CI/CD Pipeline with GitHub Actions & Docker (No cloud needed)

Prepared by: Surisetti Kamal Krishna Kumar

Batch: #05 | DevOps Intern at Elevate Labs

GitHub Repository: https://github.com/Kamalkumar25/Elevate_labs_main_project.git

Objective

The objective of this project was to design and implement a fully automated CI/CD pipeline using GitHub Actions and Docker for a professional-grade calculator application. The pipeline builds, tests, and deploys the application locally without relying on any cloud infrastructure. This project aims to demonstrate continuous integration, containerization, and automated deployment capabilities essential for modern DevOps practices.

Project Overview

The Elevate Calculator is a Node.js-based web application built using Express.js. It performs arithmetic operations (addition, subtraction, multiplication, division) through a REST API endpoint. The project was containerized using Docker, and a CI/CD pipeline was created with GitHub Actions to automatically build, test, and push the Docker image to Docker Hub on every commit. Additionally, a professional frontend user interface was designed to make it look like a modern, interactive calculator with theme toggle and extended features such as time conversion, age calculation, and base conversions.

Tools and Technologies Used

- Node.js & Express.js: Backend API development

- Docker: Containerization and deployment

- Docker Hub: Image registry for build artifacts

- GitHub Actions: CI/CD automation pipeline

- Git & GitHub: Version control and workflow management

- VS Code & CMD: Development and testing environment

Project Architecture and Workflow

1. Developer commits or pushes code to the main branch on GitHub.
2. GitHub Actions automatically triggers the workflow (ci-cd.yml).
3. Workflow installs dependencies, runs tests, builds the Docker image.
4. Docker image is pushed to Docker Hub using credentials stored as GitHub Secrets.
5. Locally, the image can be pulled and deployed using Docker Compose or the docker run command.
6. The calculator app becomes accessible via **http://localhost:9090.**

Challenges Faced:

- Encountered Docker port binding conflicts (resolved by changing ports).

- GitHub Actions authentication issues with Docker Hub (fixed by creating access tokens with proper scopes).

- Handling environment configuration and ensuring correct build context for Docker.

- Integrating the frontend UI with backend API within the same Docker container.

Benefits of This Project

- Demonstrates complete DevOps automation workflow from code commit to deployment.

- Enhances understanding of containerization and CI/CD processes.

- Reduces manual deployment time by automating build and testing stages.

- Provides a scalable, portable solution deployable anywhere using Docker.

- Adds professional real-world project experience relevant to industry standards.

Project Outcome

The CI/CD pipeline successfully builds and pushes Docker images automatically on each commit. The calculator application runs both as a REST API and as a professional user interface accessible via browser. All pipeline stages (build, test, push) were verified successfully on GitHub Actions. The project serves as a complete demonstration of practical DevOps skills, containerization, and continuous integration workflows.

Conclusion

This project demonstrates a comprehensive CI/CD pipeline implementation integrating GitHub Actions and Docker. It highlights automation, collaboration, and deployment best practices. By completing this project, I gained a deep understanding of how DevOps pipelines streamline software delivery and ensure reliable, repeatable deployments.