# 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**
    - Application Server (Mattermost)
    - Internet Gateway
    - NAT Gateway
- **Private subnet**
    - 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**

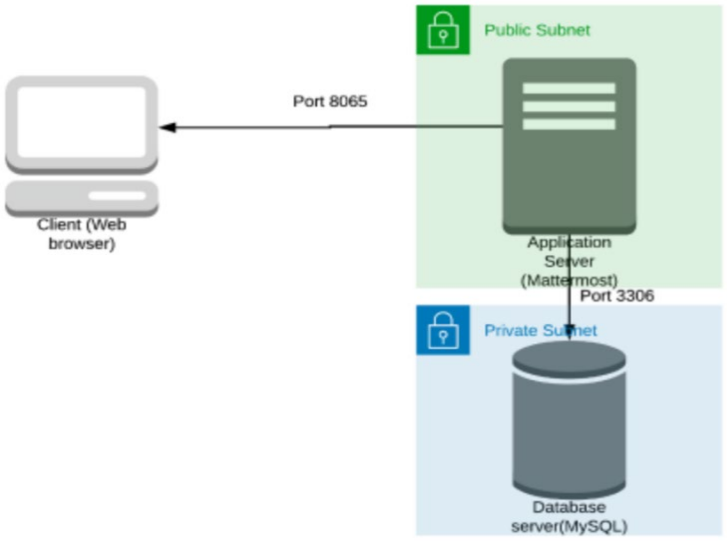
**Key Components**

- 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

**Architecture Diagram**



**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 |

**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:
    - Public Subnet (10.0.1.0/24) with auto-assigned public IPs
    - Private Subnet (10.0.2.0/24) with no public IP assignment

*Screenshots:*

- VPC creation and properties
- Public subnet creation
- Private subnet creation

**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

*Screenshots:*

- Internet Gateway attachment
- Public route table routes & subnet association
- NAT Gateway creation & status
- Private route table routes & subnet association

**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)

*Screenshots:*

- AMI selection
- Instance configuration
- Security group rules
- Running instances

**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

*Screenshots:*

- MySQL installation output
- Temporary password retrieval
- Script execution

**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

*Screenshots:*

- Script execution
- Mattermost service startup
- Web UI access confirmation

**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