Vidyavardhaka Sangha®, Mysore
VIDYAVARDHAKA COLLEGE OF ENGINEERING
Autonomous Institute, Affiliated to Visvesvaraya Technological University, Belagavi
(Approved by AICTE, New Delhi & Government of Karnataka)
Accredited by NBA (CV, CS, EE, EC, IS & ME) | NAAC with 'A' Grade
P.B. No. 206, Gokulam III Stage, Mysuru-570 002, Karnataka, India
Phone: +91 821 4276201 /202 /225, Fax: +91 824 2510677
Web: http://www.vvce.ac.in @vvceofficial
VVCE

DevOps(21CS62)

Open Ended Experiment

On

"Deploying a Static Website Using Docker and GitHub CI/CD"

Submitted By:

Keerthana S	**4VV21CS077**
Lekhana S	**4VV21CS085**
Mohammed Awez	**4VV21CS097**
Preetham Raj Gowda	**4VV21CS116**

Submitted To:

Prof. Anil Kumar BH
Department of
Computer Science and Engineering
VVCE

Abstract :

This project report provides a comprehensive guide on deploying a static HTML web page using Docker and automating the CI/CD pipeline with GitHub Actions. The primary objective is to containerize a simple web application with Docker, ensuring consistent performance across different environments, and to streamline the deployment process using GitHub Actions. The report details the step-by-step implementation of setting up the project directory, creating an HTML page, writing a Dockerfile, building and running a Docker container, and configuring a CI/CD pipeline with GitHub Actions. Additionally, it addresses challenges faced during the process, such as port conflicts, merge conflicts, and authentication issues, and provides solutions to overcome them. The project underscores the importance of containerization and continuous integration in modern software development practices, showcasing the efficiency and reliability of deploying applications through automated workflows.

Objective :

The primary objective of this project is to demonstrate the deployment of a simple HTML web page using Docker and automate the CI/CD pipeline using GitHub Actions. This involves creating a Docker image to containerize the web application and setting up a continuous integration and delivery pipeline to streamline the deployment process.

Tools Used :

1. **Docker:** For containerizing the application.
2. **GitHub:** For version control and hosting the repository.
3. **GitHub Actions:** For automating the CI/CD pipeline.
4. **Nginx:** As the web server to serve the HTML page.
5. **Linux:** As the development environment for executing commands.

Docker:

Docker is a platform that enables developers to build, ship, and run applications in containers. Containers are lightweight, portable units that encapsulate an application and all its dependencies, ensuring consistent performance across different environments.

GitHub and GitHub Actions:

GitHub is a platform for version control and collaboration that allows multiple developers to work on projects simultaneously. GitHub Actions is a CI/CD service provided by GitHub to automate workflows, including building, testing, and deploying applications.

Step-by-Step Implementation:

1. Setting Up the Project Directory:

```
mkdir my-simple-html-page
cd my-simple-html-page
```

```
git init
```

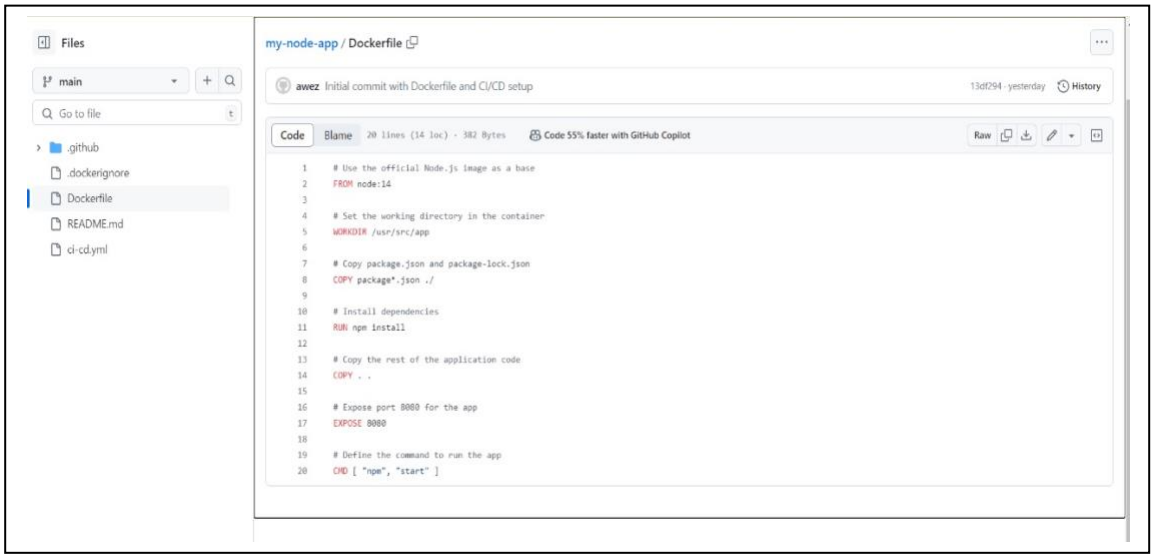
2. Creating the HTML Page :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>My Simple HTML Page</title>
</head>
<body>
    <h1>Welcome to My Simple HTML Page!</h1>
    <p>This is a simple static HTML page served using Nginx
and Docker.</p>
</body>
```

3. Creating the Dockerfile:

```
FROM nginx:alpine
COPY index.html
/usr/share/nginx/html/index.html
EXPOSE 80
```

Dockerfile



4. Creating a .dockerignore File :

```
node_modules
npm-debug.log
Dockerfile
.git
```

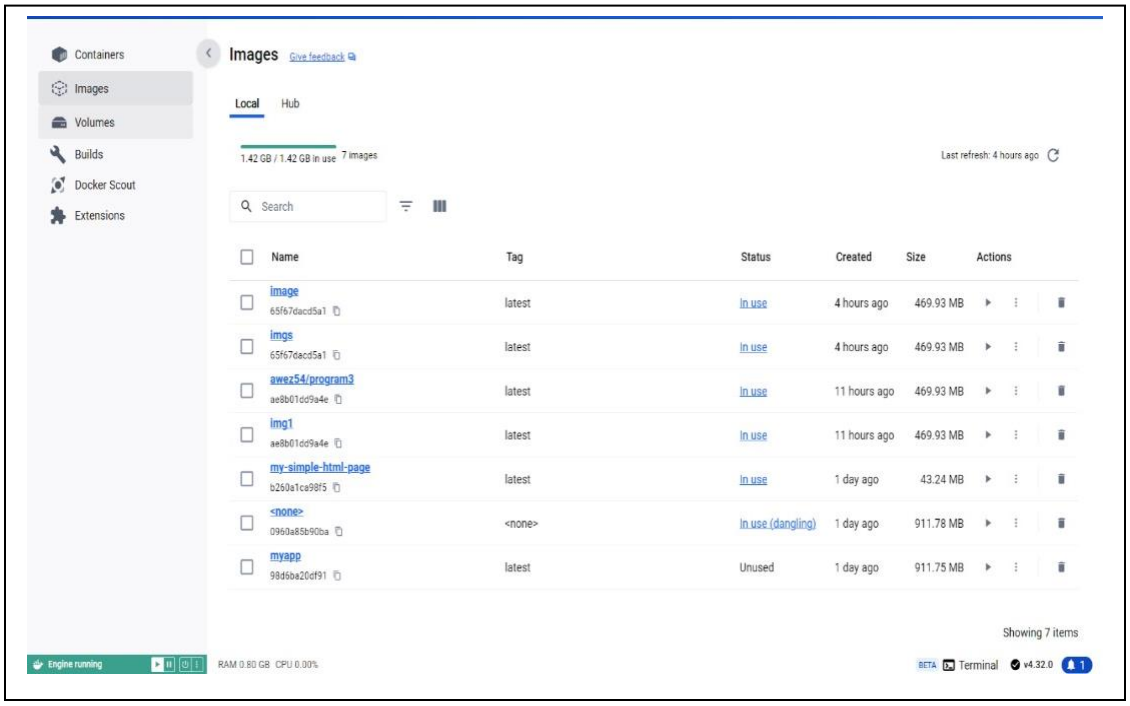
5. Building the Docker Image:

```
docker build -t my-simple-html-page .
```

6. Running the Docker Container:

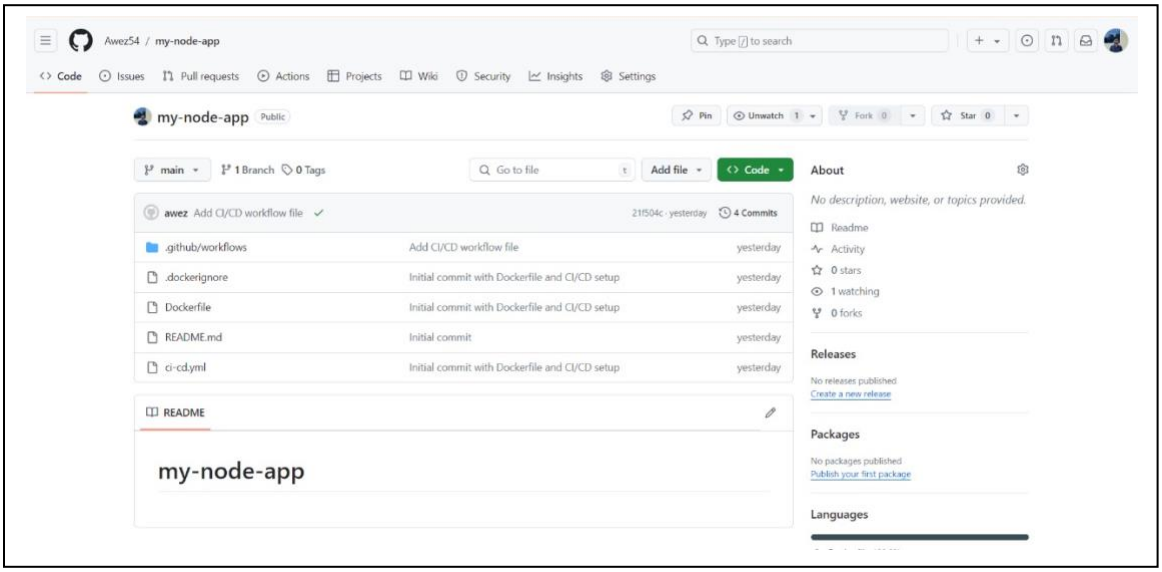
```
docker run -d -p 8080:80 my-simple-html-page
```

Docker Image :



7. Setting Up GitHub Repository:

```
git remote add origin https://github.com/your-username/my-simple-html-page.git
git add .
git commit -m "Initial commit with Dockerfile and HTML page"
```



8. Configuring CI/CD with GitHub Actions:

Directory Structure:

```
.github/
└── workflows/
    └── ci-cd.yml
```

ci-cd.yml:

```
name: CI/CD Pipeline

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up Docker Buildx
      uses: docker/setup-buildx-action@v1

    - name: Login to Docker Hub
      uses: docker/login-action@v1
      with:
        username: ${{ secrets.DOCKER_USERNAME }}
        password: ${{ secrets.DOCKER_PASSWORD }}

    - name: Build and push Docker image
```

uses: docker/build-push-action@v2

with:

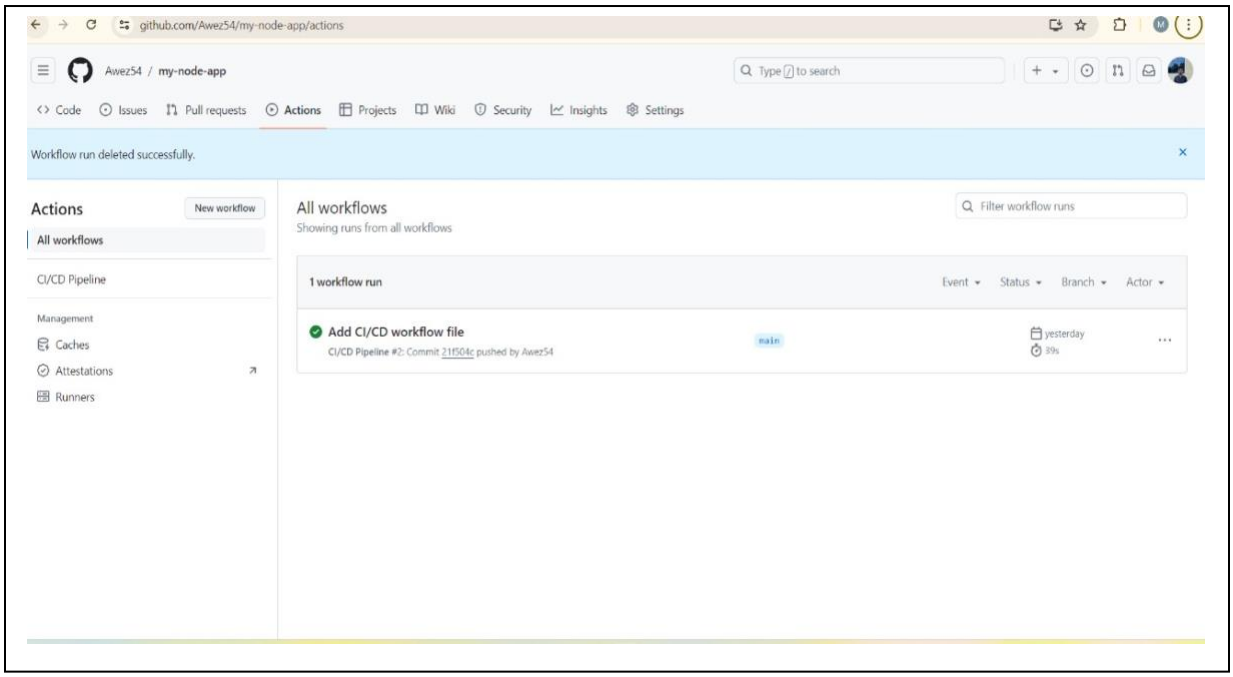
context: .

file: ./Dockerfile

push: true

tags: ${{ secrets.DOCKER_USERNAME }}/my-simple-html-page:latest

CI/CD Pipeline Image :



9. Storing Secrets in GitHub:

- Add Docker Hub credentials to GitHub Secrets:
    - Navigate to your GitHub repository.
    - Go to Settings > Secrets and Variables > Actions.
    - Add DOCKER_USERNAME and DOCKER_PASSWORD secrets.

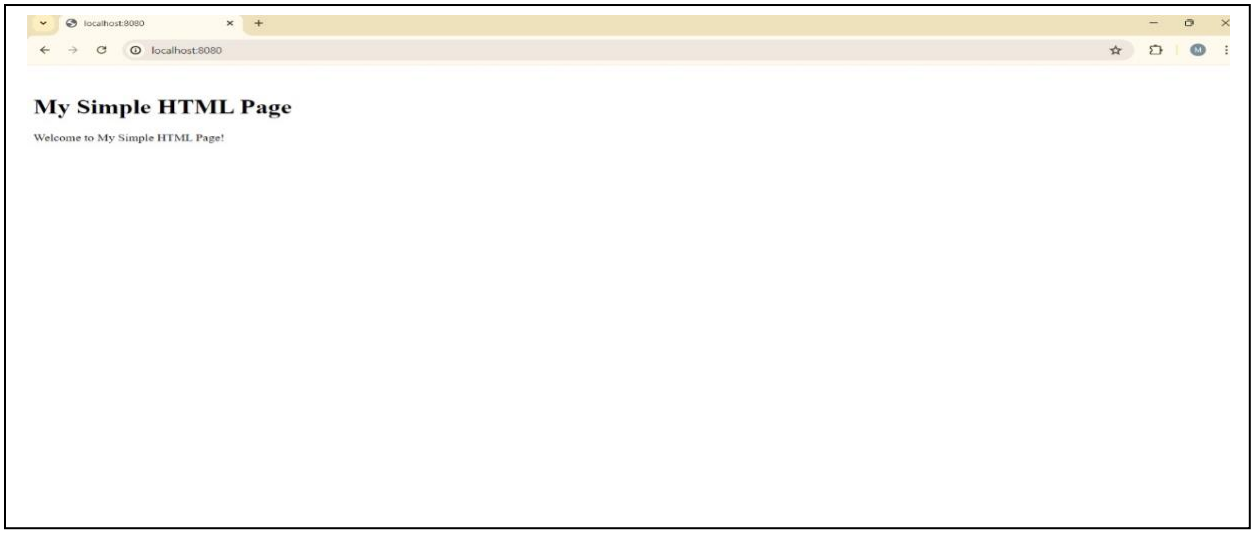
10. Deploying the Application:

```
git add .
git commit -m "Add CI/CD workflow file"
git push
```

11. **Accessing the Application:**

- Access the deployed application:

  Open a web browser and navigate to http://localhost:8080



Conclusion :

This project effectively demonstrates the deployment of a static web page using Docker and GitHub Actions. By containerizing the application with Docker and automating the CI/CD pipeline with GitHub Actions, we achieved a streamlined and efficient workflow for building, deploying, and managing applications. This exercise underscores the importance of containerization and continuous integration in modern software development practices.