CI/CD Pipeline with GitHub Actions & Docker (AWS + Minikube)

Introduction

This project demonstrates the implementation of a CI/CD pipeline using GitHub Actions, Docker, and Minikube on AWS infrastructure. The pipeline automates building, testing, and deploying a containerized application, ensuring efficient and reliable software delivery.

Abstract

The objective of this project is to design and implement a complete CI/CD workflow. The pipeline builds a Docker image, runs tests, pushes the image to Docker Hub, and deploys the application on a Minikube cluster hosted on AWS EC2. This setup eliminates manual intervention and ensures seamless deployment of code changes.

Tools Used

1. GitHub & GitHub Actions – for version control and automation.
2. Docker & Docker Hub – for containerization and image repository.
3. Minikube – for running Kubernetes locally on AWS EC2.
4. kubectl – to manage Kubernetes resources.
5. AWS EC2 – as the infrastructure to host Minikube.

Steps Involved in Building the Project

1. Created a sample Python Flask application with Dockerfile and requirements.txt.
2. Built and tested the application locally using Docker Compose.

3. Configured GitHub Actions workflow to run tests, build Docker images, and push to Docker Hub.
4. Set up Minikube on AWS EC2 and deployed the application using Kubernetes manifests.
5. Exposed the application using NodePort/LoadBalancer and validated the deployment with minikube service & kubectl port-forward.

Conclusion

This project successfully implemented a CI/CD pipeline with automated builds, testing, and deployments using GitHub Actions and Docker. Deploying on Minikube (within AWS EC2) demonstrated Kubernetes integration, making the setup scalable and production-ready. This workflow ensures faster delivery, reduced manual effort, and improved reliability in software deployment.