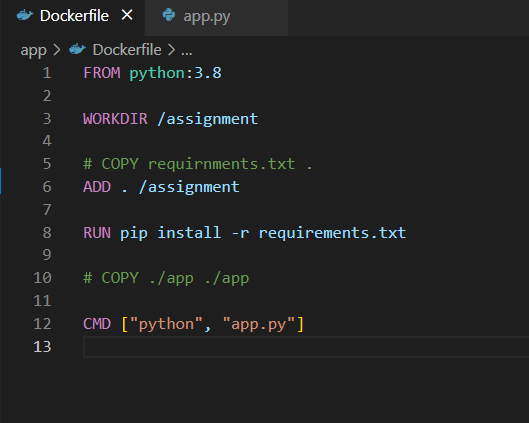
1. **Code Dockerizing**
2. Creating virtual environment

activate it using source venv/bin/activate



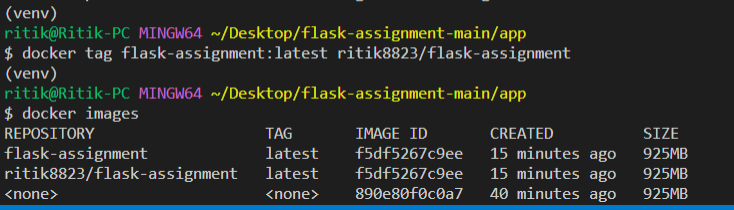
1. Building image using dockerfile



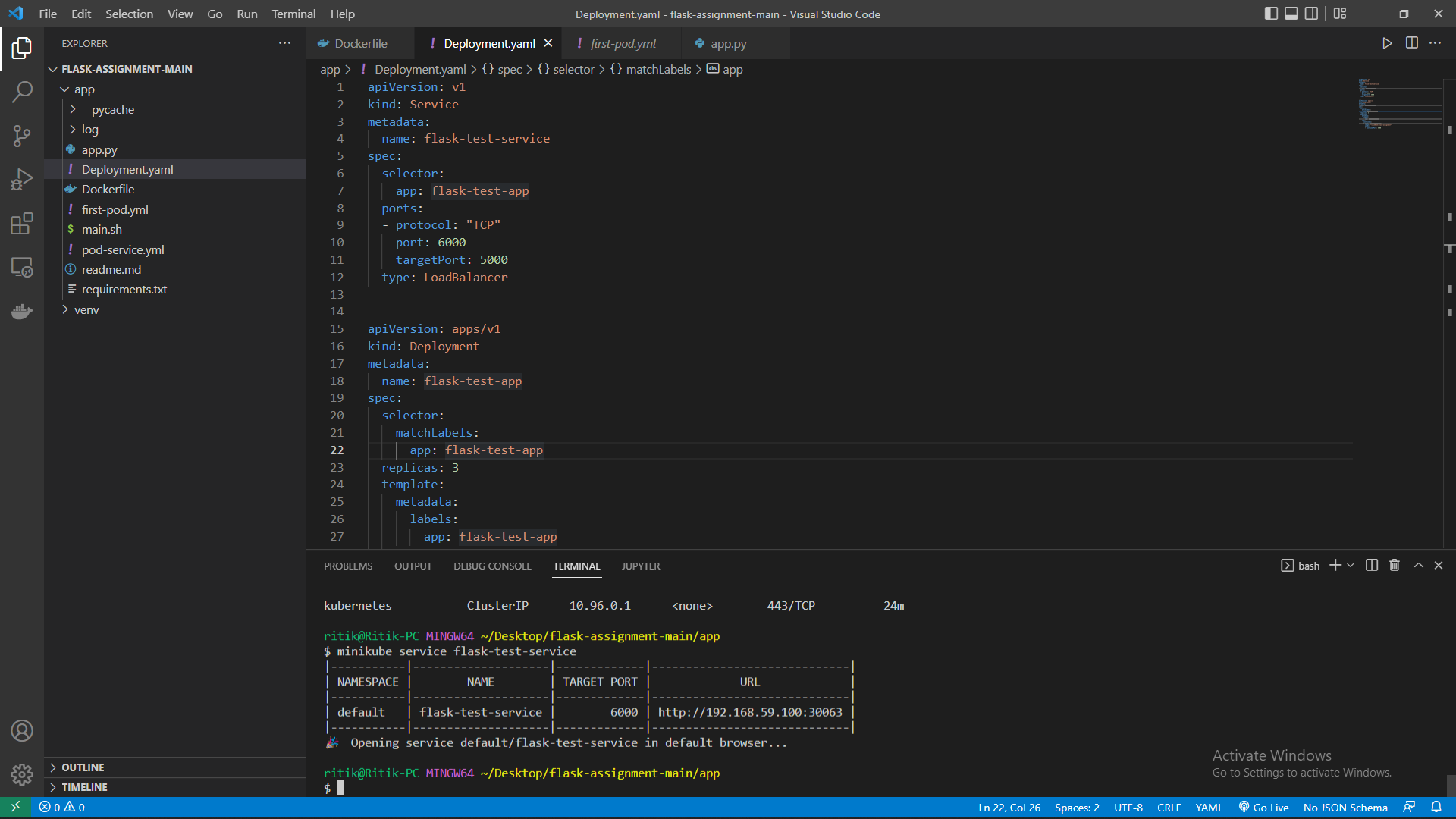
1. Building image



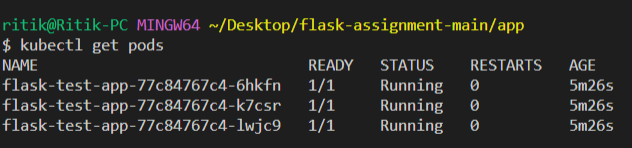
1. Pushing image to docker hub



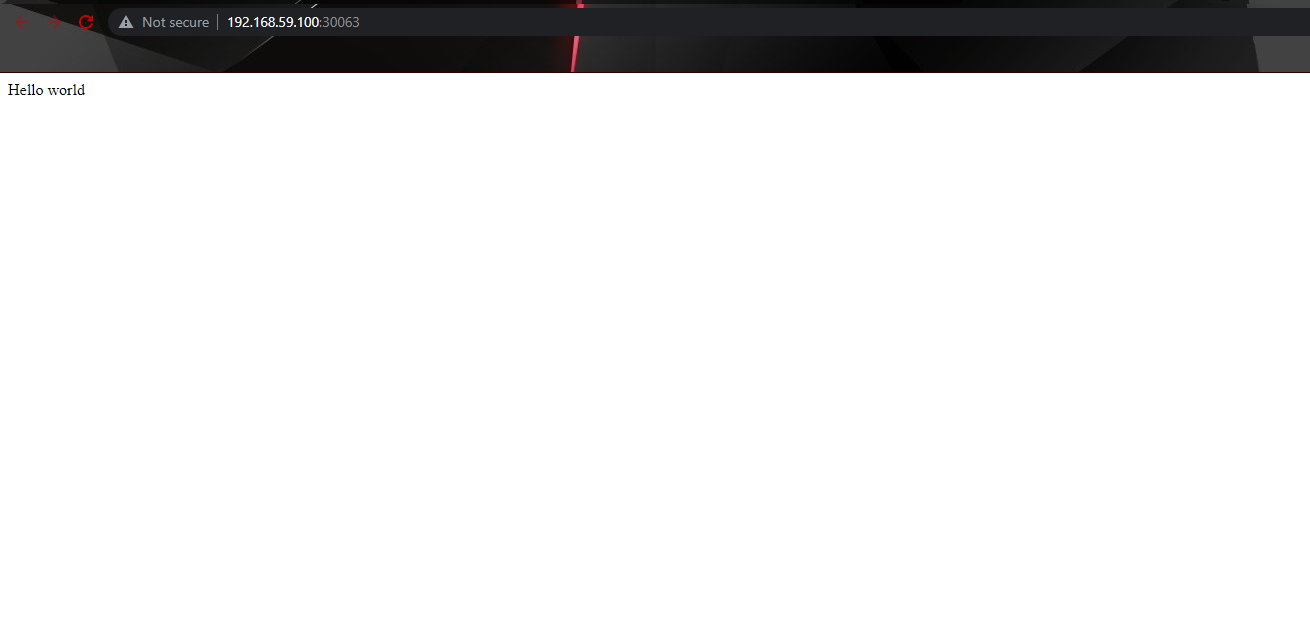
1. **Code Deployment**
2. apiVersion: v1
3. kind: Service
4. metadata:
5. name: flask-test-service
6. spec:
7. selector:
8. app: flask-test-app
9. ports:
10. - protocol: "TCP"
11. port: 6000
12. targetPort: 5000
13. type: LoadBalancer
14. ---
15. apiVersion: apps/v1
16. kind: Deployment
17. metadata:
18. name: flask-test-app
19. spec:
20. selector:
21. matchLabels:
22. app: flask-test-app
23. replicas: 3
24. template:
25. metadata:
26. labels:
27. app: flask-test-app
28. spec:
29. containers:
30. - name: "flask-test-app"
31. image: "ritik8823/flask-assignment"
32. ports:
33. - containerPort: 5000



Running flask app in k8s clusters with 3 replicas



**Output**

****

3. Continous Integration using Github actions

name: Docker Image CI

on:

  push:

    branches: [ "main" ]

jobs:

  build:

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

        name: Check out code

      - uses: docker/build-push-action@v2

        name: Build & push Docker image

        with:

          image: ritik8823/flask-assignment

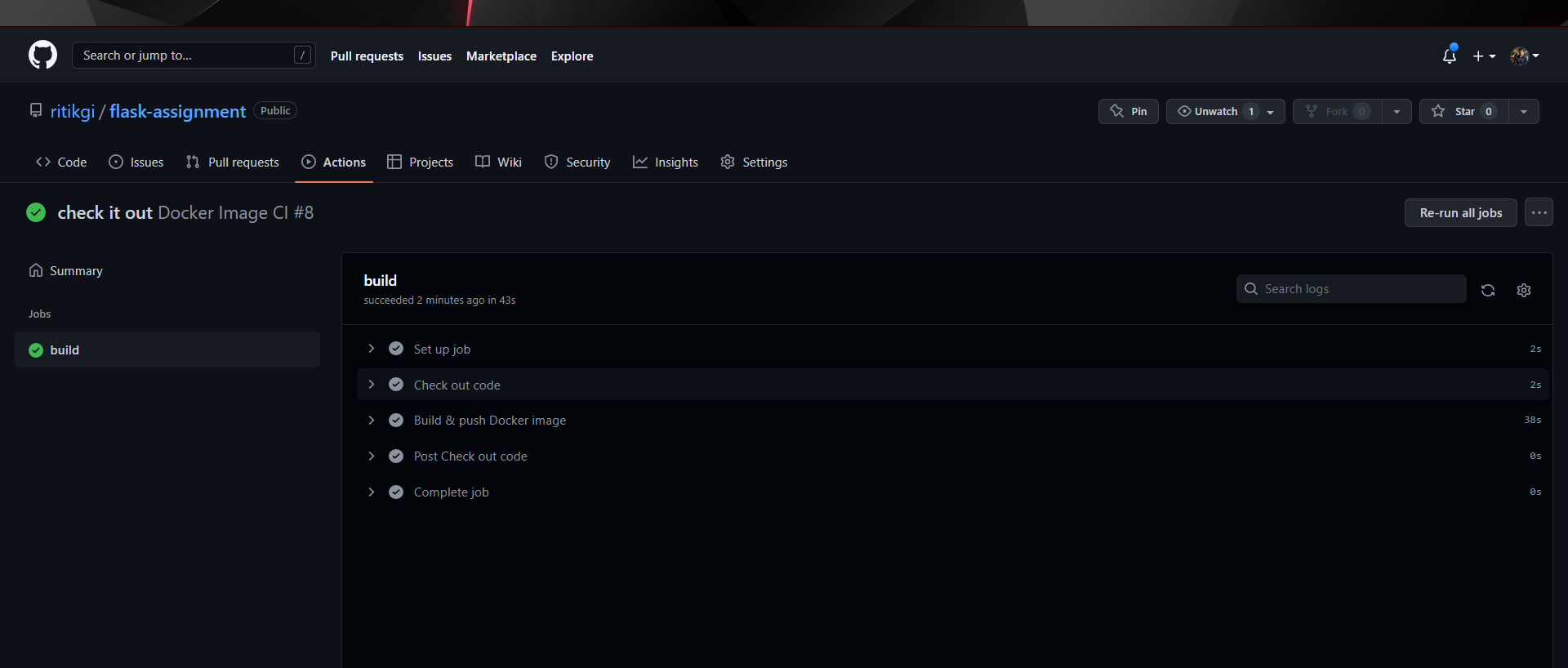
          tag: latest

          registry: docker.io

          username: ${{ secrets.DOCKER\_USERNAME }}

          password: ${{ secrets.DOCKER\_PASSWORD }}





Final code - <https://github.com/ritikgi/flask-assignment>

**4. Security**

**Point of Failures in each step, and how to avoid them**

1. Confusing continuous deployment for continuous delivery

Continuous deployment is the concept that every change made in the code base will be deployed almost immediately to production if the results of the pipeline are successful. This is terrifying to most organizations because rapid product changes can scare away users.

1. Lack of coordination between continuous integration and continuous delivery

**Point of Hacking in each step (from security point), and how to avoid them.**

1. Keep secrets safe

Secrets are authentication credentials, such as usernames and passwords, API tokens, SSH keys and encryption keys, that allow access to applications and services. They are literally the keys to a project's data and resources. If these credentials are improperly secured and used, they can open the doors to a serious data breach or intellectual property theft.

1. Continuous monitoring after deployments lead to increase security.