CI/CD Pipeline with GitHub Actions & Docker

Introduction

This project demonstrates the creation of a complete CI/CD pipeline using GitHub Actions and Docker without relying on any cloud services. The aim is to automate the process of building, testing, and deploying a Dockerized Node.js application locally using Minikube or a local virtual machine.

Abstract

Continuous Integration and Continuous Deployment (CI/CD) are key practices in modern software development. This project sets up an end-to-end CI/CD pipeline to automate testing and deployment of a Node.js app using GitHub Actions. The pipeline builds a Docker image, runs tests, and pushes the image to Docker Hub. The image is then deployed using a local Kubernetes environment like Minikube.

Tools Used

- GitHub Actions (for automation)
- Docker & Docker Hub (for containerization and image registry)
- Minikube or Local VM (for local deployment)
- PowerShell / Git Bash (for CLI commands)
- Visual Studio Code (for code editing)
- Docker Desktop (for managing Docker environment)

Steps Involved in Building the Project

1. Initialize Node.js Project (VS Code, Git Bash):
   Run **npm init -y** and install express using **npm install express**
   Create index.js and app.test.js
2. Add Test Runner (VS Code):
   Install Jest using **npm install --save-dev jest**
   Update package.json with test script
3. Create Dockerfile and docker-compose.yml (VS Code):
   Define image build instructions and exposed ports
4. Set Up GitHub Repository (GitHub Web UI):
   Push project to GitHub
5. Configure GitHub Actions (GitHub Web UI / VS Code):
   Create .github/workflows/docker-ci.yml

Define jobs to run tests, build Docker image, and push to Docker Hub

6. Add GitHub Secrets (GitHub Settings):
   Add DOCKER_USERNAME and DOCKER_PASSWORD
7. Run GitHub Workflow (GitHub → Actions Tab):
   Confirm pipeline executes successfully
8. Deploy Locally (Docker Desktop, Minikube):
   Pull image using **docker pull username/ci-cd-demo**
   Run container locally or deploy on Minikube using **kubectl**

Conclusion

The project successfully implemented a full CI/CD pipeline using GitHub Actions and Docker. It automates the entire lifecycle from code commit to deployment using local infrastructure. This workflow can be extended further with environment variables, Kubernetes YAMLs, or Helm charts for more robust deployments.