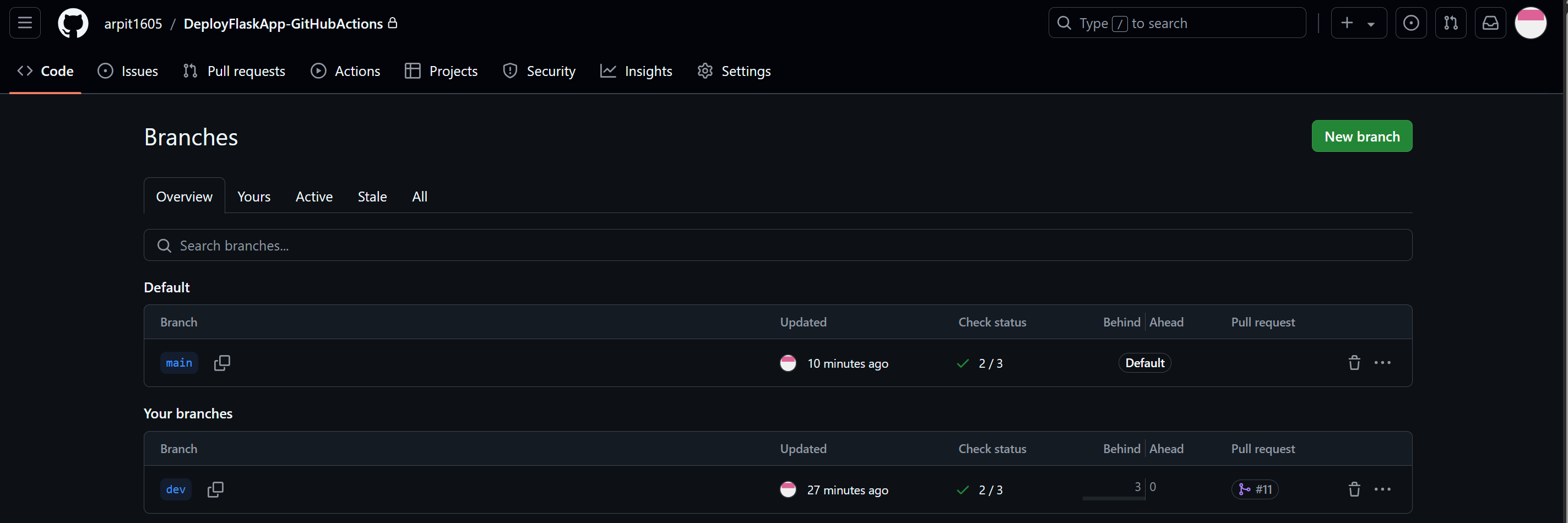
**Implement a CI/CD workflow using GitHub Actions for a Flask based Python web application**

**1. Setup a simple GitHub repository for Python Application using Flask**

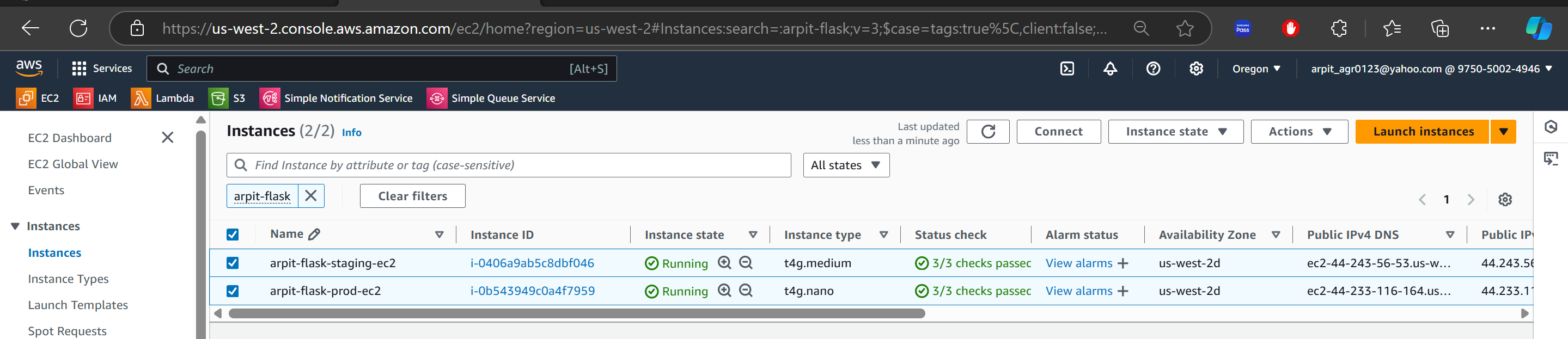
<https://www.github.com/arpit1605/DeployFlaskApp-GitHubActions>

The repository has two branches: **main** for production deployment and **dev** for staging deployment.



**2. EC2 Setup:**

We have setup two EC2 instances, one for staging and one for production.

****

**3. Workflow Overview:**

The workflow performs the following steps:

* Install Dependencies: It installs required Python packages.
* Run Tests: Executes the test suite using pytest.
* Build: Prepares the app for deployment.
* Deploy to Staging: Automatically deploys the app to a staging environment when a push occurs to the dev branch.
* Deploy to Production: Automatically deploys the app to production when a push occurs to the main branch or a new release is tagged.

**4. GitHub Actions Workflow for CI/CD:**

Setup the GitHub Actions workflow to automate the process of installing dependencies, running tests, and deploying the Flask app to the EC2 instance.

Create .github/workflows/CI-CD.yml

**GitHub Actions Workflow yml code:**

name: CI/CD for hosting a Flask-based Python Web Application

on:

push:

branches:

- dev # Trigger staging deployment on dev branch push

- main # Trigger production deployment on main branch push

*#Please note that the above line can be removed if you want to deploy changes to production only when there is a new release published.*

release:

types: [published] # Trigger production deployment on release

jobs:

# Step 1: Install dependencies, run tests, and build on the GitHub Runner (not on EC2)

test-and-build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Set up Python

uses: actions/setup-python@v4

with:

python-version: '3.x'

- name: Install dependencies globally

run: |

python -m pip install --upgrade pip

pip install --user -r requirements.txt # Install dependencies globally on the GitHub runner

- name: Run unit tests

run: |

pytest

env:

CI: true

- name: Build the application

run: |

echo "Building the application"

# Step 2: Deploy to Staging (for dev branch)

deploy-to-staging:

if: github.ref == 'refs/heads/dev' # Deploy only if it's a push to the dev branch

runs-on: ubuntu-latest

needs: test-and-build

steps:

- name: Deploy to Staging EC2 Instance

run: |

mkdir -p ~/.ssh

echo "${{ secrets.STAGING\_EC2\_SSH\_KEY }}" > ~/.ssh/id\_rsa

chmod 600 ~/.ssh/id\_rsa

ssh -o StrictHostKeyChecking=no -i ~/.ssh/id\_rsa ${{ secrets.STAGING\_EC2\_USERNAME }}@${{ secrets.STAGING\_EC2\_HOST }} << 'EOF'

set -e

echo "Starting deployment to STAGING..."

# Remove the directory if it exists

if [ -d "/home/${{ secrets.STAGING\_EC2\_USERNAME }}/SimpleFlaskApp" ]; then

rm -rf /home/${{ secrets.STAGING\_EC2\_USERNAME }}/SimpleFlaskApp

fi

# Clone the repository

git clone https://github.com/arpit1605/SimpleFlaskApp.git /home/${{ secrets.STAGING\_EC2\_USERNAME }}/SimpleFlaskApp

cd /home/${{ secrets.STAGING\_EC2\_USERNAME }}/SimpleFlaskApp

# Update and install dependencies using apt

sudo apt-get update

sudo apt-get install -y python3-flask

sudo apt-get install -y gunicorn

# Run Gunicorn in the background using nohup

nohup /usr/bin/gunicorn --workers 3 app:app --bind 0.0.0.0:${{ secrets.STAGING\_PORT }} > gunicorn.log 2>&1 &

echo "Deployment completed, Gunicorn is running in the background"

EOF

env:

STAGING\_EC2\_HOST: ${{ secrets.STAGING\_EC2\_HOST }}

STAGING\_EC2\_SSH\_KEY: ${{ secrets.STAGING\_EC2\_SSH\_KEY }}

STAGING\_EC2\_USERNAME: ${{ secrets.STAGING\_EC2\_USERNAME }}

STAGING\_PORT: ${{ secrets.STAGING\_PORT }}

# Step 3: Deploy to Production (for main branch or release)

deploy-to-production:

if: github.ref == 'refs/heads/main' || github.event\_name == 'release' # Deploy to production if it's a push to main or a release

runs-on: ubuntu-latest

needs: test-and-build

steps:

- name: Deploy to Production EC2 Instance

run: |

mkdir -p ~/.ssh

echo "${{ secrets.PRODUCTION\_EC2\_SSH\_KEY }}" > ~/.ssh/id\_rsa

chmod 600 ~/.ssh/id\_rsa

ssh -o StrictHostKeyChecking=no -i ~/.ssh/id\_rsa ${{ secrets.PRODUCTION\_EC2\_USERNAME }}@${{ secrets.PRODUCTION\_EC2\_HOST }} << 'EOF'

set -e

echo "Starting deployment to PRODUCTION..."

# Remove the directory if it exists

if [ -d "/home/${{ secrets.PRODUCTION\_EC2\_USERNAME }}/SimpleFlaskApp" ]; then

rm -rf /home/${{ secrets.PRODUCTION\_EC2\_USERNAME }}/SimpleFlaskApp

fi

# Clone the repository

git clone https://github.com/arpit1605/SimpleFlaskApp.git /home/${{ secrets.PRODUCTION\_EC2\_USERNAME }}/SimpleFlaskApp

cd /home/${{ secrets.PRODUCTION\_EC2\_USERNAME }}/SimpleFlaskApp

# Update and install dependencies using apt

sudo apt-get update

sudo apt-get install -y python3-flask

sudo apt-get install -y gunicorn

# Run Gunicorn in the background using nohup

nohup /usr/bin/gunicorn --workers 3 app:app --bind 0.0.0.0:${{ secrets.PRODUCTION\_PORT }} > gunicorn.log 2>&1 &

echo "Deployment completed, Gunicorn is running in the background"

EOF

env:

PRODUCTION\_EC2\_HOST: ${{ secrets.PRODUCTION\_EC2\_HOST }}

PRODUCTION\_EC2\_SSH\_KEY: ${{ secrets.PRODUCTION\_EC2\_SSH\_KEY }}

PRODUCTION\_EC2\_USERNAME: ${{ secrets.PRODUCTION\_EC2\_USERNAME }}

PRODUCTION\_PORT: ${{ secrets.PRODUCTION\_PORT }}

**5. Environment Secrets:**

To securely store sensitive data, use GitHub Secrets. Navigate to your GitHub repository’s settings and add the following secrets:

**For STAGING environment:**

STAGING\_EC2\_HOST: The IP address or hostname of the staging EC2 instance.

STAGING\_EC2\_USERNAME: The username to SSH into the staging EC2 instance.

STAGING\_EC2\_SSH\_KEY: The private SSH key for authenticating with the staging EC2 instance.

STAGING\_PORT: The port on which Gunicorn will bind on the staging environment.

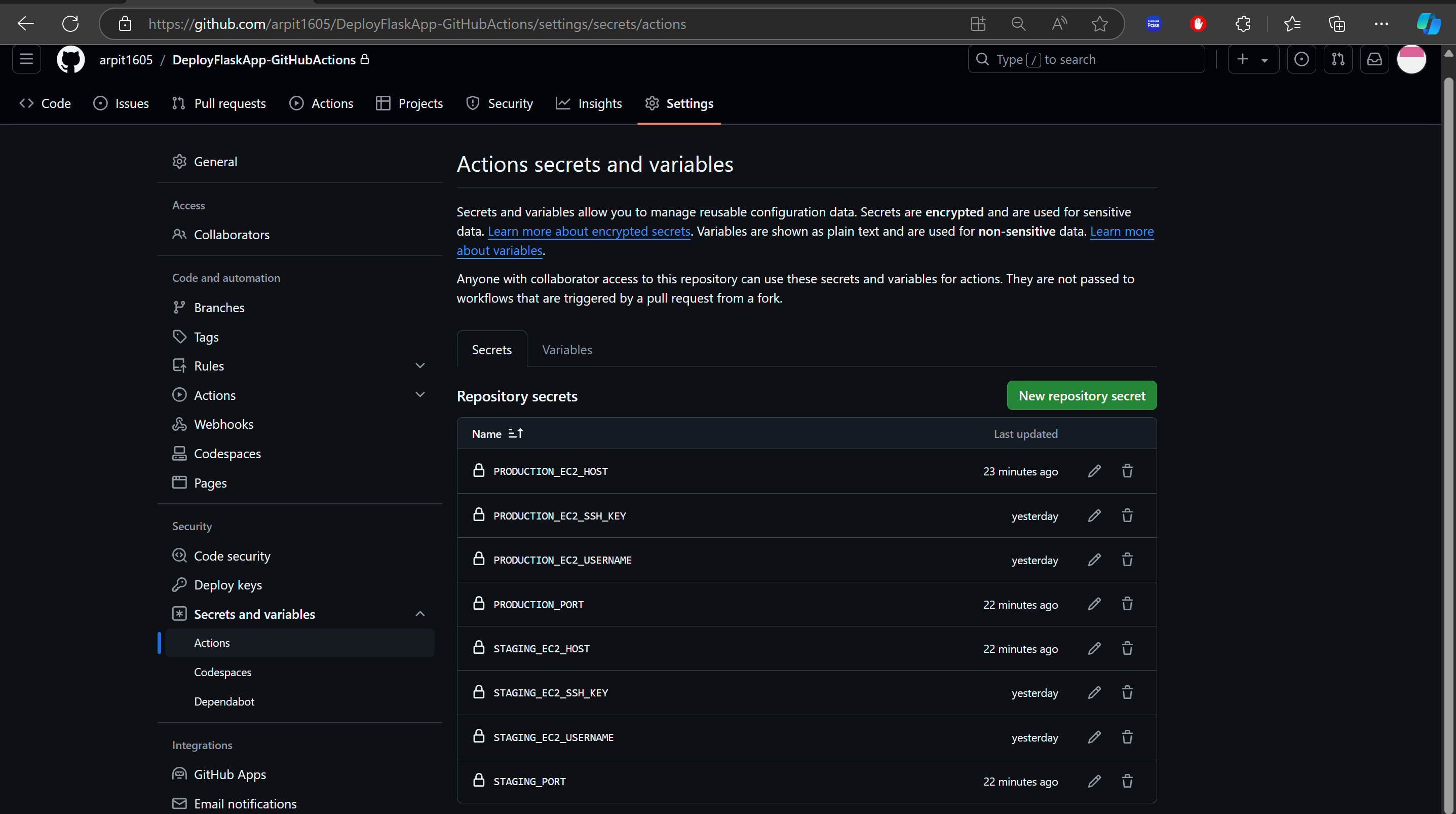
**For PRODUCTION environment:**

PRODUCTION\_EC2\_HOST: The IP address or hostname of the production EC2 instance.

PRODUCTION\_EC2\_USERNAME: The username to SSH into the production EC2 instance.

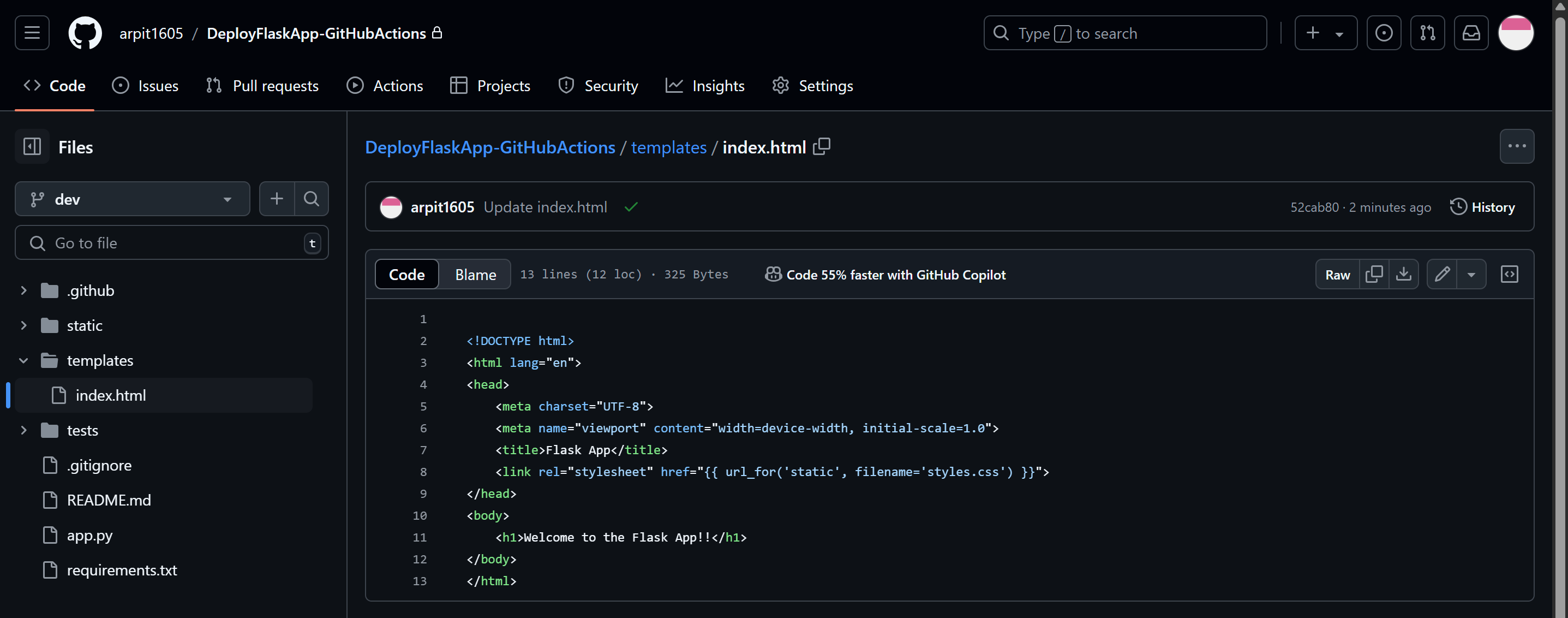
PRODUCTION\_EC2\_SSH\_KEY: The private SSH key for authenticating with the production EC2 instance.

PRODUCTION\_PORT: The port on which Gunicorn will bind on the production environment.

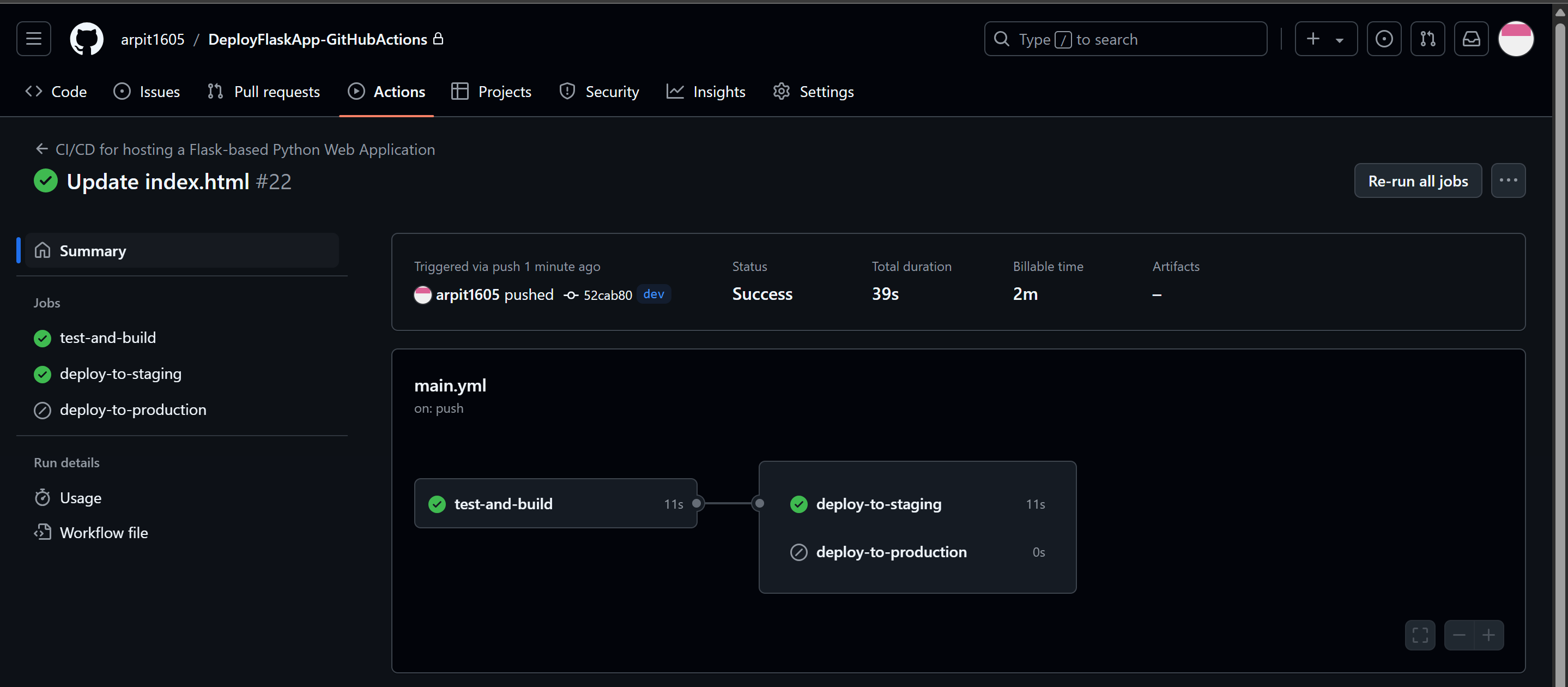
****

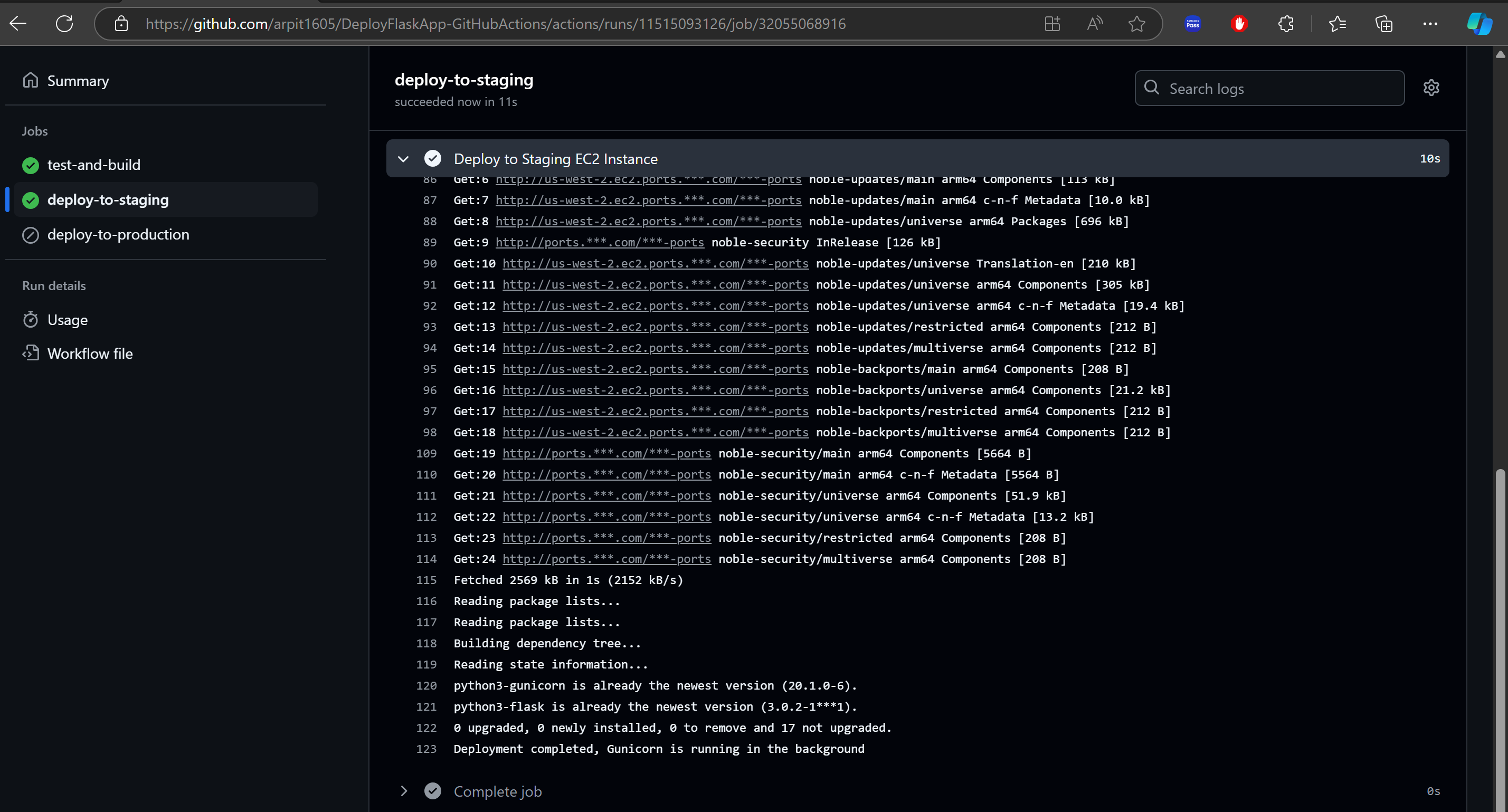
# Deployment to Staging Environment

**Made changes to index.html file in the ‘dev’ branch:**

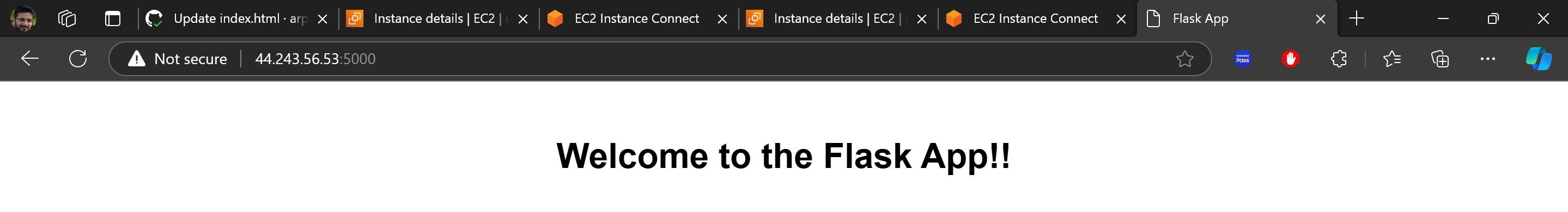
****

**The GitHub Action workflow is completed as shown below:**



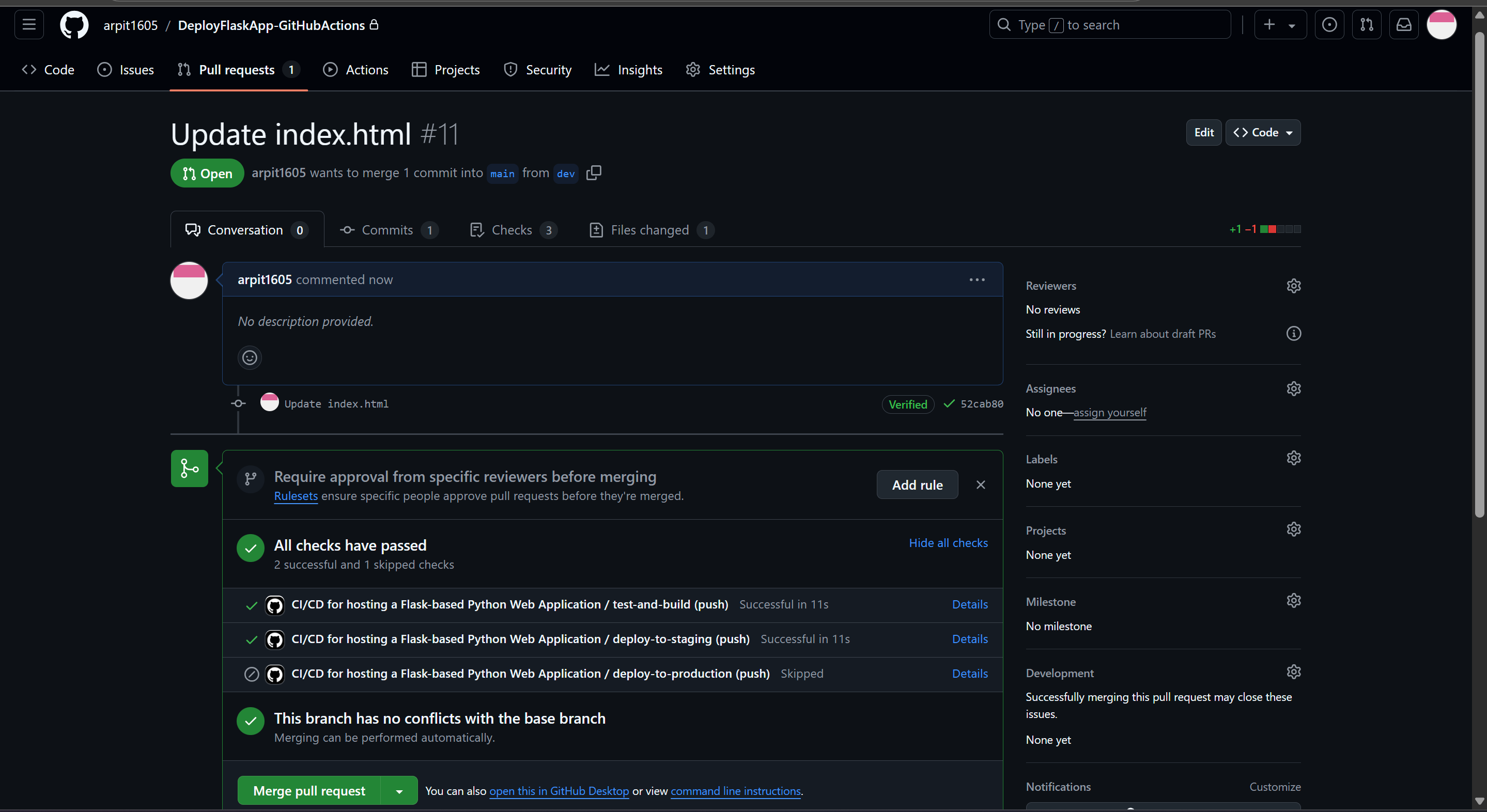
The changes are deployed successfully:

The Flask based Python web application is accessible with public IP address of staging Amazon EC2 instance as shown below:

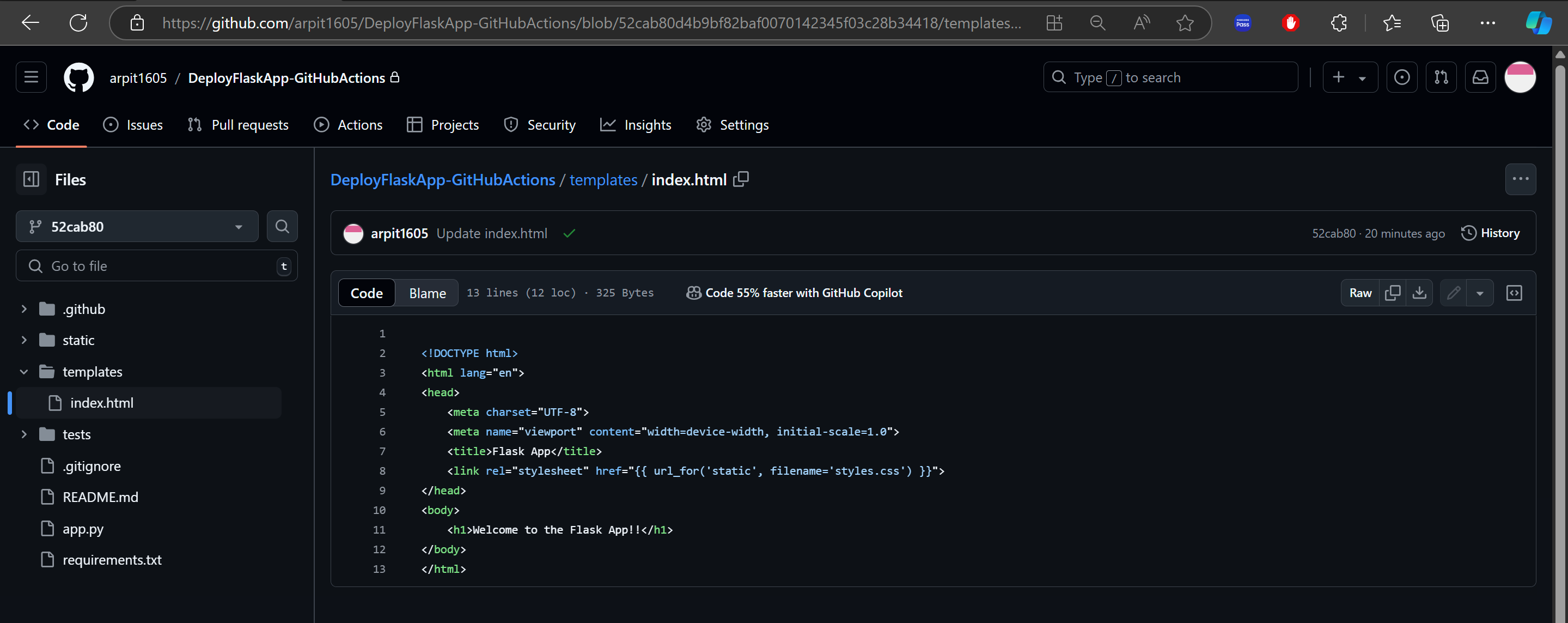


# Deployment to Production Environment

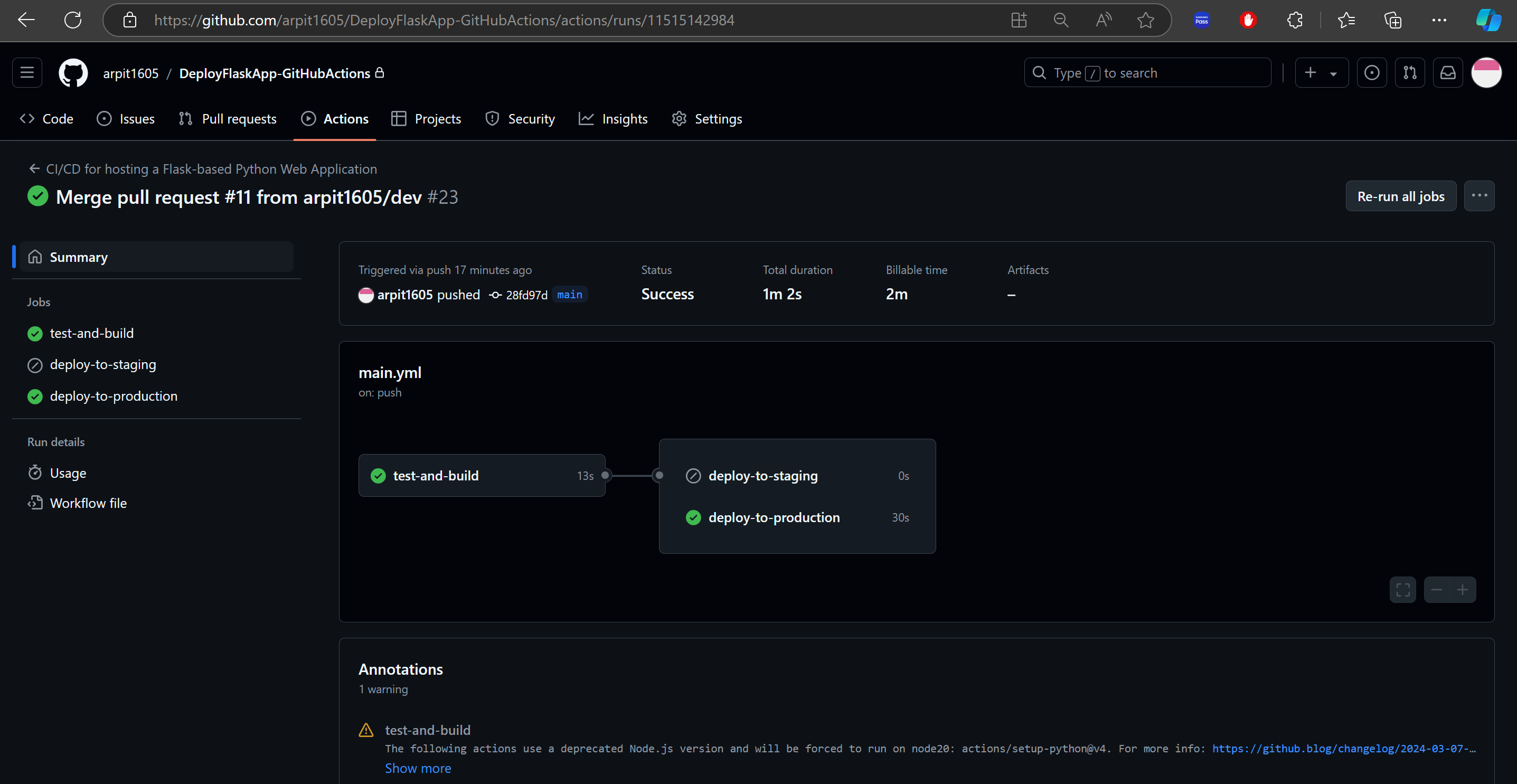
**Merged the changes for index.html file from the ‘dev’ branch to the ‘main’ branch:**



**Content of index.html file in the ‘main’ branch:**



**The GitHub Action workflow is completed as shown below:**



The Flask based Python web application is accessible with public IP address of production Amazon EC2 instance as shown below:

