**Final Report**  
  
 **Title: Cloud Migration and Monitoring project**

**Organization: TechForce Services**

**Github:** [**https://github.com/Blackmoon390/IT\_infrastructure.git**](https://github.com/Blackmoon390/IT_infrastructure.git) **Intern: Vishnu  
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  
    
  A diagram of a computer network

  AI-generated content may be incorrect.

**5. Implementation Stages**

**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 .  
 A screenshot of a computer program

AI-generated content may be incorrect.   
  
   
  
A screenshot of a computer

AI-generated content may be incorrect.

**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.  
  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
  
A screenshot of a computer program

AI-generated content may be incorrect.

**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.  
  
A screenshot of a computer program

AI-generated content may be incorrect.  
  
  
A black screen with white text

AI-generated content may be incorrect.  
  
  
  
A screenshot of a computer

AI-generated content may be incorrect.

**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.  
  
  
A close-up of a code

AI-generated content may be incorrect.  
  
  
  
  
  
  
  
  
  
**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.  
  
A screenshot of a login page

AI-generated content may be incorrect.  
  
  
  
  
  
  
**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**.

A screenshot of a login screen

AI-generated content may be incorrect.A screenshot of a login screen

AI-generated content may be incorrect.  
  
A screenshot of a computer

AI-generated content may be incorrect.

**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.