# CI/CD Pipeline for Flask App using GitHub Actions and KIND

## 1. Introduction

This project demonstrates the implementation of a CI/CD pipeline using GitHub Actions to automate the building and deployment of a Flask application on a Kubernetes cluster (KIND) running locally. The pipeline ensures that any changes pushed to the repository are automatically built, tested, and deployed.

## 2. Objectives

• Automate the build and deployment process for a Flask application.  
• Use GitHub Actions to integrate CI/CD workflows.  
• Deploy the application on a local Kubernetes cluster using KIND.  
• Ensure seamless updates when new changes are pushed.

## 3. Tools and Technologies Used

• **GitHub Actions** – Automates the CI/CD pipeline.  
• **Docker** – Containerizes the Flask application.  
• **Docker** **Hub** – Stores the application image.  
• **KIND** (Kubernetes in Docker)\*\* – Runs a Kubernetes cluster locally.  
• **Kubectl** – CLI tool to manage Kubernetes clusters.

## 4. Implementation

### 4.1 Setting Up the CI/CD Pipeline

The GitHub Actions workflow automates the following steps:  
1. Trigger on Push: Runs the pipeline when changes are pushed to the `main` branch.  
2. Build Docker Image: The Flask app is containerized.  
3. Push to Docker Hub: The image is uploaded for deployment.  
4.Deploy to KIND: KIND loads the updated image and applies Kubernetes manifests.

### 

### 4.2 Deploying the Application Locally

Ensure you have the required tools installed before proceeding.

### Step 1: Install Required Tools

```sh  
# Install KIND  
curl -Lo ./kind https://kind.sigs.k8s.io/dl/latest/kind-linux-amd64  
chmod +x ./kind  
sudo mv ./kind /usr/local/bin/kind  
  
# Install kubectl  
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"  
chmod +x kubectl  
sudo mv kubectl /usr/local/bin/kubectl  
```

### Step 2: Deploy the Application

```sh  
kind create cluster --name flask-cluster  
docker pull nihal009/flask-app:v1  
kind load docker-image nihal009/flask-app:v1 --name flask-cluster  
kubectl apply -f k8s/deployment.yml  
kubectl apply -f k8s/service.yml  
kubectl get pods  
kubectl get services  
```

## 

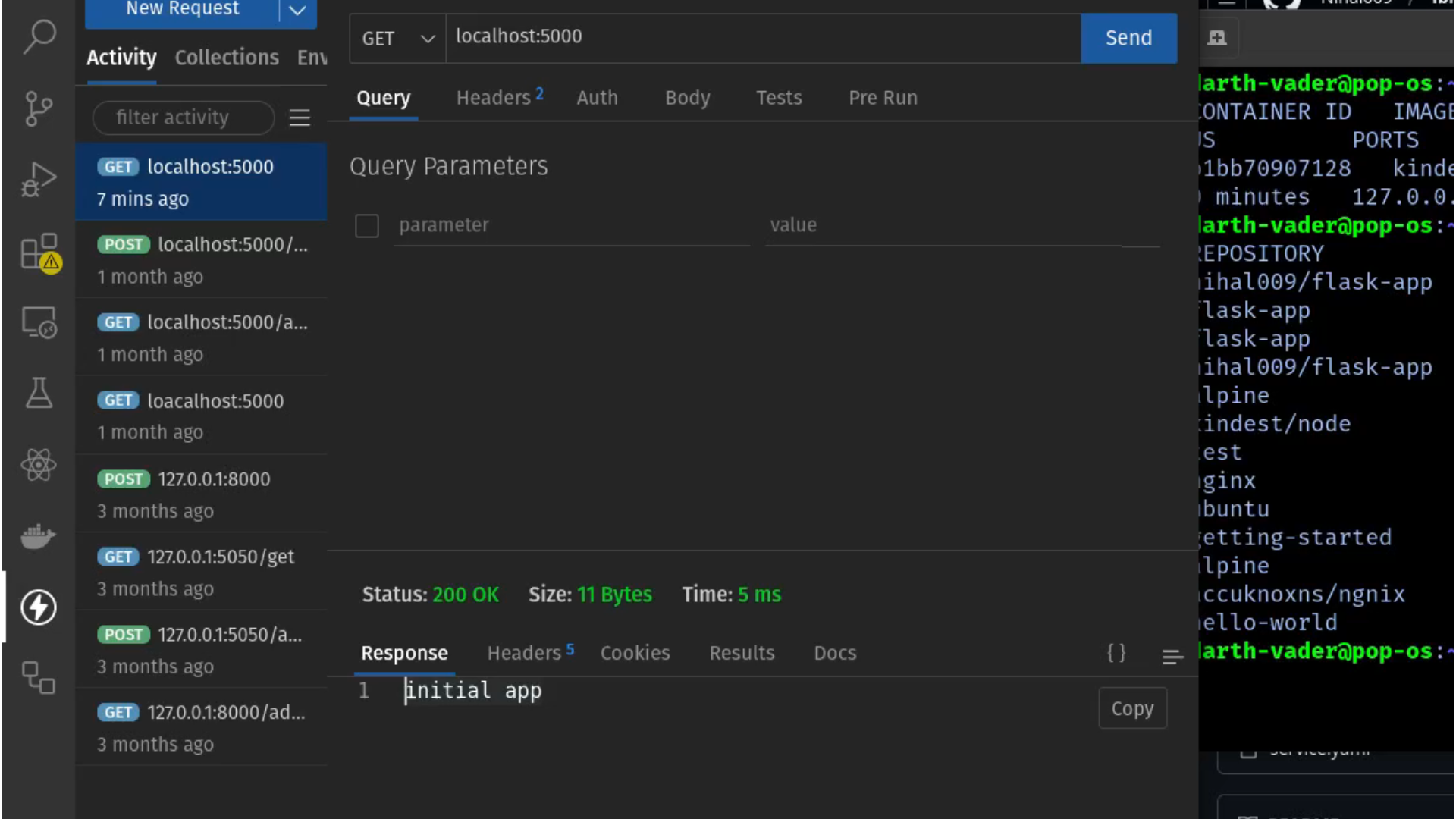
## 5. Testing the Deployment

Forward the port to access the Flask app:

```sh  
kubectl port-forward svc/flask-app 5000:5000  
```

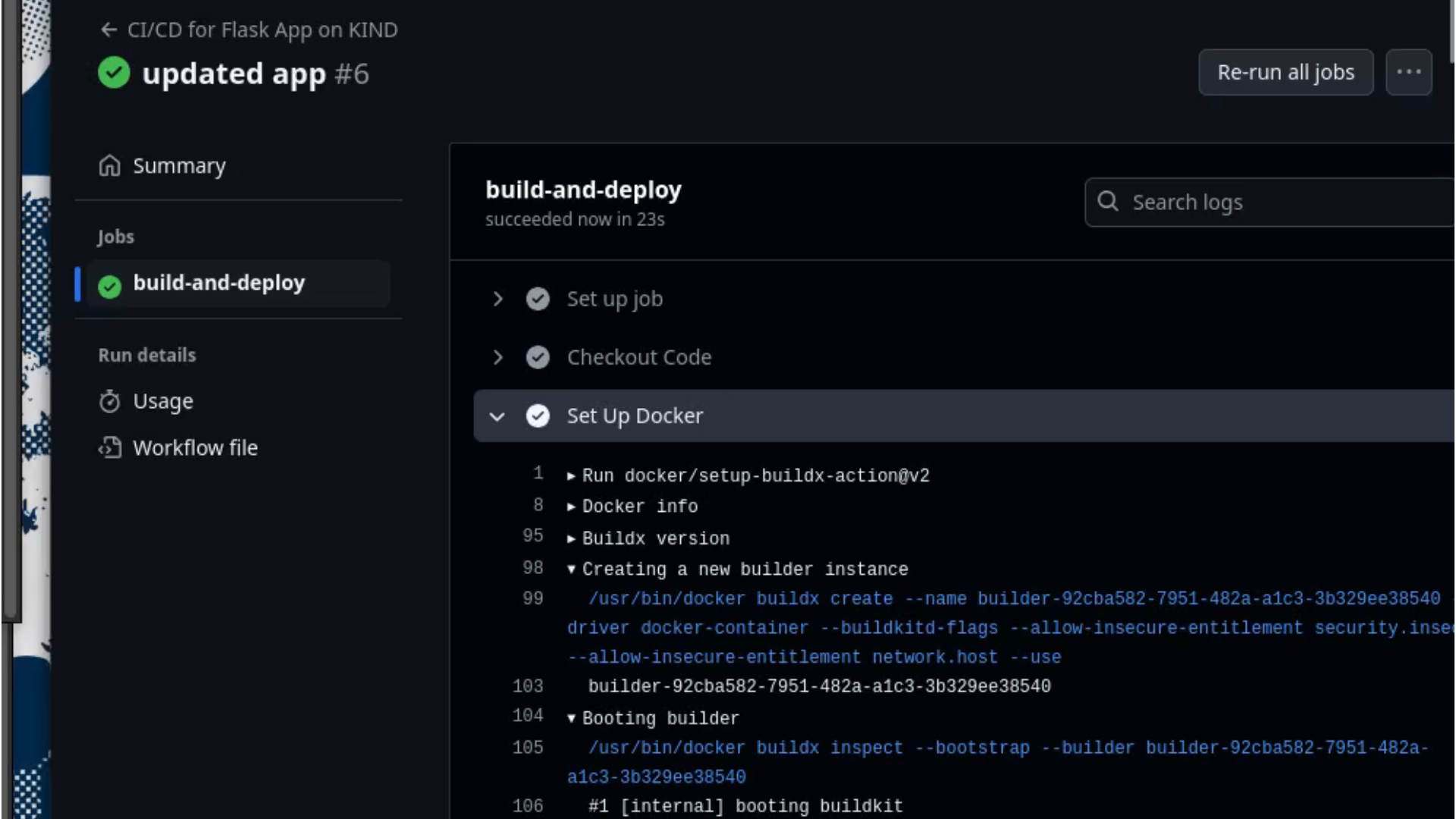
Now, open a browser and visit: `http://localhost:5000`.

## 



## 6. Updating the Application

Whenever new changes are pushed:  
1. GitHub Actions automatically builds and pushes a new Docker image.  
2. Pull the updated image and restart the deployment:  
```sh  
docker pull nihal009/flask-app:v1  
kind load docker-image nihal009/flask-app:v1 --name flask-cluster  
kubectl rollout restart deployment flask-app  
```



## 

## 7. Conclusion

This project successfully implements a CI/CD pipeline for a Flask app using GitHub Actions and KIND. By automating the build and deployment process, it ensures:  
• Faster application updates.  
• Seamless integration with Kubernetes.  
• A streamlined workflow for local testing before deploying to production.

## 8. Future Enhancements

• Implement \*\*Helm charts\*\* for better deployment management.  
• Add \*\*unit testing\*\* in GitHub Actions before deploying.  
• Automate scaling using \*\*Kubernetes Horizontal Pod Autoscaler\*\*.