CI/CD Pipeline Project Report: Wellness Tracker

Name: Arpit Mishra
Course/Program: B.Tech (CSE)
Project: CI/CD Pipeline with GitHub Actions & Docker
Date: September 2025

1. Introduction

The aim of this project is to set up a **full CI/CD pipeline** for a web application named **Wellness Tracker**, enabling automated building, testing, and deployment of the application locally using Docker and Minikube.

Continuous Integration (CI) and Continuous Deployment (CD) are key practices in DevOps that automate the software delivery process, reduce manual intervention, and ensure faster and more reliable deployments.

2. Project Objective

- Containerize the Wellness Tracker app using **Docker**.
- Automate build, test, and deployment using **GitHub Actions**.
- Deploy the app locally using **Minikube**.
- Ensure reproducibility and version control for the deployment pipeline.

3. Tools & Technologies

Tool/Technology	Purpose
Node.js (v18) & Vite	Application development
React.js	Frontend UI development
Docker	Containerization of the app
GitHub Actions	CI/CD workflow automation
Docker Hub	Docker image repository
Minikube	Local Kubernetes cluster for deployment

Tool/Technology	Purpose
kubectl	Kubernetes management CLI
TailwindCSS & Chart.js	Styling & chart components

4. Application Details

Name: Wellness Tracker
Port: 3000 (inside container)
Dependencies: React, Chart.js, TailwindCSS, UUID

Directory Structure:

```
wellness-tracker/
├── Dockerfile
├── deployment.yaml
├── service.yaml
├── package.json
├── src/
└── .github/workflows/docker.yml
```

5. Docker Setup

Dockerfile:

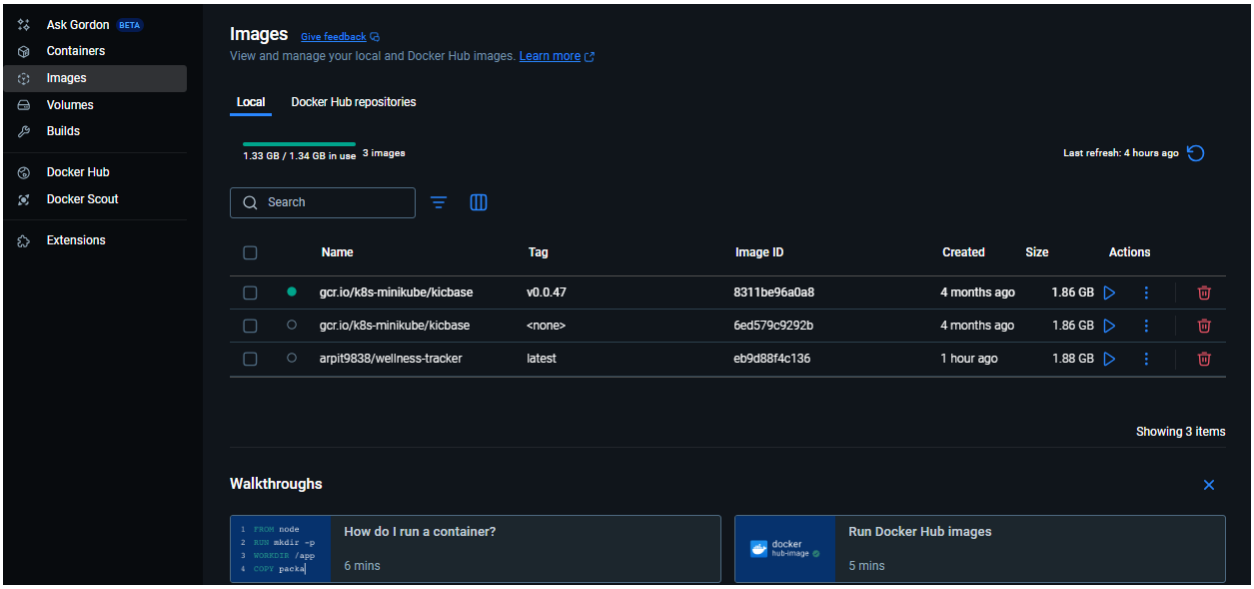
```
FROM node:18

WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
EXPOSE 3000
CMD ["npm", "run", "preview", "--", "--port", "3000", "--host"]
```

Commands to build and run locally:

```
docker build -t arpit9838/wellness-tracker:latest .
docker run -p 3000:3000 arpit9838/wellness-tracker:latest
```

Access: `http://localhost:3000`
Screenshot:



6. Kubernetes Deployment

Deployment (`deployment.yaml`):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wellness-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wellness
  template:
    metadata:
      labels:
        app: wellness
    spec:
      containers:
        - name: wellness
          image: arpit9838/wellness-tracker:latest
          ports:
            - containerPort: 3000
```

Service (`service.yaml`):

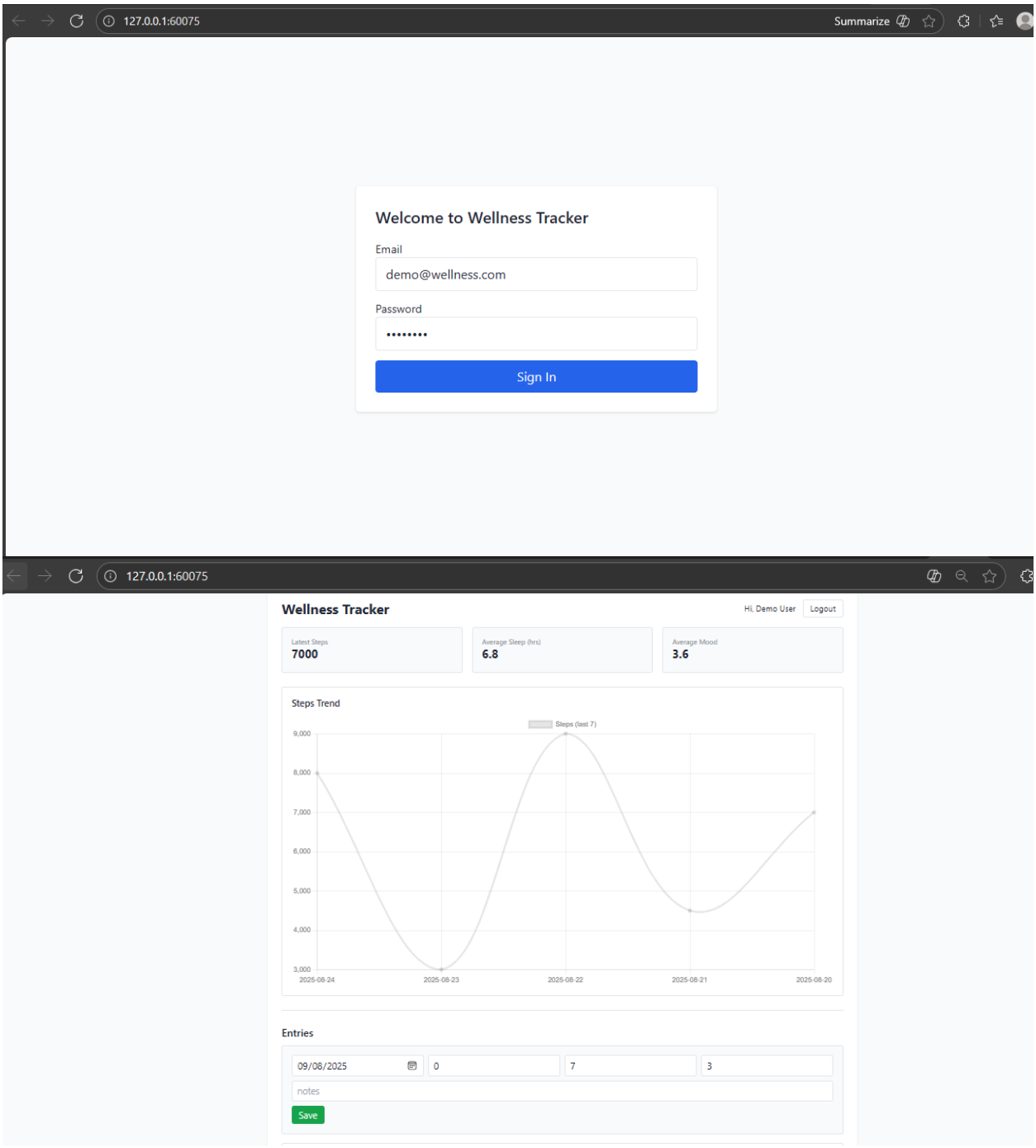
```
apiVersion: v1
kind: Service
metadata:
  name: wellness-service
spec:
  type: NodePort
```

```
  selector:
    app: wellness
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
      nodePort: 32000
```

Deployment Commands:

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
kubectl get pods
minikube service wellness-service --url
```

Expected URL: Wellness Tracker (Demo) or `http:// 192.168.58.2:32000`
Screenshot:



7. CI/CD Pipeline with GitHub Actions

Workflow File: `.github/workflows/docker.yml`

```
name: CI/CD Pipeline

on:
```

```
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: 18
      - run: npm install
      - run: npm run build
      - uses: docker/login-action@v2
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}
      - run: docker build -t arpit9838/wellness-tracker:latest .
      - run: docker push arpit9838/wellness-tracker:latest
```

Pipeline Steps:

1. Checkout code from GitHub
2. Set up Node.js environment
3. Install dependencies and build the app
4. Log in to Docker Hub
5. Build Docker image and push to Docker Hub

8. Results

Pod Status (Kubernetes):

```
kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE

wellness-service   NodePort   10.104.136.233   <none>        3000:32000/TCP
23m
```

Service URL:
Wellness Tracker (Demo) or `http:// 192.168.58.2:32000`

Docker Hub Image:
arpit9838/wellness-tracker

9. Key Notes

- NodePort exposes the application on host port `32000`.
- Ensure Minikube is running before deploying: `minikube start`.
- Always load the latest Docker image into Minikube:

```
minikube image load arpit9838/wellness-tracker:latest
```

10. Conclusion

The project demonstrates the end-to-end CI/CD pipeline setup using **GitHub Actions**, **Docker**, and **Kubernetes (Minikube)**. The pipeline successfully builds, tests, and deploys the Wellness Tracker app locally without relying on cloud infrastructure. This approach improves automation, reproducibility, and reduces manual deployment errors.

Deliverables

1. GitHub Repository with Workflow
2. Docker Image on Docker Hub
3. CI/CD workflow run screenshots
4. Local deployment screenshots