**Low-Level Design (LLD)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Cloud Migration and Monitoring Project**

**Table of Contents**

1. Objective
2. Overview of Architecture
3. Infrastructure Breakdown
4. Terraform Infrastructure as Code
5. Docker Deployment
6. Jenkins CI/CD Pipeline
7. Monitoring & Alerting
8. Logging & Observability
9. Security & IAM Roles
10. SSH Tunneling for Access
11. Folder Structure

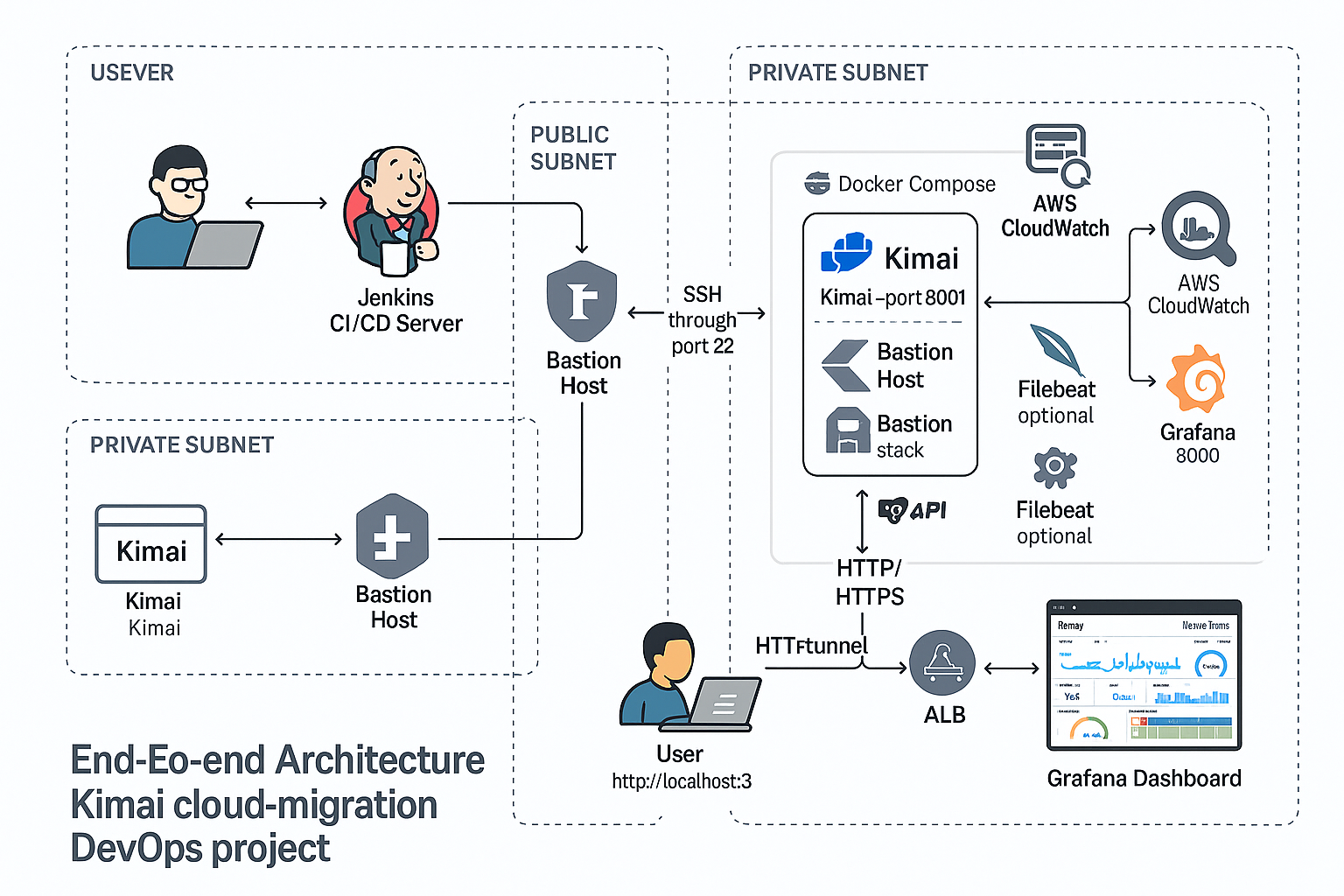
**1. Objective**

The objective of this project is to migrate the open-source **Kimai timesheet application** to the **AWS cloud**. It includes automated infrastructure provisioning with **Terraform**, containerized deployment using **Docker**, secure access via a **Bastion Host**, monitoring with **Prometheus & Grafana**, and logging with **Amazon CloudWatch**.

**2. Overview of Architecture**

**Components:**

* **VPC** with Public and Private Subnets
* **EC2 Bastion Host** (Public Subnet)
* **EC2** **App Server** for Kimai, Jenkins, Prometheus, Grafana (Private Subnet)
* **IAM Role** for CloudWatch Logs
* **Security Groups** for controlled access
* **Docker Compose** deployment of Kimai and supporting tools
* **Prometheus & Grafana** for monitoring
* CloudWatch Agent for system log collection

****

**3. Infrastructure Components**

| **Component** | **Type** | **Configuration/Tool** | **Notes** |
| --- | --- | --- | --- |
| **VPC** | AWS VPC | CIDR: 10.0.0.0/16 | Isolated virtual network |
| **Subnets** | Public & Private | /24 each | Public: Bastion, Private: Kimai |
| **EC2 (Bastion)** | t2.micro | Amazon Linux 2023 | SSH Access point |
| **EC2 (Kimai)** | t2.micro | Amazon Linux 2023 | Kimai App + Jenkins + Monitoring |
| **Security Groups** | AWS SG | Port-based access rules | SSH, 8001, 3000, 9090, 9100 |
| **IAM Role** | EC2 IAM Role | CloudWatchAgent access | Logs pushed securely |

**Infrastructure Breakdown**

**VPC**

* CIDR: 10.0.0.0/16
* Enables isolation of private and public layers

**Subnets**

* Public Subnet: 10.0.1.0/24 (for Bastion)
* Private Subnet: 10.0.10.0/24 (for App EC2)

**EC2 Instances**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Instance** | **AMI** | **Type** | **Subnet** | **Role** |
| Bastion | Amazon Linux 2023 | t2.micro | Public | SSH Access Point |
| App Server | Amazon Linux 2023 | t2.micro | Private | App Hosting |

**Security Groups**

|  |  |  |
| --- | --- | --- |
| **SG Name** | **Allowed Ports** | **Source** |
| bastion-sg | 22 (SSH) | My IP |
| app-sg | 8001, 3000, 9090 | 10.0.1.0/24 |

**IAM Role**

* Attached to App Server
* Grants permission to publish logs to CloudWatch
* Uses Instance Profile

**4. Provisioning (Terraform)**

**Files:**

* main.tf: defines EC2s, VPC, SGs, Subnets
* variables.tf: defines all inputs (region, AMI ID, etc.)
* outputs.tf: prints IP addresses

The user\_data.sh is a type of script , it calls shellscript file

**user\_data.sh** installs:

* Docker, Docker Compose
* Git and Kimai repo
* Jenkins container

| **File** | **Description** |
| --- | --- |
| main.tf | AWS EC2, VPC, Subnet, SG, Role setup |
| variables.tf | Parameterization of key configs |
| outputs.tf | Displays IPs of EC2s |
| user\_data.sh | Installs Docker, Jenkins, Kimai, CloudWatch |

**Terraform Flow:**

terraform init

terraform apply

**🐳 4. Application Deployment (Docker Compose)**

| **Service** | **Image** | **Ports** | **Notes** |
| --- | --- | --- | --- |
| **Kimai** | kimai/kimai2:apache | 8001 | Web UI |
| **MySQL** | mariadb | 3306 | DB backend for Kimai |
| **Jenkins** | jenkins/jenkins:lts | 8080, 50000 | CI/CD pipeline execution |

Kimai docker-compose.yml sample:

version: '3.7'

services:

kimai:

image: kimai/kimai2:apache

ports:

- "8001:8001"

depends\_on:

- mysql

mysql:

image: mariadb

environment:

MYSQL\_DATABASE: kimai

**🔄 5. CI/CD Pipeline (Jenkins)**

| **Stage** | **Tool/Script** | **Purpose** |
| --- | --- | --- |
| Clone Repo | GitHub | Clones kimai-app repo |
| Deploy App | docker-compose | Spins up Kimai + MySQL |
| Verify | docker ps | Confirms container status |
| Output App URL | curl + echo | Displays access link in console |

**Pipeline file: Jenkinsfile**

**📈 6. Monitoring (Prometheus + Grafana)**

| **Component** | **Docker Image** | **Port** | **Purpose** |
| --- | --- | --- | --- |
| **Prometheus** | prom/prometheus | 9090 | Pulls metrics from exporters |
| **Grafana** | grafana/grafana | 3000 | Visual dashboards |
| **Node Exporter** | prom/node-exporter | 9100 | System-level metrics |

**Prometheus**

* Installed via Docker
* Scrapes metrics from:
  + Node Exporter (host metrics)
  + Docker (app health)

**Grafana**

* Connected to Prometheus
* Dashboard created for:
  + CPU Utilization
  + Memory Usage
  + Disk IO
  + Container Health

**Prometheus scrape config (prometheus.yml):**

scrape\_configs:

- job\_name: 'node-exporter'

static\_configs:

- targets: ['localhost:9100']

**🚨 7. Alerting Rules (Grafana)**

| **Alert Name** | **Expression (PromQL)** | **Trigger Threshold** |
| --- | --- | --- |
| CPU High | 100 - avg(irate(...)) \* 100 | CPU > 70% for 1 min |
| Disk Low | node\_filesystem\_avail\_bytes | Disk < 20% remaining |
| App Down | Optional (Blackbox or port ping) | If health check fails |

**📦 8. Logging (CloudWatch Agent)**

* Installed on EC2 via user\_data.sh
* Collects:
  + /var/log/messages
  + /var/log/secure
  + /var/log/cloud-init.log
* Logs are forwarded to **CloudWatch Logs**
* Configured in /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json

**🔐 9. Security & Access**

| **Layer** | **Config** |
| --- | --- |
| SSH Access | Allowed only from my IP via Bastion |
| Kimai EC2 | Private subnet, no public SSH |
| Ports | Only 8001, 3000, 9090, 9100 exposed via Bastion |
| IAM Role | Fine-grained CloudWatch access only |

**Security & IAM Roles**

SSH Bastion Access

* SSH is allowed only to Bastion from user IP
* Kimai EC2 has no public IP
* Private EC2 is accessible only via Bastion

**🧪 10. SSH Tunneling**

Used to access Grafana & Prometheus from local browser:

ssh -i My-Aws-Key.pem \

-L 3000:localhost:3000 \

-L 9090:localhost:9090 \

-L 8001:localhost:8001 \

ec2-user@<Bastion-IP>

* This avoids exposing these tools to the internet
* Works well under Free Tier with private EC2