DevOps Final Assignment Report

Mohammod Raiyan Buhiyan Loreen

1. Project Overview

This report summarizes the implementation of the DevOps final assignment through the development, containerization, CI/CD automation, and Kubernetes deployment of a web application named **DevBlog**. The project demonstrates a complete DevOps lifecycle from development to deployment using industry-standard tools and platforms such as GitHub Actions, Docker, and Kubernetes (k0s).

2. Tools and Technologies Used

- **Frontend Framework:** React
- **Containerization:** Docker
- **CI/CD:** GitHub Actions
- **Kubernetes Platform:** k0s
- **Source Control:** GitHub

3. How to Run the Project

Locally without Docker

```
npm install
npm start
```

Using Docker

```
docker build -t devopsblog .
docker run -p 5000:5000 devopsblog
```

Pull from Docker Hub

```
docker pull raiyan77/devopsblog
docker run -p 5000:5000 raiyan77/devopsblog
```

4. GitHub and Docker Hub Links

- **GitHub Repository:** https://github.com/RAIYANBHUIYAN/DevOps$_B$*log.git*
- **Docker Hub Image:** https://hub.docker.com/r/raiyan77/devopsblog

5. CI/CD Workflow Explanation

The CI/CD pipeline is defined in the GitHub Actions workflow file located at `.github/workflows/ci.ym`
The workflow:

- Triggers on every push to the `main` branch
- Installs dependencies and builds the React app
- Builds a Docker image
- Optionally pushes the Docker image to Docker Hub

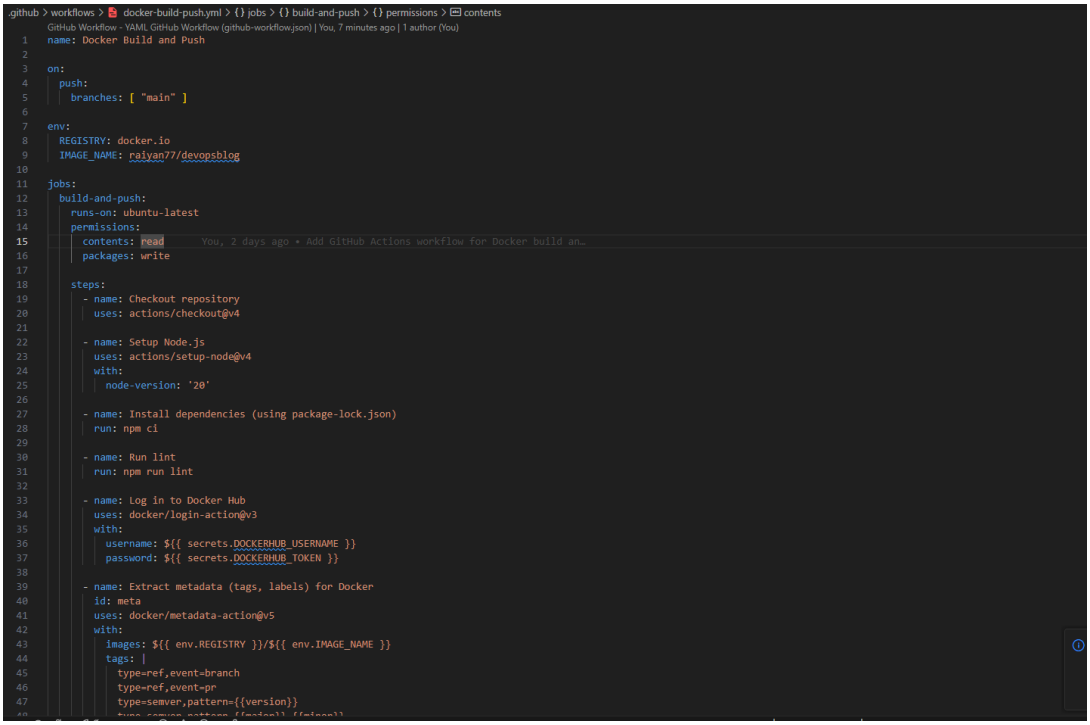
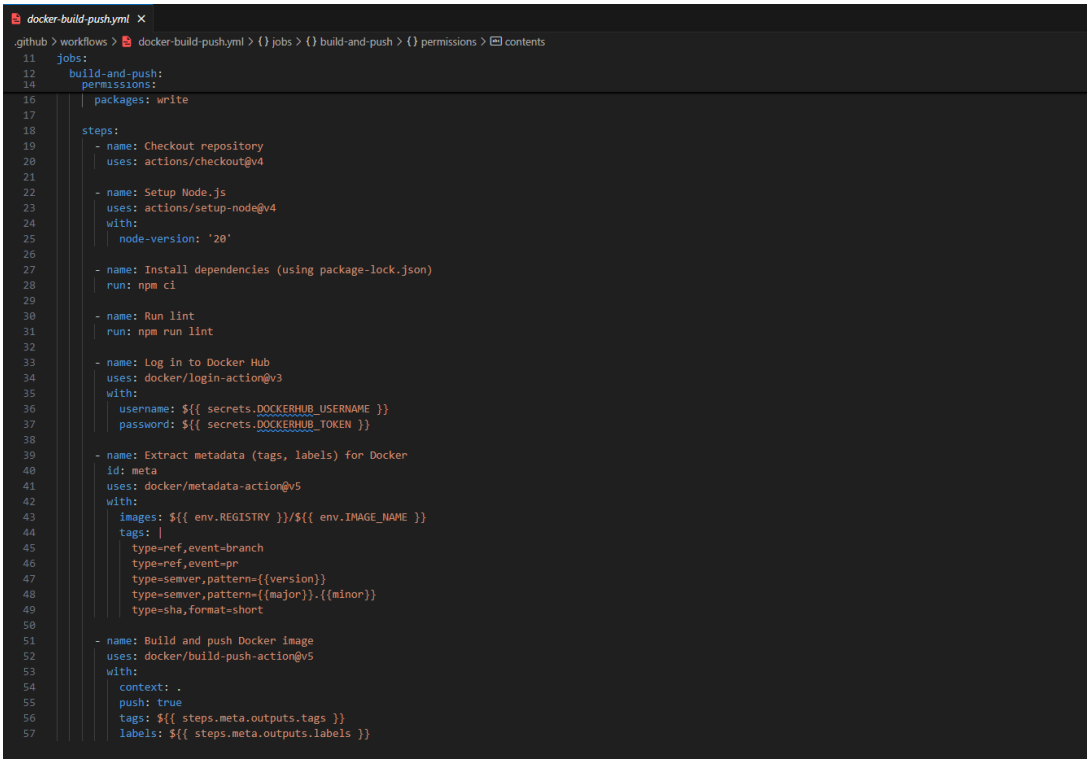


Figure 1: GitHub Actions Workflow File (Part 1)



```yaml
jobs:
  build-and-push:
    permissions:
      packages: write

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'

      - name: Install dependencies (using package-lock.json)
        run: npm ci

      - name: Run lint
        run: npm run lint

      - name: Log in to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}

      - name: Extract metadata (tags, labels) for Docker
        id: meta
        uses: docker/metadata-action@v5
        with:
          images: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}
          tags: |
            type=ref,event=branch
            type=ref,event=pr
            type=semver,pattern={{version}}
            type=semver,pattern={{major}}.{{minor}}
            type=sha,format=short

      - name: Build and push Docker image
        uses: docker/build-push-action@v5
        with:
          context: .
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
```

Figure 2: GitHub Actions Workflow File (Part 2)

6. Screenshots

Project Structure

Running Website

Docker Configuration

Docker Hub Image

GitHub Action Successful Run

Kubernetes Deployment

7. Learnings and Challenges

What I Learned:

- Setting up a CI pipeline using GitHub Actions
- Containerizing a web app with Docker
- Creating Kubernetes manifests and deploying locally using k0s
- Managing cloud-ready infrastructure in a DevOps environment

Challenges Faced:

- Initially configuring GitHub Actions with React builds

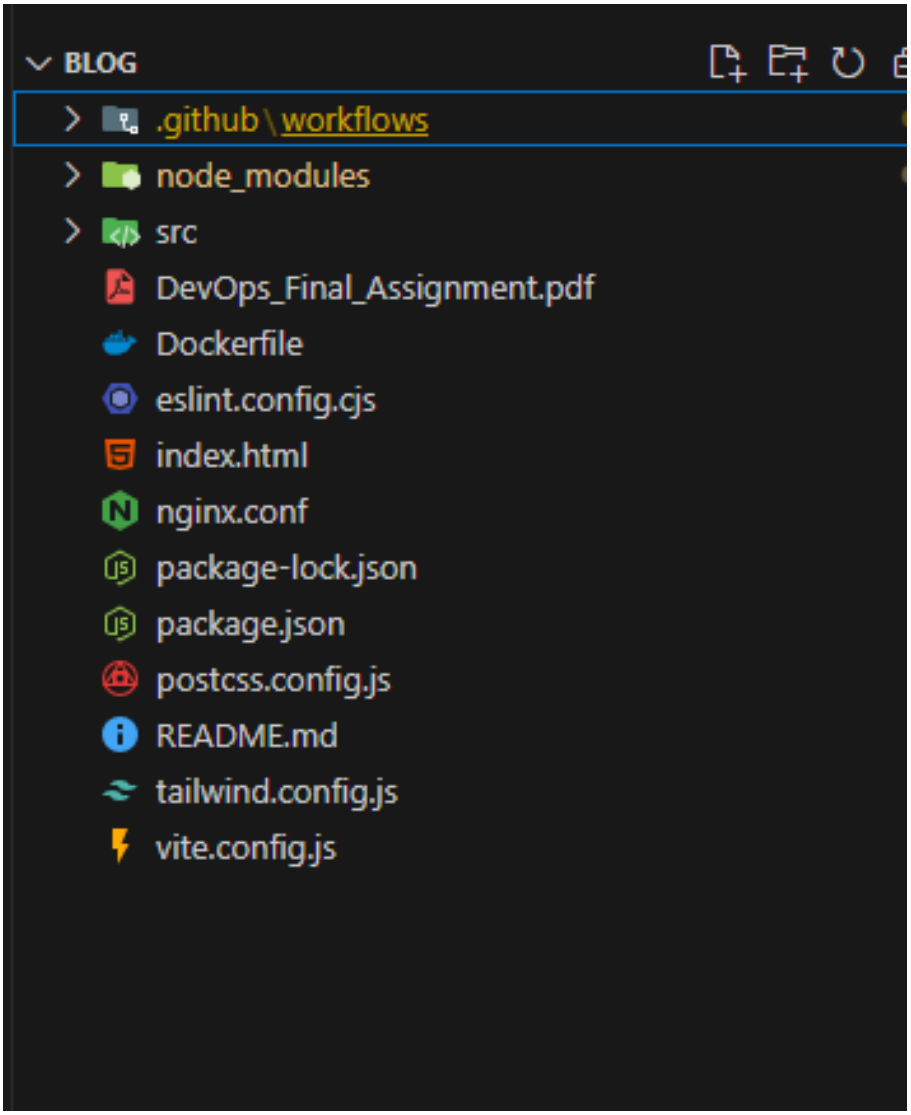


Figure 3: Project Folder Structure

- Debugging Dockerfile base image compatibility
- Managing Kubernetes port exposure and verifying the deployment status

Potential Improvements:

- Integrate unit tests for automation in CI pipeline
- Add monitoring and logging with Prometheus/Grafana
- Set up production-level deployment on cloud Kubernetes clusters

8. Video Explanation

A complete video explanation of this project can be found here: https://youtu.be/13NoAvofQ88

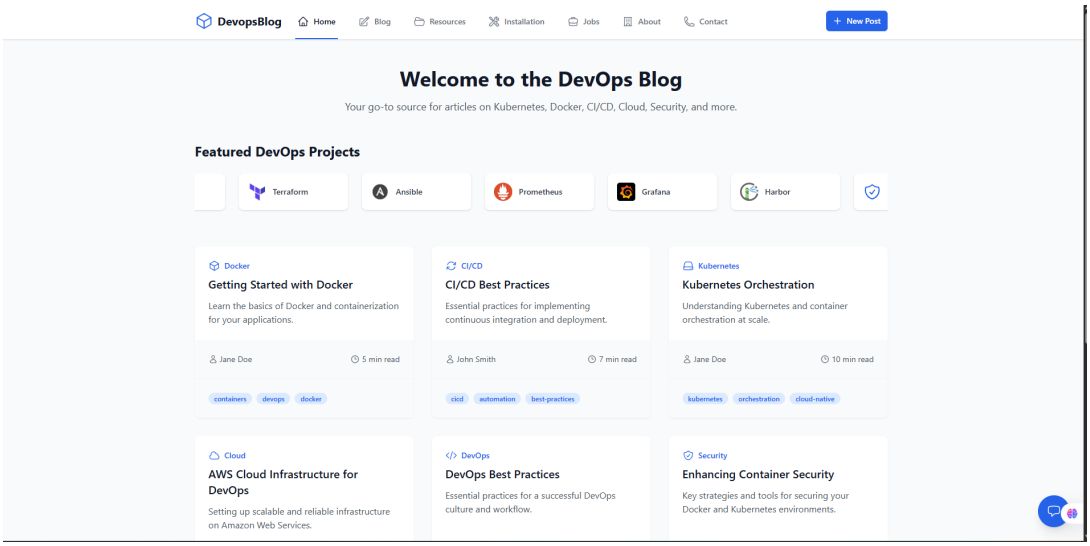


Figure 4: Blog Homepage

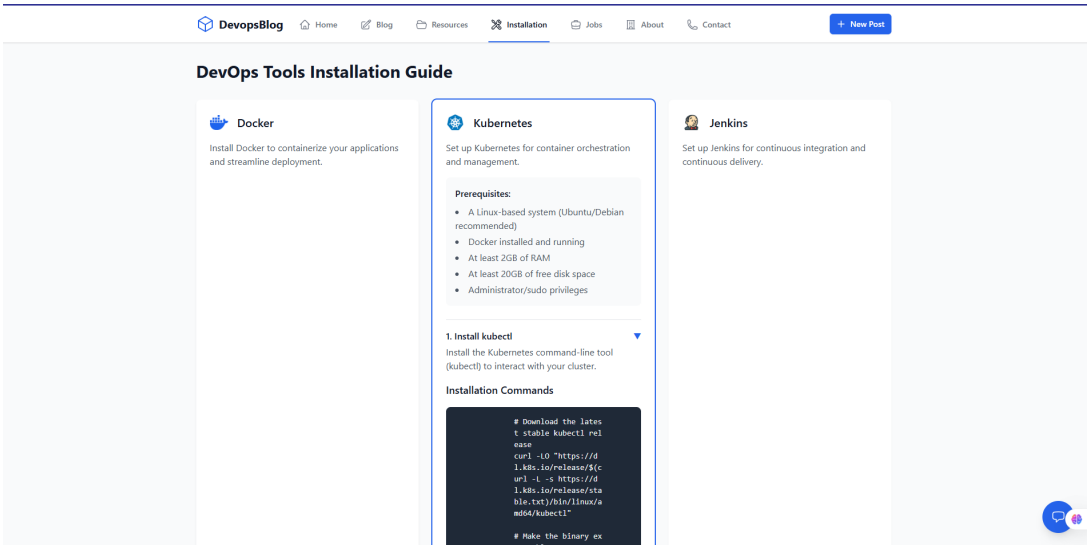


Figure 5: Blog Post Section

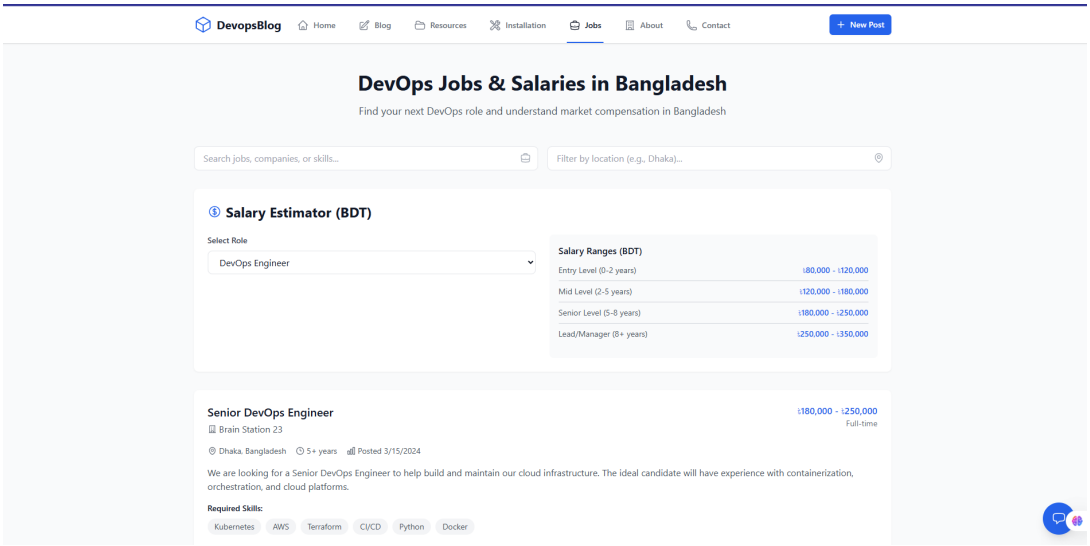


Figure 6: Additional Content View

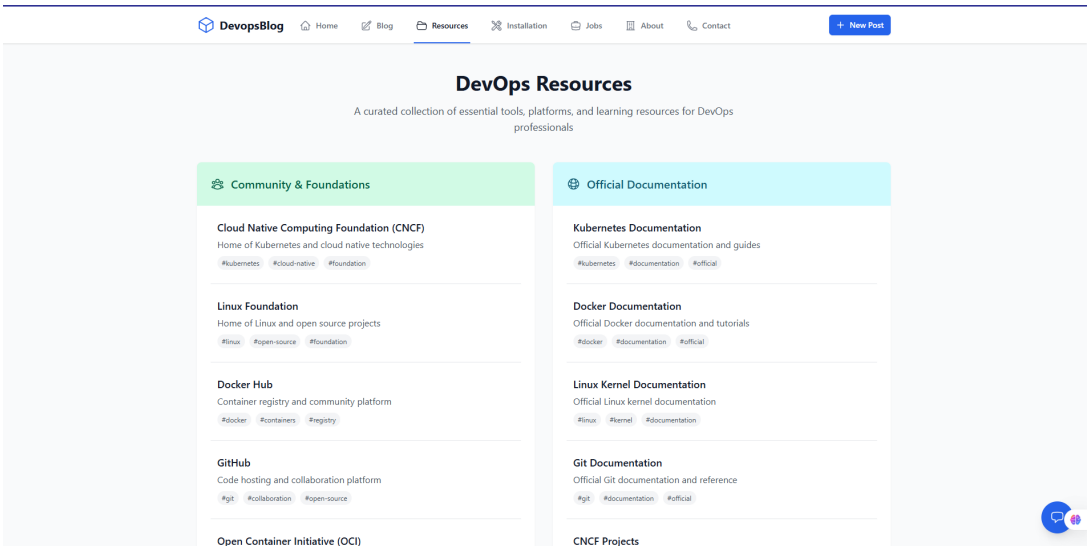
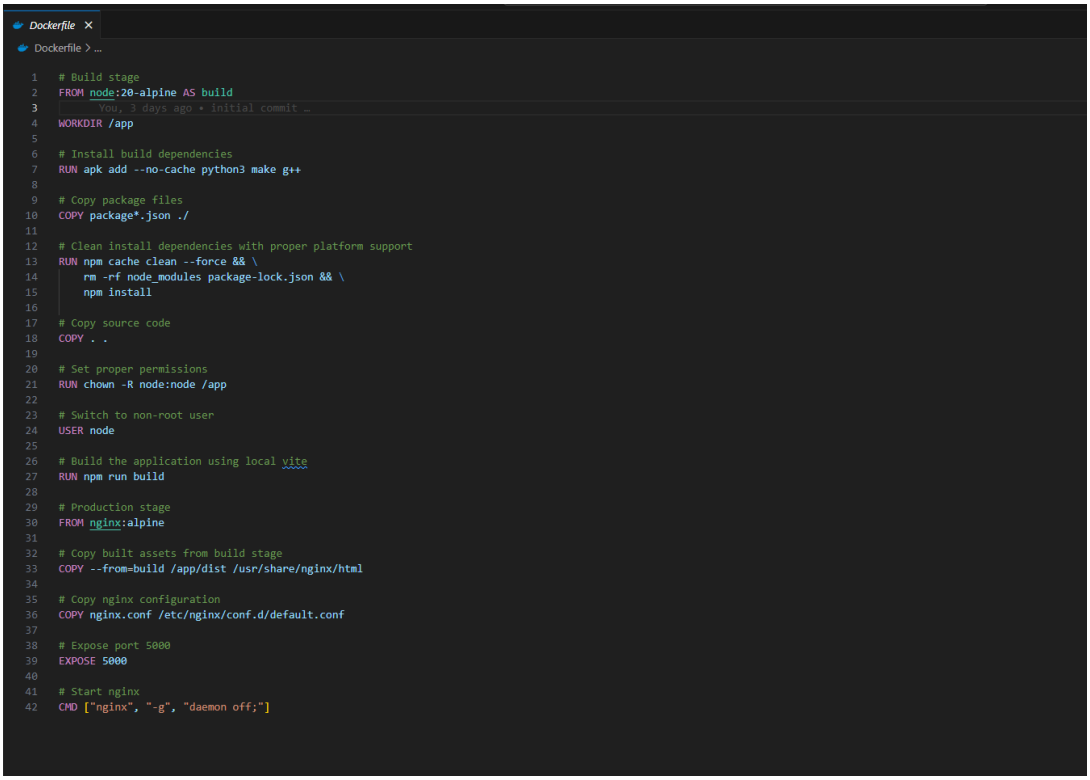


Figure 7: Mobile Responsiveness



```
Dockerfile
Dockerfile > ...

# Build stage
FROM node:20-alpine AS build
You, 3 days ago • initial commit …
WORKDIR /app

# Install build dependencies
RUN apk add --no-cache python3 make g++

# Copy package files
COPY package*.json ./

# Clean install dependencies with proper platform support
RUN npm cache clean --force && \
    rm -rf node_modules package-lock.json && \
    npm install

# Copy source code
COPY . .

# Set proper permissions
RUN chown -R node:node /app

# Switch to non-root user
USER node

# Build the application using local vite
RUN npm run build

# Production stage
FROM nginx:alpine

# Copy built assets from build stage
COPY --from=build /app/dist /usr/share/nginx/html

# Copy nginx configuration
COPY nginx.conf /etc/nginx/conf.d/default.conf

# Expose port 5000
EXPOSE 5000

# Start nginx
CMD ["nginx", "-g", "daemon off;"]
```

Figure 8: Dockerfile

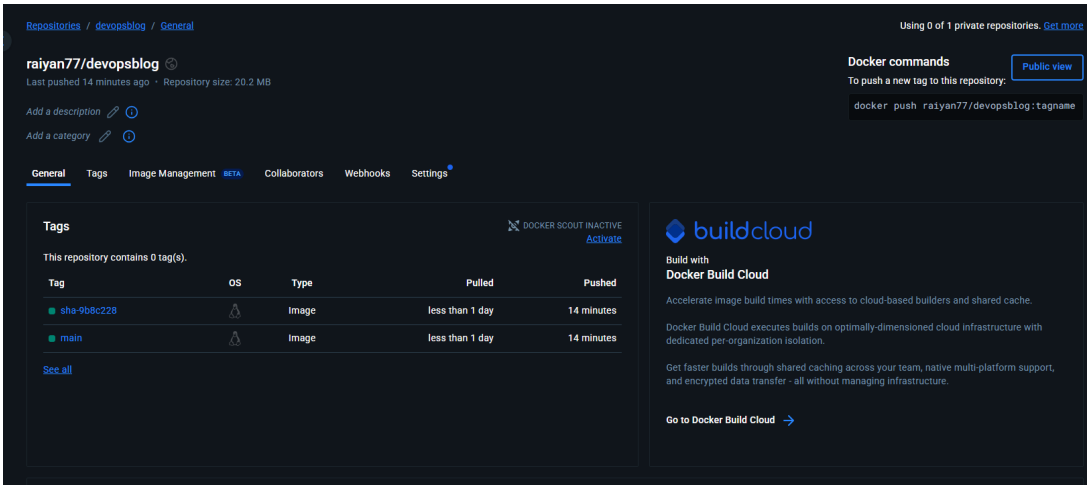


Figure 9: Docker Hub Repository

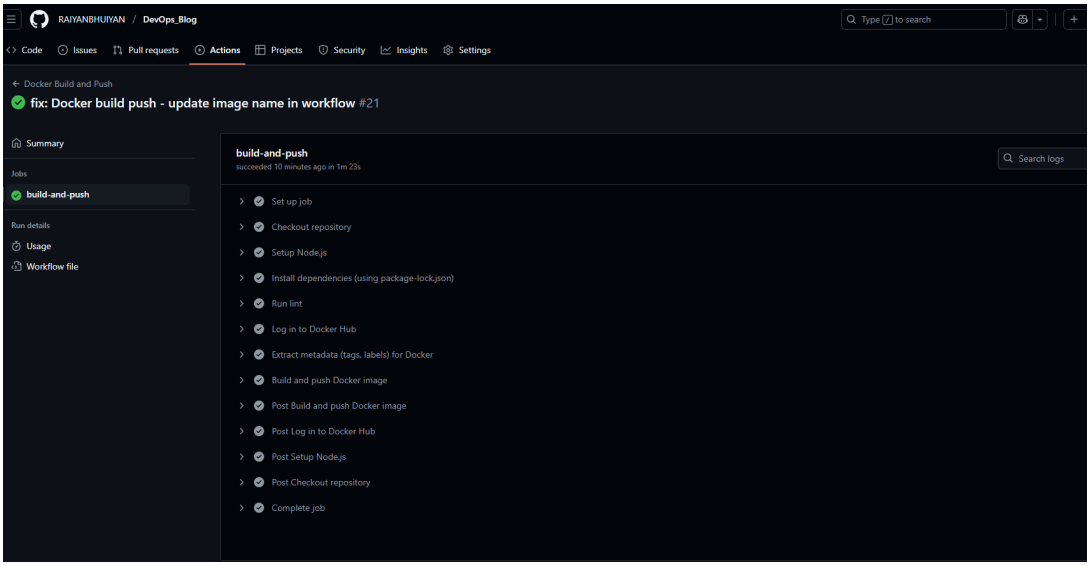


Figure 10: Successful GitHub Actions Run

```
root@raiyan-VirtualBox:/home/raiyan# k0s kubectl get pods
NAME                                   READY   STATUS    RESTARTS      AGE
devopsblog-deployment-754664854b-svnvm   1/1     Running   1 (16m ago)   8h
root@raiyan-VirtualBox:/home/raiyan# k0s kubectl get svc
NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
devopsblog-service   NodePort    10.104.103.63   <none>        80:30007/TCP   9h
kubernetes           ClusterIP   10.96.0.1       <none>        443/TCP        3d21h
root@raiyan-VirtualBox:/home/raiyan#
```

Figure 11: k0s Pods and Service Running

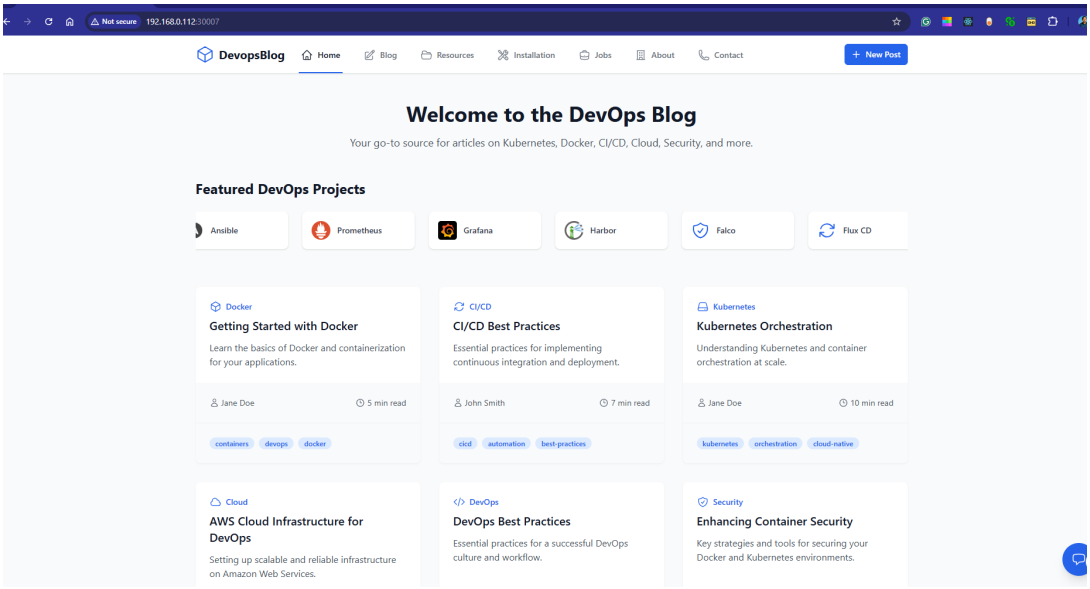


Figure 12: Application Running via k0s

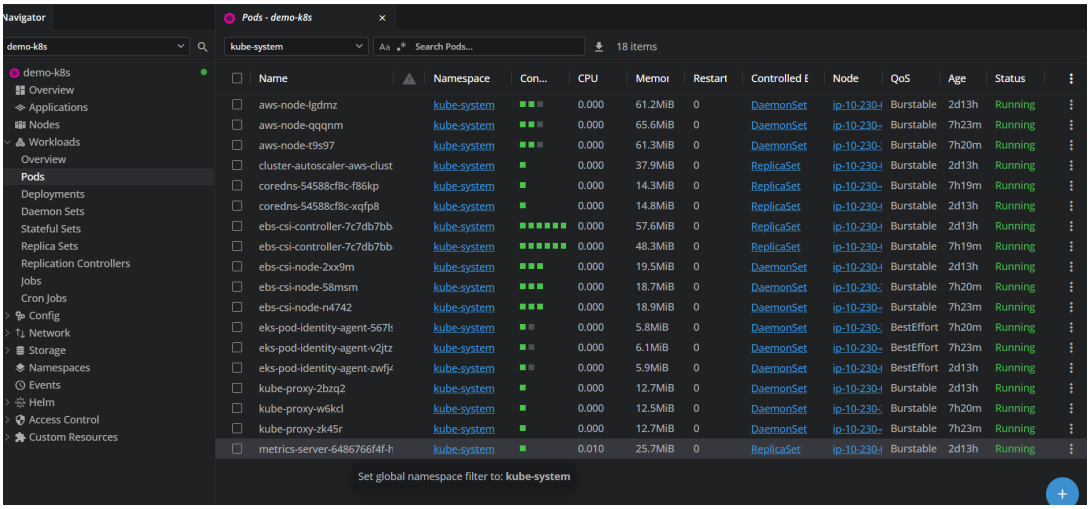


Figure 13: Pods Visualization using Lens