

Task 2: GitHub Actions CI/CD Pipeline Flask App

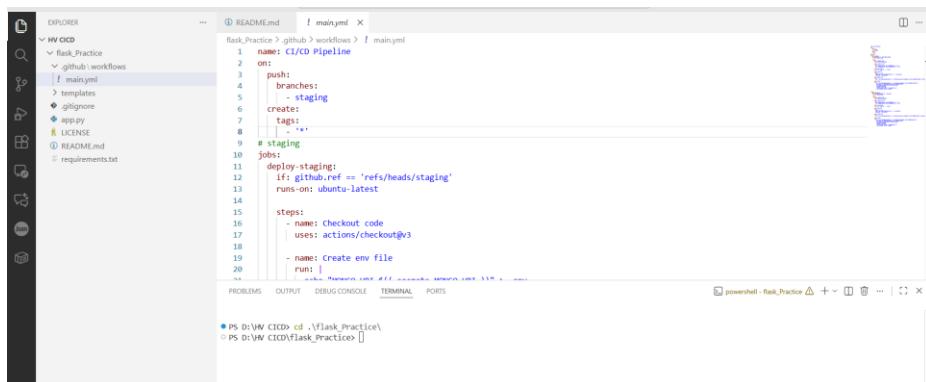
Objective:

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

Github Link: https://github.com/jatinggg/flask_Practice.git

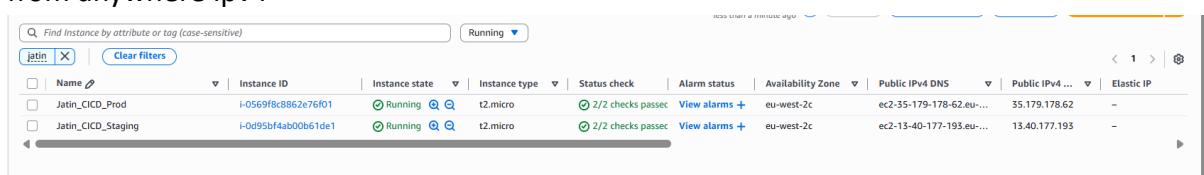
Steps:

1. Fork the provided repo in your github account.
2. Create a staging branch in your forked repo.
3. Clone the github repository in your local system.
4. Now create a new directory .github/workflows in this and add main.yml file with following code in both main and staging branch:



```
name: CI/CD Pipeline
on:
  pushes:
    branches:
      - staging
  create:
    tags:
      - '*'
  jobs:
    deploy-staging:
      if: github.ref == 'refs/heads/staging'
      runs-on: ubuntu-latest
      steps:
        - name: Checkout code
          uses: actions/checkout@v3
        - name: Create env file
          run: |
```

5. Launch 2 ec2 instances (for prod and staging) with inbound rules allowed at port 8000 from anywhere ipv4



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
Jatin_CICD_Prod	i-0569f8c8862e76f01	Running	t2.micro	2/2 checks passed	View alarms +	eu-west-2c	ec2-35-179-178-62.eu...	35.179.178.62	-
Jatin_CICD_Staging	i-0d95bf4ab0b61de1	Running	t2.micro	2/2 checks passed	View alarms +	eu-west-2c	ec2-13-40-177-193.eu...	13.40.177.193	-

6. SSH into ec2 instance and create a flaskapp.service under the sysemd directory (configure for both the environments)

Run below commands:

```
sudo apt get update
```

```
sudo apt install -y python3-pip python3-venv unzip
```

```
sudo mkdir -p /home/ubuntu/app
```

```
sudo chown ubuntu:ubuntu /home/ubuntu/app
```

```
sudo nano /etc/systemd/system/flaskapp.service
```

flaskapp.service file :

```
[Unit]
```

```
Description=Gunicorn instance to serve Flask App
```

```
After=network.target
```

```
[Service]
```

```
User=ubuntu
```

```
Group=ubuntu
```

```
WorkingDirectory=/home/ubuntu/app
```

```
Environment="PATH=/home/ubuntu/app/venv/bin:/usr/local/bin:/usr/bin"
```

```
EnvironmentFile=/home/ubuntu/app/.env
```

```
# FIX IS HERE:
```

```
# 1. Point to the 'venv' gunicorn
```

```
# 2. Bind to 0.0.0.0:5000 (Required for browser access)
```

```
ExecStart=/home/ubuntu/app/venv/bin/gunicorn --workers 3 --bind 0.0.0.0:5000 app:app
```

```
Restart=always
```

```
[Install]
```

```
WantedBy=multi-user.target
```

now run below commands:

```
sudo systemctl daemon-reload
```

```
sudo systemctl enable flaskapp.service
```

```
sudo systemctl start flaskapp.service
```

7. Update the required dependencies in the requirements.txt file on the visual studio
8. Create a MongoDB cluster with traffic allowed from all ip addresses.
9. On the GitHub repository add below secrets from settings > secrets and variables > actions > New Repository Secret (for SRCRET_KEY use any random string, I used 123456)

Repository secrets

Name	Last updated		New repository secret
MONGO_URI	17 hours ago		
PROD_EC2_HOST	18 hours ago		
PROD_EC2_KEY	18 hours ago		
SECRET_KEY	18 hours ago		
STAGING_EC2_HOST	18 hours ago		
STAGING_EC2_KEY	18 hours ago		

10. Now in the visual studio make any changes in any file (make sure you have switched to the staging branch using "git switch staging").
11. You will need to create a Personal Access Token from GitHub with permission to workflow and repository and configure it on your local machine.

The screenshot shows the GitHub settings page for tokens. A new token named "Github-actions" has been created. The token has the following scopes selected:

- repo
- repos:status
- repos:deployment
- public_repo
- repos:invite
- security_events
- workflow
- write_packages
- read_packages

```
git remote set-url origin https://<your-PAT>@github.com/jatinggg/flask_Practice.git
```

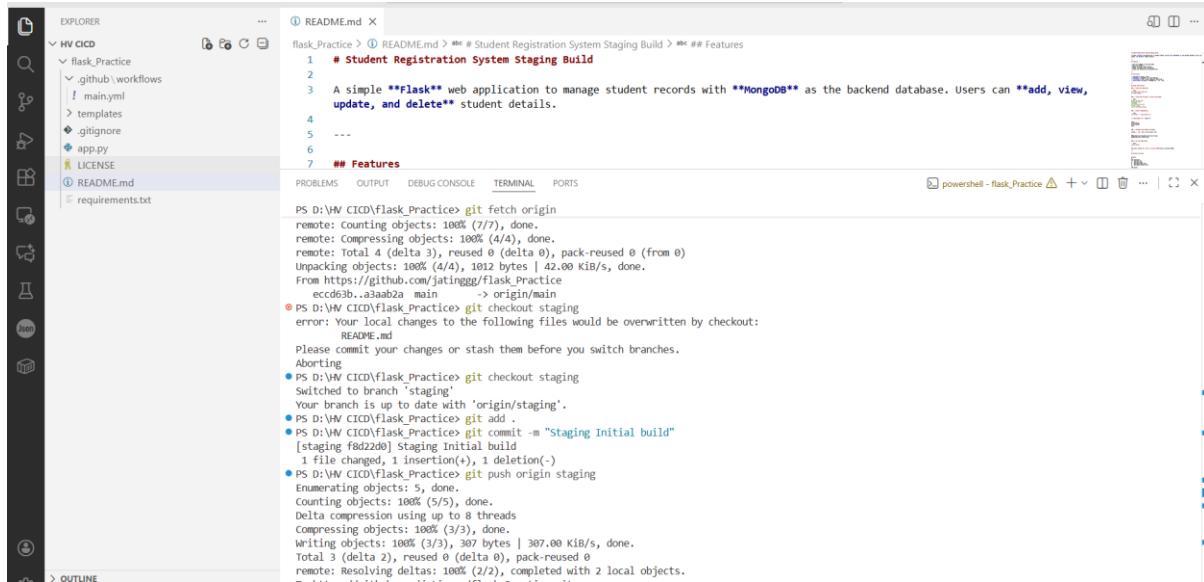
12. Now run below commands to trigger the CI/CD pipeline build

```
git checkout staging
```

```
git add .
```

```
git commit -m "Staging Initial build"
```

```
git push origin staging
```

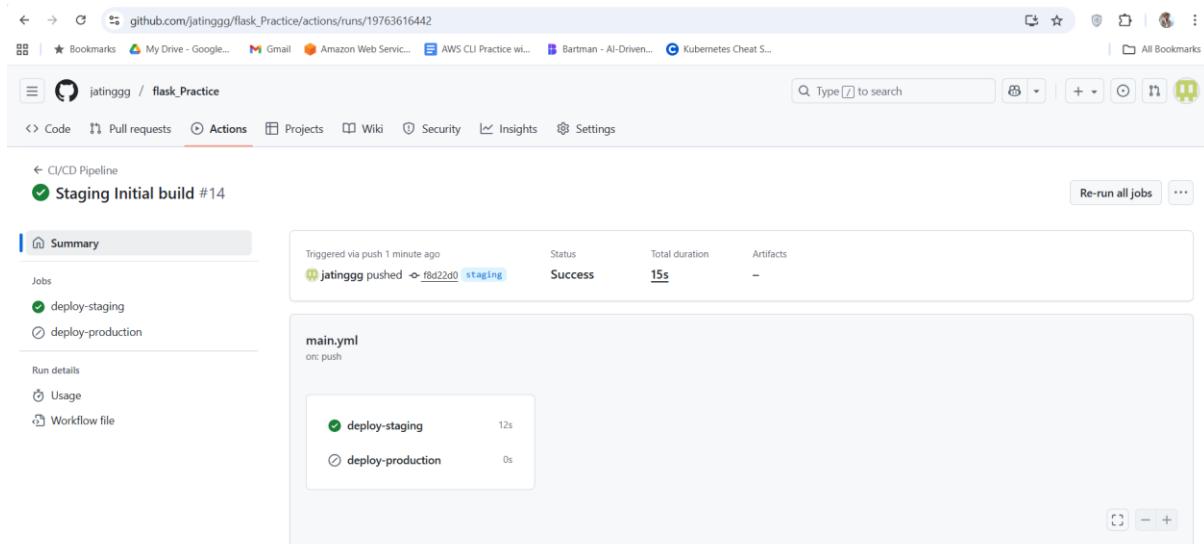


The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays a sequence of commands and their outputs:

```
git add .
git commit -m "Staging Initial build"
git push origin staging
```

The terminal also shows the file structure of the project, which includes a `HV CI/CD` folder containing `flask_Practice`, `.github\workflows`, `main.yml`, `templates`, `gitignore`, `app.py`, `LICENSE`, `README.md`, and `requirements.txt`.

13. Verify the build from the Github Actions console



The screenshot shows the GitHub Actions console for the repository `jatinggg/flask_Practice`. It displays the summary of a recent build:

Staging Initial build #14

Summary

Triggered via push 1 minute ago
Status: Success
Total duration: 15s
Artifacts: -

main.yml

on: push

deploy-staging: 12s
deploy-production: 0s

Re-run all jobs

14. SSH into the Staging server and look under the root directory you can find the app.zip artifact and this file will be used to push the changes to the prod server once it has been tested.

```

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 Fri Nov 28 12:26:41 UTC 2025

System load: 0.0          Processes:           113
Usage of /: 35.6% of 6.71GB  Users logged in: 0
Memory usage: 37%          IPv4 address for enx0: 172.31.3.29
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

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

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Thu Nov 27 20:08:29 2025 from 49.43.41.217
ubuntu@ip-172-31-3-29:~$ ls -a
. .bash_history .bash_logout .bashrc .cache .profile .ssh .sudo_as_admin_successful app app.zip
ubuntu@ip-172-31-3-29:~$ ||


```

i-0d95bf4ab00b61de1 (Jatin_CICD_Staging)
PublicIPs: 13.40.177.193 PrivateIPs: 172.31.3.29

CloudShell Feedback Console Mobile App

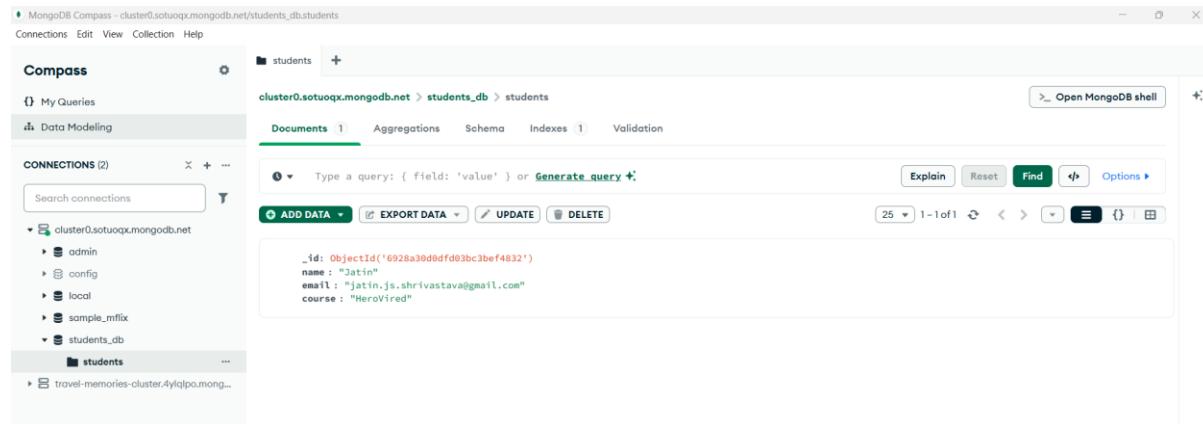
15. You can now access the application at the staging server at port 8000

Student Registration

Not secure 13.40.177.193:8000

Name	Email	Course	Actions
Jatin	jatin.js.shrivastava@gmail.com	HeroVired	Edit Delete

you can also verify this data in the mongodb compass



16. Now that we have verified the application and it has passed all the test cases on the staging server, we can push the code to the prod server.

In the visual studio switch to the main branch and run following commands:

```
git fetch origin
```

```
git pull origin main
```

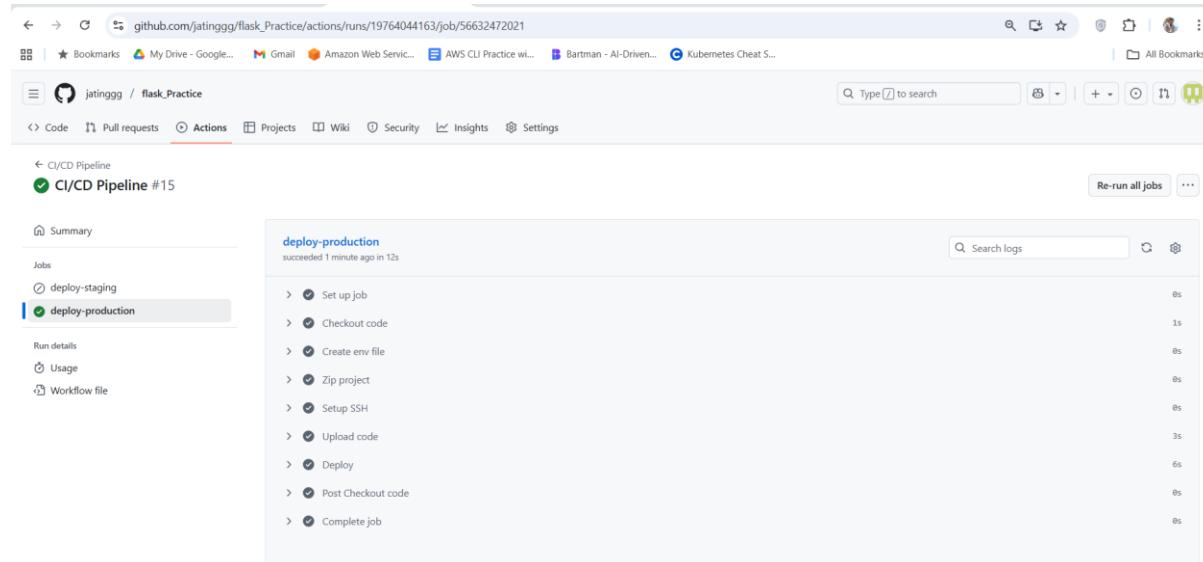
```
git checkout main
```

```
git commit -m "push to prod"
```

```
git merge origin/staging
```

```
git tag v1.0.1
```

```
git push origin v1.0.1
```



17. You can now access the application on the Prod server at port 8000.

Name	Email	Course	Actions
Jatin	jatin.js.shrivastava@gmail.com	HeroVired	<button>Edit</button> <button>Delete</button>
Jatin-Prod	jatin.js.shrivastava-Prod@gmail.com	HeroVired-prod	<button>Edit</button> <button>Delete</button>

_id	name	email	course
ObjectId('6928a30ddfd03bc3bef4832')	Jatin	jatin.js.shrivastava@gmail.com	HeroVired
ObjectId('6929985ac6798dde60eb1f75')	Jatin-Prod	jatin.js.shrivastava-Prod@gmail.com	HeroVired-prod