# Project Documentation – Trend App Deployment with CI/CD

## 1. Application Setup

- Clone the React application from GitHub: git clone https://github.com/Vennilavan12/Trend.git - Run locally to verify: npm install && npm start (runs on port 3000).

## 2. Dockerization

- Dockerfile created with Node.js for build and Nginx for serving. - Build: docker build -t trend-app:latest . - Run: docker run -d -p 3000:80 trend-app:latest - Access app at http://localhost:3000

## 3. Terraform Infrastructure

- Define VPC, Subnet, Security Groups, and EC2 for Jenkins in main.tf - Provision with: terraform init && terraform apply -auto-approve

## 4. DockerHub

- Create repository 'trend-app' in DockerHub. - Push image: docker tag trend-app:latest /trend-app:latest - docker push /trend-app:latest

## 5. Kubernetes (EKS)

- Setup AWS EKS cluster. - Deployment.yaml for application pods and Service.yaml with LoadBalancer. - Deploy: kubectl apply -f deployment.yaml && kubectl apply -f service.yaml - Get LoadBalancer URL: kubectl get svc

## 6. Version Control

- Add .gitignore and .dockerignore - Push to GitHub with: git add . && git commit -m "Initial commit" && git push origin main

## 7. Jenkins CI/CD

- Install plugins: Docker, GitHub, Kubernetes, Pipeline. - Configure GitHub Webhook to Jenkins. - Jenkinsfile: stages for Checkout, Build, Push Docker image, Deploy to Kubernetes.

## 8. Monitoring

- Option 1: Prometheus + Grafana for metrics (CPU, memory, latency). - Option 2: AWS CloudWatch Container Insights for EKS cluster monitoring.

## Workflow

1. Developer commits code → GitHub 2. Jenkins triggered via webhook → builds & pushes Docker image 3. Jenkins deploys image to EKS 4. EKS runs pods exposed via LoadBalancer 5. Monitoring via Prometheus/Grafana or CloudWatch