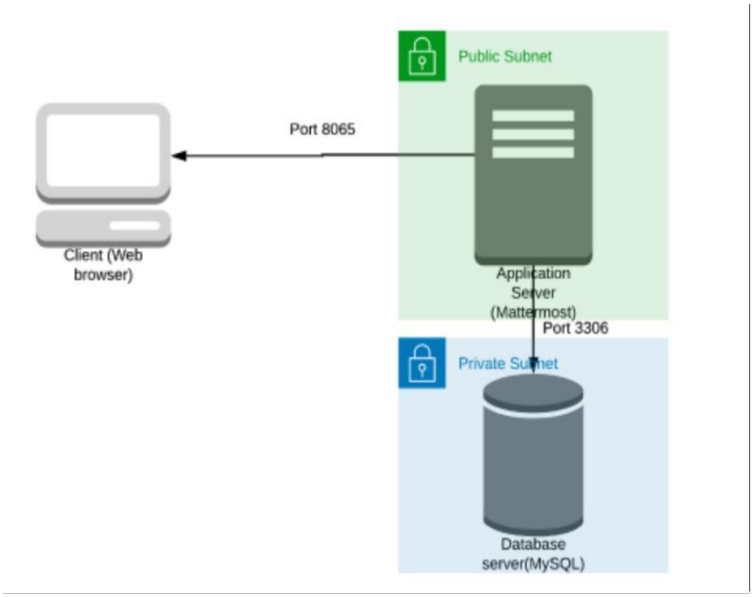
Scenario

Team communication and instant messaging solutions are an integral part of any business environment today. As of 2020, the total number of users of Slack and Microsoft Teams exceeded 20 million.

Some organizations might have compliance policies in place which do not allow them to use services managed by third parties. They will prefer solutions that can be managed and hosted on servers controlled by them. The same will extend to communication solutions as well.

Architecture diagram

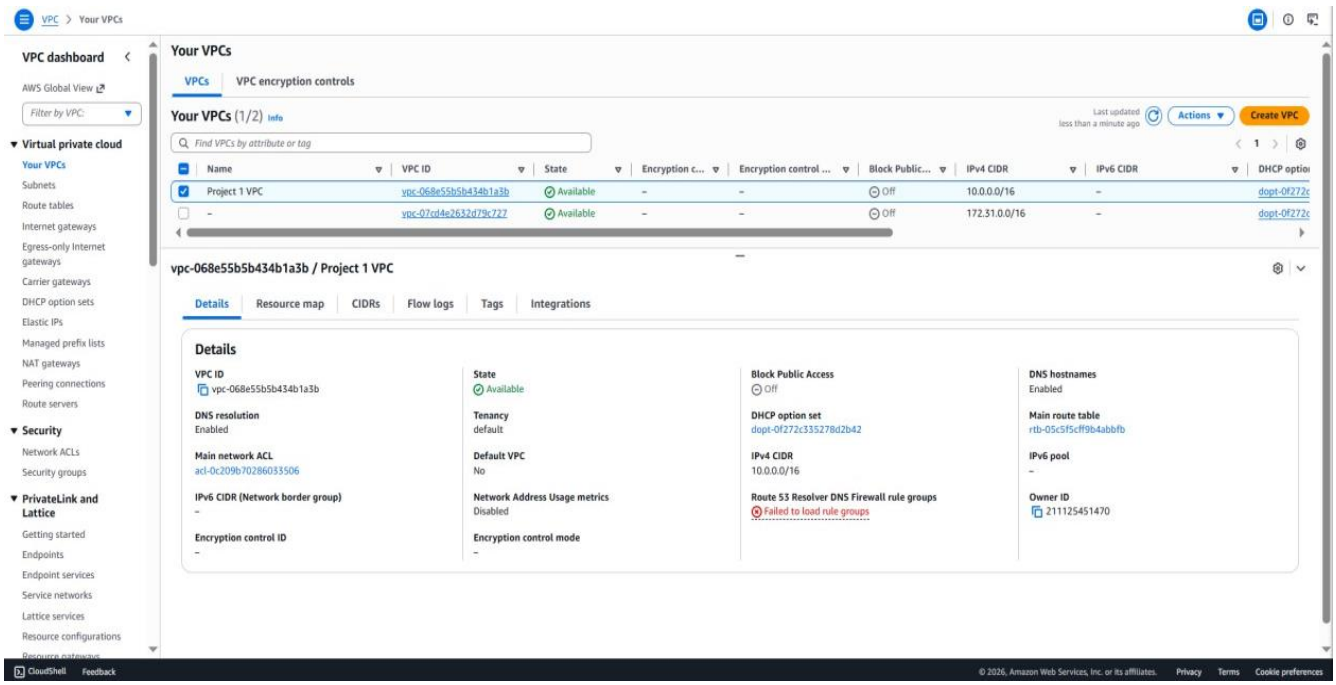


Architecture Implementation	
1	Implement 2 different subnets (one public and the other private) in a custom VPC
2	Install and configure MySQL on an Amazon Linux 2 instance on the private subnet using the instructions provided. (Hint: Use a bastion host and a NAT gateway)
3	Install and configure Mattermost on an Amazon Linux 2 instance on the public subnet using the provided instructions.
4	Configure the security groups to allow the ports as shown in the architecture.
5	Test the installation by accessing the IP of the public instance in a browser via the port 8065.

Step 1: VPC and Subnet Creation

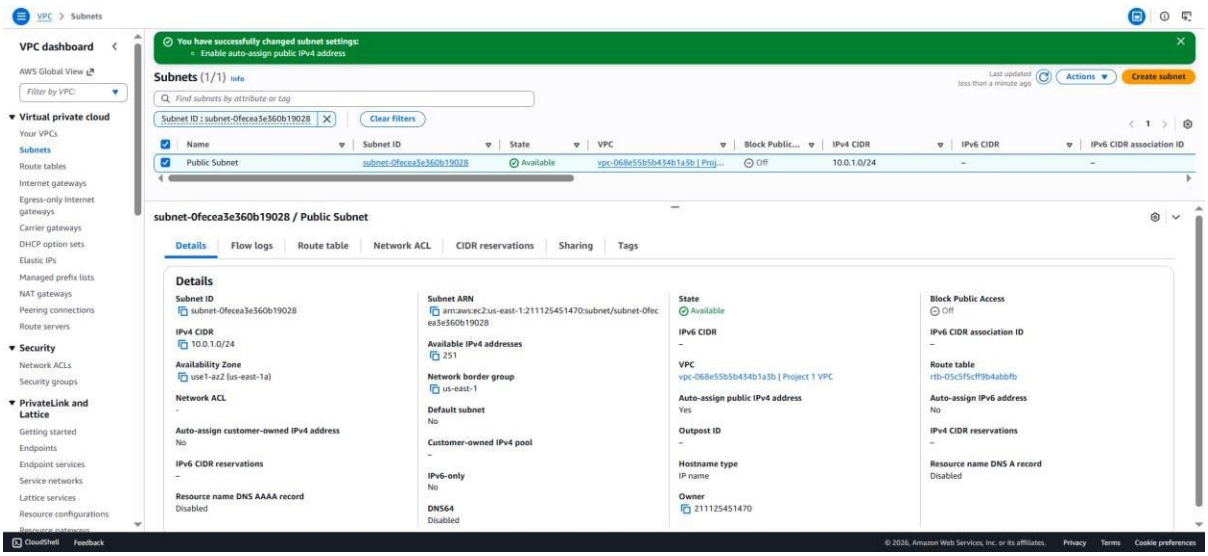
Step number	a
Step name	Creation of VPC
Instructions	1) Navigate to VPC using the Services button at the top of the screen 2) Select "Your VPCs" on the left side of the screen 3) Click on "Create VPC" 4) Enter the following fields : Name: Project 1 VPC IPv4 CIDR Block : 10.0.0.0/16 The rest of the options can be ignored 5) Select "Create VPC" 6) Select the VPC and click on Actions->Edit DNS hostnames 7) Enable DNS hostnames and click on Save
Expected screenshots	Created VPC with properties visible

<Insert Screenshot a(1) here>



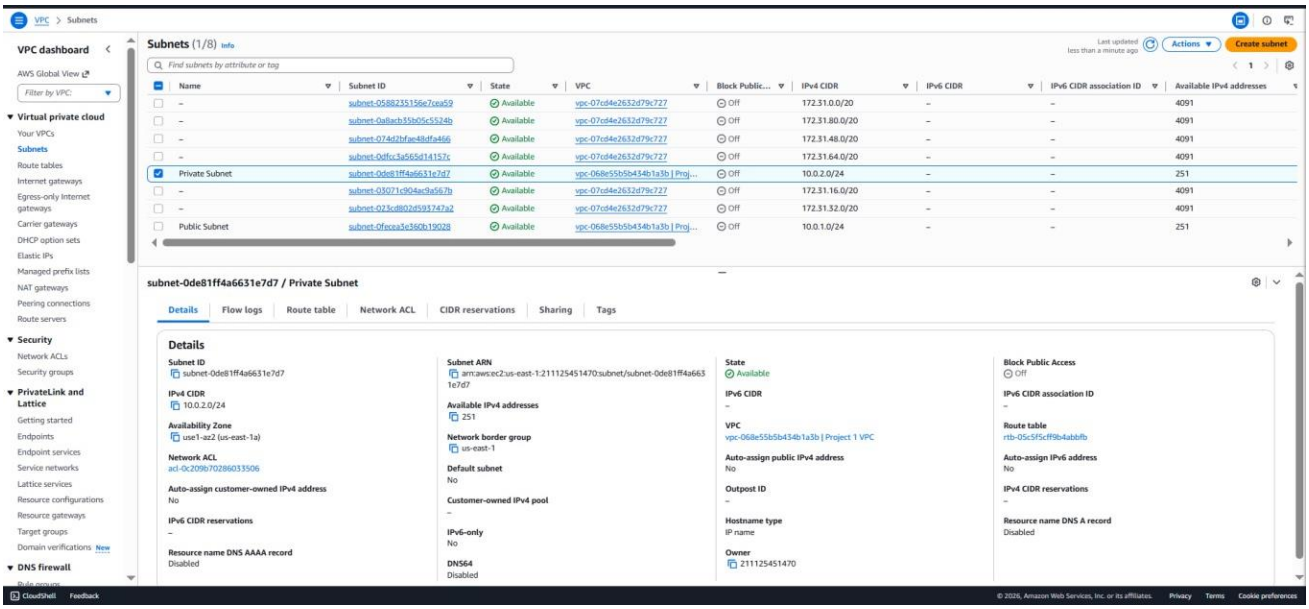
Step number	b
Step name	Creation of public subnet
Instructions	1) Navigate to VPC->Subnets 2) Click on "Create Subnet" 3) Enter the following fields Name tag : Public Subnet VPC : Select the Project 1 VPC IPv4 CIDR block : 10.0.1.0/24 The other options can be ignored 4) Click on Create 5) Once the subnet has been created, select the subnet and click on Actions->Modify Auto-assign IP settings 6) Enable the option "Auto assign IPv4" and select Save
Expected screenshots	Subnet Creation screen

<Insert Screenshot b(1) here>



Step number	c
Step name	Creation of private subnet
Instructions	1) Navigate to VPC->Subnets 2) Click on "Create Subnet" 3) Enter the following fields Name tag : Private Subnet VPC : Select the Project 1 VPC IPv4 CIDR block : 10.0.2.0/24 The other options can be ignored 4) Click on Create
Expected screenshots	Subnet Creation screen

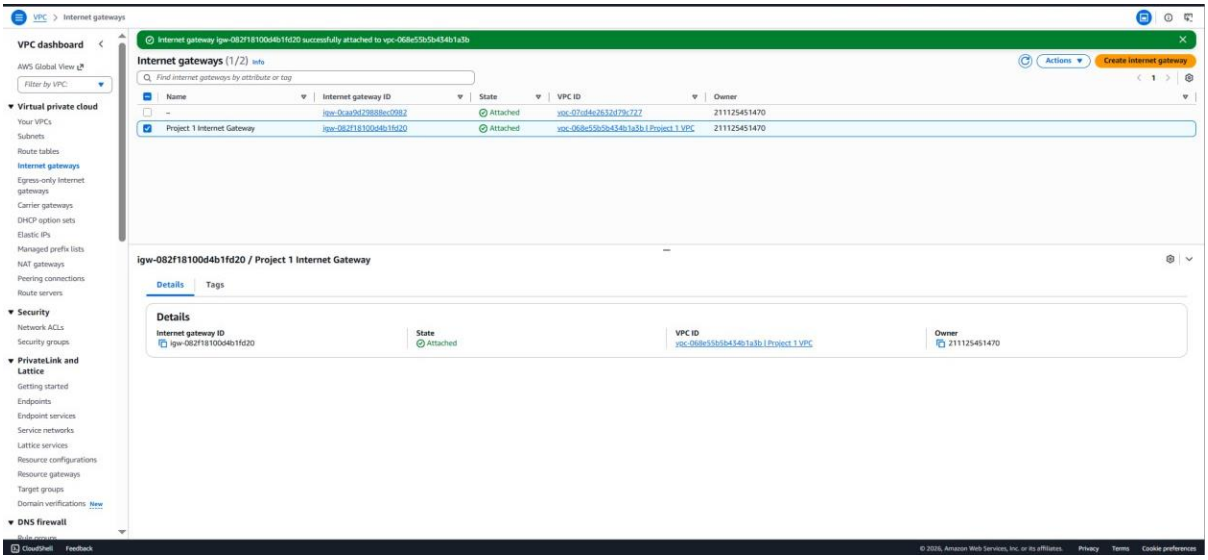
<Insert Screenshot c(1) here>



Step 2 : Internet Gateway and VPC

Step number	a
Step name	Creation and Configuration of Internet Gateway
Instructions	1) Navigate to VPCs->Internet Gateway 2) Click on "Create Internet Gateway" 3) Enter the name tag "Project 1 Internet Gateway" and click on "Create Internet Gateway" 4) After the gateway is created, select it and click on Actions->Attach to VPC 5) Select the Project 1 VPC and click on "Attach Internet Gateway"
Expected screenshots	Creation of Internet Gateway

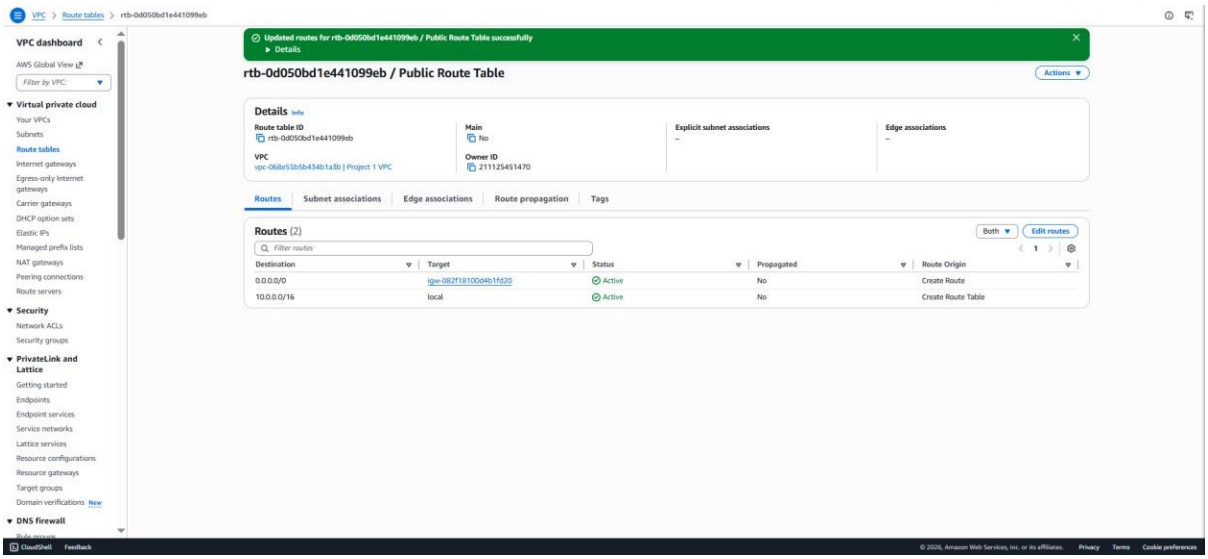
<Insert Screenshot a(1) here >



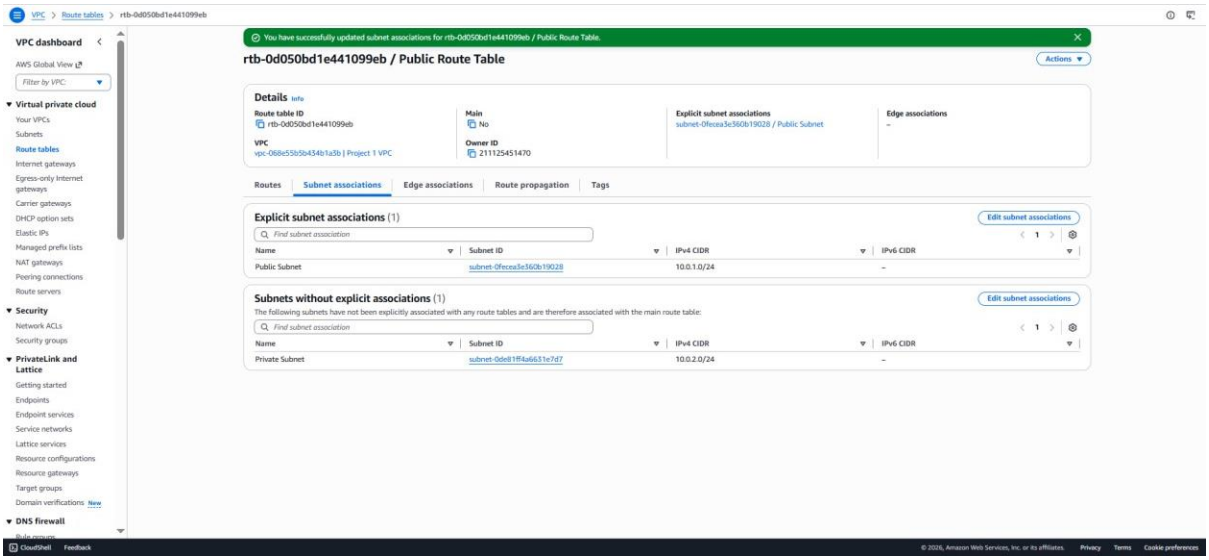
Step number	b
Step name	Creation of public route table
Instructions	1) Navigate to VPC -> Route Tables and click on Create Route table 2) Enter the name tag "Public Route Table", select the Project 1 VPC from the dropdown and click on Create 3) Once the route table is created, select it and select the Routes tab below the list of route tables 4) Click in Edit Routes and add the following route (Don't edit the existing one) - Destination : 0.0.0.0/0 - Target : Select Internet Gateway and the select the Project 1 Internet Gateway Click on Save Routes 5) Select the Subnet Associations tab and click on Edit Subnet Associations 6) Select the Public Subnet from the list and click on Save

Expected screenshots	1) Route list of the route table 2) Subnet Associations of the route table

<Insert Screenshot b(1) here>

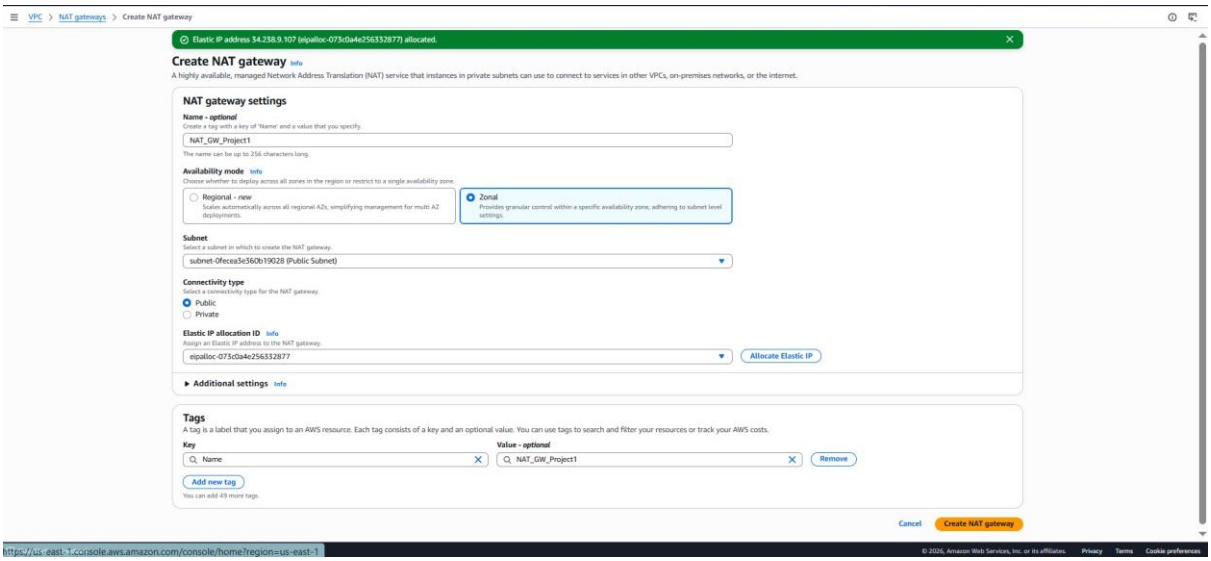


<Insert Screenshot b(2) here>

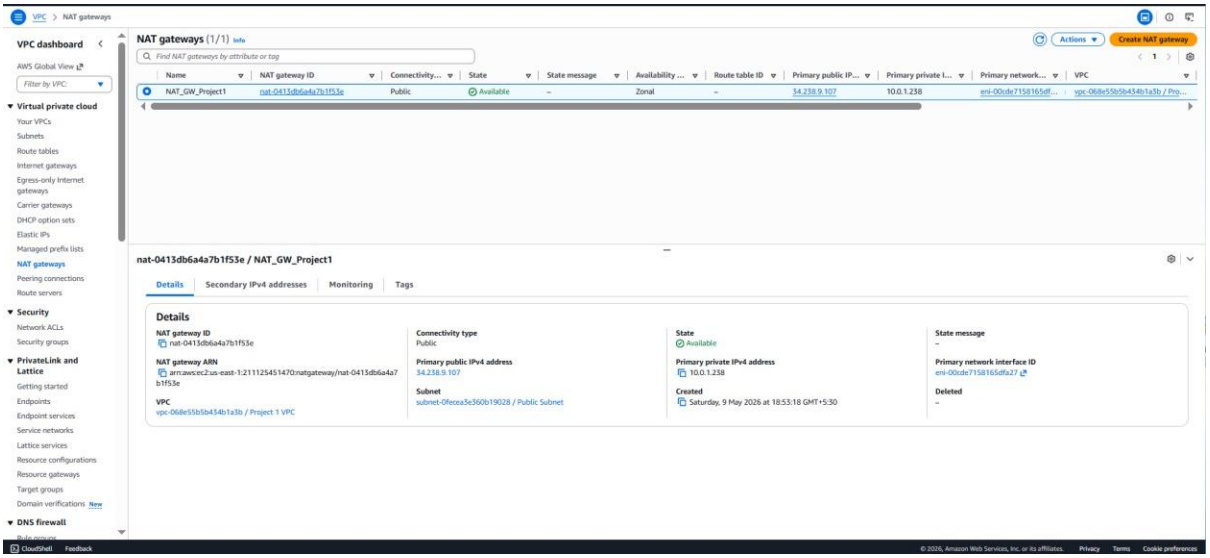


Step number	c
Step name	Creation of NAT gateway
Instructions	1) Navigate to VPC using the Services button at the top of the screen 2) Select NAT Gateway at the left side of the screen 3) Click on Create NAT Gateway - Deploy it in the public subnet - Connectivity type: Public - Allocate an elastic IP by clicking on “Allocate Elastic IP” 4) Click on “Create NAT Gateway” to create the gateway
Expected screenshots	1) NAT gateway creation details 2) Gateway after creation

<Insert Screenshot c(1) here>

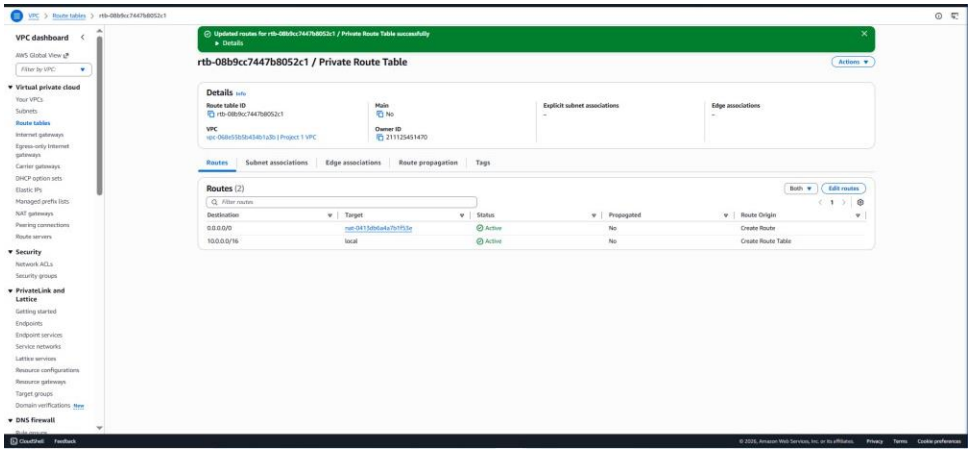


<Insert Screenshot c(2) here>

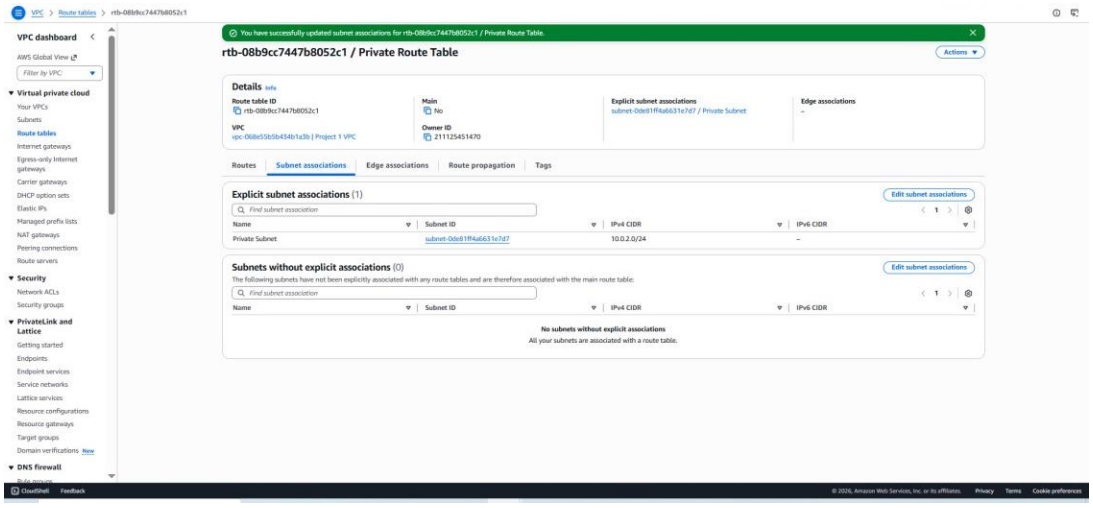


Step number	d
Step name	Creation of private route tables
Instructions	1) Navigate to VPC -> Route Tables and click on Create Route table 2) Enter the name tag "Private Route Table", select the Project 1 VPC from the dropdown and click on Create 3) Once the route table is created, select it and select the Routes tab below the list of route tables 4) Click in Edit Routes and add the following route (Don't edit the existing one) - Destination : 0.0.0.0/0 - Target: Select NAT Gateway and select the NAT Gateway created in the previous step Click on Save Routes 5) Select the Subnet Associations tab and click on Edit Subnet Associations 6) Select the private Subnet from the list and click on Save
Expected screenshots	1) Route list of the route table 2) Subnet association of the route table

<Insert Screenshot for d(1) here >



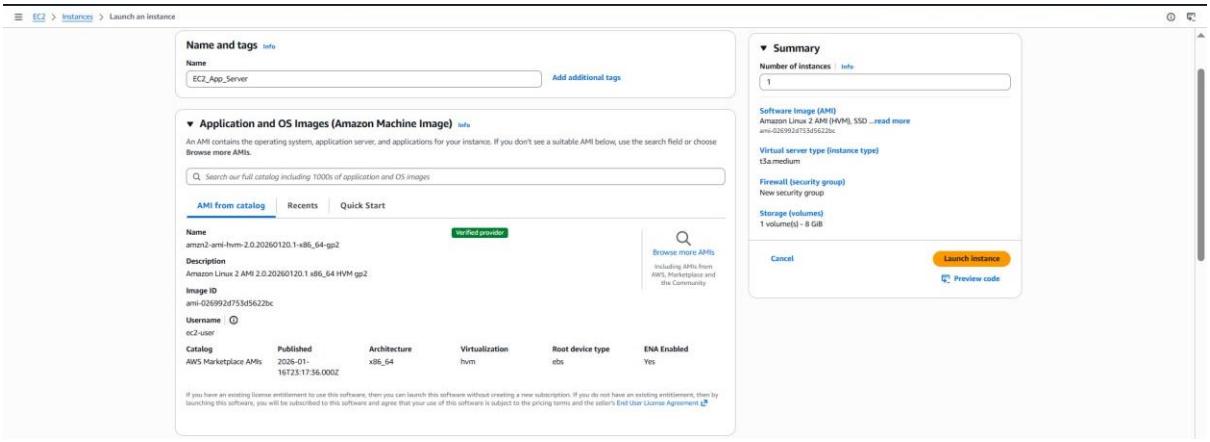
<Insert Screenshot for d(2) here>



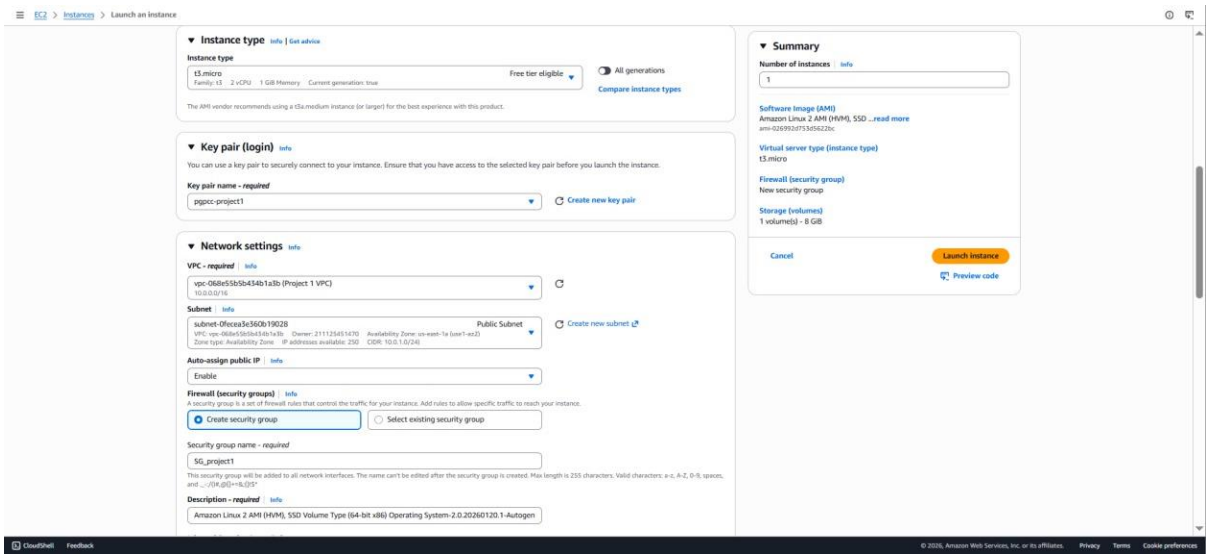
Step 3 : Creation of database and application servers

Step number	a
Step name	Creation of application server
Instructions	1) Navigate to EC2 using the Services button at the top of the screen 2) Select Instances at the left side of the screen 3) Click on Launch Instance - Select the AMI Amazon Linux 2 To find the above AMI, follow the below steps - During AMI selection, click on “Browse more AMIs” - Select the Amazon Marketplace tab - Search for Amazon Linux 2 in the search bar - Select the result – Amazon Linux 2 AMI (HVM) - Click on Subscribe Now - Select the instance type t3.micro - Select Network as "Project 1 VPC" and subnet as "Public Subnet" - For the security group, open the ports 80,443, 22 and 8065 for source set to "Anywhere" 4) Launch the instance after creating a new pem file and downloading it
Expected screenshots	1) AMI used 2) Instance configuration screen 3) Security group rules 4) Instance after creation

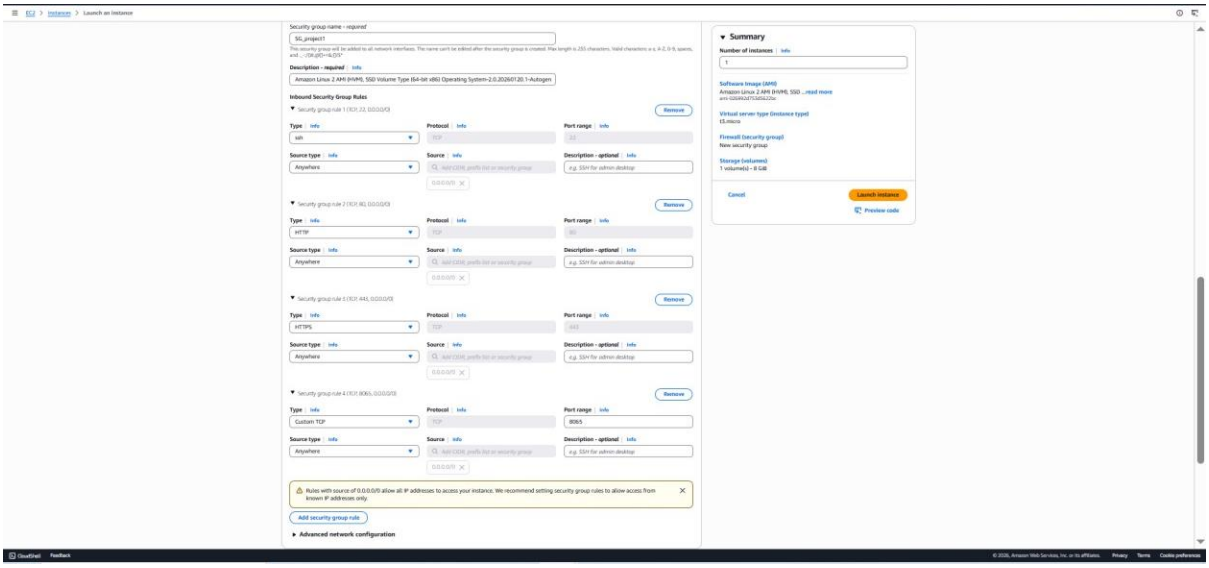
<Insert screenshot a(1) here>



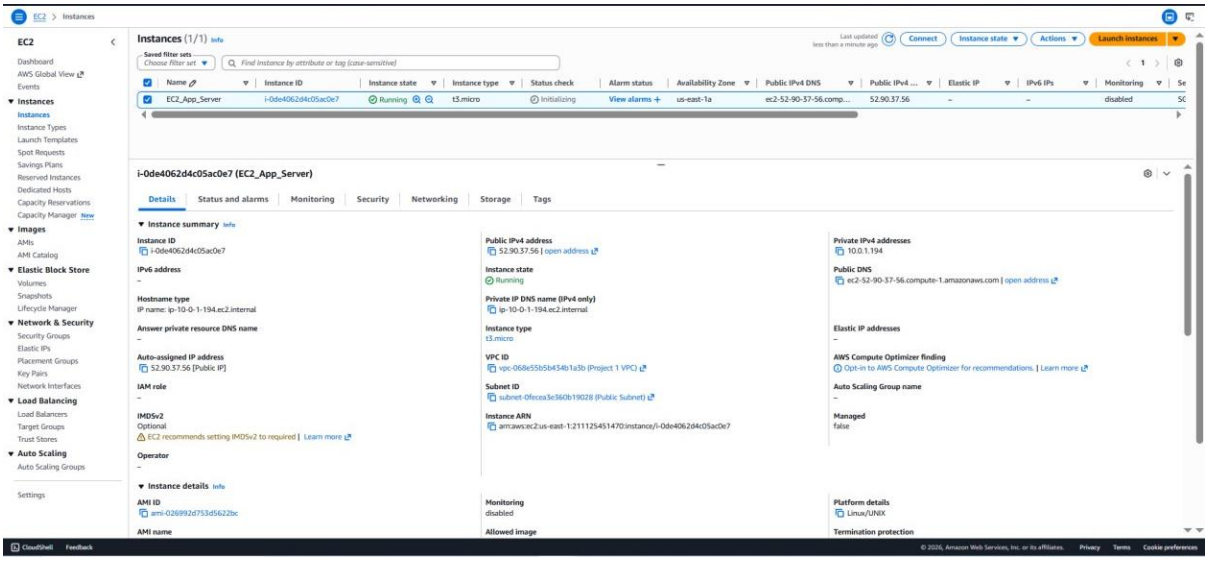
<Insert screenshot a(2) here>



<Insert screenshot a(3) here>

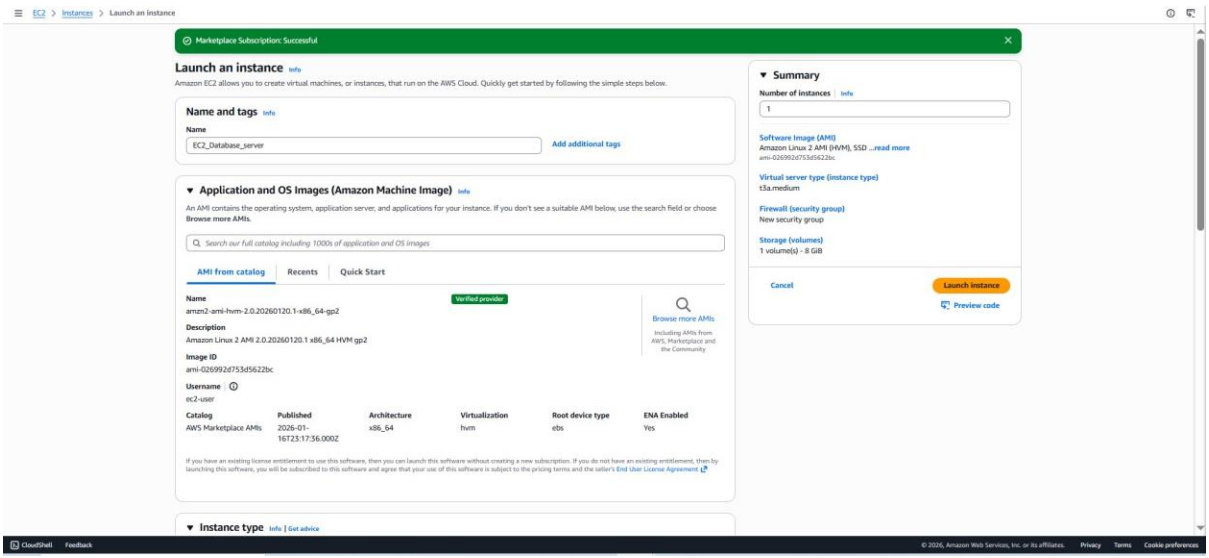


<Insert screenshot a(4) here>

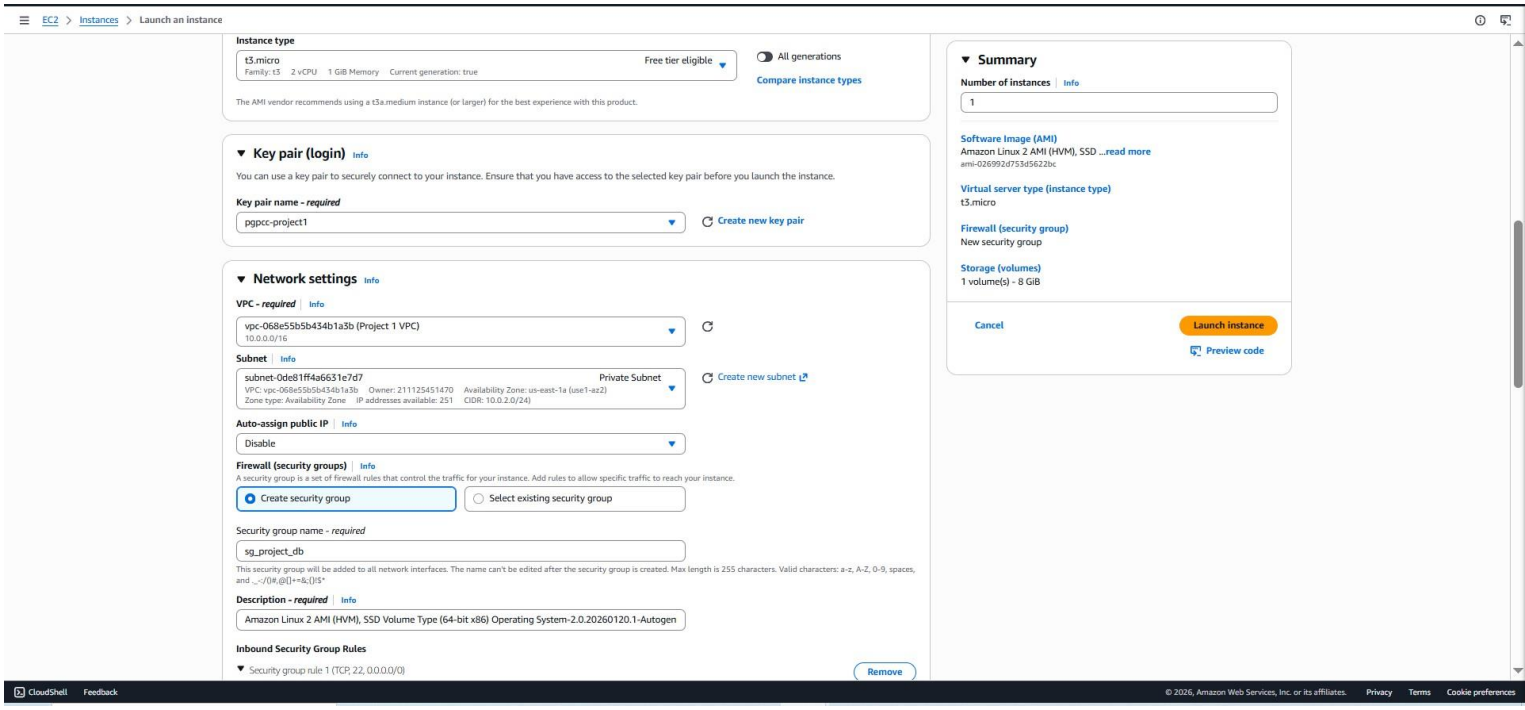


Step number	b
Step name	Creation of database server
Instructions	1) Navigate to EC2 using the Services button at the top of the screen 2) Select Instances at the left side of the screen 3) Click on Launch Instance - Select the AMI Amazon Linux 2 (Follow the same steps as the application server to find the AMI) - Select the instance type t3.micro - Select Network as "Project 1 VPC" and subnet as "Private Subnet" - For the security group, open the ports 80, 443,22 and 3306 for source set to "Anywhere" 4) Launch the instance by selecting the same pem file created in the previous step
Expected screenshots	1) AMI used 2) Instance configuration screen 3) Security group rules 4) Instance after creation

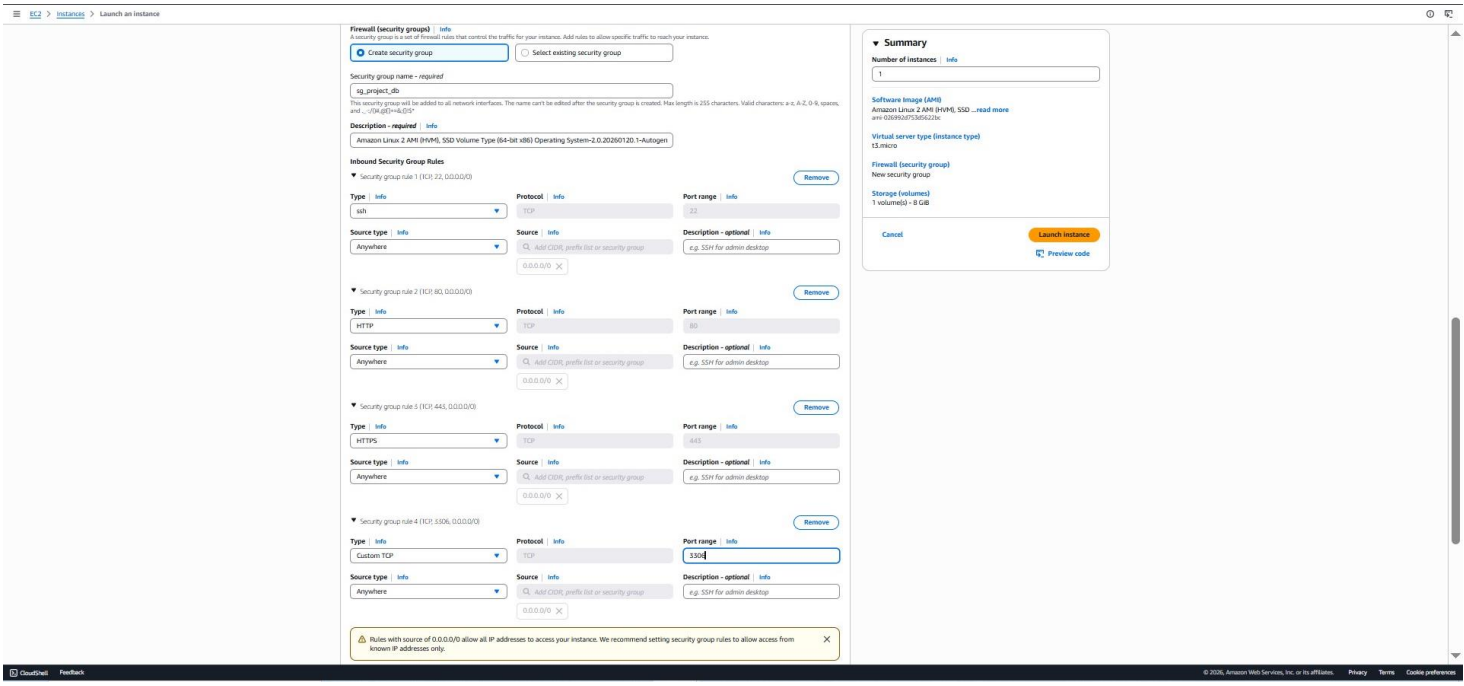
<Insert screenshot b(1) here>



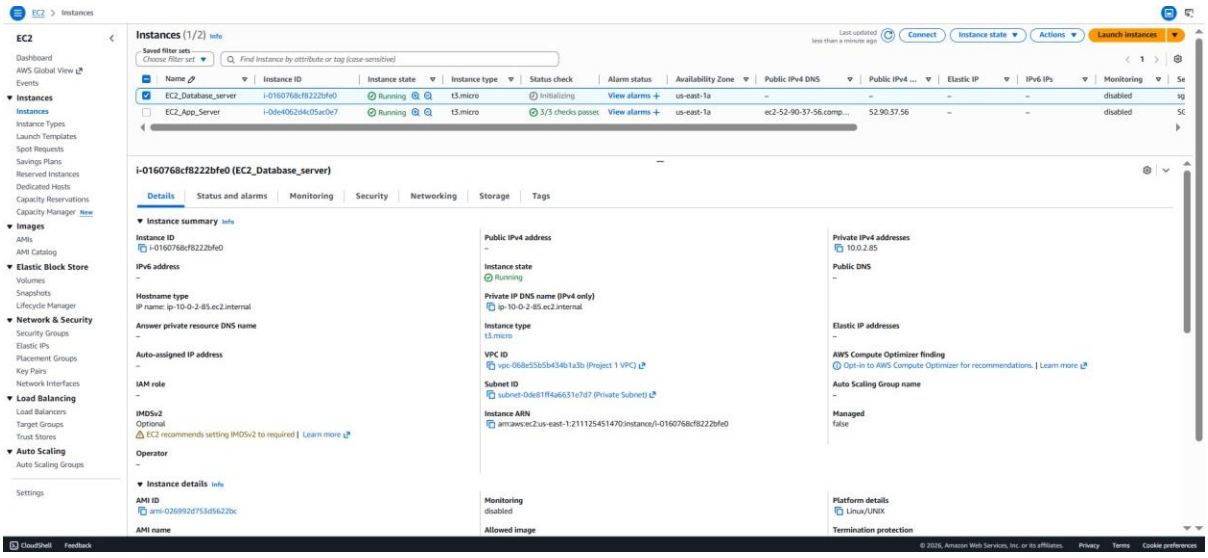
<Insert screenshot b(2) here>



<Insert screenshot b(3) here>.



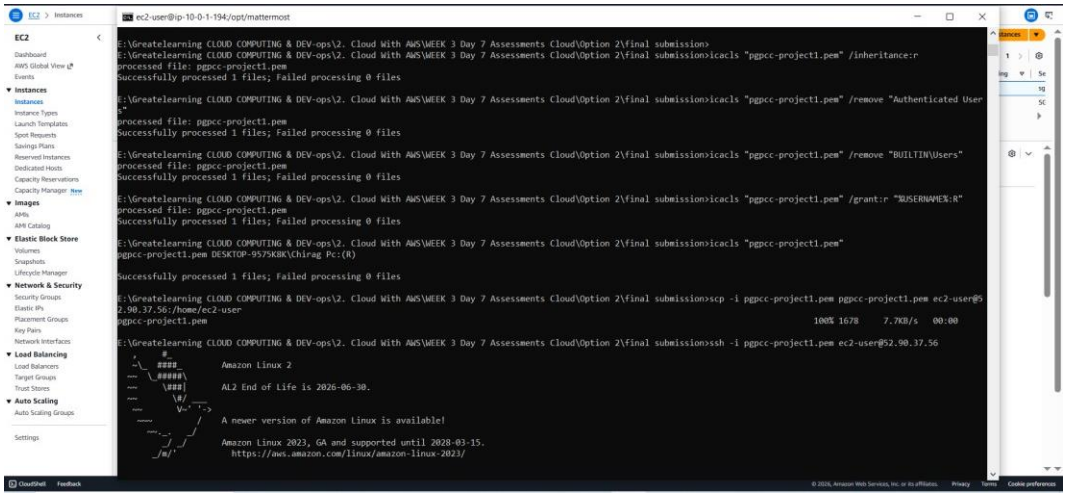
<Insert screenshot b(4) here>

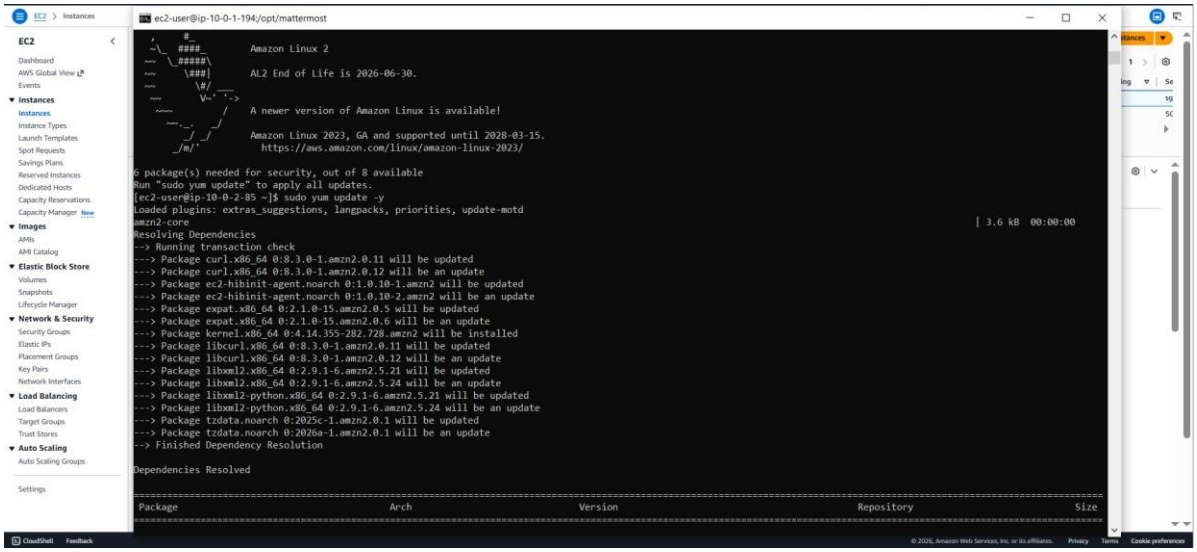


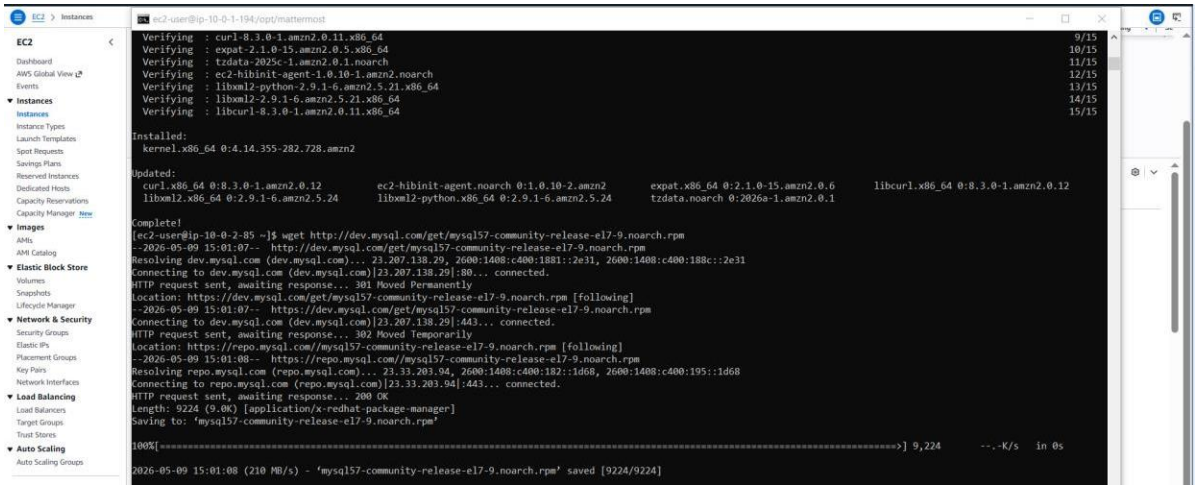
Step 4: Application and Database Installation and Testing

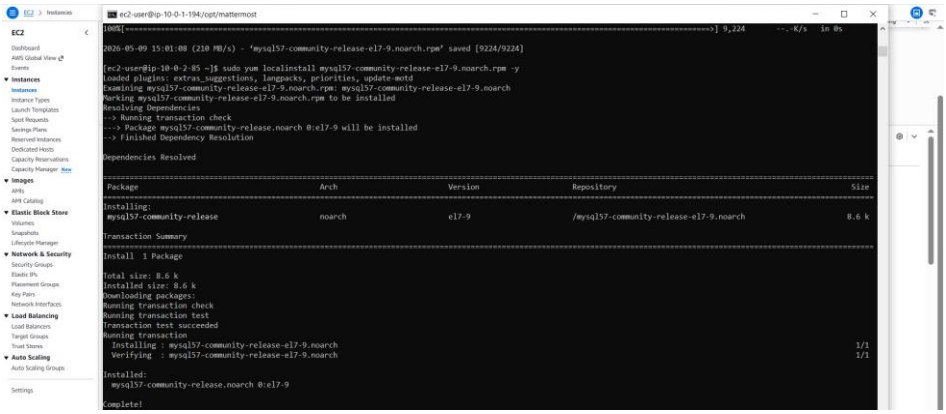
Step number	a
Step name	Installation and configuration of MySQL
Instructions	1) Copy the database pem file into the application server using the below command *scp -i YOUR_APP.pem YOUR_DB.pem ec2-user@YOUR_APP_PUBLIC_IP:/home/ec2-user* 2) Log into the application server using SSH/Putty 3) From the application server, log into the database server using the pem file copied in step 1 and the private IP address of the database server with the following command *ssh -i YOUR_DB.pem ec2-user@YOUR_DB_PRIVATE_IP* 4) Enter the following commands to install and configure MySQL on the database server *sudo yum update* *wget http://dev.mysql.com/get/mysql57-community-release-el7-9.noarch.rpm* *sudo yum localinstall mysql57-community-release-el7-9.noarch.rpm -y* *sudo yum install mysql-community-server -y --nogpgcheck* *sudo systemctl start mysqld.service* Run the below command to retrieve a temporary password for MySQL *TEMP_PWD=$(sudo grep 'temporary password' /var/log/mysqld.log | awk '{printf "%s", $11}')* Log in to MySQL with the below command *mysql -u root --password=$TEMP_PWD* Enter the below command after you login to MySQL. Do not change the password set in the below command. *ALTER USER 'root'@'localhost' IDENTIFIED BY 'Password42!';* Type ‘exit’ into the MySQL prompt and press Enter to exit out of the MySQL environment. Enter the below commands to complete the setup. Ignore any warning messages you receive. *wget https://d6opu47qoi4ee.cloudfront.net/install_mysql_linux.sh* *chmod 777 install_mysql_linux.sh* *sudo ./install_mysql_linux.sh* 5) Type *exit* to exit the database server and go back to the application server
Expected screenshots	1) Installation of MySQL 2) Retrieving the temporary password 3) Executing the provided script

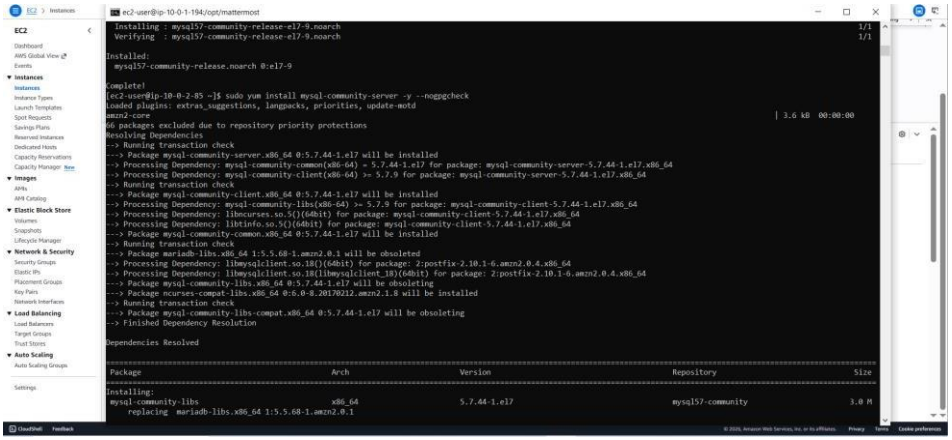
<Insert screenshot a(1) here>



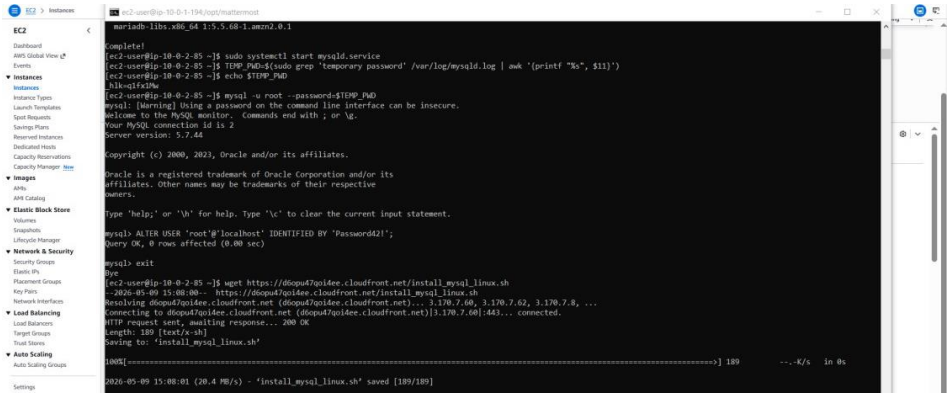


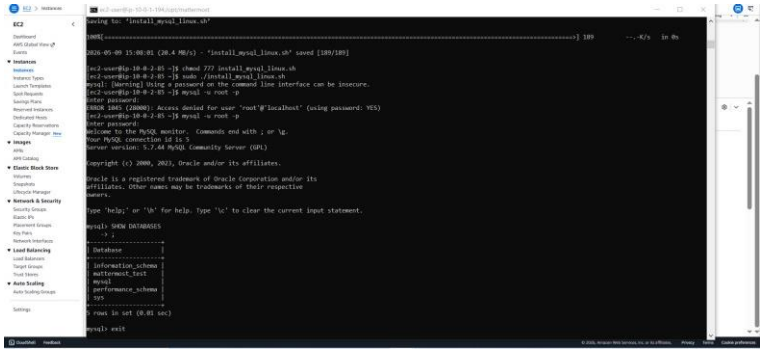




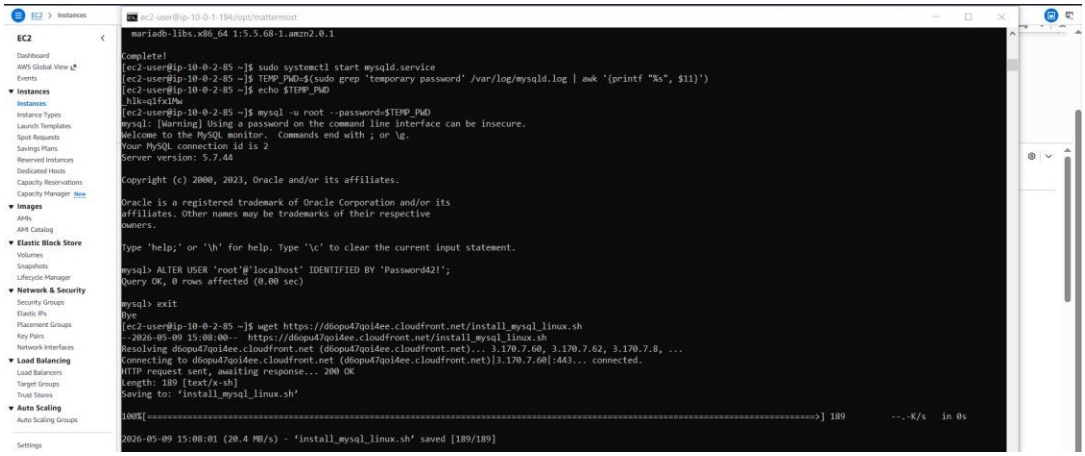


<Insert screenshot a(2) here> and Verifying Database



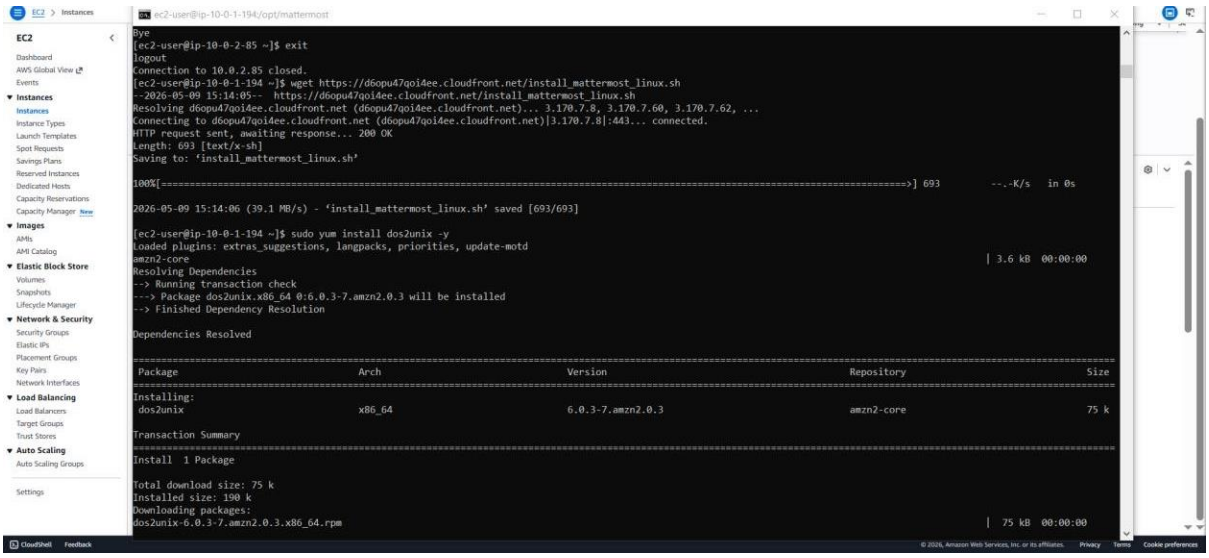


<Insert screenshot a(3) here>

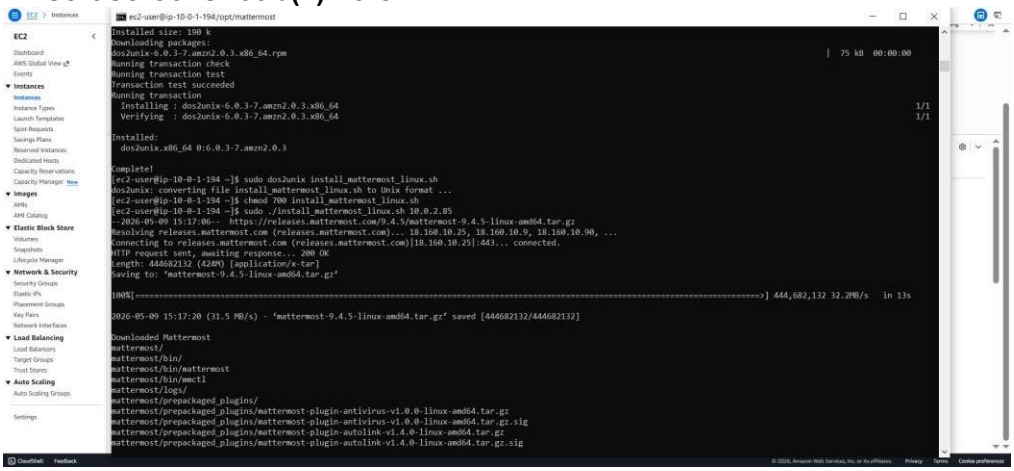


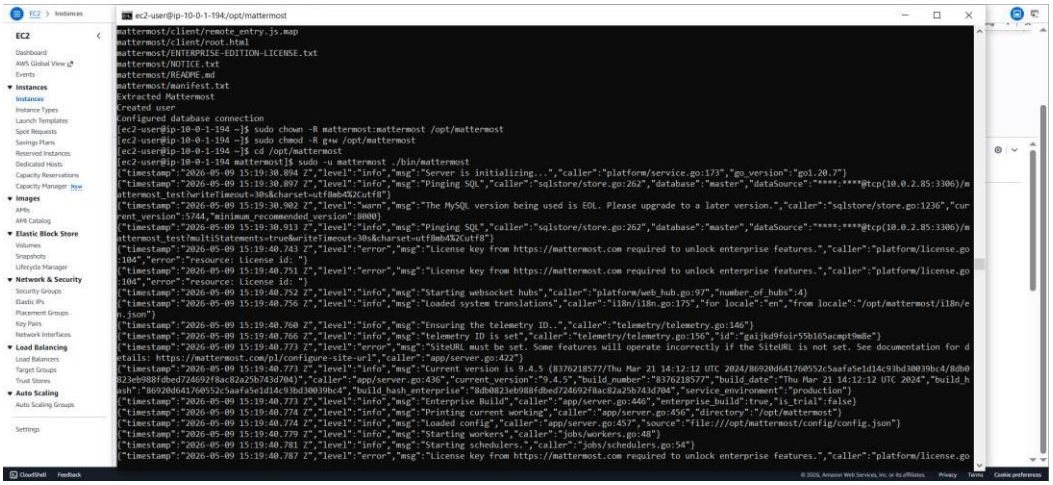
Step number	b
Step name	Installation and configuration of Mattermost
Instructions	1) Enter the following commands after logging into the application server via SSH to install and configure Mattermost. *wget https://d6opu47qoi4ee.cloudfront.net/install_mattermost_linux.sh* *sudo yum install dos2unix -y* *sudo dos2unix install_mattermost_linux.sh* *chmod 700 install_mattermost_linux.sh* *sudo ./install_mattermost_linux.sh YOUR_DB_PRIVATE_IP* *sudo chown -R mattermost:mattermost /opt/mattermost* *sudo chmod -R g+w /opt/mattermost* *cd /opt/mattermost* *sudo -u mattermost ./bin/mattermost* 2) Check whether the server has been successfully deployed by navigating to the following URL in your web browser. The web page might take a couple of minutes to load. http://YOUR_APP_PUBLIC_IP:8065
Expected screenshots	1) Executing the script 2) Starting the Mattermost server 3) Accessing the application via web browser

<Insert screenshot b(1) here>

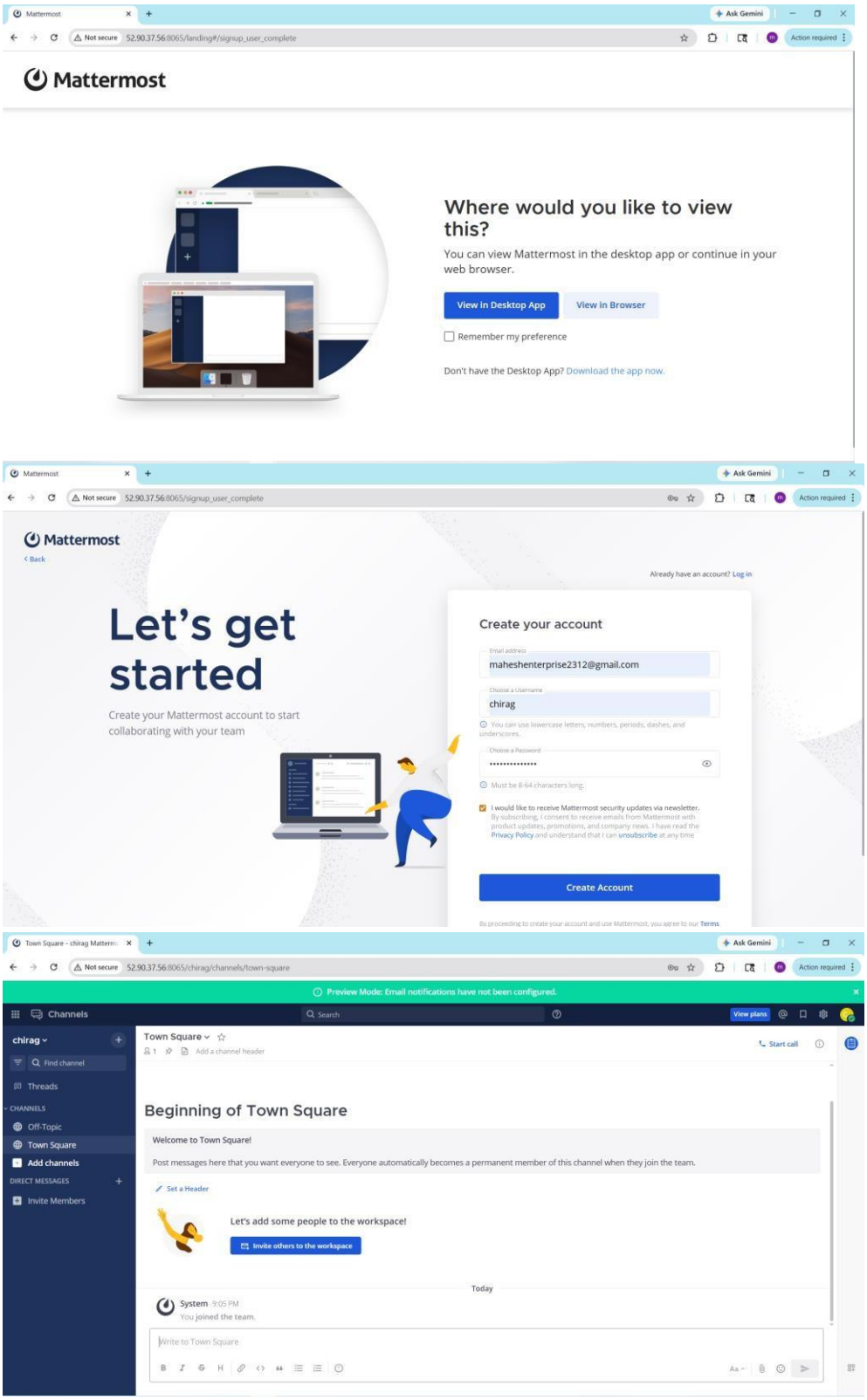


<Insert screenshot b(2) here>





<Insert screenshot b(3) here>



Step 5:
Answer the following questions

Q1 What is the default setting for DNS hostnames when a new VPC is created?

a) Enabled

b) Disabled

c) Can be set during VPC creation

d) Depends on the region used

Enter your answer here: Disabled

Q2 What is the term used for the machine when we use it to log into the database server?

a) Bastion Host

b) NAT Gateway

c) Tunnel Interface

d) SSH Gateway

Enter your answer here: Bastion Host

Q3 The database server security group in this exercise has to keep port 3306 open. Which protocol uses this port to communicate?

a) HTTPS

b) RDP

c) TCP

d) SCP

Enter your answer here: TCP

Q4 Which port is being used by Mattermost to communicate with the client application

a) 8080

b) 80

c) 443

d) 8065

Enter your answer here: 8065

Q5 Which of the following is a reason why we cannot set the CIDR block for the public subnet to 10.0.2.0/16, assuming the values for the other CIDR blocks are the same as mentioned in the instructions?

a) CIDR block overlaps with existing block

b) CIDR block is not a valid CIDR

c) CIDR block does not fall within the VPC

d) There is no reason, this is a perfectly valid CIDR

Enter your answer here | a) CIDR block overlaps with existing block

Q6 Assume that you have been asked to create 3 EC2 instances - application server, the database server and NAT instance. Each of these instances have their own security groups with a set of ports to be kept open. One of these ports is entirely unnecessary for the given architecture to function. Which of the following could it be?

a) Port 22 on the NAT instances

b) Port 3306 on the database server

c) Port 3306 on the application server

d) Port 22 on the application server

Enter your answer here | c) Port 3306 on the application server

Q7 How are we going to increase the security of the Mattermost server to ensure the users are from a specific organization and the traffic is originating from a known IP address?

Complete Security Hardening for Mattermost Server

To ensure that **only users from a specific organization** can access the Mattermost server and that **traffic originates only from a known IP address**, you must implement a combination of network-level and application-level controls.

1. Restrict Access by IP Using Security Groups (AWS) or Firewall Rules

- **Security Groups (if hosted on AWS)**:
  - Create a security group for your Mattermost EC2 instance.
  - Add an **inbound rule** for HTTPS (port 443) and/or HTTP (port 80) with a **source of your organization's known public IP address or CIDR block** (e.g., 203.0.113.0/24).
  - Remove any rule that uses 0.0.0.0/0 (open to the world).
  - Example inbound rule:
    - Type: HTTPS, Protocol: TCP, Port: 443, Source: your-org-IP-range
- **On-premises firewall / Network ACLs**:
  - Similarly, configure your firewall or AWS Network ACLs to **allow only the known IP range** and deny all others.
- **VPN Access**:
  - Require all users to connect to your corporate VPN before accessing Mattermost.
  - The VPN server's public IP becomes the "known IP address".
  - Then configure security groups to allow traffic **only from the VPN server's private IP range** (or the VPN's public egress IP).
  - This ensures that even external users must first authenticate to the VPN, adding another layer.

2. Enforce Organization-Based Login

a) Restrict by Email Domain (Simplest Method)

- In Mattermost System Console → **Users** → **User Filter** (or similar depending on version).
- Set **Allowed email domains** to your organization's domain(s), e.g., @mycompany.com.
- When a user tries to sign up or log in with SSO, Mattermost will reject any email address not matching the allowed domain.

b) Use SAML or LDAP with Group Filters (More Robust)

- Configure Mattermost to authenticate against your organization's identity provider (Microsoft Entra ID, Google Workspace, Okta, etc.).
- Use **LDAP user filters** or **SAML login filters** to permit only members of specific Active Directory groups or those carrying a specific organization attribute.

c) Multi-Factor Authentication (MFA)

- **At the application level**: Enable MFA in Mattermost (requires email or an authenticator app).
- **At the identity provider level**: If using SAML/SSO, enforce MFA on the IdP side (e.g., Microsoft Entra ID Conditional Access).
- MFA ensures that even if a password is compromised, the attacker cannot log in without the second factor.

3. Secure Communication with HTTPS and VPN

- **HTTPS is mandatory**:

- Obtain a TLS certific ate (e.g., from Let's Encrypt, or a commercial CA).
- Configure Mattermo st to use HTTPS by setting up a reverse proxy (NGINX, Apache, Caddy) or using Mattermost's built-in TLS.
- Redirect all HTTP tr affic to HTTPS.
- **Why**: Encrypts all d ata in transit, preventing eavesdropping, session hijacking, and man-in-the-mid dle attacks.

- **VPN as an additional encr yption layer**:
  - If users connect via VPN before reaching Mattermost, the entire VPN tunnel is encrypted.
  - This adds defense- n-depth: even if HTTPS were somehow compromised, the VPN protects the tr affic.

4. Putting It All Together (Recom nended Architecture)

For a **public cloud deployment (AW S)**:

1. **Create a Security Group** fo r the Mattermost instance.
   - Inbound rule: HTTP S – source = your organization's public IP range **OR** the private IP range of y ur VPN server.
2. **Launch Mattermost** behind an internal load balancer or with a public IP but restricted security group.
3. **Configure a reverse proxy** (NGINX) on the same instance to terminate HTTPS and optionally enforce IP allowlis ting again.
4. **Set allowed email domain s** in Mattermost System Console.
5. **Enable MFA** for all users.
6. **Require VPN** for any extern al access:
   - Users connect to co rporate VPN (e.g., AWS Client VPN, OpenVPN, WireGuard).
   - The VPN server for wards traffic to Mattermost only after successful VPN authentication.

Why This Combination Is Effectiv e

Threat	Mitigation
Attackers from unknown IPs	Security group allows only known IPs / VPN range
Password theft	MFA and organization-only login (email domain or SSO)
Eavesdropping / tampering	HTTPS + optional VPN encryption
Unauthorized users from a trusted IP	Organization-based login (email domain filter) blocks them
VPN credentials alone without MFA	Mattermost MFA adds another layer

No single control is perfect, but **toge ther they provide strong defense in depth**.

Q8 How do we achieve elasticity for the Mattermost server?

A. Use Auto Scaling Group (ASG)

Create an **Auto Scaling Group** for your Mattermost application servers.

- Define a **Launch Template** (with Mattermost pre-installed or using a startup script)
- Set:
  - **Minimum instances** (e.g., 2 for high availability)
  - **Desired capacity**
  - **Maximum instances** (e.g., 5 or more)

This ensures:

- New EC2 instances launch automatically when load increases
- Instances terminate when load drops (cost-efficient)

B. Attach Application Load Balancer (ALB)

Place an **Application Load Balancer** in front of your instances.

- It distributes incoming traffic across all running instances
- Performs **health checks** (unhealthy instances are replaced automatically)
- Works seamlessly with ASG

Without a load balancer, elasticity won't work properly because traffic won't be distributed.

C. Configure Scaling Policies

Set **dynamic scaling policies** based on metrics like:

- CPU Utilization (e.g., scale out if CPU > 70%)
- Network traffic
- Request count per target (better for web apps like Mattermost)

Example:

- Scale OUT: +1 instance if CPU > 70%
- Scale IN: -1 instance if CPU < 30%

This is the core of "elasticity" — automatic reaction to load.

D. Make Mattermost Stateless

This is critical and often missed.

Mattermost instances should **not store session or file data locally**.

Instead:

- Use **Amazon RDS** (MySQL/PostgreSQL) for database
- Use **Amazon S3** for file storage
- Enable **session stickiness (optional)** OR better:
  - Use shared session storage (Redis)

Why?

- New instances should be able to serve users immediately
- No dependency on a specific instance

E. Use Shared Storage for Files

If users upload files in Mattermost:

- Configure:
  - FileSettings -> DriverName = amazons3

Ensures all instances access the same files

F. Health Checks & Self-Healing

- ALB performs health checks
- ASG replaces unhealthy instances automatically

This gives **resilience + elasticity together**

G. Optional: Use Containerization (Advanced)

Instead of EC2:

- Use **Amazon ECS** or **Kubernetes (EKS)**
- Run Mattermost in containers
- Enable auto-scaling at container level

More advanced, but very scalable.

Elasticity for Mattermost is achieved using an Auto Scaling Group with scaling policies based on metrics like CPU, combined with an Application Load Balancer to distribute traffic. The application must be stateless by using external services like RDS for database and S3 for file storage, allowing instances to scale dynamically based on demand.