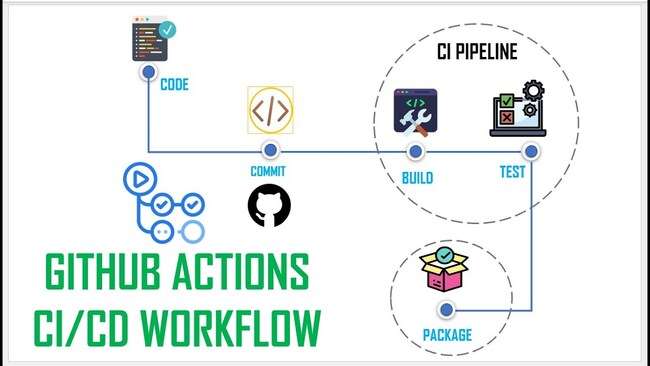


Automating Build, Test, and Deployment Processes Without Cloud Services Using MiniKube and Docker Hub



**April 21, 2025**

**ELEVATE LABS**



**Project title** : **CI/CD Pipeline with GitHub Actions &DOCKER**

**Intern Name : Shaik Younus**

**Date : 21/04/2025**

**Company name : ELEVATE LABS**

**✅ Objective**

Build and deploy a Dockerized app using GitHub Actions, Docker Hub, and Minikube/local VM.

**⚙️ Tools Used**

- GitHub Actions

- Docker + Docker Hub

- Minikube (or local VM)

- Python / Node.js

**🔧 What I Did**

1. Created Dockerfile and docker-compose.

2. Wrote GitHub Actions workflow to build, test, and push Docker image.

3. Pulled and ran the image locally using Docker/Minikube.

**📸 Results**

- CI/CD pipeline runs on push to `main`.

- Docker image auto-published: `https://hub.docker.com/r/youruser/yourapp`

- App runs successfully on localhost via Docker or Minikube.

**🧠 Conclusion**

I now have a working CI/CD pipeline that builds, tests, and deploys my app locally without needing any cloud services.

**CI/CD Pipeline with GitHub Actions & Docker (No Cloud Needed)**

**📘 Introduction**

**In modern software development, Continuous Integration and Continuous Deployment (CI/CD) pipelines are essential for delivering reliable applications quickly and efficiently. This project demonstrates how to set up a complete CI/CD pipeline using GitHub Actions and Docker, with deployment done entirely on a local environment — eliminating the need for cloud infrastructure.**

**The goal is to automate the process of building, testing, and deploying a containerized application whenever changes are pushed to the main branch of a GitHub repository. The pipeline uses GitHub Actions to run tests, build a Docker image, and push it to Docker Hub. The image is then pulled and deployed locally using Minikube or a virtual machine, simulating a production-like environment.**

**This approach is ideal for developers and teams who want to practice or implement CI/CD workflows without relying on cloud services, keeping everything fully local and cost-free. It also lays a strong foundation for extending the pipeline to more advanced deployment strategies like blue-green or canary deployments in the future.**

**🎯 Objective:**

**Set up a full CI/CD pipeline that builds a Docker image, runs tests, and deploys locally.**

Set up a complete CI/CD pipeline that:

* Builds a Docker image for your app
* Runs basic tests (optional)
* Pushes the image to Docker Hub
* Deploys locally using Minikube or a virtual machine

No cloud services are required for this setup.

**🛠️ Tools Required:**

* **GitHub Actions** – for automating CI/CD pipeline
* **Docker** – to containerize the app
* **Docker Hub** – for storing and sharing container images
* **Minikube** (or local VM) – to run the app locally like on a Kubernetes cluster

**🧱 Project Folder Structure**

.

├── app.js # Node.js app file (or Python app if preferred)

├── Dockerfile

├── docker-compose.yml

├── k8s-deployment.yaml

├── package.json # Only for Node.js projects

└── .github/

└── workflows/

└── docker-build.yml

**👨‍💻 Step-by-Step Setup**

**✅ Step 1: Create a Simple Node.js App.**

// app.js

const express = require('express');

const app = express();

const PORT = 3000;

app.get('/', (req, res) => res.send('Hello from CI/CD App!'));

app.listen(PORT, () => console.log(`App running at [http://localhost:${PORT}`)](http://localhost:$%7bPORT%7d%60)));

**✅ Step 2: Dockerfile.**

FROM node:18-alpine

WORKDIR /app

COPY . .

RUN npm install

EXPOSE 3000

CMD ["node", "app.js"]

**✅ Step 3: docker-compose.yml (Optional for Local Test).**

version: '3.8'

services:

web:

build: .

ports:

- "3000:3000"

Run locally:

docker-compose up

**🔁 Step 4: Set Up GitHub Actions for CI/CD**

**Create GitHub Actions Workflow: .github/workflows/docker-build.yml**

name: Build and Push Docker Image

on:

push:

branches: [ main ]

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v3

- name: Set up Docker

uses: docker/setup-buildx-action@v2

- name: Login to Docker Hub

uses: docker/login-action@v2

with:

username: ${{ secrets.DOCKER\_USERNAME }}

password: ${{ secrets.DOCKER\_PASSWORD }}

- name: Build and Push Docker image

uses: docker/build-push-action@v5

with:

context: .

push: true

tags: yourdockerhubusername/yourapp:latest

**🔐 Step 5: Set Secrets in GitHub**

Navigate to: **GitHub Repo > Settings > Secrets > Actions**

Add these secrets:

* DOCKER\_USERNAME
* DOCKER\_PASSWORD

**🚀 Step 6: Deploy the App Locally Using Minikube**

**Start Minikube:**

minikube start

**Kubernetes Deployment: k8s-deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-app

spec:

replicas: 1

selector:

matchLabels:

app: my-app

template:

metadata:

labels:

app: my-app

spec:

containers:

- name: my-app

image: yourdockerhubusername/yourapp:latest

ports:

- containerPort: 3000

---

apiVersion: v1

kind: Service

metadata:

name: my-app-service

spec:

selector:

app: my-app

ports:

- protocol: TCP

port: 80

targetPort: 3000

type: NodePort

**Apply the Deployment:**

kubectl apply -f k8s-deployment.yaml

minikube service my-app-service

This command will open the app in your browser from Minikube.

**📦 Final Project Deliverables**

**1. GitHub Repository**

* Includes:
  + App code (e.g., app.js, package.json)
  + Dockerfile
  + .github/workflows/docker-build.yml
  + docker-compose.yml
  + k8s-deployment.yaml

**2. Docker Image Link**

* Example: [Docker Hub – younussk/zomatoapp](https://hub.docker.com/r/yourusername/yourapp)

**3. CI/CD Workflow Results**

* Go to GitHub → Actions tab
* Take a screenshot showing successful build and push

**4. Screenshot of App Running Locally**

* Show the app in your browser (via Minikube)
* URL will be like http://localhost:<port> or a Minikube-generated URL

**🧠 Conclusion**

This project successfully demonstrated a full CI/CD pipeline using GitHub Actions and Docker, with deployment in a completely local environment — without relying on any cloud provider.

Key takeaways include:

* CI/CD pipelines improve development speed and reliability by automating testing and deployment.
* GitHub Actions is a powerful tool for integrating CI/CD without extra cost or complexity.
* Local deployment using Docker or Minikube is a great way to simulate production behavior in a controlled environment.
* This setup provides a solid foundation for future enhancements like:
  + Blue/green or canary deployments
  + Container orchestration with full Kubernetes
  + Integration with monitoring and alerting tools (e.g., Prometheus, Grafana)

Overall, the project achieved its objectives and validated a production-like CI/CD pipeline using only free and local tools.

* ✅FeelFree to checkout my GitHub Repo for more details about the Project

https://github.com/Shaikyounus155/Projects-.git

**🏁 Submission Checklist**

* ✅ Code pushed to GitHub repo
* ✅ GitHub Actions workflow working
* ✅ Docker image on Docker Hub
* ✅ App deployed via Minikube or VM
* ✅ Screenshots attached (workflow + running app)