

AWS Secure Self-Hosted Collaboration Platform.

Mattermost + MySQL on AWS (Custom VPC Architecture)

By

Esmeralda C. Cabrera Ventura

Project Overview

Many organizations rely on real-time collaboration tools such as Slack or Microsoft Teams. However, regulated environments (government, defense, healthcare, finance) often cannot use third-party managed SaaS platforms due to compliance, data residency, or security policies.

This project demonstrates how to design, deploy, and validate a secure, self-hosted team communication platform on AWS, using open-source software and standard cloud networking best practices. The final solution deploys Mattermost (an open-source Slack alternative) backed by MySQL, hosted entirely inside a custom AWS VPC with proper network isolation, routing, and security controls.

Problem Statement

The challenge was to design an architecture that:

- Allows public user access to a collaboration application
- Keeps the database fully private and inaccessible from the internet
- Supports secure administrative access for setup and maintenance
- Uses AWS-native networking components (VPC, subnets, route tables, gateways)
- Mimics a real-world enterprise deployment, not a simplified demo

Solution Summary

The solution uses a two-tier architecture inside a custom VPC:

- **Public subnet** o Application Server (Mattermost)
 - Internet Gateway
 - NAT Gateway
- **Private subnet** o Database Server (MySQL)

Administrative access to the private database server is performed indirectly via a bastion host pattern, and outbound internet access from the private subnet is handled securely through a NAT Gateway.

High-Level Architecture

Key Components

- Custom VPC (10.0.0.0/16)
- Public Subnet (10.0.1.0/24)
- Private Subnet (10.0.2.0/24)
- Internet Gateway
- NAT Gateway with Elastic IP
- Public Route Table (IGW routing)
- Private Route Table (NAT routing)
- EC2 Application Server (Mattermost)
- EC2 Database Server (MySQL)

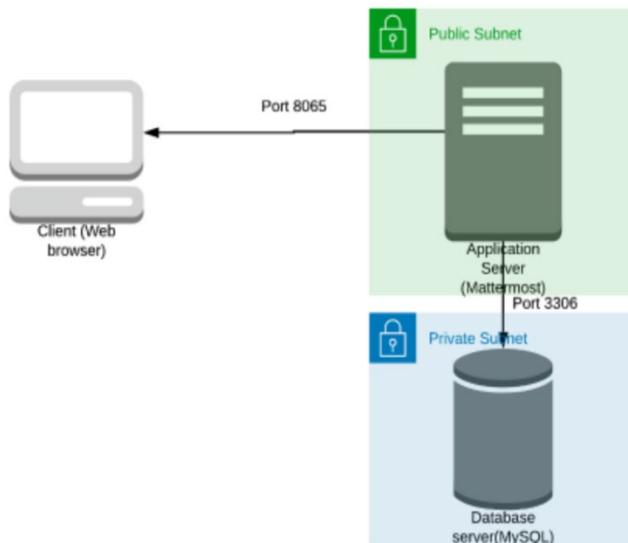
Traffic Flow

1. Users access Mattermost via `http://<public-ip>:8065`
2. Application server communicates with MySQL over the private subnet
3. Database server has no direct internet exposure
4. Outbound updates from the private subnet flow through the NAT Gateway

Architectural Design

Architecture Design Rationale Design Decision	Reason
Public/Private Subnet Separation	Enforces network isolation and least privilege
NAT Gateway	Allows private resources outbound access without inbound exposure
Bastion Host Access	Prevents direct SSH access to private resources
Custom Route Tables	Explicitly control traffic flow
Security Groups	Enforce port-level access control
Self-Hosted Stack	Full control over data and compliance posture

Architecture Diagram



Implementation

Each section below corresponds directly to the screenshots included in the /screenshots directory and follows the original execution order.

Step 1: Custom VPC and Subnet Creation

Objective: Establish isolated networking boundaries.

- Created a custom VPC with CIDR 10.0.0.0/16
- Enabled DNS hostnames for EC2 resolution
- Created:
 1. Public Subnet (10.0.1.0/24) with auto-assigned public IPs
 2. Private Subnet (10.0.2.0/24) with no public IP assignment.

VPC Creation

The screenshot shows the AWS VPC console interface. The top navigation bar includes the AWS logo, search bar, and account information (Account ID: 5242-4857-5754, vclabs/user4245679=esmeralda.cabrera.v@gmail.com). The main page displays a success message: "You have successfully modified the settings for vpc-06d46f1bdf96aab7a / Project 1 VPC." Below this, the VPC details are listed in a table:

Details		Info	
VPC ID	vpc-06d46f1bdf96aab7a	State	Available
DNS resolution	Enabled	Tenancy	default
Main network ACL	acl-0a0010d919ee95e28	Default VPC	No
IPv6 CIDR (Network border group)	-	Network Address Usage metrics	Disabled
		IPv4 CIDR	10.0.0.0/16
		DHCP option set	dopt-06319303c06d62f99
		Route 53 Resolver DNS Firewall rule groups	Failed to load rule groups
		Block Public Access	Off
		Main route table	rtb-027c59e50f92ea64f
		IPv6 pool	-
		Owner ID	524248575754
		DNS hostnames	Enabled

Below the details, there are tabs for Resource map, CIDRs, Flow logs, Tags, and Integrations. The Resource map tab is selected, showing a map view. The bottom of the page includes links for CloudShell, Feedback, and various legal and preference links.

Public Subnet Creation

The screenshot shows the AWS VPC Subnets page. A green success message at the top states: "You have successfully changed subnet settings: Enable auto-assign public IPv4 address". Below this, the "Subnets (1/1) Info" section shows a single subnet named "Public Subnet" with the ID "subnet-0c03d2eb733c0715c". The subnet is listed as "Available" and associated with the VPC "vpc-06d46f1bdf96aab7a". The "Details" tab is selected, displaying the following information:

Subnet ID	Subnet ARN	State	Block Public Access
subnet-0c03d2eb733c0715c	arn:aws:ec2:us-east-1:524248575754:subnet/subnet-0c03d2eb733c0715c	Available	Off
IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID	—
10.0.1.0/24	—	—	

Below the table, it says "Available IPv4 addresses". The page footer includes links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Private Subnet Creation

The screenshot shows the AWS VPC Subnets page. A green success message at the top states: "You have successfully created 1 subnet: subnet-007fb1af7aad3ae45". Below this, the "Subnets (1) Info" section shows a single subnet named "Private Subnet" with the ID "subnet-007fb1af7aad3ae45". The subnet is listed as "Available" and associated with the VPC "vpc-06d46f1bdf96aab7a". The "Select a subnet" section is visible below the table.

Name	Subnet ID	State
Private Subnet	subnet-007fb1af7aad3ae45	Available

The page footer includes links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Step 2: Internet Gateway, NAT Gateway, and Routing

Objective: Enable controlled internet connectivity.

- Attached an Internet Gateway to the VPC
- Created:
 - Public Route Table → 0.0.0.0/0 → Internet Gateway
 - Private Route Table → 0.0.0.0/0 → NAT Gateway
- Deployed a NAT Gateway in the public subnet with an Elastic IP
- Associated route tables with their respective subnets

Creation and Configuration of Internet Gateway

The screenshot shows the AWS VPC console interface. The top navigation bar includes the AWS logo, search bar, and account information (Account ID: 5242-4857-5754). The main navigation menu on the left is expanded to show 'Virtual private cloud' and 'Internet gateways'. The central content area displays the details of an Internet Gateway named 'igw-070b2f72a6d72a327'. A green success message at the top states: 'Internet gateway igw-070b2f72a6d72a327 successfully attached to vpc-06d46f1bdf96aab7a'. The 'Details' section shows the Internet gateway ID (igw-070b2f72a6d72a327), State (Attached), VPC ID (vpc-06d46f1bdf96aab7a), and Owner (524248575754). Below the details is a 'Tags' section with one tag: 'Name: Project 1 Internet Gateway'. At the bottom of the page, there are links for CloudShell, Feedback, and a footer with copyright information (© 2025, Amazon Web Services, Inc. or its affiliates.) and links for Privacy, Terms, and Cookie preferences.

Creation of public route table

The screenshot shows the AWS VPC console interface. In the top navigation bar, the URL is https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#RouteTableDetails:RouteTableId=rtb-0c20eabee4331e671. The account ID is 5242-4857-5754, and the region is United States (N. Virginia). The main content area displays a green success message: "Updated routes for rtb-0c20eabee4331e671 / Public Route Table successfully". Below this, the "Details" section shows the route table ID (rtb-0c20eabee4331e671), VPC (vpc-06d46f1bdf96aab7a), and owner ID (524248575754). The "Routes" tab is selected, showing two routes: one to igw-070b2f72a6d72a... (Status: Active) and one to local (Status: Active). The bottom right corner of the page includes links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

This screenshot shows the same AWS VPC console interface after updating subnet associations. The green success message now reads: "You have successfully updated subnet associations for rtb-0c20eabee4331e671 / Public Route Table." The "Subnet associations" tab is selected, showing one explicit subnet association for the Public Subnet (subnet-0c3d2eb733c0715c) with an IPv4 CIDR of 10.0.1.0/24. The bottom right corner of the page includes links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Creation of NAT gateway

The screenshot shows the AWS VPC console interface for creating a NAT gateway. At the top, a green banner indicates that an Elastic IP address (54.221.103.9) has been allocated. The main form is titled "Create NAT gateway" and includes the following fields:

- Name - optional:** my-nat-gateway-01
- Subnet:** subnet-0c03d2eb733c0715c (Public Subnet)
- Connectivity type:** Public (selected)
- Elastic IP allocation ID:** eipalloc-0a51621a0ca6d0d1d

At the bottom right of the form, there is a blue "Allocate Elastic IP" button.

The screenshot shows the AWS VPC console displaying the details of a newly created NAT gateway. The NAT gateway ID is nat-087e52748a3b50662. The "Details" section shows the following information:

NAT gateway ID	Connectivity type	State	State message
nat-087e52748a3b50662	Public	Pending	-
NAT gateway ARN	Primary public IPv4 address	Primary private IPv4 address	Primary network interface ID
arn:aws:ec2:us-east-1:52424857574:natgateway/nat-087e52748a3b50662	-	-	-
VPC	Subnet	Created	Deleted
vpc-06d46f1bd96aab7a / Project 1 VPC	subnet-0c03d2eb733c0715c / Public Subnet	Monday, September 1, 2025 at 19:10:57 PDT	-

Below the details, there are tabs for "Secondary IPv4 addresses", "Monitoring", and "Tags". A "Secondary IPv4 addresses" table is shown with one entry: "Search".

Step 3: EC2 Instance Deployment

Application Server (Public Subnet)

- Amazon Linux 2 AMI
- Instance type: t3.micro
- Security Group:
 - 22 (SSH)
 - 80 / 443 (future web access)
 - 8065 (Mattermost)
- Assigned public IP

Database Server (Private Subnet)

- Amazon Linux 2 AMI
- Instance type: t3.micro
- No public IP
- Security Group:
 - 22 (SSH via bastion)
 - 3306 (MySQL)
 - 80 / 443 (installation dependencies)

Creation of private route tables

The screenshot shows the AWS VPC console interface. The top navigation bar includes the AWS logo, search bar, and account information (Account ID: 5242-4857-5754, voleabs/user4245679=esmeralda.cabrera.v@gmail.com). The main menu on the left is expanded under 'Virtual private cloud' to show 'Route tables'. The current view is 'Route tables' for a specific VPC.

A green success message at the top right states: "Updated routes for rtb-0732ec4c757ad11c1 / Private Route Table successfully". Below this, the 'Details' section shows:

Route table ID rtb-0732ec4c757ad11c1	Main No	Explicit subnet associations -	Edge associations -
VPC vpc-06d46f1bdff96aab7a Project 1 VPC	Owner ID 524248575754		

The 'Routes' tab is selected, displaying two routes:

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	nat-087e52748a3b5...	Active	No	Create Route
10.0.0.16	local	Active	No	Create Route Table

The bottom of the page includes standard AWS footer links: CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

In the second part of the screenshot, another green success message appears: "You have successfully updated subnet associations for rtb-0732ec4c757ad11c1 / Private Route Table." The 'Subnet associations' tab is selected, showing one association:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
Private Subnet	subnet-007fb1af7aad3ae45	10.0.2.0/24	-

Creation of application server

The screenshot shows the AWS EC2 console interface for launching a new instance. The instance is being configured with the following details:

- Name and tags:** The name is set to "My App Server".
- Application and OS Images (Amazon Machine Image):** The selected AMI is "amzn2-ami-hvm-2.0.20250818.2-x86_64-gp2".
- Virtualization:** hvm
- Root device type:** ebs
- ENAv Enabled:** Yes
- Firewall (security group):** New security group
- Storage (volumes):** 1 volume(s) - 8 GiB
- Launch instance** button is visible.

The screenshot shows the AWS EC2 console interface for launching a new instance, with more detailed configuration options visible:

- Instance type:** t3.micro (selected from a dropdown).
- Key pair (login):** My Key (selected from a dropdown).
- Network settings:** VPC - required (selected). Subnet: subnet-03052e0b733c0715c (selected from a dropdown).
- Auto-assign public IP:** Enabled (selected from a dropdown).
- Security group:** A new security group is being created, named "Auto-generated security group". It contains one inbound rule: "Security group rule 1 (TCP: 22, 0.0.0.0/0)" allowing traffic on port 22 from anywhere.
- Launch instance** button is visible.

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

Inbound Security Group Rules

- Security group rule 1 (T0: 22, 0.0.0.0/0)
 - Type: ssh
 - Protocol: TCP
 - Port range: 22
 - Description - optional: e.g. SSH for admin desktop
- Security group rule 2 (T0: 80, 0.0.0.0/0)
 - Type: Custom TCP
 - Protocol: TCP
 - Port range: 80
 - Description - optional: e.g. SSH for admin desktop
- Security group rule 3 (T0: 443, 0.0.0.0/0)
 - Type: Custom TCP
 - Protocol: TCP
 - Port range: 443
 - Description - optional: e.g. SSH for admin desktop
- Security group rule 4 (T0: 8065, 0.0.0.0/0)
 - Type: Custom TCP
 - Protocol: TCP
 - Port range: 8065
 - Description - optional: e.g. SSH for admin desktop

Add security group rule

Advanced network configuration

Configure storage

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

Launch instance

Preview code

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:

EC2

Instances (1/1)

Successfully initiated starting of i-0b55b503920129a1b

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
My App Server	i-0b55b503920129a1b	Running	t3.micro	3/3 checks passed	View alarms	us-east-1d

i-0b55b503920129a1b (My App Server)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0b55b503920129a1b	54.175.121.95 open address	10.0.1.37
IPv6 address	Instance state	Public DNS
-	Running	ec2-54-175-121-95.compute-1.amazonaws.com open address

Creation of database server

The screenshot shows the AWS EC2 console at the URL <https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances>. The top navigation bar includes the AWS logo, a search bar, and account information for 'Account ID: 5242-4857-5754' and 'voclabs/user4245679-esmeralda.cabrer.v@gmail.com'. The main content area displays a green banner stating 'Marketplace Subscription: Successful'. Below this, the 'Launch an instance' section is shown, featuring a 'Name and tags' field where 'My Database Server' is entered, and an 'Add additional tags' button. The 'Application and OS Images (Amazon Machine Image)' section lists 'Amazon Linux 2 AMI (HVM, SSD ...read more)' as the selected image. Configuration options include 'Virtual server type (instance type)' set to 't3.micro', 'Firewall (security group)' set to 'New security group', and 'Storage (volumes)' showing 1 volume(s) - 8 GiB. A large orange 'Launch instance' button is prominently displayed. At the bottom, there is a note about existing license entitlements and a link to 'CloudShell Feedback Privacy Terms Cookie preferences'.

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

Search results

▼ Instance type [Info](#) [Get advice](#)

Instance type

t3.micro [Compare instance types](#)

Family: t3 - 2 vCPUs - 1 GiB Memory Current generation: true

All generations

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

▼ Key pair (login) [Info](#)

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

My Key [Create new key pair](#)

▼ Network settings [Info](#)

VPC - [Required](#) [Info](#)

vpc-0646f1bf76ab7a (Project 1 VPC)

Subnet

subnet-0270f1af72ad3a65 Private Subnet

VPC: vpc-0646f1bf76ab7a Owner: 524248575754 Availability Zone: us-east-1d (use1-az2)

Zone type: Availability Zone IP addresses available: 251 CIDR: 10.0.2.0/24

Create new subnet

Auto-assign public IP [Info](#)

Disable

Firewall (security group) [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

Amazon Linux 2 AMI (HVM, SSD Volume Type (64-bit x86) Operating System-2.0.20250818.2-Autogen

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: a-z, A-Z, 0-9, spaces.

Description - required [Info](#)

Amazon Linux 2 AMI (HVM, SSD Volume Type (64-bit x86) Operating System-2.0.20250818.2-Autogen

Inbound Security Group Rules

▼ Security group rule 1 (Tc: 22, 0.0.0/0)

Type [Info](#) Protocol [Info](#) Port range [Info](#) Remove

ssh TCP 22

Source type [Info](#) Source [Info](#) Description - optional [Info](#)

Anywhere [Q Add CIDR, prefix list or security group](#) e.g. SSH for admin desktop

▼ Summary

Number of instances [Info](#)

1

Software image (AMI) [Info](#) Amazon Linux 2 AMI (HVM, SSD ... [Read more](#)

Virtual server type (instance type) [Info](#) t3.micro

Firewall (security group) [Info](#) New security group

Storage (volumes) [Info](#) 1 volume(s) - 8 GB

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

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

Inbound Security Group Rules

- Security group rule 1 (TCP: 22, 0.0.0.0/0)**
 - Type: ssh
 - Protocol: TCP
 - Port range: 22
 - Source type: Anywhere
 - Description: e.g. SSH for admin desktop
- Security group rule 2 (TCP: 80, 0.0.0.0/0)**
 - Type: Custom TCP
 - Protocol: TCP
 - Port range: 80
 - Source type: Anywhere
 - Description: e.g. SSH for admin desktop
- Security group rule 3 (TCP: 443, 0.0.0.0/0)**
 - Type: Custom TCP
 - Protocol: TCP
 - Port range: 443
 - Source type: Anywhere
 - Description: e.g. SSH for admin desktop
- Security group rule 4 (TCP: 3306, 0.0.0.0/0)**
 - Type: Custom TCP
 - Protocol: TCP
 - Port range: 3306
 - Source type: Anywhere
 - Description: e.g. SSH for admin desktop

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.

Add security group rule Advanced network configuration

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:

EC2

Instances (1/2) Info Last updated 10 minutes ago

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
My App Server	i-0b55b503920129a1b	Running	t3.micro	3/3 checks passed	View alarms +
My Database S...	i-07165ffe740c5c23d	Running	t3.micro	3/3 checks passed	View alarms +

i-07165ffe740c5c23d (My Database Server)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary Info

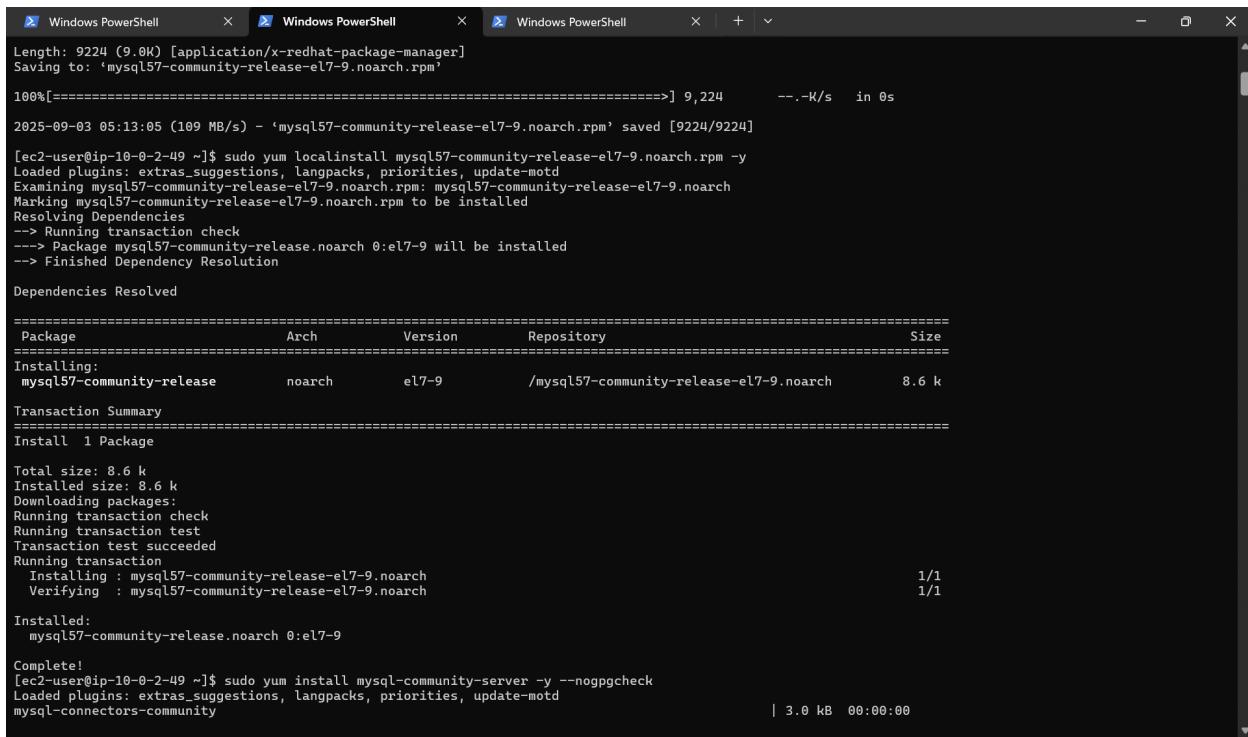
Instance ID i-07165ffe740c5c23d	Public IPv4 address -	Private IPv4 addresses 10.0.2.49
IPv6 address -	Instance state Running	Public DNS -

Step 4: MySQL Installation and Configuration (Private Server)

Objective: Prepare the database backend securely.

- Copied database PEM key to the application server
- Used SSH from the app server to reach the private database server
- Installed MySQL 5.7
- Retrieved temporary root password
- Set a permanent root password
- Executed provided initialization script

Installation and configuration of MySQL



The screenshot shows three terminal windows in a Windows PowerShell interface. The first window displays the download and save of the MySQL RPM package. The second window shows the yum localinstall command being run, including dependency resolution and transaction summary. The third window shows the sudo yum install command for the MySQL server, including the --nogpgcheck option. All commands are run by a user named 'ec2-user'.

```
Length: 9224 (9.0K) [application/x-redhat-package-manager]
Saving to: 'mysql57-community-release-el7-9.noarch.rpm'

100%[=====] 9,224      --.-k/s   in 0s

2025-09-03 05:13:05 (109 MB/s) - ‘mysql57-community-release-el7-9.noarch.rpm’ saved [9224/9224]

[ec2-user@ip-10-0-2-49 ~]$ sudo yum localinstall mysql57-community-release-el7-9.noarch.rpm -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Examining mysql57-community-release-el7-9.noarch.rpm: mysql57-community-release-el7-9.noarch
Marking mysql57-community-release-el7-9.noarch.rpm to be installed
Resolving Dependencies
--> Running transaction check
--> Package mysql57-community-release.noarch 0:el7-9 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
| Package           | Arch | Version | Repository | |
| ======           | ===  | ======  | ======     |
| Installing:      |       |          |            |
| mysql57-community-release | noarch | el7-9   | /mysql57-community-release-el7-9.noarch | 8.6 k |

Transaction Summary
=====
| Install 1 Package
| Total size: 8.6 k
| Installed size: 8.6 k
| Downloading packages:
| Running transaction check
| Running transaction test
| Transaction test succeeded
| Running transaction
|   Installing : mysql57-community-release-el7-9.noarch
|   Verifying  : mysql57-community-release-el7-9.noarch
|                                         1/1
|                                         1/1
| Installed:
|   mysql57-community-release.noarch 0:el7-9
| Complete!
[ec2-user@ip-10-0-2-49 ~]$ sudo yum install mysql-community-server -y --nogpgcheck
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
mysql-connectors-community
```

```
Windows PowerShell | ec2-user@ip-10-0-2-49:~ | + | - | X | 4/7
Verifying : ncurses-compat-libs-6.0-8.20170212.amzn2.1.8.x86_64 4/7
Verifying : mysql-community-common-5.7.44-1.el7.x86_64 5/7
Verifying : mysql-community-server-5.7.44-1.el7.x86_64 6/7
Verifying : 1:mariadb-libs-5.5.68-1.amzn2.0.1.x86_64 7/7

Installed:
mysql-community-libs.x86_64 0:5.7.44-1.el7      mysql-community-libs-compat.x86_64 0:5.7.44-1.el7
mysql-community-server.x86_64 0:5.7.44-1.el7

Dependency Installed:
mysql-community-client.x86_64 0:5.7.44-1.el7      mysql-community-common.x86_64 0:5.7.44-1.el7
ncurses-compat-libs.x86_64 0:6.0-8.20170212.amzn2.1.8

Replaced:
mariadb-libs.x86_64 1:5.5.68-1.amzn2.0.1

Complete!
[ec2-user@ip-10-0-2-49 ~]$ sudo systemctl start mysqld.service
[ec2-user@ip-10-0-2-49 ~]$ TEMP_PWD=$(sudo grep 'temporary password' /var/log/mysqld.log | awk '{printf "%s", $11}')
[ec2-user@ip-10-0-2-49 ~]$ mysql -u root --password=$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 'Password42!';
Query OK, 0 rows affected (0.00 sec)

mysql> |
```

```
Windows PowerShell | ec2-user@ip-10-0-2-49:~ | + | - | X | 4/7
[ec2-user@ip-10-0-2-49 ~]$ mysql -u root --password=$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 'Password42!';
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
[ec2-user@ip-10-0-2-49 ~]$ wget https://d6opu47qoi4ee.cloudfront.net/install_mysql_linux.sh
--2025-09-03 05:24:29-- https://d6opu47qoi4ee.cloudfront.net/install_mysql_linux.sh
Resolving d6opu47qoi4ee.cloudfront.net (d6opu47qoi4ee.cloudfront.net)... 3.162.115.128, 3.162.115.180, 3.162.115.193, ...
Connecting to d6opu47qoi4ee.cloudfront.net (d6opu47qoi4ee.cloudfront.net)|3.162.115.128|:443... connected.
HTTP request sent, awaiting response... _mysql_linux.sh
200 OK
Length: 189 [text/x-sh]
Saving to: 'install_mysql_linux.sh'

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

2025-09-03 05:24:30 (44.6 MB/s) - 'install_mysql_linux.sh' saved [189/189]

[ec2-user@ip-10-0-2-49 ~]$ chmod 777 install_mysql_linux.sh
[ec2-user@ip-10-0-2-49 ~]$ sudo ./install_mysql_linux.sh
mysql: [Warning] Using a password on the command line interface can be insecure.
[ec2-user@ip-10-0-2-49 ~]$
```

Step 5: Mattermost Installation and Validation

Objective: Deploy and validate the collaboration platform.

- Installed Mattermost on the application server
- Configured it to connect to the private MySQL instance
- Fixed permissions and ownership
- Started Mattermost service
- Verified successful deployment via browser

Access URL: http://<Application_Server_Public_IP>:8065

Installation and configuration of Mattermost

```
ec2-user@ip-10-0-2-74:~ [ec2-user@ip-10-0-1-218:/opt/ + ~] chmod +x install_mattermost_linux.sh
install_mattermost_linux.sh: 1: /usr/bin/sh: MUR_DB_PRIVATE_IP: command not found
sudo chmod +R mattermost:mattermost /opt/mattermost
sudo ./install_mattermost_linux.sh
2025-09-02 19:52:05 (0.0 0.00s) [ec2-user@ip-10-0-1-218 ~] Resolving dooput7qsi4ee.cloudfront.net (dooput7qsi4ee.cloudfront.net)... 3.162.115.191, 3.162.115.180, 3.162.115.161, ...
Connecting to dooput7qsi4ee.cloudfront.net (dooput7qsi4ee.cloudfront.net) 3.162.115.195:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 155314485 (1.92 MB)
Saving to: 'mattermost-5.19.0-linux-and64.tar.gz'

100%[=====] 155,314,485 26.2MB/s in 5.5s

2025-09-02 19:52:05 (0.0 0.00s) - 'mattermost-5.19.0-linux-and64.tar.gz' saved [155314485/155314485]

Downloaded Mattermost
mattermost
mattermost/client/
mattermost/client/18.11@F21/b2221/f7cd67.js
mattermost/client/icon_16x16.png
```

Not secure 54.175.121.95:8065/signup_email

Back

Mattermost

All team communication in one place, searchable and accessible anywhere

Let's create your account

Already have an account? [Click here to sign in.](#)

What's your email address?

Valid email required for sign-up

Choose your username

You can use lowercase letters, numbers, periods, dashes, and underscores.

Choose your password

[Create Account](#)

By proceeding to create your account and use Mattermost, you agree to our [Terms of Service](#) and [Privacy Policy](#). If you do not agree, you cannot use Mattermost.

Final Outcome

- Fully functional self-hosted collaboration platform
- Secure database isolation
- Controlled network access
- Realistic enterprise cloud architecture
- Clean separation of concerns

Security & Scalability Considerations

Security Enhancements

- Restrict SSH to known IPs or migrate to AWS SSM
- Lock MySQL access to application SG only
- Add HTTPS via ALB + ACM
- Disable open sign-ups in Mattermost

Scalability Options

- Move MySQL to Amazon RDS
- Store uploads on S3 or EFS
- Add Auto Scaling Group for application servers
- Front with Application Load Balancer

Skills Demonstrated

- AWS VPC design
- Subnetting and CIDR planning

- Internet & NAT Gateways
- Route table configuration
- Bastion host pattern
- EC2 provisioning
- Linux server administration
- Secure application deployment
- Cloud troubleshooting