Final Report

==

Title	**:**	Cloud Migration Project
Project Focus	**:**	End-to-end Cloud Migration and Automated Deployment of Kimai
Organization	**:**	Tech Force Services
Duration	**:**	15 Days (Onsite)
Internship Mode	**:**	Onsite
Location	**:**	Pallikaranai, Chennai

Intern Name	**:**	Akshaya
Role	**:**	DevOps
github repository	**:**	https://github.com/akshayaravi05/It-infrastructure

==

1. Introduction

In the course of my internship with Tech Force Services, I worked on migrating the Kimai timesheet application to a virtualized cloud infrastructure. The project involved provisioning infrastructure using Terraform, deploying the application with Docker, and automating the deployment workflow using Jenkins. The focus was on ensuring a lightweight, secure, and automated deployment using modern DevOps tools.

2. Project Objectives

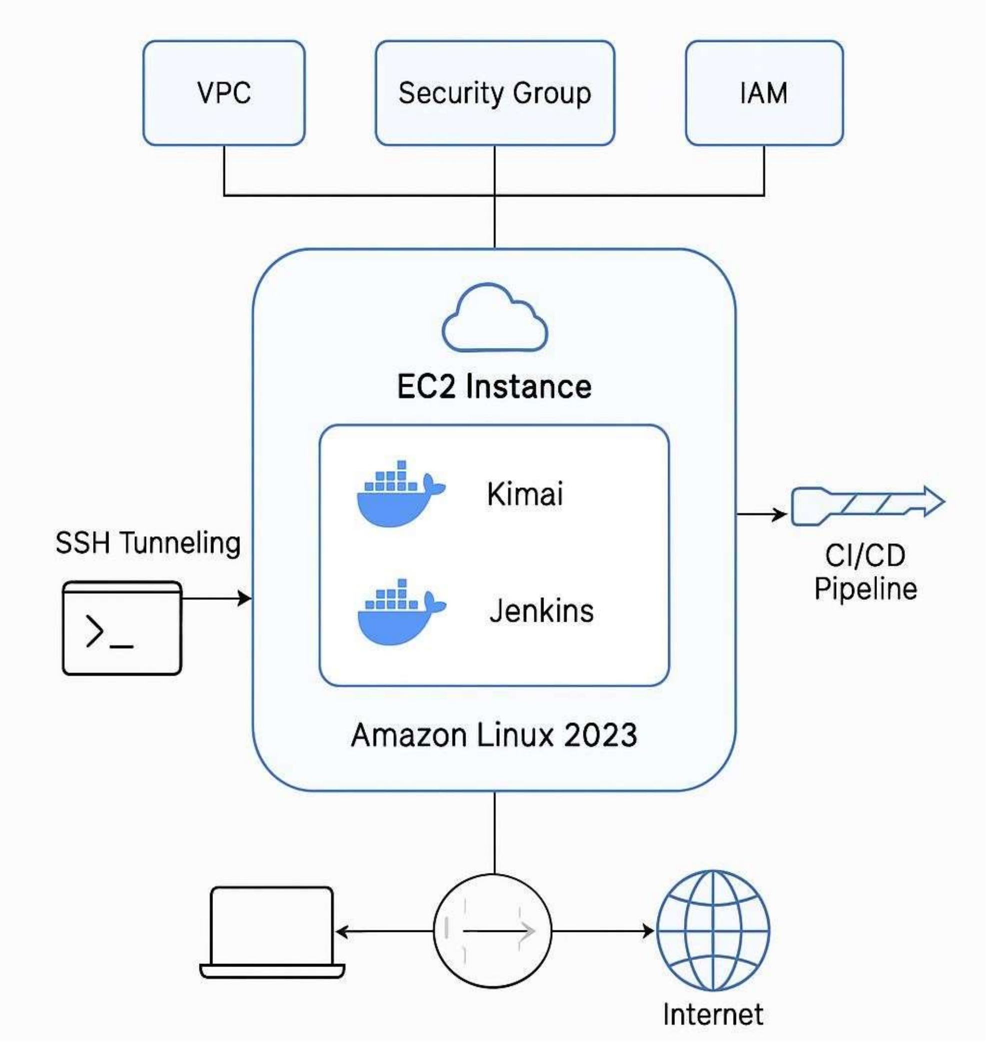
- Design a cost-efficient and scalable architecture optimized for AWS Free Tier
- Automate EC2 provisioning and setup using Terraform scripts and user data
- Deploy multi-container applications using Docker Compose
- Configure persistent storage and environment variables for Kimai containers
- Integrate version control using GitHub for tracking infrastructure and deployment code
- Use Jenkins to automate build, test, and deployment workflows
- Implement basic logging using Docker logs and EC2 system logs for debugging
- Configure SSH key-based access for secure EC2 login
- Maintain deployment documentation and configuration files in a public/private GitHub repo
- Conduct functional testing post-deployment to verify service availability
- Explore service scaling options within the limitations of Free Tier
- Document infrastructure diagrams (HLD & LLD) and provide setup instructions for replication

3. Tools & Technologies

Area	Tools/Services
Virtual Platform	Local VM / VirtualBox / On-Prem Server
Infrastructure	Terraform
Containerization	Docker, Docker Compose
Automation	Jenkins
Operating System	Linux (e.g., Ubuntu, AlmaLinux)
Version Control	GitHub

4. Architecture Overview

- All services (Kimai, Jenkins) hosted on a single virtual machine
- Multi-container environment managed using Docker Compose
- Infrastructure provisioned using Terraform (VM setup, user roles, firewall)
- SSH tunneling used to securely access internal services when required



5. Implementation Stages

Phase 1 : Design & Planning

1.1 High-Level Design (HLD)

1.2 Low-Level Design (LLD)

Phase 2 : Infrastructure as Code

2.1 Terraform Setup

Phase 3 : Deployment & CI/CD

3.1 Dockerization

3.2 CI/CD Pipeline

Phase 4 : Security

4.1 Network Hardening

4.2 Secure Remote Access

4.3 IAM Implementation

Phase 5 : Monitoring & Logging

5.1 Centralized Logging

5.2 Performance Monitoring

5.3 Alerting

Phase 6 : Assessment

6.1 Best Practice Assessment

6.2 Cost Assessment

Phase 1: Design & Planning

To conceptualize and architect the overall deployment, ensuring scalability, Maintainability, and security before implementation begins.

High-Level Design (HLD)

Low-Level Design (LLD)

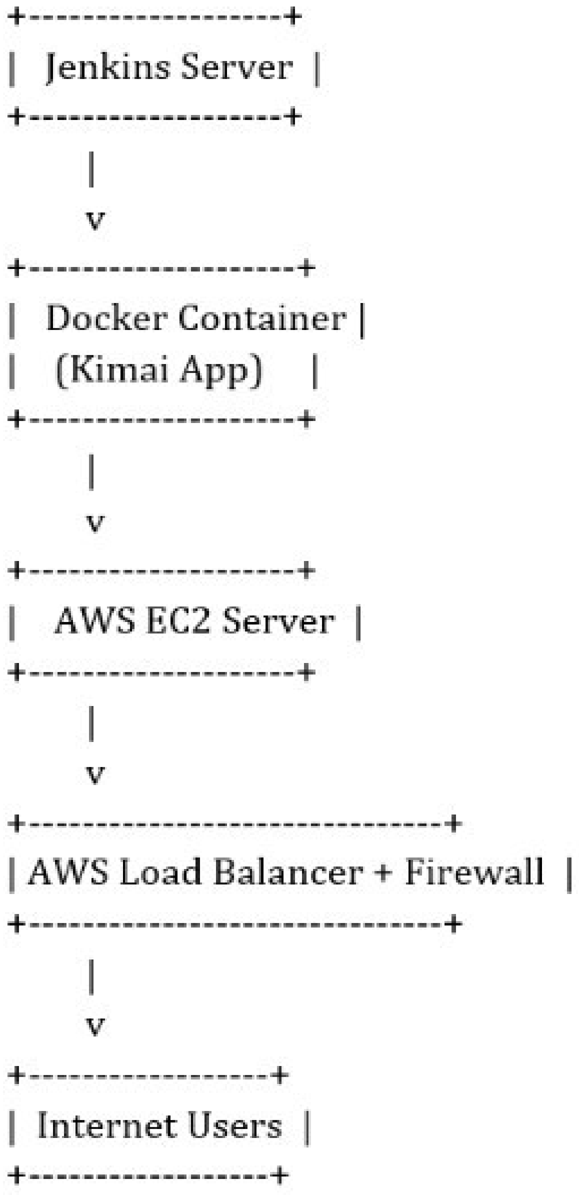
1.1 **High-Level Design (HLD)**

- Provides a top-down view of the solution architecture.
- Visualizes how different components (Kimai app, CI/CD, security, Containers) interact.
- Identifies major elements like servers, containers, users, pipelines, and Access flow.
- Highlights service connectivity, firewall boundaries, and data/control flow.

1.2Low-Level Design (LLD)

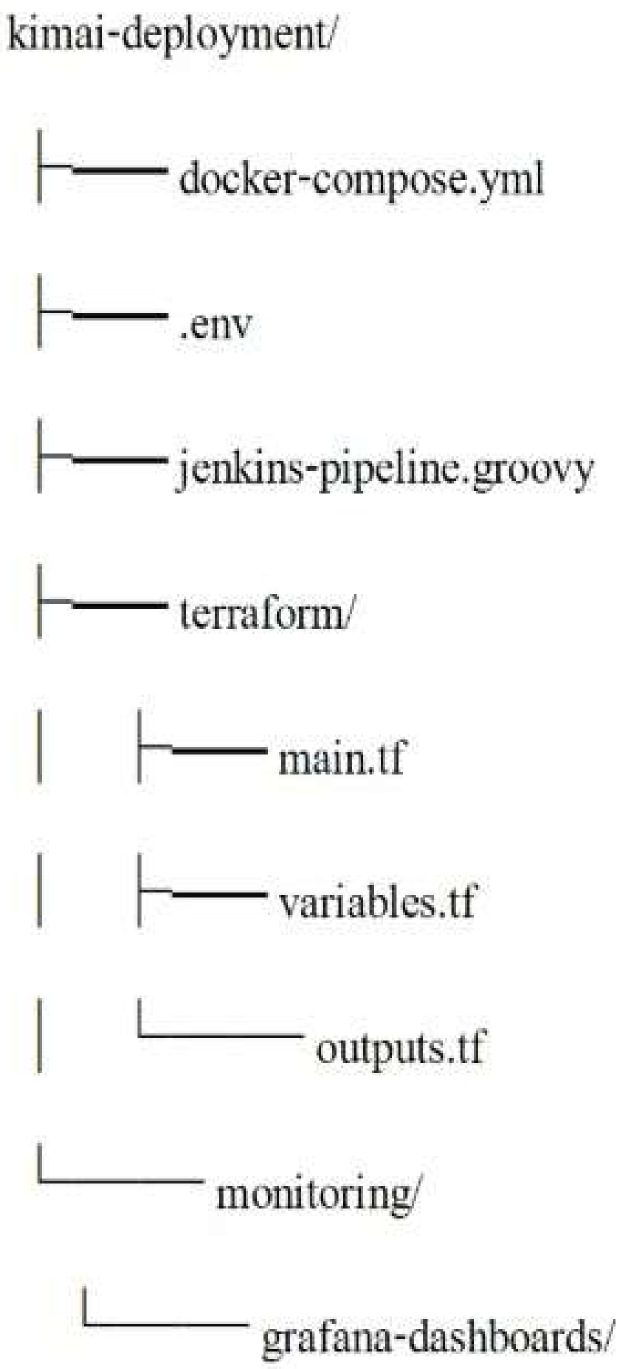
- Breaks down HLD into detailed configurations for implementation.
- Specifies exact VM/container specs, port numbers, directories, volumes, Environment variables, and service-to-service communication.
- Includes Docker Compose file structure, user-data scripts, Jenkinsfile, and Terraform resource mapping.

High-Level Design



```
+-------------------+
|  Jenkins Server  |
+-------------------+
      |
      v
+--------------------+
|  Docker Container |
|   (Kimai App)    |
+--------------------+
      |
      v
+--------------------+
|   AWS EC2 Server  |
+--------------------+
      |
      v
+-------------------------------+
| AWS Load Balancer + Firewall  |
+-------------------------------+
      |
      v
+------------------+
| Internet Users  |
+------------------+
```

Low-Level Design



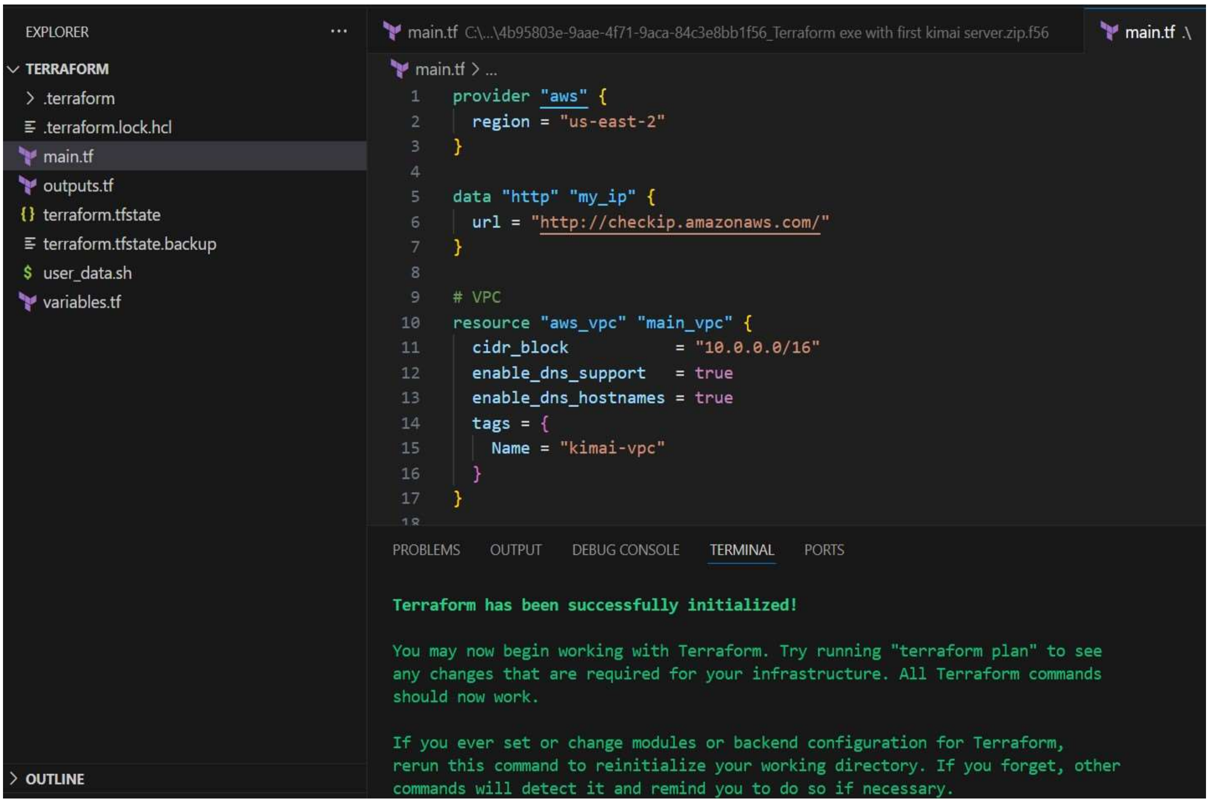
```
kimai-deployment/
├── docker-compose.yml
├── .env
├── jenkins-pipeline.groovy
├── terraform/
│   ├── main.tf
│   ├── variables.tf
│   └── outputs.tf
└── monitoring/
    └── grafana-dashboards/
```

Phase 2: Infrastructure as code(IaC)

To automate the setup of virtual infrastructure using code, ensuring repeatability, version control, and error reduction.

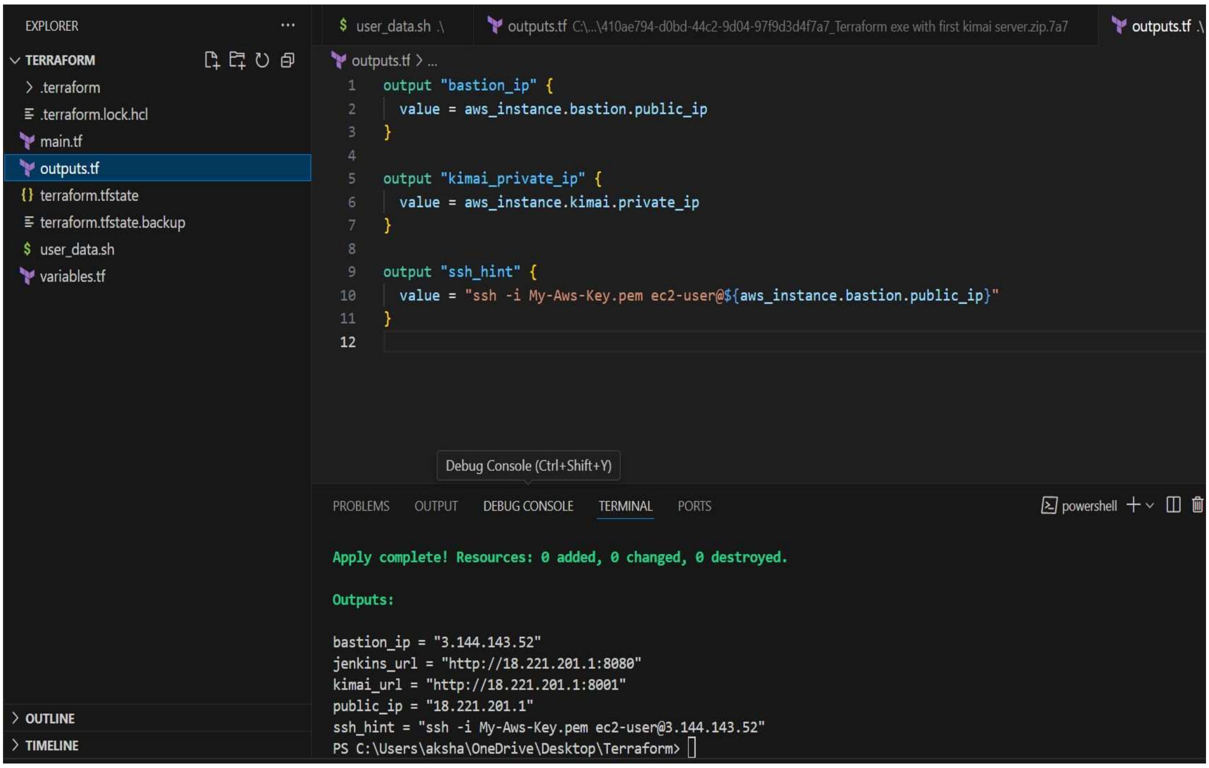
2.1 Terraform Setup

- Used **Terraform** to provision AWS resources such as EC2 instances, Security Groups, and IAM roles.
- Configured networking using **AWS Security Groups** to control inbound and outbound traffic to the EC2 instance.
- Code divided into main.tf, variables.tf, outputs.tf, and user_data.sh for modularity.
- Used remote-exec and user data to automatically install Docker and pull Kimai source code.



```
provider "aws" {
  region = "us-east-2"
}

data "http" "my_ip" {
  url = "http://checkip.amazonaws.com/"
}

# VPC
resource "aws_vpc" "main_vpc" {
  cidr_block           = "10.0.0.0/16"
  enable_dns_support   = true
  enable_dns_hostnames = true
  tags = {
    Name = "kimai-vpc"
  }
}
```

```
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```



```
output "bastion_ip" {
  value = aws_instance.bastion.public_ip
}

output "kimai_private_ip" {
  value = aws_instance.kimai.private_ip
}

output "ssh_hint" {
  value = "ssh -i My-Aws-Key.pem ec2-user@${aws_instance.bastion.public_ip}"
}

```

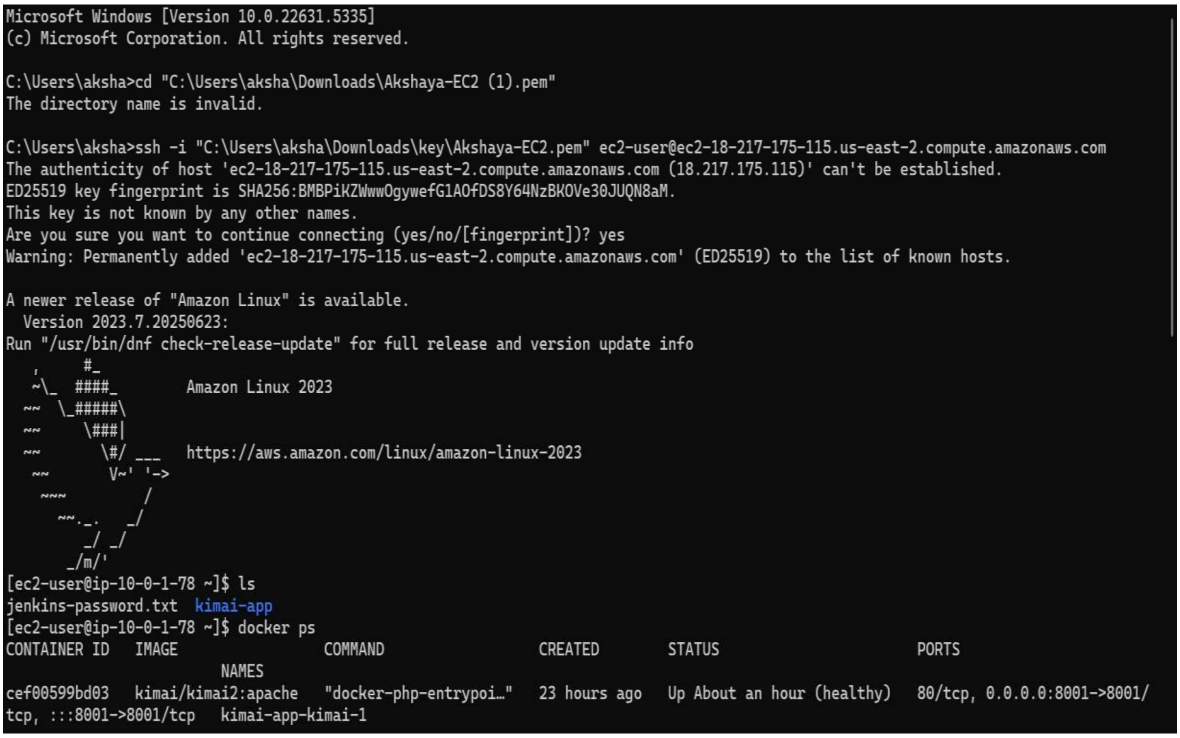
```
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

bastion_ip = "3.144.143.52"
jenkins_url = "http://18.221.201.1:8080"
kimai_url = "http://18.221.201.1:8001"
public_ip = "18.221.201.1"
ssh_hint = "ssh -i My-Aws-Key.pem ec2-user@3.144.143.52"
PS C:\Users\aksha\OneDrive\Desktop\Terraform>
```

Phase 3: Deployment & CI/CD

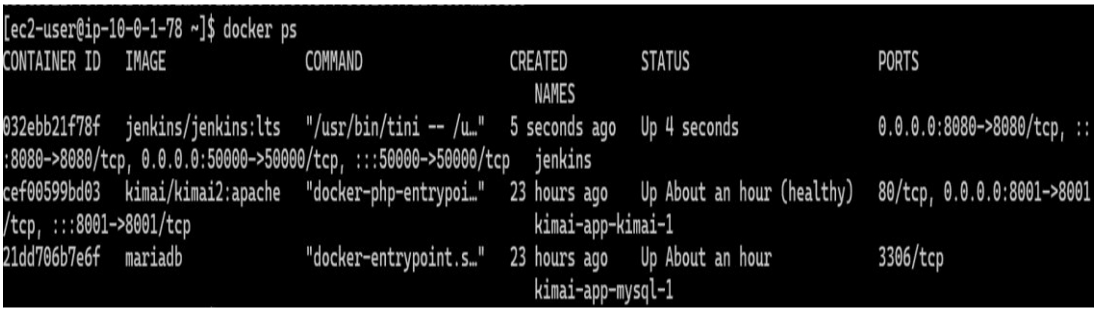
To automate the deployment and update process of the application using containers and pipelines.

3.1 Dockerization

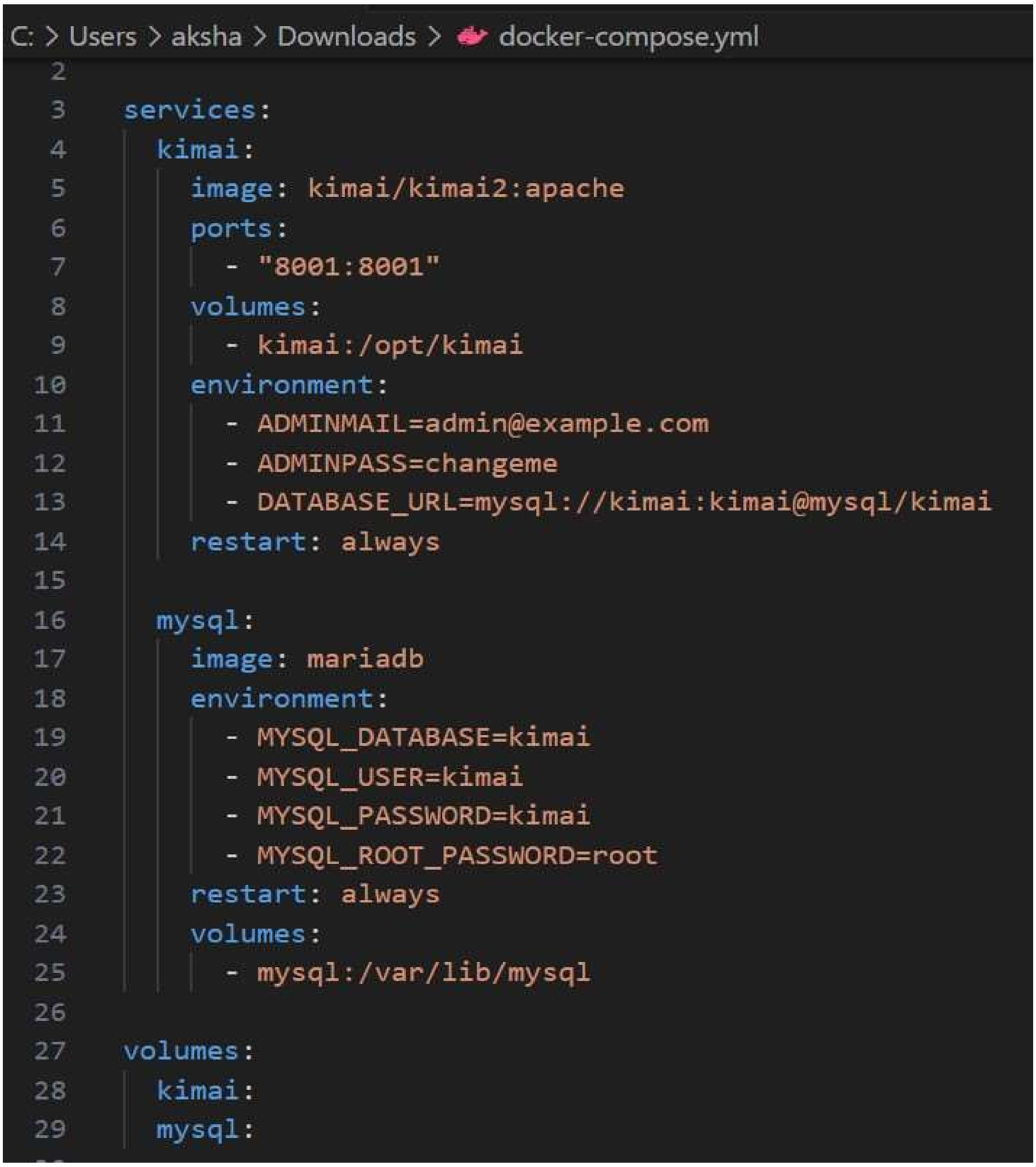
- The Kimai application and Jenkins were containerized using Docker to ensure platform independence.
- Docker Compose was configured to run both containers together, define ports, volumes, and environment variables.
- Services were designed to restart on failure and maintain persistent data using mounted volumes.



```
Microsoft Windows [Version 10.0.22631.5335]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aksha>cd "C:\Users\aksha\Downloads\Akshaya-EC2 (1).pem"
The directory name is invalid.

C:\Users\aksha>ssh -i "C:\Users\aksha\Downloads\key\Akshaya-EC2.pem" ec2-user@ec2-18-217-175-115.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-18-217-175-115.us-east-2.compute.amazonaws.com (18.217.175.115)' can't be established.
ED25519 key fingerprint is SHA256:BMBPiKZWwwOgywefG1AOfDS8Y64NzBKOVe30JUQN8aM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-217-175-115.us-east-2.compute.amazonaws.com' (ED25519) to the list of known hosts.

A newer release of "Amazon Linux" is available.
  Version 2023.7.20250623:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-1-78 ~]$ ls
jenkins-password.txt  kimai-app
[ec2-user@ip-10-0-1-78 ~]$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS                     PORTS
                        NAMES
cef00599bd03   kimai/kimai2:apache   "docker-php-entrypoi…"   23 hours ago   Up About an hour (healthy)   80/tcp, 0.0.0.0:8001->8001/
tcp, :::8001->8001/tcp   kimai-app-kimai-1
```



```
[ec2-user@ip-10-0-1-78 ~]$ docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED         STATUS                     PORTS
                                                                   NAMES
032ebb21f78f   jenkins/jenkins:lts   "/usr/bin/tini -- /u…"   5 seconds ago   Up 4 seconds               0.0.0.0:8080->8080/tcp, ::
:8080->8080/tcp, 0.0.0.0:50000->50000/tcp, :::50000->50000/tcp     jenkins
cef00599bd03   kimai/kimai2:apache   "docker-php-entrypoi…"   23 hours ago    Up About an hour (healthy)   80/tcp, 0.0.0.0:8001->8001
/tcp, :::8001->8001/tcp                                            kimai-app-kimai-1
21dd706b7e6f   mariadb               "docker-entrypoint.s…"   23 hours ago    Up About an hour           3306/tcp
                                                                   kimai-app-mysql-1
```

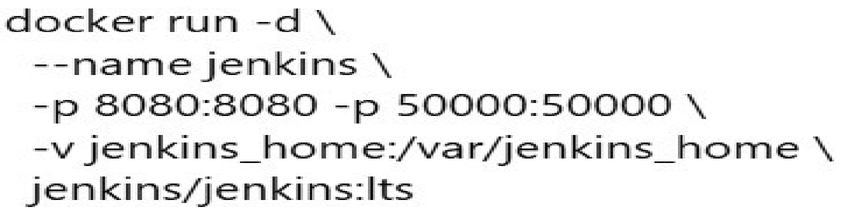


C: > Users > aksha > Downloads > docker-compose.yml

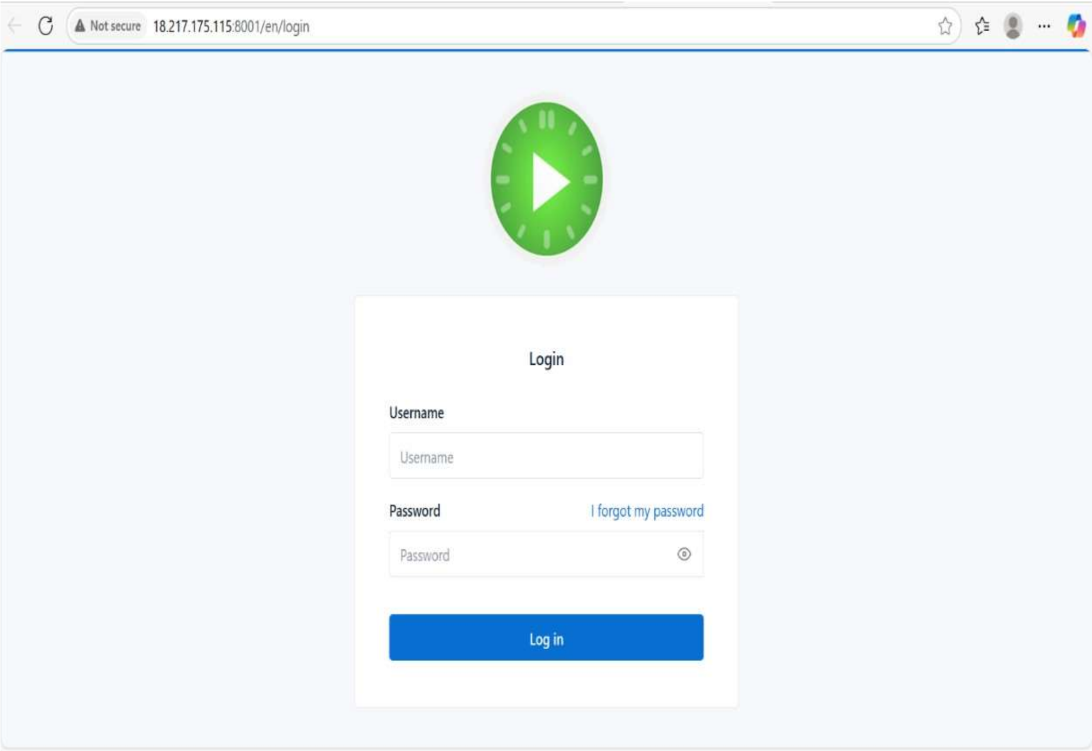
```yaml

services:
  kimai:
    image: kimai/kimai2:apache
    ports:
      - "8001:8001"
    volumes:
      - kimai:/opt/kimai
    environment:
      - ADMINMAIL=admin@example.com
      - ADMINPASS=changeme
      - DATABASE_URL=mysql://kimai:kimai@mysql/kimai
    restart: always

  mysql:
    image: mariadb
    environment:
      - MYSQL_DATABASE=kimai
      - MYSQL_USER=kimai
      - MYSQL_PASSWORD=kimai
      - MYSQL_ROOT_PASSWORD=root
    restart: always
    volumes:
      - mysql:/var/lib/mysql

volumes:
  kimai:
  mysql:
```

3.2 CI/CD Pipeline

- Jenkins was installed in a Docker container and configured for GitHub integration.
- A Jenkinsfile was created to automate steps: clone repo → build → deploy container.
- Push triggers automated container rebuild and redeployment of the updated Kimai application.



```
docker run -d \
  --name jenkins \
  -p 8080:8080 -p 50000:50000 \
  -v jenkins_home:/var/jenkins_home \
  jenkins/jenkins:lts
```



Login

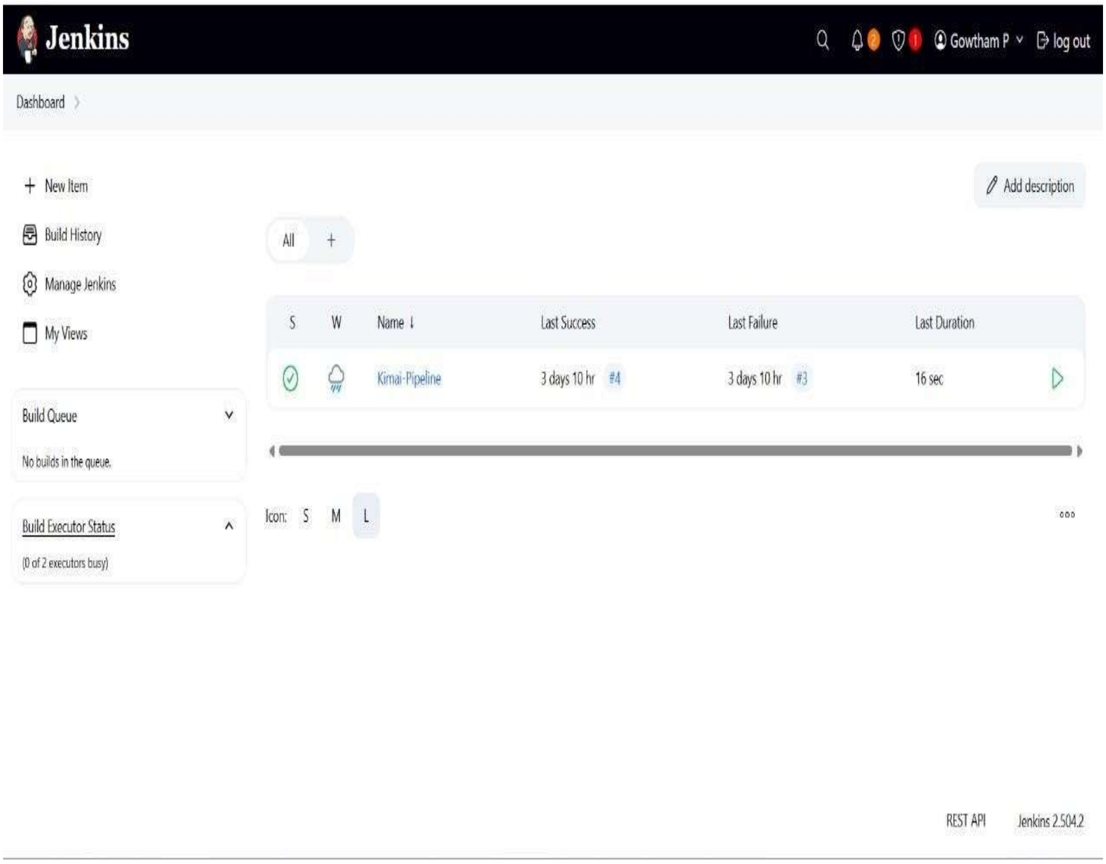
Username

Username

Password I forgot my password

Password

Log in



Build Queue

No builds in the queue.

Build Executor Status

(0 of 2 executors busy)

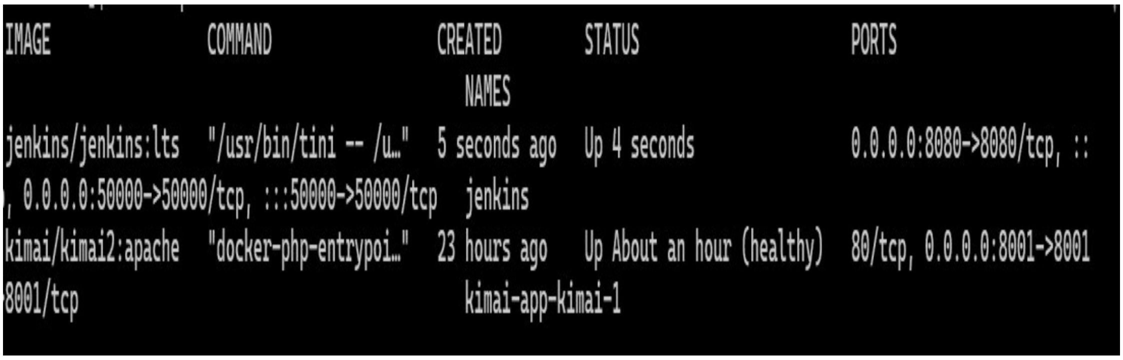
S	W	Name ↓	Last Success	Last Failure	Last Duration
✓		Kimai-Pipeline	3 days 10 hr #4	3 days 10 hr #3	16 sec

Phase 4: Security

To protect the system and data through hardening, restricted access, and least privilege principles.

4.1 Network Hardening

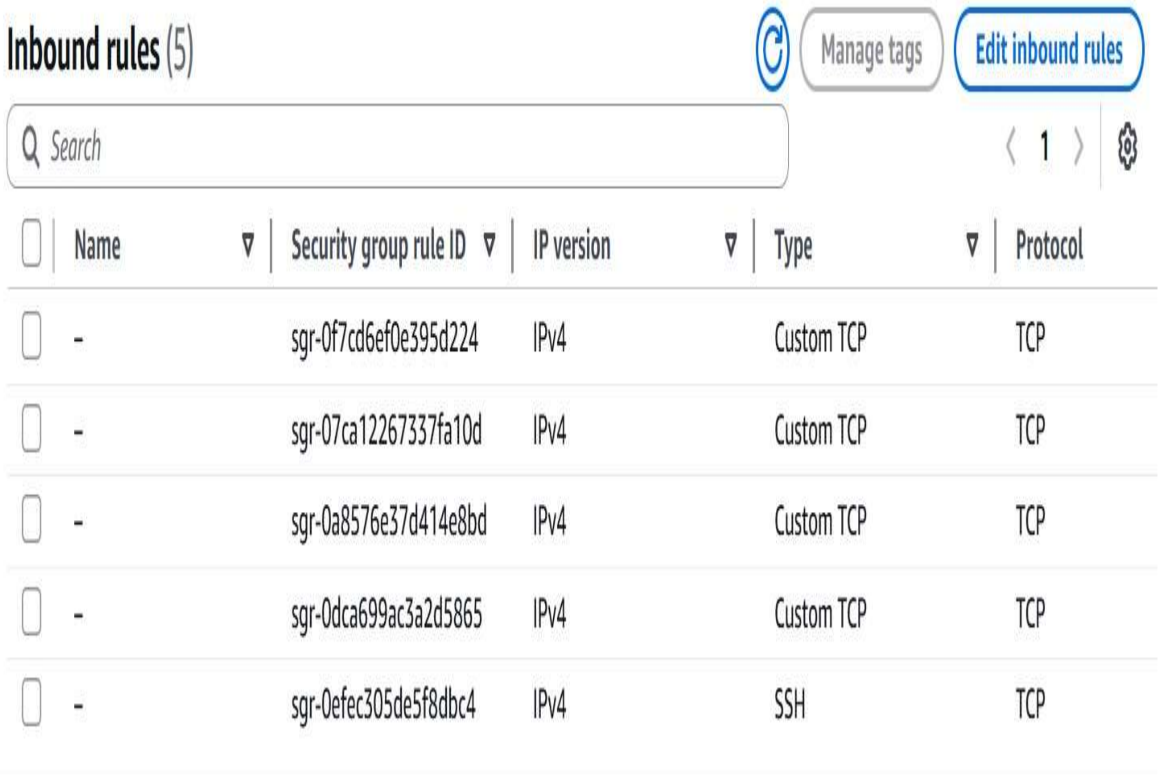
- Only essential ports (e.g., 22 for SSH, 8080 for Jenkins, 8001 for Kimai) were allowed.
- All other ports were blocked using firewall rules.
- Internal-only services were restricted to local access or through tunneling.



```
IMAGE                COMMAND                  CREATED        STATUS                     PORTS
                                                 NAMES
jenkins/jenkins:lts   "/usr/bin/tini -- /u…"   5 seconds ago   Up 4 seconds               0.0.0.0:8080->8080/tcp, ::
, 0.0.0.0:50000->50000/tcp, :::50000->50000/tcp   jenkins
kimai/kimai2:apache   "docker-php-entrypoi…"   23 hours ago    Up About an hour (healthy)  80/tcp, 0.0.0.0:8001->8001
8001/tcp                                         kimai-app-kimai-1
```

4.2 Secure Remote Access

- Access to the AWS EC2 instance was restricted using key pair-based SSH authentication.
- Password logins were disabled.
- Only trusted IPs were allowed to initiate remote sessions.

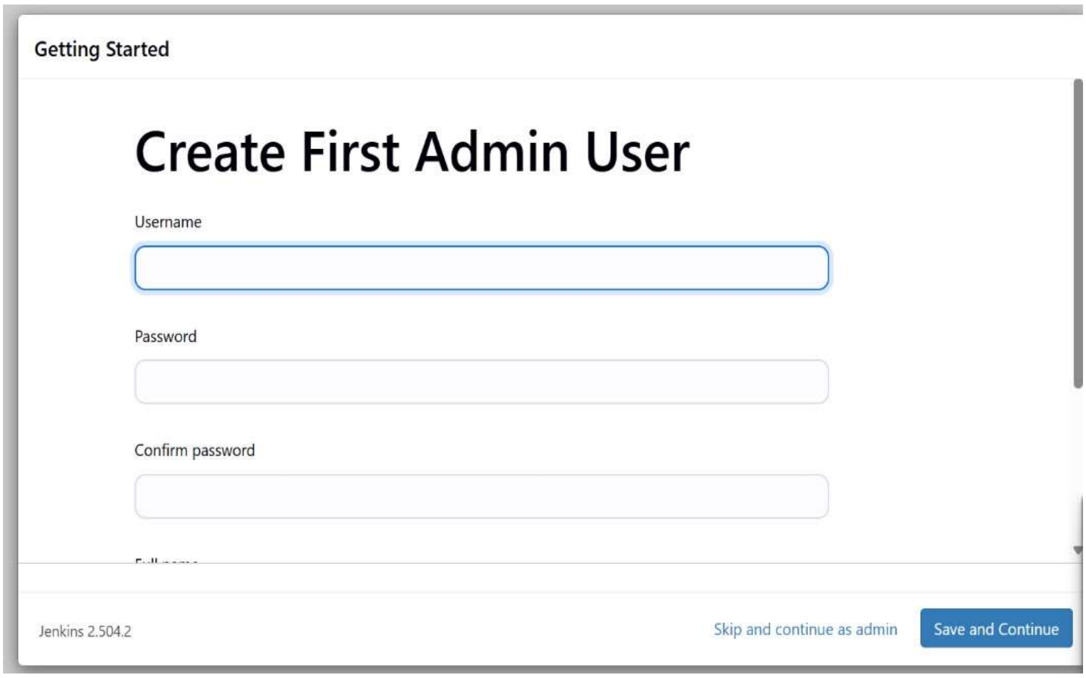


Inbound rules (5)

□	Name	Security group rule ID	IP version	Type	Protocol
□	-	sgr-0f7cd6ef0e395d224	IPv4	Custom TCP	TCP
□	-	sgr-07ca12267337fa10d	IPv4	Custom TCP	TCP
□	-	sgr-0a8576e37d414e8bd	IPv4	Custom TCP	TCP
□	-	sgr-0dca699ac3a2d5865	IPv4	Custom TCP	TCP
□	-	sgr-0efec305de5f8dbc4	IPv4	SSH	TCP

4.3 IAM Implementation

- IAM roles and instance profiles were configured to control access to AWS services securely.
- Local user roles (admin, deployer) were managed at the OS level inside the EC2 instance.
- Access rights were granted only as required, following the principle of least privilege.
- Credentials and environment secrets were not hardcoded but passed securely.



Getting Started

Create First Admin User

Username

Password

Confirm password

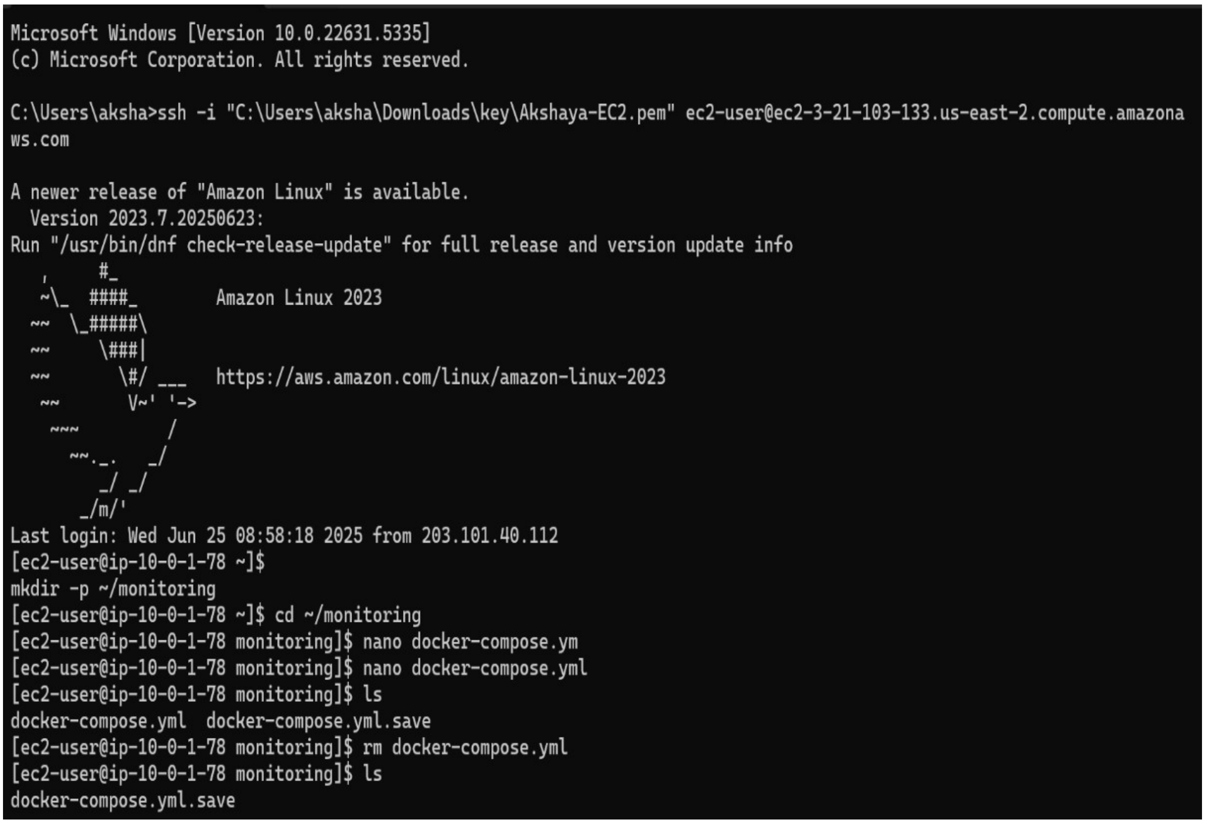
Jenkins 2.504.2

Phase 5: Monitoring & Logging

To gain visibility into system performance and application behaviour through logs, metrics, and alerts.

5.1 Centralized Logging

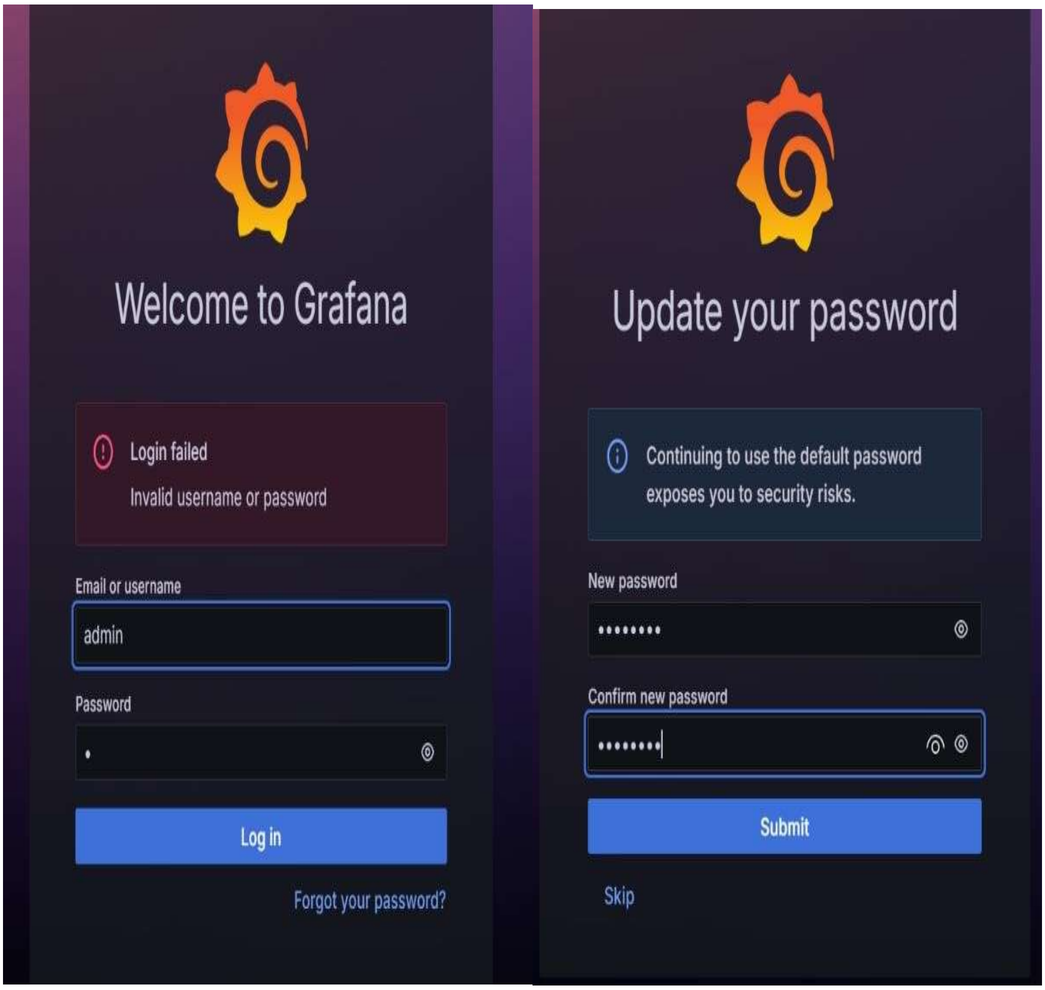
- Docker logs were captured and used to track container behavior and issues.
- Application-level logs were configured for persistent storage and review.



```
Microsoft Windows [Version 10.0.22631.5335]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aksha>ssh -i "C:\Users\aksha\Downloads\key\Akshaya-EC2.pem" ec2-user@ec2-3-21-103-133.us-east-2.compute.amazona
ws.com

A newer release of "Amazon Linux" is available.
  Version 2023.7.20250623:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Jun 25 08:58:18 2025 from 203.101.40.112
[ec2-user@ip-10-0-1-78 ~]$
mkdir -p ~/monitoring
[ec2-user@ip-10-0-1-78 ~]$ cd ~/monitoring
[ec2-user@ip-10-0-1-78 monitoring]$ nano docker-compose.ym
[ec2-user@ip-10-0-1-78 monitoring]$ nano docker-compose.yml
[ec2-user@ip-10-0-1-78 monitoring]$ ls
docker-compose.yml  docker-compose.yml.save
[ec2-user@ip-10-0-1-78 monitoring]$ rm docker-compose.yml
[ec2-user@ip-10-0-1-78 monitoring]$ ls
docker-compose.yml.save
```

5.2 Performance Monitoring

- Prometheus was installed to collect system metrics like CPU, memory, and disk usage.
- Node Exporter was used to export hardware metrics from the host machine.
- Grafana dashboards were built to visualize the real-time health of the system.



Welcome to Grafana

Login failed

Invalid username or password

Email or username

admin

Password

•

Update your password

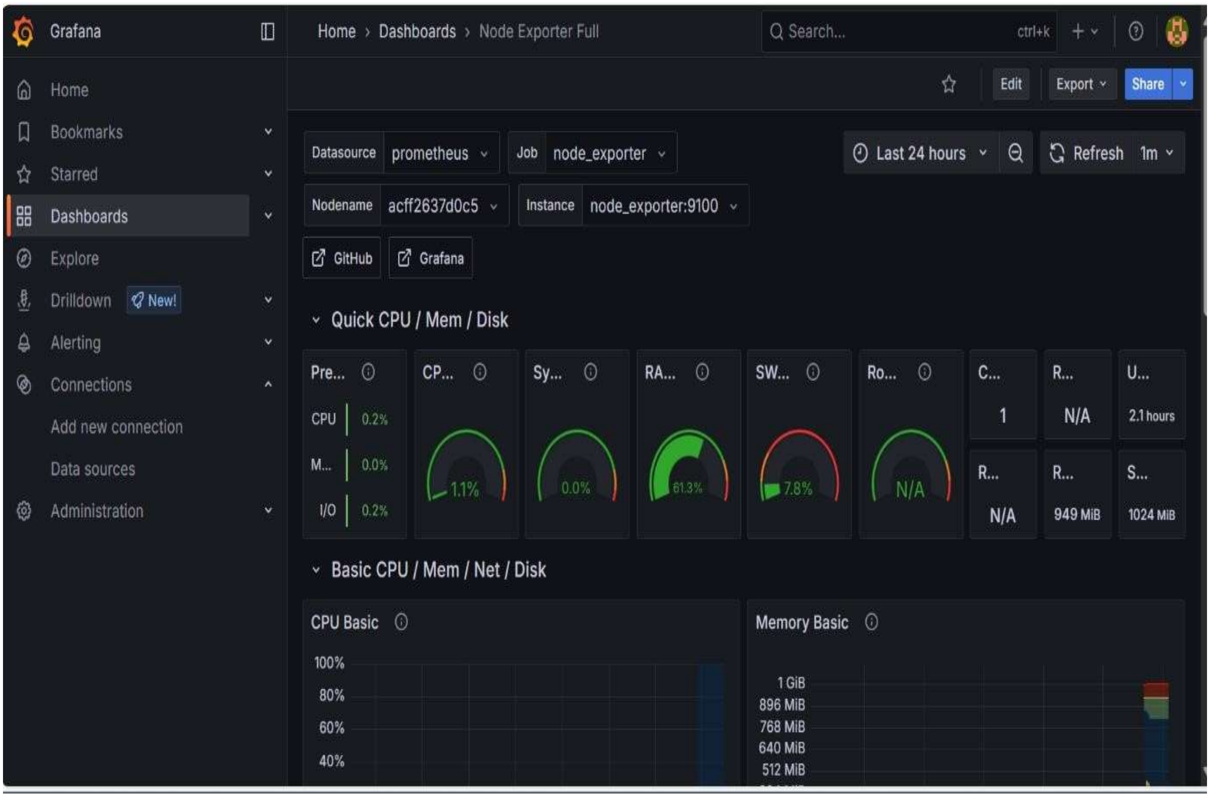
Continuing to use the default password exposes you to security risks.

New password

••••••••

Confirm new password

••••••••



Datasource prometheus

Job node_exporter

Nodename acff2637d0c5

Instance node_exporter:9100

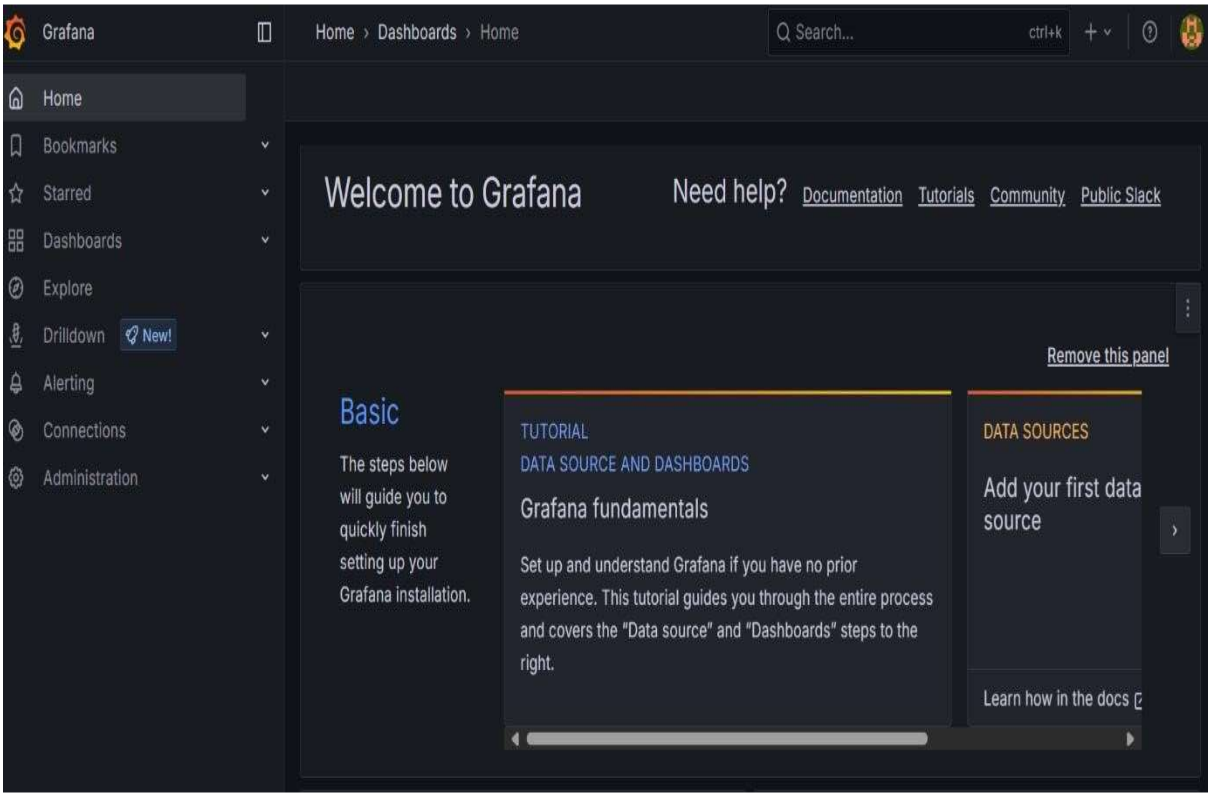
Last 24 hours

Refresh 1m

Quick CPU / Mem / Disk

Pre...	CP...	Sy...	RA...	SW...	Ro...	C...	R...	U...
CPU 0.2% M... 0.0% I/O 0.2%	1.1%	0.0%	61.3%	7.8%	N/A	1	N/A	2.1 hours
						R... N/A	R... 949 MiB	S... 1024 MiB

Basic CPU / Mem / Net / Disk



Welcome to Grafana

Need help? Documentation Tutorials Community Public Slack

Remove this panel

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

TUTORIAL

DATA SOURCE AND DASHBOARDS

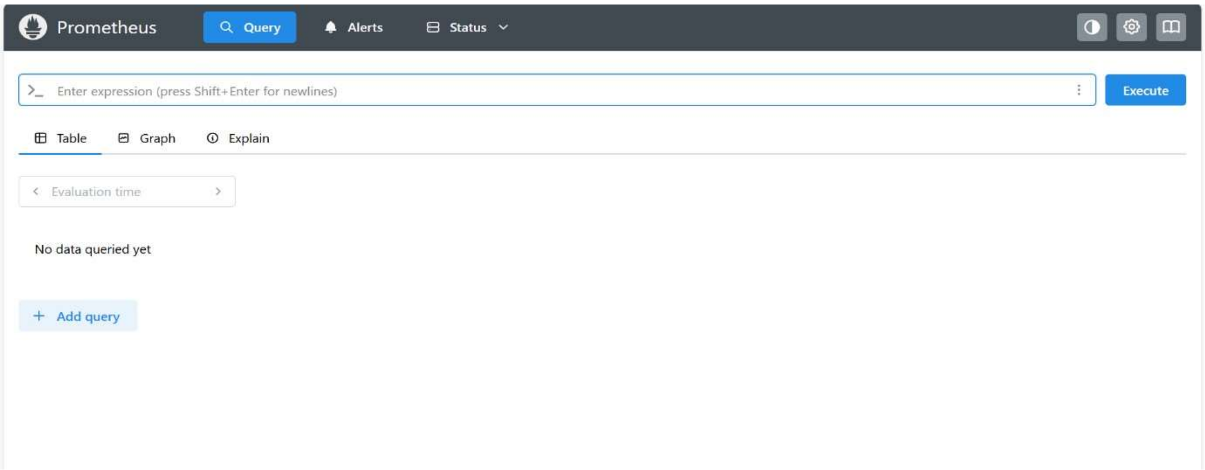
Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

DATA SOURCES

Add your first data source

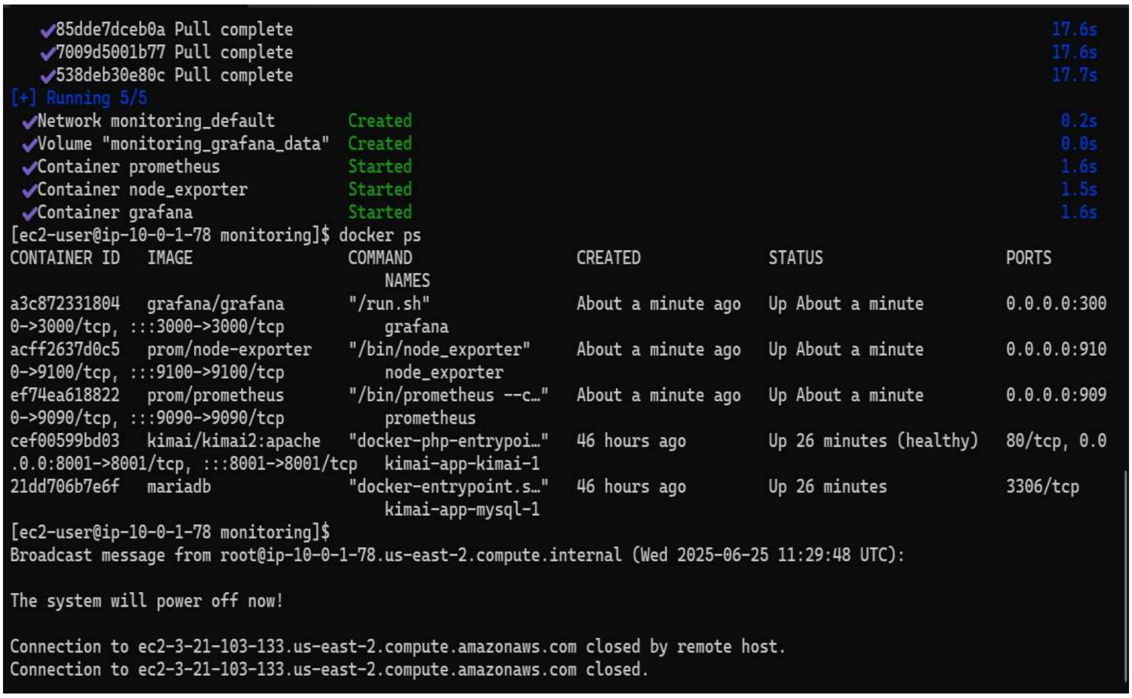
Learn how in the docs



No data queried yet

5.3 Alerting

- Prometheus alert rules were written using PromQL (e.g., trigger alert if CPU > 80% for 5 mins).
- Alerts provided early warnings for potential system overload or failures.



```
   ✓85dde7dceb0a Pull complete                                                                                     17.6s
   ✓7009d5001b77 Pull complete                                                                                     17.6s
   ✓538deb30e80c Pull complete                                                                                     17.7s
[+] Running 5/5
 ✓Network monitoring_default         Created                                                                        0.2s
 ✓Volume "monitoring_grafana_data"   Created                                                                        0.0s
 ✓Container prometheus               Started                                                                        1.6s
 ✓Container node_exporter            Started                                                                        1.5s
 ✓Container grafana                  Started                                                                        1.6s
[ec2-user@ip-10-0-1-78 monitoring]$ docker ps
CONTAINER ID   IMAGE                 COMMAND                   CREATED              STATUS                    PORTS
                                         NAMES
a3c872331804   grafana/grafana       "/run.sh"                 About a minute ago   Up About a minute         0.0.0.0:300
0->3000/tcp, :::3000->3000/tcp           grafana
acff2637d0c5   prom/node-exporter    "/bin/node_exporter"      About a minute ago   Up About a minute         0.0.0.0:910
0->9100/tcp, :::9100->9100/tcp           node_exporter
ef74ea618822   prom/prometheus       "/bin/prometheus --c…"    About a minute ago   Up About a minute         0.0.0.0:909
0->9090/tcp, :::9090->9090/tcp           prometheus
cef00599bd03   kimai/kimai2:apache   "docker-php-entrypoi…"    46 hours ago         Up 26 minutes (healthy)   80/tcp, 0.0
.0.0:8001->8001/tcp, :::8001->8001/tcp   kimai-app-kimai-1
21dd706b7e6f   mariadb               "docker-entrypoint.s…"    46 hours ago         Up 26 minutes             3306/tcp
                                         kimai-app-mysql-1
[ec2-user@ip-10-0-1-78 monitoring]$
Broadcast message from root@ip-10-0-1-78.us-east-2.compute.internal (Wed 2025-06-25 11:29:48 UTC):

The system will power off now!

Connection to ec2-3-21-103-133.us-east-2.compute.amazonaws.com closed by remote host.
Connection to ec2-3-21-103-133.us-east-2.compute.amazonaws.com closed.
```

Phase 6: Assessment

To evaluate the deployment quality, identify optimization opportunities, and ensure that the solution aligns with best practices.

6.1 Best Practice Assessment

- Reviewed the infrastructure, container configurations, and pipeline logic against DevOps best practices.
- Ensured modular Terraform code, efficient CI/CD, proper volume handling, and secure access.

6.2 Cost Assessment

- Evaluated resource usage to ensure the solution remains lightweight.
- Unused services and over-provisioned resources were removed to save cost.
- Verified that container behavior (e.g., restart policies, log rotation) optimized compute and storage usage.

6. Challenges Faced

- Errors occurred when switching AWS regions due to region-specific AMIs and configurations
- Certain services were not Free Tier eligible in all regions, requiring adjustments
- Resource limits on t2.micro impacted performance with multiple containers
- Debugging issues in user data script required log analysis and script correction

7. Key Learnings

- Gained hands-on experience in Terraform, Jenkins, Docker, and Linux server management
- Built a fully automated deployment pipeline with CI/CD practices
- Improved understanding of container orchestration and configuration
- Strengthened skills in troubleshooting and securing containerized environments

8. Limitations & Future Improvements

- Could not scale to multi-node due to Free Tier
- No load balancer or HTTPS in current setup
- WAF or Cloud Monitoring tools could be added in future

9. Future Scope

While the current setup works for a single-instance environment, the following improvements could enhance scalability and reliability:

- Introduce HTTPS and reverse proxy using Nginx
- Add a load balancer and auto-scaling group
- Implement log aggregation and centralized monitoring
- Expand CI/CD to include testing and rollback steps

10. Conclusion

This internship provided practical experience in deploying and managing applications using Infrastructure as Code and DevOps practices. I successfully deployed the Kimai timesheet application in a virtual environment, using automated tools and lightweight infrastructure, gaining strong technical and architectural insights.

Submitted by :

Akshaya R

B.E.CSE

(Artificial Intelligence & Machine Learning)