

Final Report

Title: Cloud Migration & Deployment of Kimai Timesheet Application

Organization: TechForce Services

Intern: Afshana Fathima A

Duration: 15 Days (Onsite)

Location: Pallikaranai, Chennai.

1. Introduction

In the course of my internship with TechForce 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

- 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
- Explore service scaling options within the limitations of Free Tier
- Document infrastructure diagrams (HLD & LLD) and provide setup instructions for replication

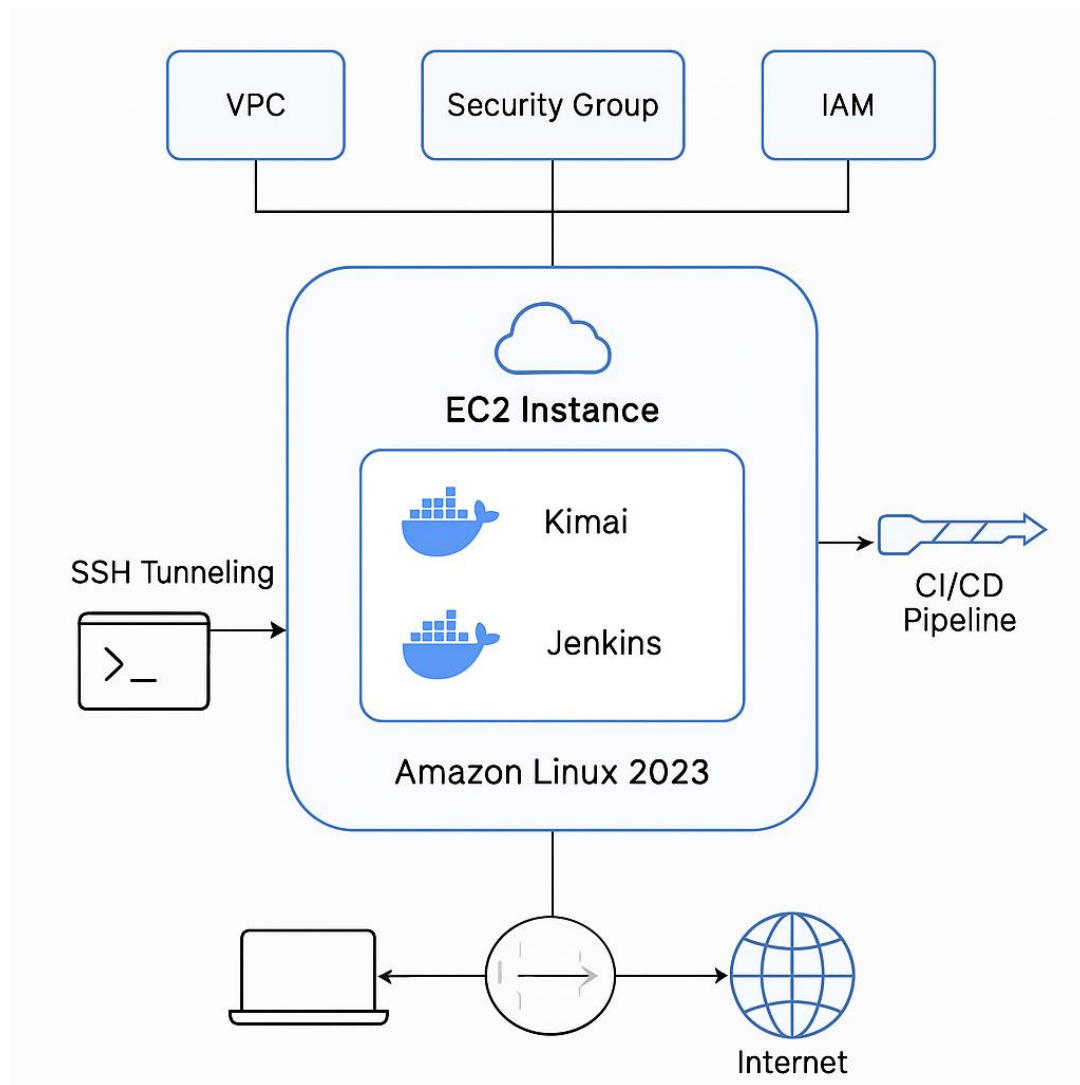
3. Tools & Technologies

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

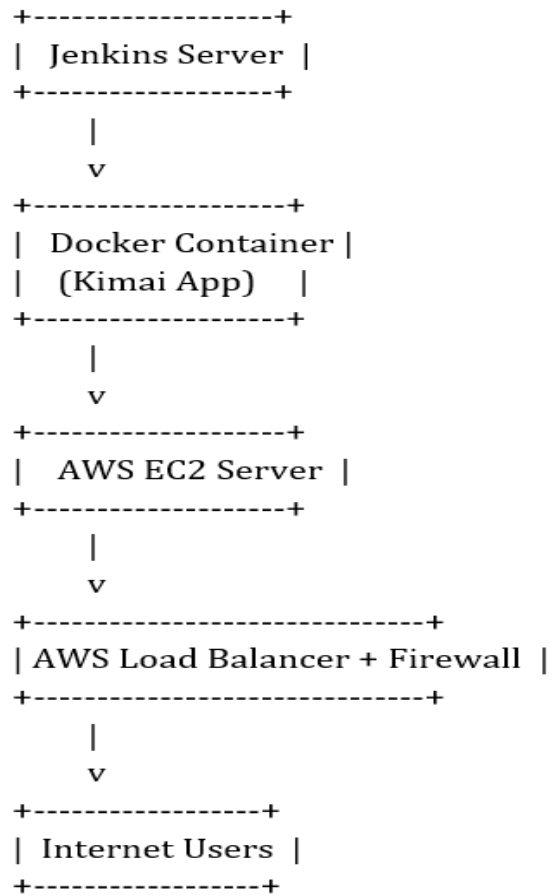


5.Implementation changes

Phase-1 : Design & Planning

A High-Level Design (HLD) and Low-Level Design (LLD) were created to outline the architecture, components, and workflows for deploying the Kimai time tracking application. The design emphasized simplicity and efficiency, ensuring it could operate

within limited infrastructure resources without compromising on scalability.



kimai-deployment/

|—— docker-compose.yml

|—— .env

|—— jenkins-pipeline.groovy

|—— terraform/

| |—— main.tf

| |—— variables.tf

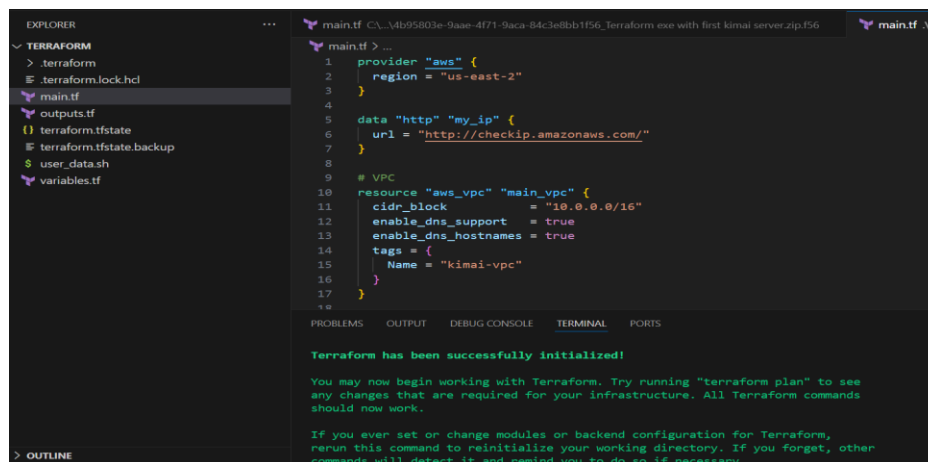
| |—— outputs.tf

|—— monitoring/

|—— grafana-dashboards/

Phase-2 : Provisioning

Infrastructure provisioning was handled using Terraform. Virtual machines were defined as code, allowing repeatable and automated setup. Key configurations included VM instance types, OS image selection, SSH access provisioning, and network rules. This streamlined the process of creating a ready-to-use environment for container deployment.

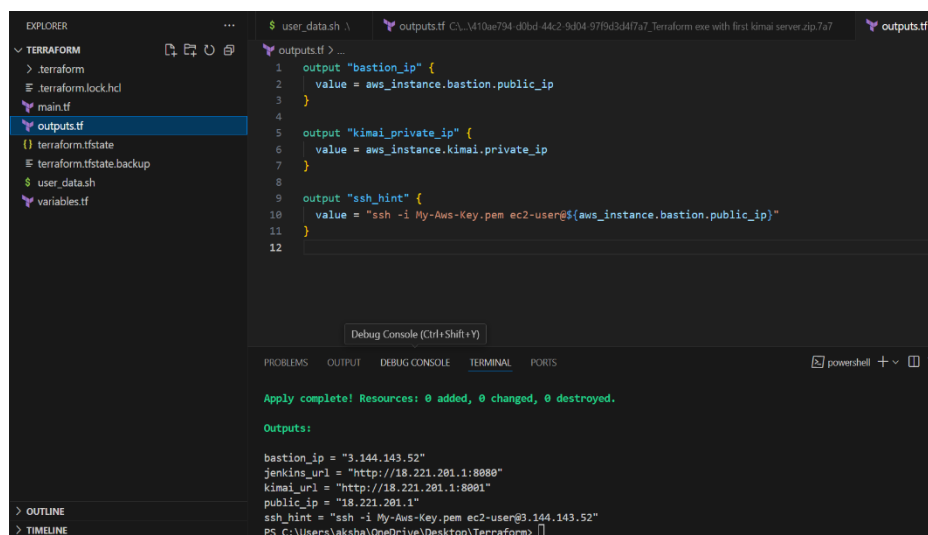


```
main.tf > ...
1 provider "aws" {
2   region = "us-east-2"
3 }
4
5 data "http" "my_ip" {
6   url = "http://checkip.amazonaws.com/"
7 }
8
9 # VPC
10 resource "aws_vpc" "main_vpc" {
11   cidr_block      = "10.0.0.0/16"
12   enable_dns_support = true
13   enable_dns_hostnames = true
14   tags = {
15     Name = "kimai-vpc"
16   }
17 }
18
```

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.



```
outputs.tf > ...
1 output "bastion_ip" {
2   value = aws_instance.bastion.public_ip
3 }
4
5 output "kimai_private_ip" {
6   value = aws_instance.kimai.private_ip
7 }
8
9 output "ssh_hint" {
10  value = "ssh -i My-Aws-Key.pem ec2-user@${aws_instance.bastion.public_ip}"
11 }
12
```

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://18.221.201.1:8081"
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

Once provisioned, Docker was installed on the VM to support containerized applications. The Kimai application and Jenkins were deployed using a `docker-compose.yml` file, ensuring modular, isolated environments. This method simplified deployment and allowed easy updates or rollbacks by modifying container configurations.

```
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	NAMES	COMMAND	CREATED	STATUS	PORTS
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 NAMES	STATUS	PORTS
032ebb21f78f	jenkins/jenkins:ls	"/usr/bin/tini -- /u..."	5 seconds ago jenkins	Up 4 seconds	0.0.0.0:8080->8080/tcp, ::
8080->8080/tcp, 0.0.0.0:50000->50000/tcp, :::50000->50000/tcp					
cef00599bd03	kimai/kimai2:apache	"docker-php-entrypoi..."	23 hours ago kimai-app-kimai-1	Up About an hour (healthy)	80/tcp, 0.0.0.0:8001->8001
/tcp, :::8001->8001/tcp					
21dd706b7e6f	mariadb	"docker-entrypoint.s..."	23 hours ago kimai-app-mysql-1	Up About an hour	3306/tcp

← ↻ ⚠ Not secure 18.217.175.115:8001/en/login ☆ ⚙ ... 🌐



Login

Username

Password [I forgot my password](#)

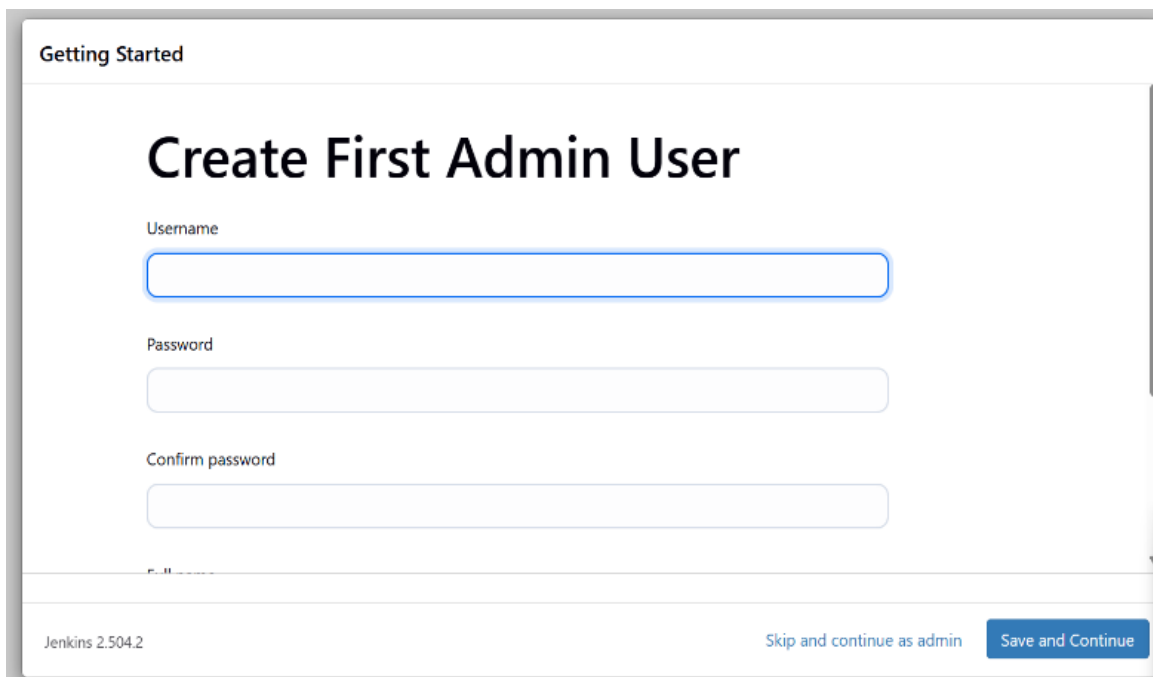
Phase-4: CI/CD Setup

Jenkins was containerized and deployed on the same host. A Jenkinsfile was configured to define an automated pipeline for Kimai. This pipeline handled pulling the latest code, building the container image, and redeploying the updated Kimai app. This setup ensured continuous integration and rapid, consistent deployments with minimal manual intervention.

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

Phase-5: Security

Security measures were put in place to protect the system from unauthorized access and vulnerabilities. These included configuring firewall rules to restrict incoming traffic to essential ports (e.g., 22, 8001, 8080), using SSH key-based authentication, running containers with limited privileges, and securing Jenkins with admin credentials and role-based access.



The image shows the 'Getting Started' screen of Jenkins 2.504.2. The main heading is 'Create First Admin User'. Below this, there are three input fields: 'Username', 'Password', and 'Confirm password'. The 'Username' field is currently active, indicated by a blue border. At the bottom of the form, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

Getting Started

Create First Admin User

Username

Password

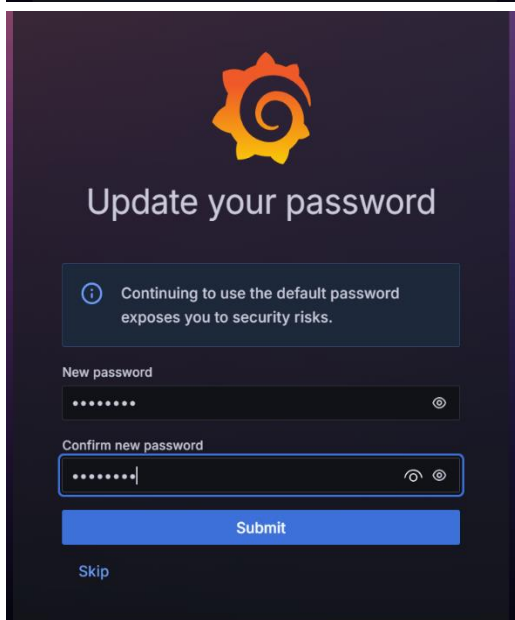
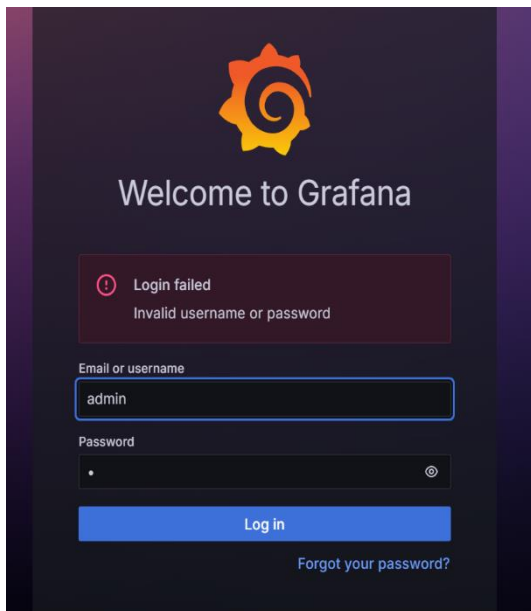
Confirm password

Jenkins 2.504.2

[Skip and continue as admin](#) [Save and Continue](#)

Phase-6: Monitoring

To ensure visibility into the health and performance of the deployed Kimai application and its host EC2 instance, monitoring was set up using Prometheus, Node Exporter, and Grafana.



Prometheus

Query

Alerts

Status

>_ Enter expression (press Shift+Enter for newlines)

Execute

Table

Graph

Explain

< Evaluation time >

No data queried yet

+ Add query

6. Challenges Faced

- Resource limitations on t2.micro instance affected multi-container performance
- Docker network conflicts resolved through correct port mapping and bridge settings
- SSH tunneling required precise port forwarding for secure development access

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. 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.