



Automated Deployment of Flask Applications Using GitLab CI/CD and Docker

Project Description:

This project is a simple **Python Flask web application** that is developed, tested, containerized, and deployed using a **GitLab CI/CD pipeline**. The pipeline automates three main stages:

1. **Testing** – The application code is validated using Python's testing framework to ensure functionality before deployment.
2. **Building** – A Docker image of the application is created and pushed to a container registry for easy deployment.
3. **Deployment** – The image is deployed to a **DigitalOcean Ubuntu server** using SSH and Docker, ensuring that the latest version is running in a clean container environment.

The deployment environment is fully containerized, making it portable and consistent across development and production. The use of **DigitalOcean Droplets** provides a reliable cloud infrastructure, while Docker ensures quick updates and minimal downtime during deployments.



1. Create Secret Variables in GitLab repo

Flags

- ☒ Protect variable
Export variable to pipelines running on protected branches and tags only.
- ☒ Expand variable reference
\$ will be treated as the start of a reference to another variable.

Description (optional)

The description of the variable's value or usage.

Key

You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. [What is the order of precedence for variables?](#)

Value

Add variable

Cancel

Flags

- ☒ Protect variable
Export variable to pipelines running on protected branches and tags only.
- ☒ Expand variable reference
\$ will be treated as the start of a reference to another variable.

Description (optional)

The description of the variable's value or usage.

Key

You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. [What is the order of precedence for variables?](#)

Value

Add variable

Cancel

- First, I tried to build and push the docker image to docker hub container registry using Git Lab CI/CD pipeline



2. Pipeline script

```
variables:
  IMAGE_NAME: thisarakandage/demo-app
  IMAGE_TAG: python-app-1.0

stages:
  - test
  - build
  - deploy

run_tests:
  stage: test
  image: python:3.9-slim-buster
  before_script:
    - apt-get update && apt-get install make
  script:
    - make test

build_image:
  stage: build
  image: docker:20.10.16
  services:
    - docker:20.10.16-dind
  variables:
    DOCKER_TLS_CERTDIR: "/certs"
  before_script:
    - docker login -u $REGISTRY_USER -p $REGISTRY_PASS
  script:
    - docker build -t $IMAGE_NAME:$IMAGE_TAG .
    - docker push $IMAGE_NAME:$IMAGE_TAG
```

3. After Completing the pipeline

All 7 Finished Branches Tags					View analytics Clear runner caches New pipeline	
Filter pipelines					<input type="text" value="Q"/>	Show Pipeline ID ▾
Status	Pipeline	Created by	Stages	Actions		
<div>✓ Passed</div> <div>⌚ 00:01:26</div> <div>📅 7 minutes ago</div>	<div>Edit requirements.txt</div> <div>#1976321715 main ac8ab234 </div> <div>latest branch fork</div>		<div>✓ ✓</div>	<div>⬇ ▾</div>		
<div>✗ Failed</div> <div>⌚ 00:00:30</div> <div>📅 12 minutes ago</div>	<div>Update3 .gitlab-ci.yml file</div> <div>#1976320286 main 8fb6bcf5 </div> <div>branch fork</div>		<div>✗ →</div>	<div>🔄 ⬇ ▾</div>		



4. Pipeline Logs



```
218 d1f6f46120b5: Pushed
219 14cbeede8d6e: Mounted from library/python
220 067ea27560c1: Mounted from library/python
221 7fb1037e08b3: Mounted from library/python
222 ae2d55769c5e: Mounted from library/python
223 e2ef8a51359d: Mounted from library/python
224 adac67a4e782: Pushed
225 python-app-1.0: digest: sha256:5f90aba3e5279b6e1037ad12554b865b16f6b7c9be0d276aff6e4c5cf324f16a size:
    2412
226 Cleaning up project directory and file based variables
227 Job succeeded
```

5. Create a droplet in Digital Ocean (Ubuntu server) for deploying the web application

Droplets Create an Autoscale Pool Create Droplet

Droplets Autoscale Pools

Search by Droplet name

Name	IP Address	Created	Tags
 thisarahost 1 GB / 25 GB Disk / NYC1 - Ubuntu 25.04 x64	174.138.37.75	Let's get to work!	 Upsize More



6. Connect to Server using SSH

```
thisarak943@cloudshell:~/.ssh$ ls
id_ed25519 id_ed25519.pub
thisarak943@cloudshell:~/.ssh$ ssh root@174.138.37.75 -i id_ed25519
The authenticity of host '174.138.37.75 (174.138.37.75)' can't be established.
ED25519 key fingerprint is SHA256:94z7txM2pkTkV4CvAIGF9XxCNmtPAHG6G8xM1sFn3Xo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '174.138.37.75' (ED25519) to the list of known hosts.
Welcome to Ubuntu 25.04 (GNU/Linux 6.14.0-23-generic x86_64)

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

System information as of Sun Aug 10 11:01:19 UTC 2025

System load:  0.14               Processes:    105
Usage of /:   8.5% of 23.10GB    Users logged in: 1
Memory usage: 21%              IPv4 address for eth0: 174.138.37.75
Swap usage:  0%                 IPv4 address for eth0: 10.10.0.5

65 updates can be applied immediately.
40 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@thisarahost:~#
```

7. Docker installed on server

```
For more help on how to use Docker, head to https://docs.docker.com/go/guides/

root@thisarahost:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@thisarahost:~#
```



8. Variables create for Private key

Flags

- ☒ **Protect variable**
Export variable to pipelines running on protected branches and tags only.
- ☒ **Expand variable reference**
\$ will be treated as the start of a reference to another variable.

Description (optional)

The description of the variable's value or usage.

Key

You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. What is the order of precedence for variables?

Value

```
X4hMF0cK6hRmQ3ezBkEF9fd8iz4vj+tyx
r+uNX
TQzQG
x0AQI
-----END OPENSsh PRIVATE KEY-----
```

Add variable

Cancel

9. Then tried to deploy the app on Digital Ocean ubuntu server

```
31 deploy:
32   stage: deploy
33   before_script:
34     - chmod 400 $SSH_KEY
35   script:
36     - ssh -o StrictHostKeyChecking=no -i $SSH_KEY root@174.138.37.75 "
37       docker login -u $REGISTRY_USER -p $REGISTRY_PASS &&
38       docker ps -aq | xargs -r docker stop &&
39       docker ps -aq | xargs -r docker rm &&
40       docker run -d -p 5000:5000 $IMAGE_NAME:$IMAGE_TAG"
41
42
43
```

10. Pipeline view

All10

Finished

Branches

Tags

View analytics

Clear runner caches

New pipeline

Filter pipelines

Q

Show Pipeline ID

Status	Pipeline	Created by	Stages	Actions
<div>Passed</div> <div>00:01:54</div> <div>1 minute ago</div>	<div>Update7 .gitlab-ci.yml file</div> <div>#1976417031main6258d752</div> <div>latestbranchfork</div>	<div></div>	<div></div> <div></div> <div></div>	<div>Download</div>



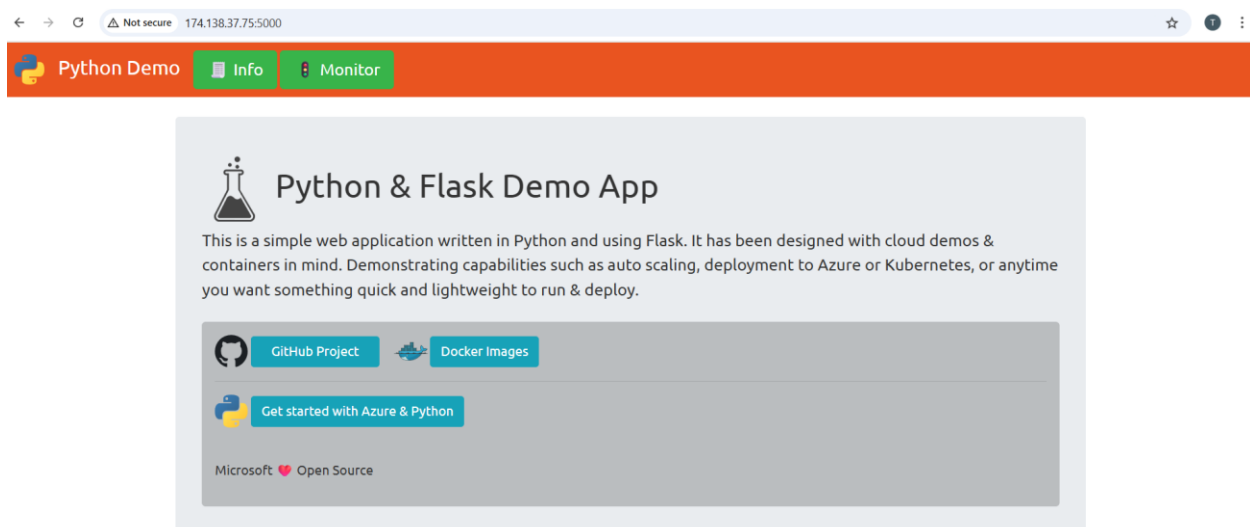
11. Pipeline logs

```
80 e24331bfb375: Pull complete
81 2248f0f9cfb3: Pull complete
82 Digest: sha256:2c92b5d03e607933f3eb4378ab586fd53a171a78cb08c95b95a320ac89823fe6
83 Status: Downloaded newer image for [MASKED]/demo-app:python-app-1.0
84 580077d08d6079ea132c3f7c95a4a8072b3e68f7dec9f316cfbee87b7e4fba20
✓ 85 Cleaning up project directory and file based variables
86 Job succeeded
```

12. Docker Container is running in server

```
root@thisarahost:~# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
580077d08d60   thisarakandage/demo-app:python-app-1.0  "gunicorn -b 0.0.0.0..."  4 minutes ago  Up 4 minutes  0.0.0.0:5000->5000/tcp, :::5000->5000/tcp  sharp_taussig
root@thisarahost:~#
```

13. Final checks in browser





Summary:

Through this project, I learned how to take a simple Flask application from local development to production using modern DevOps practices. I gained hands-on experience with:

- **Flask Application Basics** – Building and structuring a Python web application.
- **GitLab CI/CD Pipelines** – Automating testing, building, and deployment workflows.
- **Docker** – Packaging the application into portable containers for consistent runtime environments.
- **Container Registry** – Storing and retrieving application images securely for deployment.
- **DigitalOcean Droplet Management** – Setting up an Ubuntu server for hosting containerized applications.
- **Remote Deployment via SSH** – Automating remote server commands to stop old containers and start new ones.
- **Error Handling in Pipelines** – Debugging and fixing common CI/CD issues such as empty container lists and deployment failures.

This project improved my understanding of **automation, containerization, and cloud deployment**. It also strengthened my problem-solving skills in real-world deployment scenarios, making me more confident in delivering production-ready applications.