



## AWS DEVSECOPS-139

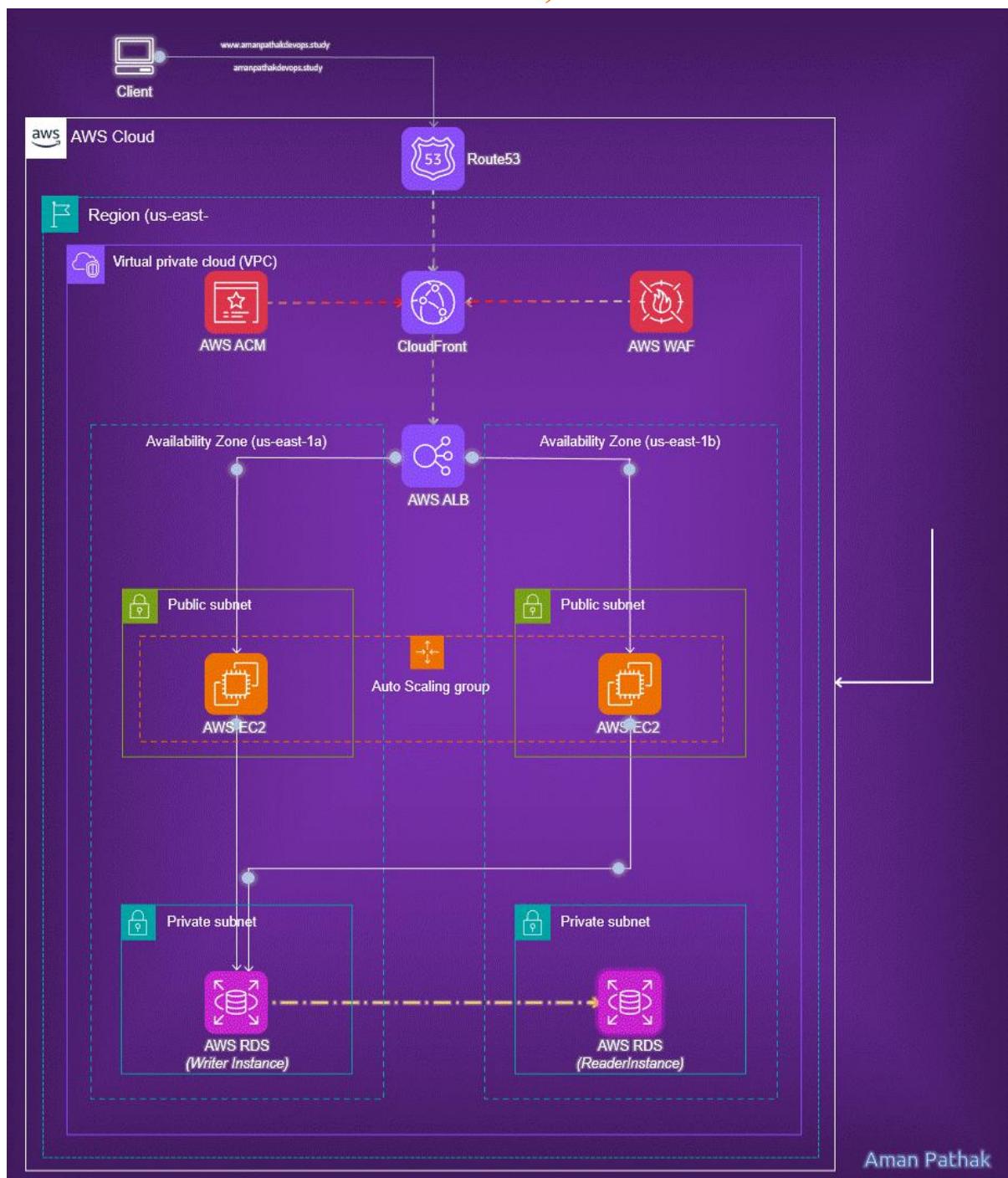
### PROJECT- 02

**AWS Secure, Scalable, High Available Web Hosting  
Architecture Using EC2, Route53, WAF, ACM,  
CloudFront, & RDS**

**BANDI ANIL KUMAR**

**Mail Id:** bandianilkumar49@gmail.com

# AWS Secure, Scalable, High Available Web Hosting Architecture Using EC2, Route53, WAF, ACM, CloudFront, & RDS



## Overview

Designed and implemented a secure, scalable, and highly available cloud infrastructure for a web application using AWS services. The architecture supports global content delivery, automated scaling, and fault-tolerant database operations across multiple Availability Zones.

It utilized Amazon EC2 instances behind an Elastic Load Balancer with Auto Scaling to handle dynamic traffic loads. DNS routing was managed through Amazon Route 53, ensuring reliable global access. Security was enforced using Security Groups, and AWS WAF to protect against common web threats. The architecture also incorporated monitoring and logging via Amazon CloudWatch, enabling proactive performance management. This project demonstrates hands on expertise in building scalable, secure, and production-ready web hosting environments using core AWS services.

## 🌟 Key Components

- **Networking:** Configured a custom VPC with public and private subnets across two Availability Zones for high availability and isolation.
- **Security & Access:** Integrated AWS WAF for threat mitigation, ACM for SSL/TLS encryption, and IAM roles for secure service access.
- **Traffic Management:** Deployed Route 53 for DNS routing and CloudFront for global content delivery with low latency.
- **Compute Layer:** Provisioned EC2 instances within an Auto Scaling Group behind an Application Load Balancer (ALB) to ensure dynamic scaling and load distribution.
- **Database Layer:** Implemented Amazon RDS with Multi-AZ deployment, featuring a writer instance and a read replica for performance and redundancy.

## 🌟 Key Benefits

- ✓ **Scalability:** Auto Scaling and CloudFront handle traffic spikes.
- ✓ **High Availability:** Multi-AZ deployment ensures uptime.
- ✓ **Security:** WAF, ACM, private subnets, and IAM roles protect resources.
- ✓ **Performance:** Load balancing and read replicas optimize response times.

## Outcomes

- Achieved seamless scalability and high fault tolerance across all layers.
- Ensured secure access and data protection through layered security controls.
- Optimized performance and cost-efficiency using auto scaling and read replicas.

## Services:

AWS VPC • EC2 • RDS • ALB • CloudFront • Route 53 • WAF • ACM • Auto Scaling • IAM  
• Multi-AZ Deployment • Infrastructure Design • Cloud Security

### ◆ Client Access & DNS

- **Route 53** handles domain name resolution and directs traffic to the application.
- **CloudFront** serves content globally with low latency.
- **AWS Certificate Manager (ACM)** provides SSL/TLS certificates for secure HTTPS access.
- **AWS WAF** protects against common web exploits and attacks.

### ◆ Networking & Infrastructure

- A **Virtual Private Cloud (VPC)** spans two Availability Zones for redundancy.
- **Public subnets** host EC2 instances behind an **Application Load Balancer (ALB)**.
- **Private subnets** contain RDS instances for secure database access.
- **NAT Gateway** allows outbound internet access from private subnets.

### ◆ Compute & Scaling

- **EC2 instances** run the application code.
- An **Auto Scaling Group** dynamically adjusts EC2 capacity based on demand.
- **ALB** distributes incoming traffic across healthy EC2 instances.

### ◆ Database Layer

- **Amazon RDS** is deployed in Multi-AZ mode:
  - **Writer instance** handles write operations.
  - **Reader instance** improves read performance and availability.

# Architecture Overview

This setup includes:

- **Global Access & DNS:** Route 53
- **Security & Delivery:** AWS Certificate Manager (ACM), CloudFront, AWS WAF
- **Core Infrastructure:** VPC with public/private subnets across two Availability Zones
- **Compute Layer:** EC2 instances in an Auto Scaling Group behind an Application Load Balancer (ALB)
- **Database Layer:** RDS with writer/reader instances for high availability and read scalability

## Steps followed to design the Architecture:

1. Set Up Networking (VPC & Subnets)
2. Configure Security
3. Provision EC2 Instances
4. Set Up Application Load Balancer (ALB)
5. Enable Auto Scaling
6. Add AWS WAF & CloudFront
7. Configure Route 53
8. Deploy RDS Instances

### 1. Set Up Networking (VPC & Subnets)

- Create a VPC in us-east-1 region (**project-route53**)

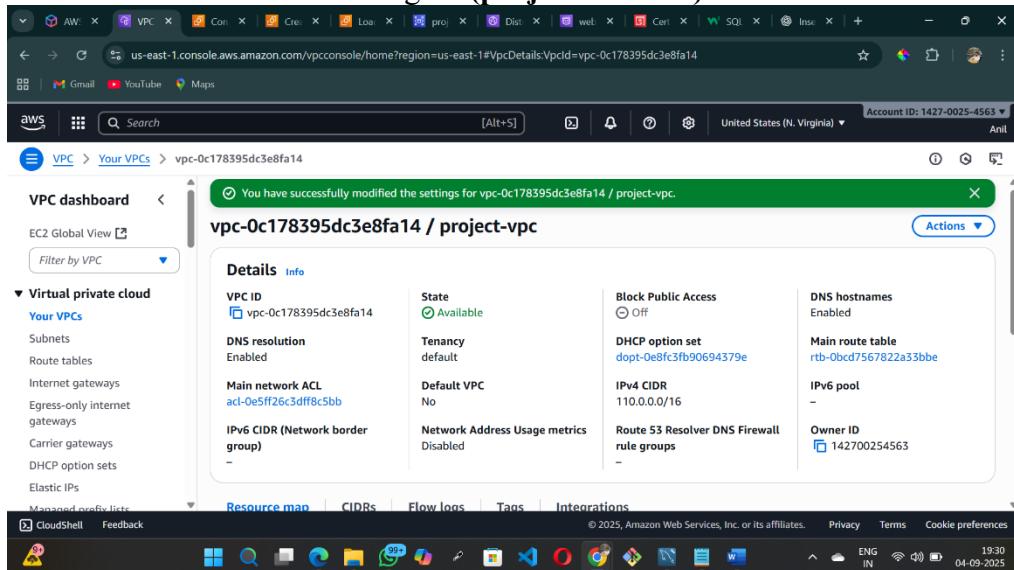
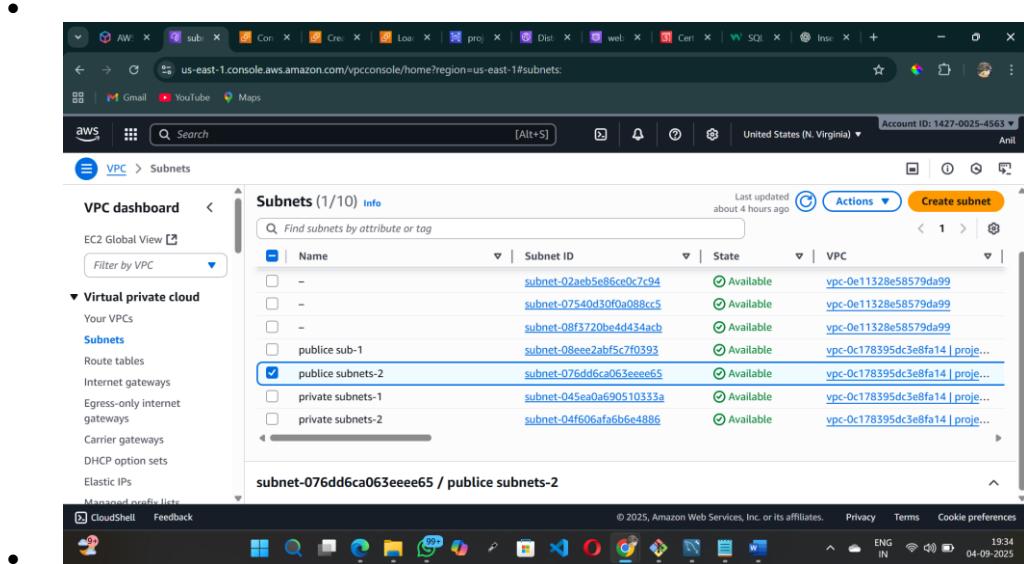


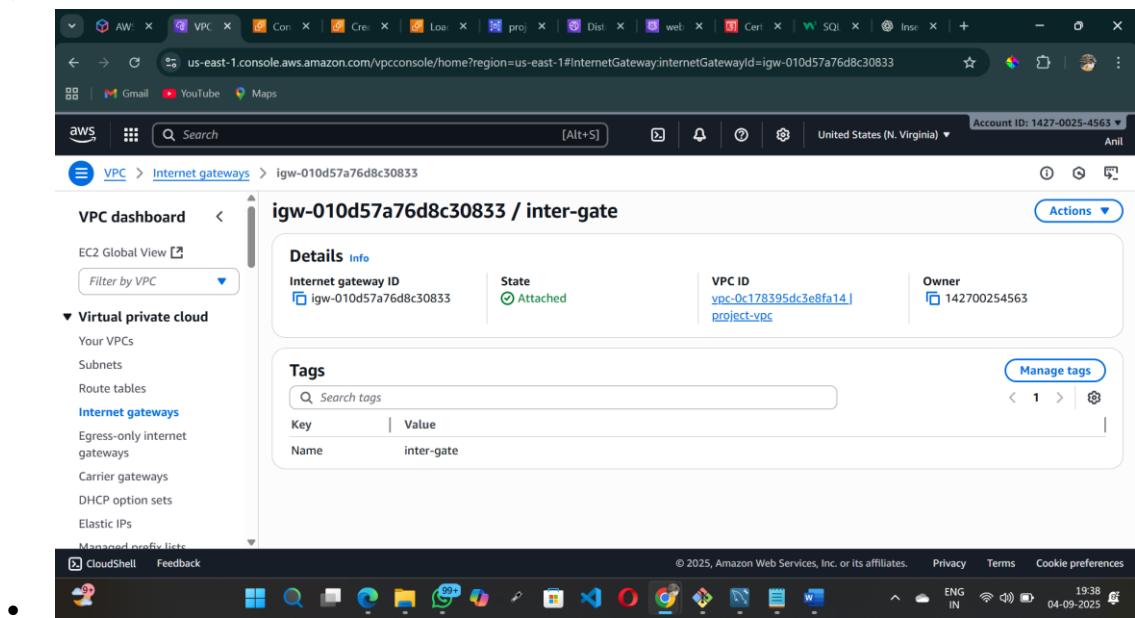
Fig 1.1: VPC

- A VPC spans two Availability Zones for redundancy (e.g., us-east-1a, us-east-1)



**Fig 1.2: Availability Zones & Subnets**

- Add 2 public subnets & 2 private subnets in the same AZs (e.g., us-east-1a, us-east-1b)
- Set up INTERNET Gateway in public subnet for private subnet internet access

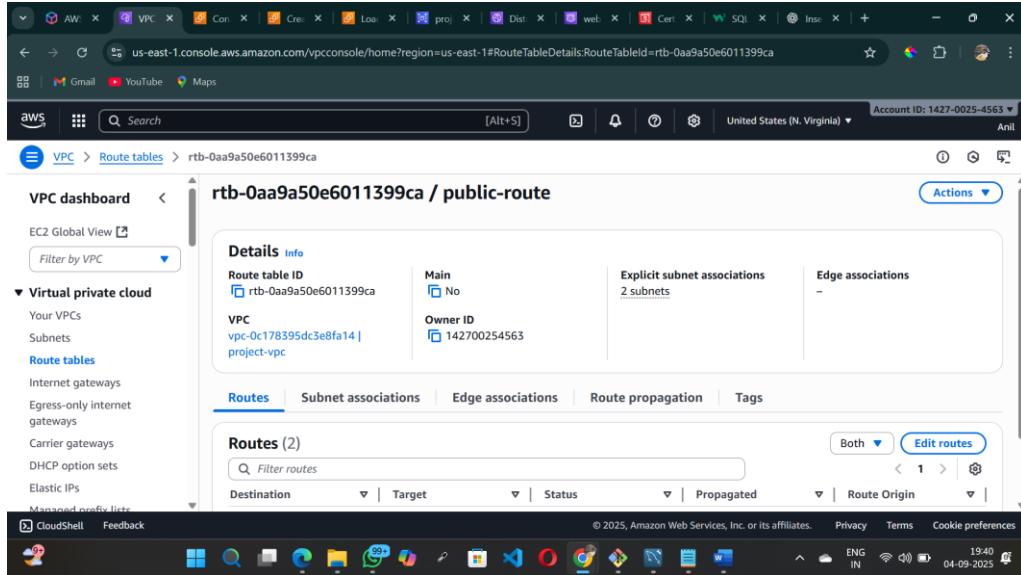


**Fig 1.3: INTERNET gateway**

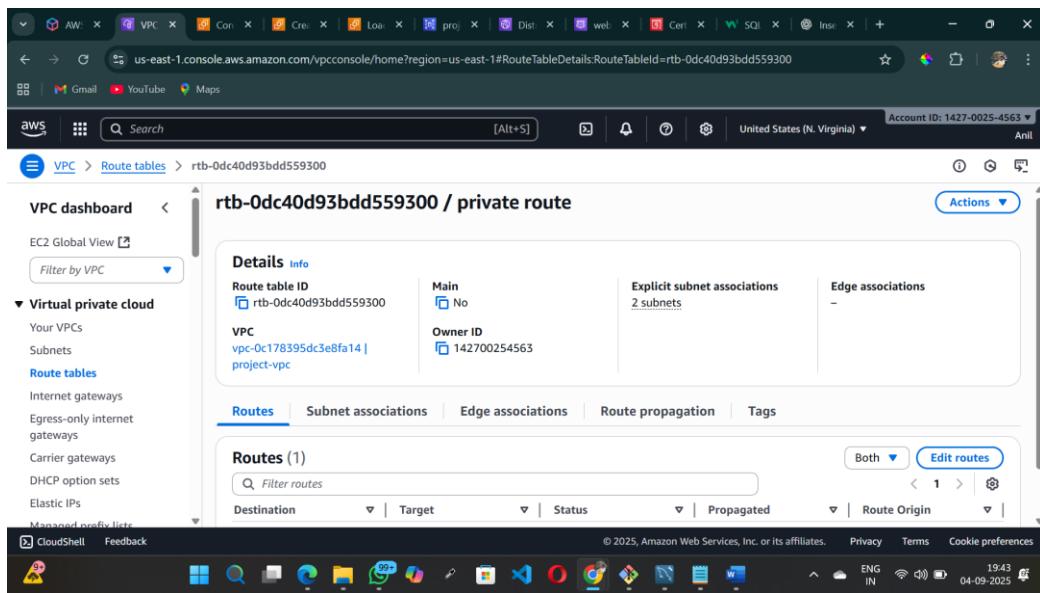
- Configure Internet Gateway and route tables for public subnets

**Fig 1.4: ROUTE TABLE**

. There are 2 route table(1.public route, 2.private route table)



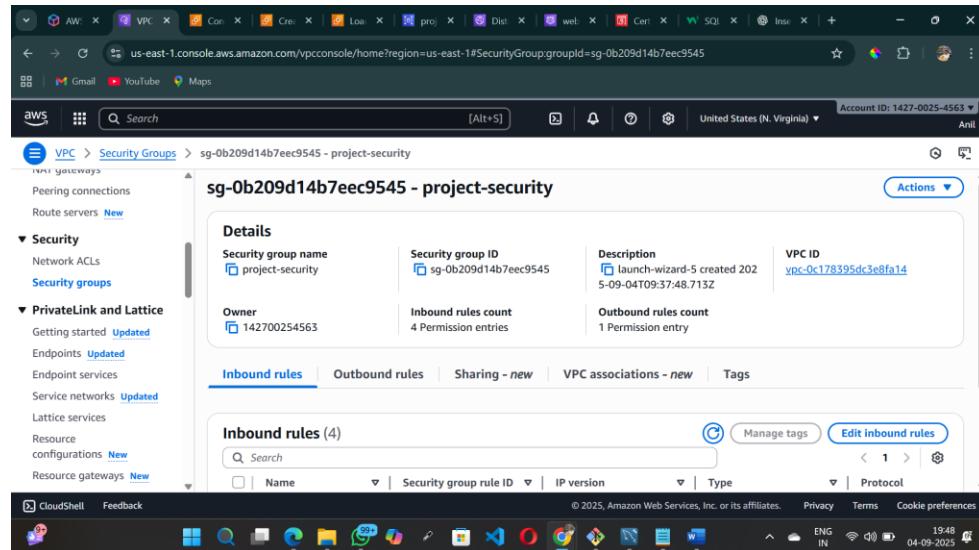
**Fig 1.5: ROUTE TABLE**



- Successfully configured VPC with all components and routes .
- Public connect with internet gateway with subnet association (PUBLIC)
- Private route connect with NAT GATEWAY with subnets association(PRIVATE)

## 2. Configure Security

- Create Security Groups for:



EC2 (allow HTTP, HTTPS &SSH traffic from anywhere-default).

Fig 2.1: Security Group for EC2

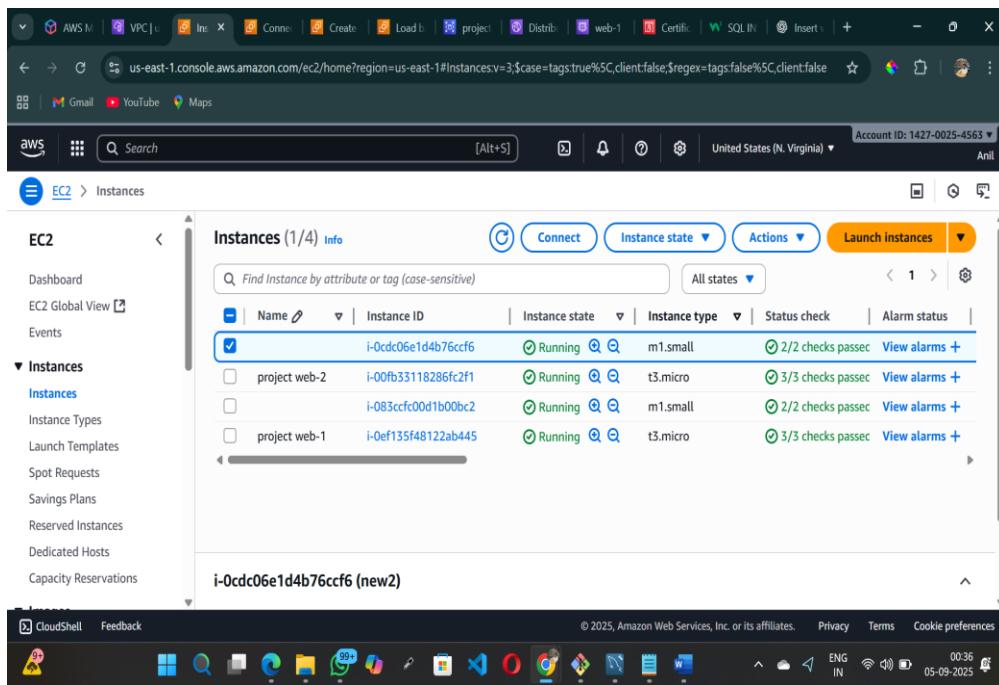
Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0f38d96391de5fd54	HTTP	TCP	80	Cus... 0.0.0.0/0	
sgr-0ce7b3e40aa61fd95	SSH	TCP	22	Cus... 0.0.0.0/0	
sgr-0919963abdfca9bd5	MySQL/Aurora	TCP	3306	Cus... 0.0.0.0/0	
sgr-031e73e27e09bcf38	HTTPS	TCP	443	Cus... 0.0.0.0/0	

RDS (allow MySQL/Aurora &SSH traffic from anywhere-rds-sg)

**Fig 3.1: EC2 INSTANCES**

### 3. Provision EC2 Instances

- Launch two EC2 instances in public subnets
  - ec2-project web-1
  - ec2-project web-2



**Fig 3.1: Ec2-1**

- **Name:** Give your instance a meaningful name (**project web-1 & project web-2**)
- **AMI:** Choose an Amazon Machine Image (**ubuntu**)
- **Instance Type:** t3.medium for better performance
- **Key Pair:** Create or select an existing key pair for SSH access

## .ec2-project web-1

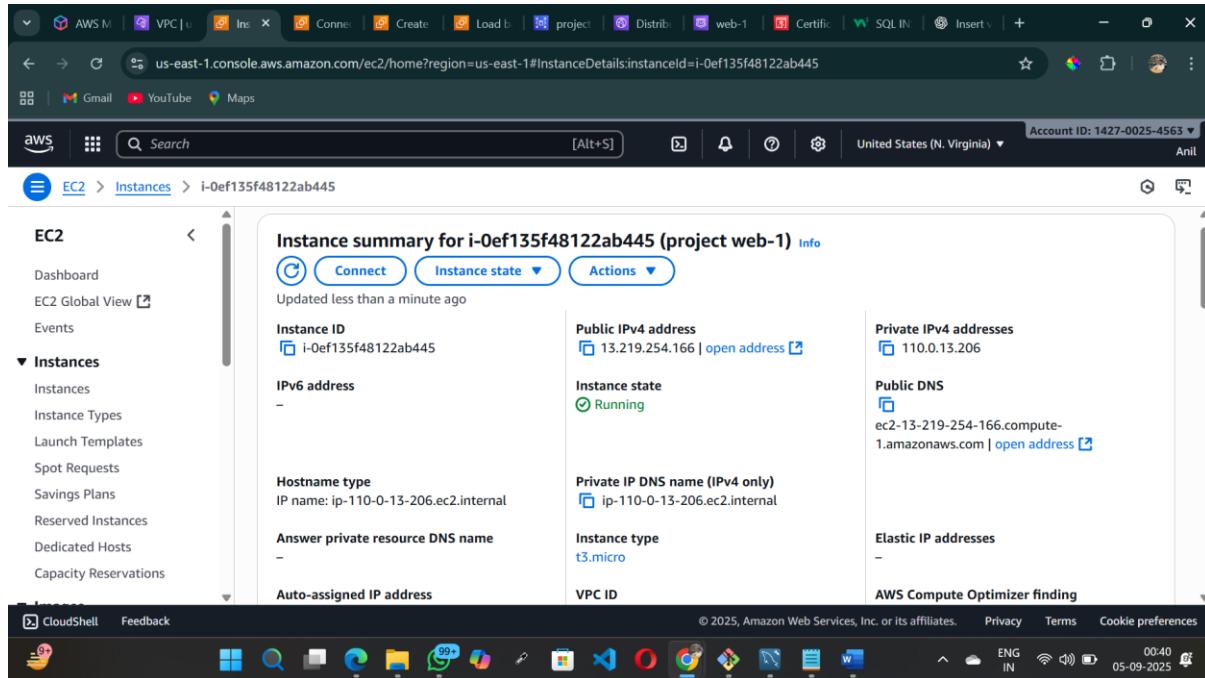
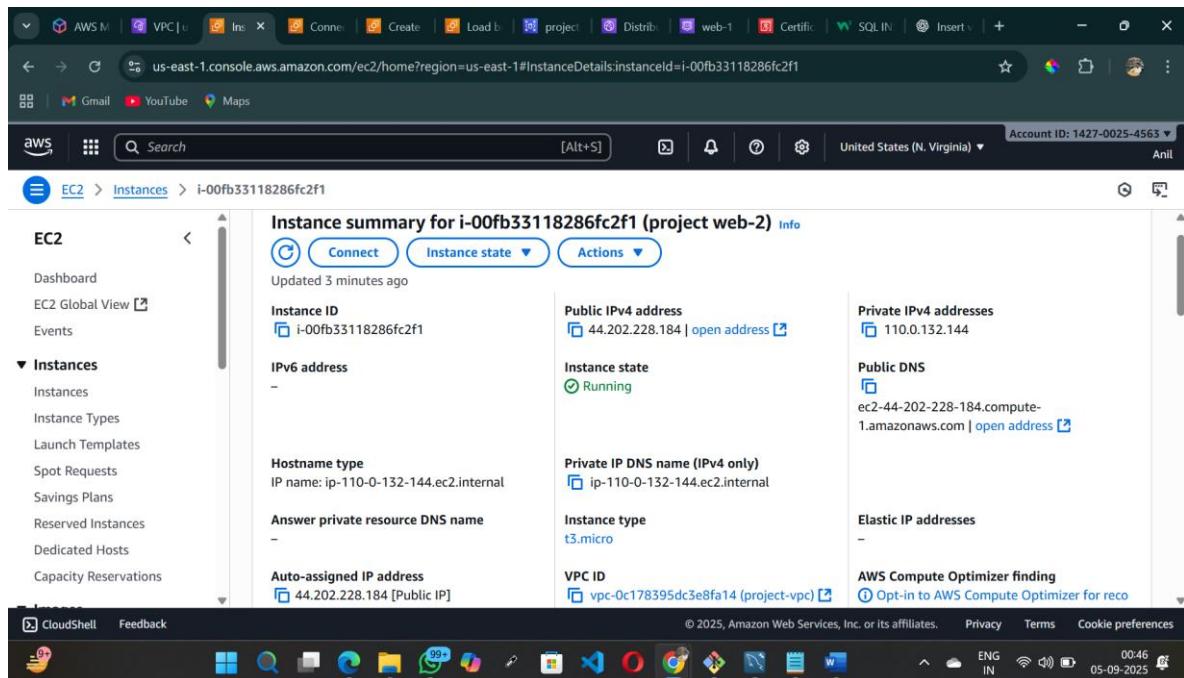


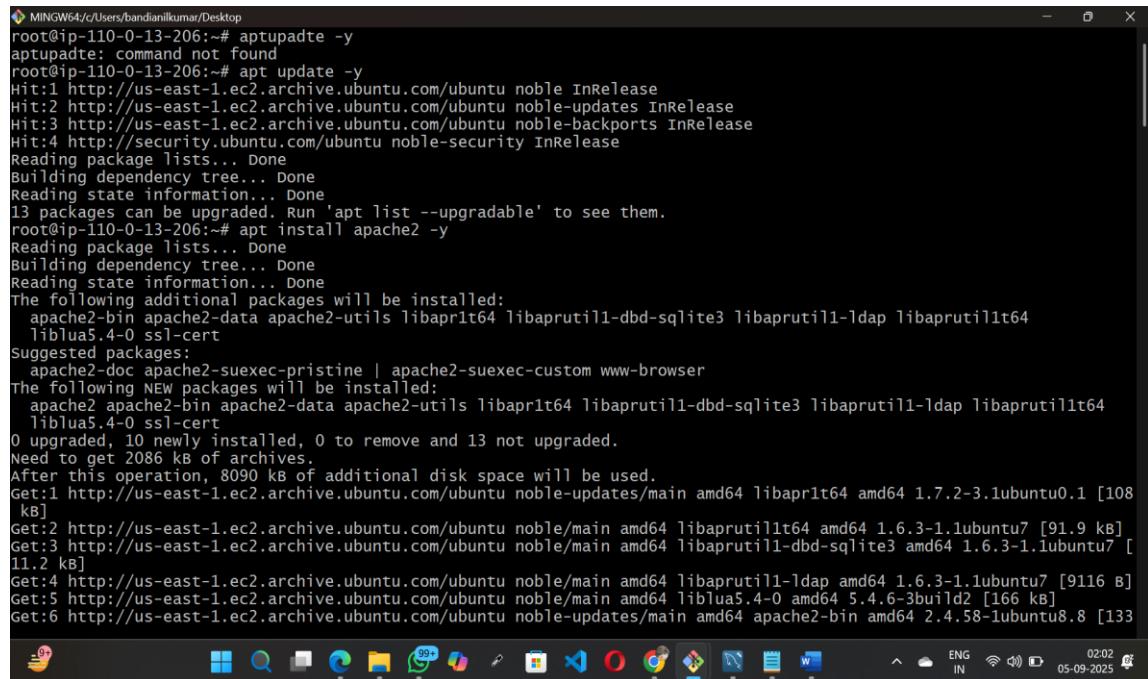
Fig 3.2: Ec2-1



Connect the both ec2 instances to gitbash with ssh link and Install your web application in them(apache2)

- Use following commands

- sudo -i
- yum update -y
- yum install apache2 -y
- cd /var/www/html/index.html (check your index.html file)
- systemctl status apache2 (if it shows inactive use below command to activate)
- Systemctl start/restart apache2

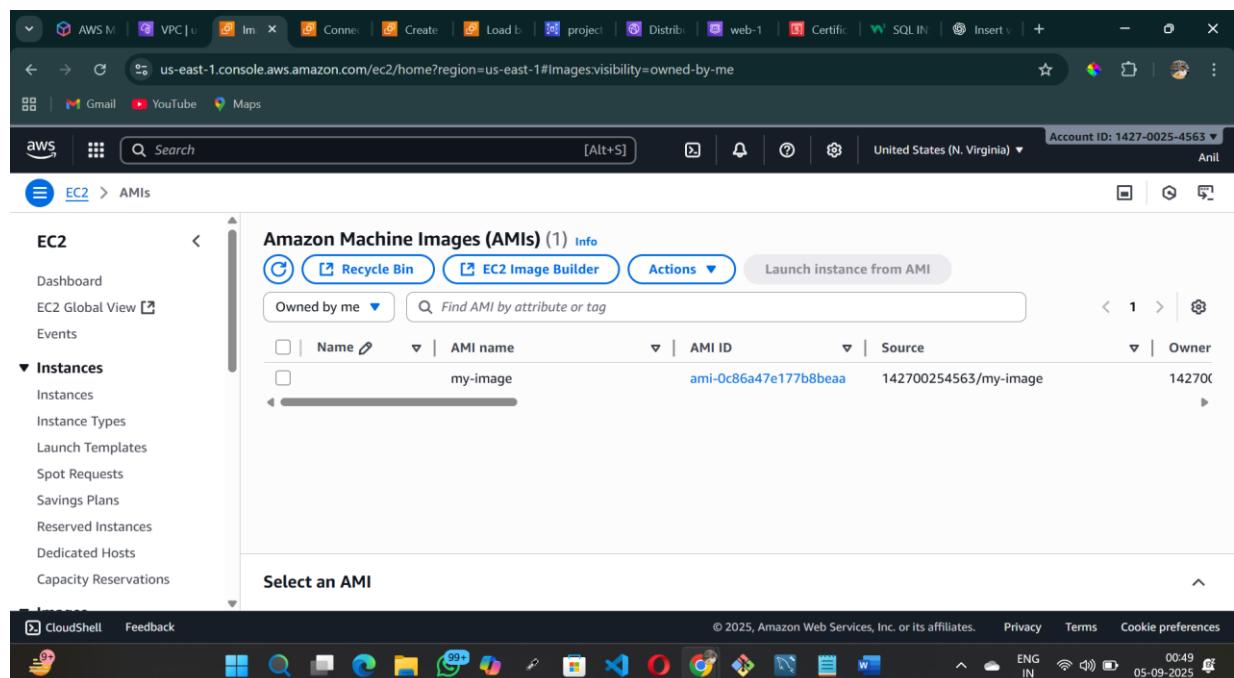


```

MINGW64/c/Users/bandianikumar/Desktop
root@ip-110-0-13-206:~# apt update -y
aptupdate: command not found
root@ip-110-0-13-206:~# apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
13 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-110-0-13-206:~# apt install apache2 -y
Reading package lists... done
Building dependency tree... done
Reading state information... done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64
  liblua5.4-0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64
  liblua5.4-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 13 not upgraded.
Need to get 2086 kB of archives.
After this operation, 8090 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libapr1t64 amd64 1.7.2-3.1ubuntu0.1 [108 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1t64 amd64 1.6.3-1.1ubuntu7 [91.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.3-1.1ubuntu7 [11.2 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-ldap amd64 1.6.3-1.1ubuntu7 [9116 B]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 liblua5.4-0 amd64 5.4.6-3build2 [166 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-bin amd64 2.4.58-1ubuntu8.8 [133 kB]
02:02 05-09-2025

```

## AMIS



**Amazon Machine Images (AMIs) (1) Info**

**Actions** ▾ **Launch instance from AMI**

Owner	Source	AMI ID	AMI name	Name
142700254563	142700254563/my-image	ami-0c86a47e177b8beaa	my-image	

**Select an AMI**

**FIG 4.1: TAGRET GROUP**

The screenshot shows the AWS EC2 Target Groups console with the URL <https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroup:targetGroupArn=arn:aws:elasticloadbalancing:us-east-1:142700254563:targetgroup/project-target/aefbf40cbc69486>. The page displays the 'Details' for the 'project-target' target group. It includes fields for Target type (Instance), Protocol (HTTP: 80), Protocol version (HTTP1), VPC (vpc-0c178395dc3e8fa14), IP address type (Load balancer), and a list of targets. The targets section shows 2 total targets, with 2 healthy (green), 0 unhealthy (red), 0 unused, 0 initial, and 0 draining. Below this, there's a section titled 'Distribution of targets by Availability Zone (AZ)' with a note to select values in the table.

Details

arn:aws:elasticloadbalancing:us-east-1:142700254563:targetgroup/project-target/aefbf40cbc69486

Target type	Protocol : Port	Protocol version	VPC
Instance	HTTP: 80	HTTP1	vpc-0c178395dc3e8fa14

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

Distribution of targets by Availability Zone (AZ)

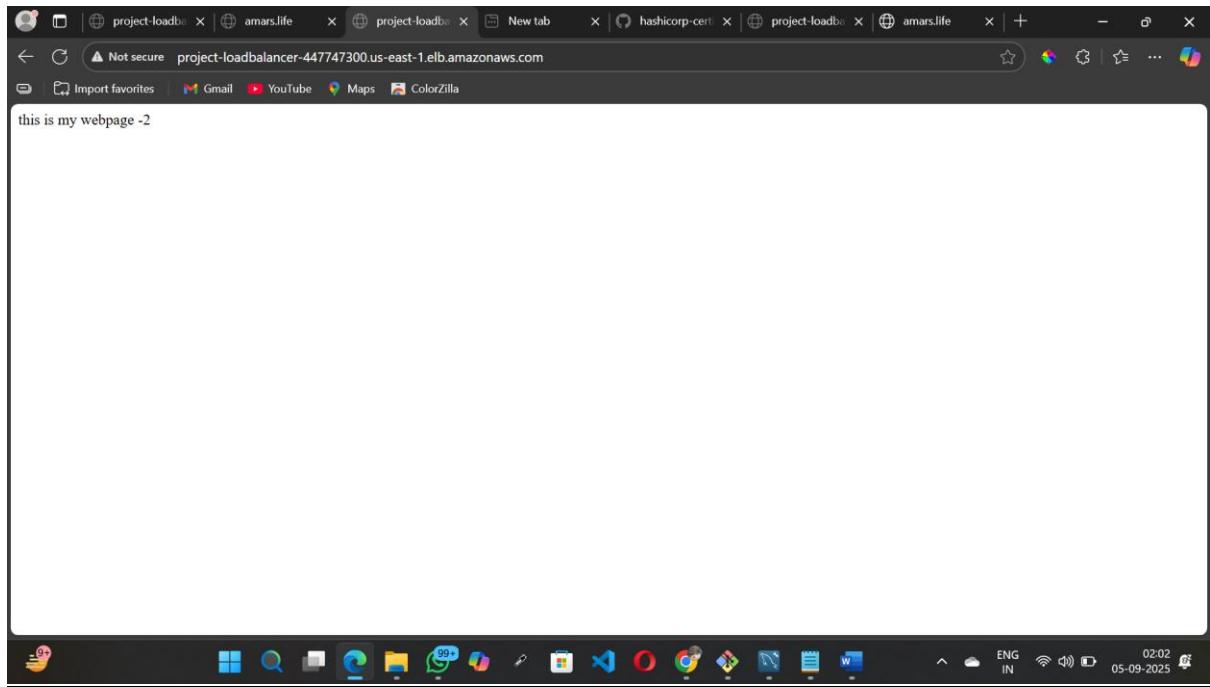
Select values in this table to see corresponding filters applied to the Registered targets table below.

## ***LOAD BALANCER***

## **FIG 4.2:LOAD BALANCER IMAGE**

The screenshot shows the AWS CloudWatch Metrics interface. The top navigation bar includes links for AWS Lambda, VPC, Create, Load balancer, project, Distrib., web-1, Certific., SQL IN, Insert, and a plus sign. The main content area displays a log stream titled 'lambda-1'. The log entries show the function's execution details, including the input payload, processing time, and status code. The log entries are timestamped and include references to AWS Lambda functions like 'lambda-1' and 'lambda-2'.

→ past DNS check ur server



Create **image** from the existing ec2 server (**img-1**).

**Fig 5.1: Amazon Machine Image (AMI)**

A screenshot of the AWS Management Console in a Microsoft Edge browser. The URL is 'us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Images:visibility=owned-by-me'. The left sidebar shows navigation options like Capacity Reservations, Images (selected), AMI Catalog, Elastic Block Store, Network &amp; Security, and CloudShell. The main content area displays a table titled 'Amazon Machine Images (AMIs) (1/1)'. The table has columns for Actions, Name, AMI ID, Source, and Owner. One entry is listed: 'my-image' with AMI ID 'ami-0c86a47e177b8beaa', Source '142700254563/my-image', and Owner '142700254563'. Below the table, there's a detailed view for 'AMI ID: ami-0c86a47e177b8beaa' with tabs for Details, Permissions, Storage, My AMI usage - new, and Tags. The Details tab shows information like AMI ID, Image type (machine), Platform details (Linux/UNIX), and Root device type (EBS). The bottom of the screen shows the Windows taskbar.

- Create **Template** for Auto scaling the servers

**Fig 5.2 AUTOSCALING**

The screenshot shows the AWS Management Console with the URL [us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AutoScalingGroupDetails:id=project-autoscalling&view=details](https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AutoScalingGroupDetails:id=project-autoscalling&view=details). The page displays the 'project-autoscalling' capacity overview, including desired capacity (2), scaling limits (Min - Max 2 - 3), and desired capacity type (Units (number of instances)). It also shows the date created (Thu Sep 04 2025 15:50:37 GMT+0530). Below the main content are tabs for Details, Integrations - new, Automatic scaling, Instance management, Instance refresh, and Actions.

→AFTER CREATING AUTOSCALING GROUPS

→GO AND CHECK EC2 INSTANCE NEW 2 SERVER ARE CREATED

→WITH NAME WE MENTIONED NEW-1 & NEW-2

**Fig 5.3: Registering Targets**

The screenshot shows the AWS Management Console with the URL [us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:v=3;\\$case=true%5C,client:false,\\$regex=tags:false%5C,client:false](https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:v=3;$case=true%5C,client:false,$regex=tags:false%5C,client:false). The page displays the 'Instances (1/4) Info' table with two entries: 'NEW-1' and 'NEW-2'. Both instances are listed as 'Running' with instance types 'm1.small' and 't3.micro' respectively. The table includes columns for Name, Instance ID, Instance state, Instance type, Status check, and Alarm status. A search bar at the top allows filtering by attribute or tag.

**Fig 6.1: RDS**

## →CREATE A SUBNETSGROUPS

The screenshot shows the AWS RDS Subnet Groups configuration page. On the left, there's a sidebar with links like Dashboard, Databases, Query editor, etc. The main area shows a subnet group named 'subnets-grop' with the following details:

VPC ID	vpc-0c178395dc3e8fa14
ARN	arn:aws:rds:us-east-1:142700254563:subgrp:subnets-grop
Supported network types	IPv4
Description	allow

Below this is a table titled 'Subnets (2)' showing two subnets:

Availability zone	Subnet name	Subnet ID	CIDR block
us-east-1b	private subnets-2	subnet-04f60afa6b6e4886	110.0.168.0/21
us-east-1a	private subnets-1	subnet-045ea0a690510333a	110.0.230.0/24

## AND CREATE A DATABASE

The screenshot shows the AWS RDS Databases configuration page. The sidebar has a 'Databases' section. The main area shows a database named 'project-database-1' with two instances: 'project-database-1' (Primary, MySQL) and 'readreplica' (Replica, MySQL). Below this is a 'Connectivity & security' tab with sections for Endpoint & port, Networking, and Security.

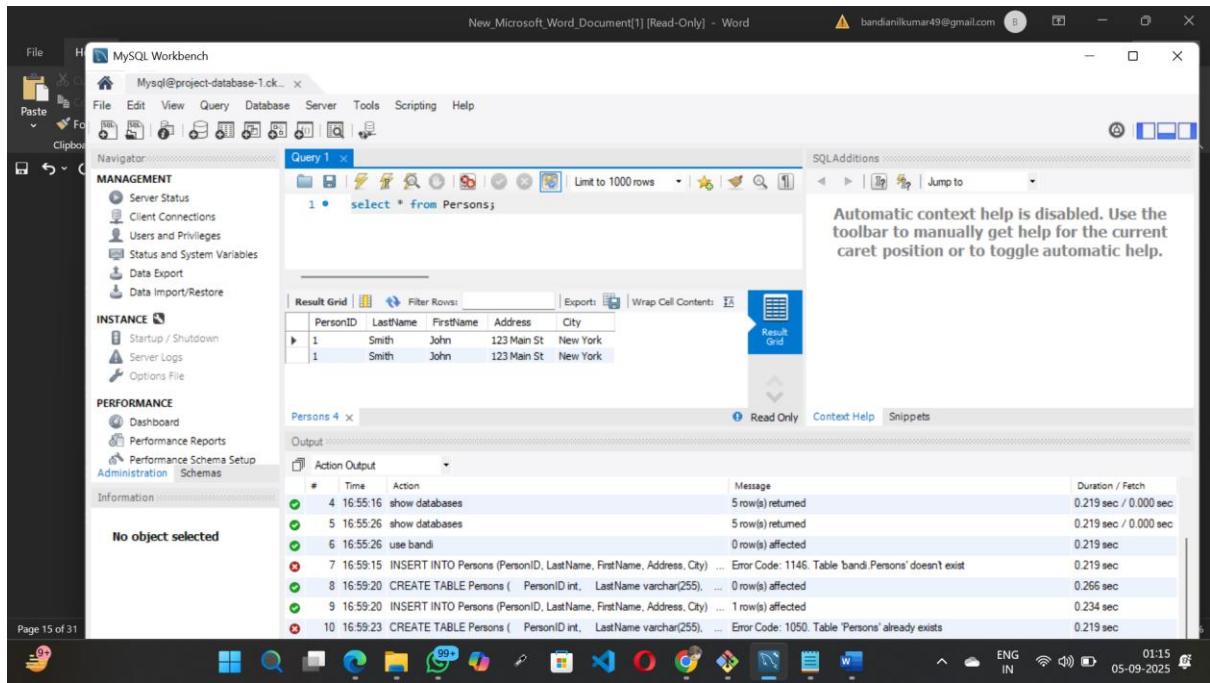
→GO SECURITY GROUPS (ADD INBOUNDRULES (MYSQL&AURORA))

→AND CREATE DATABASES IN MYSQL

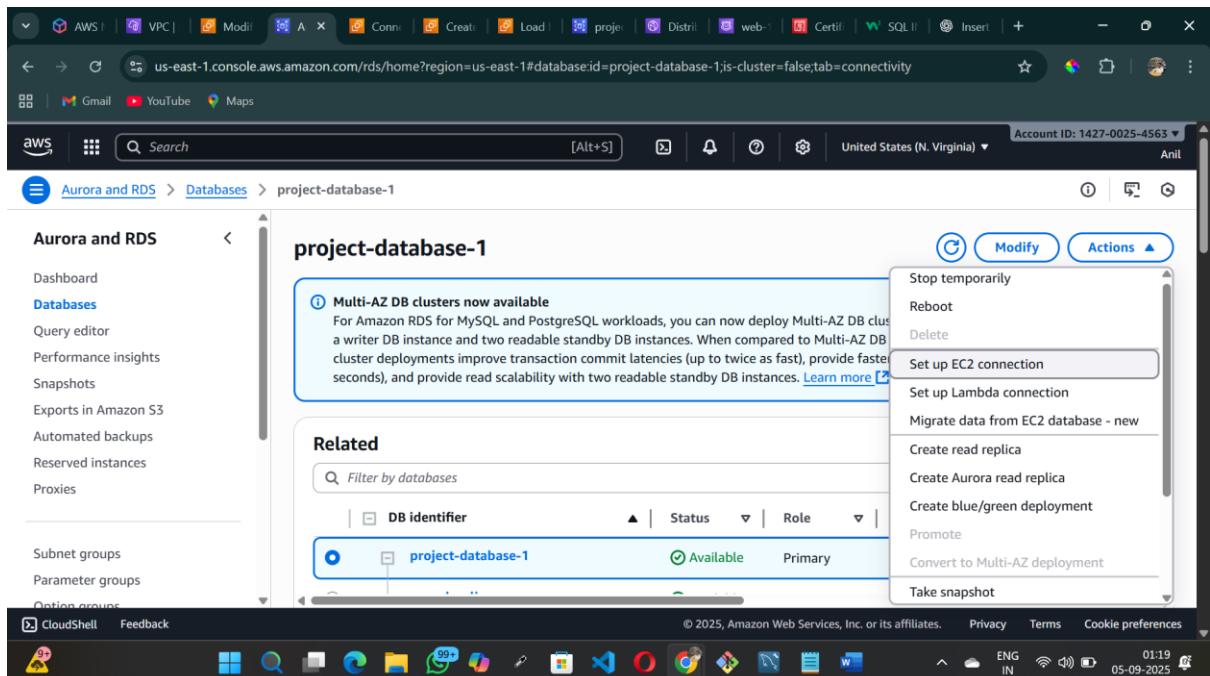
→CREATE TABLETS

→SELECT \* FROM TABLES

**FIG:6.2—MYSQL**



→ AFTER DATABASES SET UP EC2 CONNECTION



CLOUD FRONT

→ CREATE A CLOUDFRONT AND ATTACH LOADBALANCER  
→ AND ENABLE SECURITY WAF

**FIG:7.2 CLOUDFRONT**

The screenshot shows the AWS CloudFront console with the distribution 'projectcloud' selected. The 'General' tab is active. Key details shown include:

- Name:** projectcloud
- Distribution domain name:** d3vvcwmwb6rl3.cloudfront.net
- ARN:** arn:aws:cloudfront::142700254563:distribution/E1QYMANDAORZ6X
- Last modified:** September 4, 2025 at 11:48:15 AM UTC

The left sidebar shows navigation links for CloudFront, Distributions, Policies, Functions, Static IPs, VPC origins, What's new, SaaS, Multi-tenant distributions, Distribution tenants, Telemetry, Monitoring, and Alarms.

## ROUTE53

→CREATE ROUTE53

→CREATE HOSTED ZONE

→FULL DOMAIN NAME

→CREATE A 2 CREATE RECORD

→NS DOMAIN 4 SERVER CHANGE

→CNAME ROUTE53

Route 53 > Hosted zones > amars.life

Records (5) [Info](#)

[Delete record](#) [Import zone file](#) [Create record](#)

Automatic mode is the current search behavior optimized for best filter results. [To change modes go to settings.](#)

Filter records by property or value Type Routing p... Alias

Record ...	Type	Routin...	Differ...	Alias	Value
amars.life	A	Simple	-	Yes	duan
amars.life	NS	Simple	-	No	ns-1
amars.life	SOA	Simple	-	No	ns-1
amars.life	MX	Simple	-	No	ns-2

this is my webpage -2

→ TO SECURE SERVER USING (ACM)

## ACM(AMAZON CERTIFICATE MANAGER)

→ CREATE A REQUEST CERTIFICATE

→ FULL DOMAIN NAME (-----)

→ CREATE RECORDS ROUTE53

→ TO SECURE SERVER (HTTPS )

The screenshot shows the AWS Certificate Manager (ACM) interface. On the left, there's a sidebar with options like 'List certificates', 'Request certificate', 'Import certificate', and 'AWS Private CA'. The main area displays a certificate with the identifier '4189d3fd-e4bb-4054-b8f3-86807d84c96c'. The 'Certificate status' section shows 'Status' as 'Issued'. Below it, the 'ARN' is listed as 'arn:aws:acm:us-east-1:142700254563:certificate/4189d3fd-e4bb-4054-b8f3-86807d84c96c'. The 'Type' is 'Amazon Issued'. At the bottom, there's a table titled 'Domains (1)' with one entry: 'amars.life'. There are buttons for 'Create records in Route 53' and 'Export to CSV'.

→ GO TO SECURITY GROUPS (HTTPS)

→ THEN GO TO LOADBALANCER AND ADD LISTENER(HTTPS)

The screenshot shows a web browser window with a single tab open at 'https://amars.life'. The address bar shows the secure connection. The page itself contains the text 'this is my web-1'. The browser interface includes a toolbar with various icons and a taskbar at the bottom showing other open applications like Microsoft Word and Excel.

→ USE UR DOMAIN NAME PAST IN BROWSE & WE CAN SEE THE SERVER WAS SECURE

# CONCLUSION

The architecture represents a robust, scalable, and secure deployment of a web application on AWS, designed to ensure high availability and performance. It begins with Route 53, which handles DNS resolution and directs user traffic to CloudFront, a global content delivery network that accelerates access while integrating with AWS Certificate Manager (ACM) for secure HTTPS communication. AWS WAF adds a layer of protection against common web threats. Traffic is then routed to an Application Load Balancer (ALB), which distributes requests across EC2 instances hosted in public subnets across multiple Availability Zones, ensuring fault tolerance and scalability. These EC2 instances are managed by an Auto Scaling Group, which dynamically adjusts capacity based on demand. The backend database layer is securely placed in private subnets using Amazon RDS, configured with a writer instance and a reader instance across different zones to support read-heavy workloads and disaster recovery. The entire setup resides within a Virtual Private Cloud (VPC), providing network isolation and control. Overall, this architecture exemplifies AWS best practices by combining performance optimization, security hardening, and operational efficiency—making it an ideal blueprint for hosting modern, production-grade web applications.