FastAPI CI/CD Pipeline Using Docker, Docker Hub, GitHub Actions & AWS EC2

Introduction

This project demonstrates a complete CI/CD pipeline implementation where a FastAPI application is:

1. Dockerized
2. Pushed to Docker Hub automatically
3. Deployed to AWS EC2 using SSH automation

This documentation contains every step in the correct order so you can repeat the project anytime in the future.

Prerequisites

Before starting, ensure you have:

- GitHub account
- Docker Hub account
- AWS EC2 Ubuntu instance
- Git & VS Code installed
- PuTTY or SSH access (for Windows users)

Step 1: Create a project folder on your system

```
mkdir fastapi-cicd-project

cd fastapi-cicd-project
```

Inside that folder create app/:

```
mkdir app

app/
 ├── main.py
 ├── requirements.txt
 └── Dockerfile
```

In app/ inside create 3 files

- main.py
- requirements.txt
- Dockerfile

main.py

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")

def index():

    return {"message": "FastAPI CI/CD working successfully!"}
```

requirements.txt

```
fastapi

uvicorn
```

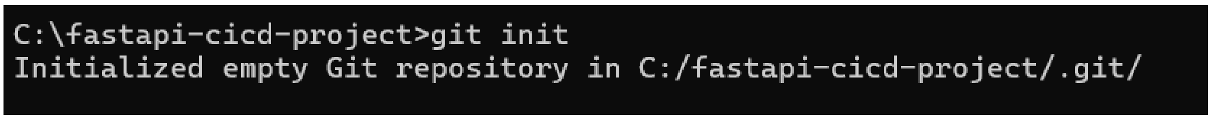
Step 2: Create Dockerfile in app folder

```
app/Dockerfile:

FROM python:3.10-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Step 3: Initialize Git Repository

In project root:

```
git init
```



```
C:\fastapi-cicd-project>git init
Initialized empty Git repository in C:/fastapi-cicd-project/.git/
```

```
git add .

git commit -m "Initial commit"
```

Connect to GitHub repo:

```
git remote add origin git@github.com:<username>/fastapi-cicd-project.git

git push -u origin main
```

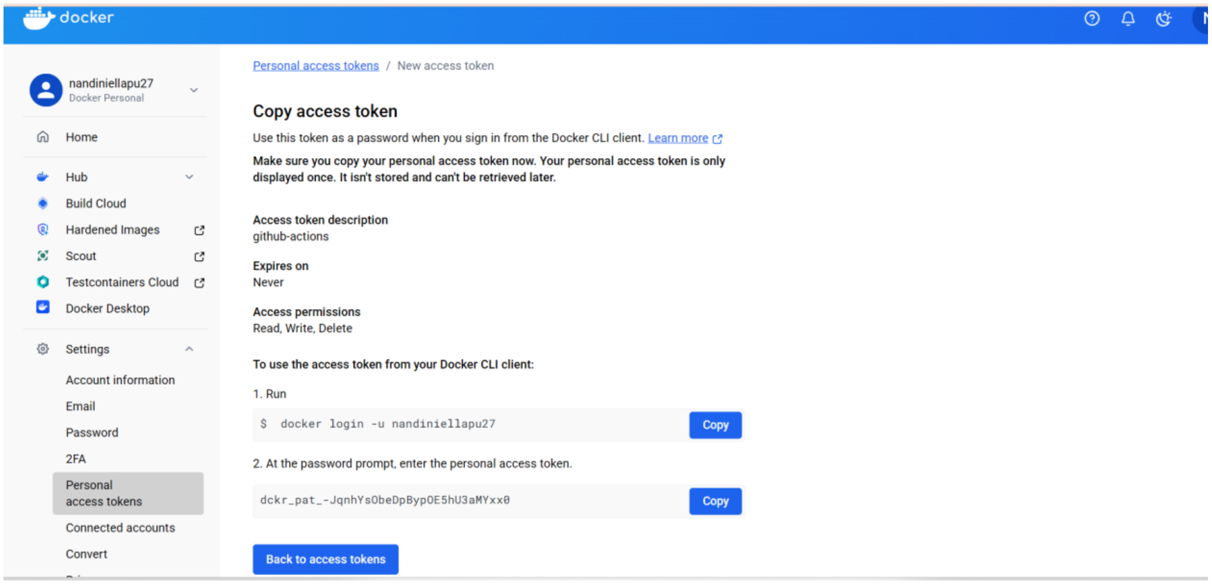
Step 4: Docker Hub Setup

1.Login to Docker Hub

2.Create repository: **fastapi-cicd**

3.Generate Access Token:

- Go to Account Settings → Security → New Access Token



Personal access tokens / New access token

Copy access token

Use this token as a password when you sign in from the Docker CLI client. Learn more

Make sure you copy your personal access token now. Your personal access token is only displayed once. It isn't stored and can't be retrieved later.

Access token description
github-actions

Expires on
Never

Access permissions
Read, Write, Delete

To use the access token from your Docker CLI client:

1. Run

```
$  docker login -u nandiniellapu27
```

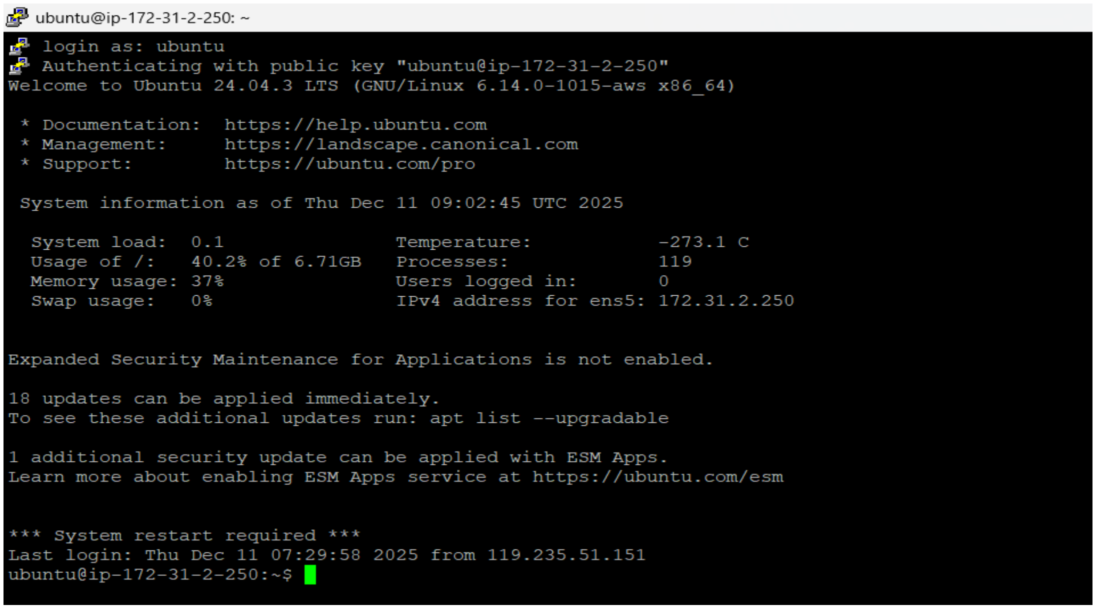
2. At the password prompt, enter the personal access token.

```
dckr_pat_-JqnhYsObeDpBypOE5hU3aMYxx0
```

Step 5: AWS EC2 Setup

Launch **Ubuntu Server** instance.

Connect using PuTTY / SSH.



```
login as: ubuntu
Authenticating with public key "ubuntu@ip-172-31-2-250"
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Dec 11 09:02:45 UTC 2025

  System load:  0.1               Temperature:           -273.1 C
  Usage of /:   40.2% of 6.71GB   Processes:             119
  Memory usage: 37%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 172.31.2.250


Expanded Security Maintenance for Applications is not enabled.

18 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


*** System restart required ***
Last login: Thu Dec 11 07:29:58 2025 from 119.235.51.151
ubuntu@ip-172-31-2-250:~$
```

Install Docker:

```
sudo apt update

sudo apt install docker.io -y

sudo systemctl enable docker

sudo systemctl start docker
```

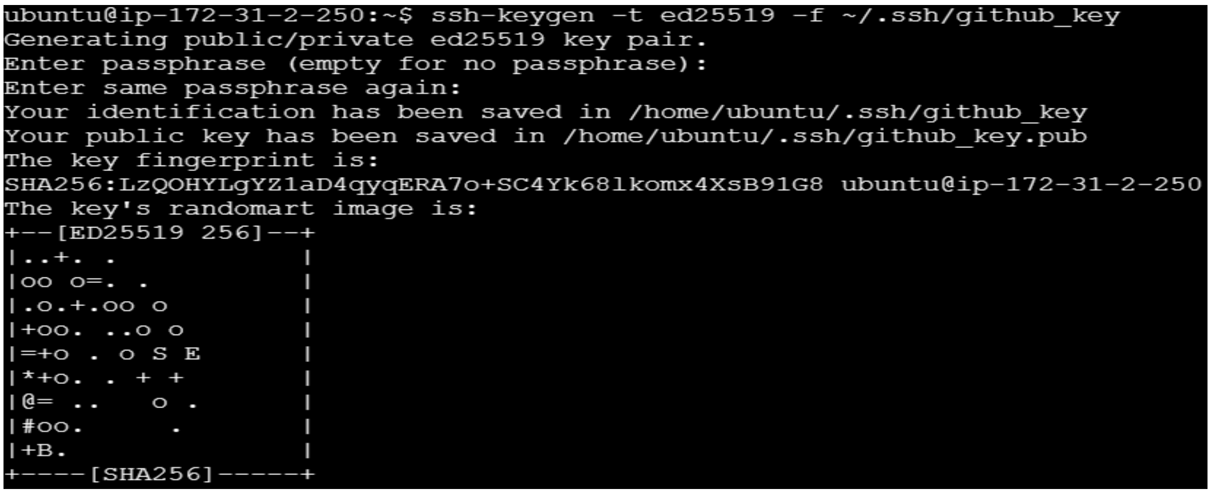
Allow port:

sudo ufw allow 8000

Step 6: Generate Deployment SSH Key

Generate key on EC2:

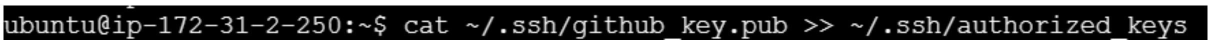
```
ssh-keygen -t ed25519 -f ~/.ssh/github_key -C "github-actions"
```



```
ubuntu@ip-172-31-2-250:~$ ssh-keygen -t ed25519 -f ~/.ssh/github_key
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/github_key
Your public key has been saved in /home/ubuntu/.ssh/github_key.pub
The key fingerprint is:
SHA256:LzQOHYLgYZ1aD4qyqERA7o+SC4Yk68lkomx4XsB91G8 ubuntu@ip-172-31-2-250
The key's randomart image is:
+--[ED25519 256]--+
|..+. .           |
|oo o=. .         |
|.o.+.oo o        |
|+oo. ..o o       |
|=+o . o S E      |
|*+o. . + +       |
|@= ..   o .      |
|#oo.     .       |
|+B.              |
+----[SHA256]-----+
```

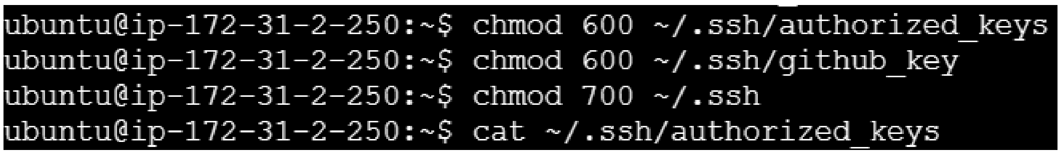
Add public key to authorized_keys:

```
cat ~/.ssh/github_key.pub >> ~/.ssh/authorized_keys
```



```
ubuntu@ip-172-31-2-250:~$ cat ~/.ssh/github_key.pub >> ~/.ssh/authorized_keys
```

```
chmod 600 ~/.ssh/authorized_keys
chmod 600 ~/.ssh/github_key
chmod 700 ~/.ssh
```



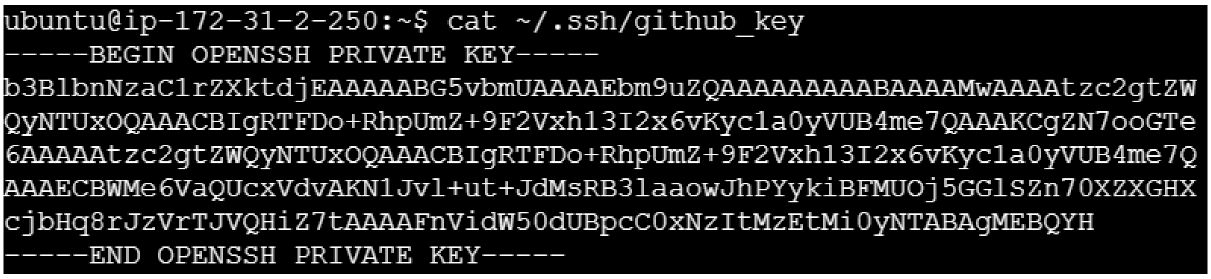
```
ubuntu@ip-172-31-2-250:~$ chmod 600 ~/.ssh/authorized_keys
ubuntu@ip-172-31-2-250:~$ chmod 600 ~/.ssh/github_key
ubuntu@ip-172-31-2-250:~$ chmod 700 ~/.ssh
ubuntu@ip-172-31-2-250:~$ cat ~/.ssh/authorized_keys
```

Copy private key:

Use:

```
cat ~/.ssh/github_key
```

Copy this and save as GitHub Secret.



```
ubuntu@ip-172-31-2-250:~$ cat ~/.ssh/github_key
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACBIgRTFDo+RhpUmZ+9F2Vxh13I2x6vKyc1a0yVUB4me7QAAAKCgZN7ooGTe
6AAAAAtzc2gtZWQyNTUxOQAAACBIgRTFDo+RhpUmZ+9F2Vxh13I2x6vKyc1a0yVUB4me7Q
AAAECBWMe6VaQUcxVdvAKN1Jvl+ut+JdMsRB3laaowJhPYykiBFMUOj5GGlSZn70XZXGHX
cjbHq8rJzVrTJVQHiZ7tAAAAFnVidW50dUBpcC0xNzItMzEtMi0yNTABAgMEBQYH
-----END OPENSSH PRIVATE KEY-----
```

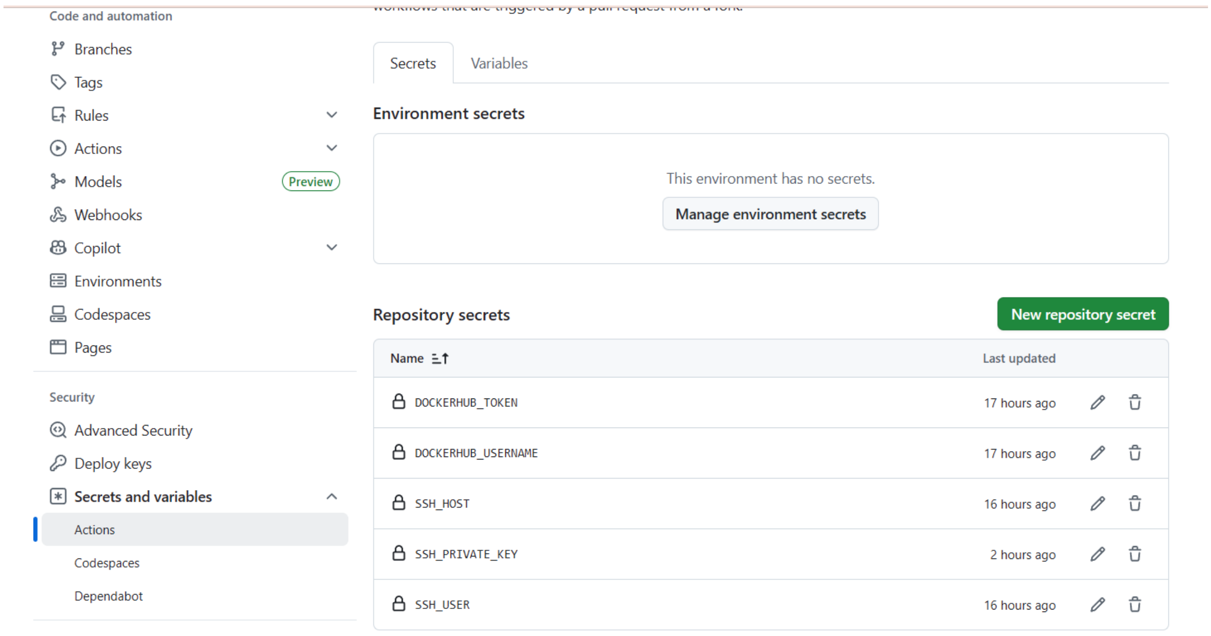
Step 7: Add GitHub Secrets

Go to:

GitHub Repo → Settings → Secrets → Actions

Add:

Secret Name	**Value**
DOCKERHUB_USERNAME	your Docker Hub username
DOCKERHUB_TOKEN	Docker Hub token
SSH_HOST	EC2 public IP
SSH_USER	ubuntu
SSH_PRIVATE_KEY	content of ~/.ssh/github_key



Environment secrets

This environment has no secrets.

Repository secrets

Name	Last updated
DOCKERHUB_TOKEN	17 hours ago
DOCKERHUB_USERNAME	17 hours ago
SSH_HOST	16 hours ago
SSH_PRIVATE_KEY	2 hours ago
SSH_USER	16 hours ago

Step 8: Create GitHub Actions Workflow

Create folder:

.github/workflows/deploy.yml

Add this:

```
name: CI/CD Pipeline
on:
  push:
    branches: [ "main" ]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout code
      uses: actions/checkout@v3
    - name: Login to Docker Hub
      uses: docker/login-action@v2
      with:
        username: ${{ secrets.DOCKERHUB_USERNAME }}
        password: ${{ secrets.DOCKERHUB_TOKEN }}
    - name: Build Docker image
      run: docker build -t ${{ secrets.DOCKERHUB_USERNAME }}/fastapi-cicd:latest ./app
    - name: Push Docker image
      run: docker push ${{ secrets.DOCKERHUB_USERNAME }}/fastapi-cicd:latest
  deploy:
    needs: build
    runs-on: ubuntu-latest
    steps:
    - name: Deploy to EC2
      uses: appleboy/ssh-action@v1.0.0
      with:
        host: ${{ secrets.SSH_HOST }}
        username: ${{ secrets.SSH_USER }}
        key: ${{ secrets.SSH_PRIVATE_KEY }}
        script: |
          sudo docker pull ${{ secrets.DOCKERHUB_USERNAME }}/fastapi-cicd:latest
          sudo docker stop fastapi-app || true
          sudo docker rm fastapi-app || true
          sudo docker run -d --name fastapi-app -p 8000:8000 ${{ secrets.DOCKERHUB_USERNAME }}/fastapi-cicd:latest
```

Fix deploy key

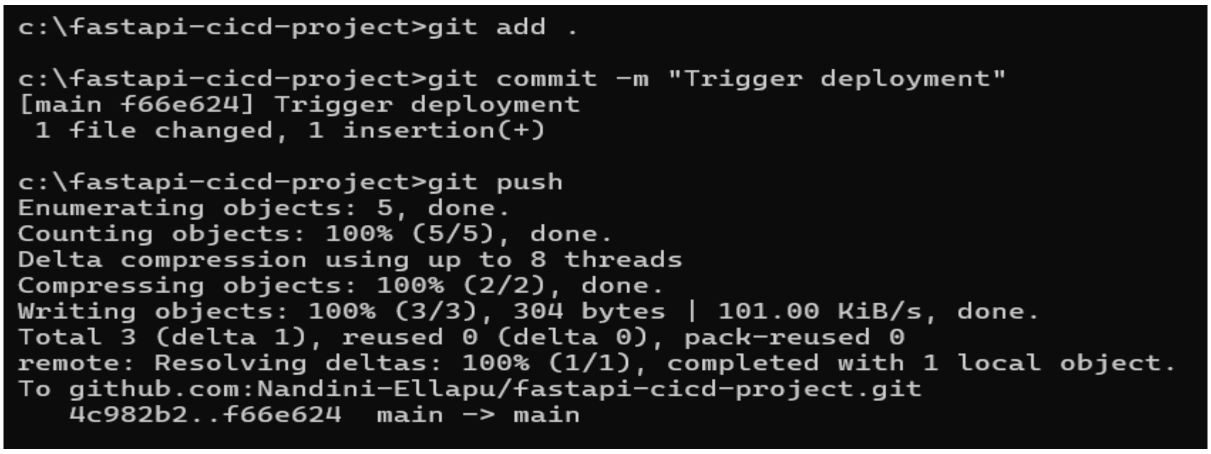
Step 9: Trigger CI/CD Pipeline

Make a small change:

```
echo "update" >> README.md
```

Push:

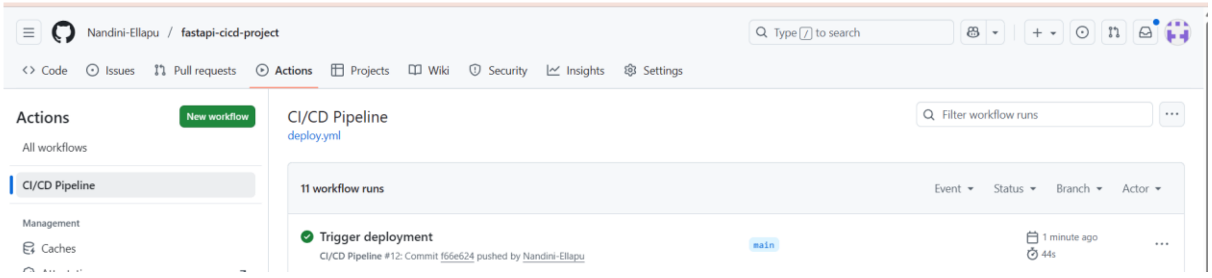
```
git add .

git commit -m "Trigger deployment"

git push
```



```
c:\fastapi-cicd-project>git add .

c:\fastapi-cicd-project>git commit -m "Trigger deployment"
[main f66e624] Trigger deployment
 1 file changed, 1 insertion(+)

c:\fastapi-cicd-project>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 304 bytes | 101.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Nandini-Ellapu/fastapi-cicd-project.git
   4c982b2..f66e624  main -> main
```

GitHub Actions will:

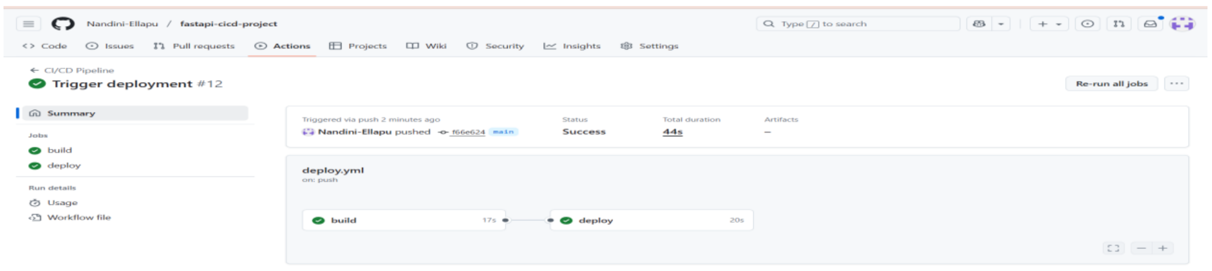
1. Build Docker image
2. Push to Dockerhub
3. SSH to EC2
4. Pull + restart container



CI/CD Pipeline
deploy.yml

11 workflow runs

Trigger deployment
CI/CD Pipeline #12: Commit f66e624 pushed by Nandini-Ellapu
main
1 minute ago
44s



Trigger deployment #12

Triggered via push 2 minutes ago
Nandini-Ellapu pushed f66e624 main

Status: Success
Total duration: 44s
Artifacts: –

deploy.yml
on: push

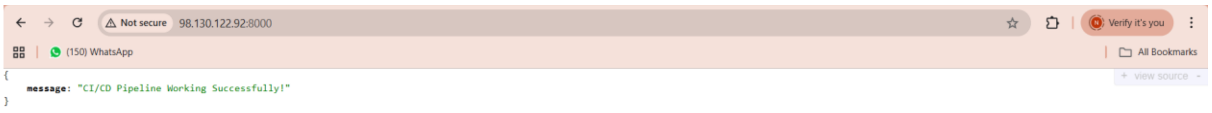
build 17s → deploy 20s

Then open in browser:

http://VM_PUBLIC_IP:8000

You should see:

{"message": "CI/CD Pipeline Working Successfully!"}



```
{
    message: "CI/CD Pipeline Working Successfully!"
}
```

That means:

You have completed a real CI/CD project with GitHub Actions + Docker + VM. 🎉