**PROJECT REPORT**

# CI/CD with GitHub Actions → Docker Hub → Kubernetes (K3s/Minikube)

# Introduction

This project is part of a DevOps internship program and aims to build a hands-on understanding of how modern applications are containerized, deployed, and managed in the cloud. The selected project focuses on deploying a Dockerized application on an AWS EC2 instance. The project also integrates Kubernetes configuration for future scalability, thus combining key DevOps concepts of automation, containerization, and orchestration. By completing this project, the objective is to showcase the ability to take an application from source code to a running production-like environment while documenting the process.

# Abstract

In today’s cloud-native development landscape, containerization and orchestration have become crucial for scalability, reliability, and automation. This project demonstrates the end-to-end process of hosting a containerized application. A GitHub repository was used for version control, containing the application code, Dockerfile, and Kubernetes manifests. An EC2 instance was provisioned on AWS to serve as the deployment environment. Docker was installed and configured, followed by building the image and running the application container. The deployment was verified by exposing the service and testing it through a browser using the EC2 public IP. The results include logs, screenshots, and configuration files, all stored in the GitHub repository for final submission.

# Tools Used

- \*\*AWS EC2\*\*: Provided the compute environment for deployment. - \*\*Docker\*\*: Enabled application containerization with an isolated and reproducible runtime. - \*\*GitHub\*\*: Hosted the source code and served as the central repository. - \*\*Kubernetes (k8s)\*\*: Configuration files prepared for container orchestration and scaling. - \*\*Linux (Amazon Linux)\*\*: Used as the operating system for hosting the Docker runtime. - \*\*Browser & Client Tools\*\*: Verified deployment through browser access and logs.

# Steps Involved

1. \*\*Environment Setup\*\*: A fresh AWS EC2 instance was launched with proper inbound rules for SSH (22) and HTTP (80).

2. \*\*Installing Dependencies\*\*: Docker was installed on the EC2 instance and configured with user permissions.

3. \*\*Cloning the Repository\*\*: The GitHub repository containing the Dockerfile, application code, and Kubernetes manifests was cloned.

4. \*\*Building the Docker Image\*\*: The Dockerfile was used to build a custom image of the application.

5. \*\*Running the Container\*\*: The container was started by mapping EC2 port 80 to the application’s exposed port 8080.

6. \*\*Verifying Deployment\*\*: The application was accessed via the EC2 public IP in a browser, confirming successful deployment.

7. \*\*Capturing Evidence\*\*: Screenshots of the running container (`docker ps`) and browser output were taken.

8. \*\*Preparing Kubernetes Manifests\*\*:

YAML files for Kubernetes deployment and service were added for potential scaling on k8s.

9. \*\*Documentation\*\*: A short project report and screenshots were stored in the `docs/` folder of the repository.

# Conclusion

This project successfully demonstrates the process of containerizing and deploying an application using Docker on AWS EC2. It highlights practical DevOps skills including environment setup, dependency installation, container image creation, deployment execution, and verification. The project also includes Kubernetes manifests to illustrate how the same application can be scaled in a production-ready orchestration environment. By integrating screenshots, logs, and documentation, the project is not only functional but also well-documented for submission. This practical experience will support future work in CI/CD, Kubernetes-based orchestration, and cloud-native application deployments.

Submmited By,

Avishkar padir