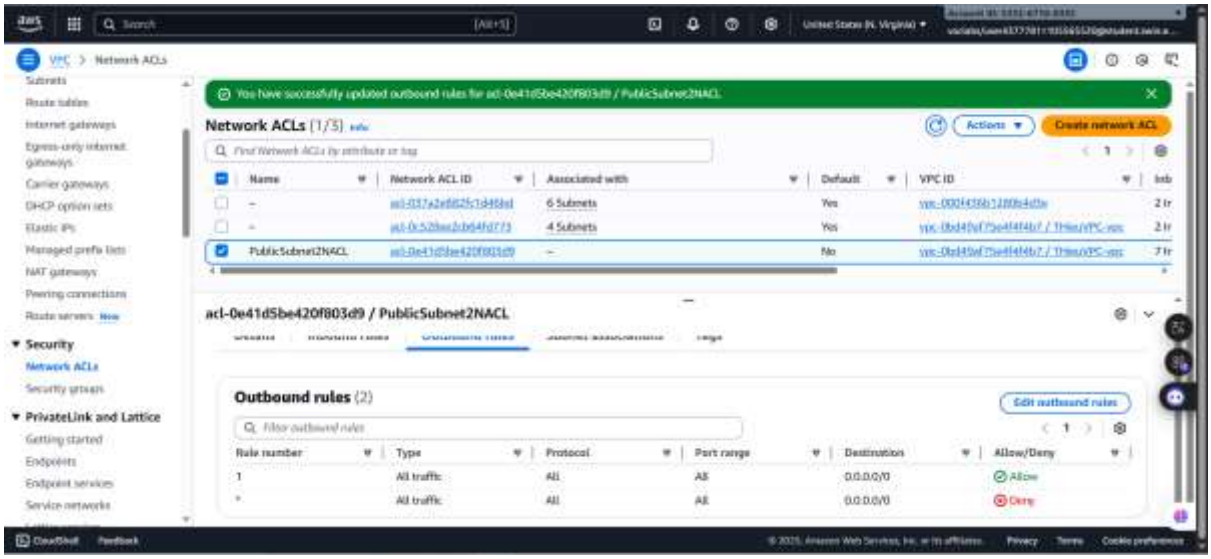


Figure 52: Outbound rule set successfully

Step 1.5.4: Next, select the **Subnet associations**

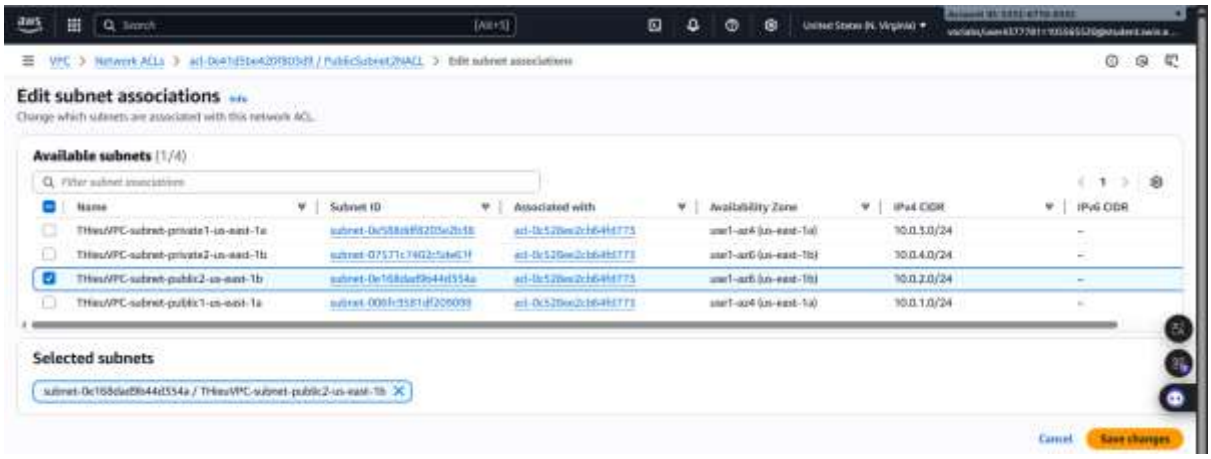


Figure 53: Subnet associations configuration

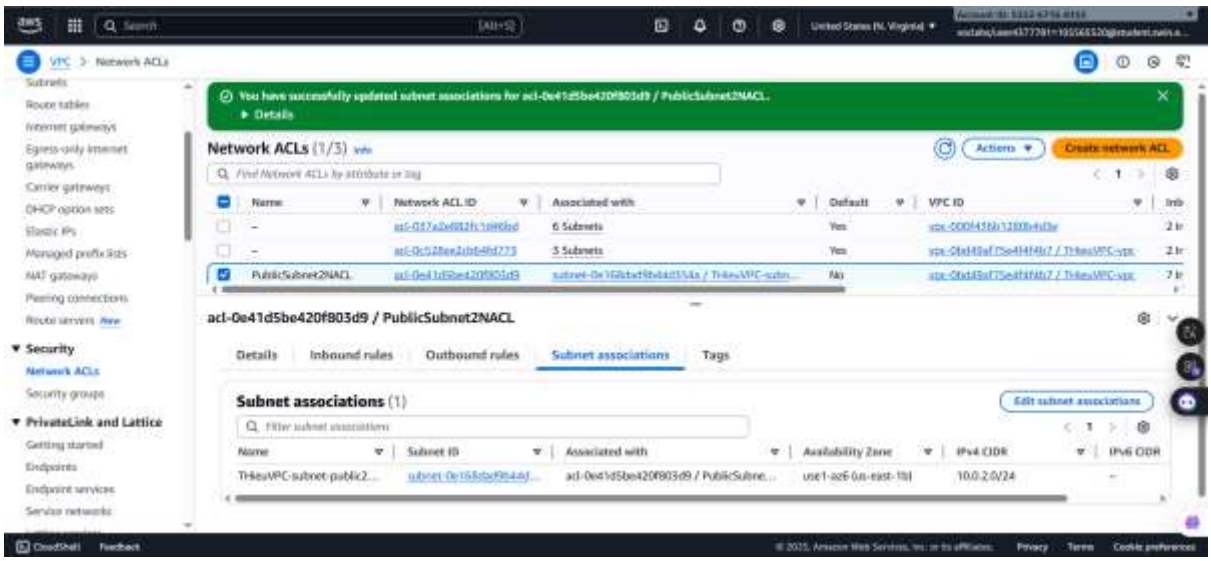


Figure 54: Subnet associations successfully

Part II. Infrastructure deployment

Functional requirements of Photo Album website Your Photo Album website must have the following functional requirements.

Step 2.1: Photo storage

Step 2.1.1: Search and select the **S3** and choose **Create bucket** button

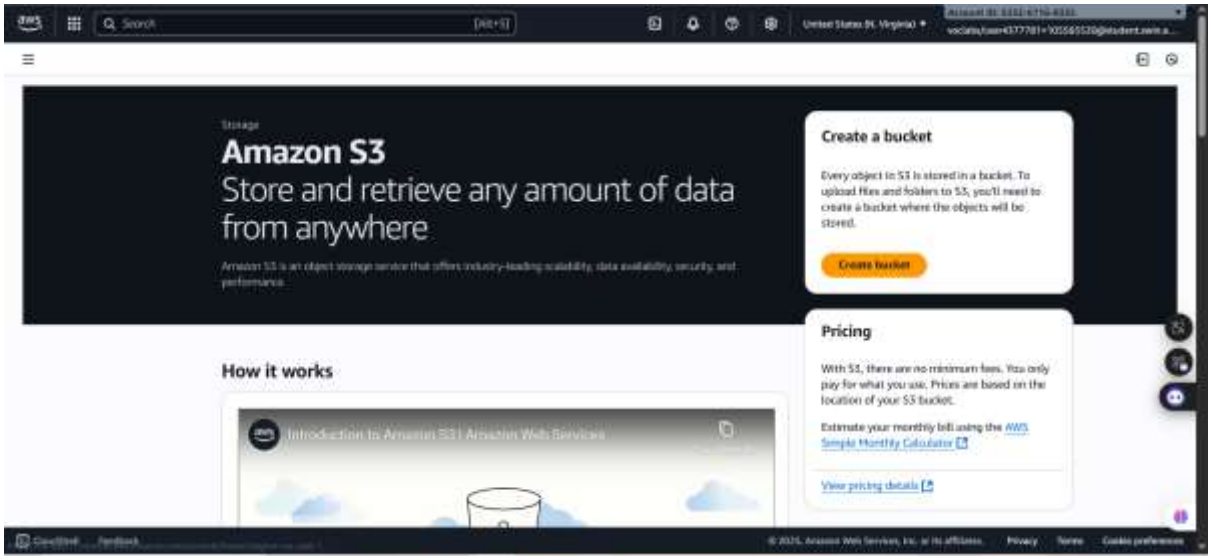


Figure 55: S3 Homepage

Step 2.1.2: Configure with following settings:

+ **Bucket Name:** ghieu-assign1b-bucket
+ **Object Ownership:** ACLs disabled (recommended)
+ **Block Public Access settings for this bucket:** Tick the box *Turning off block all public access might result in this bucket and the objects within becoming public*

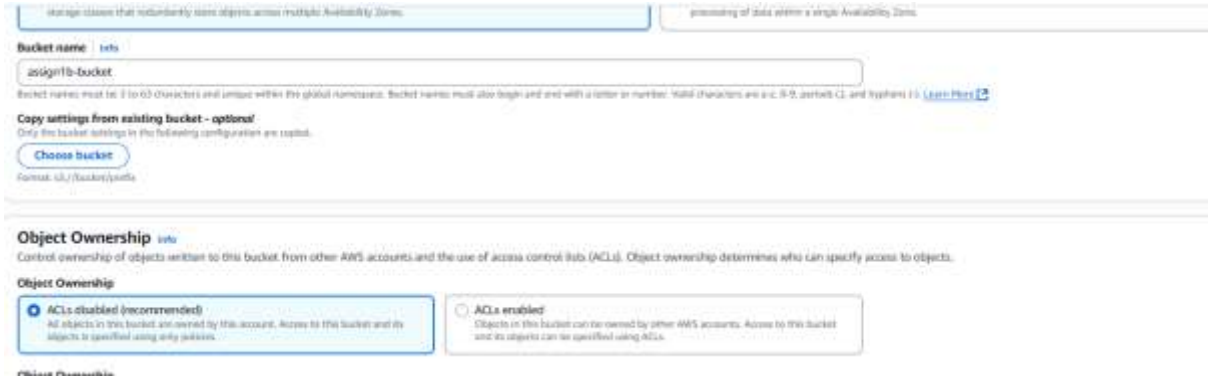


Figure 54: General Configuration

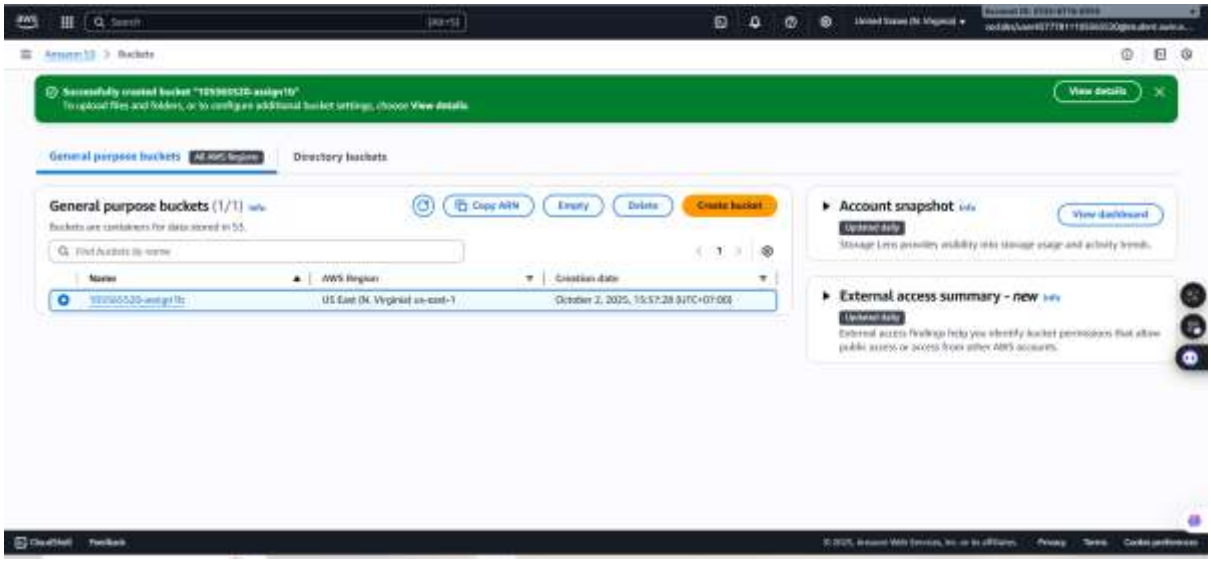


Figure 55: Created successfully

Step 2.1.3: After creating successfully, select itself, and choose tab **Permissions.** Next, select the **Edit** button in **Bucket Policy** part.

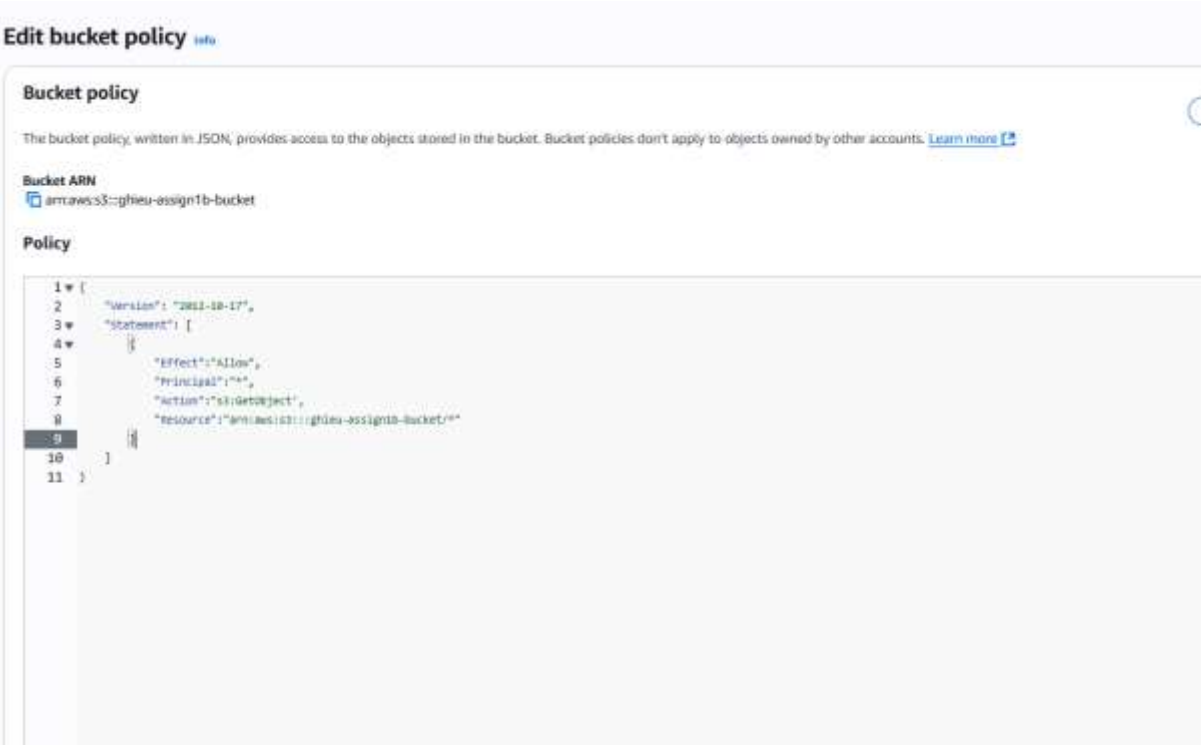


Figure 56: Bucket policy configuration

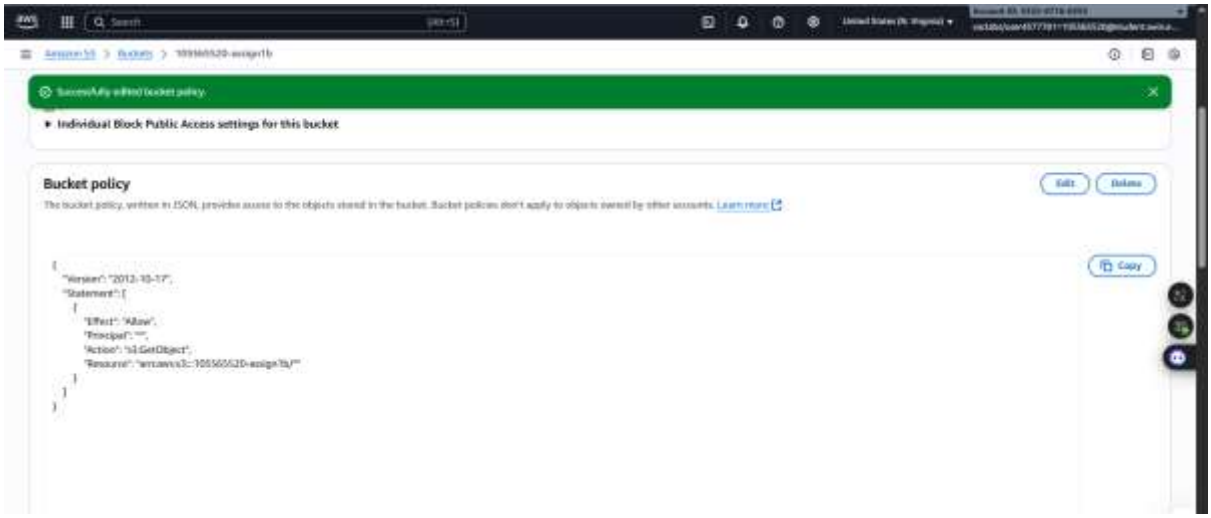


Figure 57: Saved changed successfully

Step 2.1.4: Then, upload two example of images to S3

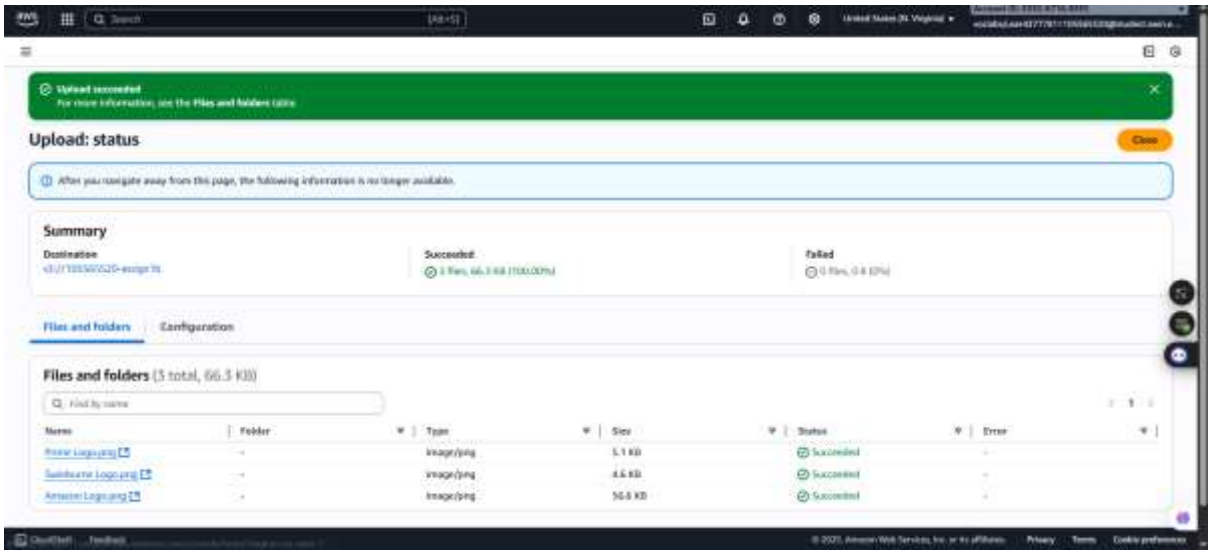


Figure 58: Uploaded successfully

Step 2.1.5: Open the phpAdmin and add value

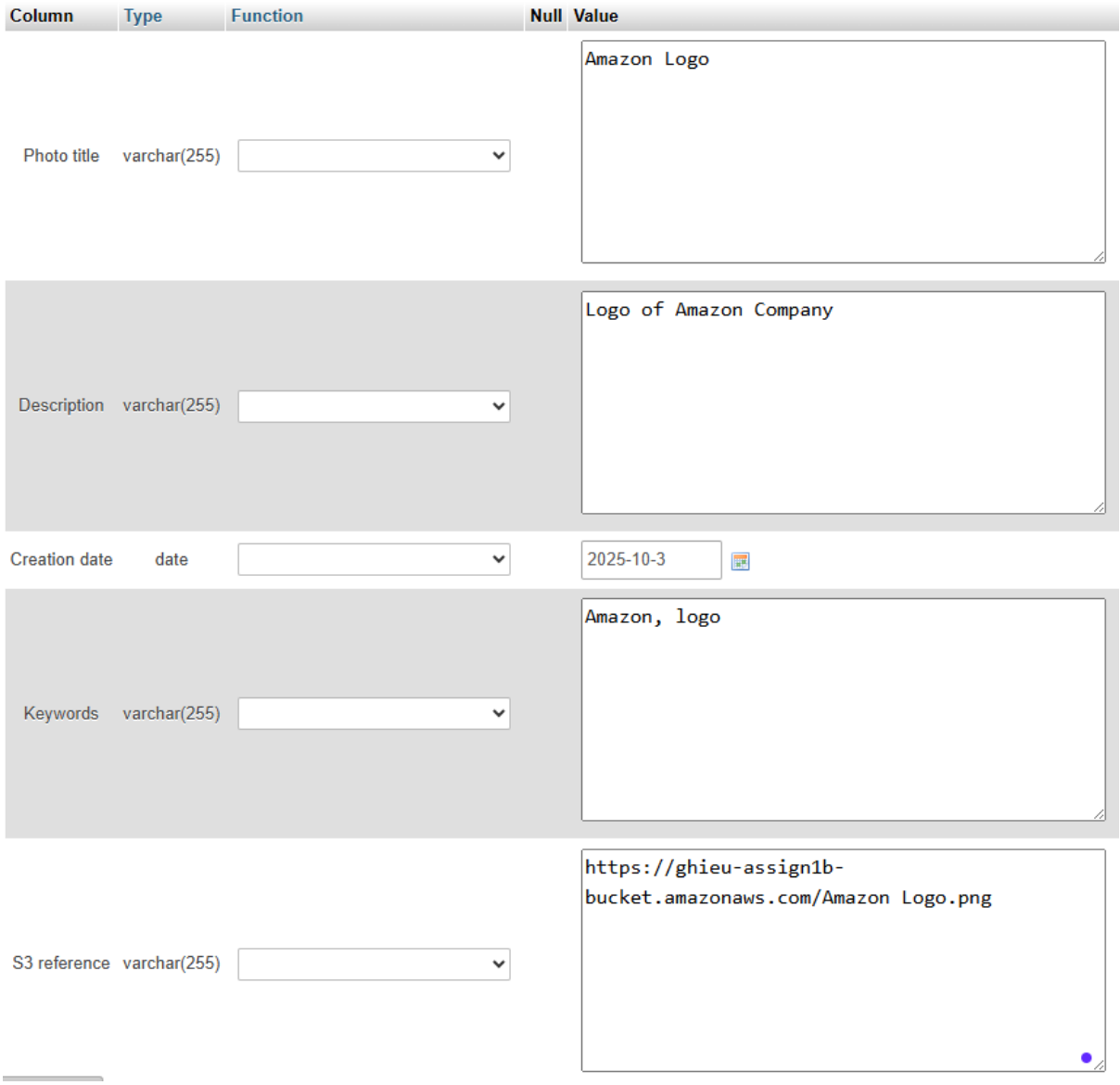


Figure 59: Amazon Logo configuration

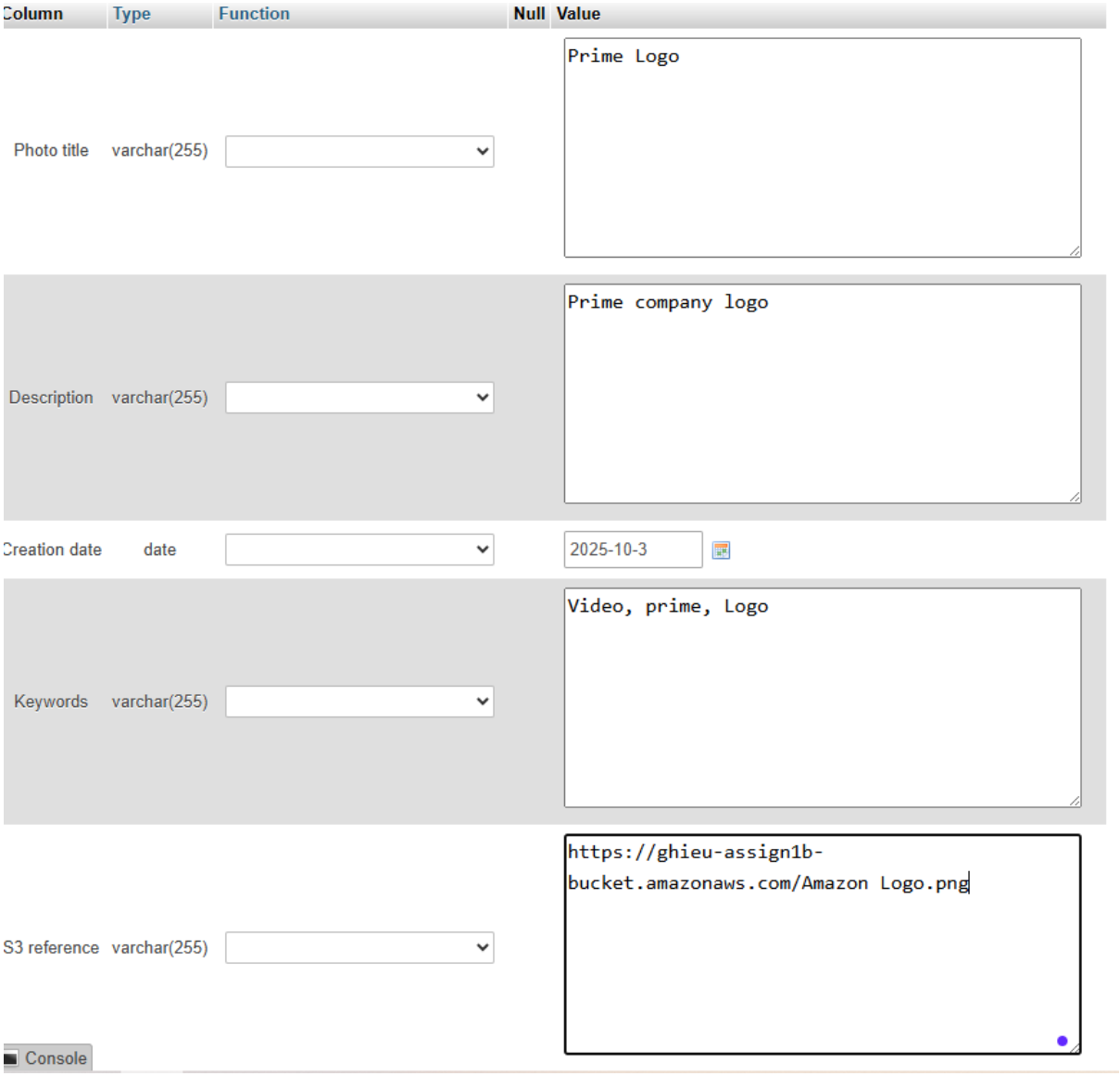


Figure 60: Prime Logo configuration

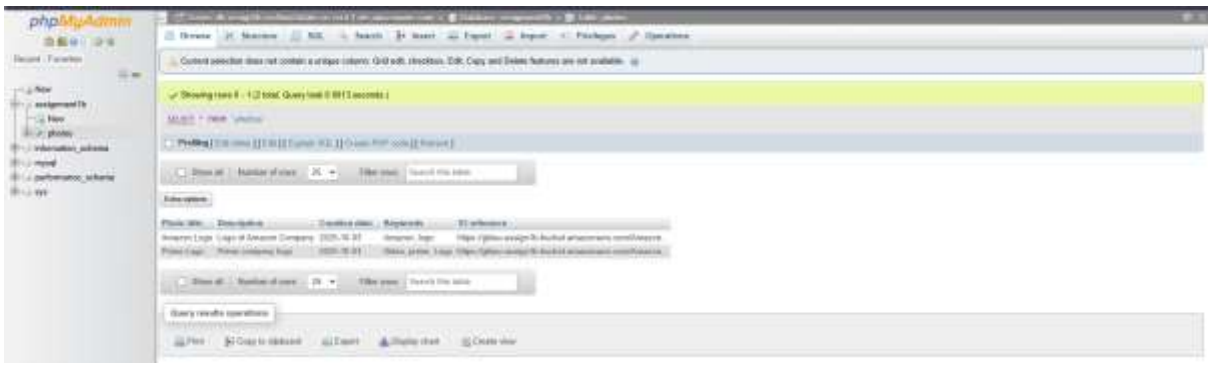


Figure 61: Configuration created successfully

Step 2.1.6: Open the WinSCP to configure the datafile **constants.php**

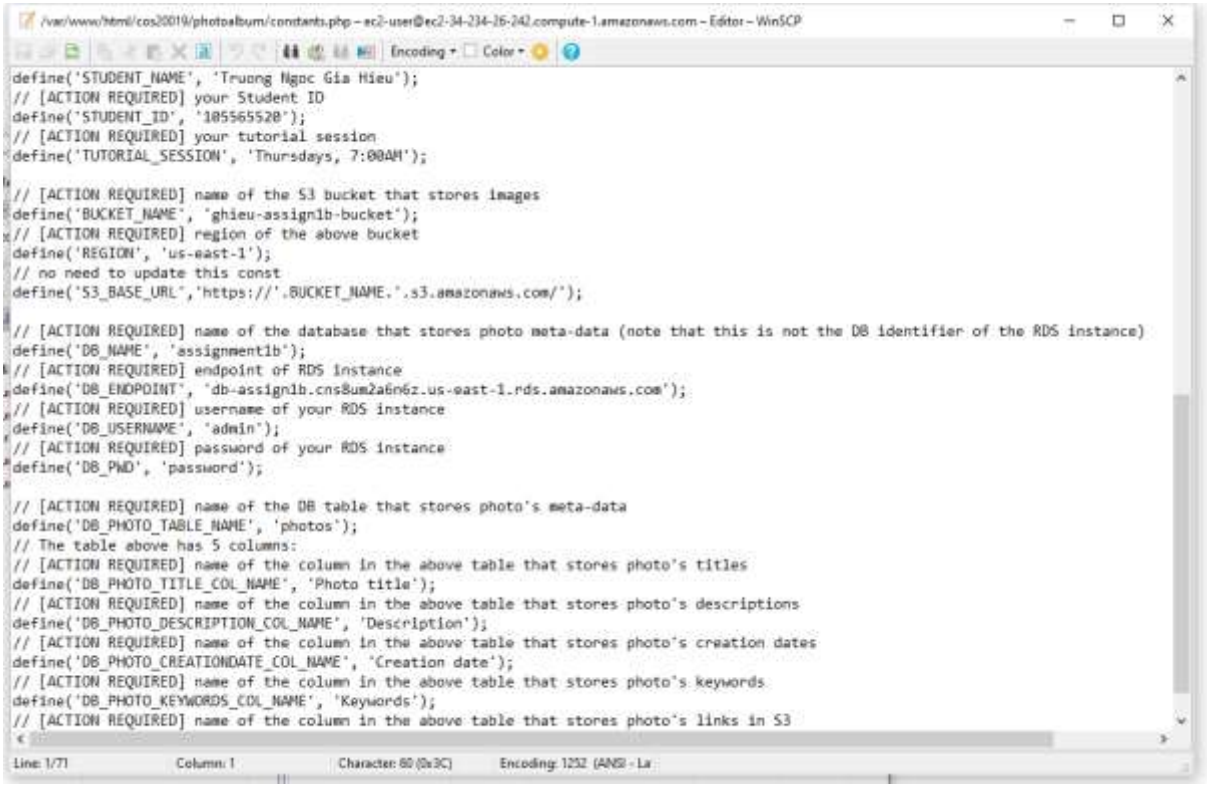


Figure 62: File configuration

Step 2.1.7: By pasting the link below which allow the result displays
/http://34.234.26.242/cos20019/photoalbum/album.php

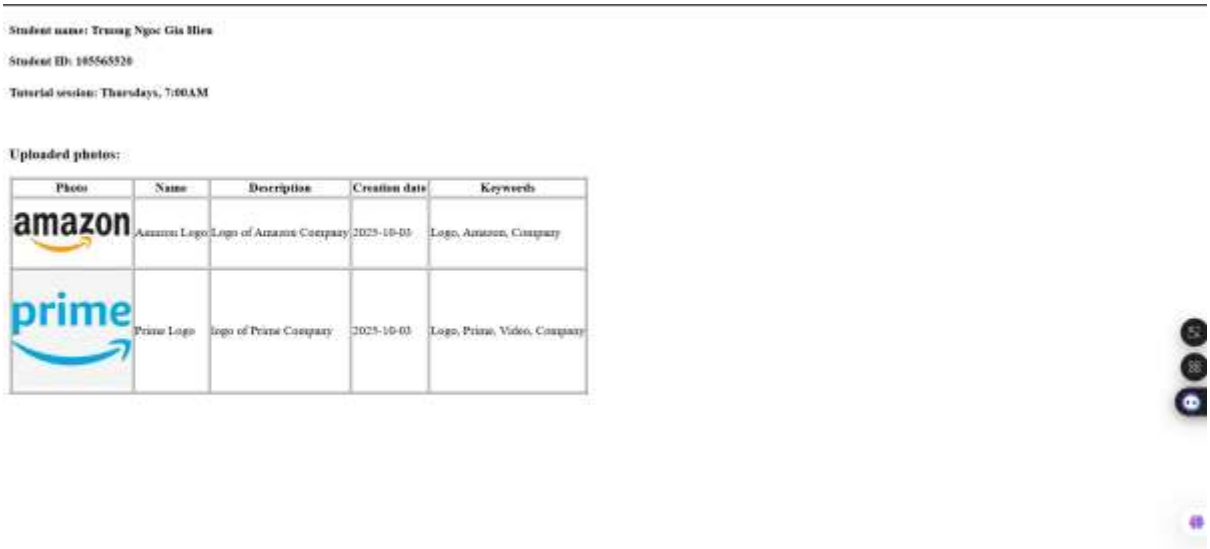


Figure 62: Completed task

Part III. Conclusion

- In completing this assignment, I successfully designed and deployed a secure cloud-based photo album application on AWS. The process involved creating a custom VPC with both public and private subnets, configuring routing tables, and applying appropriate security

groups and network ACLs to enforce the principle of least privilege. I deployed an Apache web server on an EC2 instance, integrated an RDS MySQL database for photo meta-data storage, and configured an S3 bucket for hosting the photo objects with proper access policies.

- Through this implementation, I gained hands-on experience in setting up a scalable and secure cloud infrastructure, troubleshooting configuration issues, and ensuring the seamless connection between different AWS services. The project not only strengthened my technical understanding of cloud architecture but also highlighted the importance of security, resource management, and documentation in a real-world deployment.

- Overall, this assignment provided me with valuable insights into cloud computing practices and gave me confidence in applying AWS services to build reliable web applications.