

# Kimai Cloud Migration Project

## GitHub Repository

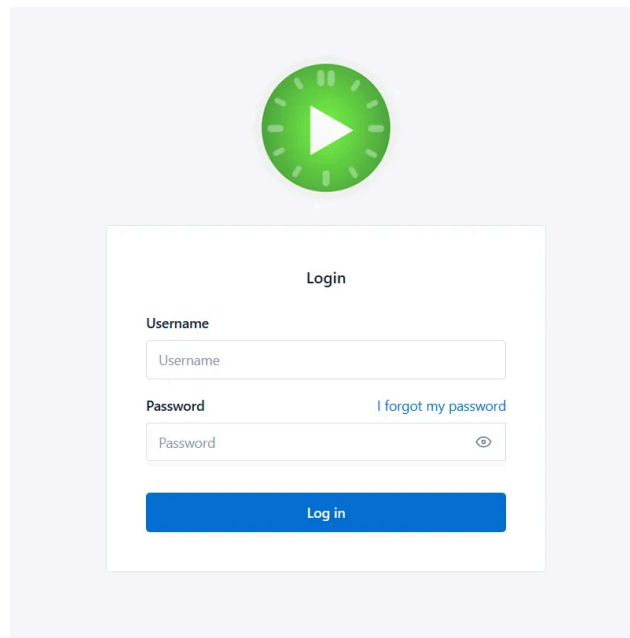
This project is maintained on GitHub:

**Repo Name:** `Cloud-Migration-Project`

**Link:** <https://github.com/angelinedev/Cloud-Migration-Project>

It contains:

- `terraform/` — Infrastructure-as-Code setup (modular with S3 backend)
- `kimai/` — Dockerfile and Docker Compose setup for Kimai
- `Jenkinsfile` — Jenkins Pipeline as Code
- `docs/` — High-Level Design (HLD) and Low-Level Design (LLD)
- `README.md` — Setup guide, architecture documentation, and cost estimation



<http://13.48.47.64:8001/>

<http://kimai-alb-653895671.eu-north-1.elb.amazonaws.com/en/login>

---

## Tech Stack

Component	Tech
Backend	PHP (Symfony Framework)
Frontend	HTML/CSS/JS (Bundled)
DB	MariaDB
Server	Nginx
Runtime	PHP-FPM
OS	Amazon Linux 2 (EC2)
IaC	Terraform
Containerization	Docker
CI/CD	Jenkins
Monitoring	CloudWatch, Grafana
Logging	CloudWatch Logs
Access Control	AWS IAM, Bastion Host

---

## Deployment Architecture

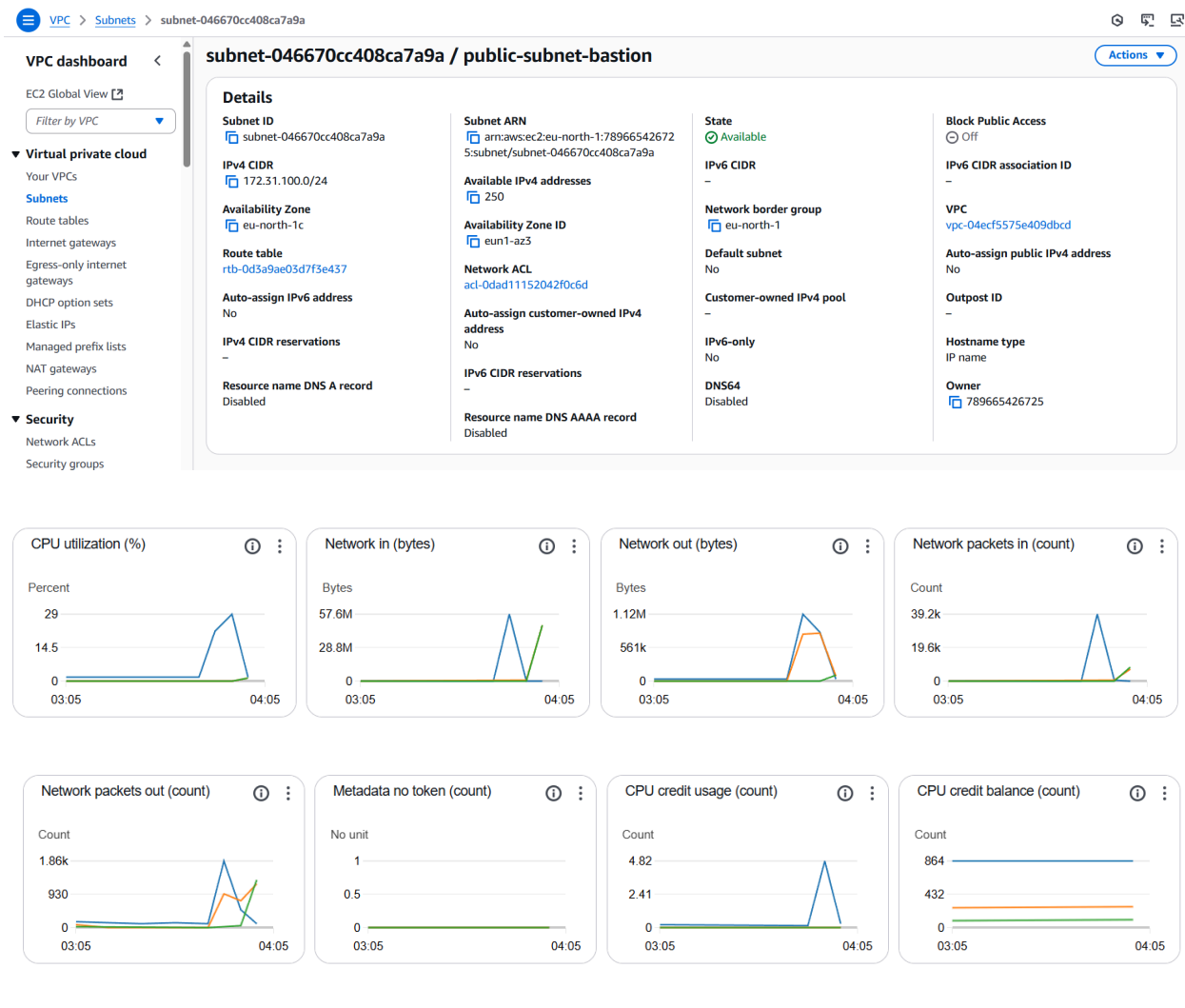
### Textual Architecture Diagram

```
[ User ]
|
[ Load Balancer (Public) + WAF ]
|
[ Bastion Host (Public Subnet) ]
|
[ VPC ]
```

- EC2 (Kimai + Docker) [Private Subnet]
- RDS (MySQL DB) [Private Subnet]

## Security Highlights:

- Bastion host for secure SSH
- IAM with least privilege
- Security Groups with only required ports open
- ALB protected by AWS WAF



# Repository Structure

```
Cloud-Migration-Project/
├─ terraform/
│   ├─ main.tf
│   ├─ variables.tf
│   ├─ outputs.tf
│   └─ modules/
│       └─ ec2/
│           ├─ main.tf
│           ├─ variables.tf
│           └─ outputs.tf
├─ Jenkinsfile
├─ kimai/
│   ├─ Dockerfile
│   └─ docker-compose.yml
├─ docs/
│   ├─ HLD.md
│   └─ LLD.md
└─ README.md
```

## Terraform Setup

All Terraform modules and configuration files are located in the `terraform/` directory inside the [GitHub repository](#).

- Modular setup for reusability
- Remote state managed in S3
- Uses t3.large for Kimai EC2 and t3.micro for Bastion Host
- Outputs public/private IPs, instance IDs

To deploy:

```
cd terraform
terraform init
terraform apply -auto-approve
```

```
[ec2-user@ip-172-31-12-187 Cloud-Migration-Project]$ cd terraform
[ec2-user@ip-172-31-12-187 terraform]$ ls
main.tf  modules  outputs.tf  provider.tf  variables.tf
[ec2-user@ip-172-31-12-187 terraform]$ terraform plan
module.kimai_ec2.aws_instance.kimai: Refreshing state... [id=i-0c00442d34128fbd3]
aws_s3_bucket.kimai_backup: Refreshing state... [id=kimai-backup-bucket-angel-69]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are
needed.

Warning: Argument is deprecated

    with aws_s3_bucket.kimai_backup,
    on main.tf line 1, in resource "aws_s3_bucket" "kimai_backup":
     1: resource "aws_s3_bucket" "kimai_backup" {

versioning is deprecated. Use the aws_s3_bucket_versioning resource instead.

[ec2-user@ip-172-31-12-187 terraform]$ terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.100.0

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.
[ec2-user@ip-172-31-12-187 terraform]$ |
```

## Docker Setup

Kimai is containerized with a multi-stage Dockerfile for optimized builds. The container is:

- Non-root
- Has healthcheck
- Ready for production

To build locally:

```
docker build -t kimai-app .
docker run -p 80:8001 kimai-app
```

```
[ec2-user@ip-172-31-12-187 Cloud-Migration-Project]$ cd kimai
[ec2-user@ip-172-31-12-187 kimai]$ ls
assets          composer.json  CONTINUING.md  eslint.config.mjs  kimai.sh  migrations  phpstan.neon  public  src  tests  UPGRADING-3.md  yarn.lock
bin             composer.lock  docker-compose.yml  HLD.pdf  LICENSE  package.json  phpstan.sh  README.md  symfony.lock  translations  UPGRADING.md
CHANGELOG.md   config        Dockerfile      index.php  LLD.pdf  php-cs-fixer.sh  phpunit.xml.dist  SECURITY.md  templates  UPGRADING-1.md  webpack.config.js
[ec2-user@ip-172-31-12-187 kimai]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
b95c26799339  kimai/kimai2:apache  "docker-php-entrypoi..."  34 hours ago  Up 34 hours  80/tcp, 0.0.0.0:8001->8001/tcp, :::8001->8001/tcp  kimai_app
3f8c687f4ce8  mysql:5.7       "docker-entrypoint.s..."  34 hours ago  Up 34 hours  3306/tcp, 33060/tcp      kimai_db
[ec2-user@ip-172-31-12-187 kimai]$
```

## CI/CD Pipeline (Jenkins)

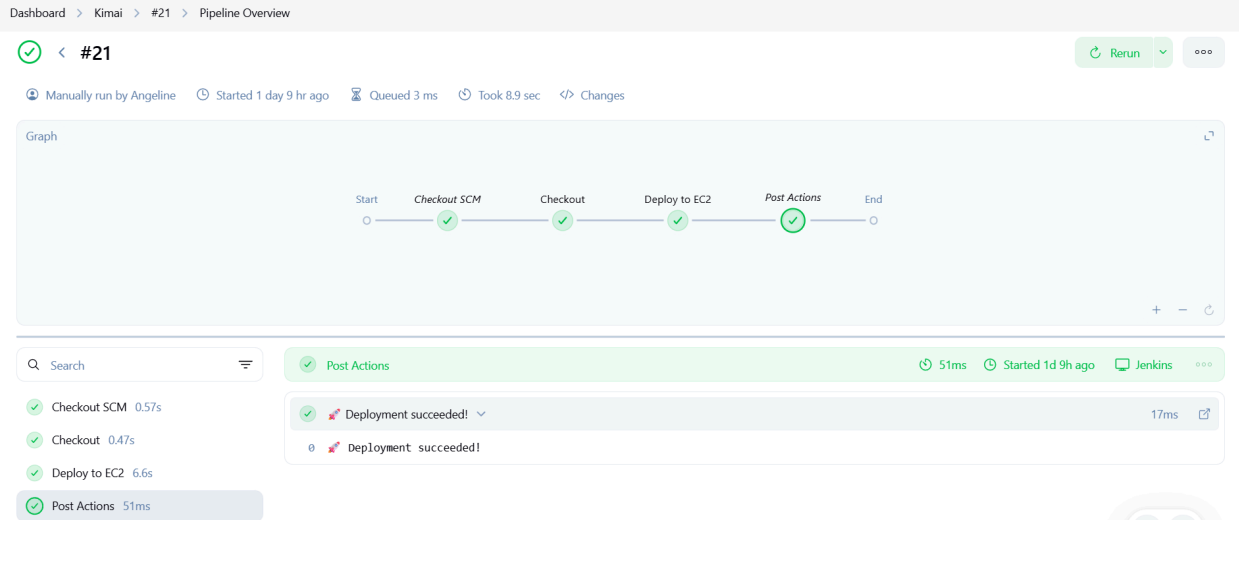
Pipeline is stored as `Jenkinsfile` in the [GitHub repository](#).

Trigger: Push to `main` branch

Stages:

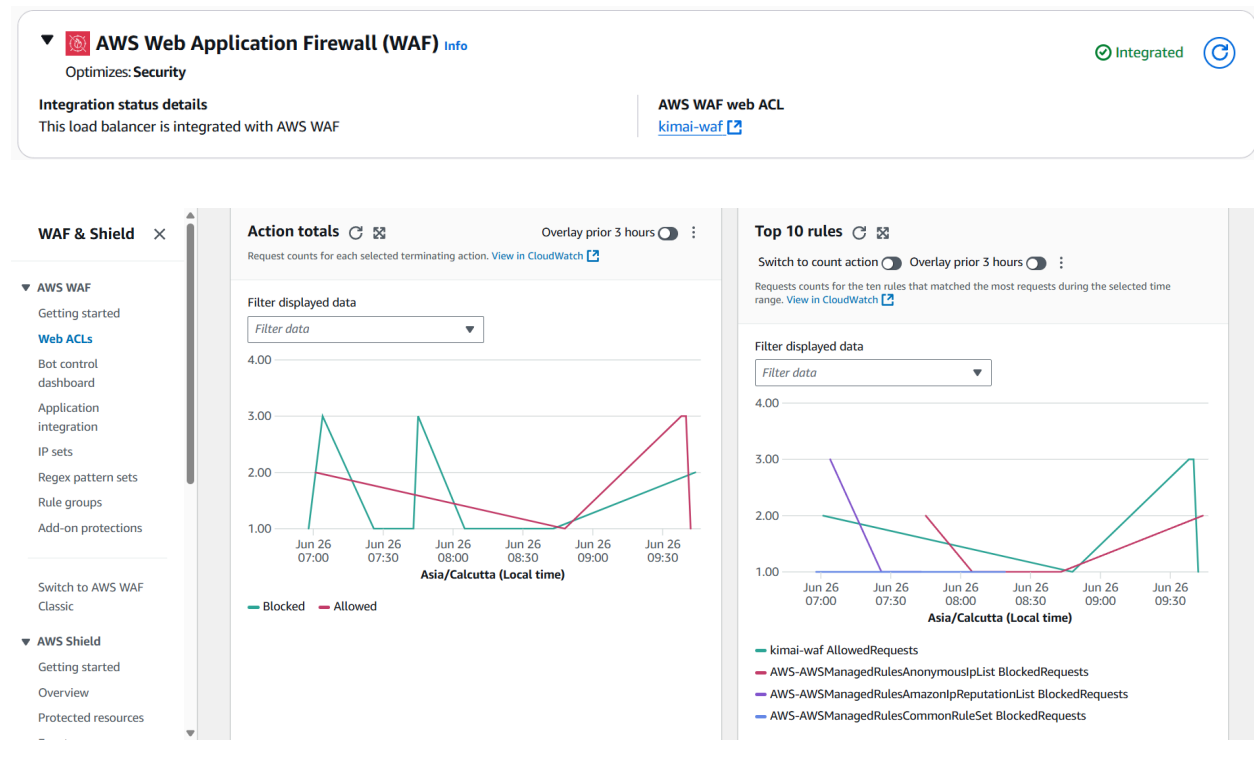
- Checkout
- Build Docker Image
- Run Tests
- Push to DockerHub
- Deploy via SSH to EC2

Pipeline is fully automated using a `Jenkinsfile`.



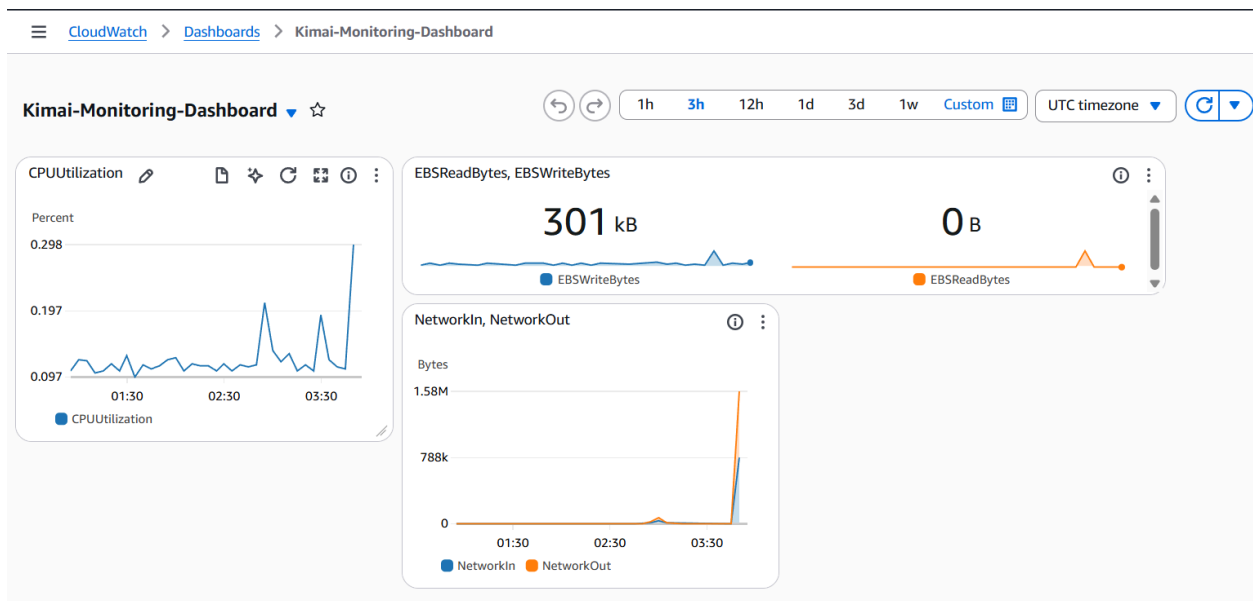
## Security

- WAF filters traffic on Load Balancer
- Bastion Host enables safe key-based access to private instances
- IAM Role attached to Kimai EC2 for CloudWatch agent and S3 access



# Monitoring & Logging

- CloudWatch Agent installed and configured on EC2
- Monitors:
  - CPU usage
  - Memory and Disk space
  - Application logs
- Alerts configured for:
  - High CPU (> 80%)
  - Health Check failures





The screenshot displays the AWS CloudWatch console. The top navigation bar shows the path: CloudWatch > Log groups > /aws/kimai-docker > kimai-app-log. The left sidebar contains navigation options like Dashboards, Alarms, Logs, and Metrics. The main content area is divided into two sections.

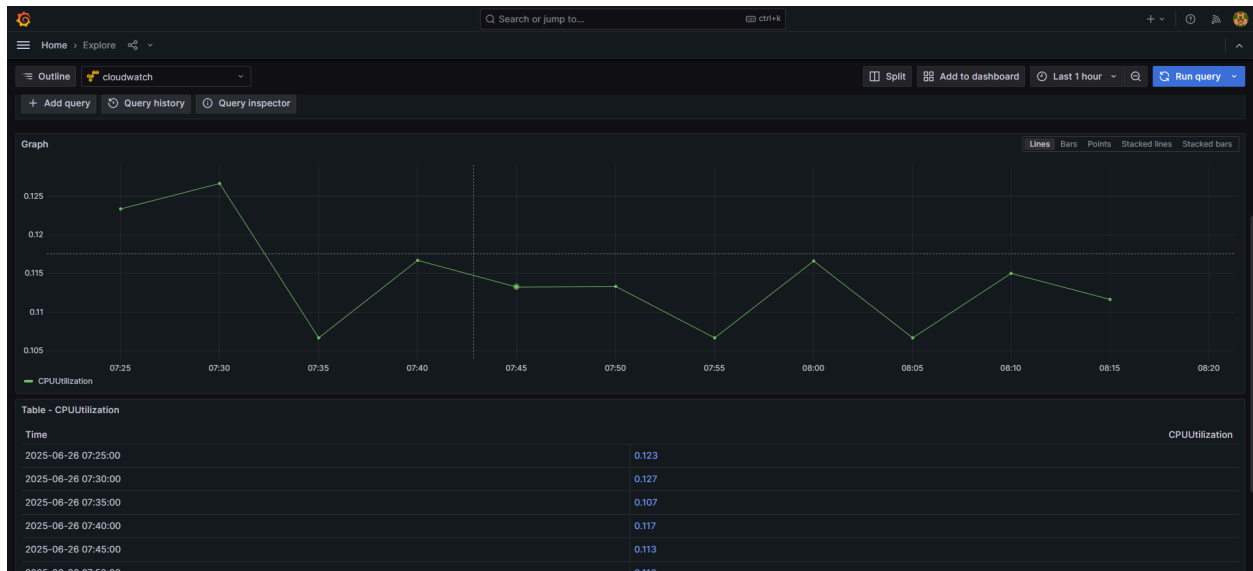
**Log events section:** This section shows a list of log events with columns for Timestamp and Message. The messages are JSON logs from the 'kimai-app' container, showing various HTTP requests and responses. For example, one log shows a GET request to '/ELB-HealthChecker/2.0/' with a 200 status code. Another shows a GET request to '/cur/1/7.88.1/' with a 200 status code. The logs are timestamped from 2025-06-26T04:08:43.550Z to 2025-06-26T04:09:28.261Z.

**Alarms section:** This section shows a list of alarms. The 'Kimai' alarm is highlighted, showing its details. The alarm is a 'Metric alarm' for 'CPUUtilization' with a threshold of '> 80 for 1 datapoints within 5 minutes'. The alarm is currently in a 'OK' state. The details section includes fields for Name, State, Type, Description, Namespace, Metric name, Instanceld, Instance name, Statistic, Period, Datapoints to alarm, Missing data treatment, Percentiles with low samples, and ARN.

## Grafana Setup

- Grafana installed on EC2
- Connected to CloudWatch via IAM role
- Dashboards created for:
  - CPU Utilization
  - Memory Usage
  - Disk Usage
  - Network Traffic

- Alerts enabled with thresholds



## Cost Estimation (Monthly)

Resource	Cost
EC2 (t3.large)	~\$33.28
Bastion Host (t3.micro)	~\$7.62
EBS (28GB total)	~\$2.80
S3 (backups + state)	~\$0.12
CloudWatch	~\$2.00
<b>Total</b>	<b>~\$45-48/month</b>

## Notes for Teams

- Use Git to always pull latest changes
- All setup scripts are idempotent
- Logs and backups go to AWS
- Easily replicable using Terraform anywhere

## Author

This project was fully designed, implemented, and documented by **Angeline Hephzibah J** as part of the TechForce Cloud Migration initiative.

Repository: <https://github.com/angelinedev/Cloud-Migration-Project>

All infrastructure, automation, monitoring, and security measures were independently built and validated on AWS using best practices.