Design and Implementation of a Secure, Scalable, Production-Ready AWS 3-Tier Architecture

“Built using Amazon VPC, ALB, Auto Scaling, EC2, RDS, NAT Gateway, and DynamoDB”

Summary:

This project demonstrates the real-world design and implementation of a secure, scalable, and highly available **3-Tier Web Application Architecture on AWS**. The architecture strictly follows industry best practices by isolating public and private resources, enforcing least-privilege access, and enabling horizontal scaling. The solution uses an **Application Load Balancer** as the single public entry point, **Auto Scaling EC2 instances** in private subnets for application processing, and **Amazon RDS (MySQL)** in isolated database subnets for persistent storage. **Amazon DynamoDB** is integrated for serverless, low-latency data access. This design eliminates single points of failure, improves security posture, optimizes cost, and reflects architectures used in **real production environments**.

Problem Statement:

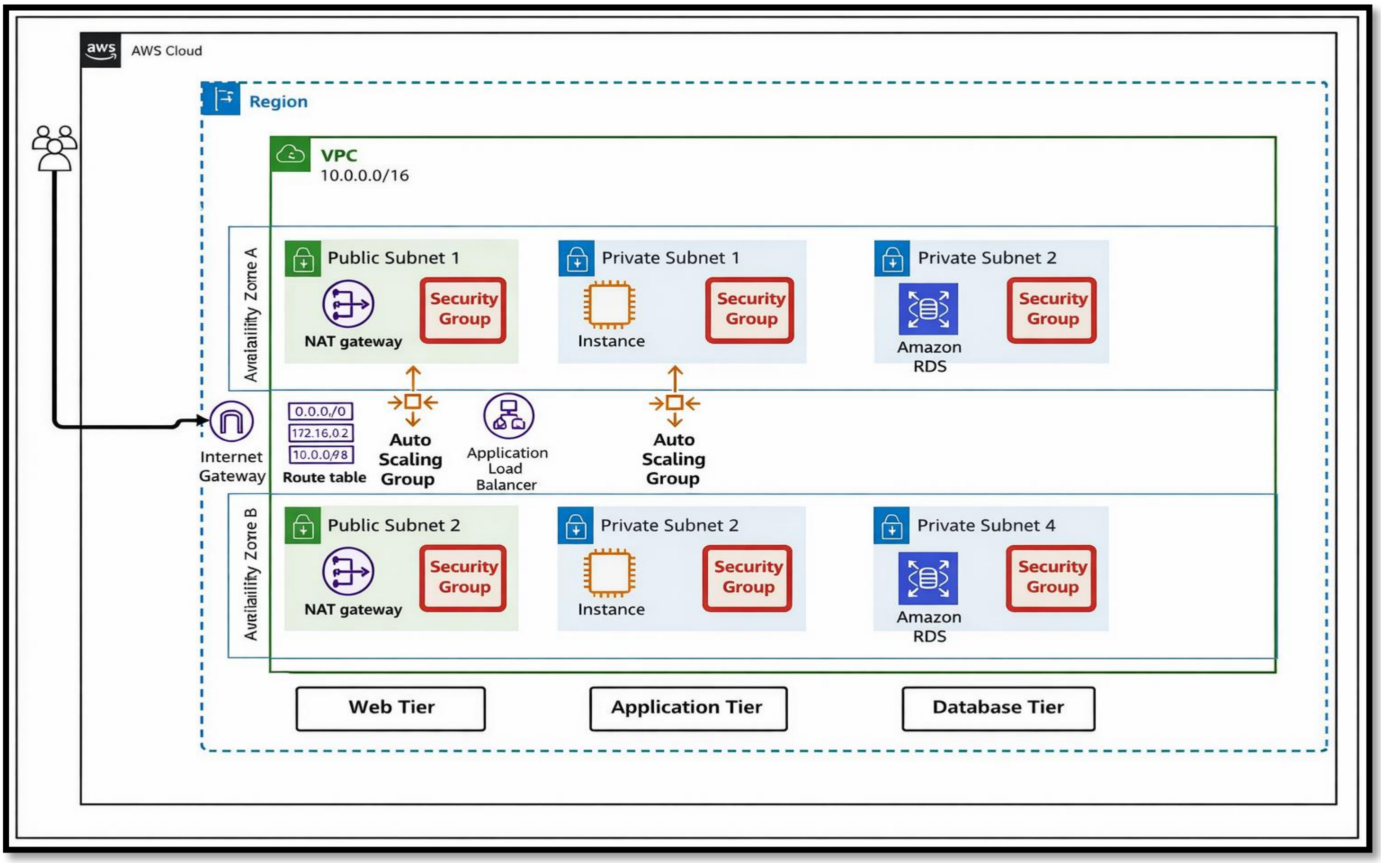
Traditional single-server deployments suffer from critical limitations such as single points of failure, poor scalability, security vulnerabilities, and difficult maintenance. As application traffic grows, such architectures fail to provide resilience and performance guarantees. This project addresses these limitations by implementing a **3-Tier architecture** that separates presentation, business logic, and data storage into independent, scalable layers.

Architecture Overview:

The architecture follows a secure and scalable 3-Tier design within a single Amazon VPC. Incoming user traffic enters through an Internet Gateway and is handled by an Application Load Balancer deployed in public subnets across multiple Availability Zones. The ALB forwards requests to healthy EC2 instances managed by an Auto Scaling Group in private application subnets, ensuring high availability and fault tolerance. These backend instances process application logic and securely communicate with Amazon RDS and DynamoDB deployed in private database subnets. Outbound internet access for private resources is enabled through a NAT Gateway, while inbound access is strictly restricted. This architecture ensures strong security isolation, scalability, and production-grade reliability.

- Eliminates single point of failure using multi-AZ
- Ensures security by isolating private resources
- Prevents direct access to backend and database

- Supports automatic scaling using Auto Scaling Group
- Enables controlled internet access via NAT Gateway
- Follows AWS Well-Architected Framework principles



Here, we have 3-architecture and it is as:

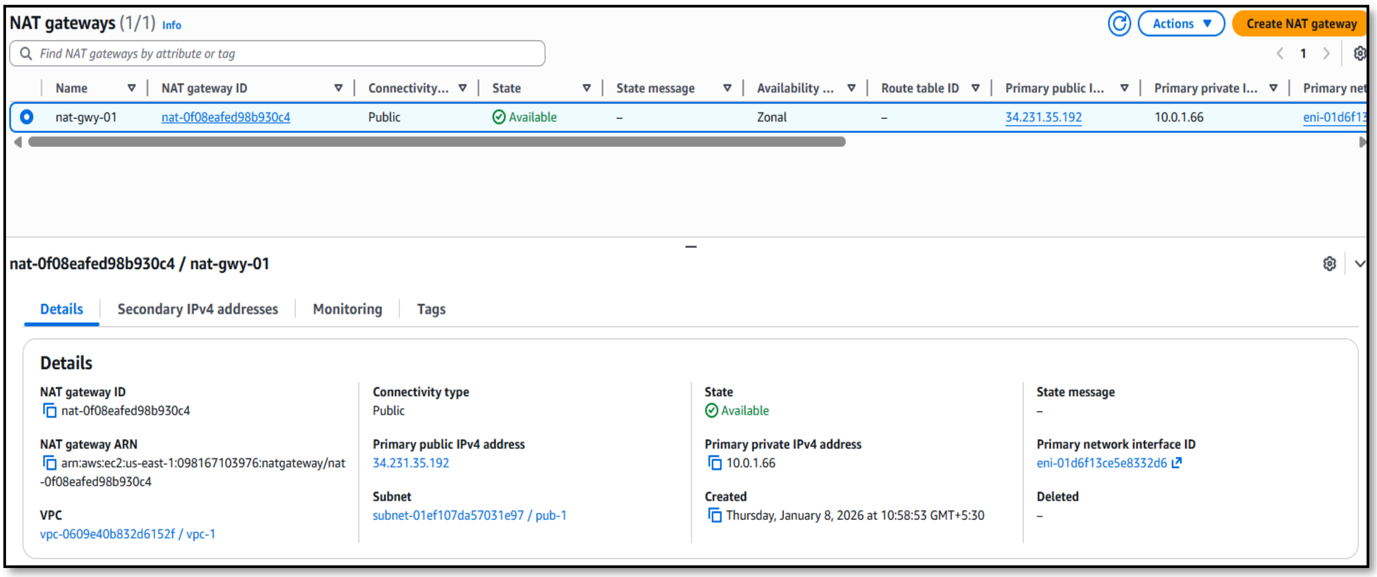
Security Design:

Security Groups were designed using tier-to-tier trust boundaries. The Load Balancer Security Group allows HTTP/HTTPS traffic from the internet. Application instances accept traffic only from the Load Balancer Security Group. Database instances accept traffic only from the Application Security Group on port 3306. This enforces strict isolation and prevents lateral movement within the network.

Network Design (VPC Layer)

- **Virtual Private Cloud (VPC)**

  1. A dedicated VPC acts as an isolated network boundary for all resources
  2. Provides full control over IP addressing, routing, and security



	Name	NAT gateway ID	Connectivity...	State	State message	Availability ...	Route table ID	Primary public I...	Primary private I...	Primary net
●	nat-gwy-01	nat-0f08eafed98b930c4	Public	✓ Available	–	Zonal	–	34.231.35.192	10.0.1.66	eni-01d6f13

nat-0f08eafed98b930c4 / nat-gwy-01

Details

NAT gateway ID
nat-0f08eafed98b930c4

NAT gateway ARN
arn:aws:ec2:us-east-1:098167103976:natgateway/nat-0f08eafed98b930c4

VPC
vpc-0609e40b832d6152f / vpc-1

Connectivity type
Public

Primary public IPv4 address
34.231.35.192

Subnet
subnet-01ef107da57031e97 / pub-1

State
✓ Available

Primary private IPv4 address
10.0.1.66

Created
Thursday, January 8, 2026 at 10:58:53 GMT+5:30

State message
–

Primary network interface ID
eni-01d6f13ce5e8332d6

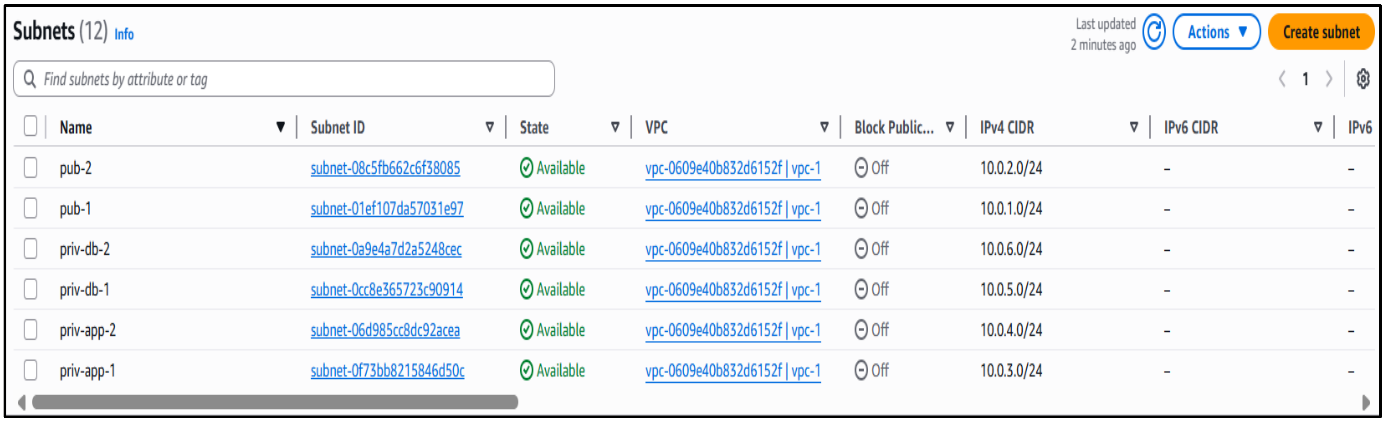
Deleted
–

- **Subnet Design**

The VPC is divided into multiple subnets across two Availability Zones:

1. **Public Subnets:** Host the Application Load Balancer and NAT Gateway
2. **Private Application Subnets:** Host backend EC2 instances
3. **Private Database Subnets:** Host Amazon RDS

This separation ensures that sensitive resources remain inaccessible from the internet.



□	Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR	IPv6 CIDR	IPv6
□	pub-2	subnet-08c5fb662c6f38085	✓ Available	vpc-0609e40b832d6152f | vpc-1	⊖ Off	10.0.2.0/24	–	–
□	pub-1	subnet-01ef107da57031e97	✓ Available	vpc-0609e40b832d6152f | vpc-1	⊖ Off	10.0.1.0/24	–	–
□	priv-db-2	subnet-0a9e4a7d2a5248cec	✓ Available	vpc-0609e40b832d6152f | vpc-1	⊖ Off	10.0.6.0/24	–	–
□	priv-db-1	subnet-0cc8e365723c90914	✓ Available	vpc-0609e40b832d6152f | vpc-1	⊖ Off	10.0.5.0/24	–	–
□	priv-app-2	subnet-06d985cc8dc92acea	✓ Available	vpc-0609e40b832d6152f | vpc-1	⊖ Off	10.0.4.0/24	–	–
□	priv-app-1	subnet-0f73bb8215846d50c	✓ Available	vpc-0609e40b832d6152f | vpc-1	⊖ Off	10.0.3.0/24	–	–

Web / Presentation Tier

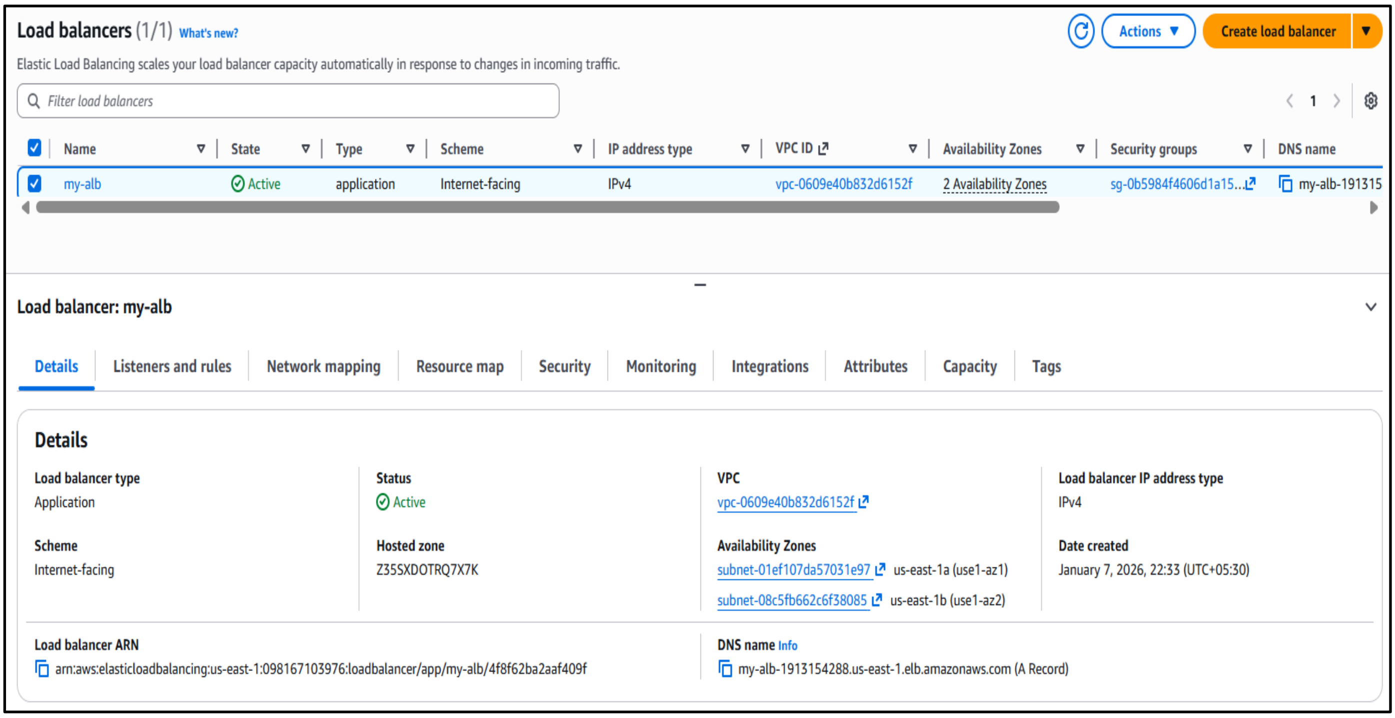
- **Application Load Balancer (ALB)**

The Application Load Balancer serves as the single public entry point to the application.

Responsibilities:

1. Accepts HTTP/HTTPS traffic from users
2. Distributes requests across healthy backend instances
3. Performs health checks
4. Protects backend resources from direct exposure

The ALB is deployed across multiple public subnets to ensure high availability.



Load balancers (1/1) What's new?

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

☑	Name	State	Type	Scheme	IP address type	VPC ID	Availability Zones	Security groups	DNS name
☑	my-alb	✓ Active	application	Internet-facing	IPv4	vpc-0609e40b832d6152f	2 Availability Zones	sg-0b5984f4606d1a15...	my-alb-191315

Load balancer: my-alb

Details

Load balancer type
Application

Status
✓ Active

VPC
vpc-0609e40b832d6152f

Load balancer IP address type
IPv4

Scheme
Internet-facing

Hosted zone
Z35SXDOTRQ7X7K

Availability Zones
subnet-01ef107da57031e97 us-east-1a (use1-az1)
subnet-08c5fb662c6f38085 us-east-1b (use1-az2)

Date created
January 7, 2026, 22:33 (UTC+05:30)

Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:098167103976:loadbalancer/app/my-alb/4f8f62ba2aaf409f

DNS name Info
my-alb-1913154288.us-east-1.elb.amazonaws.com (A Record)

Application Tier

- **EC2 Auto Scaling Group**

  The Application Tier runs on EC2 instances hosted in private subnets and managed by an Auto Scaling Group.

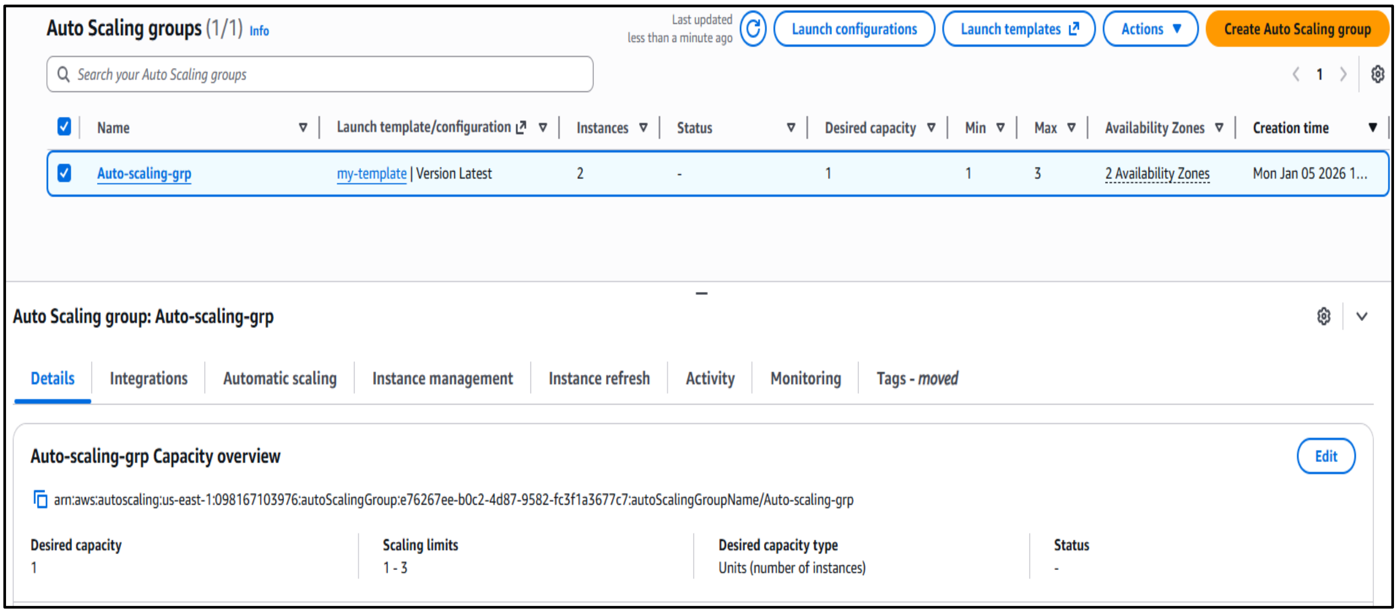
  Key characteristics:

  a. No public IP addresses

  b. Scales automatically based on demand

  c. Automatically replaces unhealthy instances

  d. Accessible only from the ALB



Auto Scaling groups (1/1) Info

☑	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones	Creation time
☑	Auto-scaling-grp	my-template | Version Latest	2	-	1	1	3	2 Availability Zones	Mon Jan 05 2026 1...

Auto Scaling group: Auto-scaling-grp

Auto-scaling-grp Capacity overview

arn:aws:autoscaling:us-east-1:098167103976:autoScalingGroup:e76267ee-b0c2-4d87-9582-fc3f1a3677c7:autoScalingGroupName/Auto-scaling-grp

Desired capacity
1

Scaling limits
1 - 3

Desired capacity type
Units (number of instances)

Status
-

- **Launch Template:**

  A Launch Template defines:

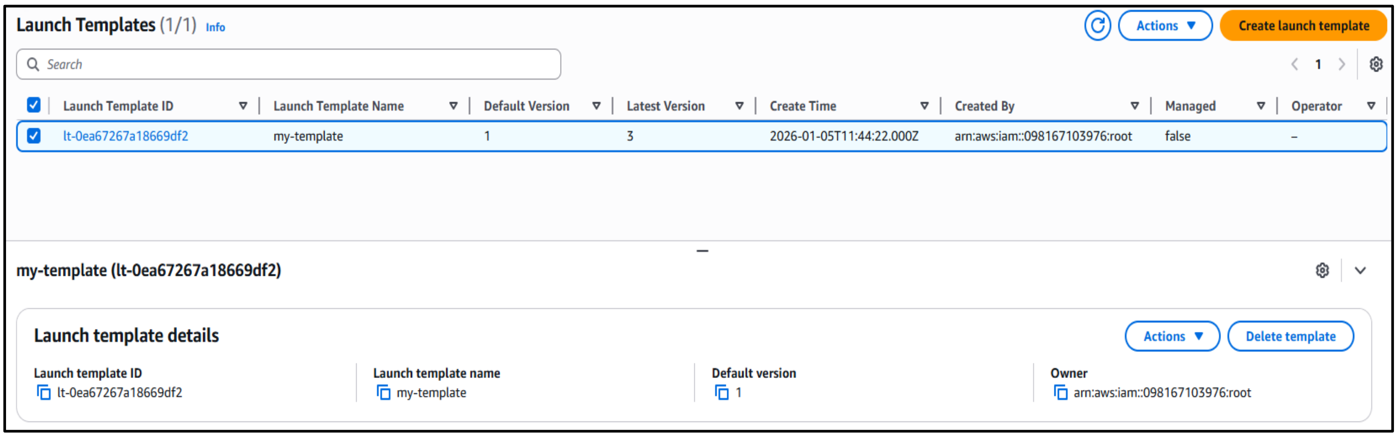
  a. Ubuntu AMI

  b. Instance type

  c. Security groups

  d. Startup scripts (user data)

  This ensures consistency and repeatability when instances are launched or replaced.



☑	Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By	Managed	Operator
☑	lt-0ea67267a18669df2	my-template	1	3	2026-01-05T11:44:22.000Z	arn:aws:iam::098167103976:root	false	–

my-template (lt-0ea67267a18669df2)

Launch template details

Launch template ID	Launch template name	Default version	Owner
lt-0ea67267a18669df2	my-template	1	arn:aws:iam::098167103976:root

Database Tier

- **Amazon RDS (MySQL)**

  Amazon RDS is used as the primary relational database.

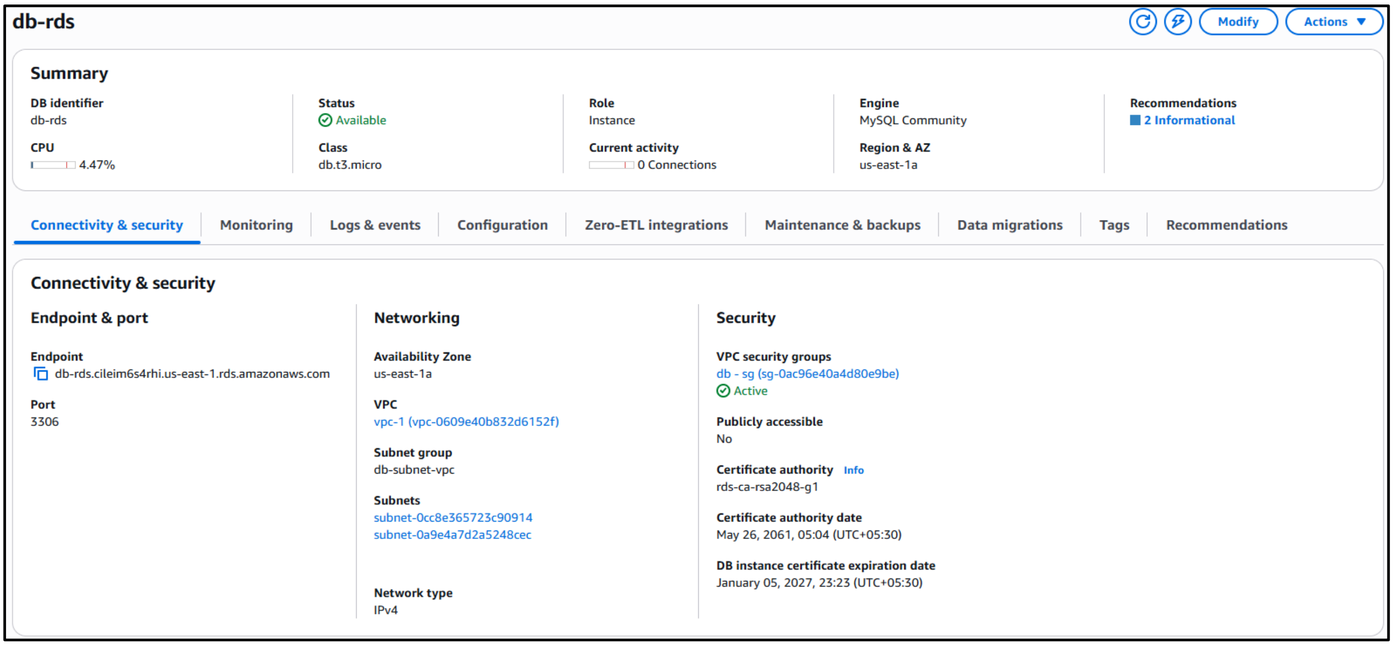
  Features:

  a. Deployed in private database subnets

  b. No public accessibility

  c. Accessible only from the Application Tier

  d. Managed backups and automated maintenance



db-rds

Summary

DB identifier	Status	Role	Engine	Recommendations
db-rds	✓ Available	Instance	MySQL Community	2 Informational
CPU	**Class**	**Current activity**	**Region & AZ**	
4.47%	db.t3.micro	0 Connections	us-east-1a	

Connectivity & security

Endpoint & port

Endpoint
db-rds.cileim6s4rhi.us-east-1.rds.amazonaws.com

Port
3306

Networking

Availability Zone
us-east-1a

VPC
vpc-1 (vpc-0609e40b832d6152f)

Subnet group
db-subnet-vpc

Subnets
subnet-0cc8e365723c90914
subnet-0a9e4a7d2a5248cec

Network type
IPv4

Security

VPC security groups
db - sg (sg-0ac96e40a4d80e9be)
✓ Active

Publicly accessible
No

Certificate authority Info
rds-ca-rsa2048-g1

Certificate authority date
May 26, 2061, 05:04 (UTC+05:30)

DB instance certificate expiration date
January 05, 2027, 23:23 (UTC+05:30)

- **Amazon DynamoDB**

  DynamoDB is integrated as a serverless NoSQL datastore.

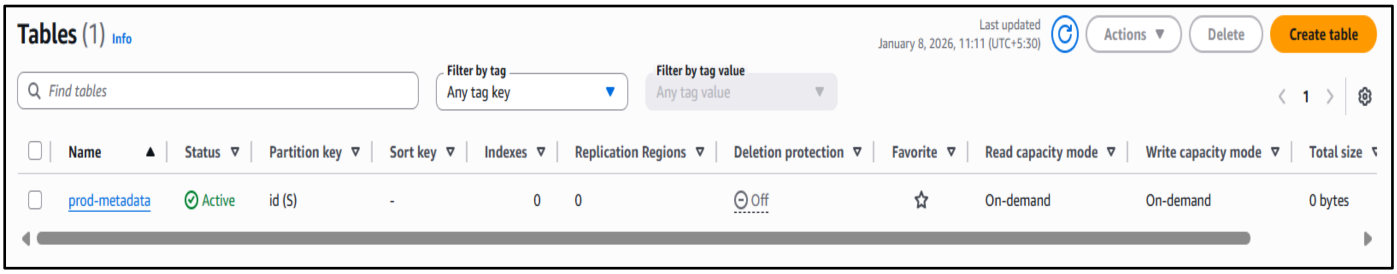
  Use cases:

  a. Metadata storage

  b. Session data

  c. Fast key-value lookups

  DynamoDB requires no subnet configuration and scales automatically.



□	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode	Write capacity mode	Total size
□	prod-metadata	✓ Active	id (S)	-	0	0	Off	☆	On-demand	On-demand	0 bytes

Security Design

Security is enforced using defense-in-depth principles.

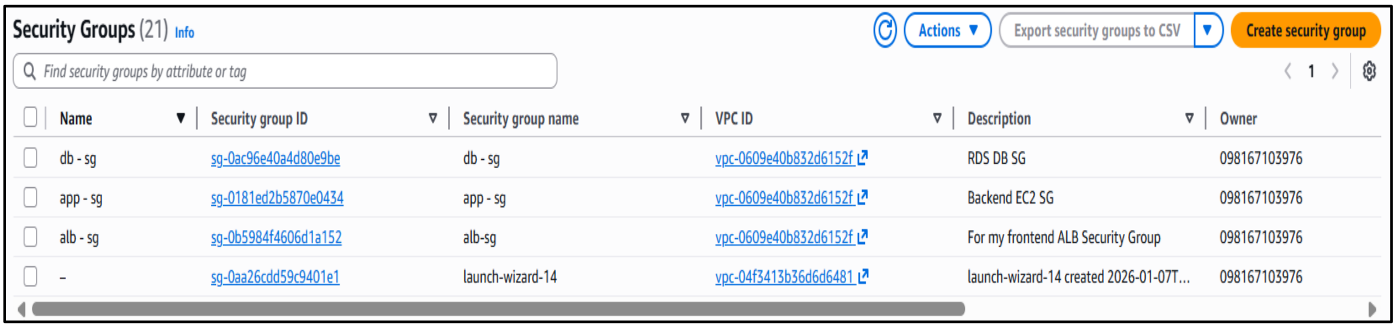
- **Security Groups**

  a. **ALB Security Group:** Allows HTTP/HTTPS traffic from the internet

  b. **Application Security Group:** Allows traffic only from the ALB

  c. **Database Security Group:** Allows database access only from the application tier

  This ensures strict tier-to-tier communication and prevents unauthorized access.



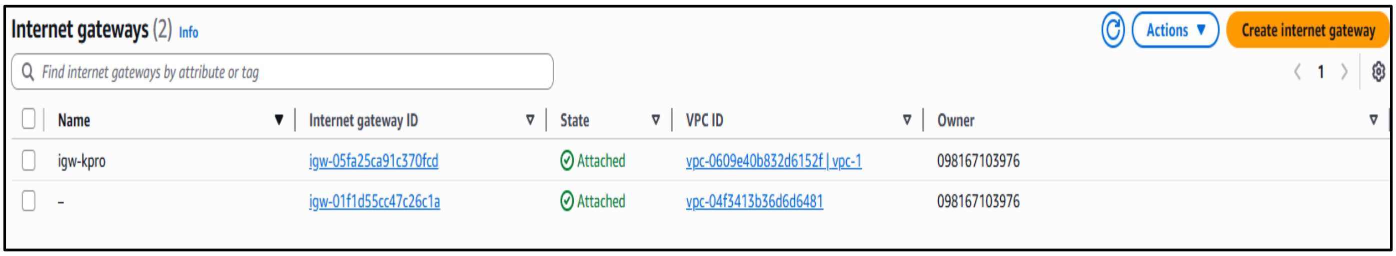
Security Groups (21) Info
Actions
Export security groups to CSV
Create security group
Find security groups by attribute or tag
1

□	Name	Security group ID	Security group name	VPC ID	Description	Owner
□	db - sg	sg-0ac96e40a4d80e9be	db - sg	vpc-0609e40b832d6152f	RDS DB SG	098167103976
□	app - sg	sg-0181ed2b5870e0434	app - sg	vpc-0609e40b832d6152f	Backend EC2 SG	098167103976
□	alb - sg	sg-0b5984f4606d1a152	alb-sg	vpc-0609e40b832d6152f	For my frontend ALB Security Group	098167103976
□	-	sg-0aa26cdd59c9401e1	launch-wizard-14	vpc-04f3413b36d6d6481	launch-wizard-14 created 2026-01-07T...	098167103976

Internet Access Control

- **Internet Gateway**

  - Enables inbound and outbound internet connectivity for public resources

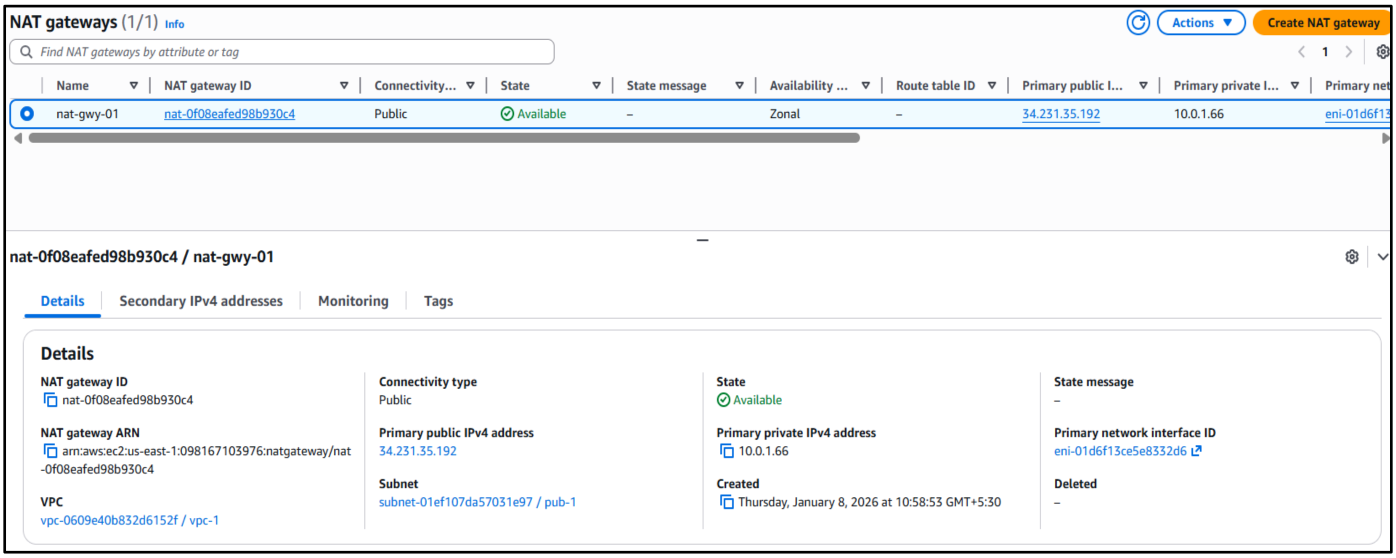


Internet gateways (2) Info
Actions
Create internet gateway
Find internet gateways by attribute or tag
1

□	Name	Internet gateway ID	State	VPC ID	Owner
□	igw-kpro	igw-05fa25ca91c370fcd	✓ Attached	vpc-0609e40b832d6152f | vpc-1	098167103976
□	-	igw-01f1d55cc47c26c1a	✓ Attached	vpc-04f3413b36d6d6481	098167103976

- **NAT Gateway**
  a. Enables outbound-only internet access for private EC2 instances
  b. Prevents inbound internet traffic to private resources

This design allows private instances to install updates or access external services securely.



	Name	NAT gateway ID	Connectivity...	State	State message	Availability ...	Route table ID	Primary public I...	Primary private I...	Primary net
●	nat-gwy-01	nat-0f08eafed98b930c4	Public	✓ Available	–	Zonal	–	34.231.35.192	10.0.1.66	eni-01d6f13

nat-0f08eafed98b930c4 / nat-gwy-01

Details

NAT gateway ID
nat-0f08eafed98b930c4

NAT gateway ARN
arn:aws:ec2:us-east-1:098167103976:natgateway/nat-0f08eafed98b930c4

VPC
vpc-0609e40b832d6152f / vpc-1

Connectivity type
Public

Primary public IPv4 address
34.231.35.192

Subnet
subnet-01ef107da57031e97 / pub-1

State
✓ Available

Primary private IPv4 address
10.0.1.66

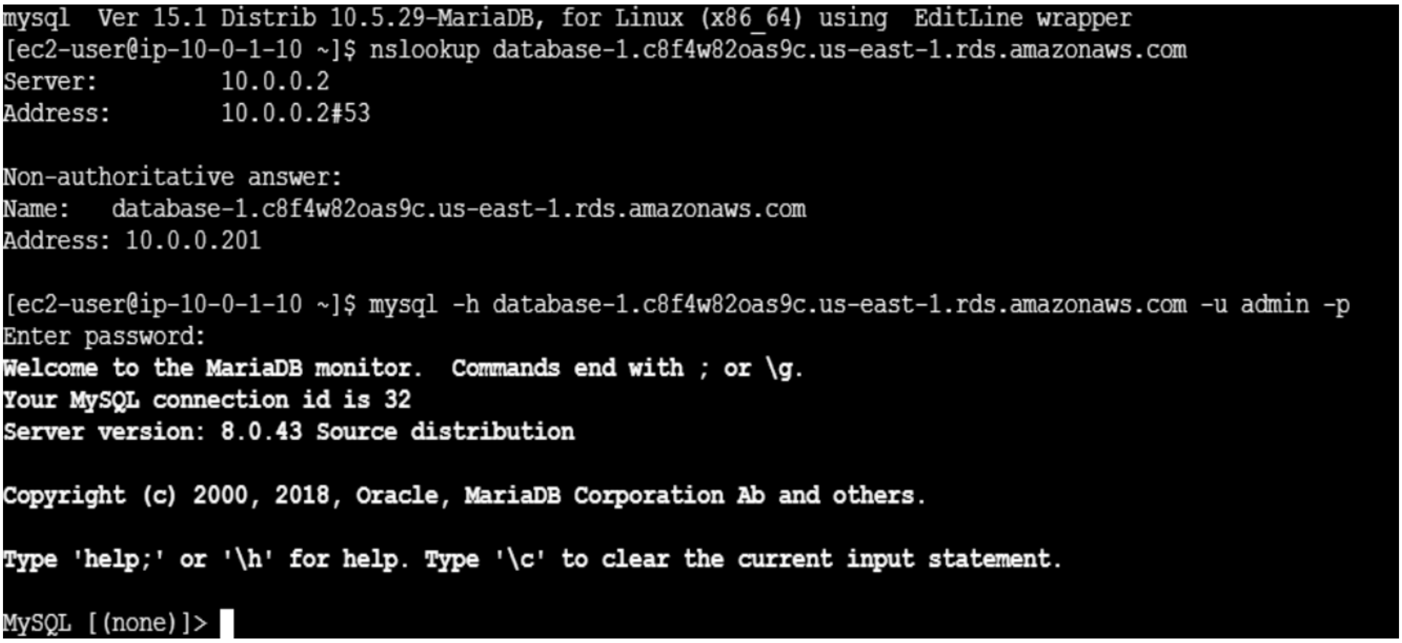
Created
Thursday, January 8, 2026 at 10:58:53 GMT+5:30

State message
–

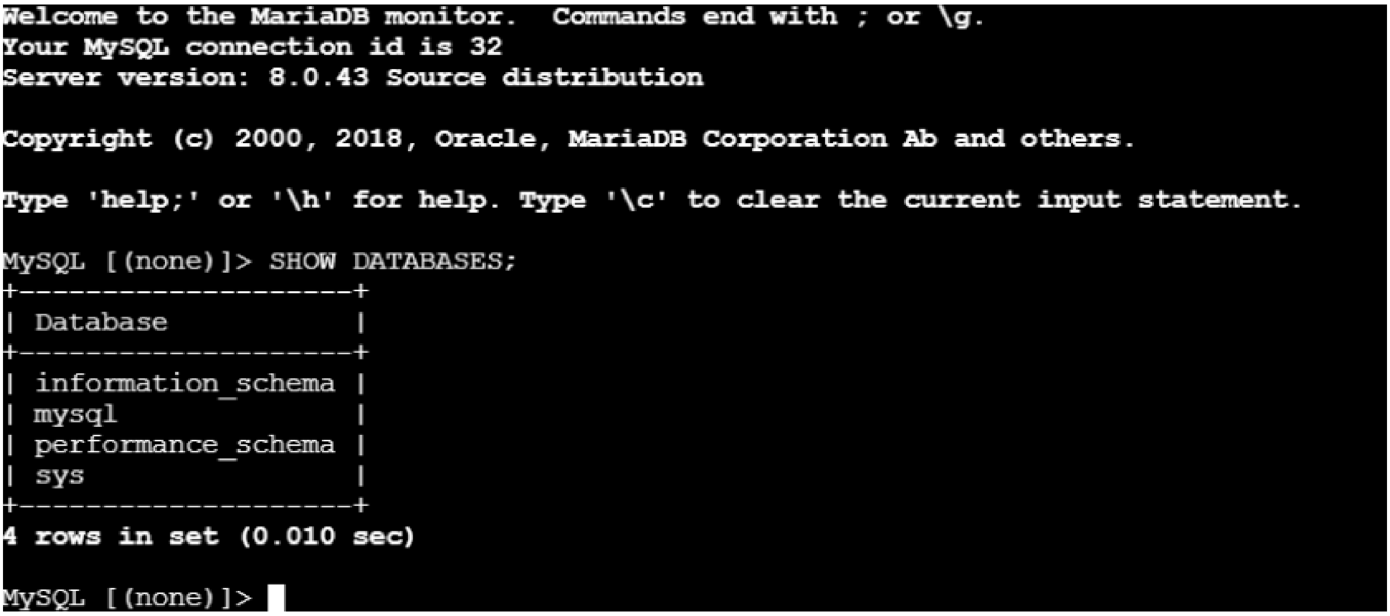
Primary network interface ID
eni-01d6f13ce5e8332d6

Deleted
–

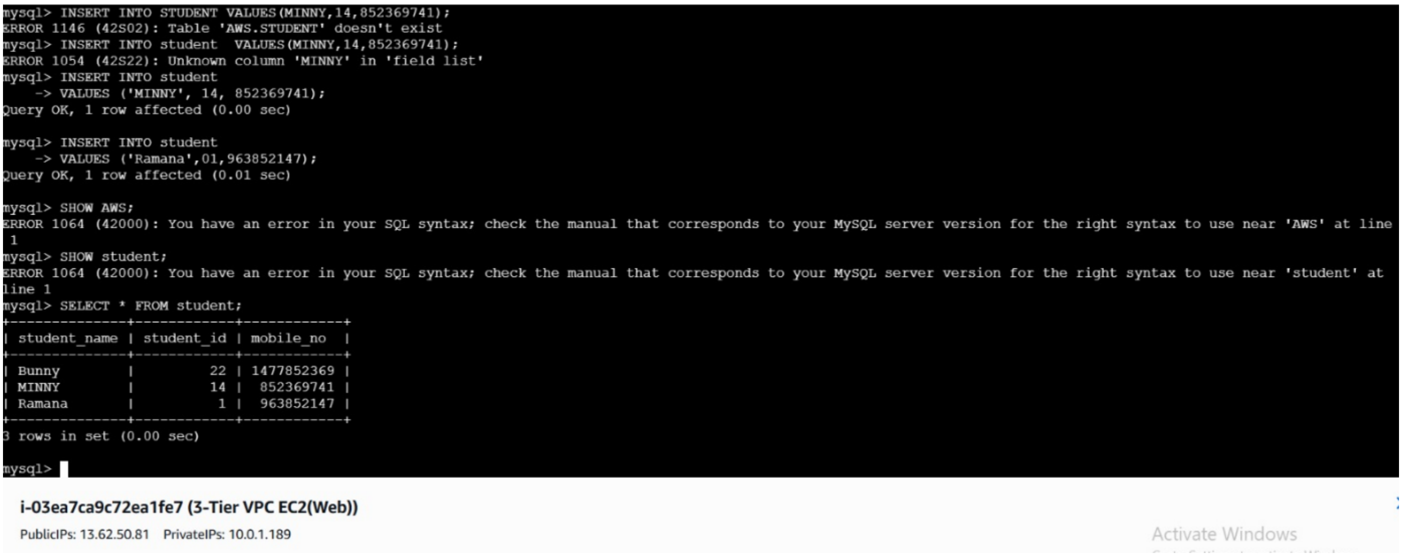
Verify End-to-End Connectivity by connecting RDS to EC2 instance:



```
mysql  Ver 15.1 Distrib 10.5.29-MariaDB, for Linux (x86_64) using  EditLine wrapper
[ec2-user@ip-10-0-1-10 ~]$ nslookup database-1.c8f4w82oas9c.us-east-1.rds.amazonaws.com
Server:         10.0.0.2
Address:        10.0.0.2#53

Non-authoritative answer:
Name:   database-1.c8f4w82oas9c.us-east-1.rds.amazonaws.com
Address: 10.0.0.201

[ec2-user@ip-10-0-1-10 ~]$ mysql -h database-1.c8f4w82oas9c.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 8.0.43 Source distribution

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

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

MySQL [(none)]>
```



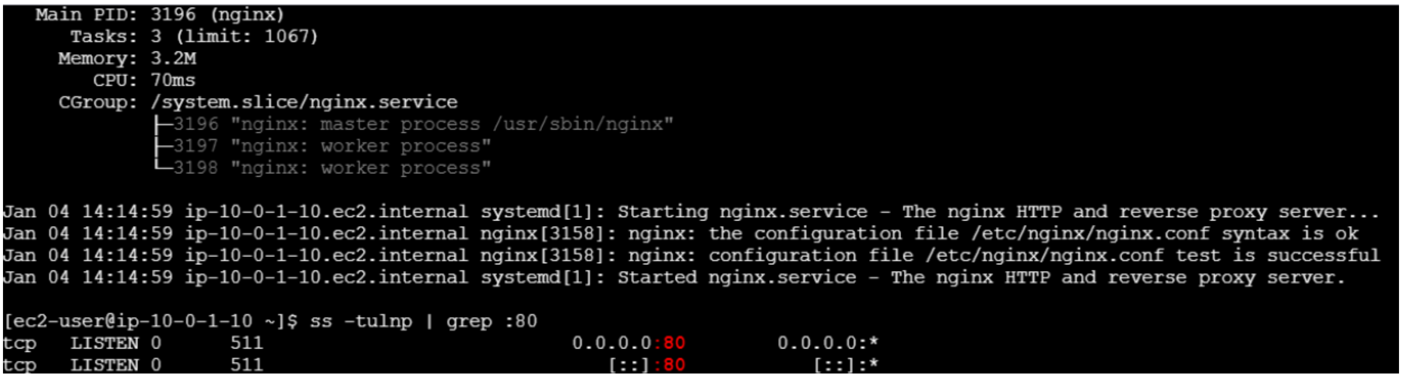
```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 8.0.43 Source distribution

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

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

MySQL [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.010 sec)

MySQL [(none)]> 
```



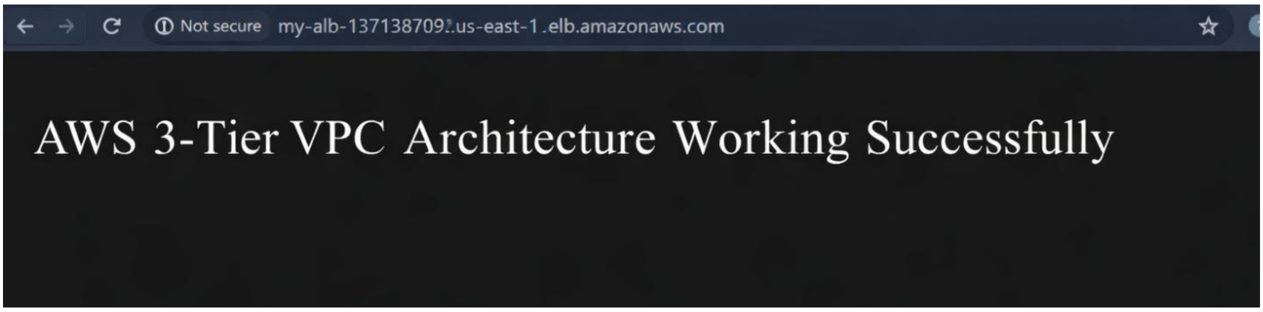
```
mysql> INSERT INTO STUDENT VALUES(MINNY,14,852369741);
ERROR 1146 (42S02): Table 'AWS.STUDENT' doesn't exist
mysql> INSERT INTO student  VALUES(MINNY,14,852369741);
ERROR 1054 (42S22): Unknown column 'MINNY' in 'field list'
mysql> INSERT INTO student
    -> VALUES ('MINNY', 14, 852369741);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student
    -> VALUES ('Ramana',01,963852147);
Query OK, 1 row affected (0.01 sec)

mysql> SHOW AWS;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'AWS' at line
 1
mysql> SHOW student;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'student' at
line 1
mysql> SELECT * FROM student;
+--------------+------------+------------+
| student_name | student_id | mobile_no  |
+--------------+------------+------------+
| Bunny        |         22 | 1477852369 |
| MINNY        |         14 |  852369741 |
| Ramana       |          1 |  963852147 |
+--------------+------------+------------+
3 rows in set (0.00 sec)

mysql> 
```

i-03ea7ca9c72ea1fe7 (3-Tier VPC EC2(Web))

PublicIPs: 13.62.50.81   PrivateIPs: 10.0.1.189



```
  Main PID: 3196 (nginx)
     Tasks: 3 (limit: 1067)
    Memory: 3.2M
       CPU: 70ms
    CGroup: /system.slice/nginx.service
             ├─3196 "nginx: master process /usr/sbin/nginx"
             ├─3197 "nginx: worker process"
             └─3198 "nginx: worker process"

Jan 04 14:14:59 ip-10-0-1-10.ec2.internal systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server...
Jan 04 14:14:59 ip-10-0-1-10.ec2.internal nginx[3158]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Jan 04 14:14:59 ip-10-0-1-10.ec2.internal nginx[3158]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Jan 04 14:14:59 ip-10-0-1-10.ec2.internal systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.

[ec2-user@ip-10-0-1-10 ~]$ ss -tulnp | grep :80
tcp   LISTEN 0      511                     0.0.0.0:80        0.0.0.0:*
tcp   LISTEN 0      511                        [::]:80           [::]:*
```



AWS 3-Tier VPC Architecture Working Successfully

High Availability and Scalability

The architecture achieves high availability through:

- Multi-AZ subnet deployment
- Application Load Balancer
- Auto Scaling Group

Scalability is achieved by:

- Horizontal scaling of EC2 instances
- Automatic health checks and replacement

Validation and Testing

The implementation was validated through:

a. Successful access of the application using ALB DNS endpoint
b. Verification of healthy target group status
c. Automatic instance replacement during termination tests
d. End-to-end request flow confirmation

Cost and Operational Awareness

To optimize costs:

a. Auto Scaling capacity is reduced when not in use
b. RDS instances are stopped during idle periods
c. NAT Gateway is deleted when not required

This demonstrates awareness of cloud cost management best practices.

Conclusion:

This project successfully demonstrates the implementation of a production-ready 3-Tier architecture on AWS using DevOps best practices. By separating the presentation, application, and database layers, the system achieves high availability, scalability, and security.

Through this project, I gained practical experience in AWS networking, load balancing, auto scaling, security group design, and database isolation. It provided real-world exposure to designing cloud infrastructure that follows industry standards and reliability principles.

This project strengthened my foundation in DevOps and cloud engineering and prepared me to work with real-world AWS environments.

By -
R. Adithi
Aspiring DevOps Engineer
(Batch - 06)