

COS20019

Cloud Computing Architecture

Assignment 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

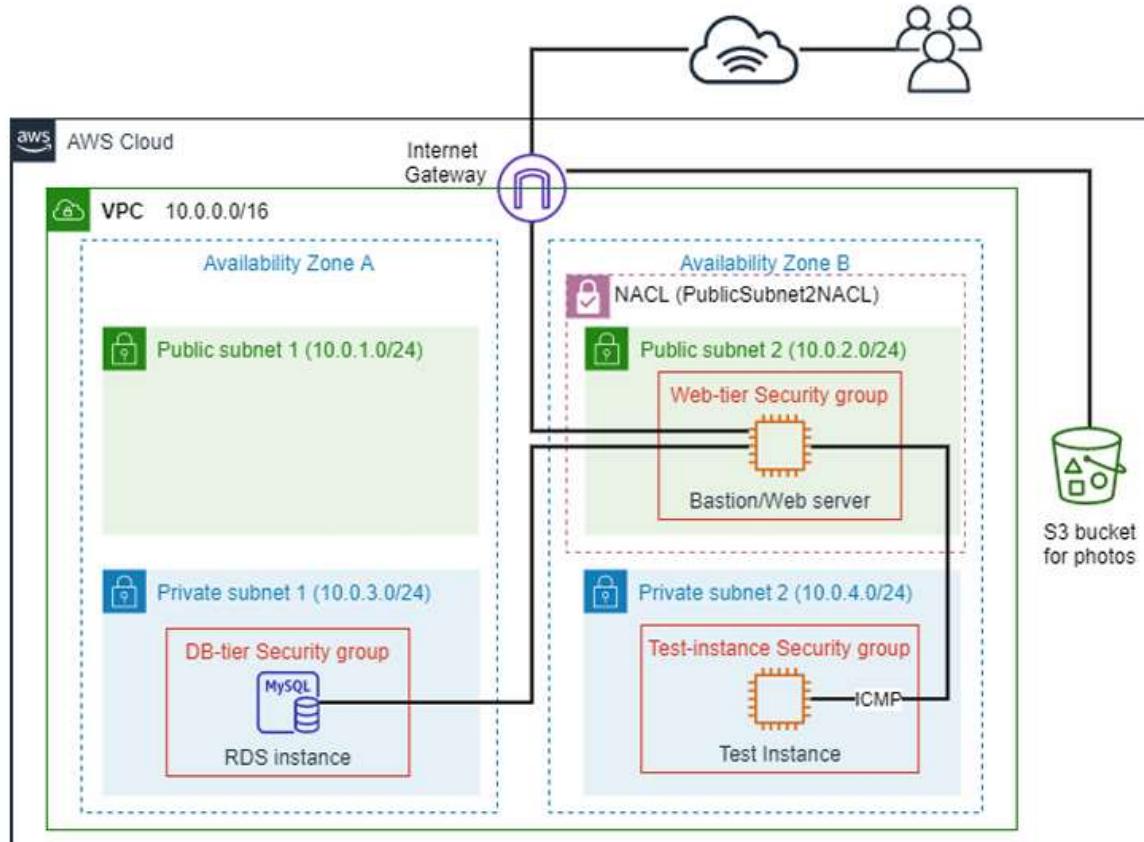
This assignment has the following objectives:

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

VPC settings

Resources to create [Info](#)
Create only the VPC resource or the VPC and other networking resources.

VPC only VPC and more

Name tag auto-generation [Info](#)
Enter a value for the Name tag. This value will be used to auto-generate Name tags for all resources in the VPC.

Auto-generate
GHieuVPC

IPv4 CIDR block [Info](#)
Determine the starting IP and the size of your VPC using CIDR notation.

10.0.0.0/16 65.536 IPs

CIDR block size must be between /16 and /28.

IPv6 CIDR block [Info](#)
 No IPv6 CIDR block
 Amazon-provided IPv6 CIDR block

Tenancy [Info](#)
Default ▾

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)

Number of Availability Zones (AZs) [Info](#)

Choose the number of AZs in which to provision subnets. We recommend at least two AZs for high availability.

1 | 2 | 3

▼ Customize AZs

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

▼ Customize subnets CIDR blocks

Public subnet CIDR block in us-east-1a

10.0.1.0/24 256 IPs

Public subnet CIDR block in us-east-1b

10.0.2.0/24 256 IPs

Private subnet CIDR block in us-east-1a

10.0.3.0/24 256 IPs

Private subnet CIDR block in us-east-1b

10.0.4.0/24 256 IPs

◀ ▶ ⌂ ⌃

Figure 4: IP address configuration

Preview**Figure 5: Preview check**

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

Create VPC workflow**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**”.

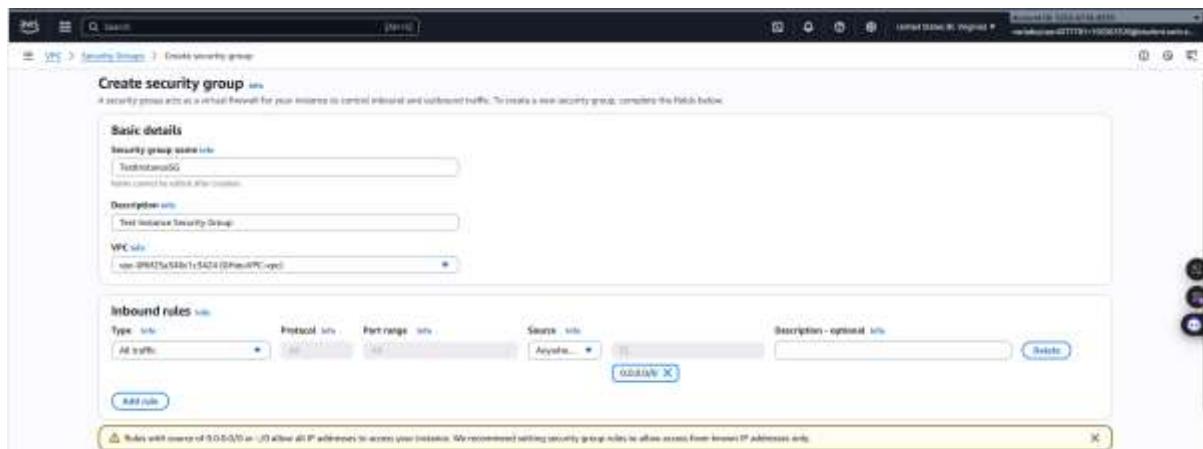


Figure 7: TestInstanceSG configuration

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

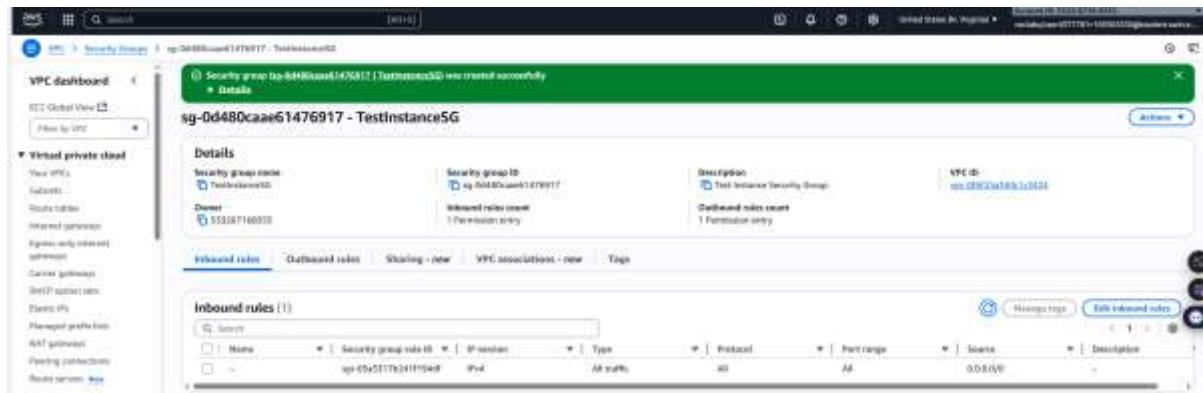


Figure 8: TestInstanceSG created successfully

Step 1.2.3: Similarly to WebServerSG

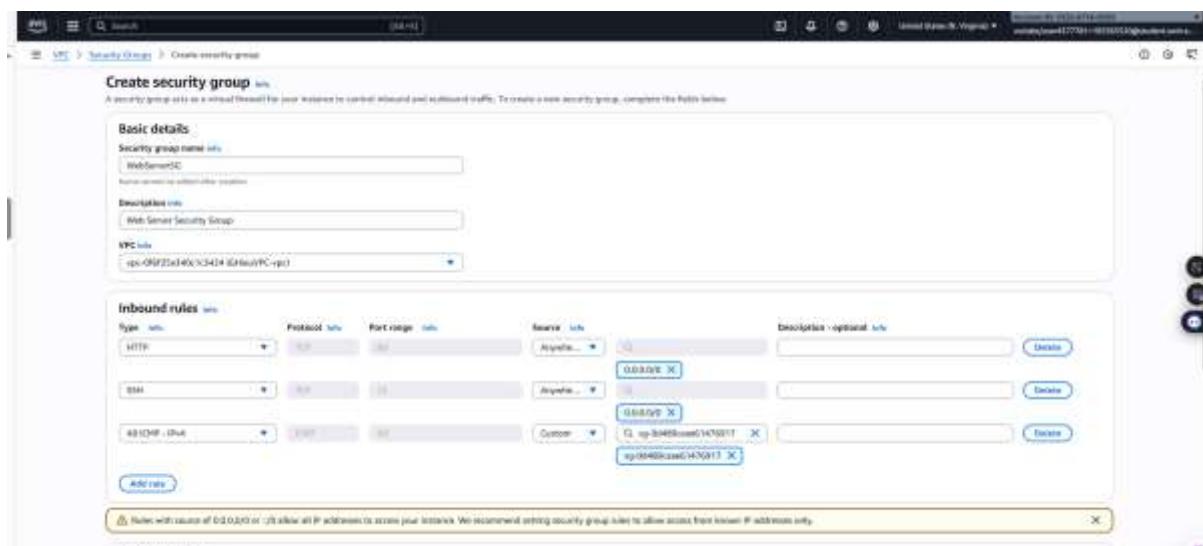


Figure 9: WebServerSG configuration

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

The screenshot shows the AWS VPC dashboard with the 'Security Groups' section selected. A table lists several security groups, including 'WebServerSG' which was just created. The 'WebServerSG' row shows its details: Name (sg-025738c8ef001736), Security group ID (sg-025738c8ef001736), VPC (vpc-0000428012345678), and Description (Web Server Security Group). It also shows the inbound rules: All traffic (All TCP - IPv4) from 0.0.0.0/0 to port 80, SSH (TCP) from 0.0.0.0/0 to port 22, and HTTP (TCP) from 0.0.0.0/0 to port 80.

Figure 10: WebServerSG created successfully

Step 1.2.5: Finally, created the DBServerSG

The screenshot shows the 'Create security group' wizard. In the 'Basic details' step, the security group name is set to 'DBServerSG' and the description is 'Database Server Security Group'. The VPC is selected as 'vpc-0000428012345678'. In the 'Inbound rules' step, three rules are defined: All traffic (All TCP - IPv4) from 0.0.0.0/0 to port 3306, SSH (TCP) from 0.0.0.0/0 to port 22, and MySQL (TCP) from 0.0.0.0/0 to port 3306.

Figure 11: DBServerSG configuration

The screenshot shows the AWS VPC dashboard with the 'Security Groups' section selected. A table lists several security groups, including 'DBServerSG' which was just created. The 'DBServerSG' row shows its details: Name (sg-0924cd1c5bd778fa5a), Security group ID (sg-0924cd1c5bd778fa5a), Owner (105565520), and Description (Database Server Security Group). It also shows the inbound rules: MySQL (TCP) from 0.0.0.0/0 to port 3306.

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

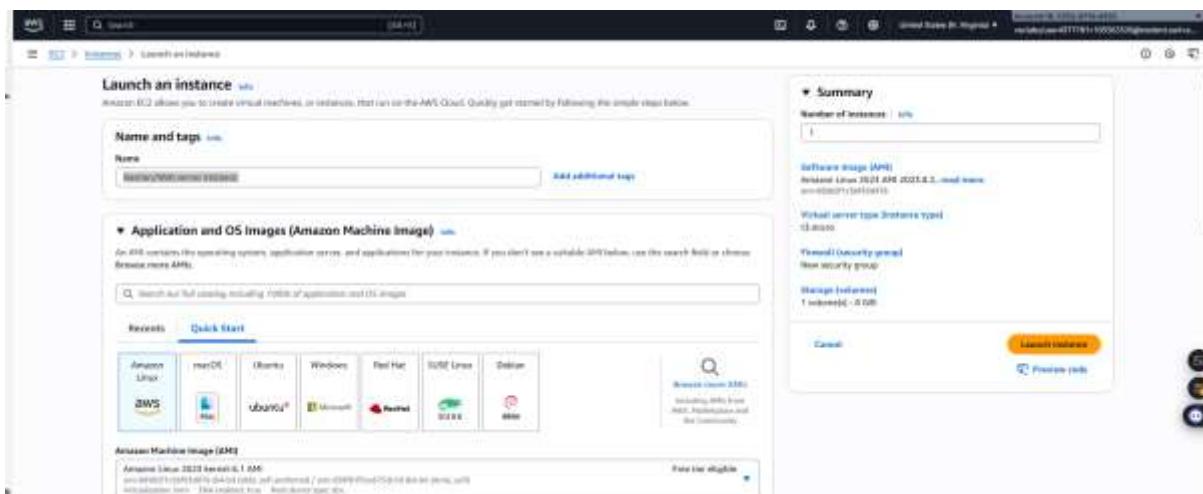


Figure 13: Name and AMI configuration

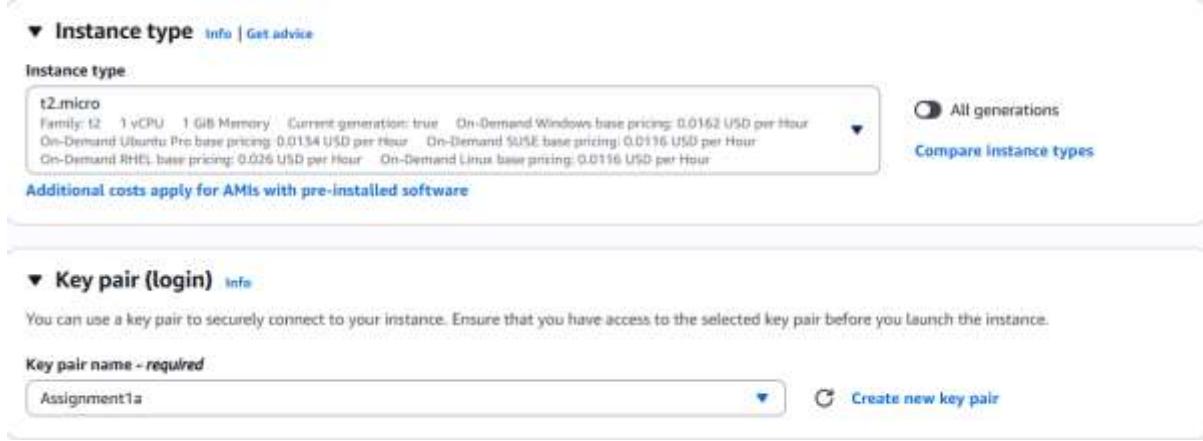


Figure 14: Instance and key pair configuration

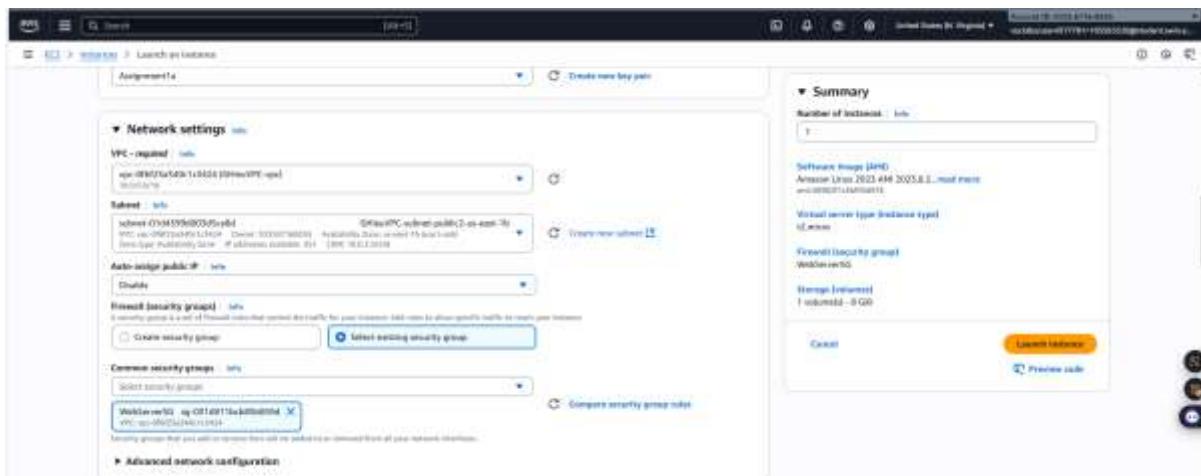


Figure 15: Network settings configuration

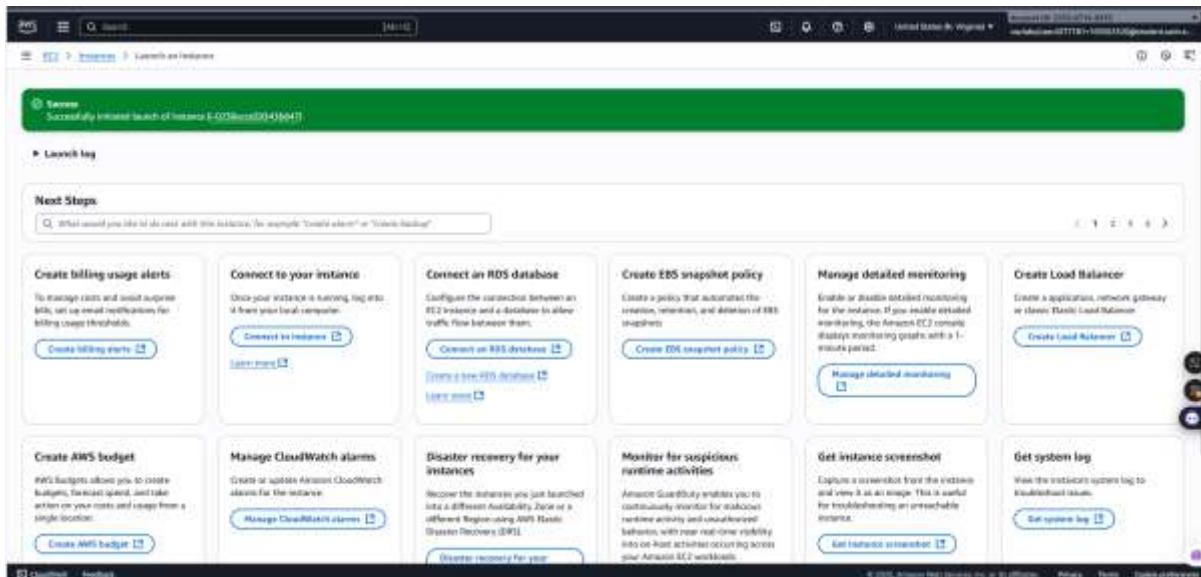


Figure 16: Instance created successfully

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

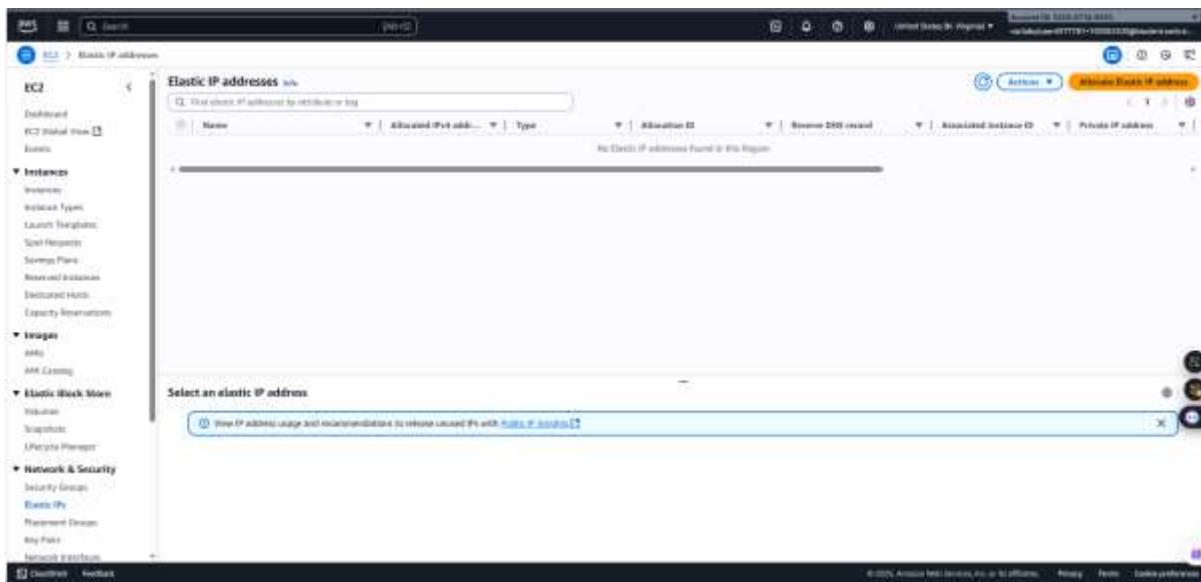


Figure 17: Elastic IPs window

Step 1.3.3: After clicking the button, choose Allocate

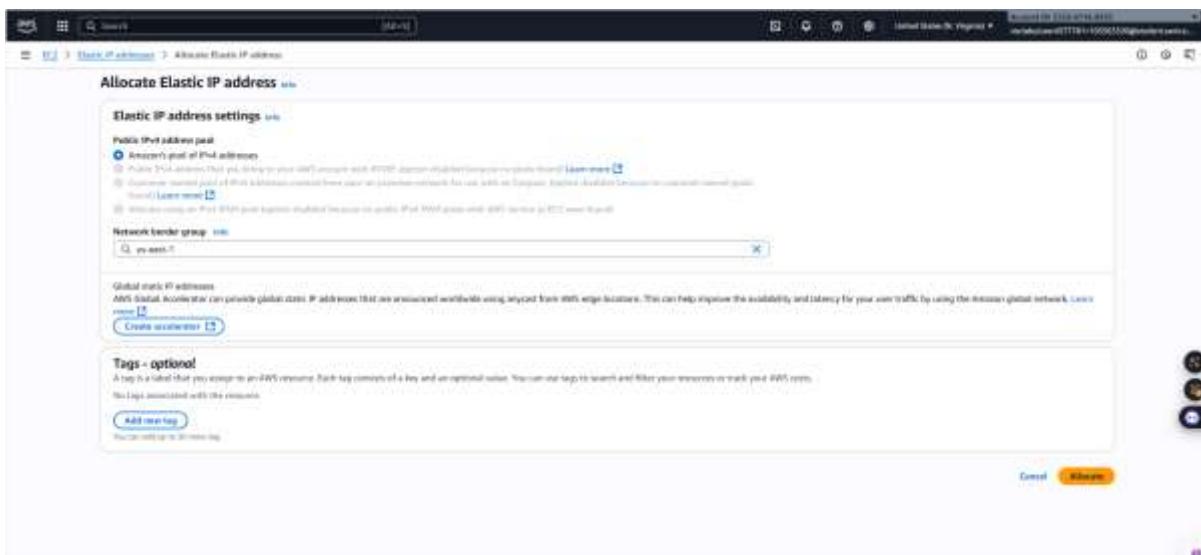


Figure 18: Allocate configuration

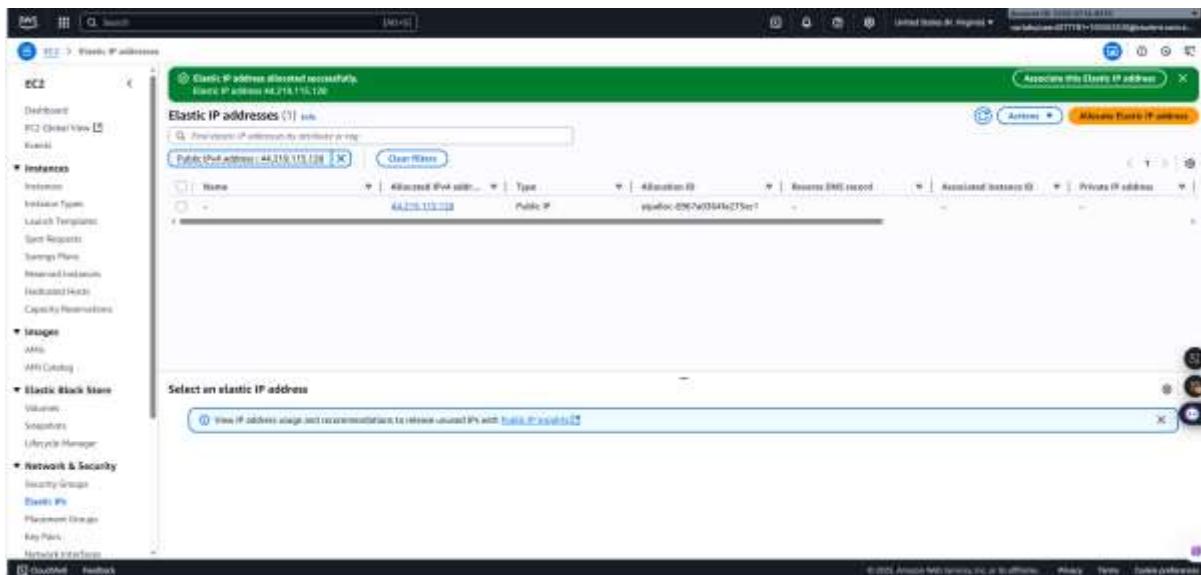


Figure 19: Allocate created successfully

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

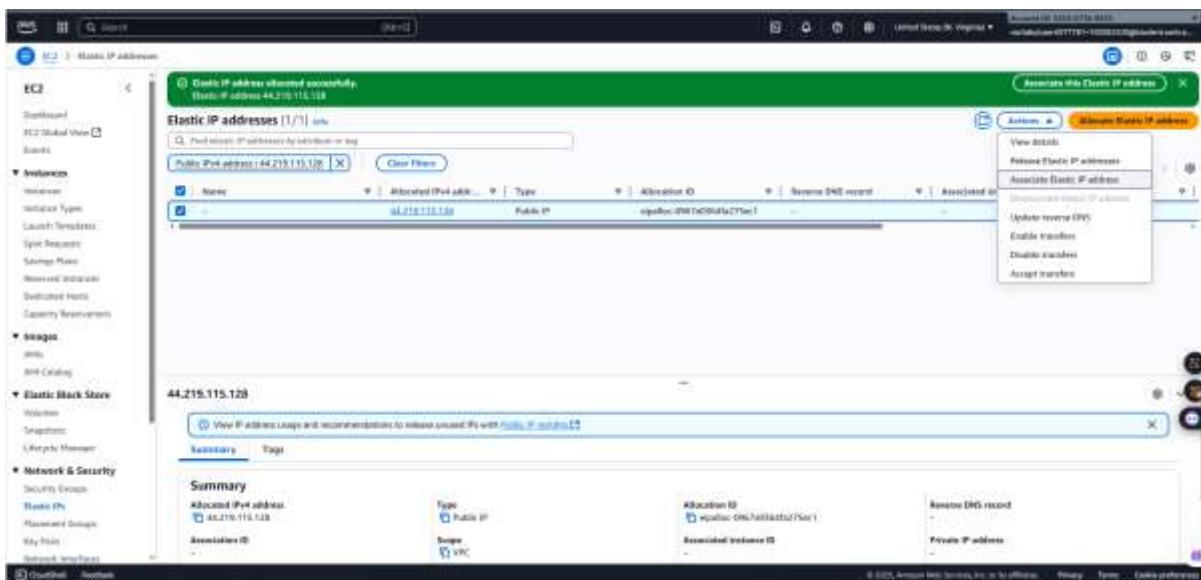


Figure 20: Action selection

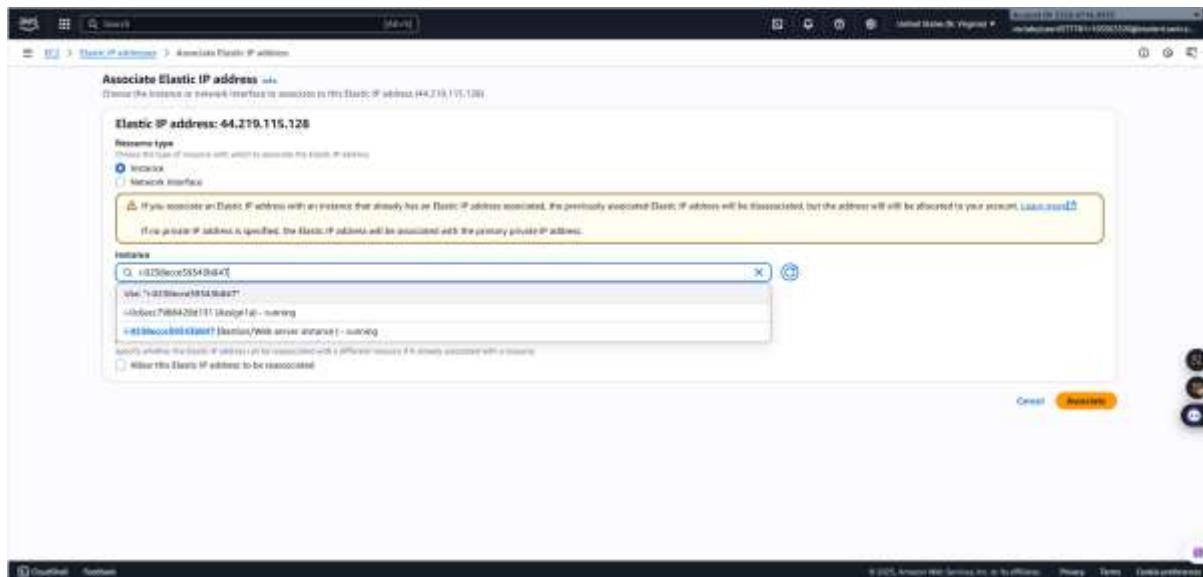


Figure 21: Allocation configuration

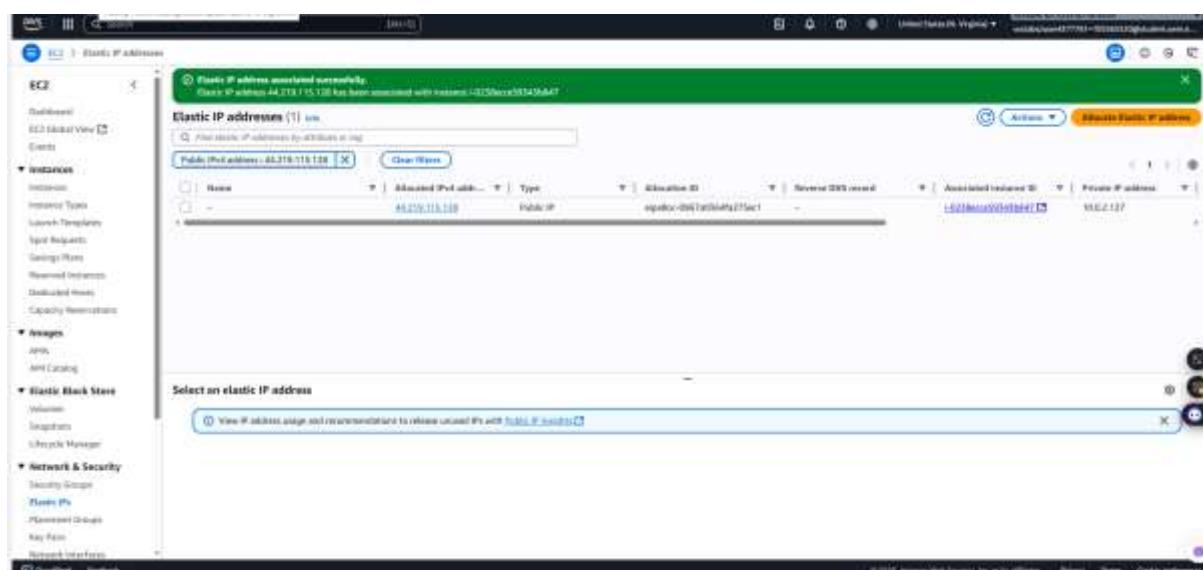


Figure 22: Allocation created successfully

Step 1.3.5: Create Test Instance

Name and tags Info

Name
Test Instance Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recents **Quick Start**

| | | | | | | |
|---|--|---|--|--|---|---|
|  Amazon Linux |  macOS |  Ubuntu |  Windows |  Red Hat |  SUSE Linux |  Debian |
|---|--|---|--|--|---|---|

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI Free tier eligible

arn:aws:ami:ami-08982f1c5bf93d976 (64-bit (x86), uefi-preferred) / ami-039ff81f5ce6752b10 (64-bit (Arm), uefi)
Virtualization: hvm - ENA enabled: true - Root device type: ebs

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Figure 23: Test Instance's Name and AMI Configuration

Instance type Info | Get advice

Instance type

t2.micro
Family: t2 - 1 vCPU - 1 GiB Memory - Current generation: true - On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour - On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour - On-Demand Linux base pricing: 0.0116 USD per Hour

All generations Compare instance types

Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required
Assignment1a

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

This screenshot shows the 'Network settings' section of the AWS Management Console for a specific EC2 instance. It includes fields for VPC (selected: 'vpc-0f6f25a340c1c3424 (GHieuVPC-vpc)'), Subnet ('subnet-081b80f090b13efbc'), Auto-assign public IP ('Disable'), Firewall (security groups) ('Select existing security group'), and Common security groups ('TestInstanceSG sg-0d75070cac9fa722'). A note at the bottom states: 'Security groups that you add or remove here will be added to or removed from all your network interfaces.'

Figure 25: Test Instance's network settings configuration

This screenshot shows the 'Instances' page in the AWS Management Console. A green success message at the top reads: 'Success: Successfully started launch of instance i-0f6f25a340c1c3424'. Below this, there is a 'Launch log' section and a 'Next Steps' summary. The summary includes links for 'Create billing usage alerts', 'Connect to your instance', 'Connect an RDS database', 'Create EBS snapshot policy', 'Manage detailed monitoring', 'Create Load Balancer', 'Create AWS budget', 'Manage CloudWatch alarms', 'Disaster recovery for your instances', 'Monitor for suspicious runtime activities', 'Get instance screenshot', and 'Get system log'. The bottom of the page shows navigation links for 'Dashboard' and 'Health'.

Figure 26: Test Instance's created successfully

Step 1.3.6: Choose Instances to view all Instances

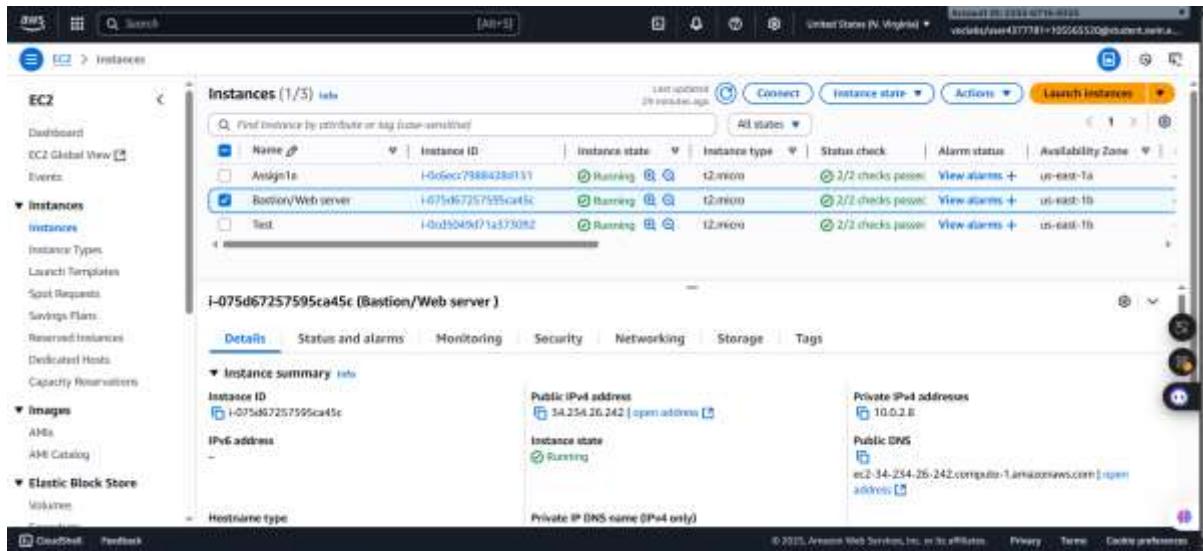


Figure 27: Instance board

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 from Test to Bastion instance constraint

```
[ec2-user@ip-10-0-2-8 ~]$ ssh -i ./Assignment1a.pem ec2-user@10.0.4.106
   _#
  ~\ _###_#      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 Bastion instance successfully

Step 1.4: RDS database instance

Step 1.4.1: Search and select the Aurora and RDS service

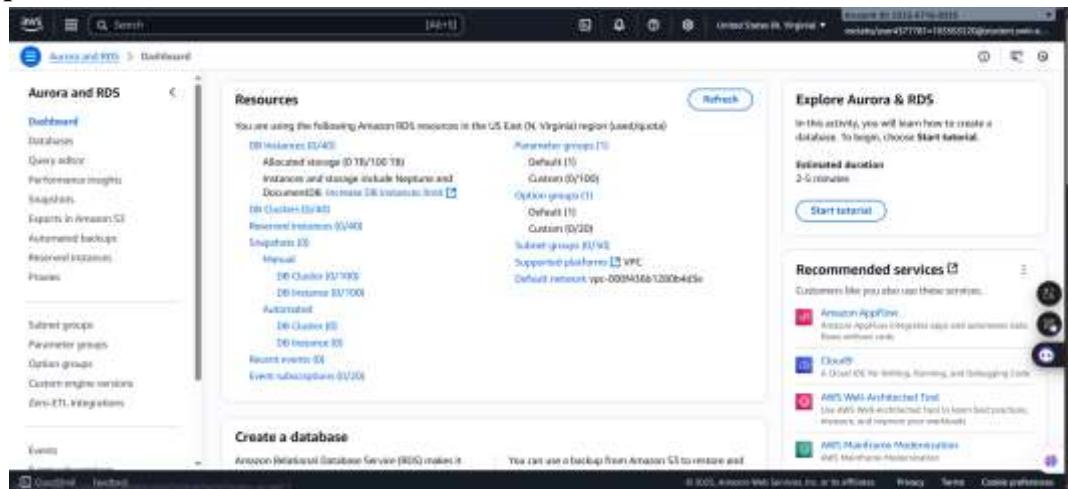


Figure 30: Aurora and RDS mainpage

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

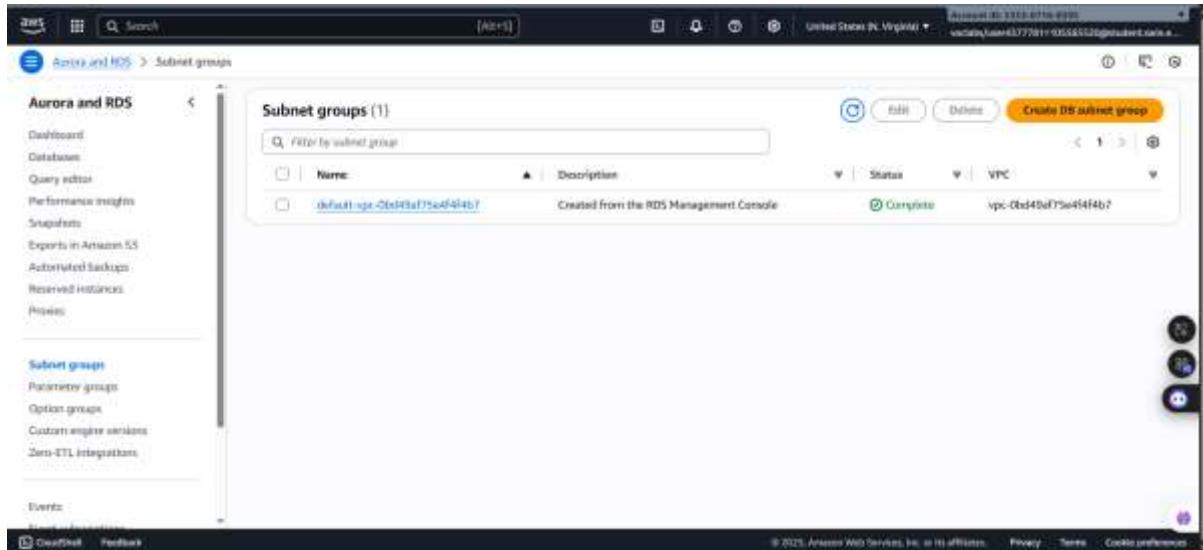


Figure 31: Create DB Subnet group subpage

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

The screenshot shows the 'Create DB subnet group' configuration page. It has a header 'Create DB subnet group' and a sub-header 'To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.' Below this is a section titled 'Subnet group details'.

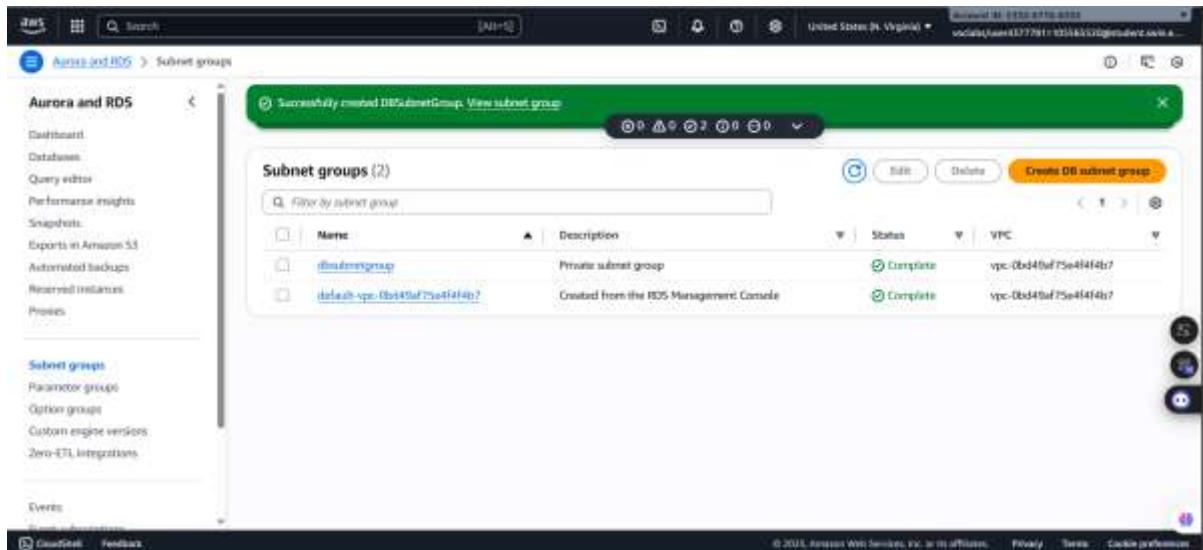
Name: DBSubnetGroup (Note: You won't be able to modify the name after your subnet group has been created.)

Description: Private subnet group

VPC: THieuVPC-vpc (vpc-0bd49af75e4f4b7) (Note: Choose a VPC identifier that corresponds to the subnet you want to use for your DB subnet group. You won't be able to change a different VPC identifier after your subnet group has been created.)

Figure 32: Subnet Group details configuration

| Availability zone | Subnet name | Subnet ID | CIDR block |
|-------------------|-------------------------------------|--------------------------|-------------|
| us-east-1a | ThieuVPC-subnet-private1-us-east-1a | subnet-0e588ddc8205e2b18 | 10.0.1.0/24 |
| us-east-1b | ThieuVPC-subnet-private2-us-east-1b | subnet-07531c7402e5de61f | 10.0.4.0/24 |

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

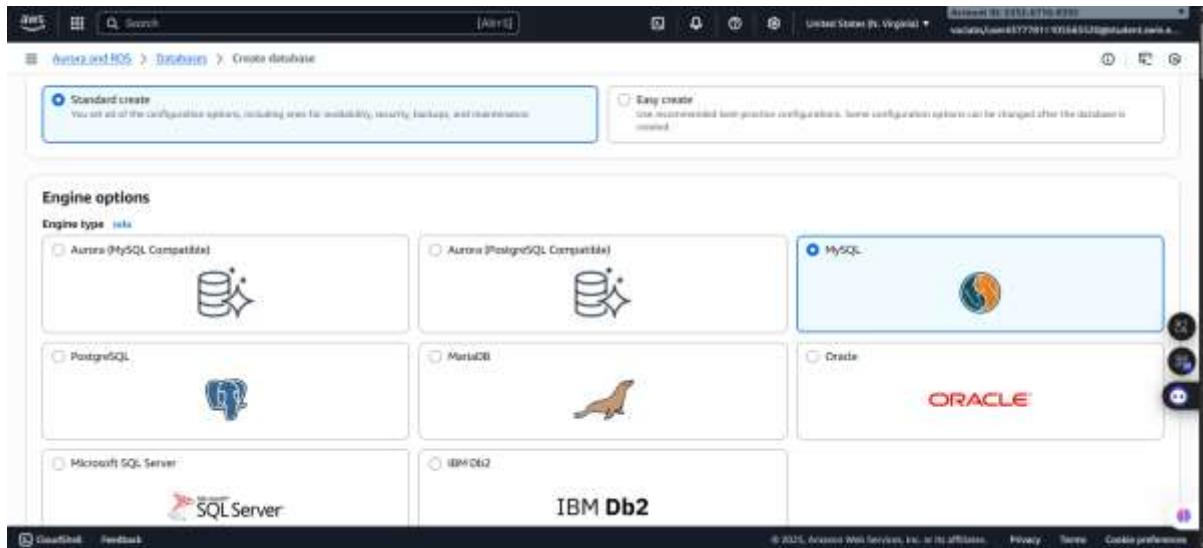


Figure 35: Database method and engine configuration

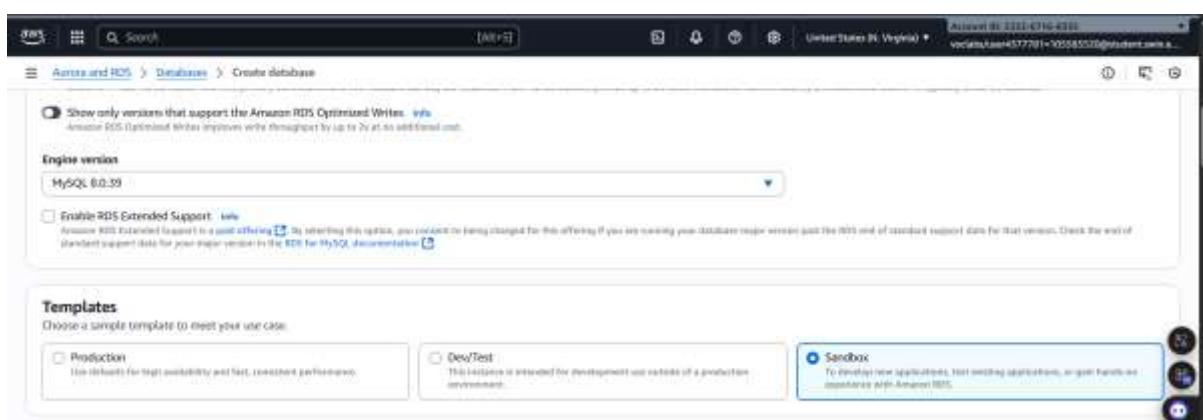


Figure 36: Engine version and Templates configuration

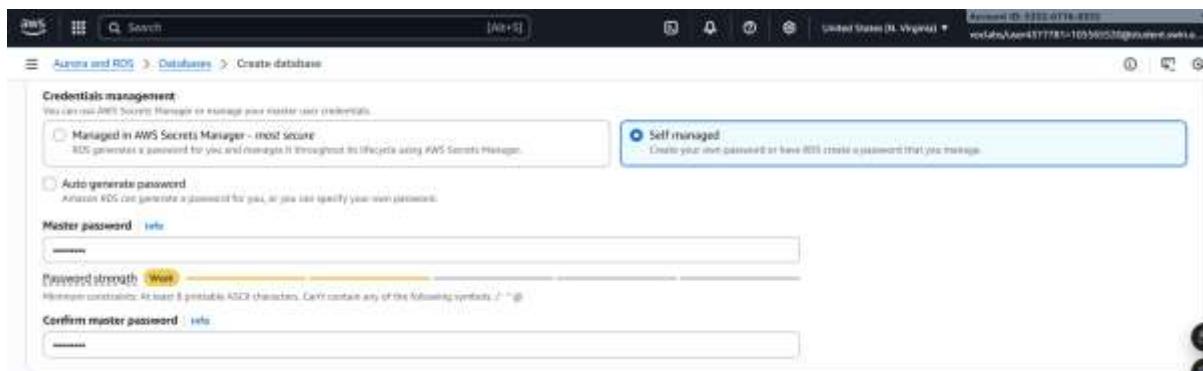


Figure 37: Hostname and password configuration

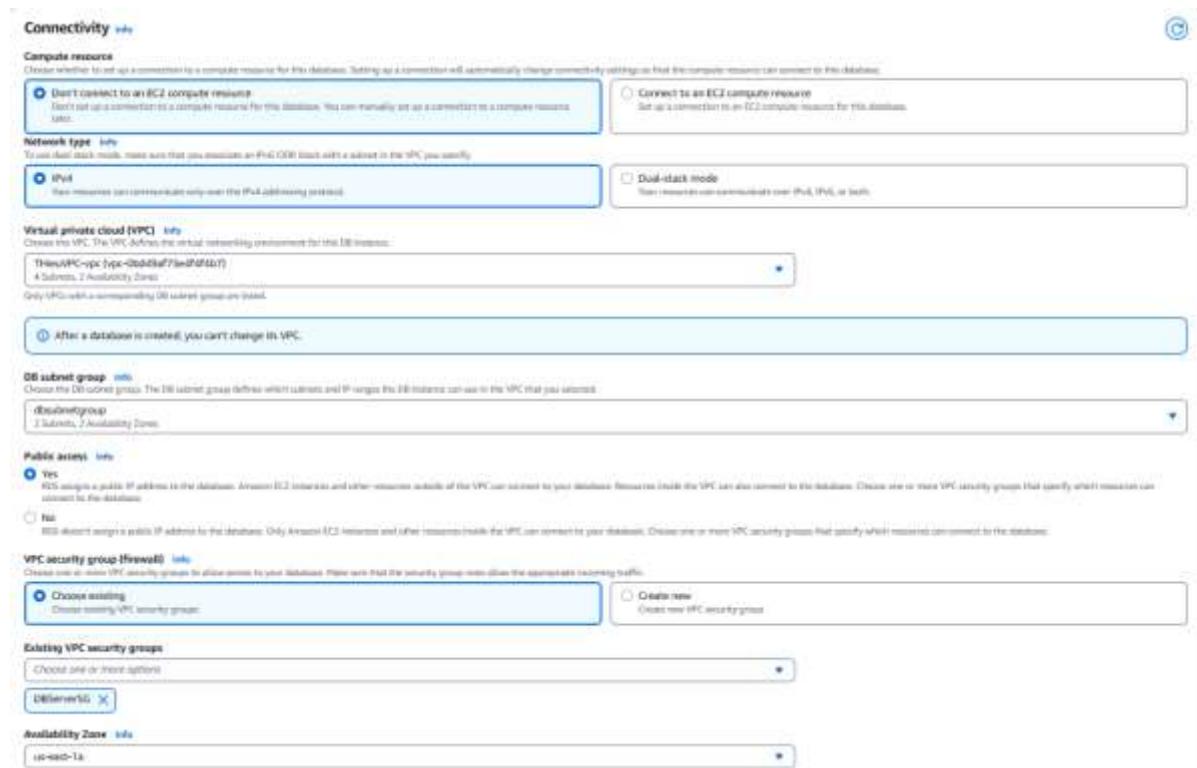


Figure 38: Connectivity configuration

Step 1.4.5: Open the Putty.exe, paste the public DNS link of the Bastion/Web server instance and upload the private key pair

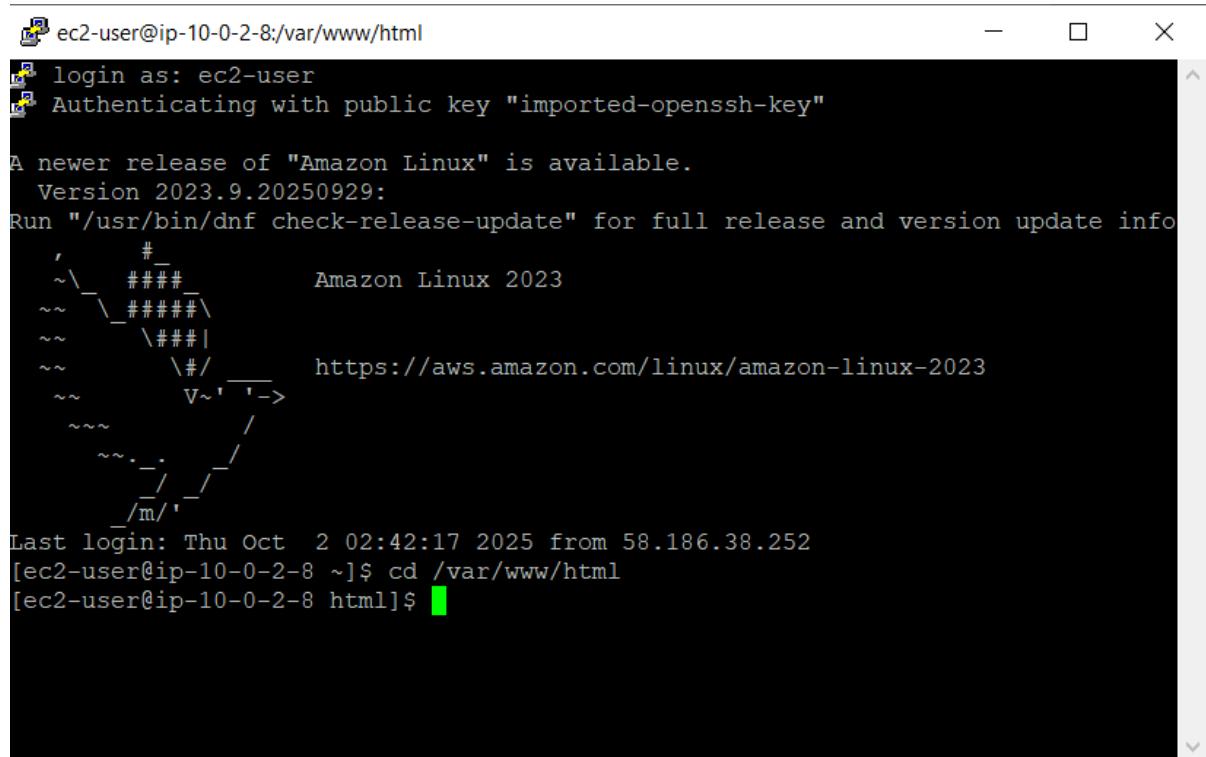
```
ec2-user@ip-10-0-2-8:~ - □ X
login as: ec2-user
Authenticating with public key "imported-openssh-key"

A newer release of "Amazon Linux" is available.
Version 2023.9.20250929:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,
# ~\_\#\#\#_ Amazon Linux 2023
~~\_\#\#\#\\
~~ \#\#\|
~~ \#/ https://aws.amazon.com/linux/amazon-linux-2023
~~ V~'`->
~~ /
~~ ./
~~ /`/
/m/,'-
Last login: Thu Oct 2 02:42:17 2025 from 58.186.38.252
[ec2-user@ip-10-0-2-8 ~]$
```

Figure 39: Login successfully

Step 1.4.6: Navigate to the Apache document root directory:

```
cd /var/www/html
```

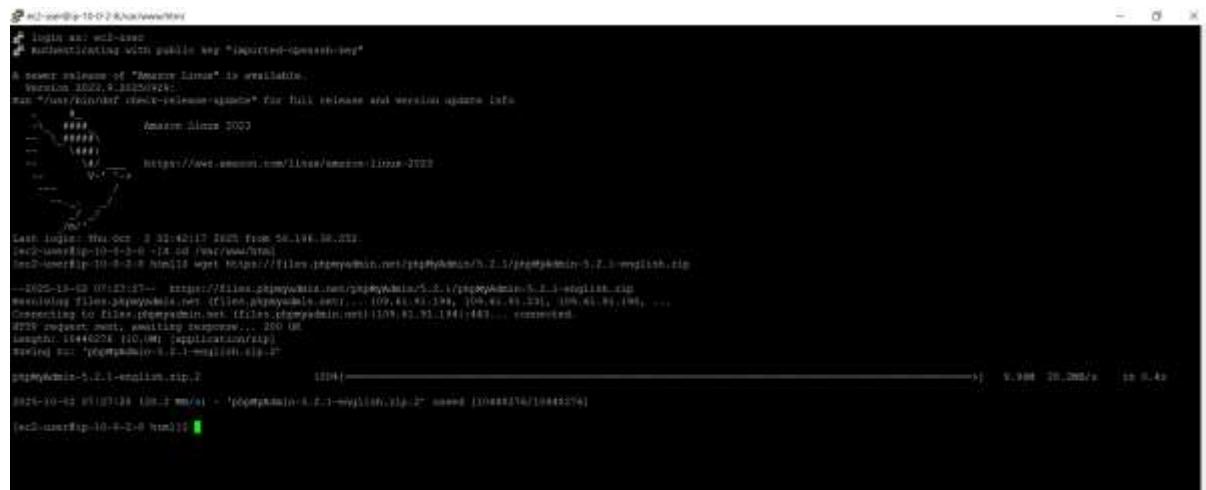


```
ec2-user@ip-10-0-2-8:~$ cd /var/www/html
[ec2-user@ip-10-0-2-8 ~]$
```

Figure 40: Changed directory successfully

Step 1.4.7: Download phpMyAdmin source file:

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



```
ec2-user@ip-10-0-2-8:~$ wget https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-english.zip
--2023-10-02 07:42:37-- https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-english.zip
Resolving files.phpmyadmin.net (files.phpmyadmin.net)... 109.61.61.194, 109.61.61.195, 109.61.61.196...
Connecting to files.phpmyadmin.net (files.phpmyadmin.net) (109.61.61.194:443)... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1644224 (1.0 MiB) [application/xzip]
Saving to: 'phpMyAdmin-5.2.1-english.zip'

phpMyAdmin-5.2.1-english.zip          100%[=====]  1234      28.288/s    in 6.4s
[ec2-user@ip-10-0-2-8 ~]$
```

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

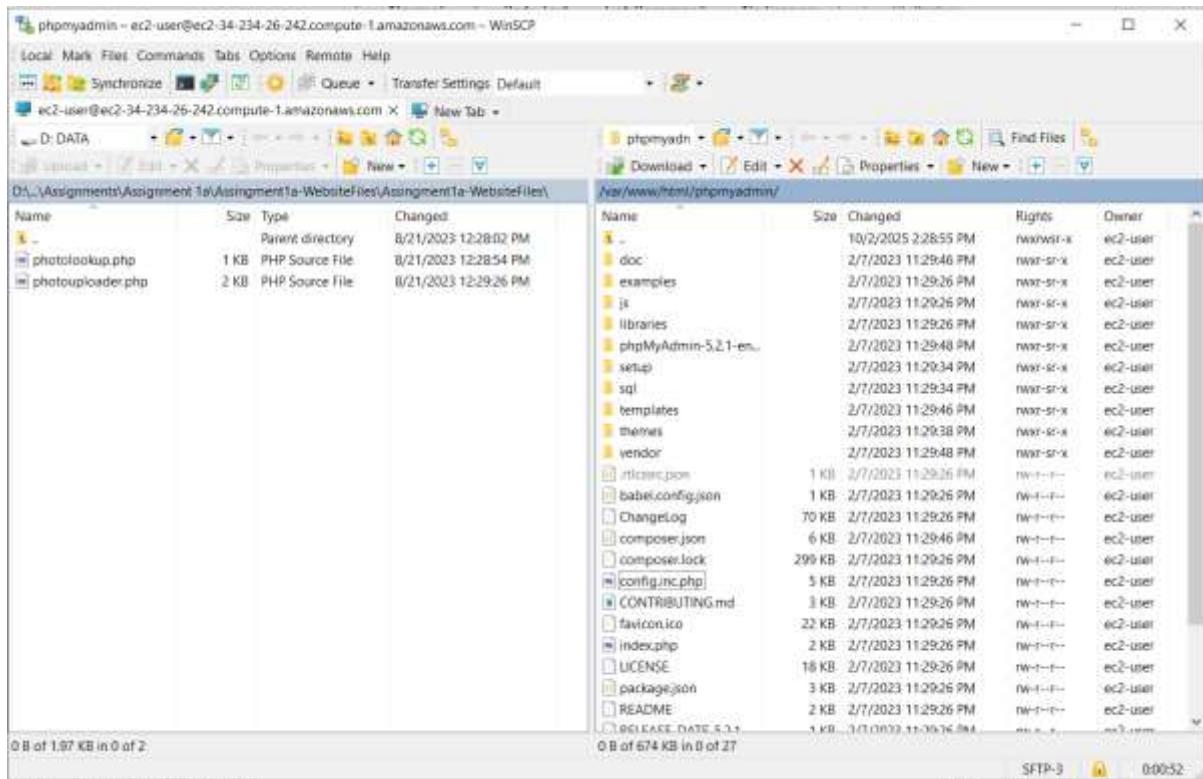


Figure 43: NamedChange successfully

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.

```

#!/var/www/html/phpmyadmin/config.inc.php - ec2-user@ec2-34-234-26-242.compute-1.amazonaws.com - Editor - WinSCP
Encoding: Color
/* or at <https://docs.phpmyadmin.net/>.
*/
declare(strict_types=1);

/**
 * This is needed for cookie based authentication to encrypt the cookie.
 * Needs to be a 32-bytes long string of random bytes. See FAQ 2.10.
 */
$cfg['blowfish_secret'] = ''; /* YOU MUST FILL IN THIS FOR COOKIE AUTH! */

/**
 * Servers configuration
 */
$i = 0;

/**
 * First server
 */
$i++;
/* Authentication type */
$cfg['Servers'][$i]['auth_type'] = 'cookie';
/* Server parameters */
$cfg['Servers'][$i]['host'] = 'assignment1b-database.cns8um2a6n6z.us-east-1.rds.amazonaws.com';
$cfg['Servers'][$i]['compress'] = false;
$cfg['Servers'][$i]['AllowNoPassword'] = false;

/**
 * phpMyAdmin configuration storage settings.
*/
/* User used to manipulate with storage */
// $cfg['Servers'][$i]['controlhost'] = '';
// $cfg['Servers'][$i]['controlport'] = '';
// $cfg['Servers'][$i]['controluser'] = 'pma';

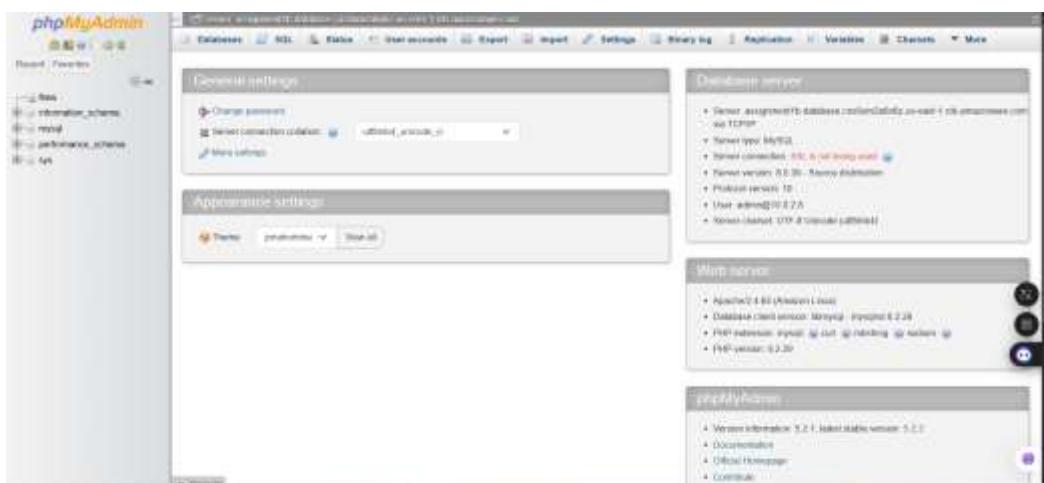
```

Line: 30/180 Column: 94 Character: 39 (0x27) Encoding: 1252 (ANSI - Lz - Modified)

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)

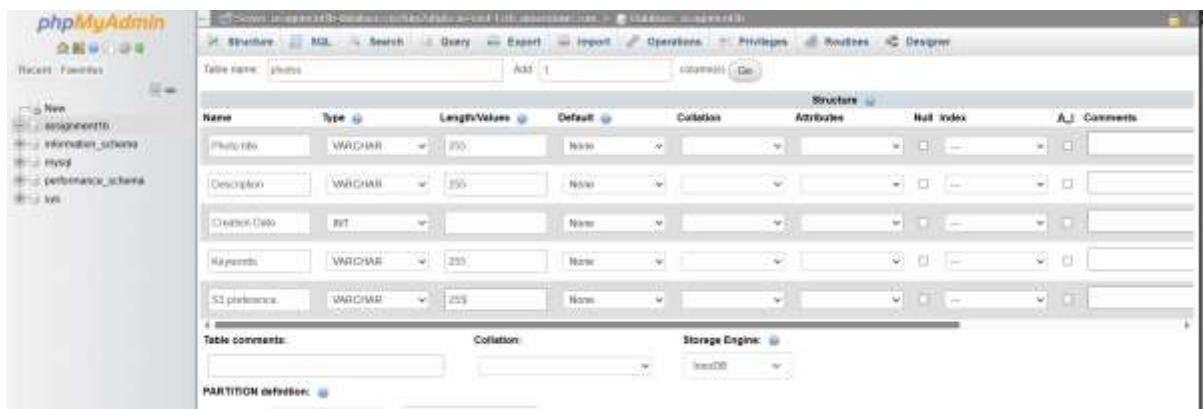


Figure 46: Table configuration



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

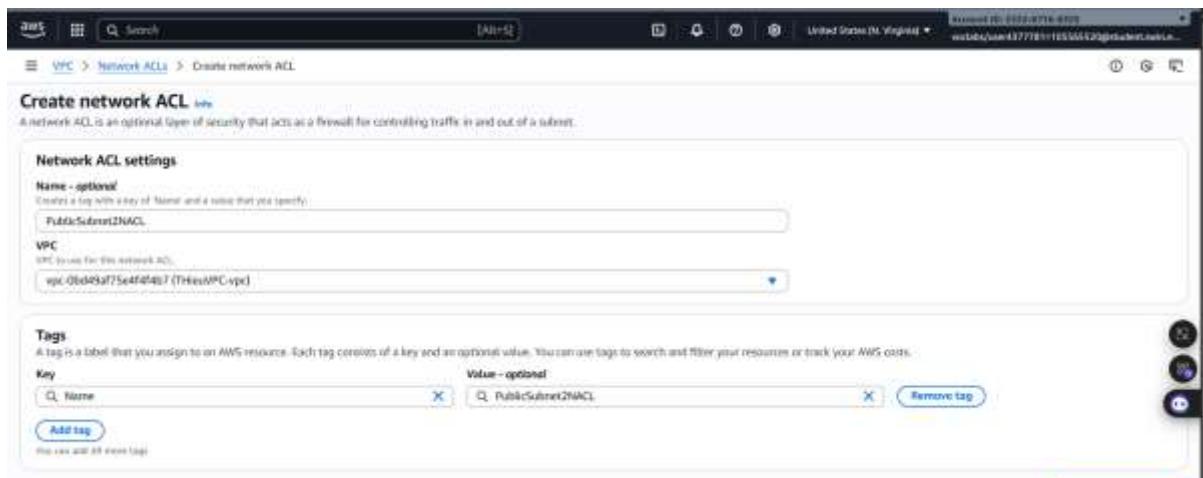


Figure 48: Basic configuration

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

- + must ALLOW SSH(22) traffic 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.

| Rule number | Type info | Protocol info | Port range info | Source info | Allow/Deny info |
|-------------|-----------------|---------------|-----------------|-------------|-----------------|
| 1 | SSH (22) | TCP (6) | 22 | 0.0.0.0/0 | Allow |
| 2 | All ICMP - IPv4 | ICMP (1) | All | 10.0.4.0/24 | Allow |
| 3 | HTTP (80) | TCP (6) | 80 | 0.0.0.0/0 | Allow |
| 4 | HTTPS (443) | TCP (6) | 443 | 0.0.0.0/0 | Allow |
| 5 | All TCP | TCP (6) | All | 10.0.3.0/24 | Allow |
| 6 | All traffic | TCP (6) | All | 10.0.4.0/24 | Allow |

Figure 49: Inbound configuration

| Name | Network ACL ID | Associated with | Default | VPC ID |
|--------------------------|-----------------------------|-----------------|---------|------------------------------------|
| ml-047a2e082fc1e6fb0 | - | 6 Subnets | Yes | vpc-20041bb1280c42d |
| ml-0c57bae2b1846f773 | - | 4 Subnets | Yes | vpc-2bd41175e4ff4b7 / ThreeVPC-vpn |
| PublicSubnet2NACL | ml-0e41d5be420f803d9 | - | No | vpc-2bd41175e4ff4b7 / ThreeVPC-vpn |

| Rule number | Type | Protocol | Port range | Source | Allow/Deny |
|-------------|-----------------|----------|------------|-------------|------------|
| 1 | SSH [22] | TCP (6) | 22 | 0.0.0.0/0 | Allow |
| 2 | All ICMP - IPv4 | ICMP (1) | All | 10.0.4.0/24 | Allow |
| 3 | HTTP [80] | TCP (6) | 80 | 0.0.0.0/0 | Allow |
| 4 | HTTPS [443] | TCP (6) | 443 | 0.0.0.0/0 | Allow |
| 5 | All TCP | TCP (6) | All | 10.0.3.0/24 | Allow |

Figure 50: Inbound rule set successfully

Step 1.5.3: Then, edit the outbound rules

| Rule number | Type info | Protocol info | Port range info | Destination info | Allow/Deny info |
|-------------|-------------|---------------|-----------------|------------------|-----------------|
| 1 | All traffic | TCP (6) | All | 0.0.0.0/0 | Allow |
| | All traffic | TCP (6) | All | 0.0.0.0/0 | Deny |

Figure 51: Outbound rule configuration

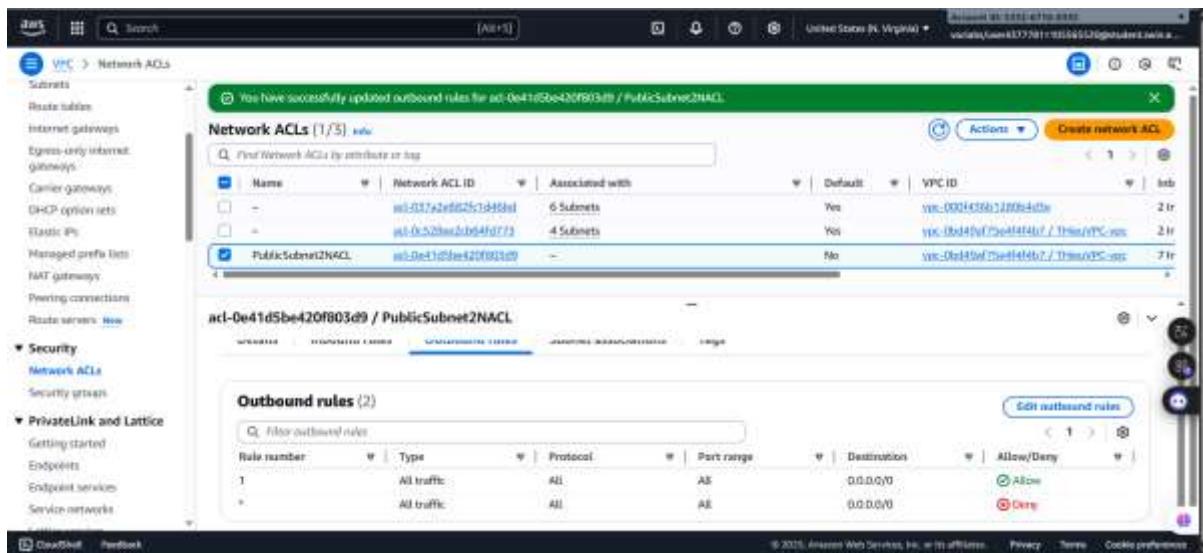


Figure 52: Outbound rule set successfully

Step 1.5.4: Next, select the Subnet associations

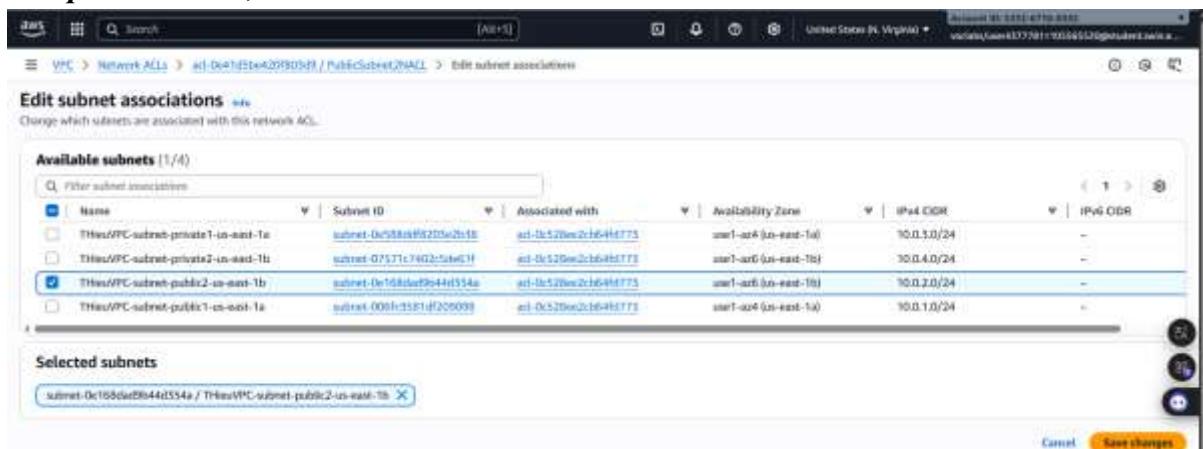


Figure 53: Subnet associations configuration

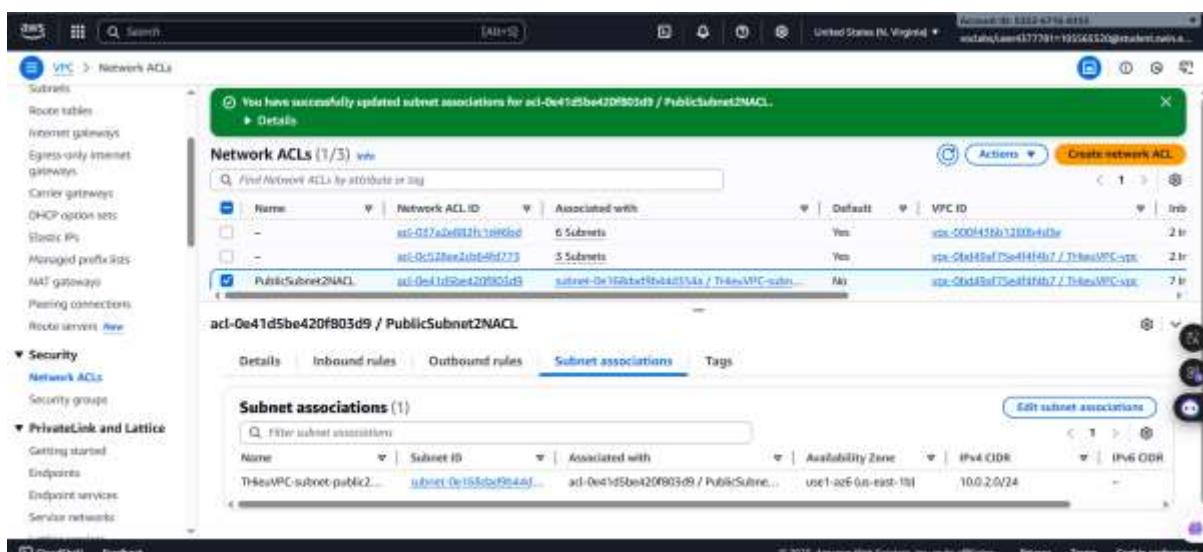


Figure 54: Subnet associations successfully

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

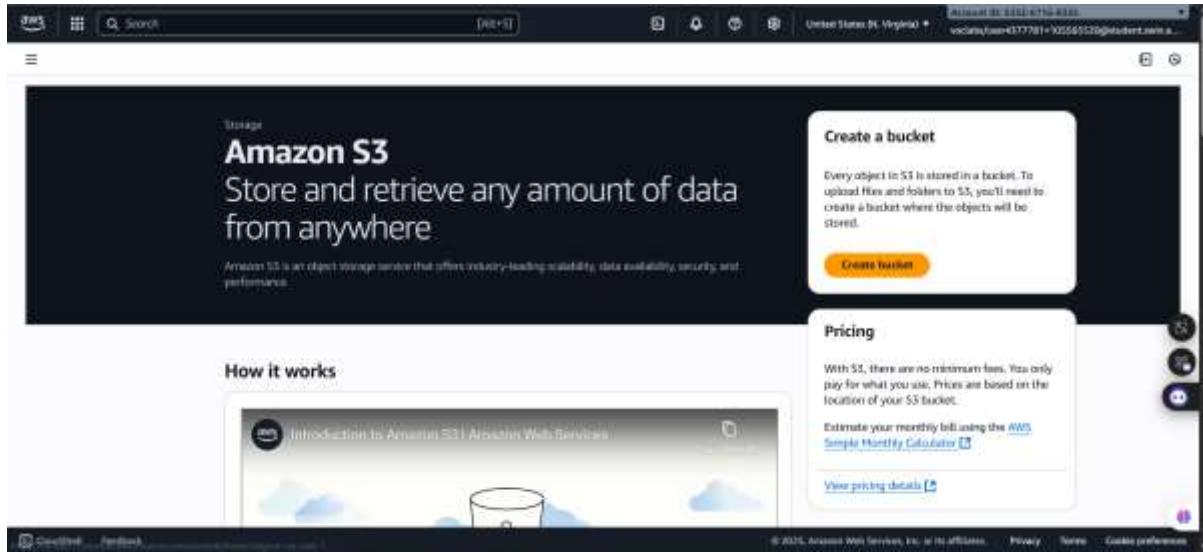


Figure 55: S3 Homepage

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*

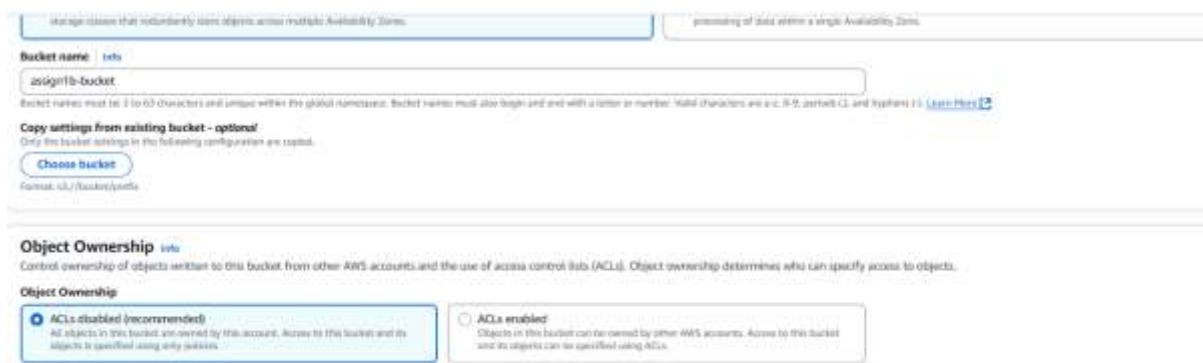


Figure 54: General Configuration

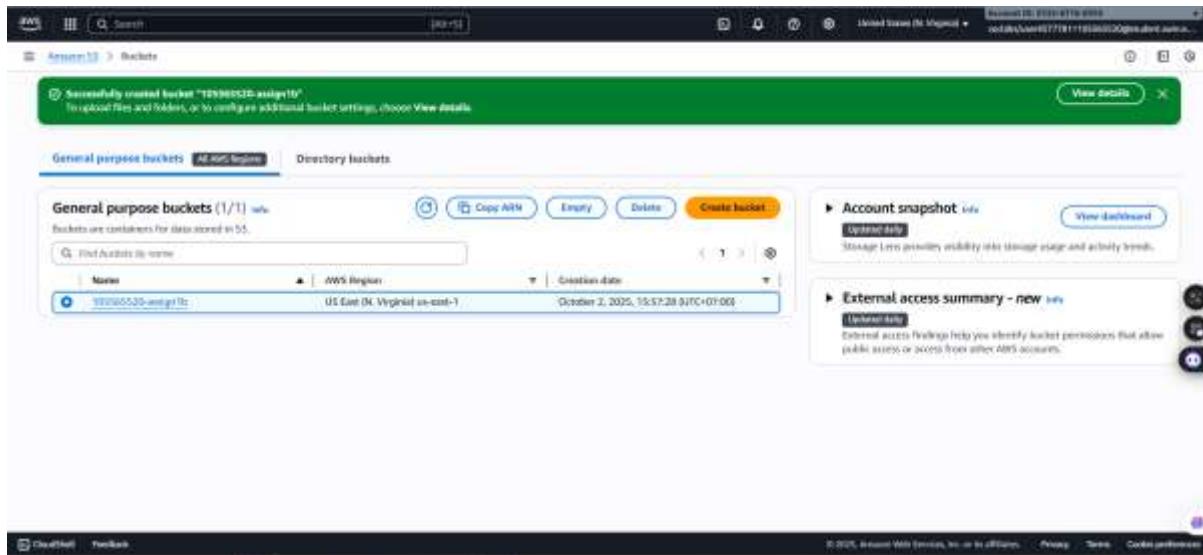


Figure 55: Created successfully

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

The screenshot shows the "Edit bucket policy" page for the "105565520-assign1b" bucket. It includes sections for "Bucket policy" (with a note about JSON syntax), "Bucket ARN" (arn:aws:s3:::gihieu-assign1b-bucket), and "Policy". The "Policy" section displays a JSON-based bucket policy:

```

1 * [
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Principal": "*",
7             "Action": "s3:GetObject",
8             "Resource": "arn:aws:s3:::gihieu-assign1b-bucket/*"
9         }
10    ]
11 ]

```

Figure 56: Bucket policy configuration

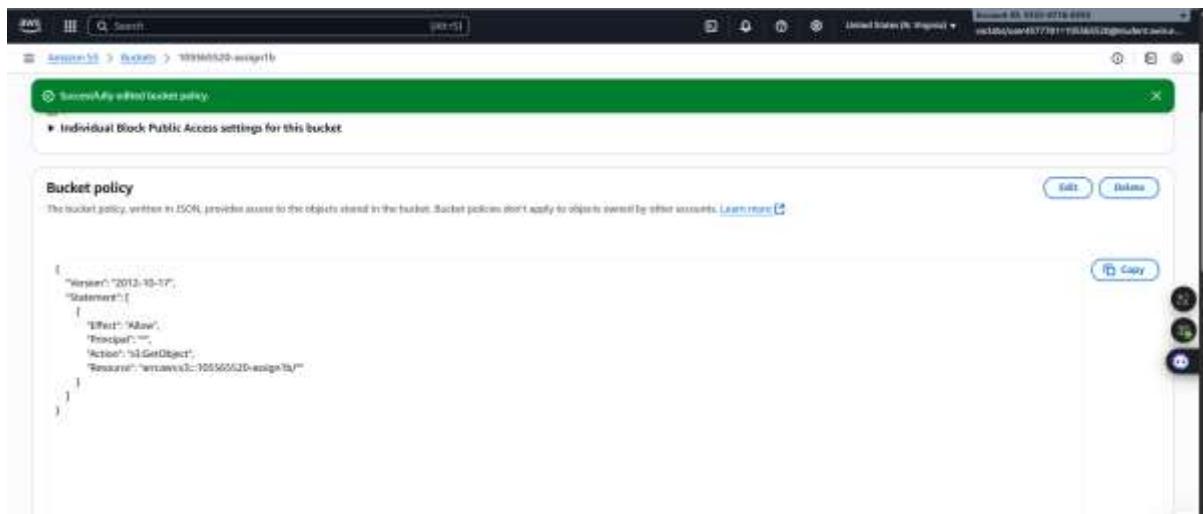


Figure 57: Saved changed successfully

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

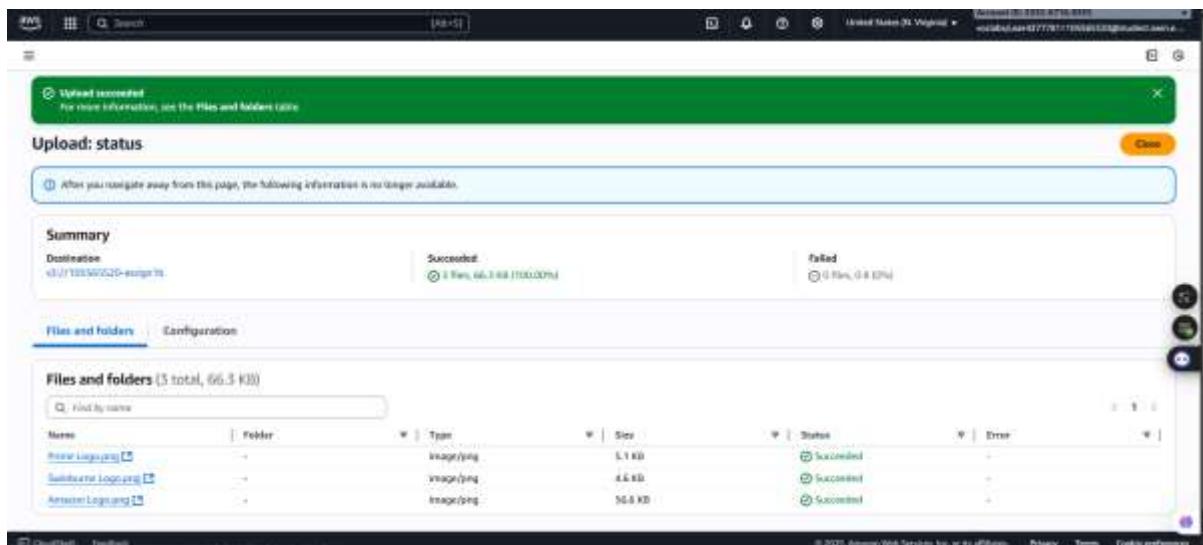


Figure 58: Uploaded successfully

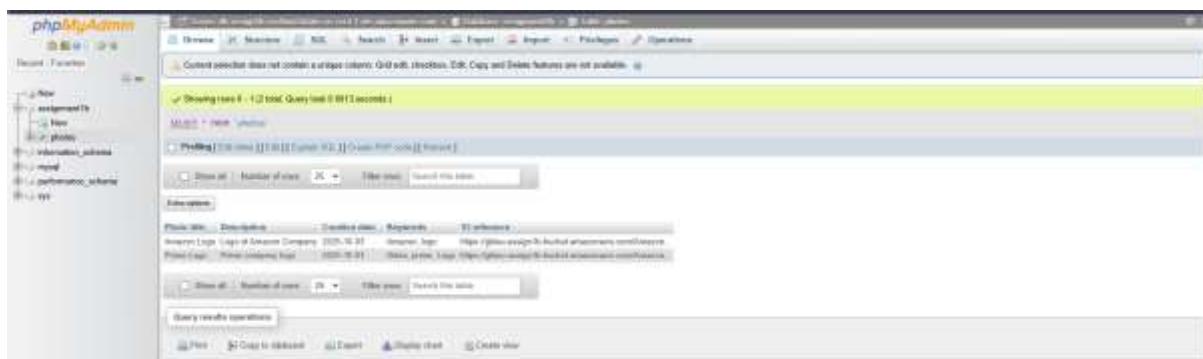
Step 2.1.5: Open the phpAdmin and add value

| Column | Type | Function | Null | Value |
|---------------|--------------|----------|------|---|
| Photo title | varchar(255) | | | Amazon Logo |
| Description | varchar(255) | | | Logo of Amazon Company |
| Creation date | date | | | 2025-10-3 |
| Keywords | varchar(255) | | | Amazon, logo |
| S3 reference | varchar(255) | | | https://ghieu-assign1b-bucket.amazonaws.com/Amazon Logo.png |

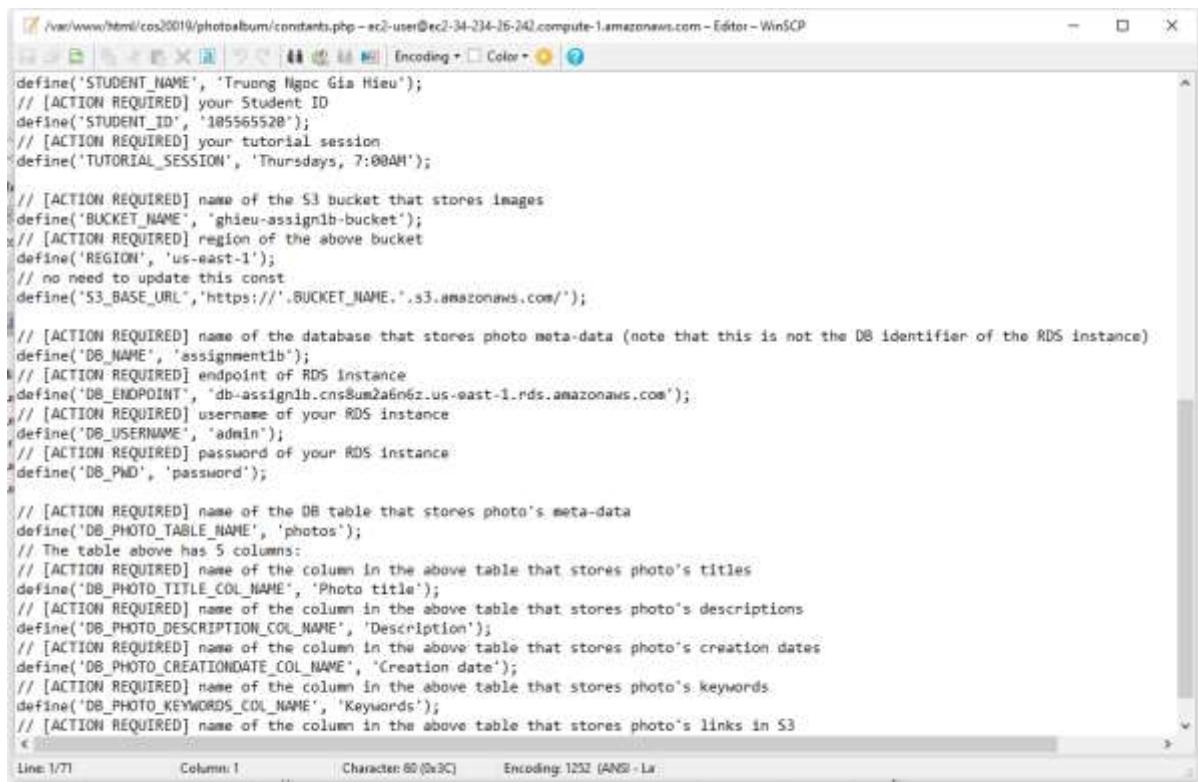
Figure 59: Amazon Logo configuration

| Column | Type | Function | Null | Value |
|---------------|--------------|----------|---|-------|
| Photo title | varchar(255) | | Prime Logo | |
| Description | varchar(255) | | Prime company logo | |
| Creation date | date | | 2025-10-3 | |
| Keywords | varchar(255) | | Video, prime, Logo | |
| S3 reference | varchar(255) | | https://ghieu-assign1b-bucket.amazonaws.com/Amazon Logo.png | |

Console

Figure 60: Prime Logo configuration*Figure 61: Configuration created successfully*

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



```

#!/var/www/html/cos20019/photoalbum/constants.php - ec2-user@ec2-34-234-26-24.compute-1.amazonaws.com - Editor - WinSCP
Encoding: Color

define('STUDENT_NAME', 'Truong Ngoc Gia Hieu');
// [ACTION REQUIRED] your Student ID
define('STUDENT_ID', '105565520');
// [ACTION REQUIRED] your tutorial session
define('TUTORIAL_SESSION', 'Thursdays, 7:00AM');

// [ACTION REQUIRED] name of the S3 bucket that stores images
define('BUCKET_NAME', 'ghieu-assign1b-bucket');
// [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 that this is not the DB identifier of the RDS instance)
define('DB_NAME', 'assignment1b');
// [ACTION REQUIRED] endpoint of RDS instance
define('DB_ENDPOINT', 'db-assign1b.cn8um26n6z.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', 'password');

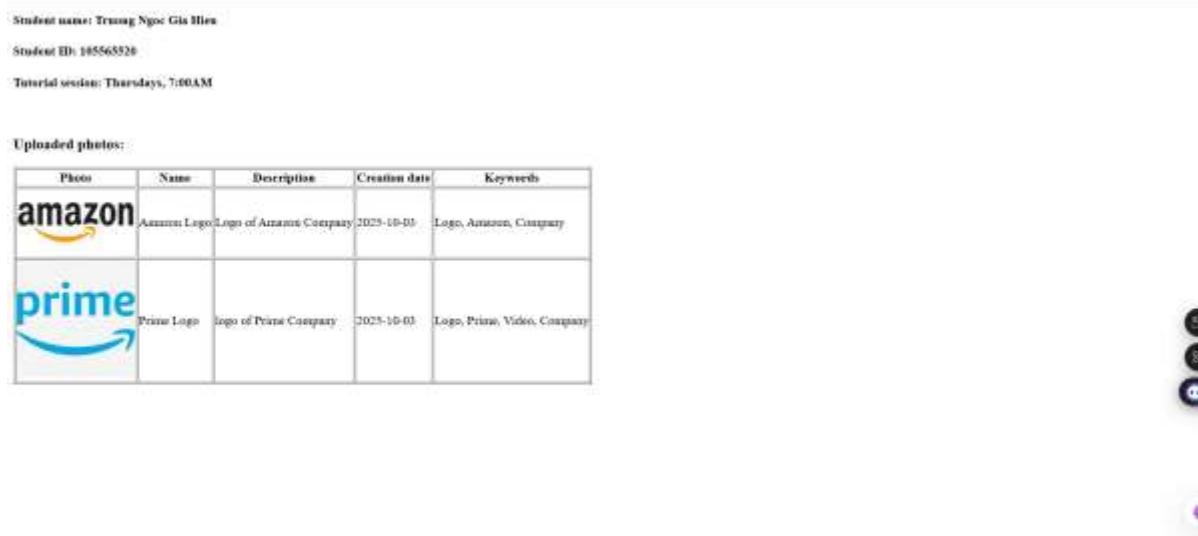
// [ACTION REQUIRED] name of the DB table that stores photo's meta-data
define('DB_PHOTO_TABLE_NAME', 'photos');
// The table above has 5 columns:
// [ACTION REQUIRED] name of the column in the above table that stores photo's titles
define('DB_PHOTO_TITLE_COL_NAME', 'Photo title');
// [ACTION REQUIRED] 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', 'Creation date');
// [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
//

Line: 1/71 Column: 1 | Character: 60 (0x3C) Encoding: 1252 (ANSI - Latin 1)

```

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>



Student name: Truong Ngoc Gia Hieu
 Student ID: 105565520
 Tutorial session: Thursdays, 7:00AM

Uploaded photos:

| Photo | Name | Description | Creation date | Keywords |
|-------|-------------|------------------------|---------------|-----------------------------|
| | Amazon Logo | Logo of Amazon Company | 2025-10-05 | Logo, Amazon, Company |
| | prime | Logo of Prime Company | 2025-10-05 | Logo, Prime, Video, Company |

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.