Title: Scalable Web Application Infrastructure on AWS

By Mohith KumarReddy

Cloud Based Solution Architect – 22SDCS04A

Section 31

Under the guidance of

Dr. S.Kavitha

Abstract

This project demonstrates the deployment of a scalable and secure 3-tier web application architecture on Amazon Web Services (AWS). The architecture follows a standard model comprising the presentation layer (web tier), application layer (logic tier), and database layer (data tier). The infrastructure is provisioned within a Virtual Private Cloud (VPC) to ensure network isolation and security.

The Application Load Balancer (ALB) distributes incoming HTTP/HTTPS traffic across multiple EC2 instances hosted in private subnets, ensuring high availability and fault tolerance. These EC2 instances host the application logic, processing requests from the frontend and interacting with the backend database. The Relational Database Service (RDS) is used to manage the data tier, providing a fully managed, scalable, and secure MySQL/PostgreSQL database solution.

The best security practices such as security groups, subnet isolation, and IAM roles are implemented. This demonstration showcases not only the deployment strategy but also emphasizes high availability, scalability, and security using native AWS services.

MODULES

. **VPC** Module (Virtual Private Cloud)**:-**

The VPC forms the foundation of the network infrastructure in AWS. In this demo, a custom VPC is created to logically isolate the environment. Inside the VPC, multiple subnets are configured—public subnets for the web-facing components (like the Load Balancer) and private subnets for backend services such as EC2 instances running the application logic and the RDS database. Route Tables, Internet Gateway, and NAT Gateway are configured to control traffic flow. The public subnets are associated with the Internet Gateway for direct access, while the private subnets use the NAT Gateway for outbound internet access without being exposed to the public.

. ALB Module (Application Load Balancer) :-

The Application Load Balancer (ALB) sits in front of the application and serves as the entry point for all incoming web traffic. It is deployed in the public subnet and automatically distributes incoming HTTP/HTTPS requests across multiple EC2 instances in different availability zones. This helps achieve high availability, scalability, and fault tolerance. The ALB also supports advanced routing capabilities, such as path-based routing, which can be helpful for microservice-based applications.

. EC2 Module (Elastic Compute Cloud) :-

This module forms the Application Layer and partially the Presentation Layer. EC2 instances are used to host both the front-end (in some setups) and the backend application logic. These instances are placed in private subnets for security reasons. The

EC2 instances receive traffic from the ALB and handle business logic, process user input, and manage communication with the database layer. Auto Scaling Groups (ASG) can be added for automatic scaling based on traffic load, and Security Groups are configured to allow traffic only from the ALB and to the database.

. RDS Module (Relational Database Service) :-

Amazon RDS is used in the Database Layer of the 3-tier architecture. It hosts a managed relational database (such as MySQL or PostgreSQL) and is deployed in the private subnets to prevent direct access from the internet. RDS offers features like automated backups, snapshots, multi-AZ deployment for high availability, and easy scaling. The application EC2 instances securely connect to RDS using internal IP addresses. Security groups and subnet groups are configured to ensure secure, efficient database access and performance.

Event-Driven Architecture Overview

In this demo, a 3-tier web application is enhanced with an event-driven architecture to improve responsiveness, decoupling, and scalability. The architecture includes three layers—Presentation (Web Tier), Application (Logic Tier), and Database (Data Tier)—all deployed within a secure VPC environment. The system is designed to respond to real-time events triggered by user actions or system processes.

. **Web** Tier (Event Ingress) :-

User requests (e.g., form submissions, button clicks) enter the system through the Application Load Balancer (ALB), which routes HTTP/HTTPS requests to appropriate EC2 instances. Each request is treated as an event that initiates a chain of actions in the backend. ALB also supports WebSocket connections for real-time event streaming if needed.

. **Application** Tier (Event Processing) :-

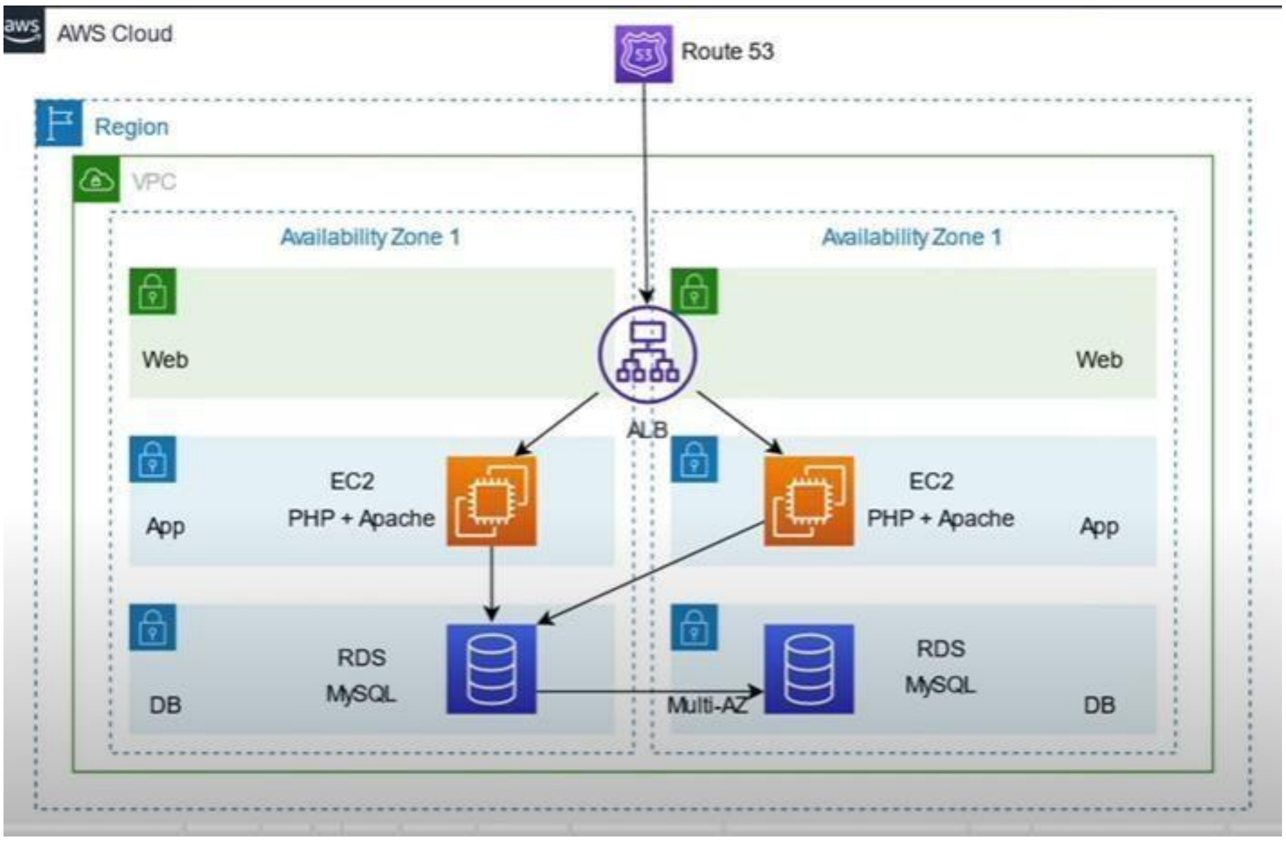
The EC2 instances in private subnets process the incoming events. These instances run the core business logic and may publish or respond to events using services like Amazon Simple Notification Service (SNS) or Amazon Simple Queue Service (SQS) to decouple tasks (e.g., sending confirmation emails, logging activities, triggering downstream data processing). This separation enables better scalability and fault tolerance.

. Database Tier (Event Persistence) :-

After processing, relevant data persisted in Amazon RDS, which stores user data, transaction logs, or application state. The application tier triggers database write or reads based on event outcomes. Event-driven logging mechanisms can also use Amazon CloudWatch Logs or even Kinesis Firehose for real-time analytics.

. Supporting Event Services (Optional) :-

Additional AWS services such as Lambda (for serverless event handling), Event Bridge (for routing events between services), or Step Functions (for workflow orchestration) can be integrated into the architecture to automate backend processes, improve observability, and reduce direct service dependencies.



Services Used in the Project

. Amazon RDS (Relational Database Service):-

Used to set up a managed relational database (MySQL/PostgreSQL) for the data layer. Deployed in private subnets for secure access by backend EC2 instances.

. Amazon S3 (Simple Storage Service) (optional but common):-

Used to store static assets like HTML, CSS, JS, images, or logs if needed.

. Amazon Cloud Watch: -

Monitors logs, metrics, and system performance across EC2, ALB, and RDS. Useful for debugging and alerting.

. Amazon IAM (Identity and Access Management):-

Manages secure access to AWS services and resources. IAM roles are assigned to EC2 instances and other services for permission control.

. NAT Gateway: -

Allows instances in private subnets to access the internet for software updates and external API calls, without exposing them to inbound traffic.

. Internet Gateway: -

Enables internet access for resources in public subnets, such as the ALB and optionally frontend EC2 instances.

. Security Groups and Network **ACLs: -**

Used to define firewall rules for inbound and outbound traffic between the tiers (Web → App → DB) and to/from the internet.

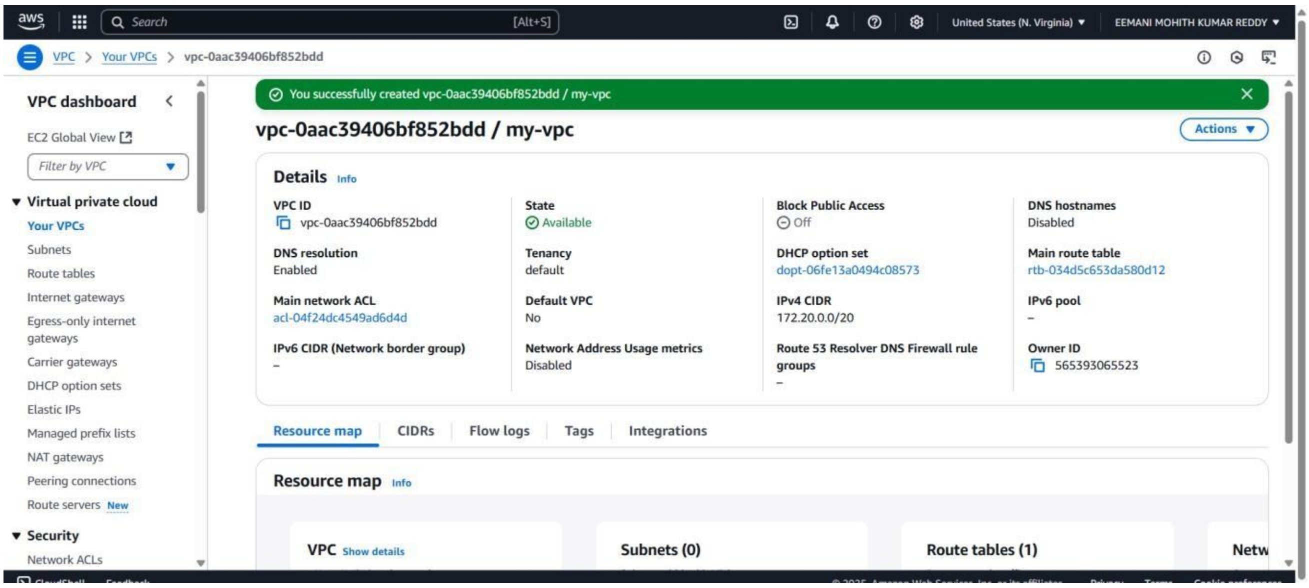
. Auto Scaling Group (optional):-

Automatically adjusts the number of EC2 instances based on traffic load to maintain performance and availability.

module implementations: -

step 1: -

- Create a Custom VPc
  - Click **"Create VPC".**
  - Choose **"VPC only"** option.
  - Provide:
    - **Name Tag:** my-vpc
    - **IPv4 CIDR block:** e.g., 172.20.0.0/20
    - Optionally enable/disable DNS resolution & hostnames
  - Click **Create VPC**



You successfully created vpc-0aac39406bf852bdd / my-vpc

vpc-0aac39406bf852bdd / my-vpc

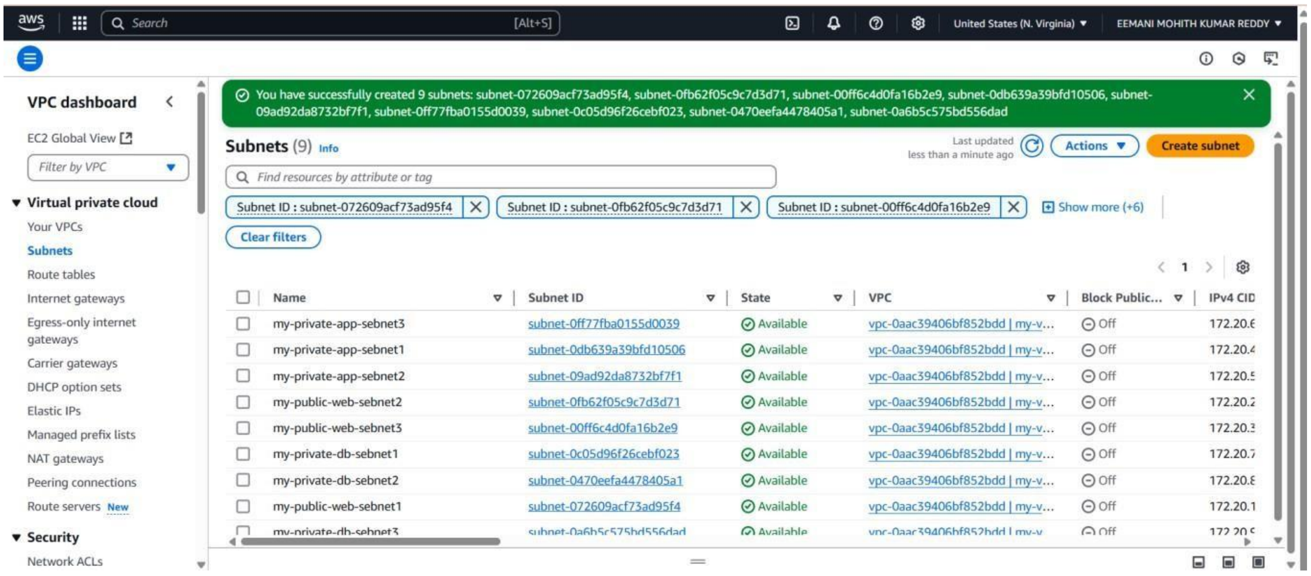
Details Info

VPC ID vpc-0aac39406bf852bdd	State Available	Block Public Access Off	DNS hostnames Disabled
DNS resolution Enabled	Tenancy default	DHCP option set dopt-06fe13a0494c08573	Main route table rtb-034d5c653da580d12
Main network ACL acl-04f24dc4549ad6d4d	Default VPC No	IPv4 CIDR 172.20.0.0/20	IPv6 pool –
IPv6 CIDR (Network border group) –	Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups –	Owner ID 565393065523

Step2: -

Subnet Name	Purpose	Type	State	IPv4 CIDR Block
my-private-app-sebnet1	App Tier	Private	Available	172.20.4.0/24 (e.g.)
my-private-app-sebnet2	App Tier	Private	Available	172.20.5.0/24 (e.g.)
my-private-app-sebnet3	App Tier	Private	Available	172.20.6.0/24 (e.g.)
my-public-web-sebnet1	Web Tier	Public	Available	172.20.0.0/24 (e.g.)
my-public-web-sebnet2	Web Tier	Public	Available	172.20.1.0/24 (e.g.)
my-public-web-sebnet3	Web Tier	Public	Available	172.20.2.0/24 (e.g.)
my-private-db-sebnet1	DB Tier	Private	Available	172.20.7.0/24 (e.g.)
my-private-db-sebnet2	DB Tier	Private	Available	172.20.8.0/24 (e.g.)
my-private-db-sebnet3	DB Tier	Private	Available	172.20.9.0/24 (e.g.)

- Clicked on **Create Subnet**.
- Select your VPC: my-vpc .
- Entered subnet names, CIDR blocks, and AZs.
- Created 9 subnets.
- Subnets are currently all set to **"Block Public Access: Off".**


You have successfully created 9 subnets: subnet-072609acf73ad95f4, subnet-0fb62f05c9c7d3d71, subnet-00ff6c4d0fa16b2e9, subnet-0db639a39bfd10506, subnet-09ad92da8732bf7f1, subnet-0ff77fba0155d0039, subnet-0c05d96f26cebf023, subnet-0470eefa4478405a1, subnet-0a6b5c575bd556dad

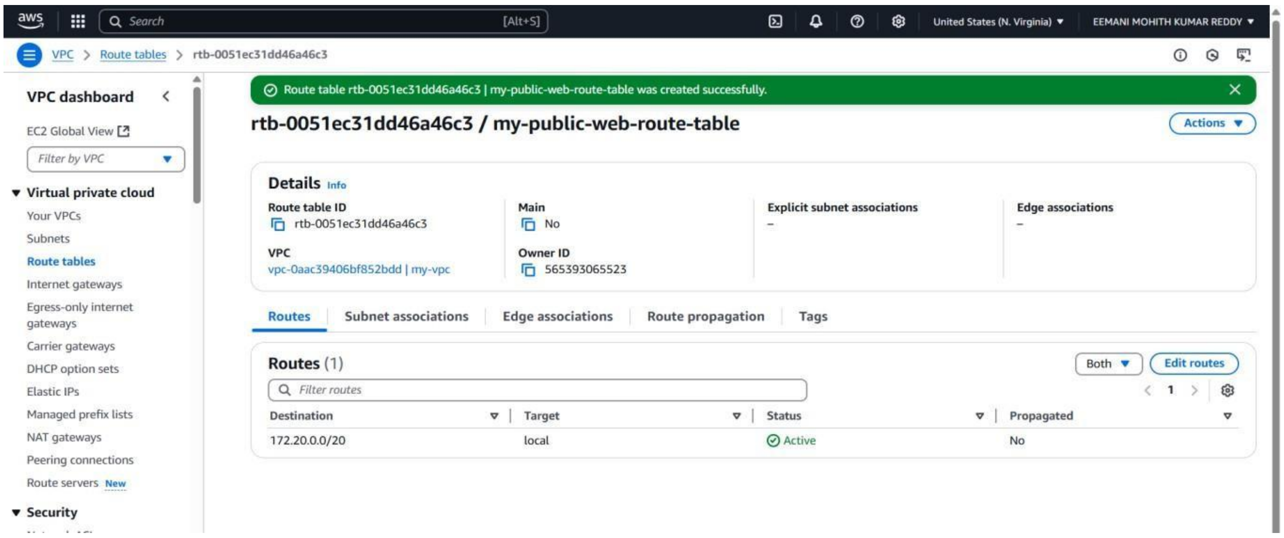
Subnets (9)

Subnet ID : subnet-072609acf73ad95f4 | Subnet ID : subnet-0fb62f05c9c7d3d71 | Subnet ID : subnet-00ff6c4d0fa16b2e9 | Show more (+6)

Name	Subnet ID	State	VPC	Block Public...	IPv4 CID
my-private-app-sebnet3	subnet-0ff77fba0155d0039	Available	vpc-0aac39406bf852bdd | my-v...	Off	172.20.6
my-private-app-sebnet1	subnet-0db639a39bfd10506	Available	vpc-0aac39406bf852bdd | my-v...	Off	172.20.4
my-private-app-sebnet2	subnet-09ad92da8732bf7f1	Available	vpc-0aac39406bf852bdd | my-v...	Off	172.20.5
my-public-web-sebnet2	subnet-0fb62f05c9c7d3d71	Available	vpc-0aac39406bf852bdd | my-v...	Off	172.20.2
my-public-web-sebnet3	subnet-00ff6c4d0fa16b2e9	Available	vpc-0aac39406bf852bdd | my-v...	Off	172.20.3
my-private-db-sebnet1	subnet-0c05d96f26cebf023	Available	vpc-0aac39406bf852bdd | my-v...	Off	172.20.7
my-private-db-sebnet2	subnet-0470eefa4478405a1	Available	vpc-0aac39406bf852bdd | my-v...	Off	172.20.8
my-public-web-sebnet1	subnet-072609acf73ad95f4	Available	vpc-0aac39406bf852bdd | my-v...	Off	172.20.1
my-private-db-sebnet3	subnet-0a6b5c575bd556dad	Available	vpc-0aac39406bf852bdd | my-v	Off	172.20.9

Step3: -

- Click **Create route table**.
- Name it is something like my-public-web-route-table.
- Choose the correct VPC: my-vpc.
- Click create.


Route table rtb-0051ec31dd46a46c3 | my-public-web-route-table was created successfully.

rtb-0051ec31dd46a46c3 / my-public-web-route-table

Details

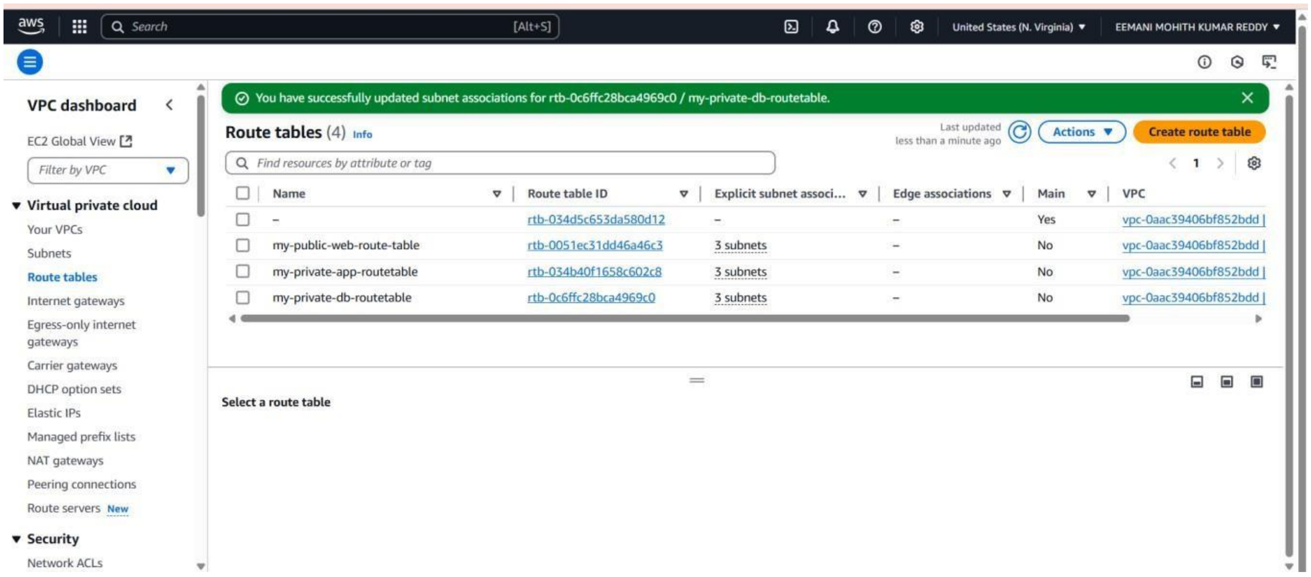
Route table ID	Main	Explicit subnet associations	Edge associations
rtb-0051ec31dd46a46c3	No	–	–
VPC	**Owner ID**		
vpc-0aac39406bf852bdd | my-vpc	565393065523		

Routes (1)

Destination	Target	Status	Propagated
172.20.0.0/20	local	Active	No

Step 4: -

- Select a route table (e.g., my-public-web-route-table).
- Click **“Subnet associations”.**
- Select the 3 respective subnets (public, private app, or private DB).
- Click **Save associations.**



You have successfully updated subnet associations for rtb-0c6ffc28bca4969c0 / my-private-db-routetable.

Route tables (4) Info

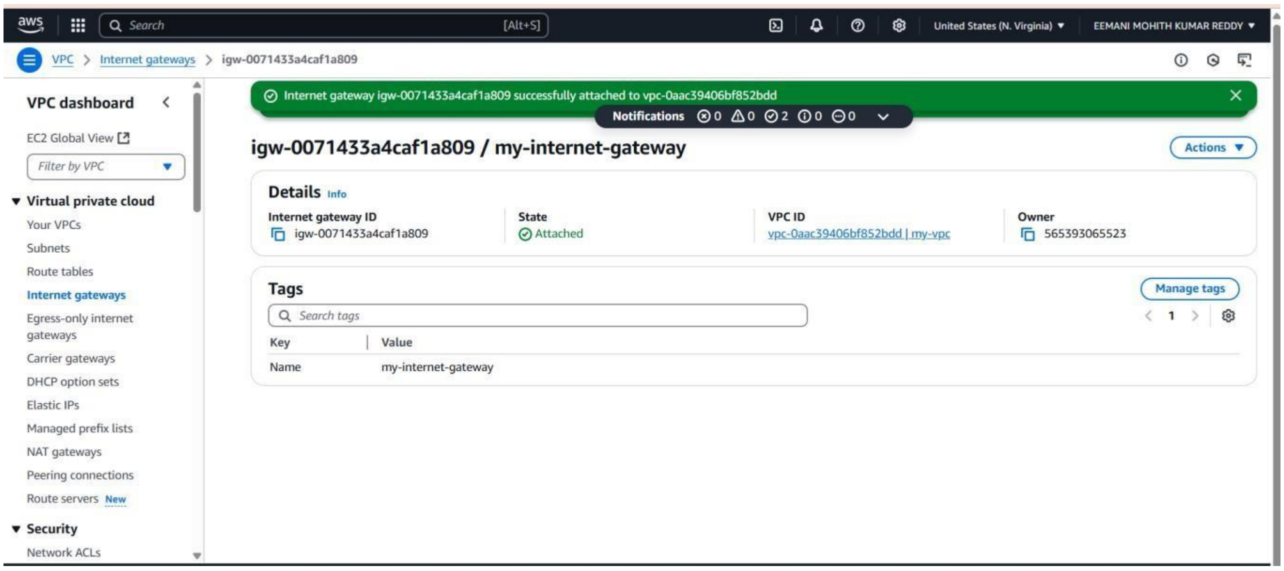
Last updated less than a minute ago · Actions ▼ · Create route table

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
–	rtb-034d5c653da580d12	–	–	Yes	vpc-0aac39406bf852bdd |
my-public-web-route-table	rtb-0051ec31dd46a46c3	3 subnets	–	No	vpc-0aac39406bf852bdd |
my-private-app-routetable	rtb-034b40f1658c602c8	3 subnets	–	No	vpc-0aac39406bf852bdd |
my-private-db-routetable	rtb-0c6ffc28bca4969c0	3 subnets	–	No	vpc-0aac39406bf852bdd |

Select a route table

Step 5: -

- Click "**Create internet gateway**".
- Enter a name, e.g. my-internet-gateway.
- Click "**Create internet gateway**".
- **Attach Internet Gateway to Your VPC.**
- Once you have been created, select the new internet gateway.
- Click "**Actions**" > "**Attach to VPC**".
- Select your VPC (e.g. my-vpc or the VPC ID you created.)
- Click "**Attach internet gateway**".



Internet gateway igw-0071433a4caf1a809 successfully attached to vpc-0aac39406bf852bdd

igw-0071433a4caf1a809 / my-internet-gateway

Details Info

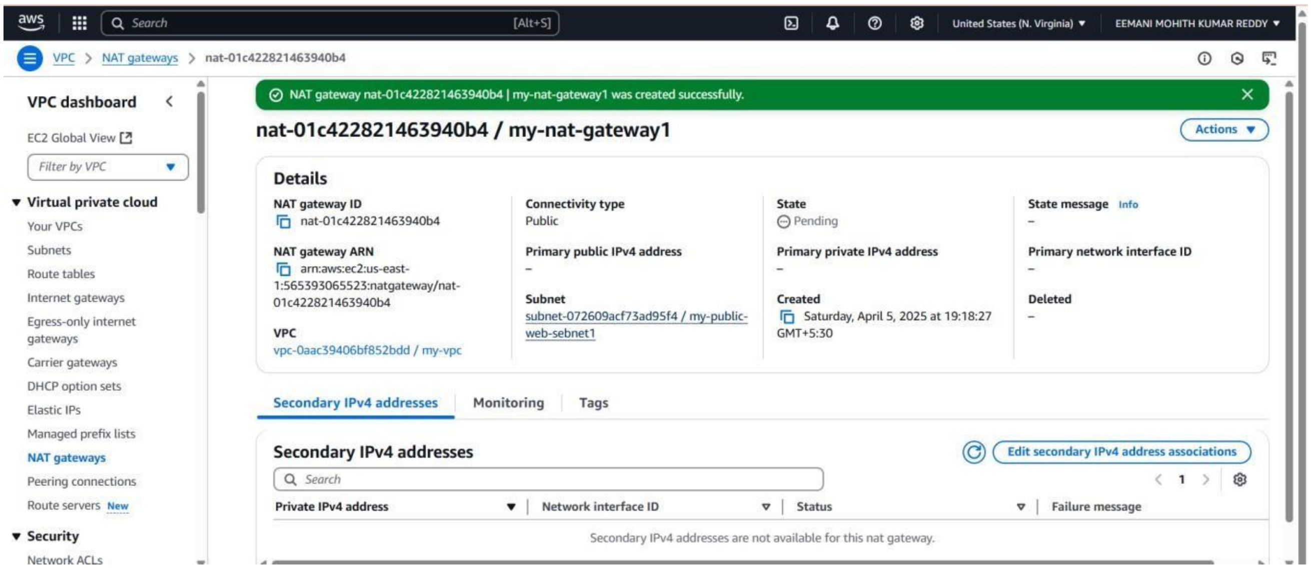
Internet gateway ID	State	VPC ID	Owner
igw-0071433a4caf1a809	Attached	vpc-0aac39406bf852bdd | my-vpc	565393065523

Tags

Key	Value
Name	my-internet-gateway

Step 6: -

-Click on **Elastic IPs** under “Network & Security”.
- Click **Allocate Elastic IP address.**
- Choose default settings and click **Allocate.**
- Copy the EIP for next step.
- Click **Create NAT Gateway.**
- Enter a name (e.g., my-nat-gateway1).
- Choose the **Public Subnet** (e.g., web-subnet1).
- Attach the **Elastic IP.**
- Click **Create NAT Gateway.**



NAT gateway nat-01c422821463940b4 | my-nat-gateway1 was created successfully.

nat-01c422821463940b4 / my-nat-gateway1

Details

NAT gateway ID nat-01c422821463940b4	Connectivity type Public	State Pending	State message Info –
NAT gateway ARN arn:aws:ec2:us-east-1:565393065523:natgateway/nat-01c422821463940b4	Primary public IPv4 address –	Primary private IPv4 address –	Primary network interface ID –
VPC vpc-0aac39406bf852bdd / my-vpc	Subnet subnet-072609acf73ad95f4 / my-public-web-sebnet1	Created Saturday, April 5, 2025 at 19:18:27 GMT+5:30	Deleted –

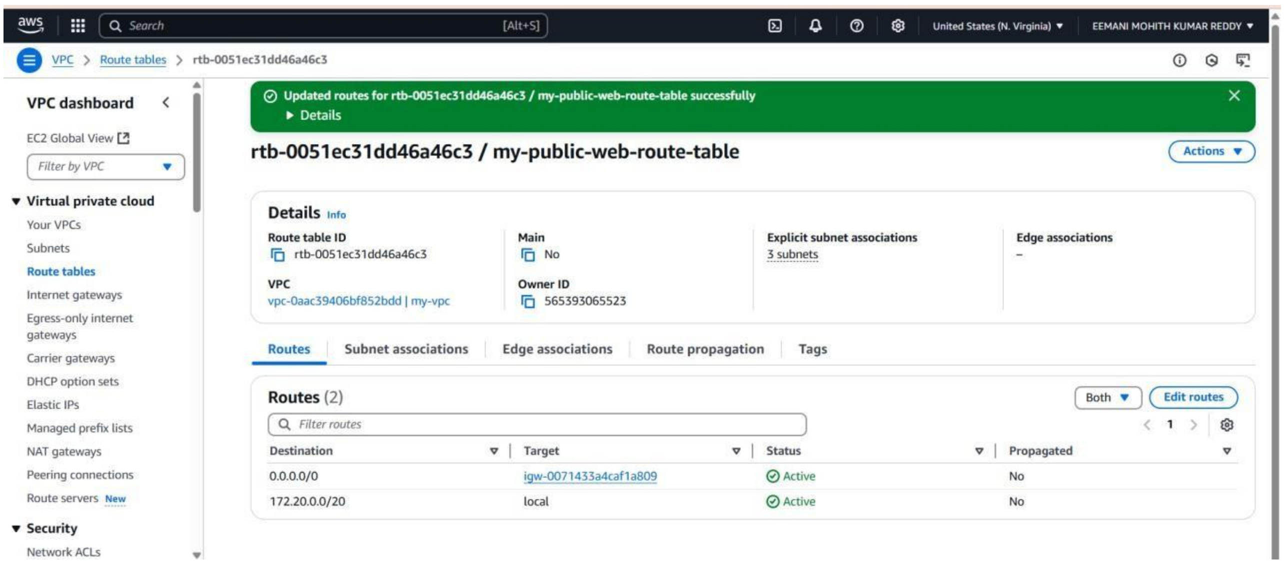
Secondary IPv4 addresses

Private IPv4 address	Network interface ID	Status	Failure message

Secondary IPv4 addresses are not available for this nat gateway.

Step 7: -

- Select the **Route Table** associated with your **Private Subnet** (not this public one),
- (It will **not** have a route to the Internet Gateway).
  . Under **"Routes"**, click **Edit routes.**
  . Click **Add route.**
- **Destination:** 0.0.0.0/0.
- **Target:** Select your **NAT Gateway** (e.g., nat-01c422821463940b4).
- Click **Save routes.**



Updated routes for rtb-0051ec31dd46a46c3 / my-public-web-route-table successfully
▶ Details

rtb-0051ec31dd46a46c3 / my-public-web-route-table

Details Info

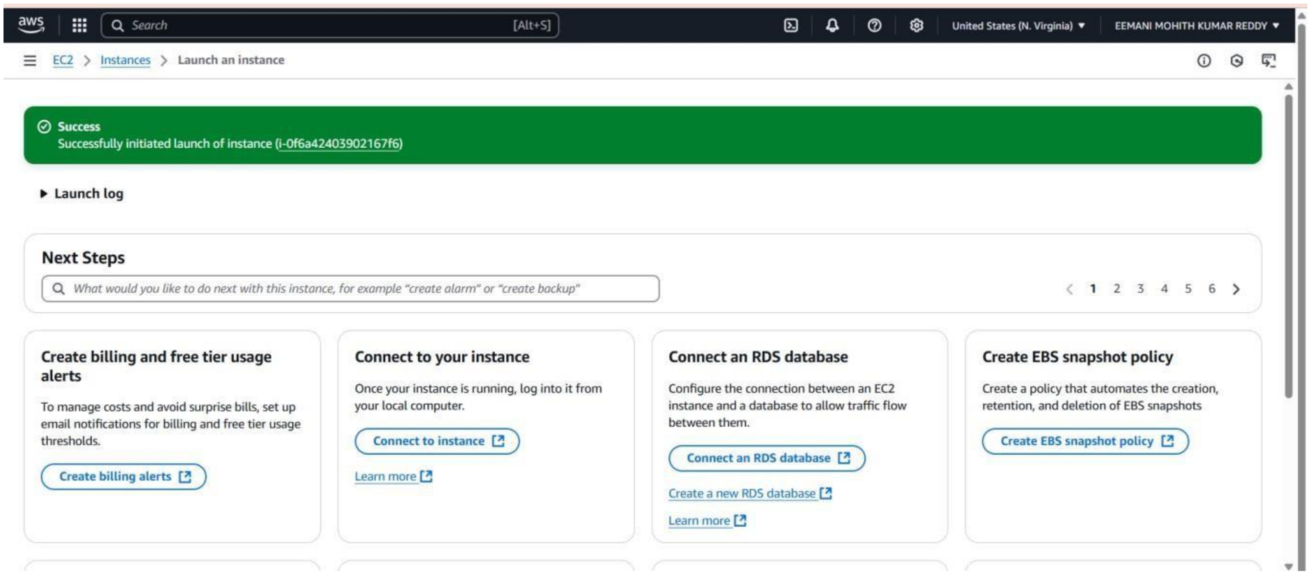
Route table ID rtb-0051ec31dd46a46c3	Main No	Explicit subnet associations 3 subnets	Edge associations –
VPC vpc-0aac39406bf852bdd | my-vpc	Owner ID 565393065523		

Routes (2)

Destination	Target	Status	Propagated
0.0.0.0/0	igw-0071433a4caf1a809	Active	No
172.20.0.0/20	local	Active	No

Step 8: -

- **Go to EC2 Dashboard.**
- **Click Launch Instance.**
- **Configure Instance Basics.**
- **Key Pair (Login).**
- **Network Settings.**
- Click Lanch Instance.



EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-0f6a42403902167f6)

Launch log

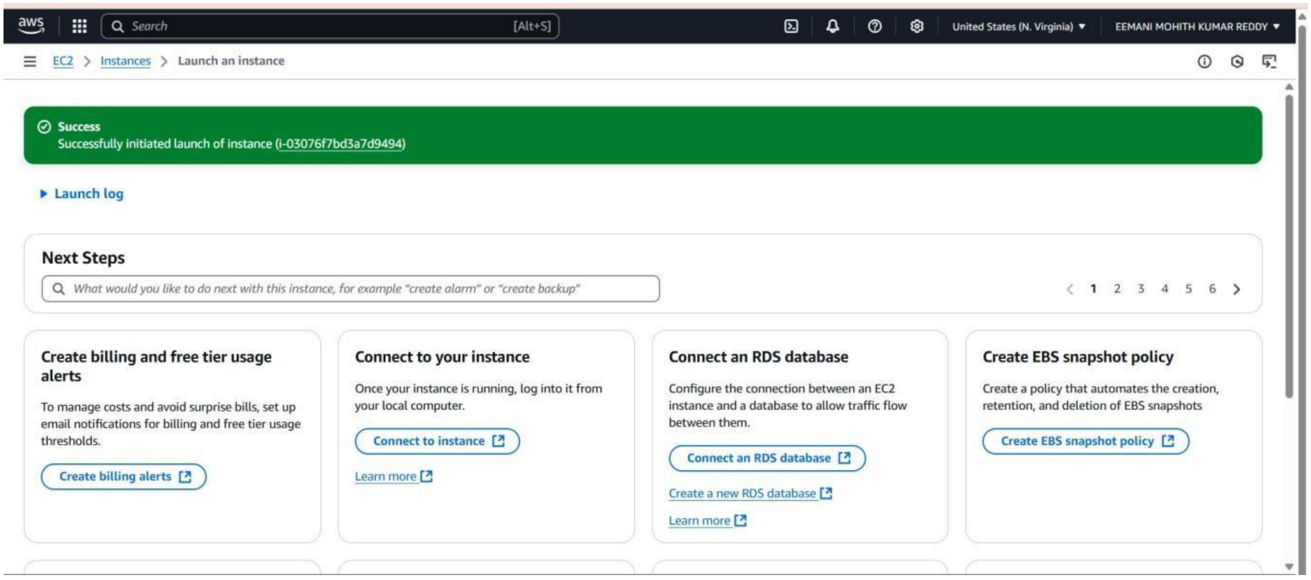
Next Steps

Create billing and free tier usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
Create billing alerts

Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more

Connect an RDS database
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more

Create EBS snapshot policy
Create a policy that automates the creation, retention, and deletion of EBS snapshots
Create EBS snapshot policy



EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-03076f7bd3a7d9494)

Launch log

Next Steps

Create billing and free tier usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
Create billing alerts

Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more

Connect an RDS database
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more

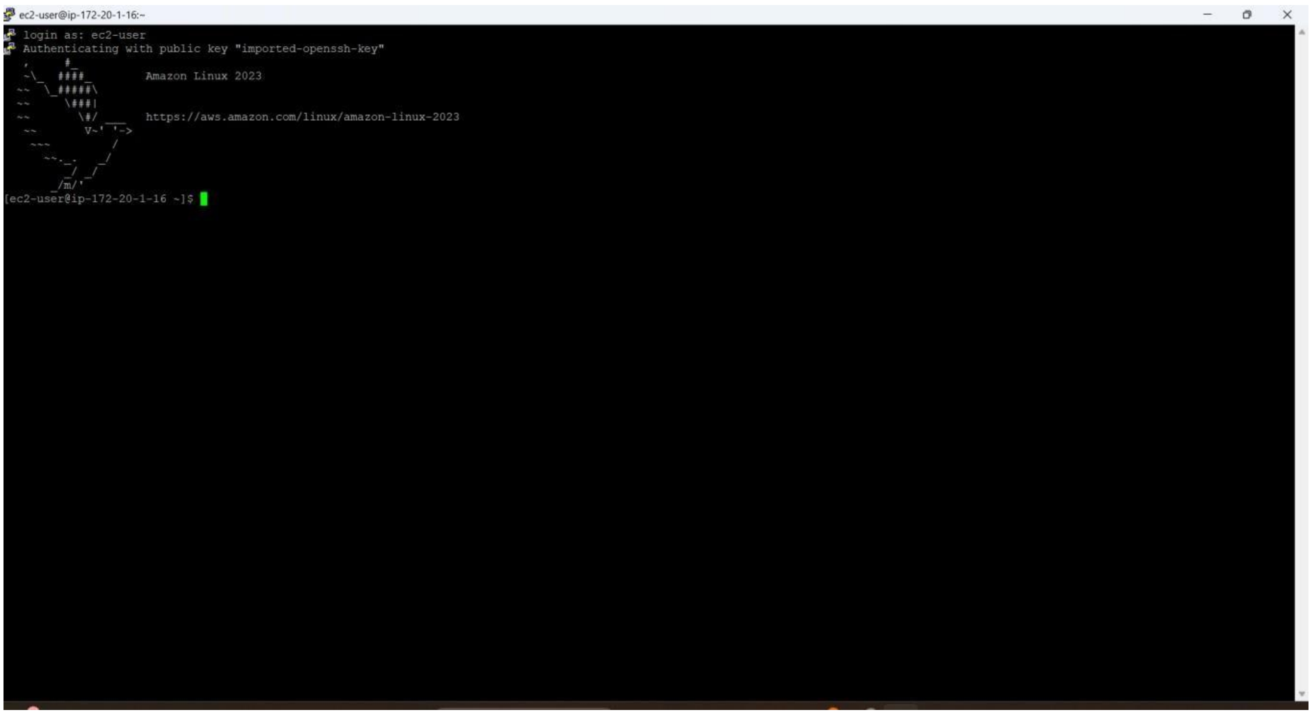
Create EBS snapshot policy
Create a policy that automates the creation, retention, and deletion of EBS snapshots
Create EBS snapshot policy

-Create a 3 EC2 Instance.
. my-jump-server.
. my-php-app-server1.
. my-php-app-server2.

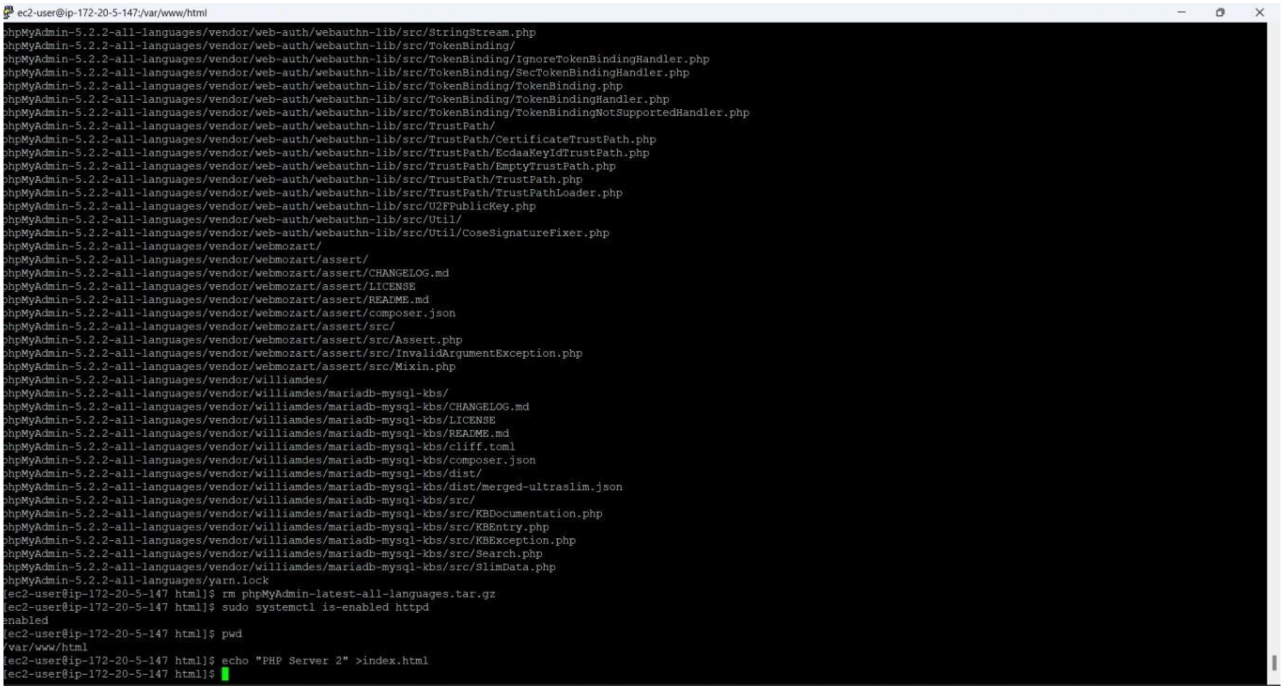
Step 9: -

-Go to SSH in that give the port number.

- Run the commands.



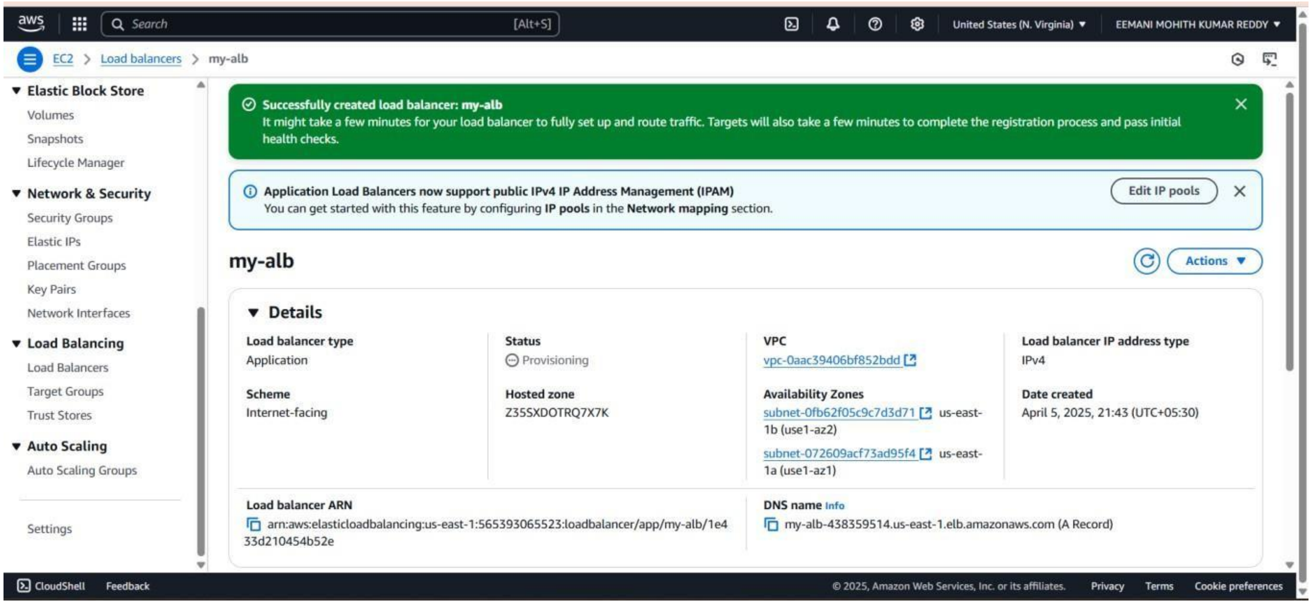
```
ec2-user@ip-172-20-1-16:~
login as: ec2-user
Authenticating with public key "imported-openssh-key"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-20-1-16 ~]$ 
```



```
ec2-user@ip-172-20-5-147:/var/www/html
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/StringStream.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TokenBinding/
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TokenBinding/IgnoreTokenBindingHandler.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TokenBinding/SecTokenBindingHandler.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TokenBinding/TokenBinding.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TokenBinding/TokenBindingHandler.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TokenBinding/TokenBindingNotSupportedHandler.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TrustPath/
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TrustPath/CertificateTrustPath.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TrustPath/EcdaaKeyIdTrustPath.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TrustPath/EmptyTrustPath.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TrustPath/TrustPath.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/TrustPath/TrustPathLoader.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/U2FPublicKey.php
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/Util/
phpMyAdmin-5.2.2-all-languages/vendor/web-auth/webauthn-lib/src/Util/CoseSignatureFixer.php
phpMyAdmin-5.2.2-all-languages/vendor/webmozart/
phpMyAdmin-5.2.2-all-languages/vendor/webmozart/assert/
phpMyAdmin-5.2.2-all-languages/vendor/webmozart/assert/CHANGELOG.md
phpMyAdmin-5.2.2-all-languages/vendor/webmozart/assert/LICENSE
phpMyAdmin-5.2.2-all-languages/vendor/webmozart/assert/README.md
phpMyAdmin-5.2.2-all-languages/vendor/webmozart/assert/composer.json
phpMyAdmin-5.2.2-all-languages/vendor/webmozart/assert/src/
phpMyAdmin-5.2.2-all-languages/vendor/webmozart/assert/src/Assert.php
phpMyAdmin-5.2.2-all-languages/vendor/webmozart/assert/src/InvalidArgumentException.php
phpMyAdmin-5.2.2-all-languages/vendor/webmozart/assert/src/Mixin.php
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/CHANGELOG.md
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/LICENSE
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/README.md
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/cliff.toml
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/composer.json
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/dist/
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/dist/merged-ultraslim.json
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/src/
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/src/KBDocumentation.php
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/src/KBEntry.php
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/src/KBException.php
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/src/Search.php
phpMyAdmin-5.2.2-all-languages/vendor/williamdes/mariadb-mysql-kbs/src/SlimData.php
phpMyAdmin-5.2.2-all-languages/yarn.lock
[ec2-user@ip-172-20-5-147 html]$ rm phpMyAdmin-latest-all-languages.tar.gz
[ec2-user@ip-172-20-5-147 html]$ sudo systemctl is-enabled httpd
enabled
[ec2-user@ip-172-20-5-147 html]$ pwd
/var/www/html
[ec2-user@ip-172-20-5-147 html]$ echo "PHP Server 2" >index.html
[ec2-user@ip-172-20-5-147 html]$ 
```

Step 10: -

-Load Balancer Name: my-alb
-Type: Application
-Scheme: Internet facing (publicly accessible)
-IP Address Type: IPv4
-VPC: vpc-0aac39406bf852bdd
-Availability Zones/Subnets:
. subnet-0fb62f05c9c7d3d71 (us-east-1b)
. subnet-072609acf73ad95f4 (us-east-1a)
-DNS
my-alb-438359514.us-east-1.elb.amazonaws.com



Successfully created load balancer: my-alb
It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

Application Load Balancers now support public IPv4 IP Address Management (IPAM)
You can get started with this feature by configuring **IP pools** in the **Network mapping** section.

my-alb

Details

Load balancer type	**Status**	**VPC**	**Load balancer IP address type**
Application	Provisioning	vpc-0aac39406bf852bdd	IPv4
Scheme	**Hosted zone**	**Availability Zones**	**Date created**
Internet-facing	Z35SXDOTRQ7X7K	subnet-0fb62f05c9c7d3d71 us-east-1b (use1-az2) subnet-072609acf73ad95f4 us-east-1a (use1-az1)	April 5, 2025, 21:43 (UTC+05:30)

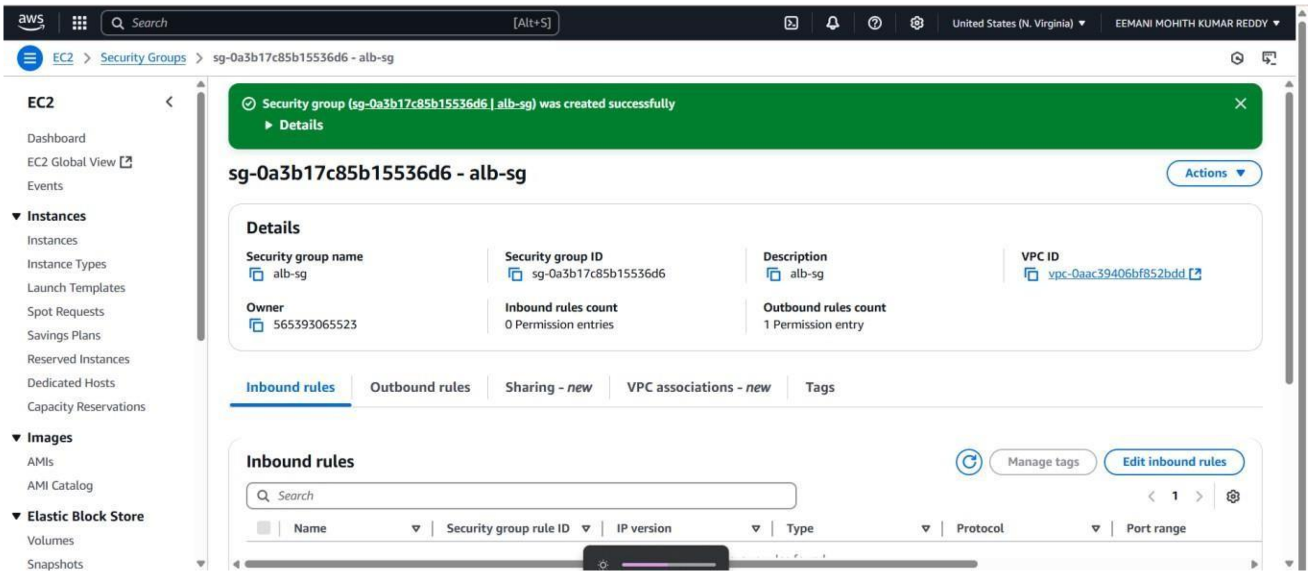
Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:565393065523:loadbalancer/app/my-alb/1e433d210454b52e

DNS name Info
my-alb-438359514.us-east-1.elb.amazonaws.com (A Record)

- click "**Security Groups**" under **Network & Security**.
- Click the "**Create security group**" button.
- Click "**Edit inbound rules**" (as shown in the screenshot).
- Click "**Add rule**".
- Configure the rule like this:

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0 (Anywhere)
HTTPS	TCP	443	0.0.0.0/0 (Anywhere)

Add both rules if you expect HTTPS later; otherwise, just start with HTTP.
-Click "**Save rules**".



Security group (sg-0a3b17c85b15536d6 | alb-sg) was created successfully

▶ Details

sg-0a3b17c85b15536d6 - alb-sg

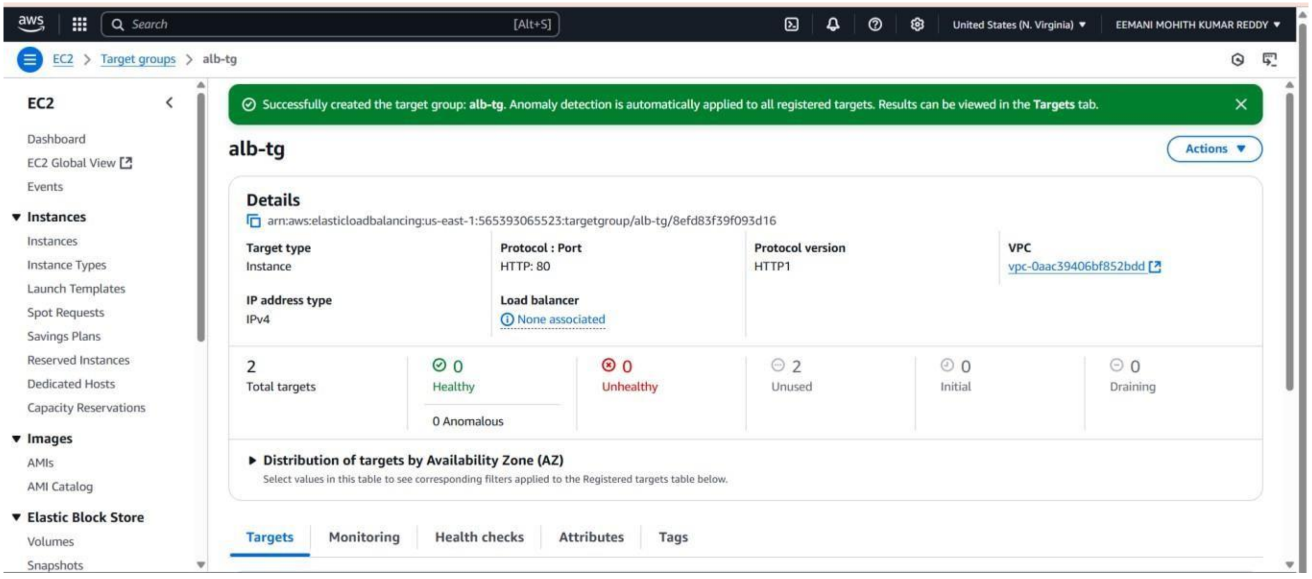
Details

Security group name alb-sg	Security group ID sg-0a3b17c85b15536d6	Description alb-sg	VPC ID vpc-0aac39406bf852bdd
Owner 565393065523	Inbound rules count 0 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules

	Name	Security group rule ID	IP version	Type	Protocol	Port range

-Click "Target Group".
-Choose "Existing target group" and select alb-tg.
-Click **Register targets** if not done already.
-Click **Create.**



Successfully created the target group: **alb-tg**. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the **Targets** tab.

alb-tg

Details

arn:aws:elasticloadbalancing:us-east-1:565393065523:targetgroup/alb-tg/8efd83f39f093d16

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-0aac39406bf852bdd
IP address type IPv4	Load balancer None associated		

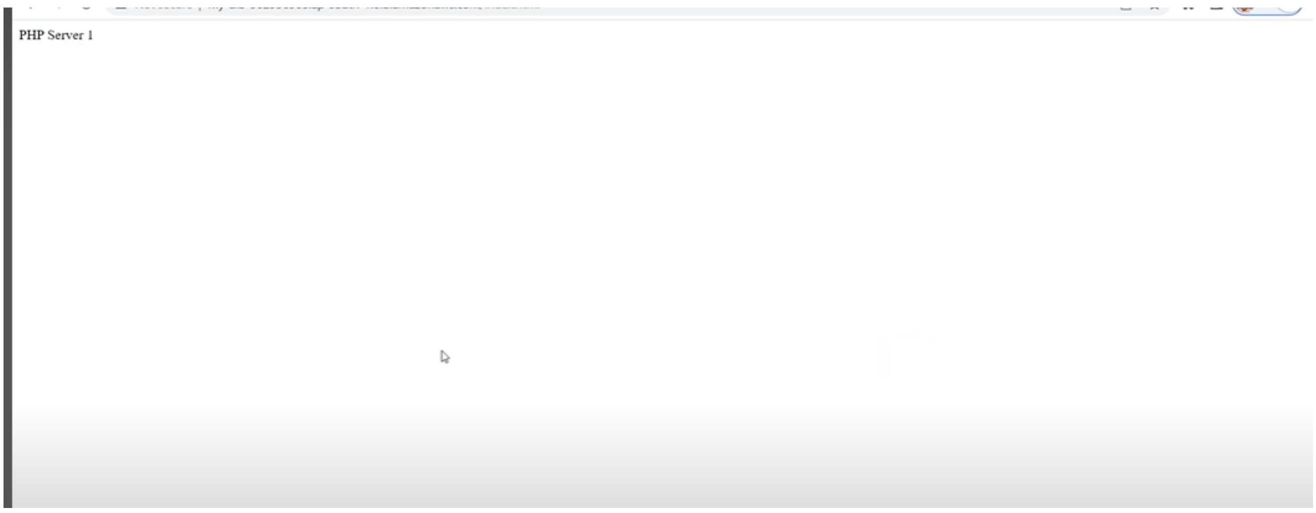
2 Total targets	0 Healthy 0 Anomalous	0 Unhealthy	2 Unused	0 Initial	0 Draining

▶ Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Step 11: -

- Go to the **DNS Name** of your ALB (you can find it under the ALB details)
- Open it in your browser.
- You should see: PHP Server 1.
- or PHP server 2.


PHP Server 1

Step 12: -

-Go to RDS → Databases → Create database

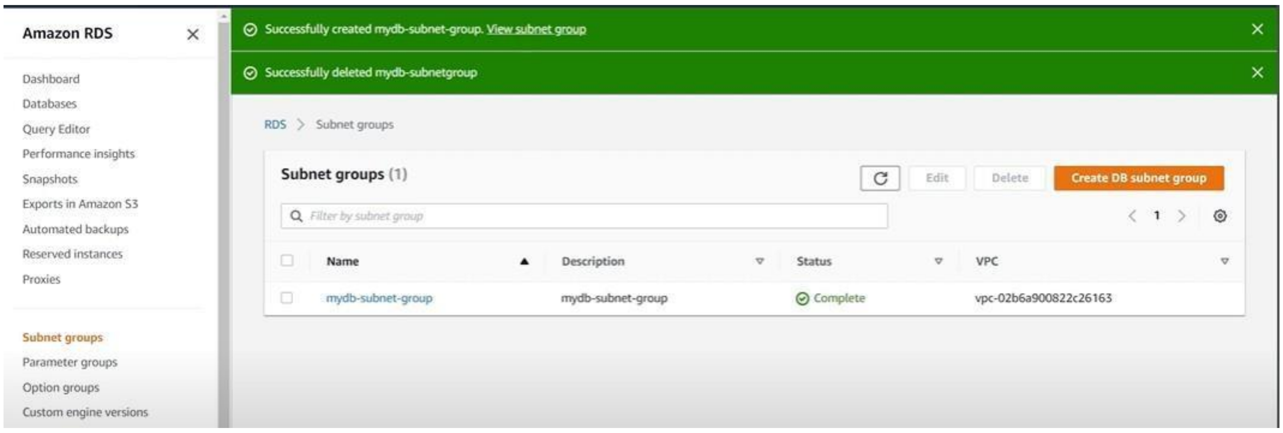
- Choose Settings

-DB Instance Settings

-Connectivity

-Database options & Storage

-Create Database



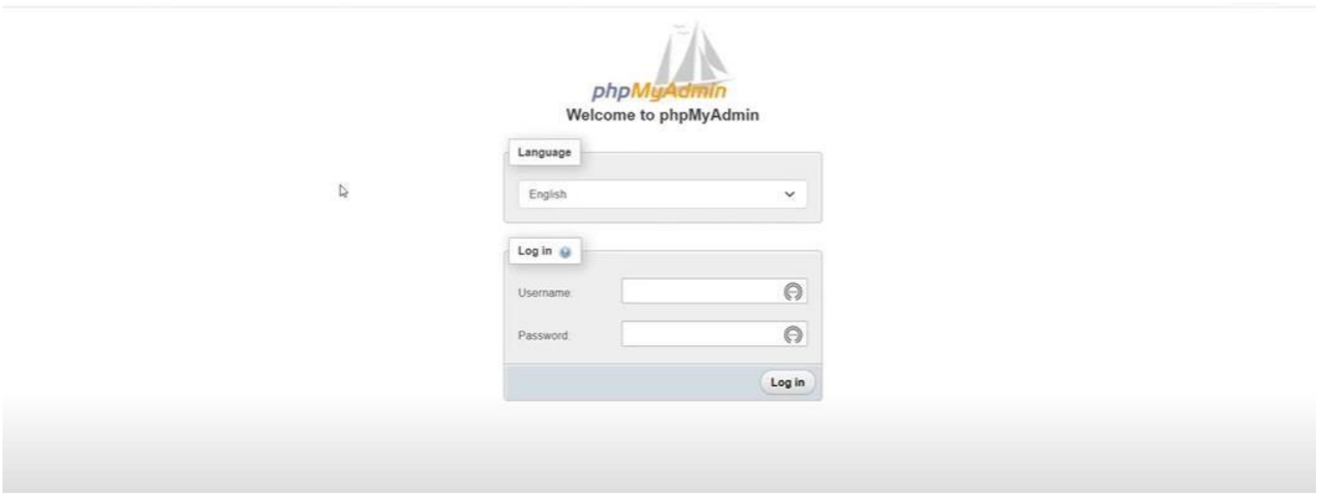
Successfully created mydb-subnet-group. View subnet group

Successfully deleted mydb-subnetgroup

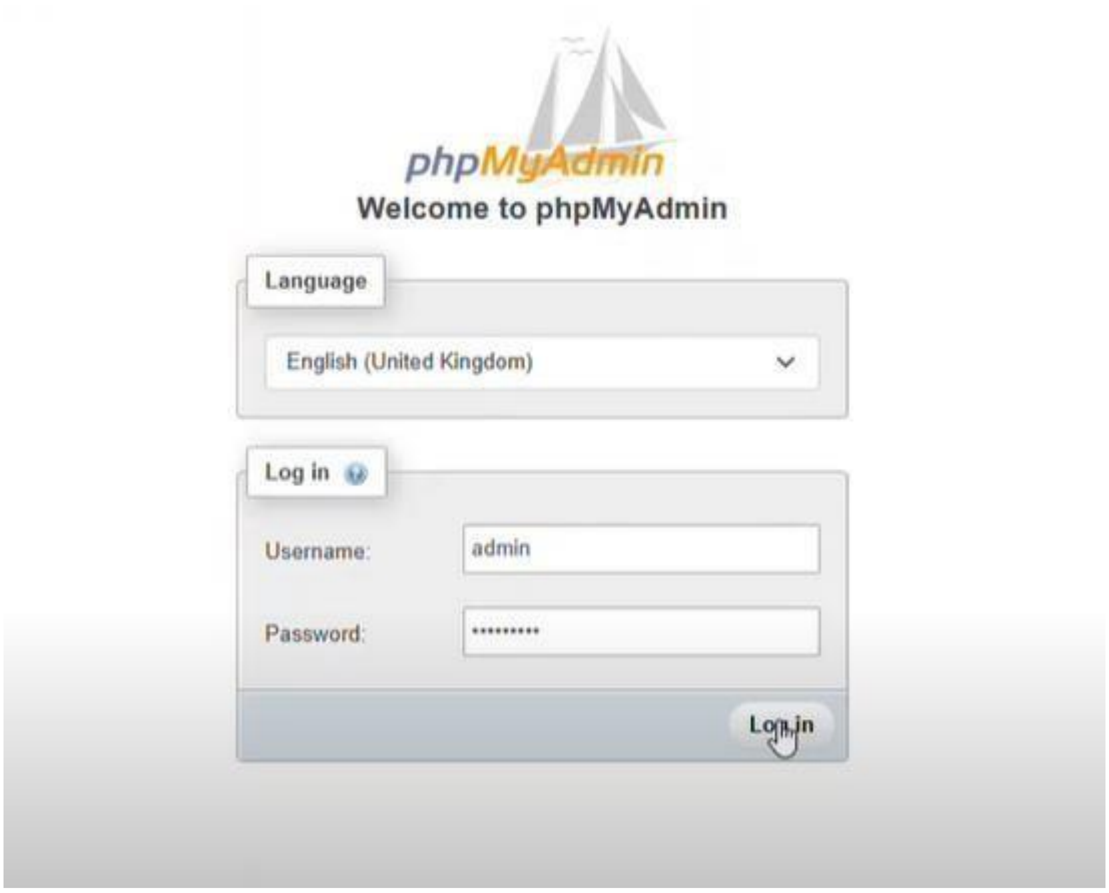
Subnet groups (1)

Name	Description	Status	VPC
mydb-subnet-group	mydb-subnet-group	Complete	vpc-02b6a900822c26163

Step 13: -


Welcome to phpMyAdmin

Output:

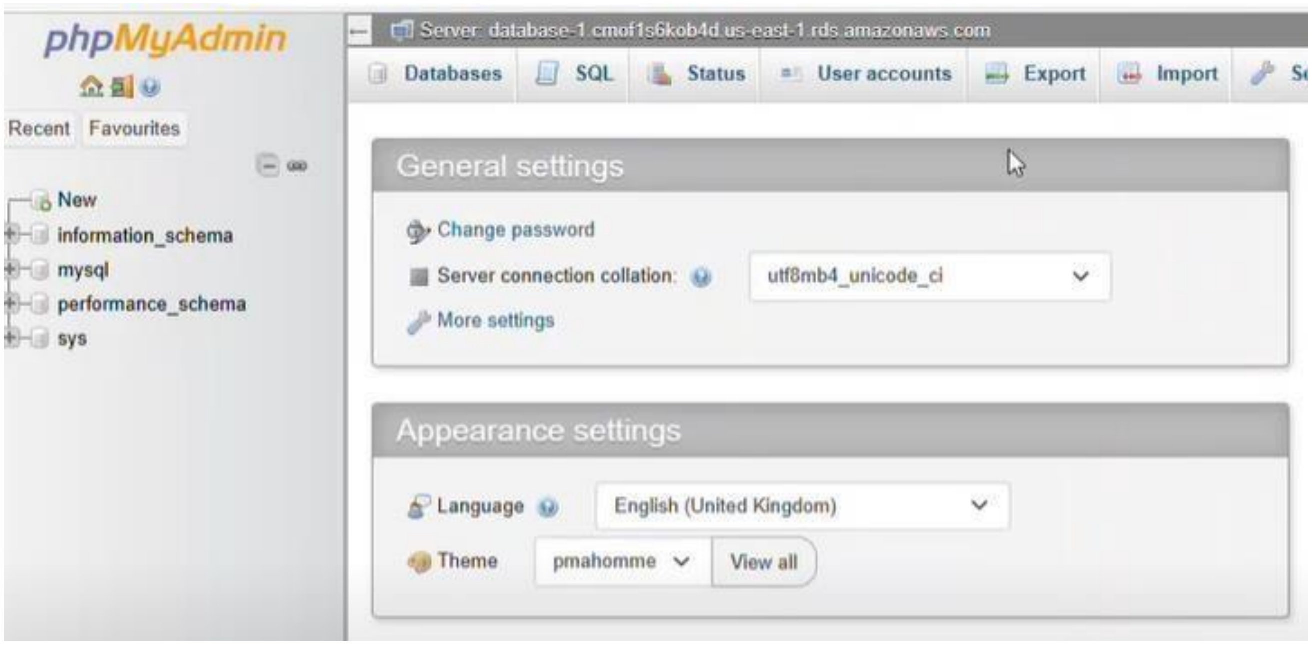

Welcome to phpMyAdmin

Language: English (United Kingdom)

Username: admin

Password: **********

Enter username and password



Server: database-1.cmof1s6kob4d.us-east-1.rds.amazonaws.com

General settings

Server connection collation: utf8mb4_unicode_ci

Appearance settings

Language: English (United Kingdom)

Theme: pmahomme

Successfully login

SSH commands

```
sudo dnf update -y
sudo dnf install -y httpd wget php-fpm php-mysqli php-json php php-
devel
sudo systemctl start httpd
sudo systemctl enable httpd
sudo systemctl is-enabled httpd
sudo usermod -a -G apache ec2-user
exit
groups
curl http://localhost
sudo chown -R ec2-user: apache /var/www
sudo chmod 2775 /var/www && find /var/www -type d -exec sudo
chmod 2775 {} \;
find /var/www -type f -exec sudo chmod 0664 {} \;
sudo dnf install php-mbstring php-xml -y
sudo systemctl restart httpd
sudo systemctl restart php-fpm

cd /var/www/html
wget https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-
all-languages.tar.gz
mkdir phpMyAdmin && tar -xvzf phpMyAdmin-latest-all-
languages.tar.gz -C phpMyAdmin --strip-components 1
rm phpMyAdmin-latest-all-languages.tar.gz
sudo systemctl is-enabled httpd

echo "PHP Server 1" >index.html do for php server1
echo "PHP Server 2" >index.html do for php server2
sudo systemctl is-enabled httpd

my-alb-902956905.ap-south-1.elb.amazonaws.com


after cd phpMyAdmin
mv config.sample.inc.php  config.inc.php
vi config.inc.php
```

replace host with ALB DNS Name Entry
user data for ec2 instances

```
#!/bin/bash
yum update -y
yum update httpd -y
systemctl restart httpd
```

Application Server user data

```
#!/bin/bash
yum update -y

wget http://repos.fedorapeople.org/repos/dchen/apache-
maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-
maven.repo
sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
yum install -y apache-maven
yum install java-1.8.0-openjdk-devel.x86_64 -y
```

db-server

```
#!/bin/bash
yum update -y
yum install mysql
```

```
ssh-add -K dev-account.pem

ssh -A ec2-user@<dns-name>

ssh ec2-user@<ip-address-application-server>
```

Login to webserver OR jump server

from there login to php-app-server
install

```
sudo yum update -y
sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
sudo yum install -y httpd24 php72 mysql57-server php72-mysqlnd

sudo yum install -y httpd
sudo systemctl start httpd
sudo systemctl enable httpd
sudo systemctl is-enabled httpd
curl http://localhost

sudo usermod -a -G apache ec2-user

sudo chown -R ec2-user: apache /var/www

sudo chmod 2775 /var/www && find /var/www -type d -exec sudo
chmod 2775 {} \;

find /var/www -type f -exec sudo chmod 0664 {} \;

sudo yum install php-mbstring php-xml -y
sudo systemctl restart httpd
sudo systemctl restart php-fpm
cd /var/www/html

echo "PHP Server 1" >index.html do for php server1
echo "PHP Server 2" >index.html do for php server2
```

Conclusion

The implementation of a 3-tier web application on AWS using VPC, ALB, EC2, and RDS demonstrates a highly scalable, secure, and modular cloud architecture. By logically separating the application into three layers—presentation, application, and data, the system ensures better manageability, security, and performance. The Virtual Private Cloud (VPC) provides a secure networking environment, with public and private subnets designed for controlled access. The Application Load Balancer (ALB) efficiently distributes incoming traffic across multiple EC2 instances, ensuring high availability and fault tolerance. The EC2 instances in the application layer handle core logic and business operations, while the Amazon RDS service offers a fully managed and scalable relational database solution for reliable data storage.

This architecture follows AWS best practices and lays a strong foundation for deploying production-ready applications with improved reliability, fault isolation, and the ability to scale on demand. Overall, the 3-tier design not only simplifies development and maintenance but also provides flexibility for future enhancements and integrations with other AWS services.