2. GitHub Actions CI/CD Pipeline Flask App

Objective:

Implement a CI/CD workflow using GitHub Actions for a Python application.

Requirements:

1. Setup:

- Use a provided Python application repository on GitHub (provide a link to a sample Python application repository).

- Ensure the repository has a main branch and a staging branch.

2. GitHub Actions Workflow:

- Create a .github/workflows directory in your repository.

- Inside the directory, create a YAML file to define the workflow.

3. Workflow Steps:

- Define a workflow that performs the following jobs:

- Install Dependencies: Install all necessary dependencies for the Python application using pip.

- Run Tests: Execute the test suite using a framework like pytest.

- Build: If tests pass, prepare the application for deployment.

- Deploy to Staging: Deploy the application to a staging environment when changes are pushed to the staging branch.

- Deploy to Production: Deploy the application to production when a release is tagged.

4. Environment Secrets:

- Use GitHub Secrets to store sensitive information required for deployments (e.g., deployment keys, API tokens).

5. Documentation:

- Update the README.md file with instructions on how the GitHub Actions workflow works and how to configure the necessary secrets.

6. Submission:

- Provide the URL to the GitHub repository with the workflow file and updated README.md.

- Include screenshots of the GitHub Actions workflow runs showing successful execution of all steps.

Deliverables:

- GitHub repository with the workflow file.

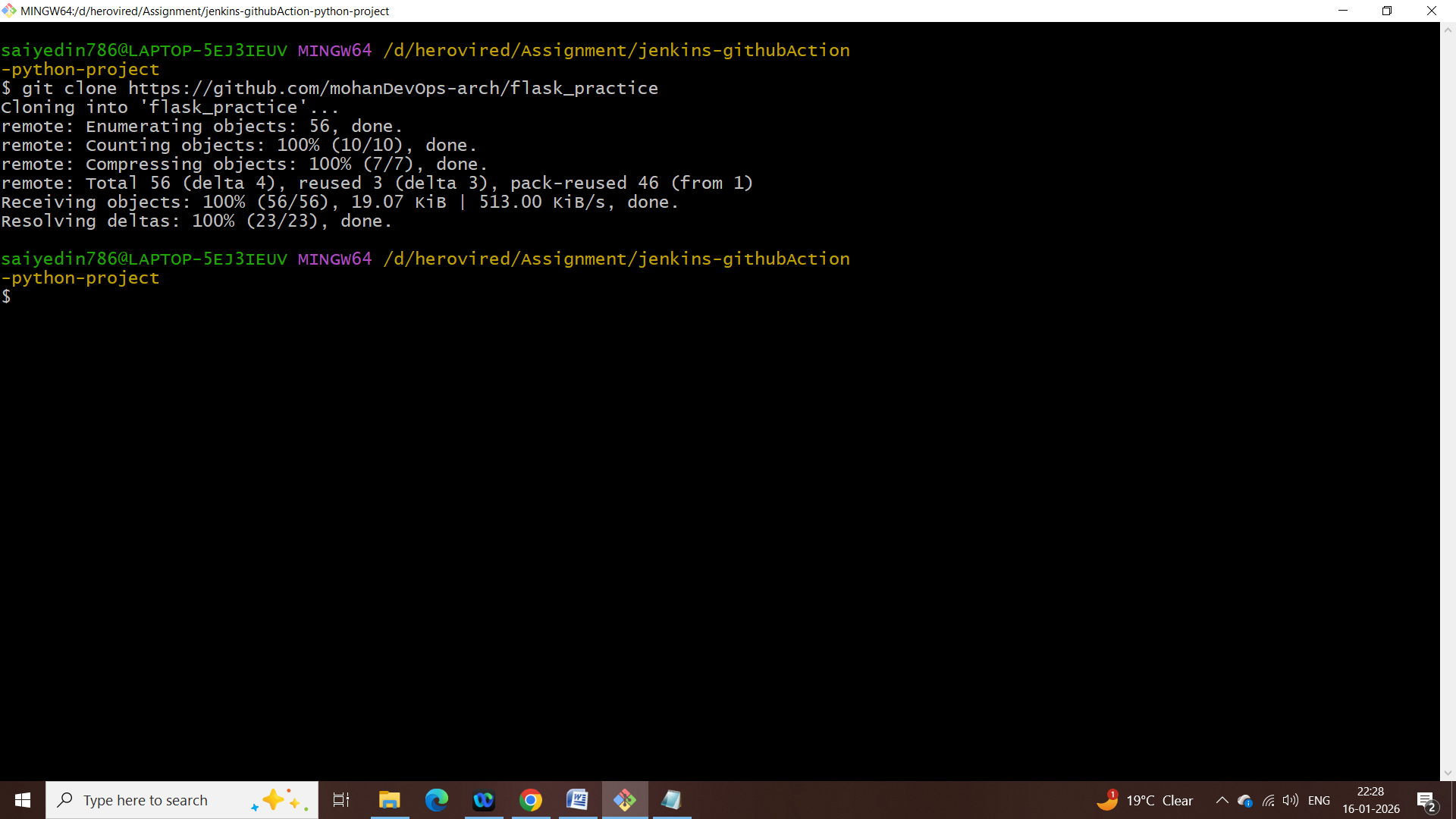
- Documentation in README.md.

- Screenshots of the GitHub Actions workflow runs.

Project Solution:

Step :1 – Cloning flask app from github repository

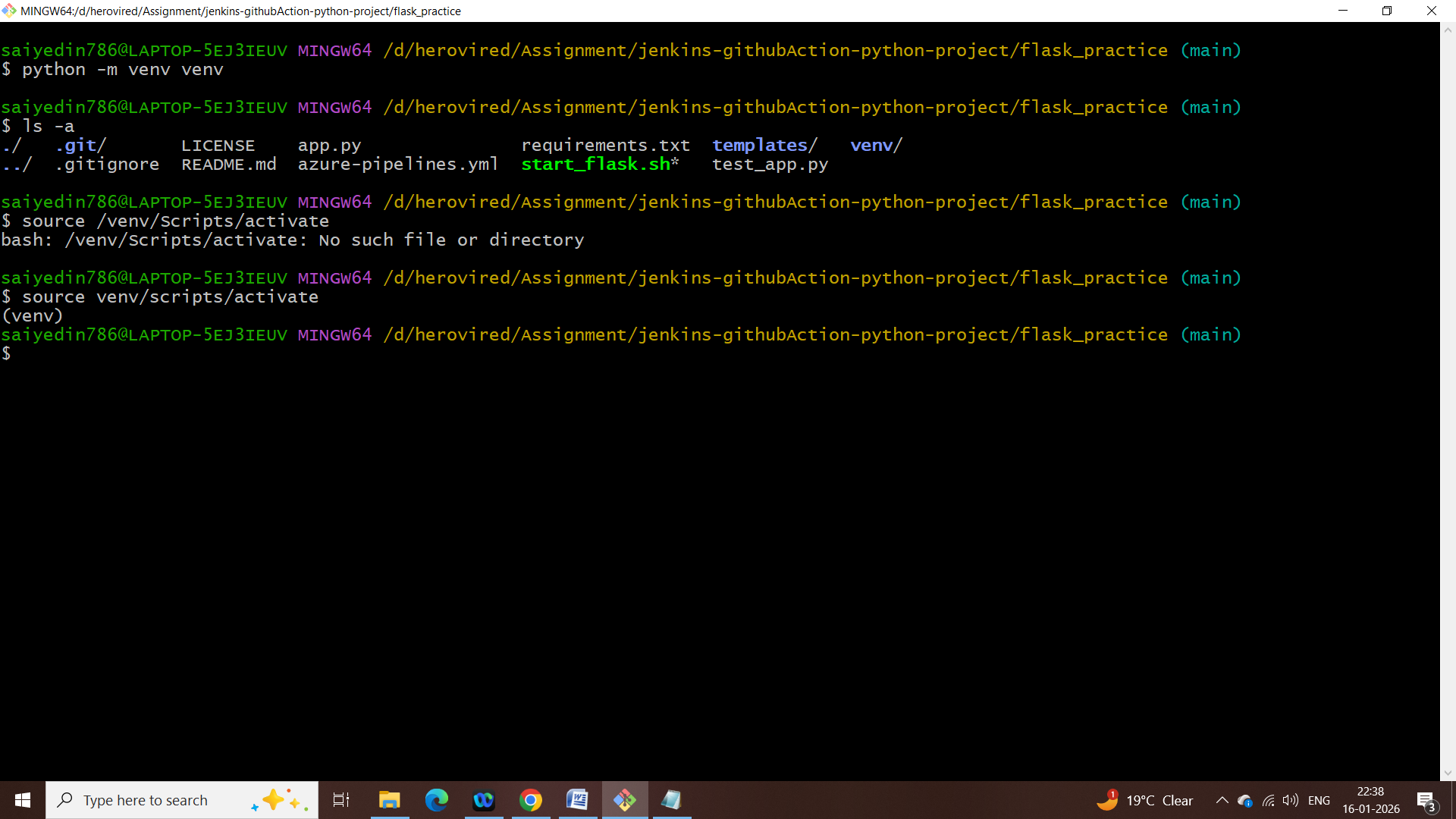
Repo link: <https://github.com/mohanDevOps-arch/flask_practice>



Step: 2- creating an virtual environment and activating it

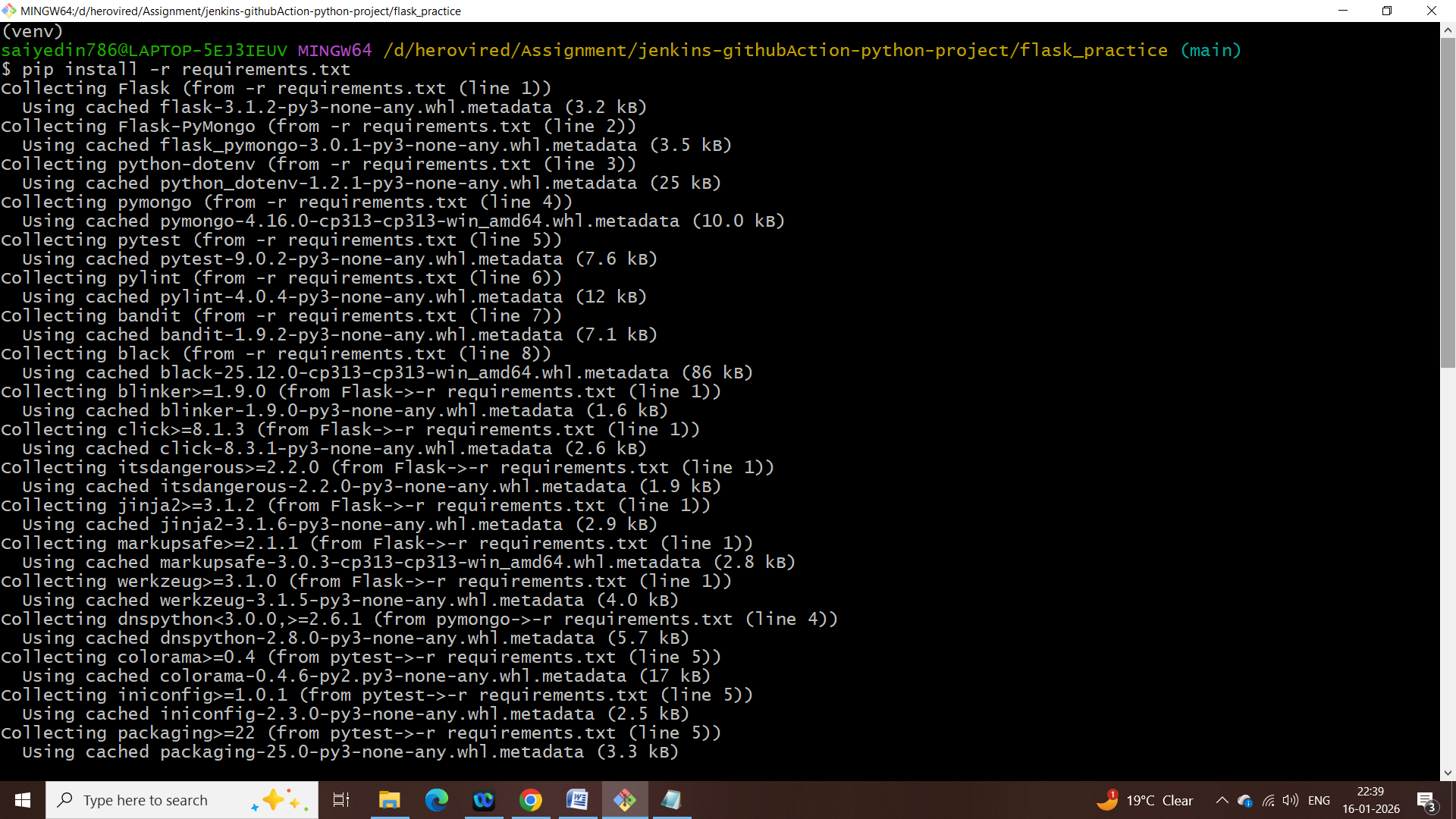
python -m venv venv

source /venv/scripts/activate



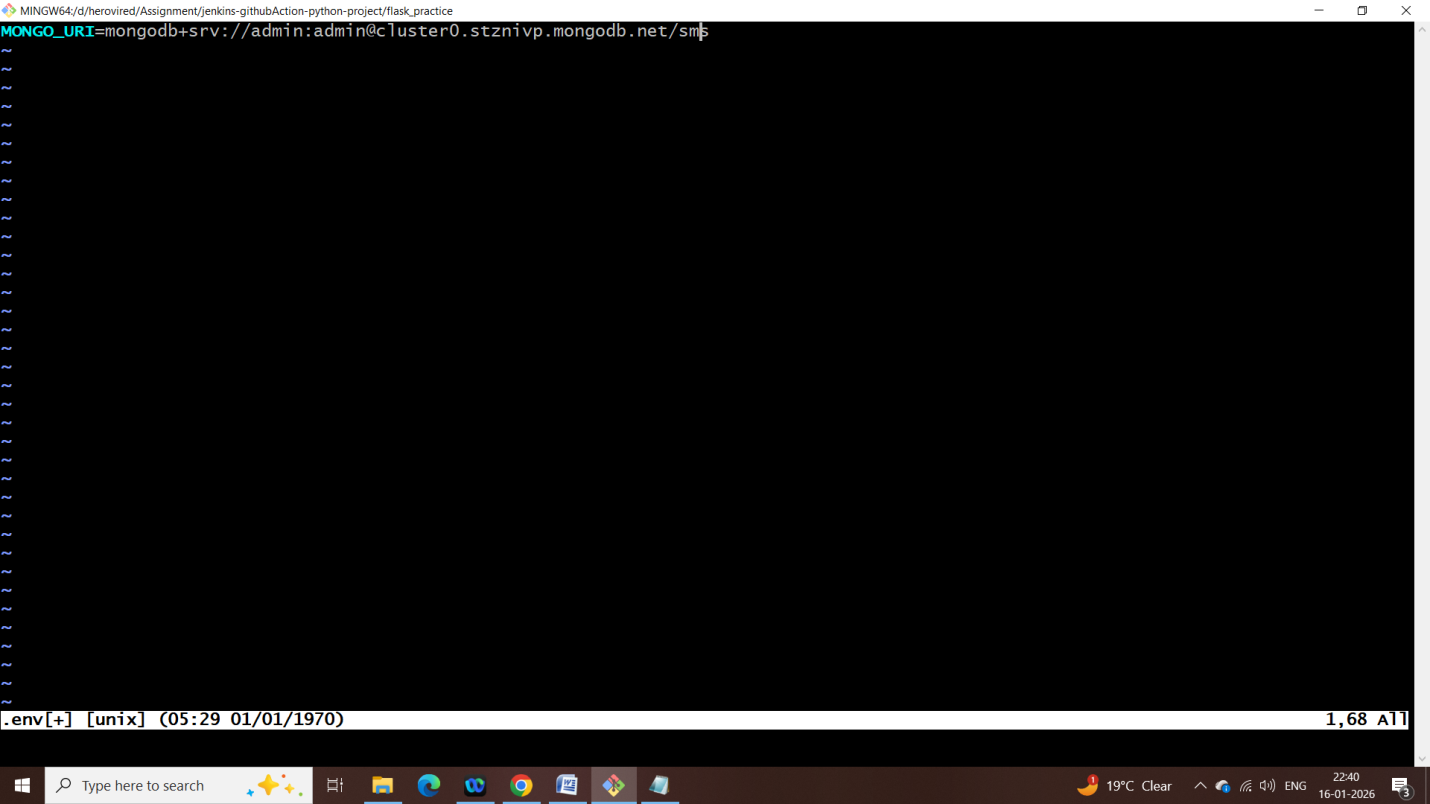
Step :3 - Installing dependencies from requirements.txt in local computer

Pip install –r requirements.txt



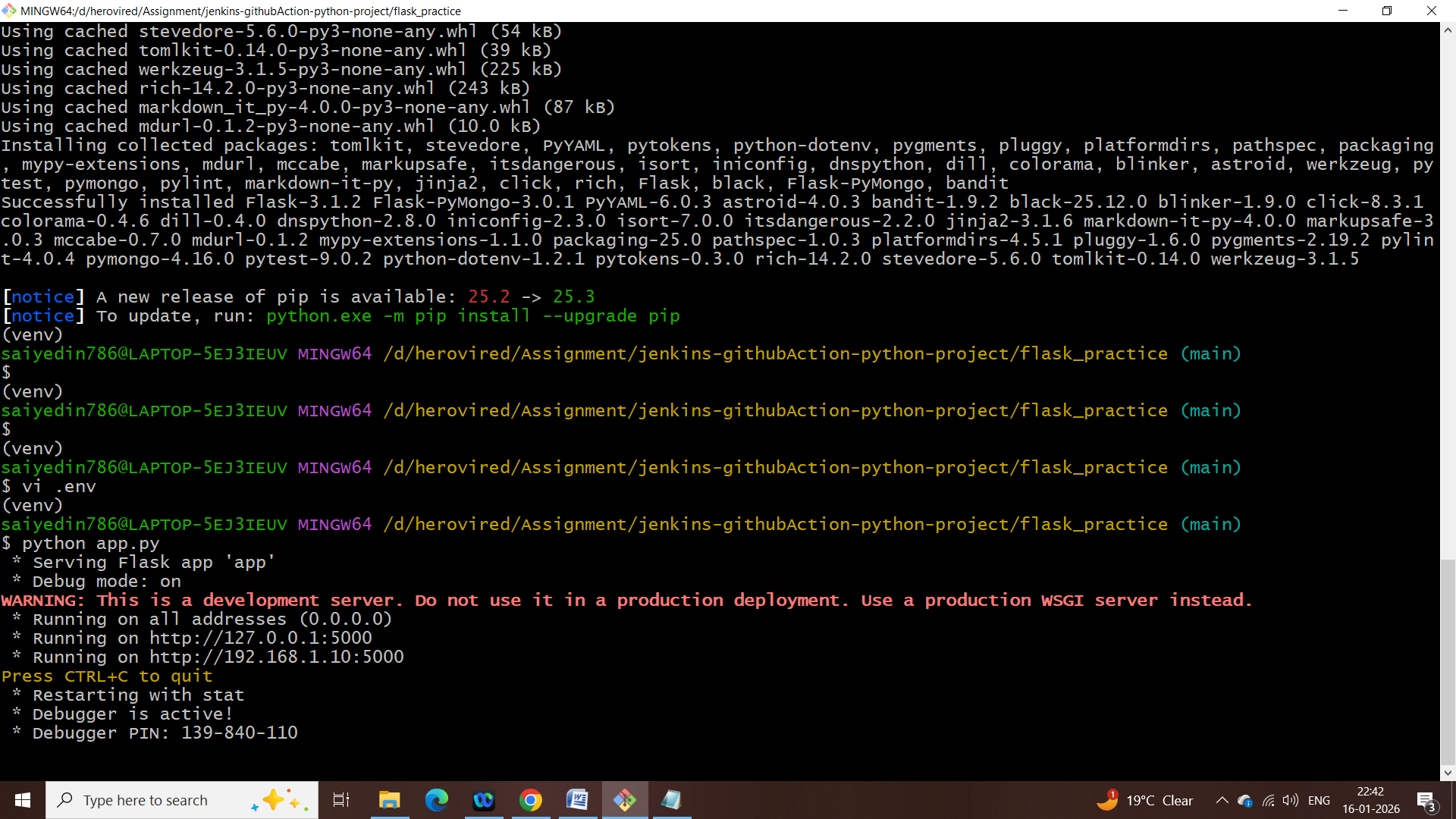
Step-4 : creating an .env file

MONGO\_URI=mongodb+srv://admin:admin@cluster0.stznivp.mongodb.net/sms

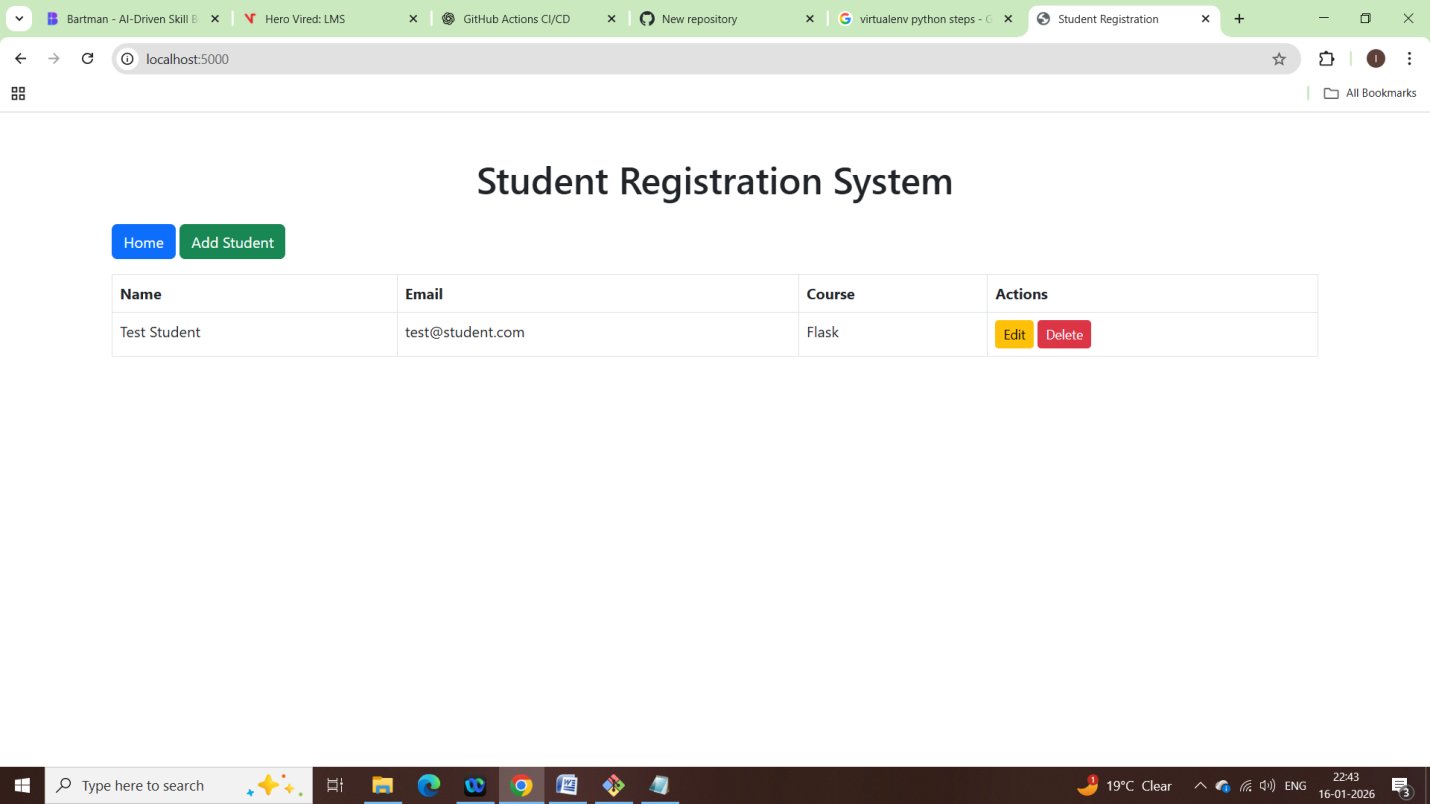


Step:5 running the flask app

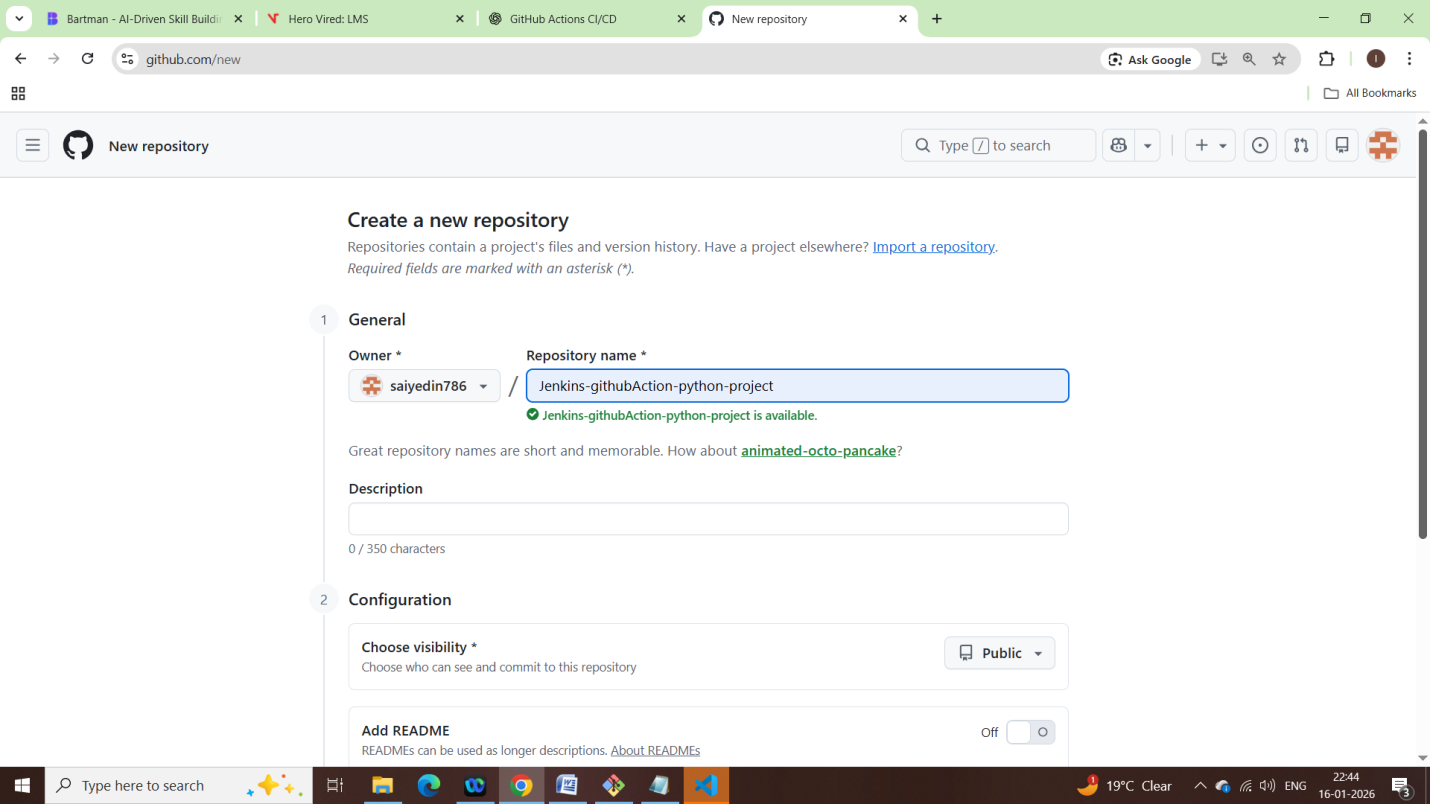
Python app.py

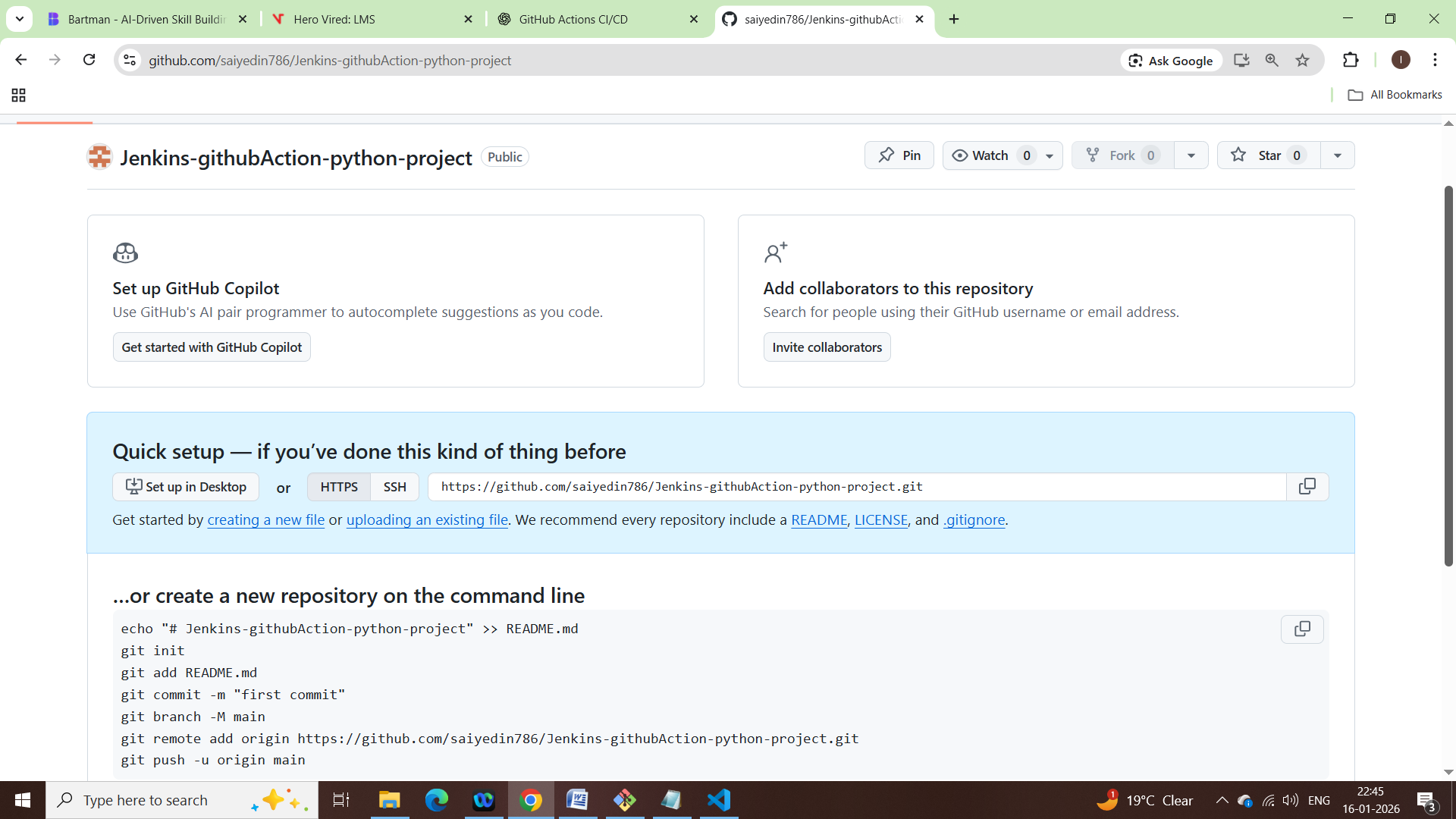


Accessing application in browser



Creating github repository





echo "# Jenkins-githubAction-python-project" >> README.md

git init

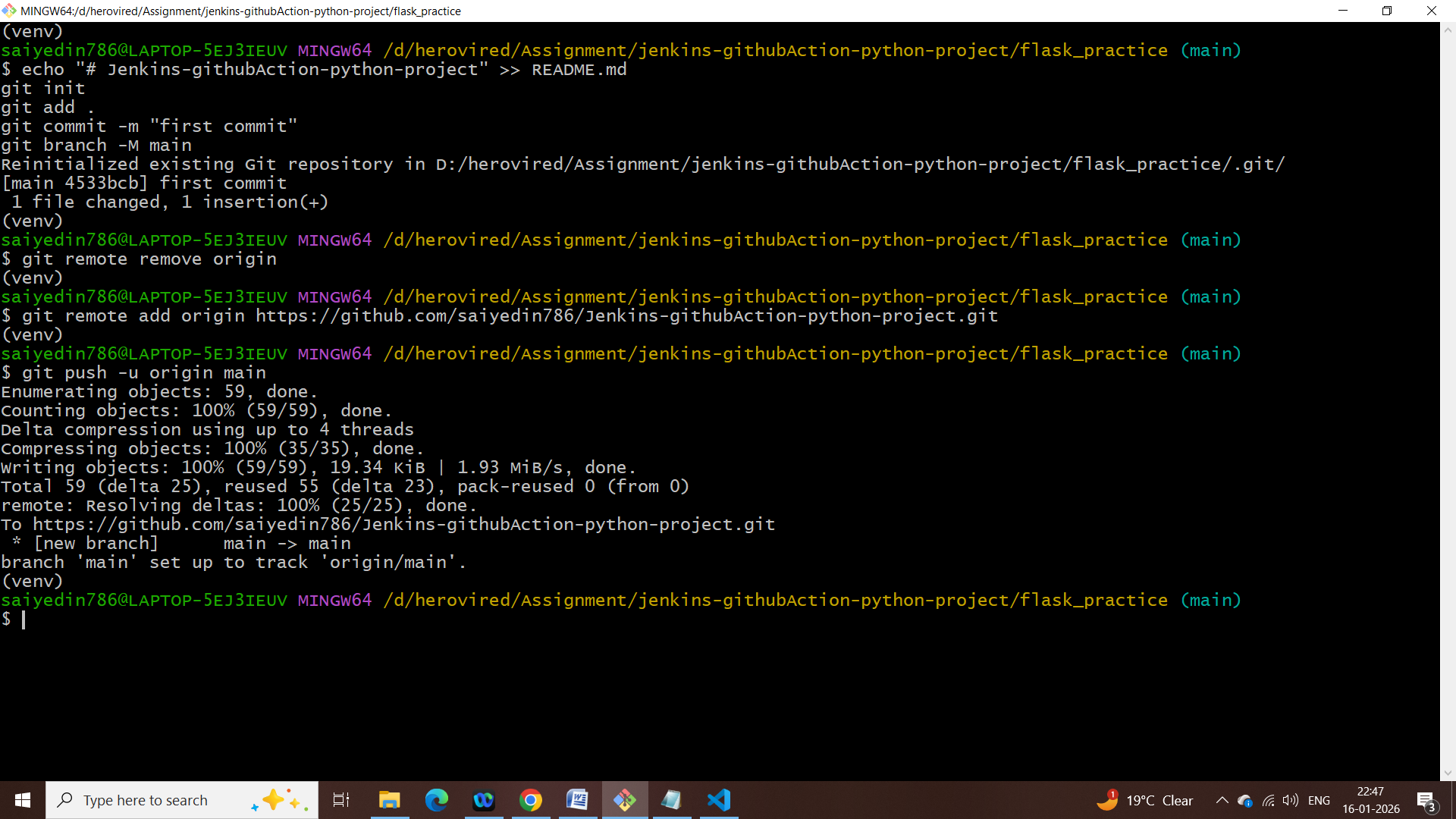
git add .

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/saiyedin786/Jenkins-githubAction-python-project.git

git push -u origin main



Creating branch staging:

Git branch staging

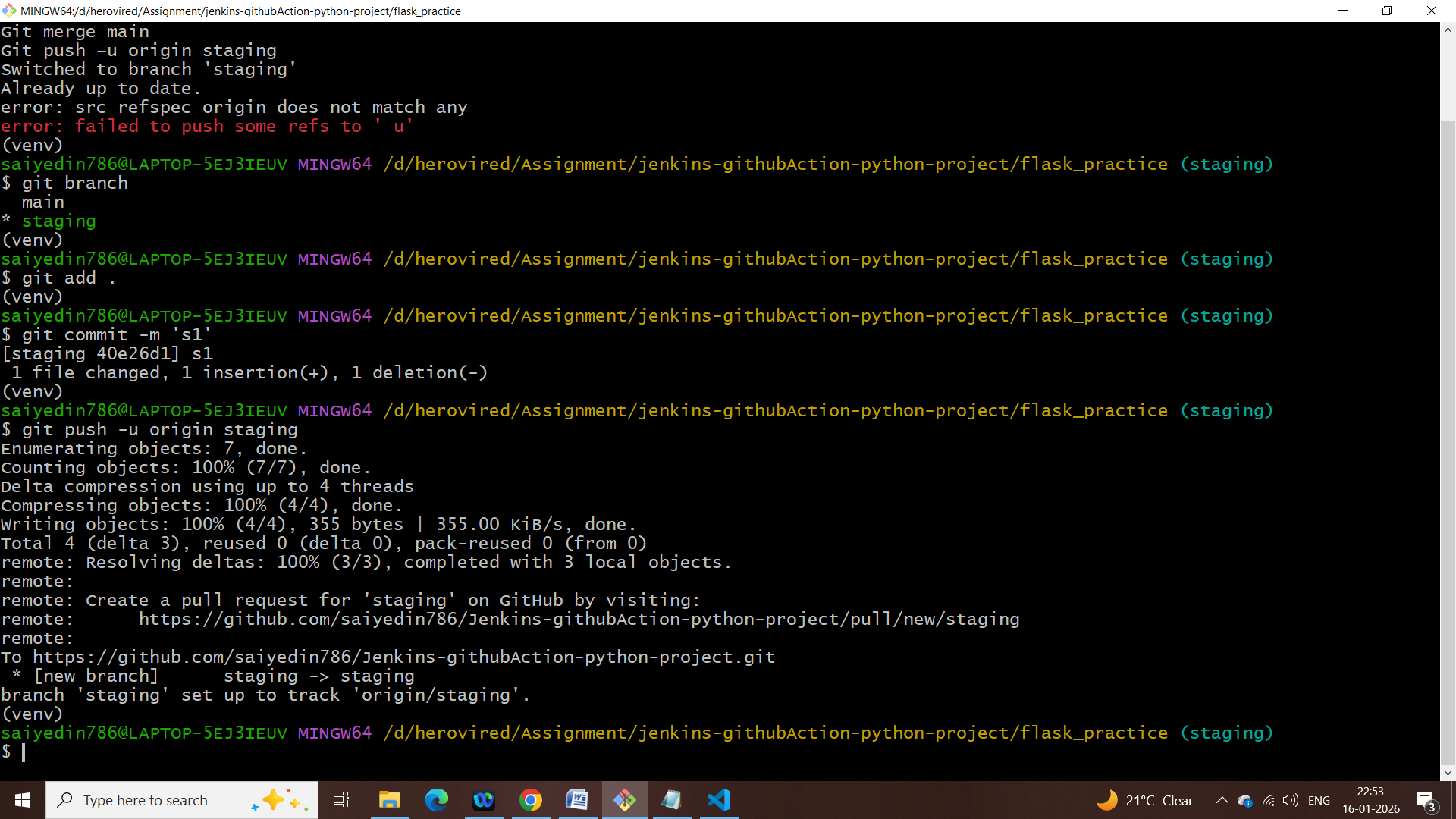
Git checkout staging

Git merge main

Git add .

Git commit –m ‘s1’

Git push –u origin stagin



Creating github workflow named deploy.yml

Deploy.yml

name: Flask CI/CD Pipeline

on:

push:

branches:

- main

- staging

release:

types: [created]

jobs:

build-test:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Set up Python

uses: actions/setup-python@v5

with:

python-version: "3.10"

- name: Install dependencies

run: |

python -m pip install --upgrade pip

pip install -r requirements.txt

- name: Run tests

run: pytest

deploy-staging:

needs: build-test

runs-on: ubuntu-latest

if: github.ref == 'refs/heads/staging'

steps:

- name: Deploy to Staging

run: |

echo "Deploying to STAGING environment"

echo "Using STAGING\_SECRET=${{ secrets.STAGING\_SECRET }}"

deploy-production:

needs: build-test

runs-on: ubuntu-latest

if: github.event\_name == 'release'

steps:

- name: Deploy to Production

run: |

echo "Deploying to PRODUCTION environment"

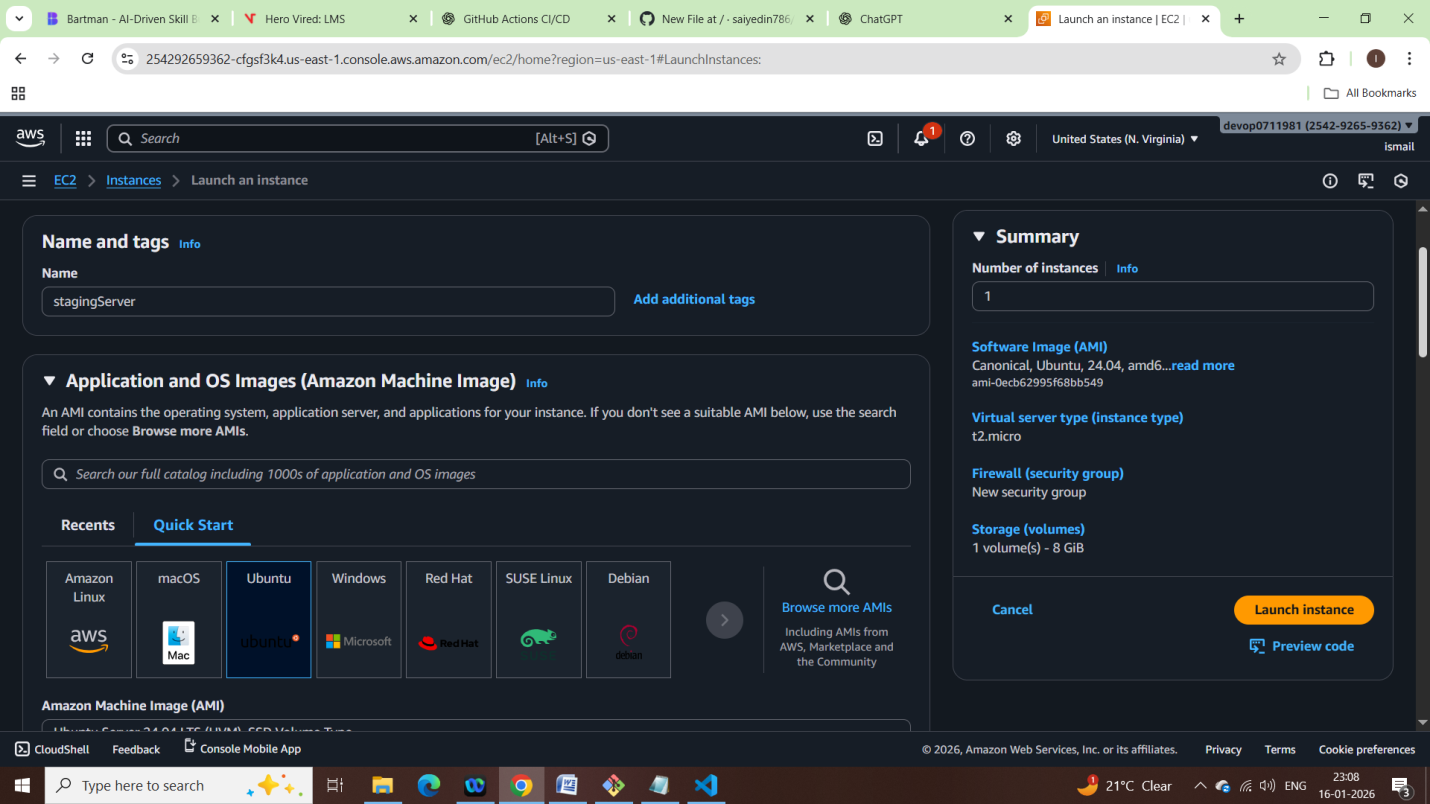
echo "Using PROD\_SECRET=${{ secrets.PROD\_SECRET }}"

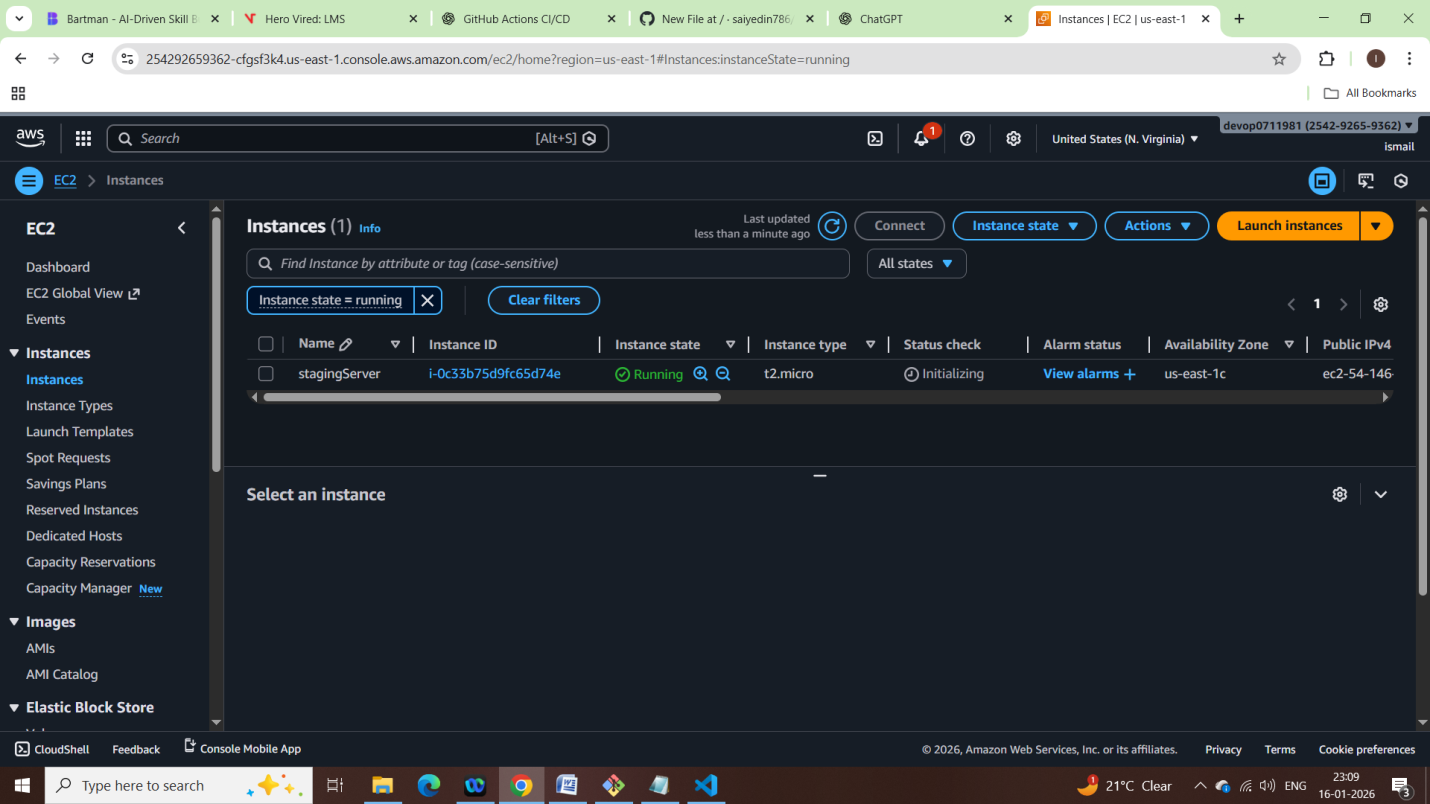
ec2 instance setup:

name : stagingServer

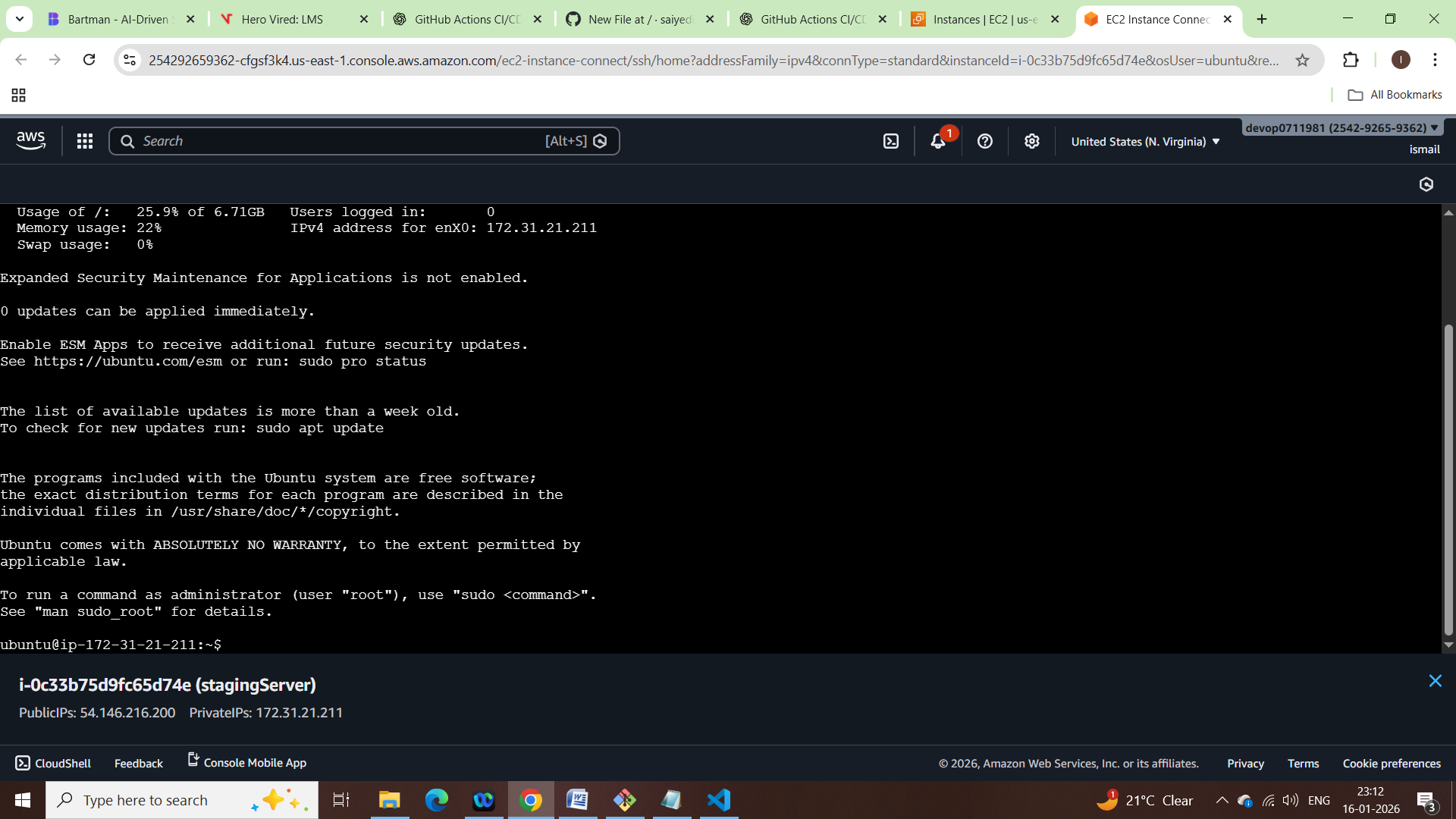
ami : ubuntu

type: t2micro





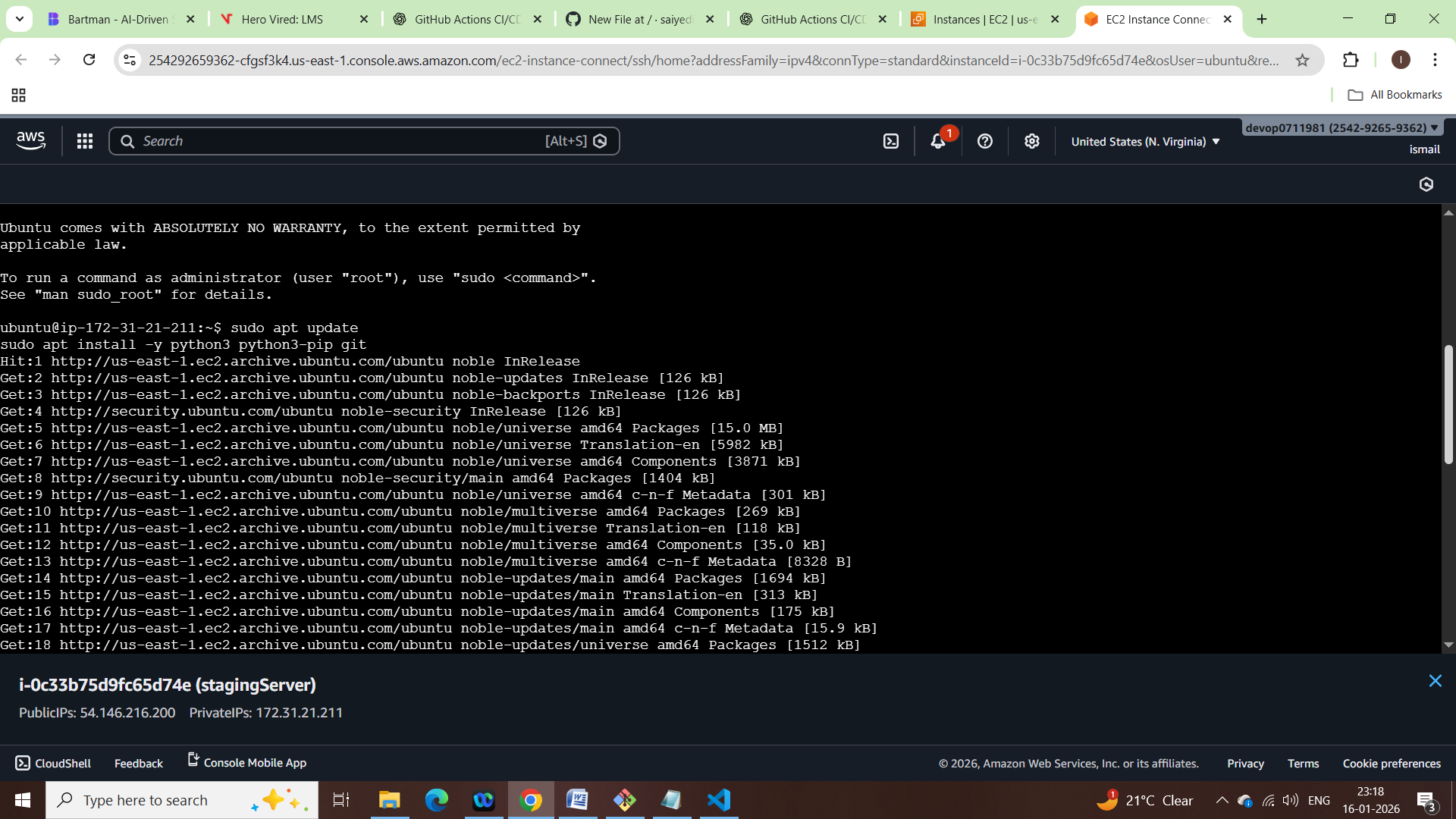
Connecting to ec2 instance using instaconnect:



Install dependencies:

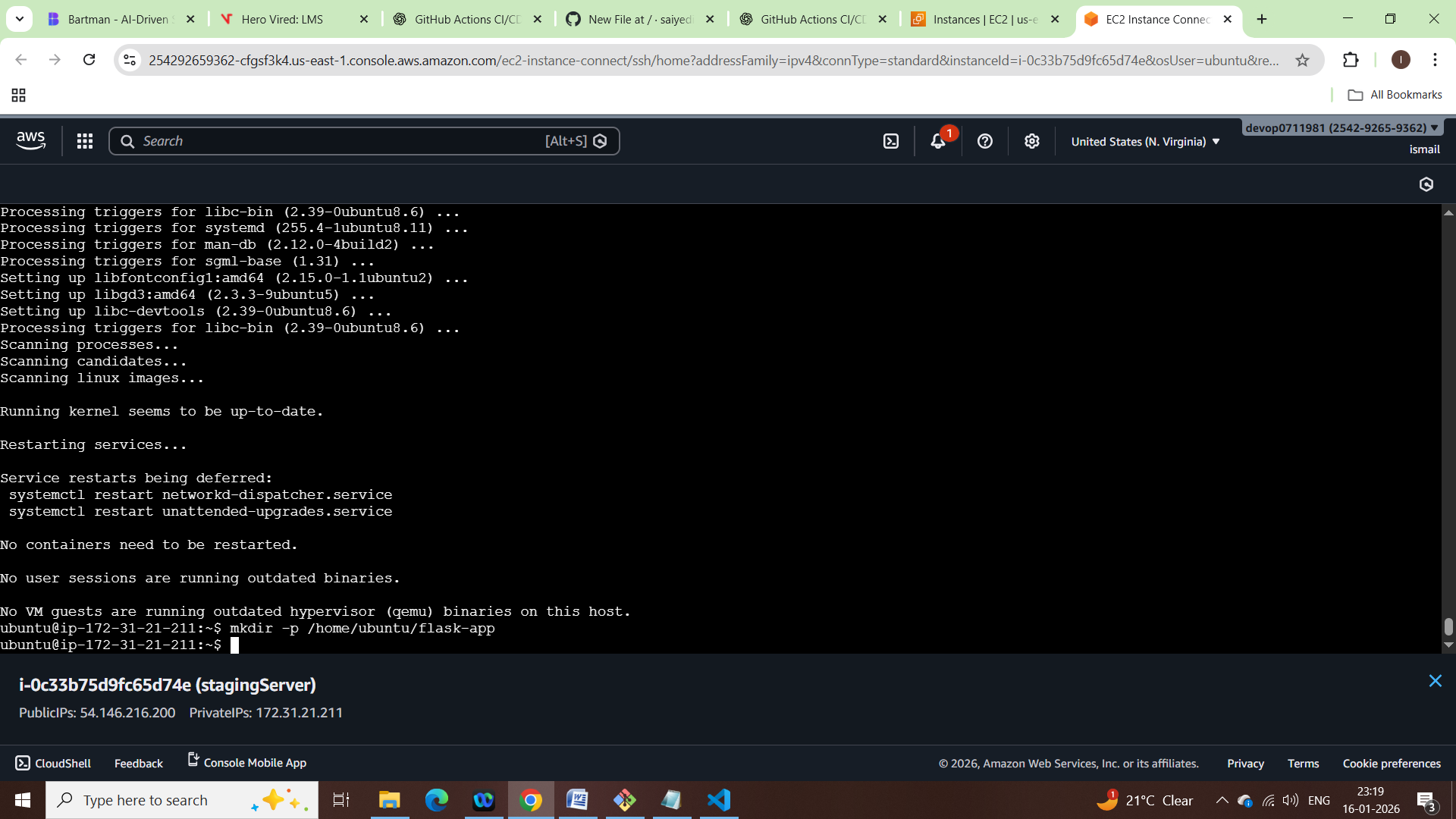
sudo apt update

sudo apt install -y python3 python3-pip git



Create App Directory

mkdir -p /home/ubuntu/flask-app



Generate SSH Key (Local Machine)

ssh-keygen -t rsa -b 4096 -C "github-actions"

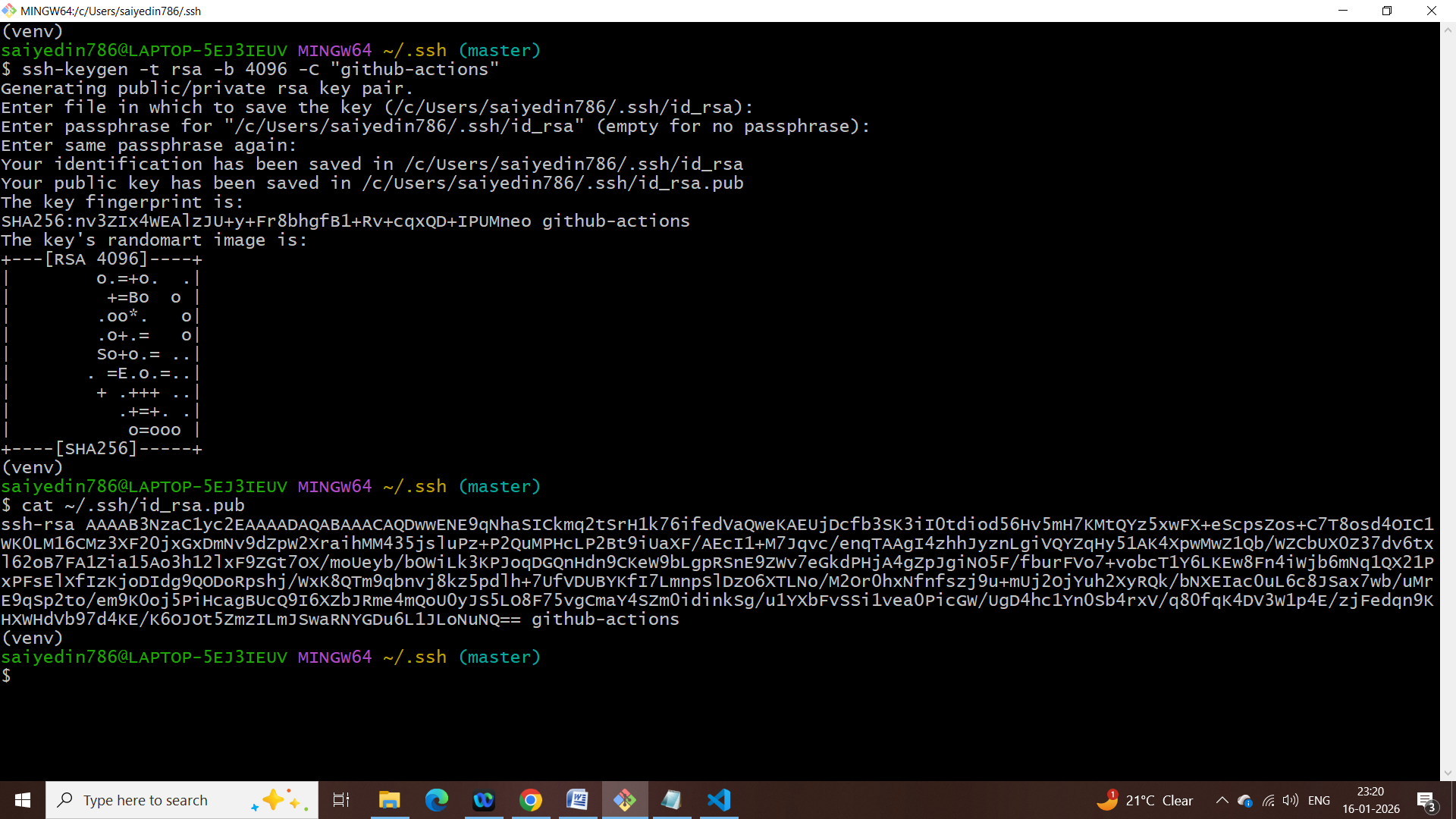
this will generate public and private key at location ~/.ssh

~/.ssh/id\_rsa (private key)

~/.ssh/id\_rsa.pub (public key)

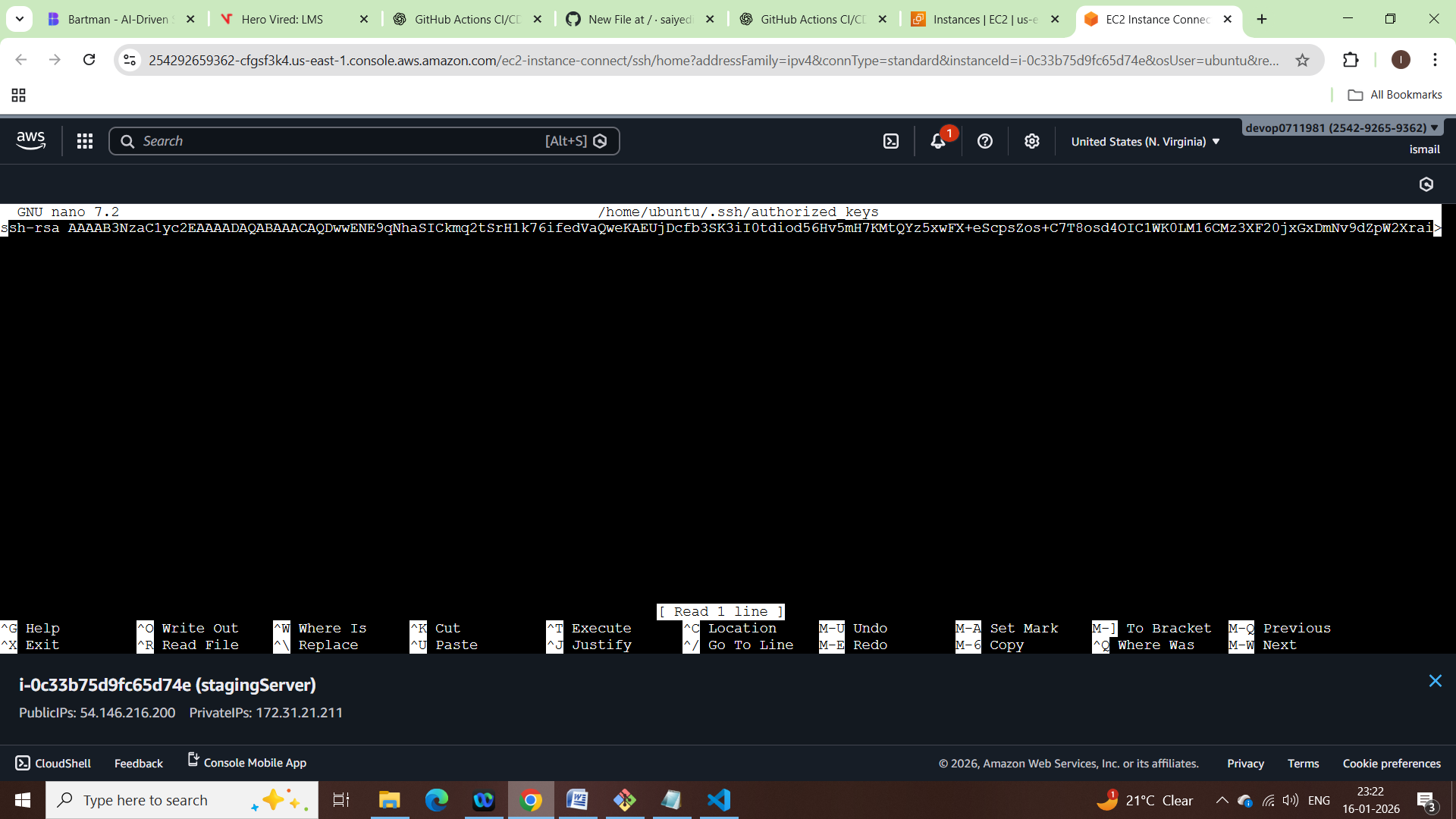
Copy content of

cat ~/.ssh/id\_rsa.pub



paste it into ec2 instance at location

nano ~/.ssh/authorized\_keys



Configure GitHub Secrets

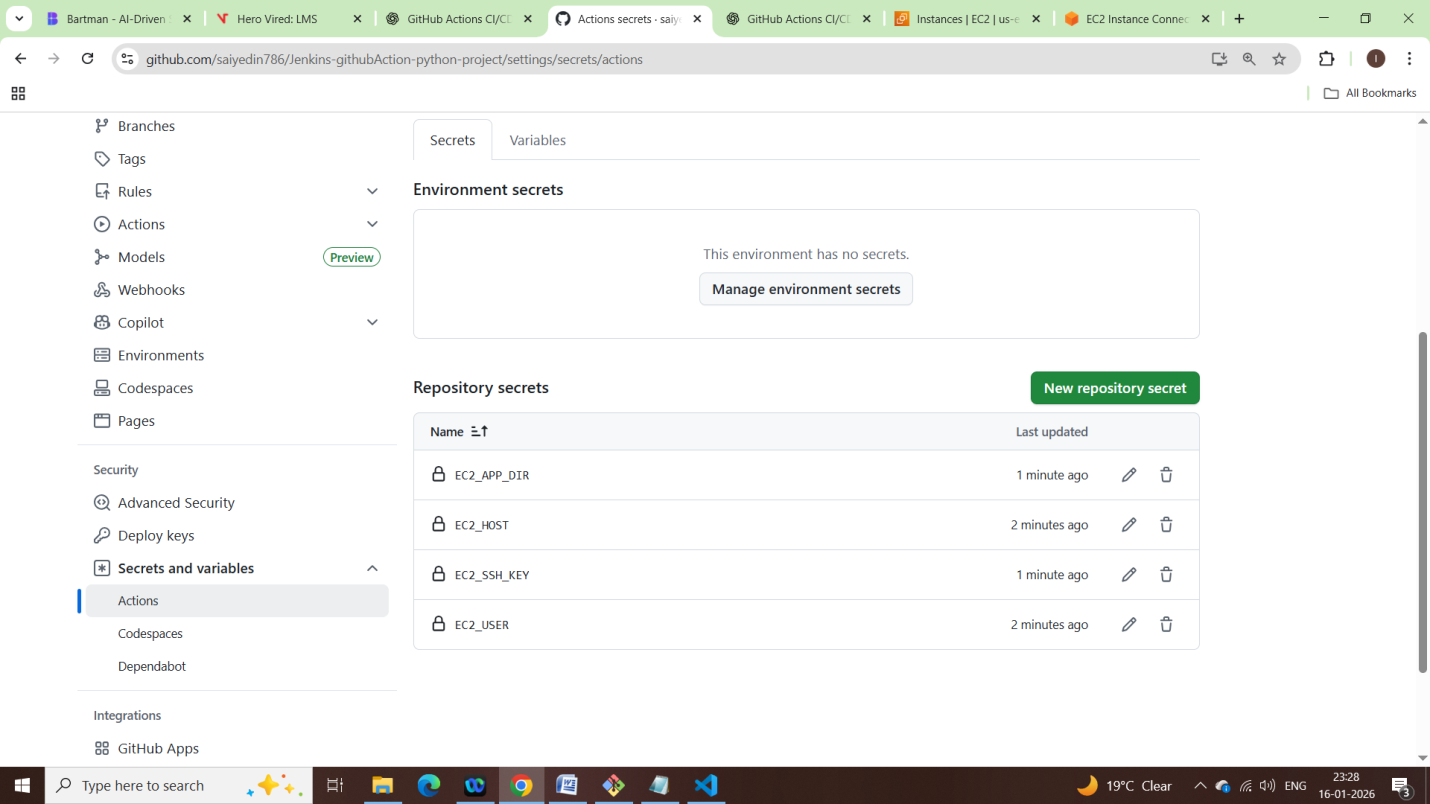
Repo → Settings → Secrets and variables → Actions

|  |  |
| --- | --- |
| EC2\_HOST | : 3.91.215.197 |

|  |  |
| --- | --- |
| EC2\_USER: | ubuntu |

|  |  |
| --- | --- |
| EC2\_SSH\_KEY: | contents of id\_rsa |

|  |  |
| --- | --- |
| EC2\_APP\_DIR | : /home/ubuntu/gitbhu-repo |
|  |  |
|  |  |



GIthub workflow -> deply.yml

name: Flask CI/CD with EC2

on:

push:

branches:

- staging

release:

types: [created]

jobs:

build-test:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Set up Python

uses: actions/setup-python@v5

with:

python-version: "3.10"

- name: Install dependencies

run: |

pip install -r requirements.txt

- name: Run tests

env:

MONGO\_URI: ${{ secrets.MONGO\_URI }}

run: pytest

deploy-staging:

needs: build-test

runs-on: ubuntu-latest

if: github.ref == 'refs/heads/staging'

steps:

- name: Deploy to EC2 (Staging)

uses: appleboy/ssh-action@v1.0.3

with:

host: ${{ secrets.EC2\_HOST }}

username: ${{ secrets.EC2\_USER }}

key: ${{ secrets.EC2\_SSH\_KEY }}

script: |

sudo apt update && sudo apt upgrade -y

git clone https://github.com/saiyedin786/Jenkins-githubAction-python-project.git

cd Jenkins-githubAction-python-project.git

python3 -m venv venv

source venv/bin/activate

pip install -r requirements.txt

pip install gunicorn

gunicorn --bind 0.0.0.0:8000 app:app

sudo nginx -t

sudo systemctl restart nginx

deploy-production:

needs: build-test

runs-on: ubuntu-latest

if: github.event\_name == 'release'

steps:

- name: Deploy to EC2 (Production)

uses: appleboy/ssh-action@v1.0.3

with:

host: ${{ secrets.EC2\_HOST }}

username: ${{ secrets.EC2\_USER }}

key: ${{ secrets.EC2\_SSH\_KEY }}

script: |

sudo apt update && sudo apt upgrade -y

git clone https://github.com/saiyedin786/Jenkins-githubAction-python-project.git

cd Jenkins-githubAction-python-project.git

python3 -m venv venv

source venv/bin/activate

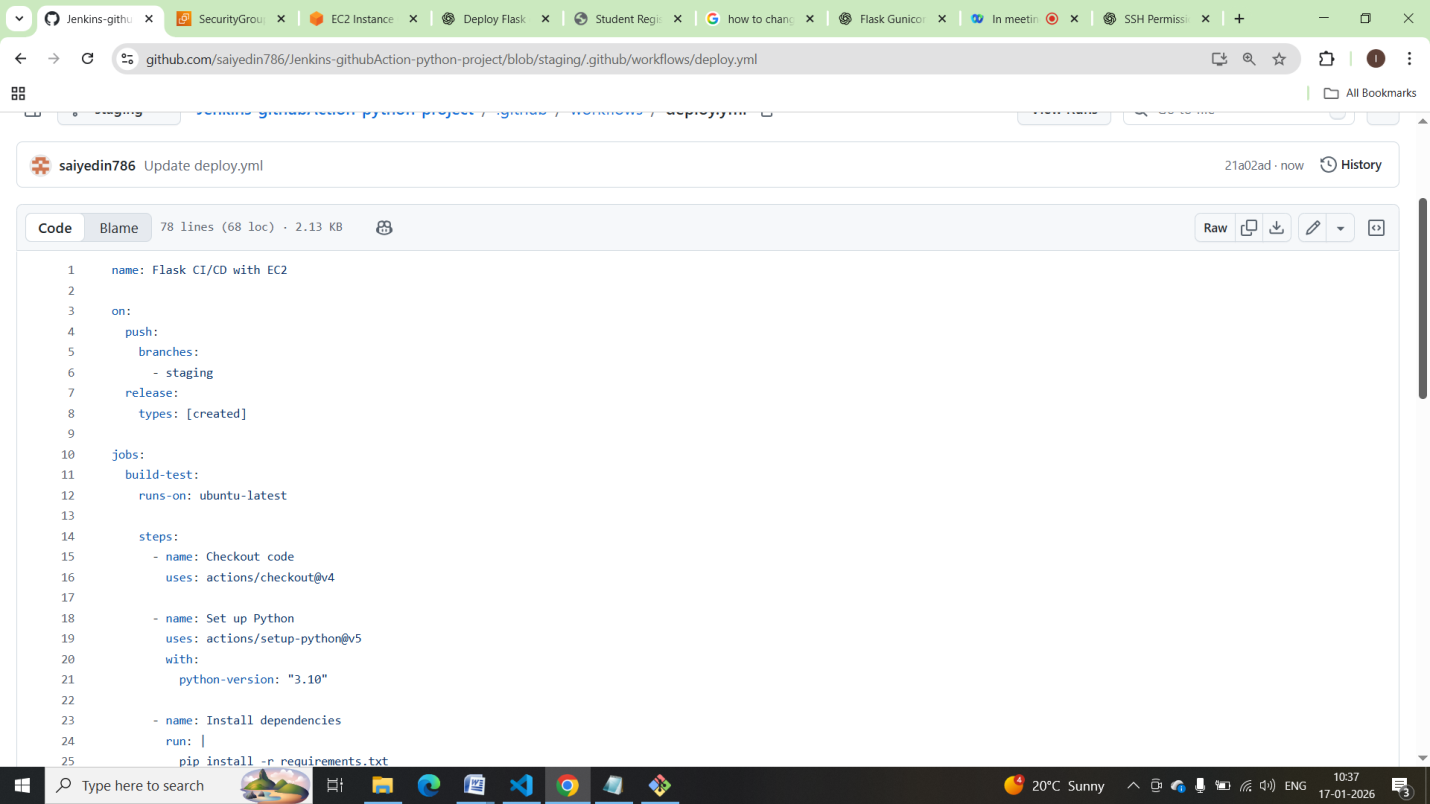
pip install -r requirements.txt

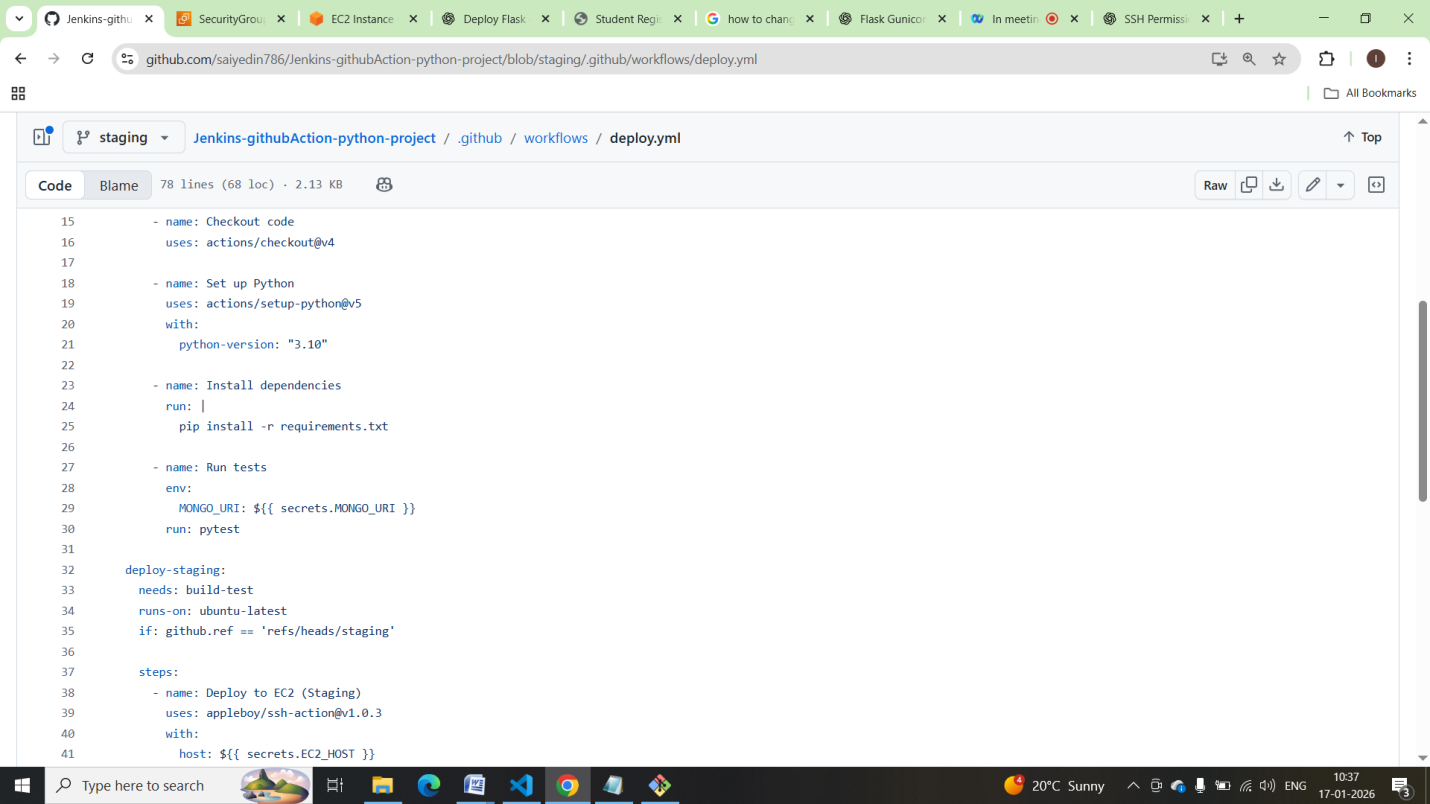
pip install gunicorn

gunicorn --bind 0.0.0.0:8000 app:app

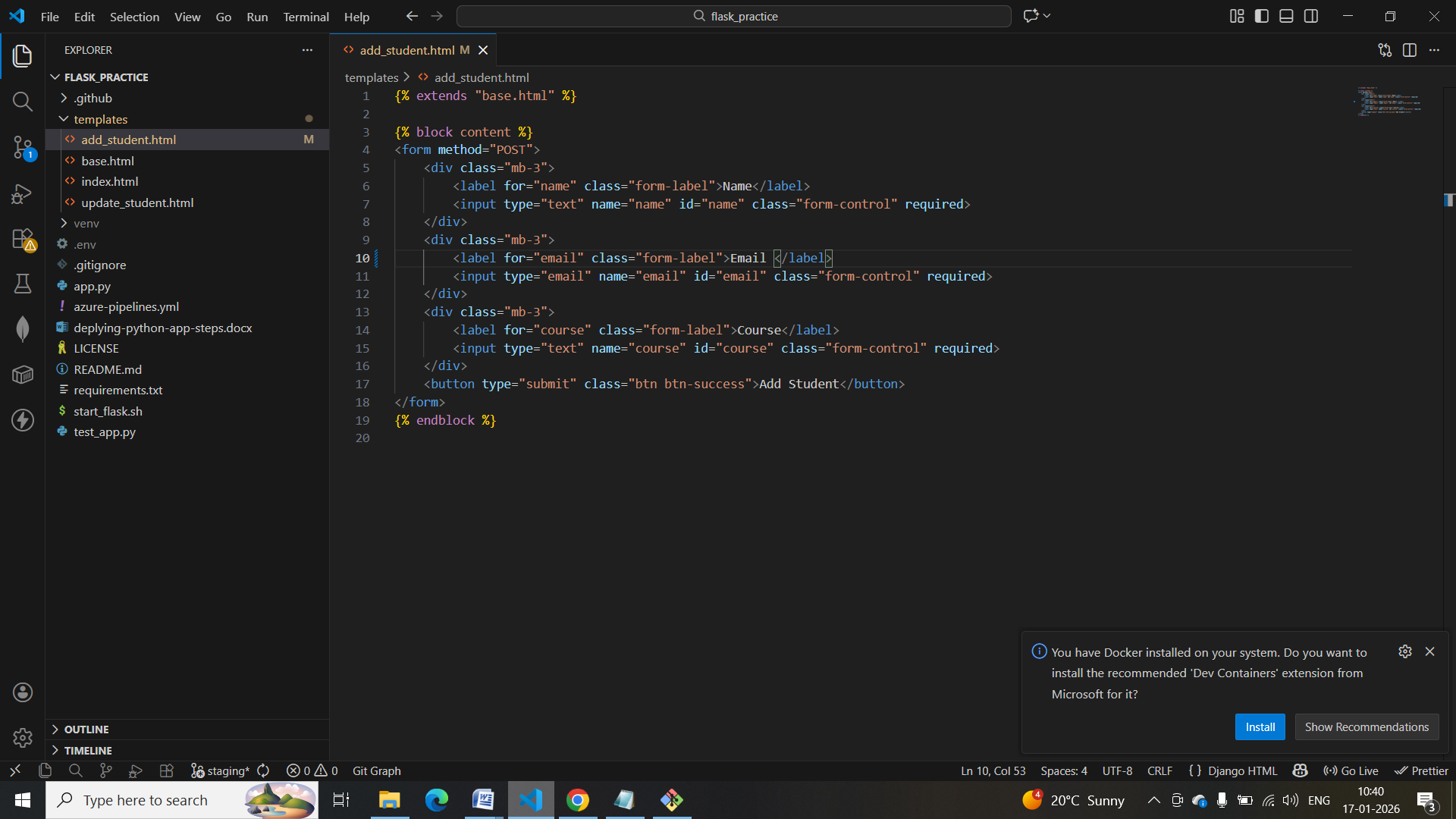
sudo nginx -t

sudo systemctl restart nginx





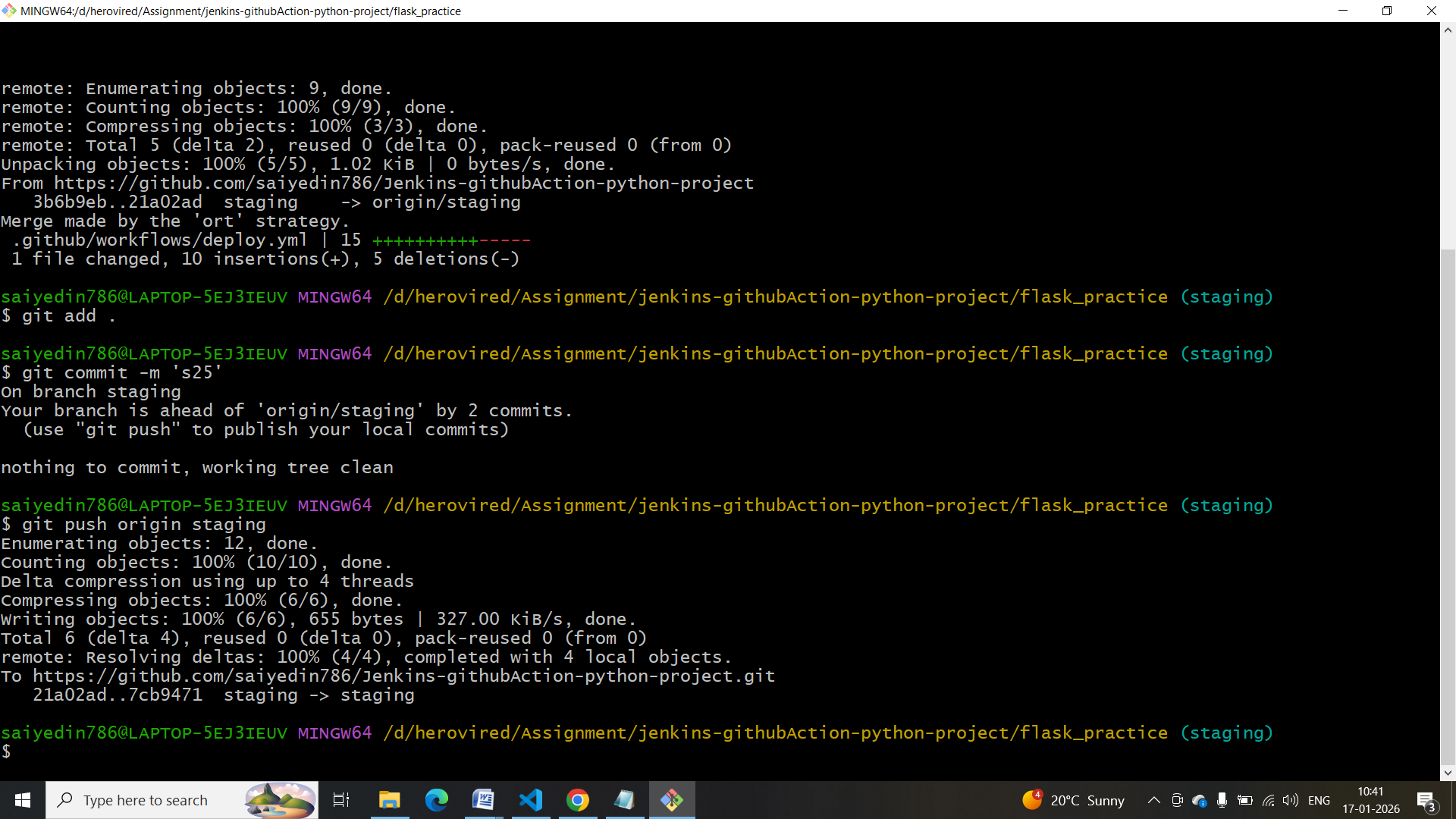
Make some changes in the add student.html and save it



Git add .

Git commit -m ‘s25’

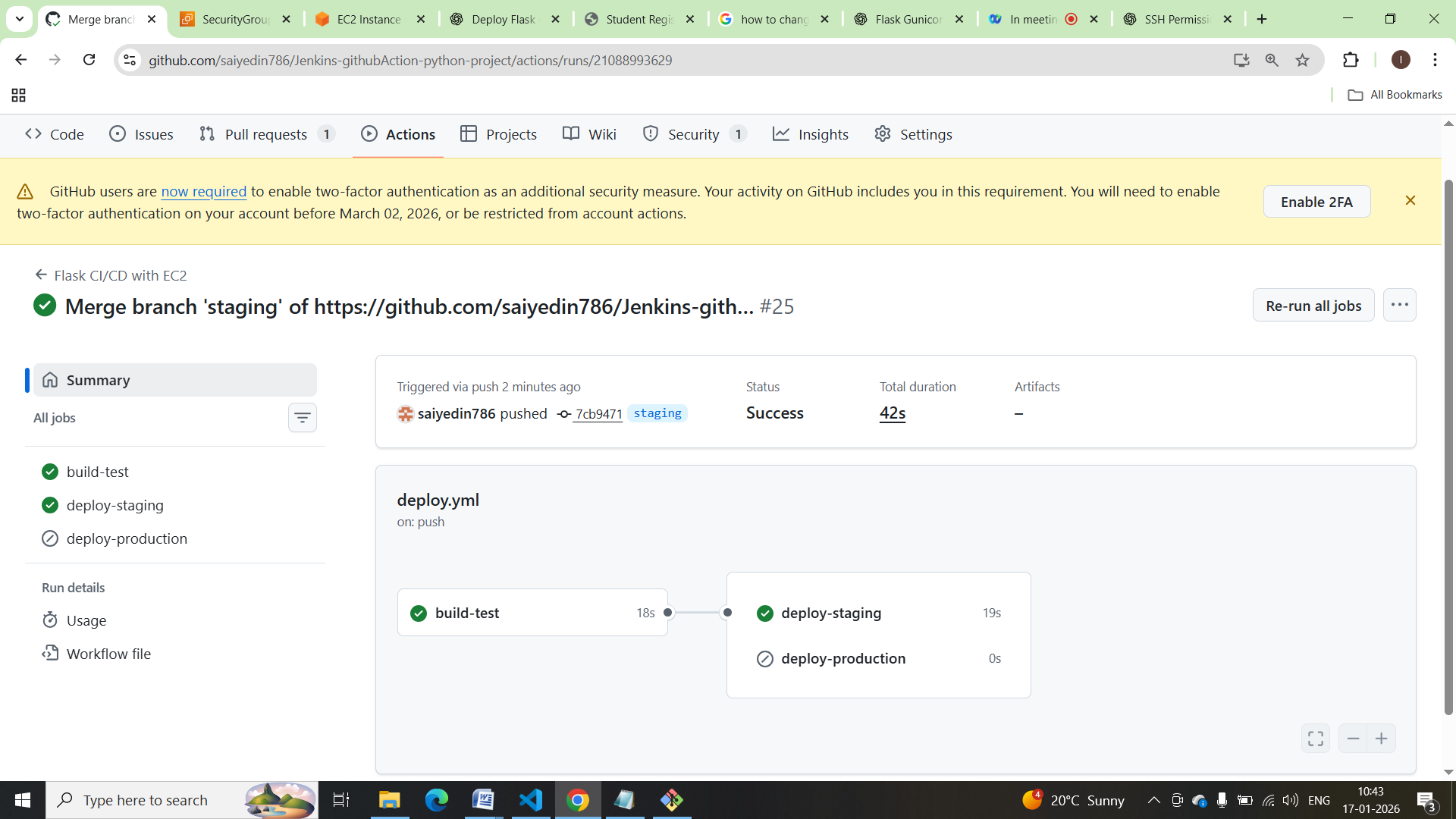
Git push origin staging



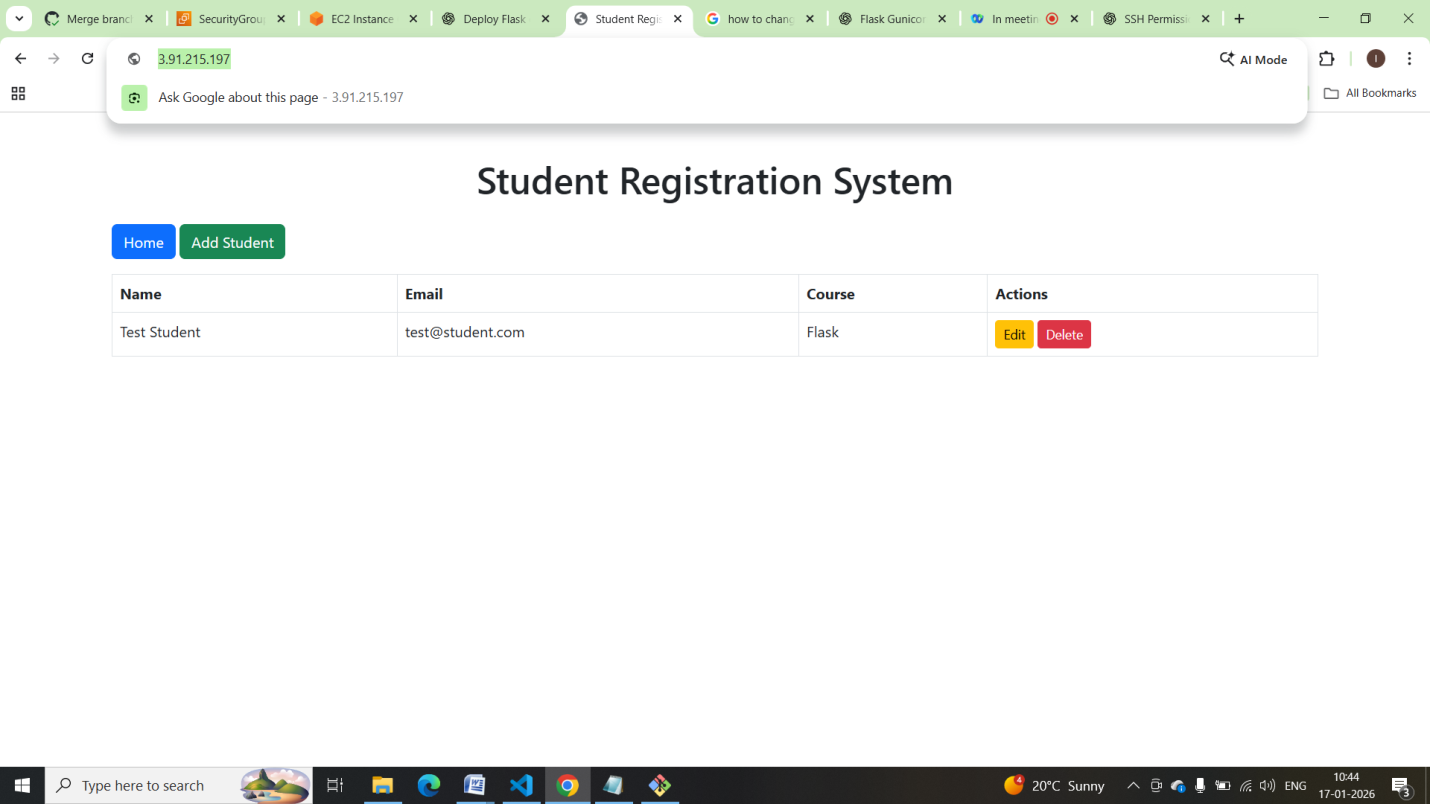
Now deploy.yml triggered

Script in deploy ran as per written steps in script

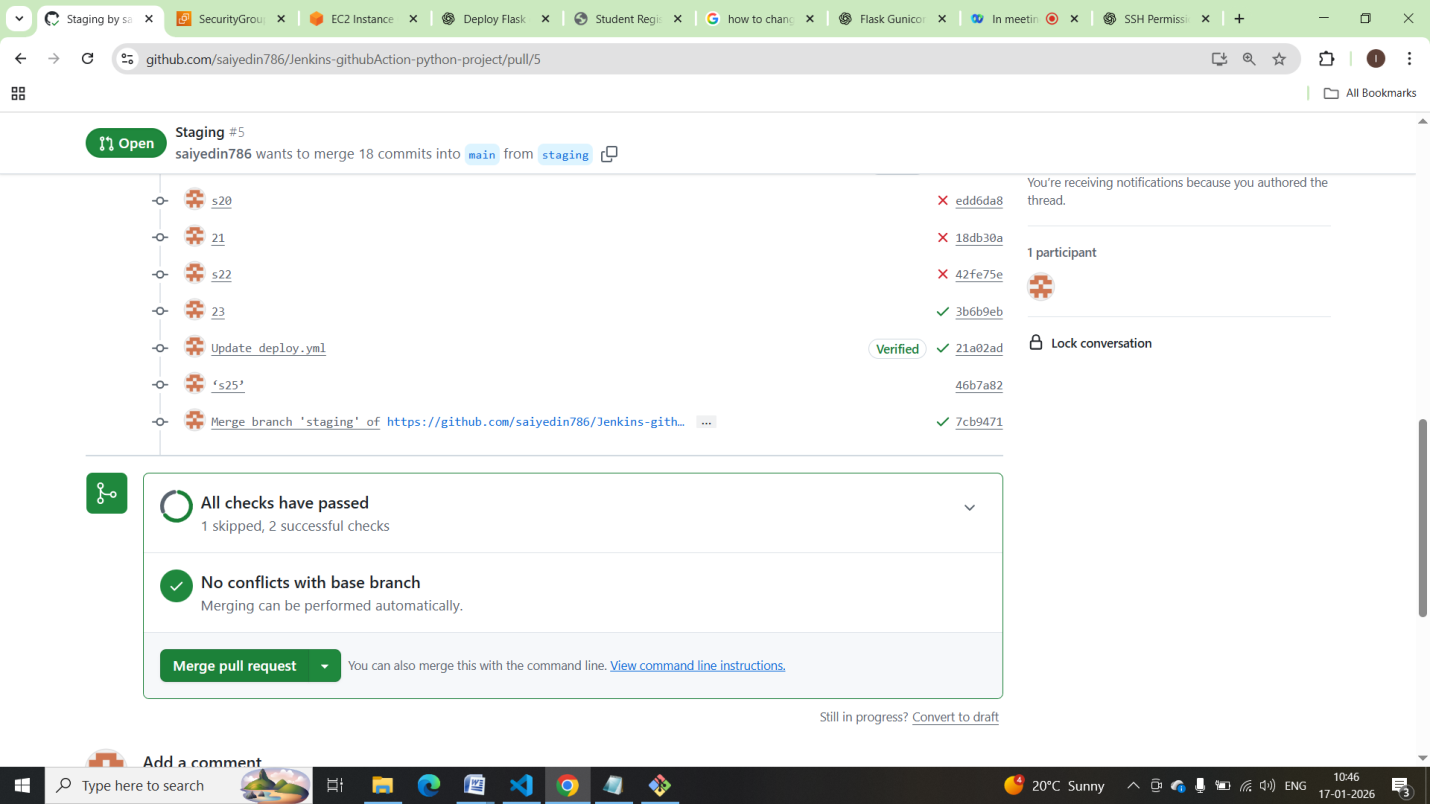
Now my app deployed in ec2 instance

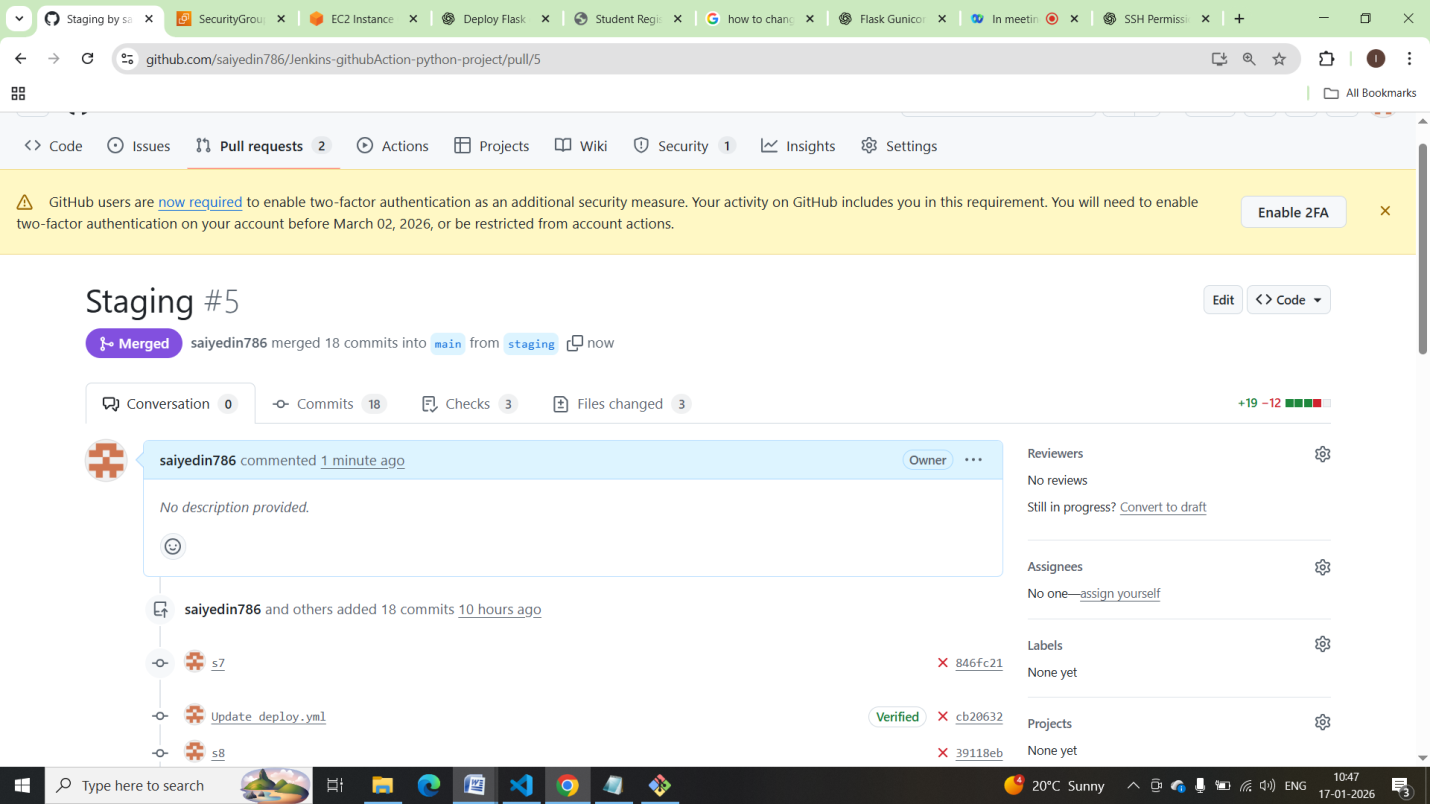


Testing in browser:

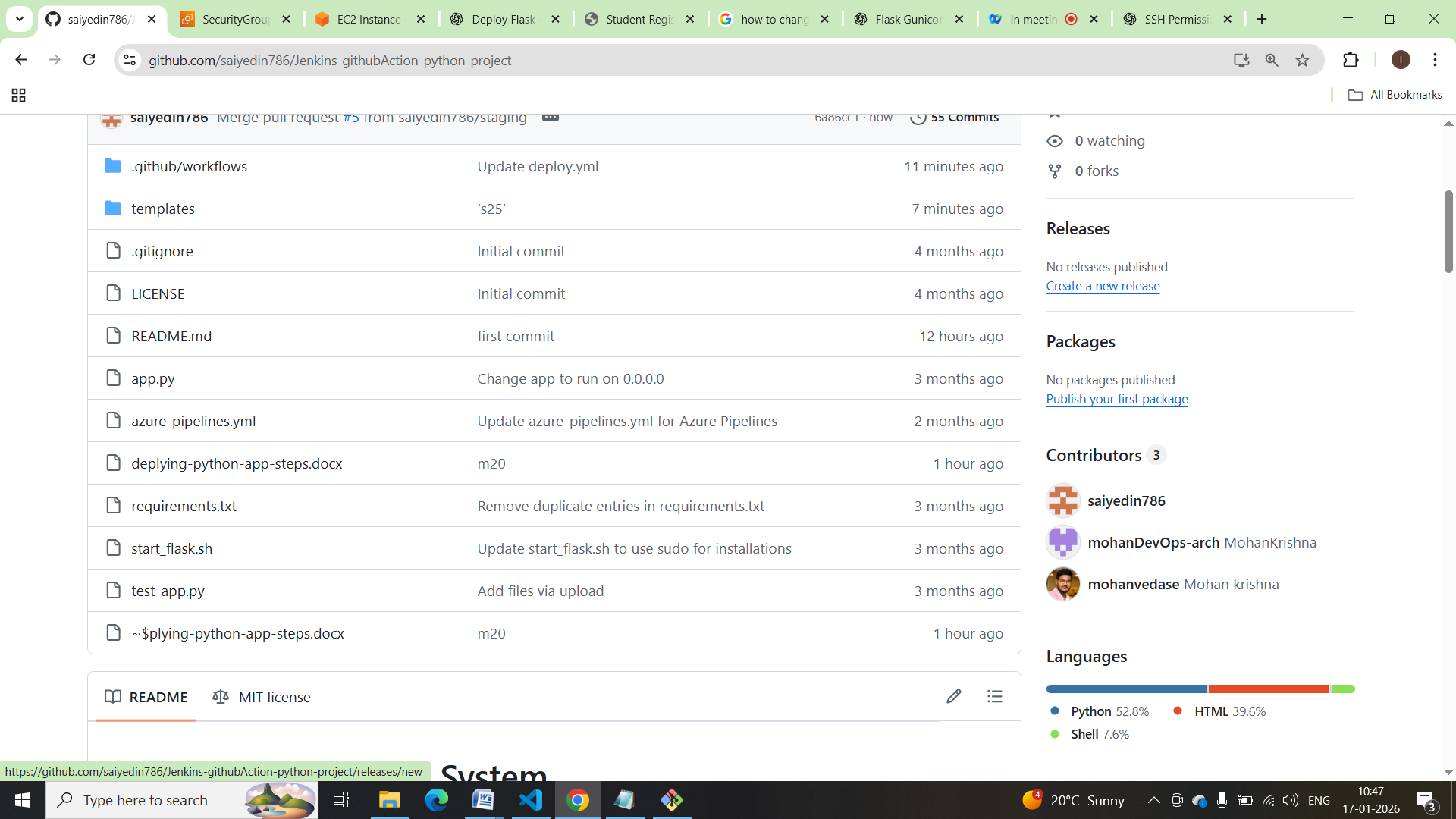


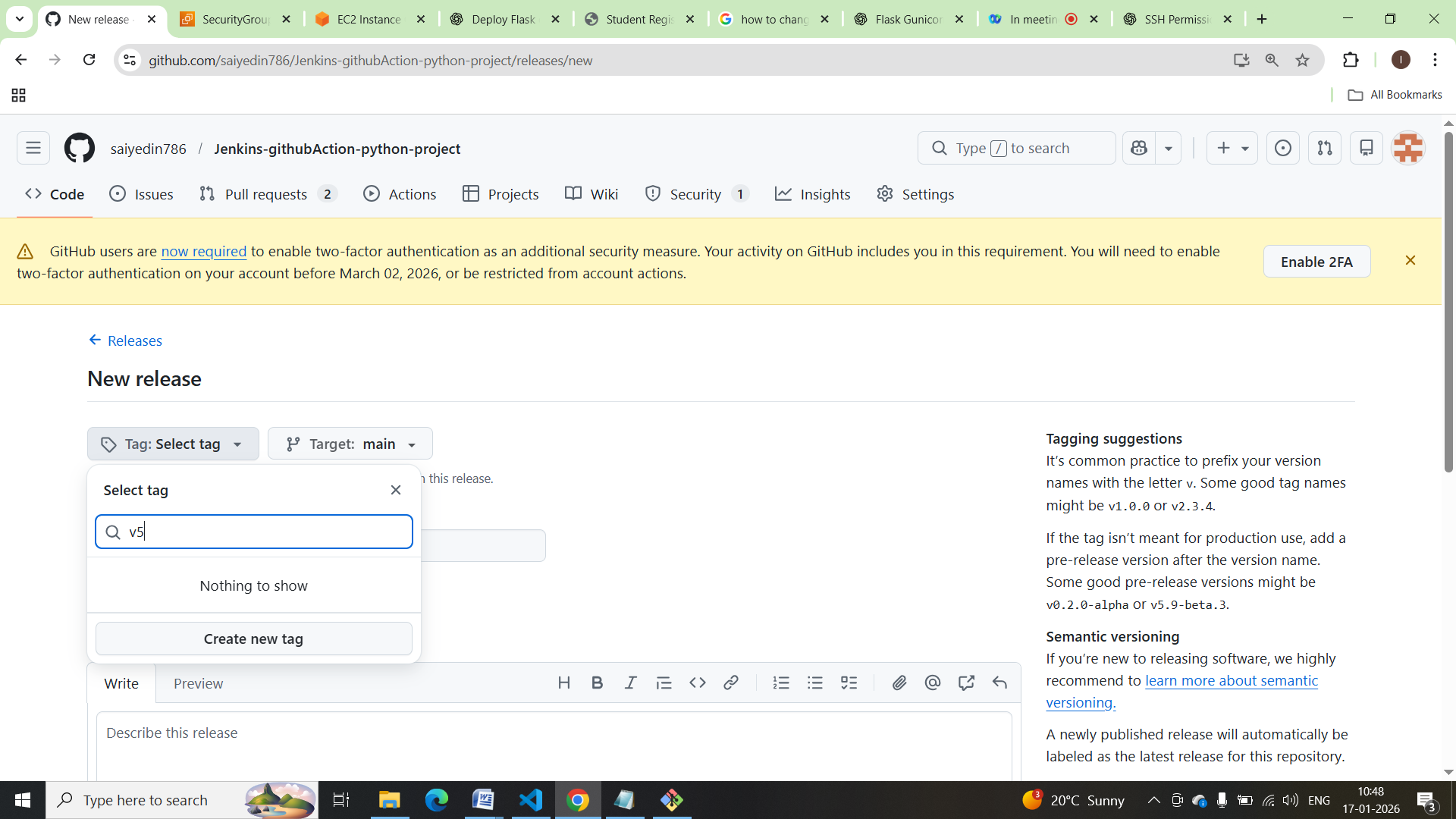
Merging it with main branch

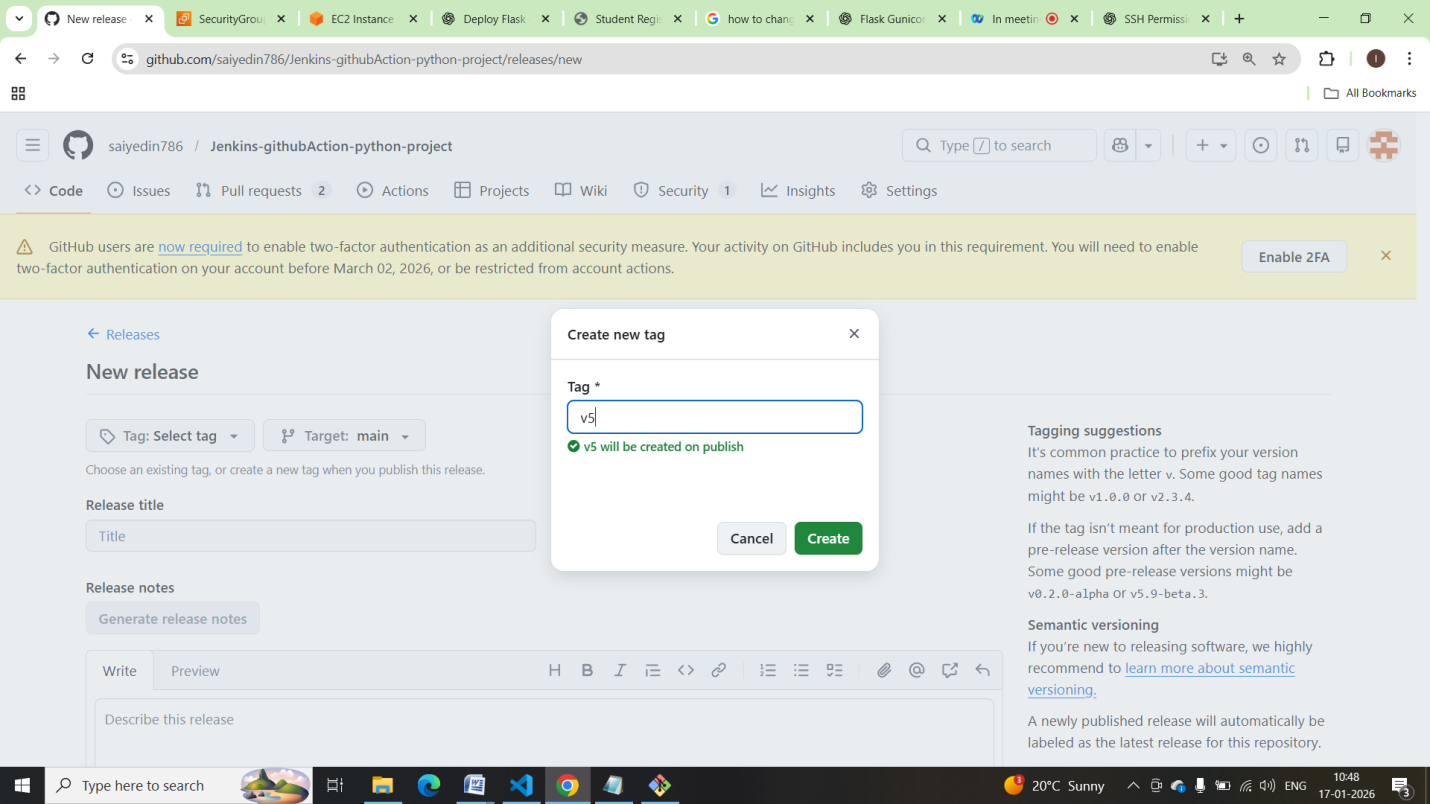


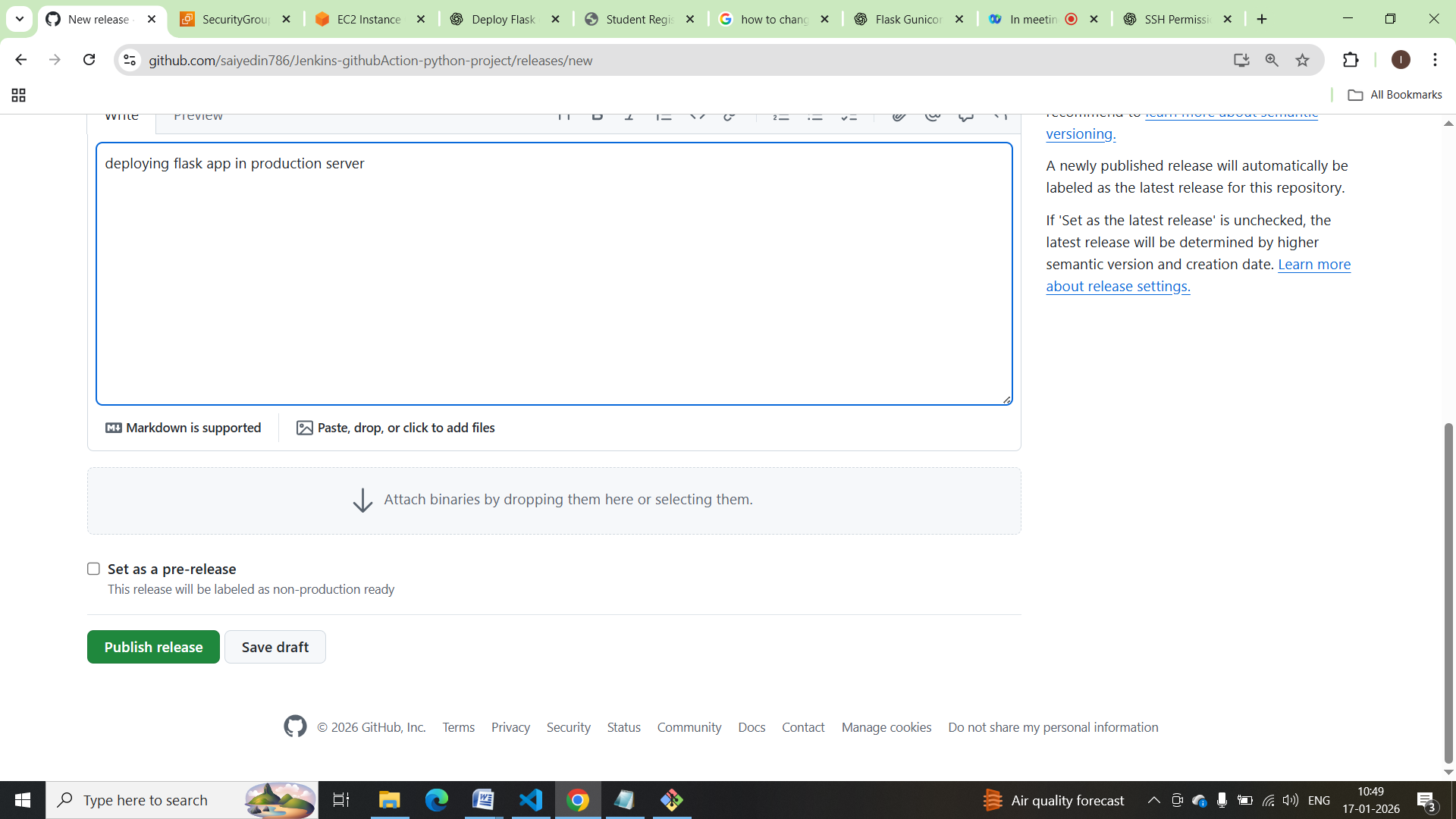


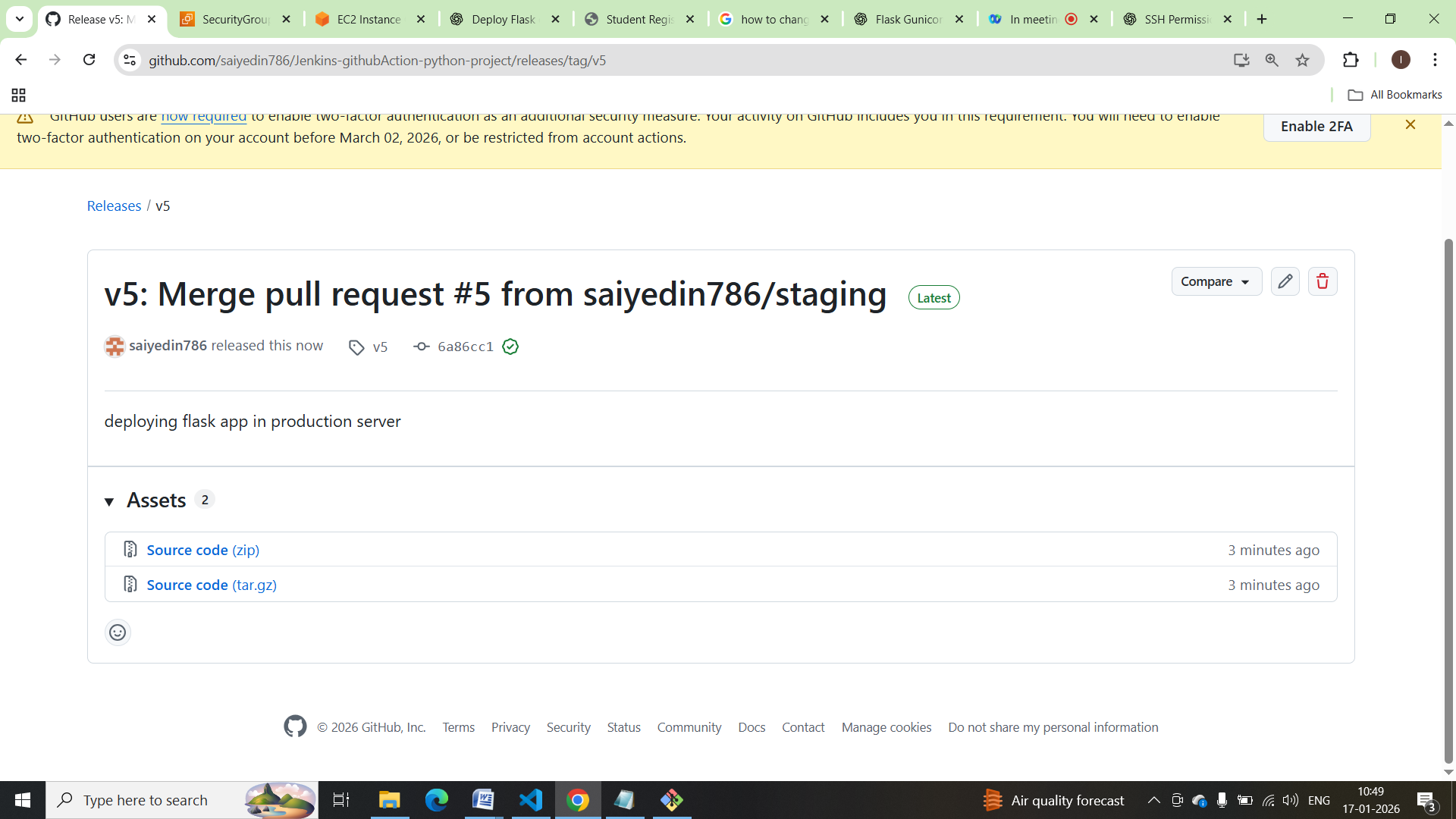
Creating a release event:



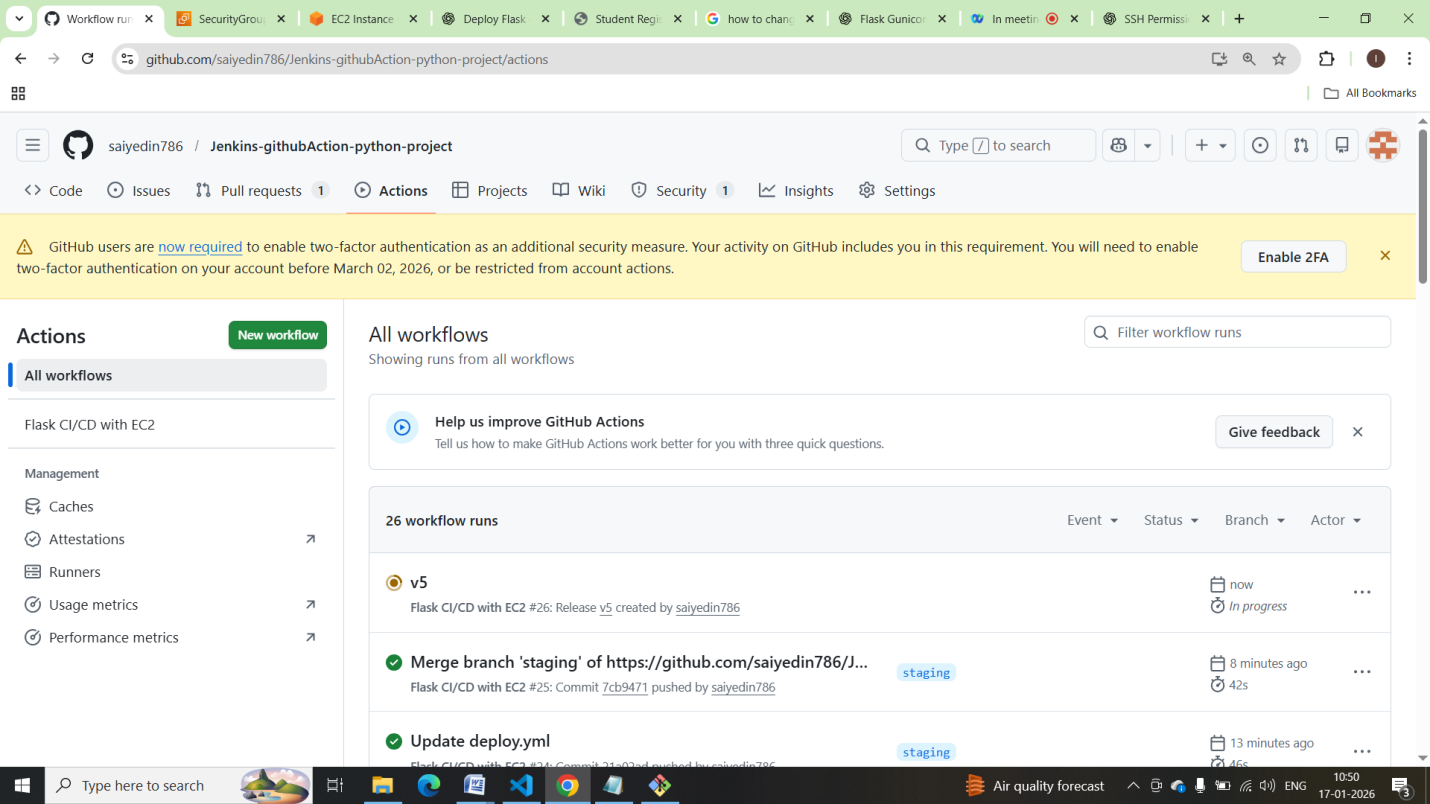


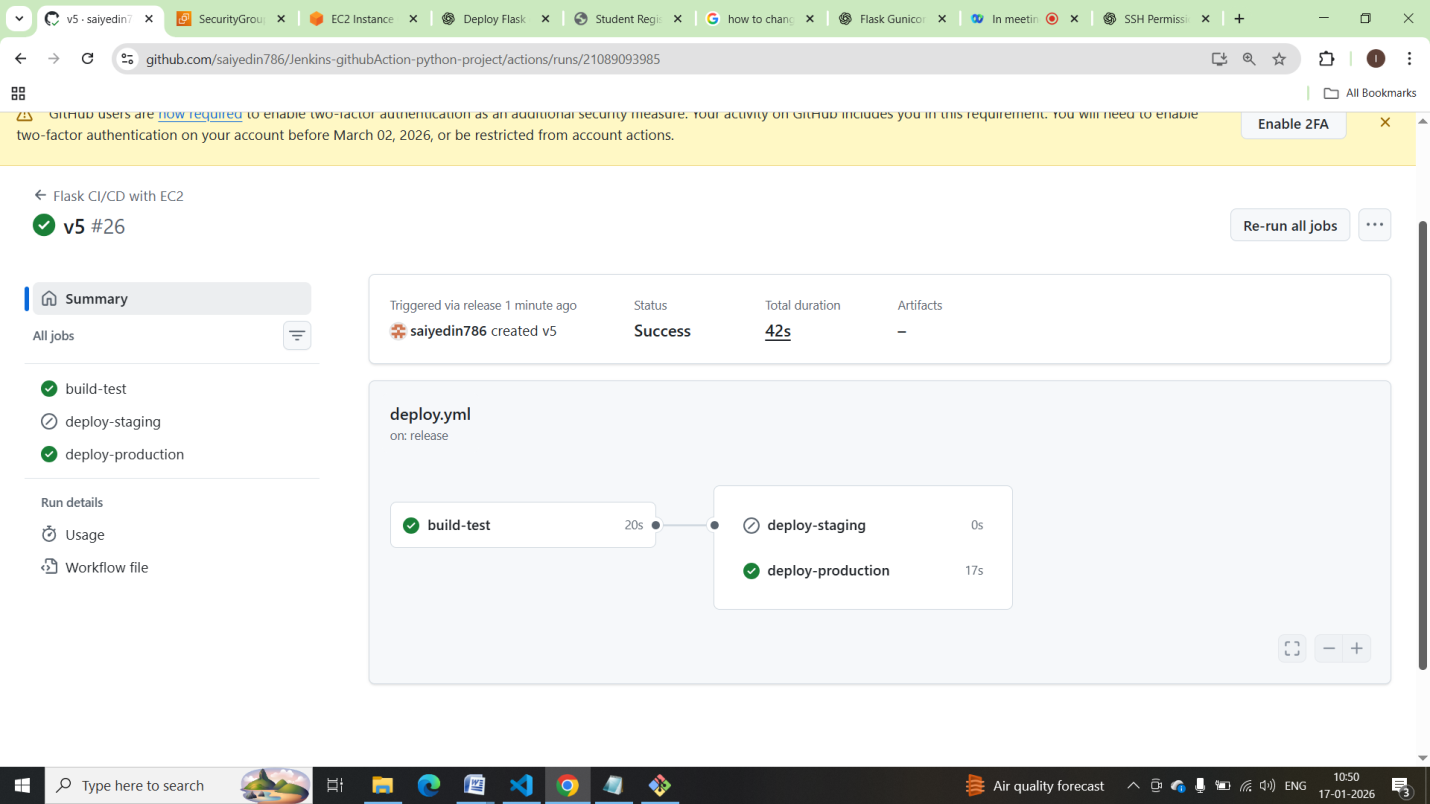


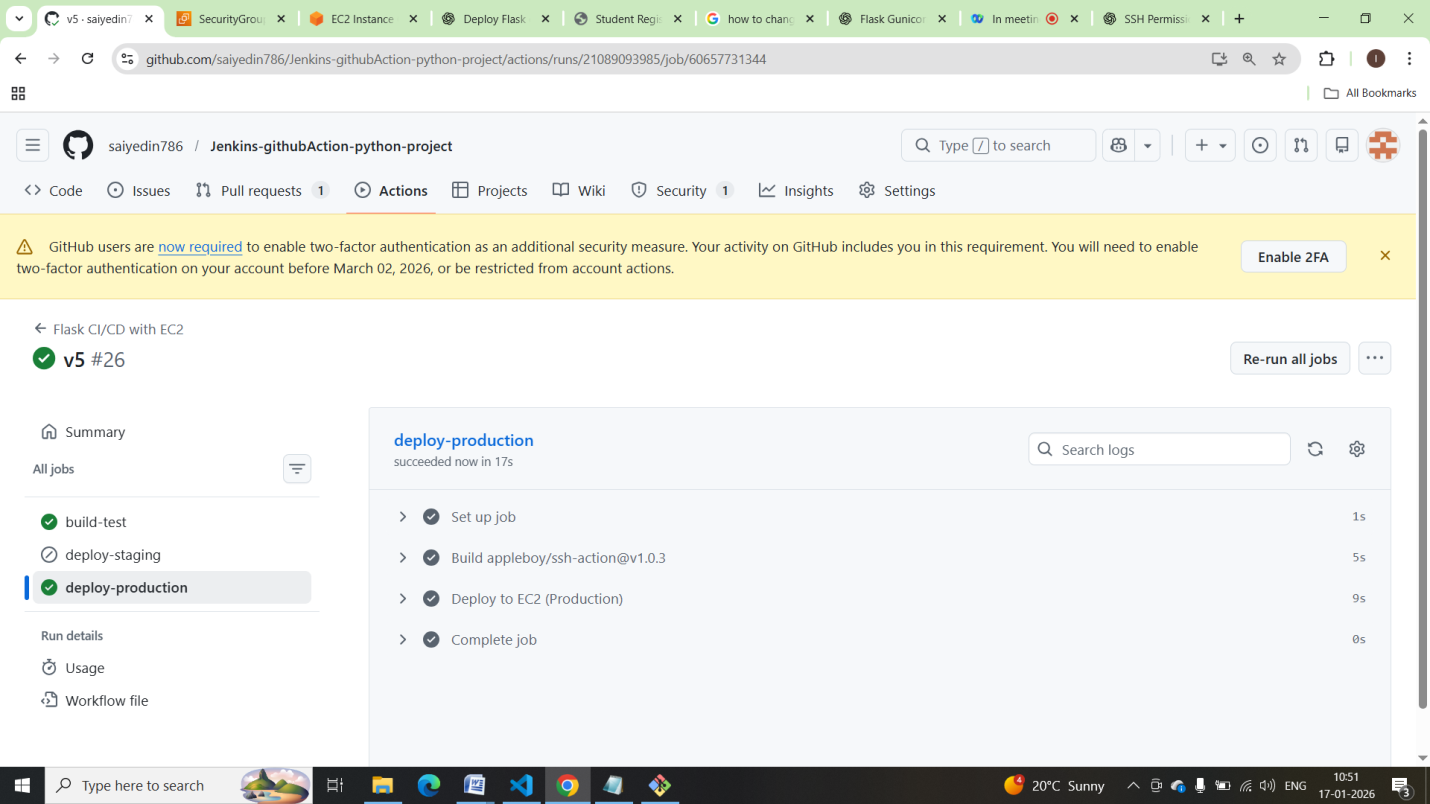




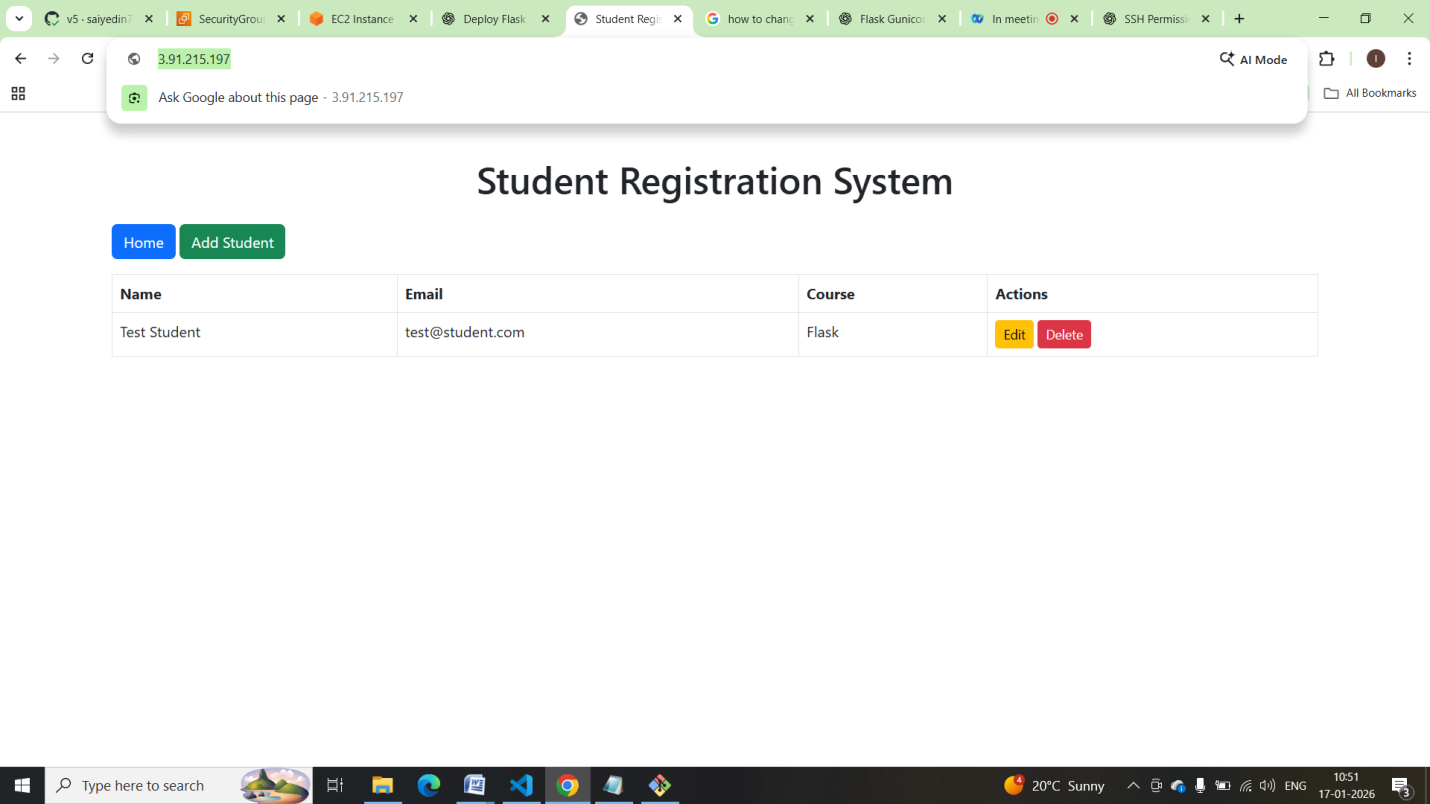
Deploy.yml triggered:





****

**Testing in browser**

****