CI/CD Pipeline with Jenkins & Docker

1. Introduction

In modern software development, Continuous Integration and Continuous Deployment (CI/CD) play a vital role in automating build, testing, and deployment processes. This project demonstrates how to set up a full CI/CD pipeline using Jenkins and Docker to automate the entire workflow — from code commit to deployment — without relying on any cloud service. The goal is to create a seamless automation setup where Jenkins automatically builds a Docker image, tests the application, pushes it to Docker Hub, and finally deploys it locally using Minikube or a local virtual machine.

2. Abstract

This project implements a local CI/CD pipeline that ensures efficient, repeatable deployments for containerized applications. Using Jenkins as the automation server, the pipeline pulls source code from GitHub, runs tests, builds a Docker image, pushes it to Docker Hub, and deploys it using Docker Compose or Minikube. The approach enhances consistency between environments, reduces manual intervention, and demonstrates how DevOps principles can be applied even in a fully local setup.

3. Tools Used

Tool	Purpose
Jenkins	Automation server to implement the CI/CD pipeline
Docker	Containerization of the application
Docker Hub (Free)	Remote registry for storing built images
Minikube / Local VM	Local deployment environment
Git & GitHub	Version control and source repository
Node.js	Application runtime (sample app used in project)

4. Steps Involved in Building the Project

Step 1: Setup Jenkins
Installed Jenkins locally and configured Docker and Git plugins. Created a new Freestyle project or Pipeline job in Jenkins. Integrated Jenkins with GitHub to automatically fetch code from the repository.

Step 2: Create Application and Dockerfile
Built a simple Node.js app (index.js) with a basic web server. Created a Dockerfile as shown below: FROM node:18-alpine WORKDIR /app COPY package*.json ./ RUN npm install COPY . . EXPOSE 3000 CMD ["node", "index.js"]
Step 3: Configure Jenkins Pipeline
Defined stages in Jenkinsfile — build, test, push image to Docker Hub, and deploy locally.
pipeline { agent any stages { stage('Build') { steps { sh 'docker build -t yourname/jenkins-cicd-demo:latest .' } } stage('Push to Docker Hub') { steps { withCredentials([string(credentialsId: 'dockerhub-pass', variable: 'DOCKER_PASS')]) { sh 'echo $DOCKER_PASS | docker login -u yourname --password-stdin' sh 'docker push

yourname/jenkins-cicd-demo:latest' } } } stage('Deploy Locally') { steps { sh 'docker-compose down || true' sh 'docker-compose up -d' } } } }

Step 4: Run and Verify

Triggered the Jenkins pipeline automatically after each Git push. Verified successful image push to Docker Hub and app deployment on localhost:3000.

5. Conclusion

The project successfully demonstrated a complete CI/CD workflow using Jenkins and Docker in a local environment. It automated build, test, and deployment stages, ensuring faster and more reliable application delivery. This implementation shows how DevOps principles — even without cloud tools — can enhance local development efficiency, reproducibility, and scalability.

Deliverables

• GitHub Repository: https://github.com/your-username/jenkins-cicd-demo
• Docker Hub Image: https://hub.docker.com/r/your-username/jenkins-cicd-demo
• Screenshots: Jenkins pipeline success, Docker image on Docker Hub, and deployed app running locally.