**End-to-End DevOps Project with Kubernetes and Grafana Monitoring**

**Project Overview**

This project demonstrates a complete **CI/CD pipeline** for deploying an application on **Kubernetes** using **Jenkins, Terraform, Docker, and Prometheus-Grafana monitoring**. The goal is to simulate a real-world DevOps workflow suitable for a **2-year experienced engineer**.

**Tech Stack Used**

* **Version Control:** Git & GitHub
* **CI/CD Tool:** Jenkins
* **Infrastructure as Code (IaC):** Terraform
* **Containerization:** Docker
* **Orchestration:** Kubernetes
* **Monitoring:** Prometheus & Grafana
* **Cloud Provider:** AWS (for EC2, EKS, and VPC setup)

**Project Breakdown (5 Days Plan)**

**Day 1: Infrastructure Setup with Terraform**

* Created an AWS **VPC, subnets, security groups**.
* Deployed an **EKS (Elastic Kubernetes Service) cluster**.
* Provisioned an **EC2 instance** for Jenkins setup.

**Terraform Script Snippet:**

resource "aws\_vpc" "devops\_vpc" {

cidr\_block = "10.0.0.0/16"

}

resource "aws\_subnet" "public\_subnet" {

vpc\_id = aws\_vpc.devops\_vpc.id

cidr\_block = "10.0.1.0/24"

}

**Day 2: Jenkins CI/CD Pipeline**

* Installed Jenkins on EC2 and configured it with GitHub Webhooks.
* Created a **Jenkins Pipeline** to build and push Docker images.
* Used **GitHub Actions** to trigger builds.

**Jenkinsfile Snippet:**

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/user/repo.git'

}

}

stage('Build Docker Image') {

steps {

sh 'docker build -t myapp:latest .'

}

}

stage('Push to Docker Hub') {

steps {

sh 'docker push myapp:latest'

}

}

}

}

**Day 3: Kubernetes Deployment**

* Created **Kubernetes YAML manifests** for deployment and service.
* Used **kubectl apply** to deploy the application on EKS.

**Deployment YAML:**

apiVersion: apps/v1

kind: Deployment

metadata:

name: myapp-deployment

spec:

replicas: 2

selector:

matchLabels:

app: myapp

template:

metadata:

labels:

app: myapp

spec:

containers:

- name: myapp

image: myapp:latest

ports:

- containerPort: 80

**Day 4: Exposing Application & Testing**

* Created a **LoadBalancer Service** to expose the application.
* Verified deployment using kubectl get svc.

**Service YAML:**

apiVersion: v1

kind: Service

metadata:

name: myapp-service

spec:

type: LoadBalancer

selector:

app: myapp

ports:

- protocol: TCP

port: 80

targetPort: 80

**Day 5: Monitoring with Prometheus & Grafana**

* Installed **Prometheus and Grafana** on Kubernetes.
* Set up **Grafana dashboards** for Kubernetes monitoring.
* Used **port-forwarding** to access Grafana UI.

**Command to Port-Forward Grafana:**

kubectl port-forward svc/prometheus-grafana 3000:80

**Imported Prebuilt Dashboard (Grafana Dashboard ID 3119)**

* Go to **Import → Enter Dashboard ID**
* Select **Prometheus** as a data source
* View **Cluster CPU, Memory, and Pod Metrics**

**Project Outcome**

* Successfully deployed an end-to-end **CI/CD pipeline**.
* Automated **infrastructure provisioning with Terraform**.
* Achieved **containerized deployment on Kubernetes**.
* Implemented **real-time monitoring with Grafana & Prometheus**.

**Future Enhancements**

✅ Integrate **ArgoCD for GitOps** ✅ Implement **Helm charts for easier deployments** ✅ Set up **Log Aggregation with ELK stack**

🚀 **Congratulations! Your DevOps project is complete and ready for your resume.** 🎉

Let me know if you need further refinements! 😃