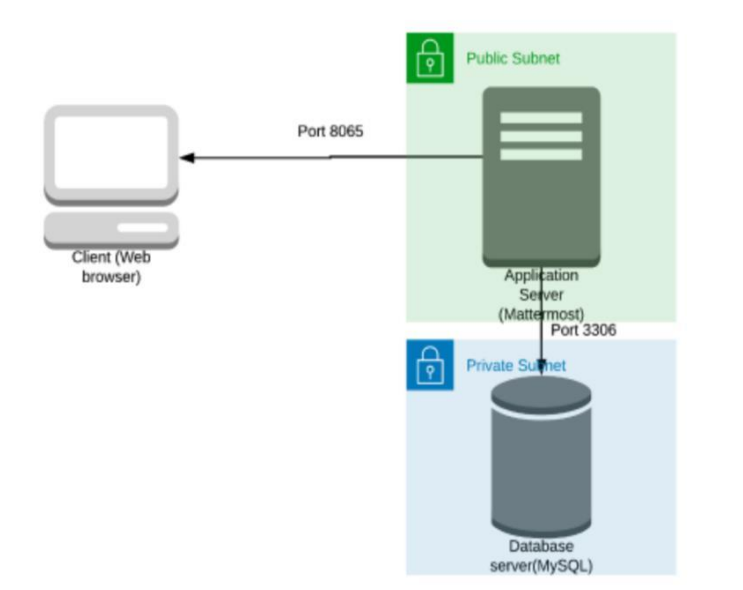


Scenario

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>

The screenshot displays the AWS VPC console interface. On the left, a navigation sidebar lists various services under 'Virtual private cloud', 'Security', and 'PrivateLink and Lattice'. The main content area is titled 'Your VPCs' and shows a table of existing VPCs. Two VPCs are listed: 'Project 1 VPC' (ID: vpc-068e55b5b434b1a3b) and another VPC (ID: vpc-07c04e2632d79c727). Both are in an 'Available' state. Below the table, the details for 'vpc-068e55b5b434b1a3b / Project 1 VPC' are shown. The 'Details' tab is active, displaying various configuration options: VPC ID, State (Available), Tenancy (default), Default VPC (No), Network Address Usage metrics (Disabled), Encryption control ID, Block Public Access (Off), DHCP option set (dopt-0f272c335278d2b42), IPv4 CIDR (10.0.0.0/16), Route 53 Resolver DNS Firewall rule groups (Failed to load rule groups), DNS hostnames (Enabled), Main route table (rtb-05c3f5cf9b4abbfb), IPv6 pool, and Owner ID (211125451470).

Name	VPC ID	State	Encryption c...	Encryption control ...	Block Public...	IPv4 CIDR	IPv6 CIDR	DHCP option
Project 1 VPC	vpc-068e55b5b434b1a3b	Available	-	-	Off	10.0.0.0/16	-	dopt-0f272c
-	vpc-07c04e2632d79c727	Available	-	-	Off	172.31.0.0/16	-	dopt-0f272c

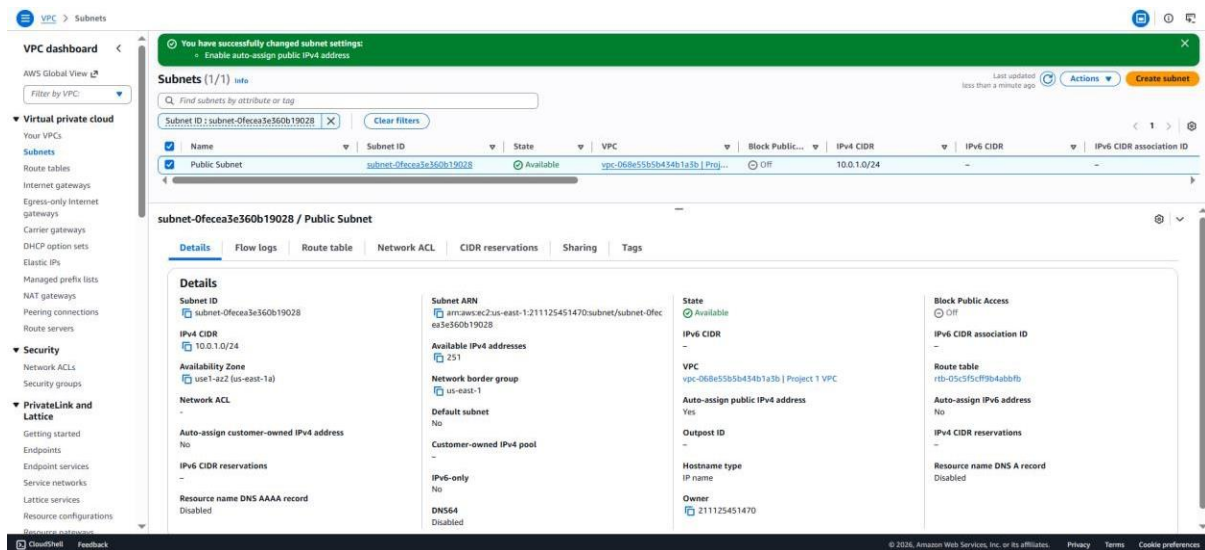
vpc-068e55b5b434b1a3b / Project 1 VPC

Details

- VPC ID: vpc-068e55b5b434b1a3b
- State: Available
- Tenancy: default
- Default VPC: No
- Network Address Usage metrics: Disabled
- Encryption control ID: -
- Encryption control mode: -
- Block Public Access: Off
- DHCP option set: dopt-0f272c335278d2b42
- IPv4 CIDR: 10.0.0.0/16
- Route 53 Resolver DNS Firewall rule groups: Failed to load rule groups
- DNS hostnames: Enabled
- Main route table: rtb-05c3f5cf9b4abbfb
- IPv6 pool: -
- Owner ID: 211125451470

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>

The screenshot shows the AWS Management Console interface for the 'Subnets' page. On the left, there is a navigation menu with options like 'VPC dashboard', 'Virtual private cloud', 'Security', 'PrivateLink and Lattice', and 'DNS firewall'. The main area displays a table of subnets. The 'Private Subnet' is selected, and its details are shown in a panel below the table. The details panel includes information such as Subnet ID, IPv4 CIDR, Availability Zone, Network ACL, Auto-assign customer-owned IPv4 address, IPv6 CIDR reservations, Resource name DNS AAAA record, Subnet ARN, Available IPv4 addresses, Network border group, Default subnet, Customer-owned IPv4 pool, IPv6-only, DNS64, State, Block Public Access, IPv6 CIDR association ID, Route table, Auto-assign IPv6 address, IPv4 CIDR reservations, Resource name DNS A record, and Hostname type.

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID	Available IPv4 addresses
-	subnet-0588235156c7c0a59	Available	vpc-07c04e263d79c727	Off	172.31.0.0/20	-	-	4091
-	subnet-0a1a2b35b0c5c524b	Available	vpc-07c04e263d79c727	Off	172.31.80.0/20	-	-	4091
-	subnet-074d2bfac8dffa66	Available	vpc-07c04e263d79c727	Off	172.31.48.0/20	-	-	4091
-	subnet-0dfc3a565d14157c	Available	vpc-07c04e263d79c727	Off	172.31.64.0/20	-	-	4091
Private Subnet	subnet-0de81ff4a6631e7d7	Available	vpc-068e55b5b434b1a3b Proj...	Off	10.0.2.0/24	-	-	251
-	subnet-03071c904ec9a567b	Available	vpc-07c04e263d79c727	Off	172.31.16.0/20	-	-	4091
-	subnet-023cd802d93747a2	Available	vpc-07c04e263d79c727	Off	172.31.32.0/20	-	-	4091
Public Subnet	subnet-0fcca3c360b19028	Available	vpc-068e55b5b434b1a3b Proj...	Off	10.0.1.0/24	-	-	251

subnet-0de81ff4a6631e7d7 / Private Subnet

Details

Subnet ID: subnet-0de81ff4a6631e7d7

Subnet ARN: arn:aws:ec2:us-east-1:211125451470:subnet/subnet-0de81ff4a6631e7d7

State: Available

Block Public Access: Off

IPv4 CIDR: 10.0.2.0/24

Available IPv4 addresses: 251

Availability Zone: us-east-1a

Network border group: us-east-1

Network ACL: acl-0c209b70286033506

Default subnet: No

Auto-assign customer-owned IPv4 address: No

Customer-owned IPv4 pool: -

IPv6 CIDR reservations: -

IPv6-only: No

Resource name DNS AAAA record: Disabled

DNS64: Disabled

VPC: vpc-068e55b5b434b1a3b | Project 1 VPC

Auto-assign public IPv4 address: No

Outpost ID: -

Hostname type: IP name

Owner: 211125451470

Route table: rtb-05c3f5c195b4ab0fb

Auto-assign IPv6 address: No

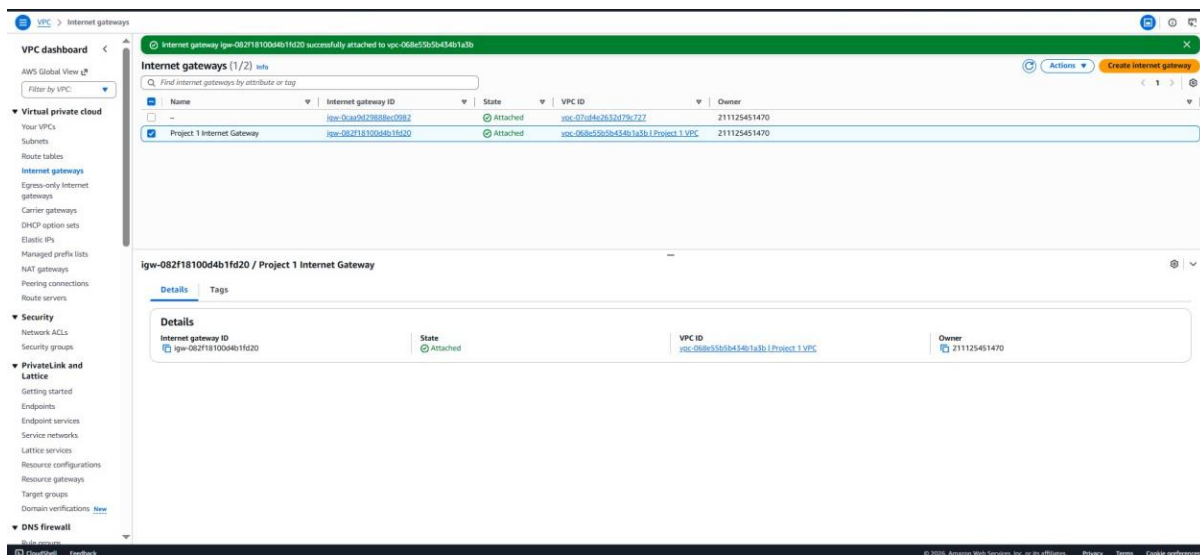
IPv4 CIDR reservations: -

Resource name DNS A record: Disabled

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 >

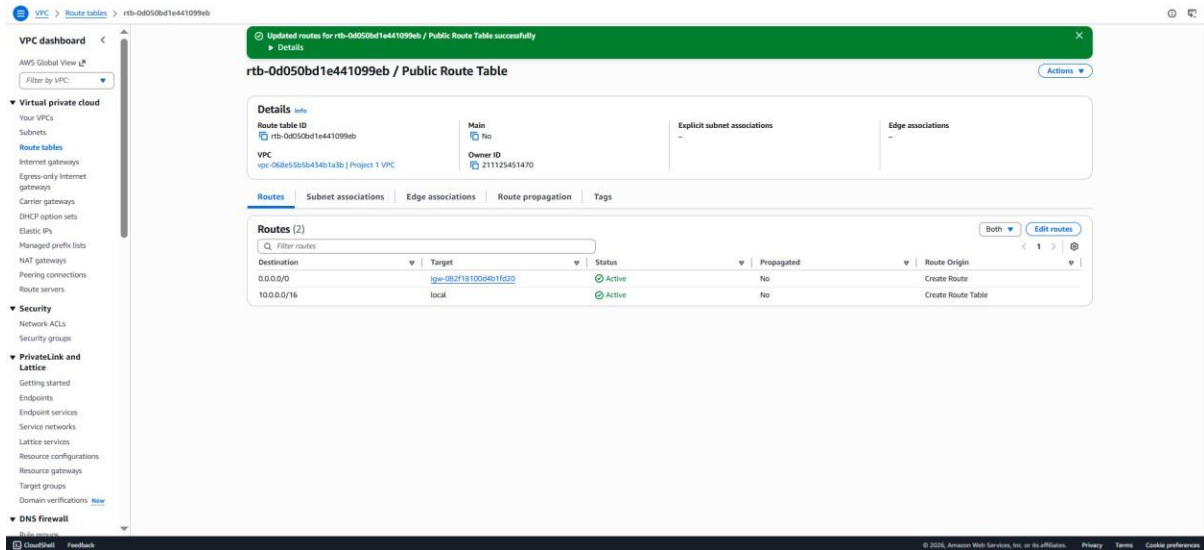


Step number	b
Step name	Creation of public route table
Instructions	1) Navigate to VPC -> Route Tables and click on Create Route table 2) Enter the name tag "Public Route Table", select the Project 1 VPC from the dropdown and click on Create 3) Once the route table is created, select it and select the 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

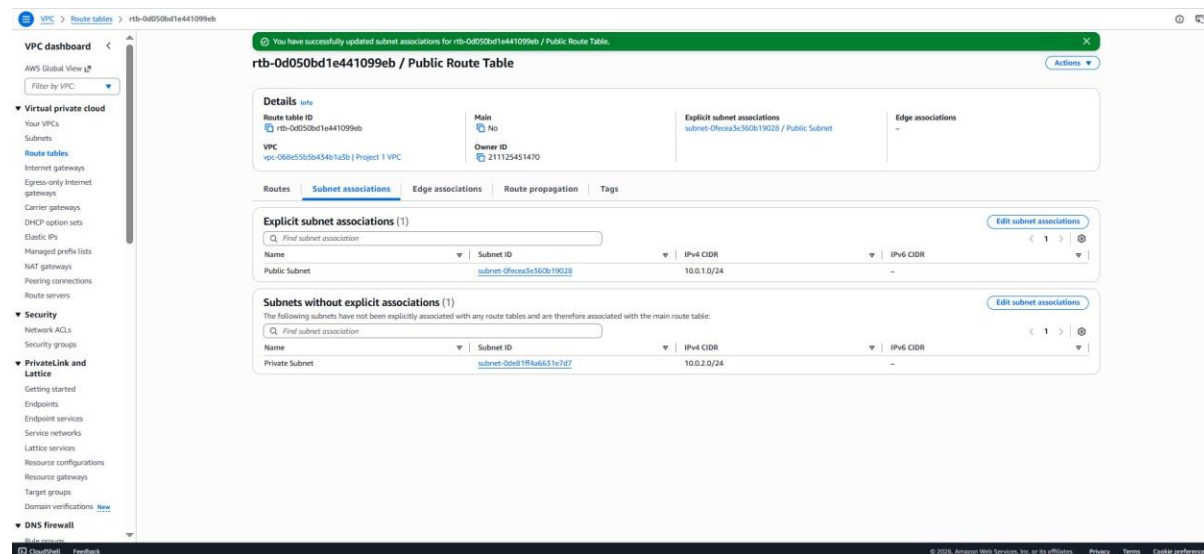
Expected screenshots

- 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>

Create NAT gateway

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the Internet.

NAT gateway settings

Name - optional
Create a tag with a key of "Name" and a value that you specify.
NAT_GW_Project1
The name can be up to 256 characters long.

Availability mode
Choose whether to deploy across all zones in the region or restrict to a single availability zone.
☐ Regional - new
Scales automatically across all regional AZs, simplifying management for multi-AZ deployments.
☒ Zonal
Provides granular control within a specific availability zone, adhering to subnet-level settings.

Subnet
Select a subnet in which to create the NAT gateway.
subnet-0feca5e360b19028 (Public Subnet)

Connectivity type
Select a connectivity type for the NAT gateway.
☒ Public
☐ Private

Elastic IP allocation ID
Assign an Elastic IP address to the NAT gateway.
eipalloc-073c0a4c256332877 [Allocate Elastic IP](#)

Additional settings

Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
Key: Name Value: NAT_GW_Project1
[Add new tag](#)
You can add 49 more tags.

[Cancel](#) [Create NAT gateway](#)

<Insert Screenshot c(2) here>

NAT gateways (1/1)

Name	NAT gateway ID	Connectivity...	State	State message	Availability ...	Route table ID	Primary public IP...	Primary private ...	Primary network...	VPC
NAT_GW_Project1	nat-0413db6a4a7b1f53e	Public	Available	-	Zonal	-	34.238.9.107	10.0.1.238	eni-0008e7158165dfe27	vpc-068e550b9434b1a3b / Proj...

nat-0413db6a4a7b1f53e / NAT_GW_Project1

Details Secondary IPv4 addresses Monitoring Tags

Details

NAT gateway ID nat-0413db6a4a7b1f53e	Connectivity type Public	State Available	State message -
NAT gateway ARN arn:aws:c2.us-east-1:21112543:1470:natgateway/nat-0413db6a4a7b1f53e	Primary public IPv4 address 34.238.9.107	Primary private IPv4 address 10.0.1.238	Primary network interface ID eni-0008e7158165dfe27
VPC vpc-068e550b9434b1a3b / Project 1 VPC	Subnet subnet-0feca5e360b19028 / Public Subnet	Created Saturday, 9 May 2025 at 18:53:18 GMT+5:30	Deleted -

Step number d

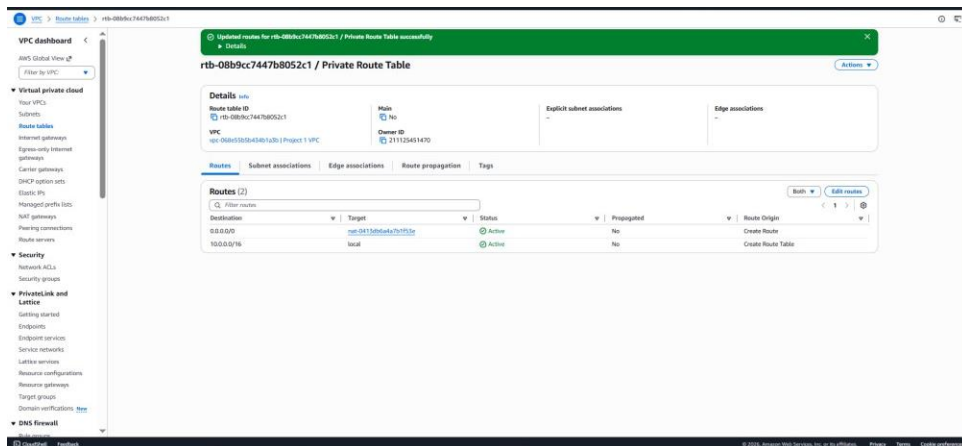
Step name Creation of private route tables

- 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 stepClick 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

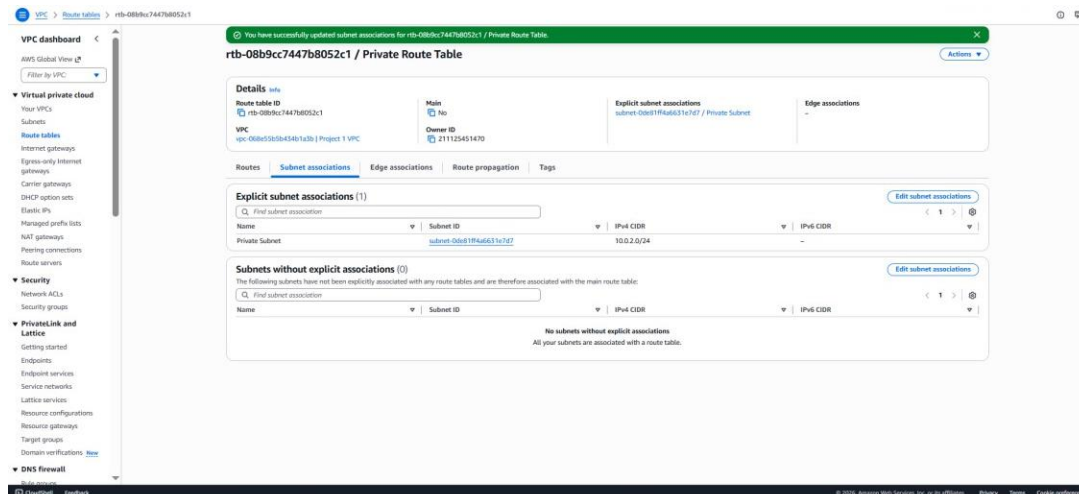
Expected 1) Route list of the route table

screenshots 2) Subnet association of the route table

<Insert Screenshot for d(1) here >



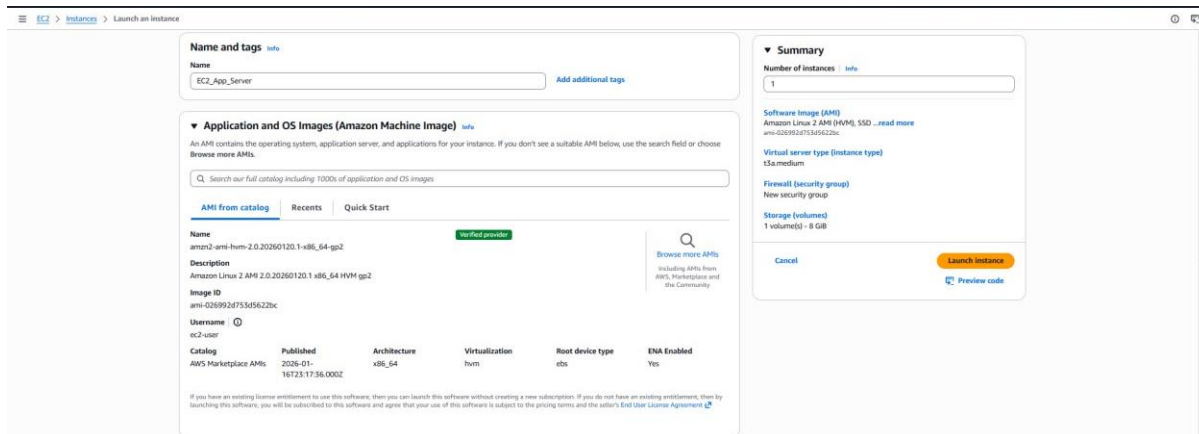
<Insert Screenshot for d(2) here>



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>



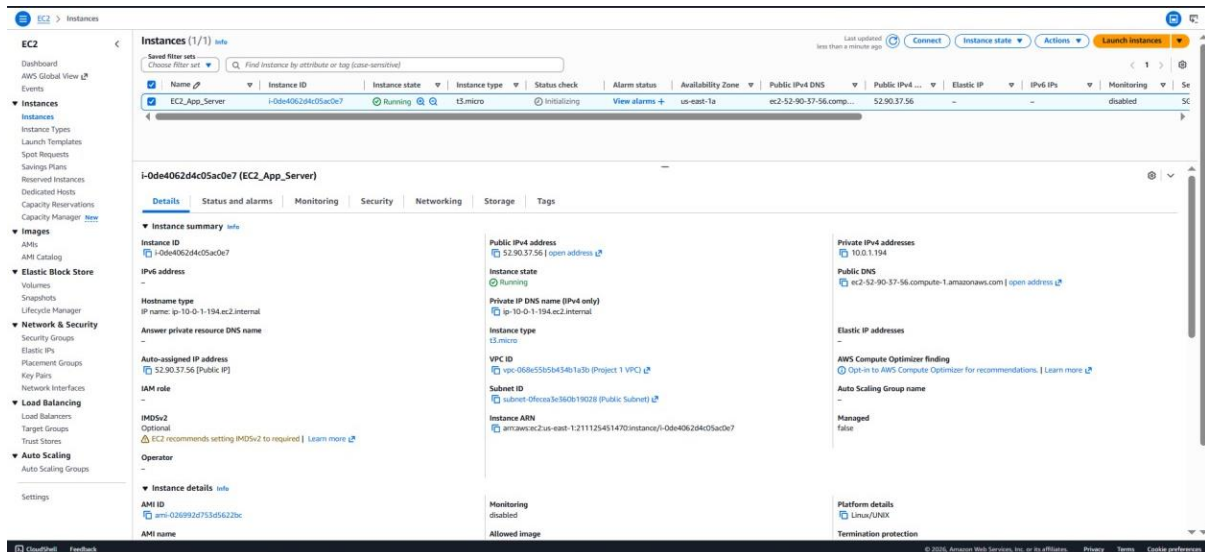
<Insert screenshot a(2) here>

The screenshot shows the 'Launch an instance' page in the AWS Management Console. The 'Instance type' section is set to 't3.micro' (Family t3, 2 vCPU, 1 GB Memory, Current generation, true). The 'Key pair (login)' section shows a key pair named 'pgccc-project1'. The 'Network settings' section shows the VPC 'vpc-068e5b5d34b141b' (Project 1 VPC) and Subnet 'subnet-0f6eca3e362b19c28' (Public Subnet). The 'Auto-assign public IP' is set to 'Enable'. The 'Firewall (security group)' section shows a new security group named 'sg_project1'. The 'Summary' section on the right shows the instance type 't3.micro', the software image 'Amazon Linux 2 AMI (HVM, SSD ...)', and the storage '1 volumes - 8 GB'. The 'Launch instance' button is visible.

<Insert screenshot a(3) here>

The screenshot shows the 'Launch an instance' page in the AWS Management Console, focusing on the 'Security group name - required' section. The security group name is 'sg_project1'. The 'Description - required' field contains 'Amazon Linux 2 AMI (HVM, SSD Volume Type (64-bit x86) Operating System-2.0.20240720.1-Autogun)'. The 'Inbound Security Group Rules' section shows four rules: 'SSH' (Type: SSH, Protocol: TCP, Port range: 22, Source: Anywhere, Destination: sg-068e5b5d34b141b), 'HTTP' (Type: HTTP, Protocol: TCP, Port range: 80, Source: Anywhere, Destination: sg-068e5b5d34b141b), 'HTTPS' (Type: HTTPS, Protocol: TCP, Port range: 443, Source: Anywhere, Destination: sg-068e5b5d34b141b), and 'Custom TCP' (Type: Custom TCP, Protocol: TCP, Port range: 8080, Source: Anywhere, Destination: sg-068e5b5d34b141b). The 'Summary' section on the right shows the instance type 't3.micro', the software image 'Amazon Linux 2 AMI (HVM, SSD ...)', and the storage '1 volumes - 8 GB'. The 'Launch instance' button is visible.

<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>

Marketplace Subscriptions: Successful

Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: [Add additional tags](#)

Application and OS Images (Amazon Machine Image)

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

AMI from catalog | Recents | Quick Start

Name: amzn2-ami-hvm-2.0.20260120.1-x86_64-gp2 Verified provider

Description: Amazon Linux 2 AMI 2.0.20260120.1 x86_64 HVM gp2

Image ID: ami-026992d75345622bc

Username: ec2-user

Catalog	Published	Architecture	Virtualization	Root device type	ENA Enabled
AWS Marketplace AMIs	2026-01-18T23:17:36.000Z	x86_64	hvm	efs	Yes

Instance type

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2 AMI (HVM, SSD) ...read more

Virtual server type (instance type): t3a.medium

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GB

[Cancel](#) [Launch instance](#) [Preview code](#)

<Insert screenshot b(2) here>

Instance type

t3.micro
Family: t3 2 vCPU 1 GB Memory Current generation: true [Free tier eligible](#)

The AMI vendor recommends using a t3a.medium instance (or larger) for the best experience with this product.

Key pair (login)

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

Key pair name - required: [Create new key pair](#)

Network settings

VPC - required: [Create new VPC](#)

Subnet: [Create new subnet](#)

Auto-assign public IP:

Firewall (security groups)

Create security group ☒ Select existing security group ☐

Security group name - required:

Description - required:

Inbound Security Group Rules

Security group rule 1 (TCP 22, 0.0.0.0/0) [Remove](#)

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2 AMI (HVM, SSD) ...read more

Virtual server type (instance type): t3.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GB

[Cancel](#) [Launch instance](#) [Preview code](#)

<Insert screenshot b(3) here>.

Firewall (security groups) [info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group ☐ Select existing security group

Security group name - required

Description - required [info](#)
This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters are a-z, 0-9, spaces, and _ (underscore).

Inbound Security Group Rules
▼ Security group rule 1 (TCP:22, 0.0.0.0/0) [Remove](#)

Type	Protocol	Port range
info	info	info
TCP	TCP	22
Source type	Source	Description - optional info
info	info	info
Anywhere	Add CIDR, prefix list or security group	e.g. SSH for admin desktop
	0.0.0.0/0	

▼ Security group rule 2 (TCP:80, 0.0.0.0/0) [Remove](#)

Type	Protocol	Port range
info	info	info
HTTP	TCP	80
Source type	Source	Description - optional info
info	info	info
Anywhere	Add CIDR, prefix list or security group	e.g. SSH for admin desktop
	0.0.0.0/0	

▼ Security group rule 3 (TCP:443, 0.0.0.0/0) [Remove](#)

Type	Protocol	Port range
info	info	info
HTTPS	TCP	443
Source type	Source	Description - optional info
info	info	info
Anywhere	Add CIDR, prefix list or security group	e.g. SSH for admin desktop
	0.0.0.0/0	

▼ Security group rule 4 (TCP:3306, 0.0.0.0/0) [Remove](#)

Type	Protocol	Port range
info	info	info
Custom TCP	TCP	3306
Source type	Source	Description - optional info
info	info	info
Anywhere	Add CIDR, prefix list or security group	e.g. SSH for admin desktop
	0.0.0.0/0	

[Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.](#)

Summary
[Number of instances](#) | [Info](#)

[Software image \(AMI\)](#)
Amazon Linux 2 AMI (HVM, SSD ... [read more](#)
ami-0269026753d5622bc
[Virtual server type \(instance type\)](#)
t3.micro
[Firewall \(security group\)](#)
New security group
[Storage \(volumed\)](#)
1 volume(s) - 8 GB
[Cancel](#) [Launch instance](#) [Preview code](#)

<Insert screenshot b(4) here>

Instances (1/2) [info](#)
Last updated less than a minute ago [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4	Elastic IP	IPv6 IPs	Monitoring	Se
EC2_Database_server	i-0160768cf8222bfe0	Running	t3.micro	Initializing	View alarms +	us-east-1a	-	-	-	-	disabled	19
EC2_App_Server	i-0db406204c05a0e7	Running	t3.micro	3/3 checks pass	View alarms +	us-east-1a	ec2-52-90-57-56.com...	52.90.57.56	-	-	disabled	5C

i-0160768cf8222bfe0 (EC2_Database_server)
[Details](#) [Status and alarms](#) [Monitoring](#) [Security](#) [Networking](#) [Storage](#) [Tags](#)

▼ Instance summary [info](#)
Instance ID
[i-0160768cf8222bfe0](#)
IPv6 address
-
Hostname type
IP name: ip-10-0-2-85.ec2.internal
Answer private resource DNS name
-
Auto-assigned IP address
-
IAM role
-
IMDSv2
Optional
EC2 recommends setting IMDSv2 to required | [Learn more](#)
Operator
-
▼ Instance details [info](#)
AMI ID
[ami-0269026753d5622bc](#)
AMI name

Public IPv4 address
-
Instance state
Running
Private IP DNS name (IPv4 only)
[ip-10-0-2-85.ec2.internal](#)
Instance type
t3.micro
VPC ID
[vpc-968c5385b434b1a3b \(Project 1 VPC\)](#)
Subnet ID
[subnet-0db1f4a6b31e7d7 \(Private Subnet\)](#)
Instance ARN
[arn:aws:ec2:us-east-1:121125451470:instance/i-0160768cf8222bfe0](#)

Monitoring
disabled
Allowed image

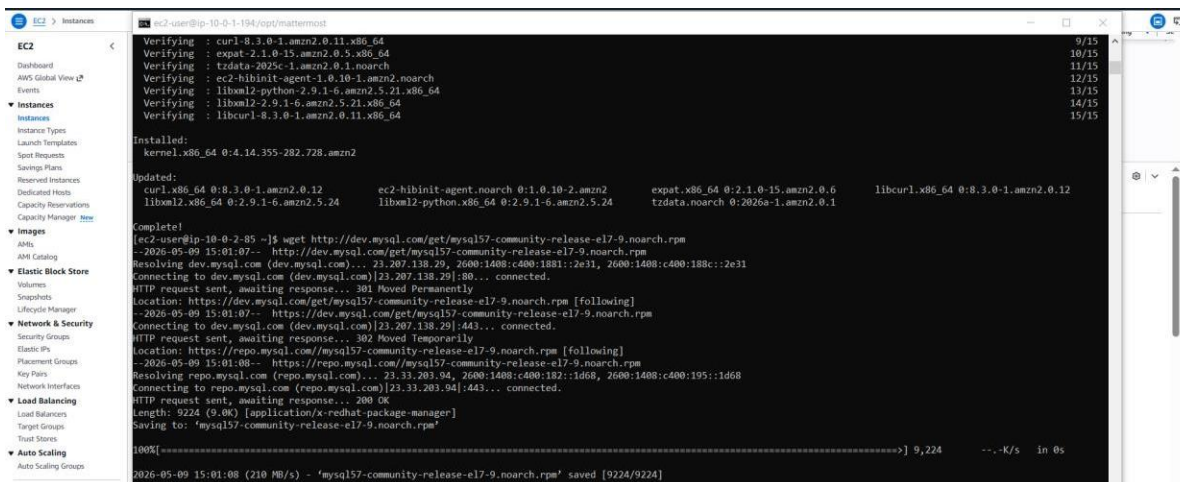
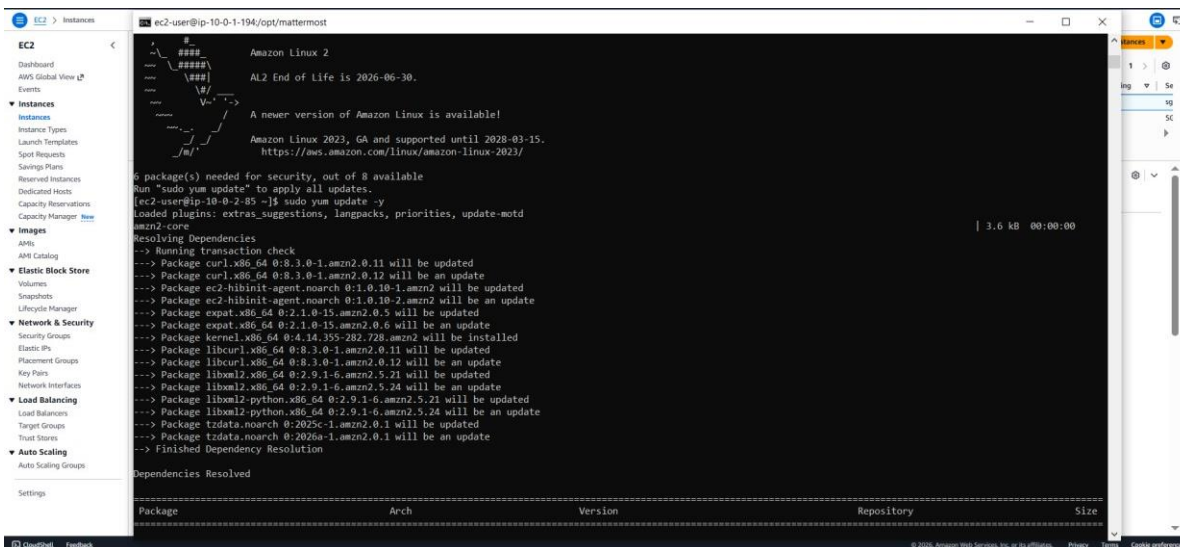
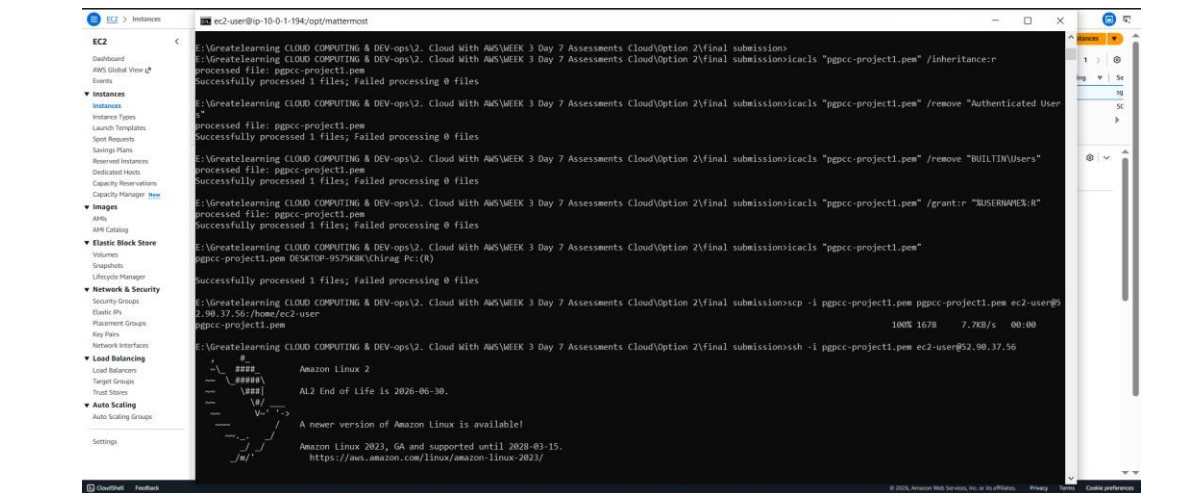
Private IPv4 addresses
[10.0.2.85](#)
Public DNS
-
Elastic IP addresses
-
AWS Compute Optimizer finding
[Opt-in to AWS Compute Optimizer for recommendations.](#) | [Learn more](#)
Auto Scaling Group name
-
Managed
false

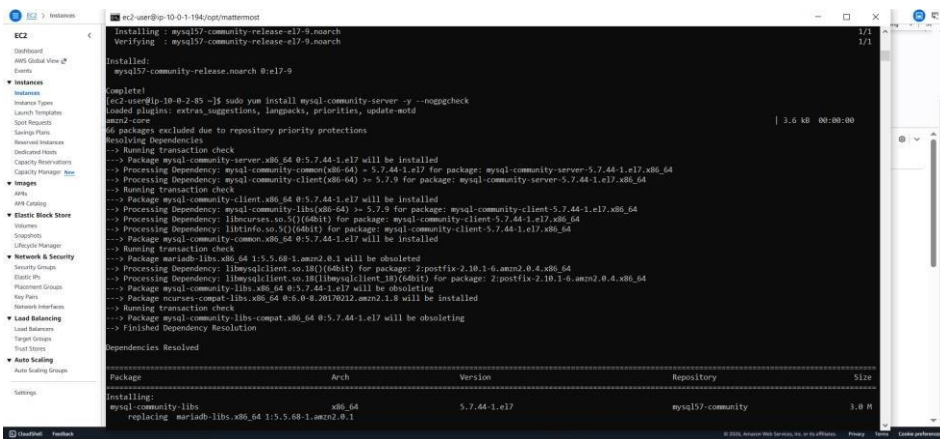
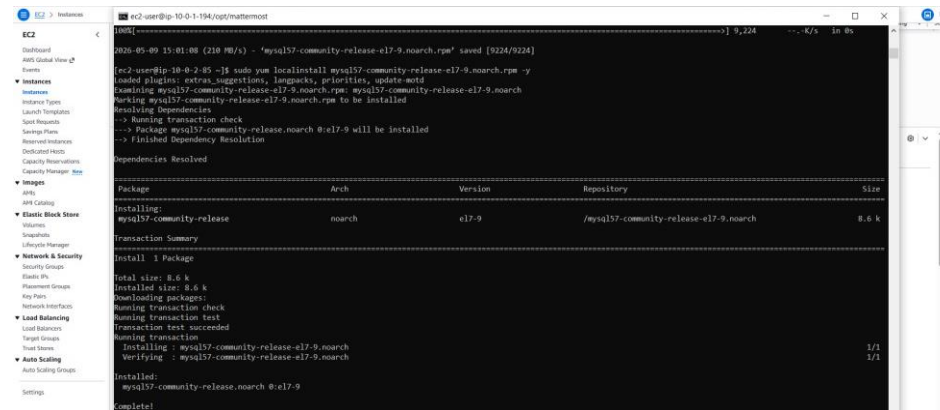
Platform details
[Linux/UNIX](#)
Termination protection

Step 4: Application and Database Installation and Testing

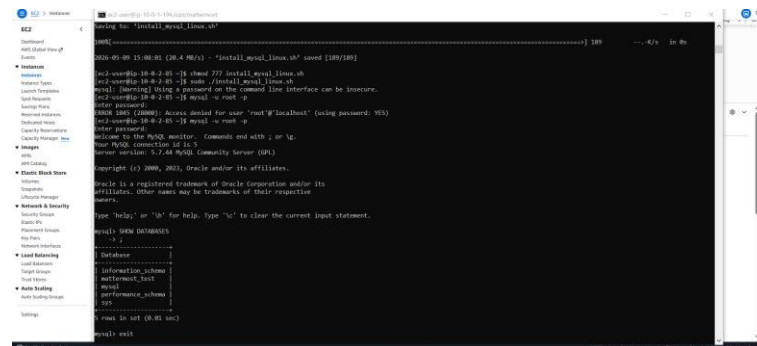
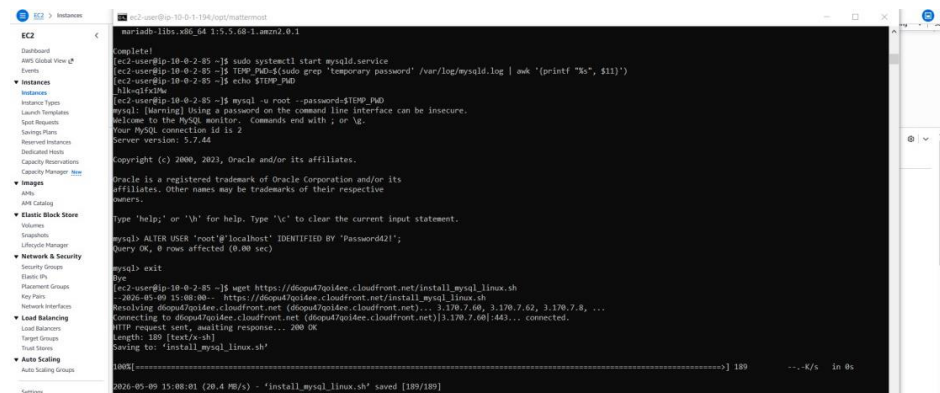
Step number	a
Step name	Installation and configuration of MySQL
Instructions	1) Copy the database pem file into the application server using the below command <pre>scp -i YOUR_APP.pem YOUR_DB.pem ec2-user@YOUR_APP_PUBLIC_IP:/home/ec2-user</pre> 2) Log into the application server using SSH/Putty 3) From 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 <pre>ssh -i YOUR_DB.pem ec2-user@YOUR_DB_PRIVATE_IP</pre> 4) Enter the following commands to install and configure MySQL on the database server <pre>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</pre> Run the below command to retrieve a temporary password for MySQL <pre>TEMP_PWD=\$(sudo grep 'temporary password' /var/log/mysqld.log awk '{printf "%s", \$11}')</pre> Log in to MySQL with the below command <pre>mysql -u root --password=\$TEMP_PWD</pre> Enter the below command after you login to MySQL. Do not change the password set in the below command. <pre>ALTER USER 'root'@'localhost' IDENTIFIED BY 'Password42!';</pre> 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. <pre>wget https://d6opu47qoi4ee.cloudfront.net/install_mysql_linux.sh chmod 777 install_mysql_linux.sh sudo ./install_mysql_linux.sh</pre> 5) Type <i>exit</i> 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>

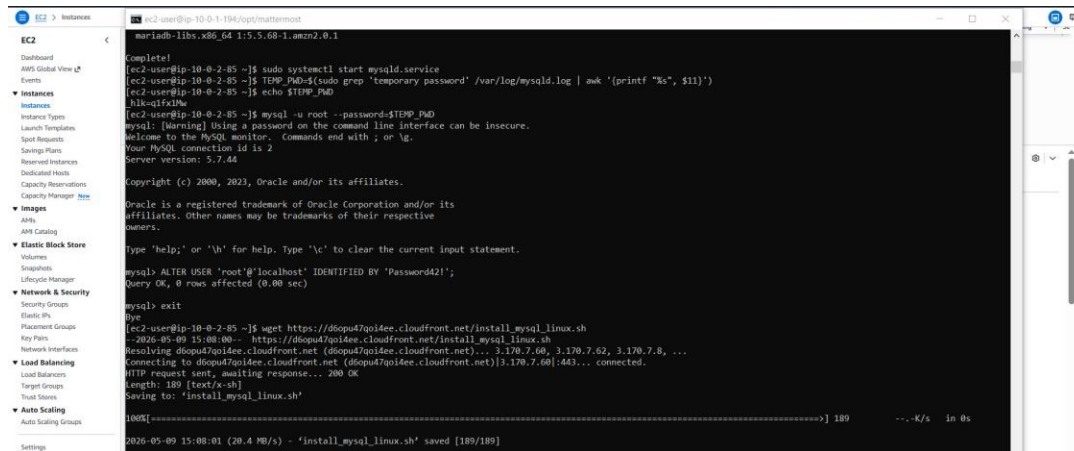




<Insert screenshot a(2) here> and Verifying Database



<Insert screenshot a(3) here>



```
marindb-11bs.x86_64 1:5:5.68-1.amzn2.0.1

Complete!
[ec2-user@ip-10-0-2-85 ~]$ sudo systemctl start mysqld.service
[ec2-user@ip-10-0-2-85 ~]$ TEMP_PWD=$(sudo grep 'temporary password' /var/log/mysqld.log | awk '{print $5}')
[ec2-user@ip-10-0-2-85 ~]$ echo $TEMP_PWD
_b1k-q1fxJw
[ec2-user@ip-10-0-2-85 ~]$ mysql -u root -p$TEMP_PWD
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.44

Copyright (c) 2000, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'Password21!';
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
[ec2-user@ip-10-0-2-85 ~]$ wget https://d6opu47qoi4ee.cloudfront.net/install_mysql_linux.sh
2026-05-09 15:08:06 -- https://d6opu47qoi4ee.cloudfront.net/install_mysql_linux.sh
Resolving d6opu47qoi4ee.cloudfront.net (d6opu47qoi4ee.cloudfront.net)... 3.170.7.60, 3.170.7.62, 3.170.7.8, ...
Connecting to d6opu47qoi4ee.cloudfront.net (d6opu47qoi4ee.cloudfront.net)[3.170.7.60]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 189 [text/x-sh]
Saving to: 'install_mysql_linux.sh'

install_mysql_linux.sh 100% [=====] 189  --K/s  in 0s

2026-05-09 15:08:01 (20.4 MB/s) - 'install_mysql_linux.sh' saved [189/189]
```

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

```
EC2 > Instances
Dashboard
AWS Global View
Events
▼ Instances
  Instance Profiles
  Launch Templates
  Spot Requests
  Savings Plans
  Reserved Instances
  Dedicated Hosts
  Capacity Reservations
  Capacity Manager
▼ Images
  AMIs
  AMI Catalog
▼ Elastic Block Store
  Volumes
  Snapshots
  Lifecycle Manager
▼ Network & Security
  Security Groups
  Elastic IPs
  Placement Groups
  Key Pairs
  Network Interfaces
▼ Load Balancing
  Load Balancers
  Target Groups
  Trust Stores
▼ Auto Scaling
  Auto Scaling Groups

Settings

Bye
[ec2-user@ip-10-0-0-2-85 ~]$ exit
Logout
Connection to 10.0.2.85 closed.
[ec2-user@ip-10-0-0-1-194 ~]$ wget https://d6opu47qoi4ee.cloudfront.net/install_mattermost_linux.sh
--2026-05-09 15:14:05-- https://d6opu47qoi4ee.cloudfront.net/install_mattermost_linux.sh
Resolving d6opu47qoi4ee.cloudfront.net (d6opu47qoi4ee.cloudfront.net)... 3.170.7.8, 3.170.7.60, 3.170.7.62, ...
Connecting to d6opu47qoi4ee.cloudfront.net (d6opu47qoi4ee.cloudfront.net)[3.170.7.8]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 693 [text/x-sh]
Saving to: 'install_mattermost_linux.sh'

100%[=====] 693 --.-K/s in 0s

2026-05-09 15:14:06 (39.1 MB/s) - 'install_mattermost_linux.sh' saved [693/693]

[ec2-user@ip-10-0-0-1-194 ~]$ sudo yum install dos2unix -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core | 3.6 kB 00:00:00
Resolving Dependencies
--> Running transaction check
--> Package dos2unix.x86_64 0:6.0.3-7.amzn2.0.3 will be installed
--> Finished Dependency Resolution

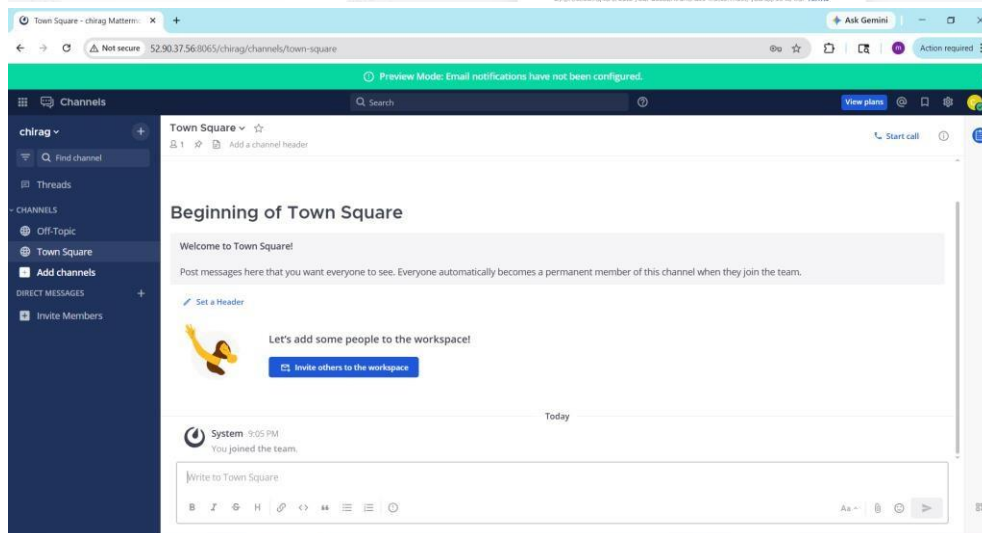
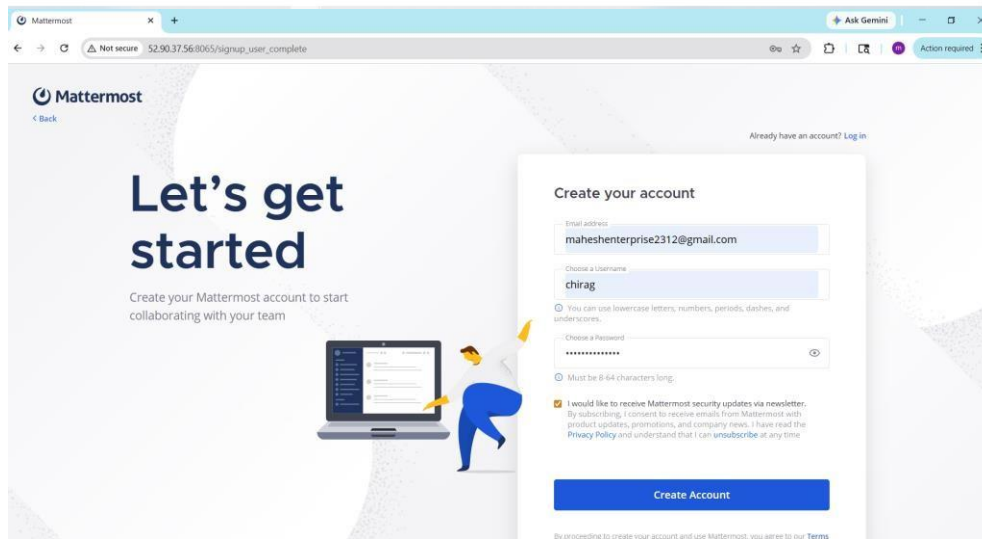
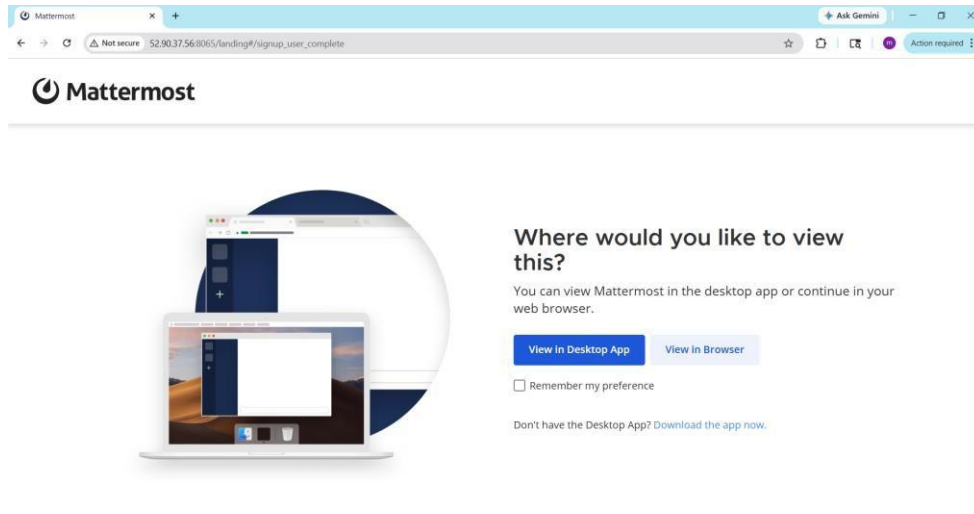
Dependencies Resolved

=====
Package Arch Version Repository Size
-----
Installing:
dos2unix x86_64 6.0.3-7.amzn2.0.3 amzn2-core 75 k
Transaction Summary
-----
Install 1 Package

Total download size: 75 k
Installed size: 190 k
Downloading packages:
dos2unix-6.0.3-7.amzn2.0.3.x86_64.rpm | 75 kB 00:00:00
```

[illegible]

<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 certificate (e.g., from Let's Encrypt, or a commercial CA). ○ Configure Mattermost to use HTTPS by setting up a reverse proxy (NGINX, Apache, Caddy) or using Mattermost's built-in TLS. ○ Redirect all HTTP traffic to HTTPS. ○ Why: Encrypts all data in transit, preventing eavesdropping, session hijacking, and man-in-the-middle attacks.
	• VPN as an additional encryption layer: ○ If users connect via VPN before reaching Mattermost, the entire VPN tunnel is encrypted. ○ This adds defense-in-depth: even if HTTPS were somehow compromised, the VPN protects the traffic.
	4. Putting It All Together (Recommended Architecture)
	For a public cloud deployment (AWS) :
	1. Create a Security Group for the Mattermost instance. ○ Inbound rule: HTTPS – source = your organization's public IP range OR the private IP range of your 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 allowlisting again. 4. Set allowed email domains in Mattermost System Console. 5. Enable MFA for all users. 6. Require VPN for any external access: ○ Users connect to corporate VPN (e.g., AWS Client VPN, OpenVPN, WireGuard). ○ The VPN server forwards traffic to Mattermost only after successful VPN authentication.
	Why This Combination Is Effective
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 together 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.