# REST API: Weather Info for Dhaka – CI/CD + Kubernetes + Terraform

This project showcases a

- ⇨ Complete DevOps pipeline
- ⇨ For developing, containerizing, and automating the deployment of a REST API to Kubernetes using:

  CI/CD,
  Terraform and
  best practices in security, scalability, and observability.

#Project Overview

Objective:

Develop and deploy a REST API with the following capabilities:

Endpoint /api/hello:

- Returns:
- Hostname
- Current datetime
- Application version
- Live weather data for Dhaka

Endpoint /api/health:

- Returns health status of the API
- Verifies connectivity with the 3rd-party weather API

#Technologies Used

Language	Python (Flask)
Containerization	Docker, Docker Compose
CI/CD	Jenkins
Infrastructure	Terraform (modular), AWS
Orchestration	Kubernetes (K8s Manifests)
Secrets Management	Kubernetes Secrets
Observability	Prometheus, Grafana

#Project Structure

```
Terraform-REST-API/
│
├── app/                    # REST API source code
│   ├── main.py               # Flask app
│   ├── requirements.txt       # Python packages
│   └── ...
│
├── Dockerfile                 # Secure and optimized
├── docker-compose.yml         # Local development
│
├── Jenkinsfile         # CI/CD pipeline
│
│
├── terraform/                 # Terraform infrastructure (EKS)
```

```
│   ├── modules/eks/          # Modular EKS cluster
│   └── environments/dev/     # Environment-specific config
│
├── k8s/                      # Kubernetes manifests
│   ├── deployment. yaml
│   ├── service. yaml
│   └── secret. yaml
│
└── README.md                 # This file
```

Kubernetes Deployment

Manifests

- deployment.yaml: Deploys the app with health checks
- service.yaml: Exposes the app internally through Load Balancer
- secret.yaml: Stores weather API key securely

Terraform Infrastructure

- Modular and environment-specific structure
- Uses remote backend for state (e.g., S3 + DynamoDB)
- Creates an EKS cluster on AWS with node groups

Observability

Placeholder for monitoring stack:

- **Prometheus** to scrape metrics
- **Grafana** visual dashboards

Security Best Practices

- API keys are stored as **Kubernetes Secrets**
- Docker image is based on minimal secure base image and multi-stage build to reduce the size and non-root user has been used for security.
- Before pushing the image into DockerHub, by using Trivy, we perform the Vulnerability scanning.
- CI/CD uses secrets (not hardcoded tokens)

Scalability & Modularity

- Kubernetes handles auto-scaling
- Infrastructure is modular for multi-environment deployment

Contact

Mohammad Kamrul Islam
Email: kamrul.emc.199@example.com
GitHub: @mkislam-wipro-2015