# **DevOps CI/CD Pipeline using Jenkins, Docker, and Kubernetes**

## **Project Objective**

Design and automate a robust CI/CD pipeline to deploy a multi-tier web application (frontend and backend) using:

- 1. Jenkins for Continuous Integration and Continuous Deployment
- 2. **Docker** for containerization
- 3. **Kubernetes** for orchestration (via **Minikube**)
- 4. AWS EC2 (t2.medium) as the host environment
- 5. **GitHub** as the code repository

The goal is to build a reliable, production-ready DevOps workflow.

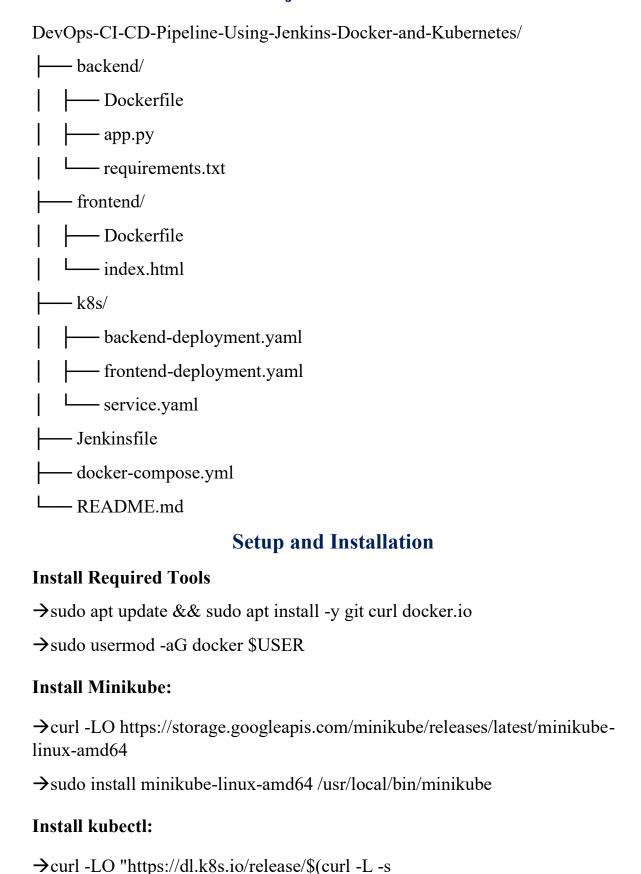
## **Hosting Environment**

- 1. Cloud Platform: AWS EC2
- 2. Instance Type: t2.medium (2 vCPU, 4 GB RAM)
- 3. Operating System: Ubuntu 22.04 LTS
- 4. **Security Group Ports Opened**: 22 (SSH), 8080 (Jenkins), 30080 (Backend), 30081 (Frontend)
- 5. **Minikube Tunnel**: Enabled using minikube tunnel --bind-address 0.0.0.0

## **Tools & Technologies**

Tool	Purpose
GitHub	Version Control
Docker	Containerize frontend/backend apps
Jenkins	Automate CI/CD pipeline
Kubernetes	Manage containerized workloads
Minikube	Local Kubernetes cluster
Nginx	Frontend static file server
Flask	Python backend web server

## **Project Structure**



https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

→chmod +x kubectl && sudo my kubectl /usr/local/bin/

#### **Start Minikube:**

- →minikube start --driver=docker
- →minikube tunnel --bind-address 0.0.0.0

#### **Docker: Containerization**

#### **Backend Dockerfile**

FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt.

RUN pip install Flask==2.0.3 Werkzeug==2.0.3

COPY app.py.

EXPOSE 5000

CMD ["python", "app.py"]

#### **Frontend Dockerfile**

FROM nginx:alpine

COPY index.html /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

### **Build & Push Images**

- →docker build -t tpathak21/devops-backend:latest ./backend
- →docker build -t tpathak21/devops-frontend:latest ./frontend
- →docker push tpathak21/devops-backend:latest
- →docker push tpathak21/devops-frontend:latest

## **Kubernetes Deployment**

### **Backend Deployment**

• Kind: Deployment

• Name: devops-backend

• Replicas: 2

• Container Image: tpathak21/devops-backend:latest

• Container Port: 5000

• Label: app: devops-backend

Purpose: Launches two backend container replicas managed by Kubernetes

#### **Backend Service**

• Kind: Service

• Name: devops-backend-service

• Type: NodePort

• Selector: app: devops-backend

• Ports:

• Port: 5000

• TargetPort: 5000

• NodePort: 30080

Purpose: Exposes backend app outside the cluster on EC2 instance's

IP:30080

## **Frontend Deployment**

• Kind: Deployment

• Name: devops-frontend

• Replicas: 2

• **Container Image**: tpathak21/devops-frontend:latest

• Container Port: 80

• Label: app: devops-frontend

• Purpose: Launches two frontend container replicas

#### **Frontend Service**

Kind: Service

• Name: devops-frontend-service

• Type: NodePort

• Selector: app: devops-frontend

• Ports:

• **Port**: 80

• TargetPort: 80

• **NodePort**: 30081

Purpose: Exposes frontend app outside the cluster on EC2 instance's

IP:30081

#### **Apply Resources**

- →kubectl apply -f k8s/
- →kubectl get pods -o wide
- →kubectl get svc

# Jenkins CI/CD Pipeline

## **Agent Declaration**

• Type: any

• Purpose: Allows pipeline to run on any available Jenkins agent (node)

## **Environment Setup**

- DOCKERHUB CREDENTIALS:
- Secure Jenkins credentials (ID: dockerhub-creds) used for DockerHub login

#### **Stages**

## Stage 1: Checkout Code from GitHub

Clones the main branch from your GitHub repository:

https://github.com/TejPATHAK/DevOps-CI-CD-Pipeline-using-Jenkins-

## Docker-and-Kubernetes

Stage 2: Build Backend Docker Image

- Builds Docker image:
- Image: tpathak21/devops-backend:latest
- Context: ./backend

### Stage 3: Build Frontend Docker Image

- Builds Docker image:
  - Image: tpathak21/devops-frontend:latest
  - Context: ./frontend

#### Stage 4: Login to DockerHub

- Authenticates Jenkins with DockerHub using securely stored credentials
- Stage 5: Push Docker Images
- Pushes both frontend and backend images to DockerHub:
  - tpathak21/devops-backend:latest
  - tpathak21/devops-frontend:latest

#### **Stage 6:** Deploy to Kubernetes

- Uses kubectl to apply manifests for:
  - Backend Deployment
  - Frontend Deployment
  - NodePort Services
- Uses the Kubeconfig file located at: /var/lib/jenkins/.kube/config

#### **End Result**

- Fully automated CI/CD pipeline from GitHub to DockerHub to a live Kubernetes cluster
- Runs seamlessly inside an AWS EC2 instance with Jenkins managing every stage

# **Output Validation**

#### **curl Test**

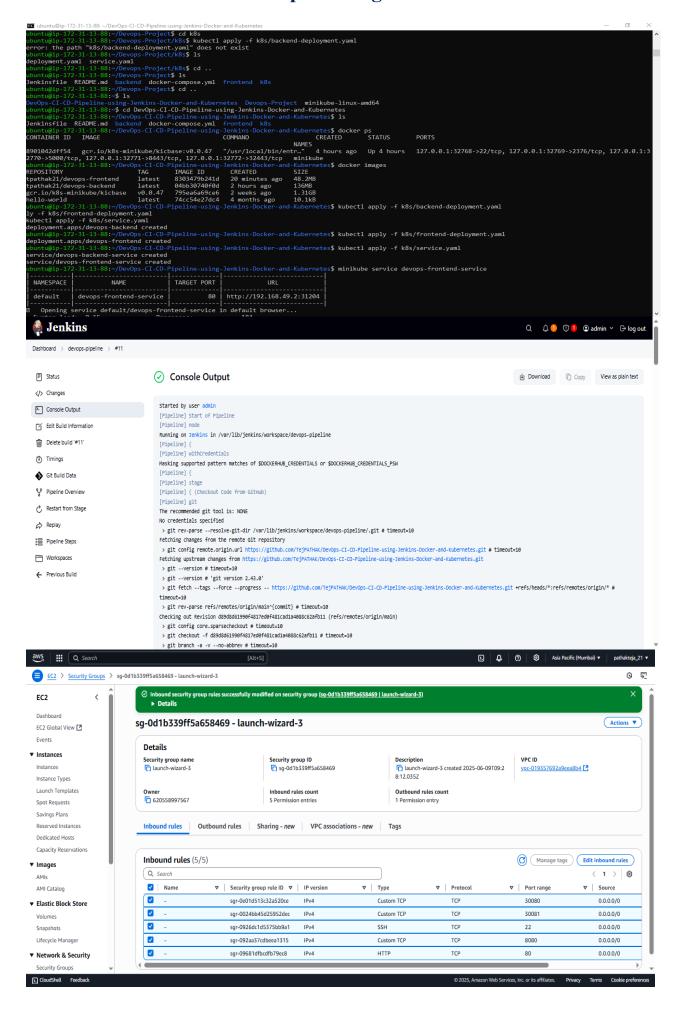
→curl http://<EC2-PUBLIC-IP>:30080 # Backend

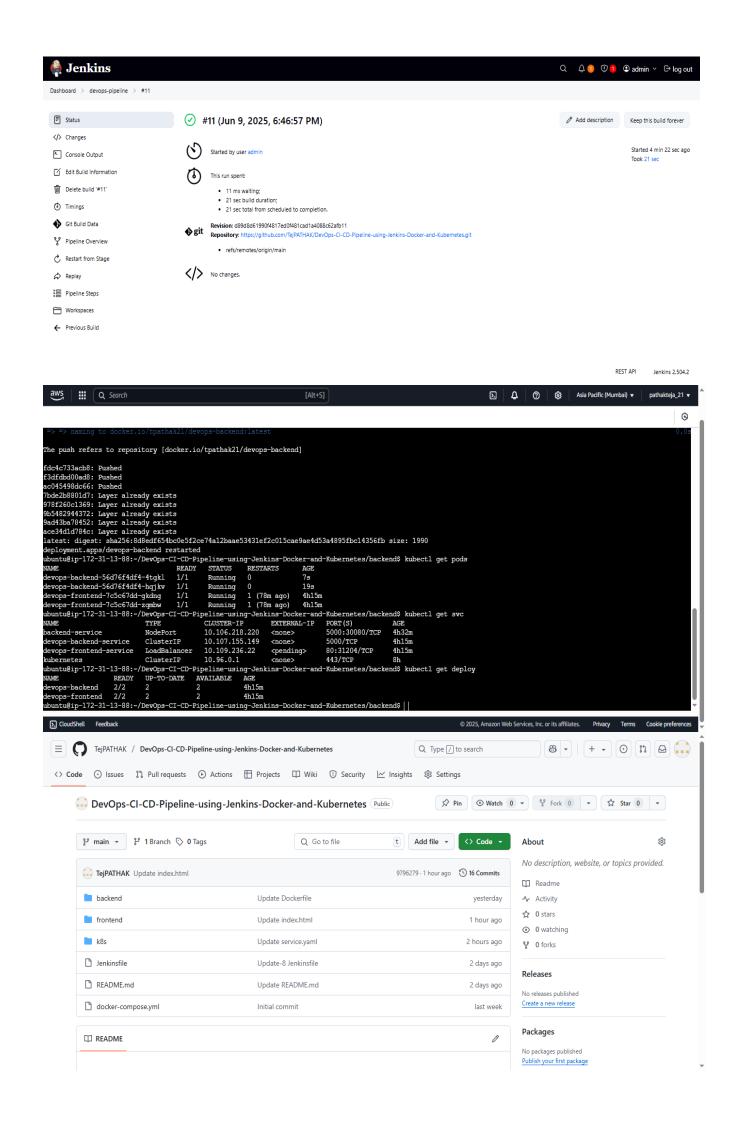
→curl http://<EC2-PUBLIC-IP>:30081 # Frontends

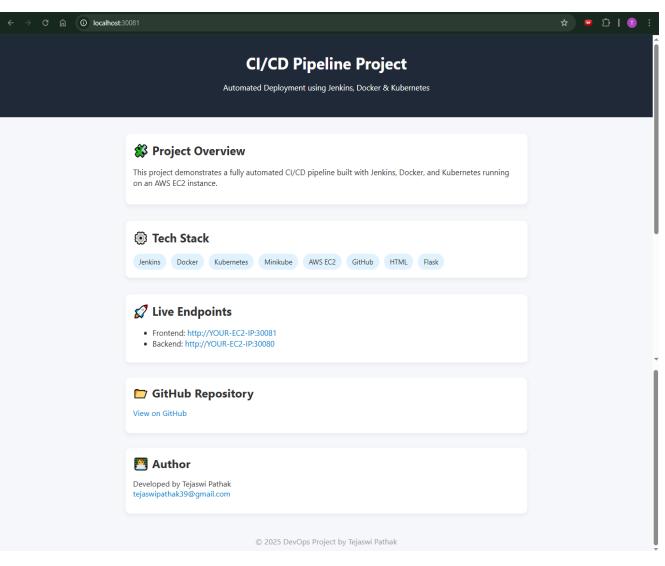
### Jenkins Pipeline

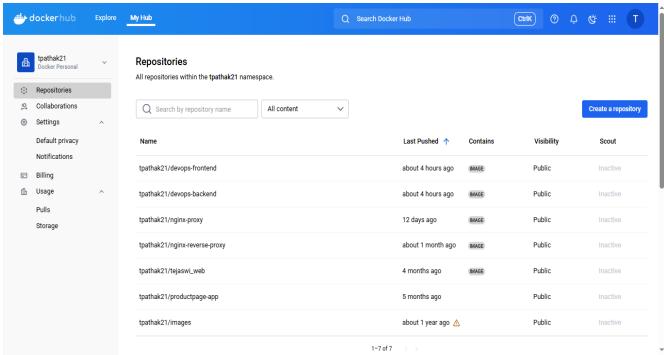
All stages: Clone > Build > Push > Deploy should show as green

## **Outputs Images**









# Links

GitHub Repository: <u>DevOps-CI-CD-Pipeline</u>

LinkedIn Profile: <u>Tejaswi Pathak</u>

DockerHub Profile: <a href="mailto:tpathak21">tpathak21</a>

## **Conclusion**

This project is a full-fledged demonstration of how to containerize applications, automate deployments using Jenkins, and orchestrate containers via Kubernetes - all deployed on an AWS EC2 instance. It proves your ability to integrate multiple DevOps tools in a production-like setup.