

1. Multi-Container Flask Application with PostgreSQL Using Docker Compose

Step 1: Download Docker .

Sudo apt install docker-compose-plugin

Step 2: Clone the git repository and move to directory cd Flask-Docker

```
master@master-vm: ~/Flask-Docker
master@master-vm:~$ docker-compose version
Command 'docker-compose' not found, but can be installed with:
sudo snap install docker          # version 27.5.1, or
sudo snap install docker          # version 27.2.0
sudo apt install docker-compose  # version 1.29.2-1
See 'snap info <snapname>' for additional versions.
master@master-vm:~$ sudo apt install docker-compose-plugin
[sudo] password for master:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker-compose-plugin is already the newest version (2.33.1-1~ubuntu.22.04~jammy).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
master@master-vm:~$ git clone https://github.com/KPkm25/Flask-Docker
Cloning into 'Flask-Docker'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 15 (delta 3), reused 13 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (15/15), 663.98 KiB | 2.63 MiB/s, done.
Resolving deltas: 100% (3/3), done.
master@master-vm:~$ cd Flask-Docker
```

Step 3: Before build the containers login to docker hub using command “docker login”. Build and start the containers using docker-compose up -d --build.

```
master@master-vm: ~/Flask-Docker
Waiting for authentication in the browser...

WARNING! Your credentials are stored unencrypted in '/home/master/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

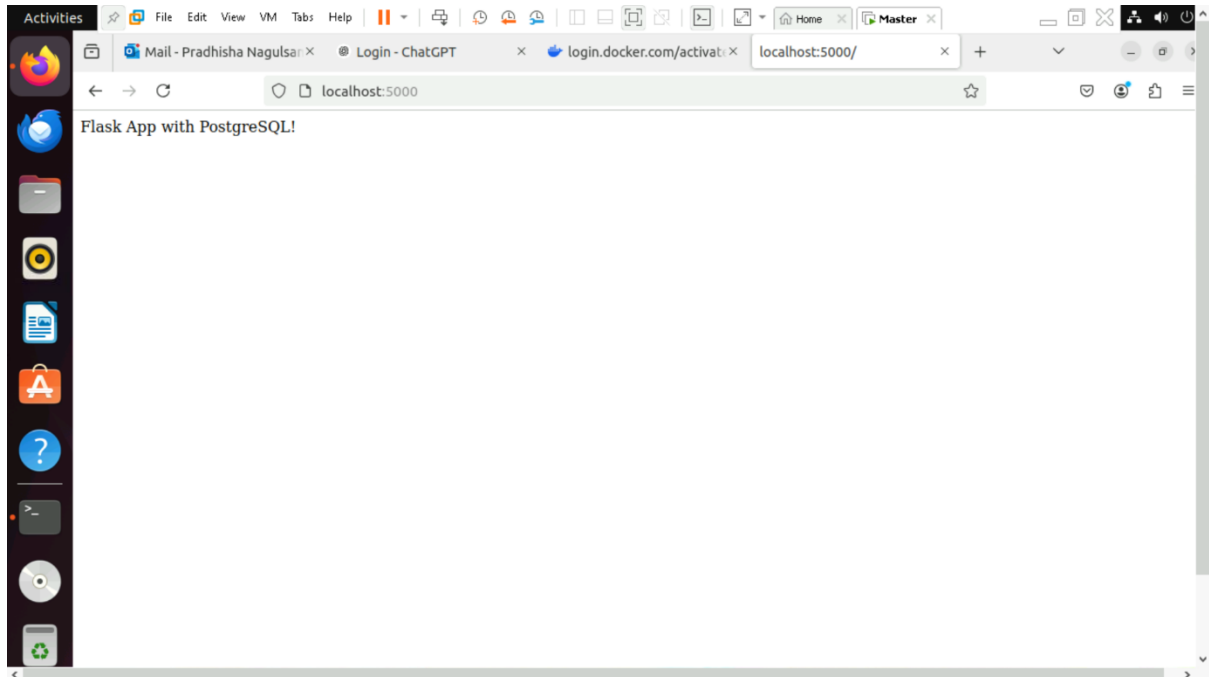
Login Succeeded
master@master-vm:~/Flask-Docker$ docker-compose up -d --build
Pulling db (postgres:...)
latest: Pulling from library/postgres
7cf63256a31a: Already exists
543c6dea2e39: Pull complete
dc87fb4dbc03: Pull complete
55c54708c8e7: Pull complete
878a40f56a67: Pull complete
6424ae1ae883: Pull complete
600e770d797e: Pull complete
a21a08dbca2c: Pull complete
783086ffbe8e: Pull complete
42e76ffa3e07: Pull complete
fcccafd45a4d: Pull complete
420a047e4570: Pull complete
553d1749e29f: Pull complete
bc13f9b1d80d: Pull complete
Digest: sha256:81f32a88ec561664634637dd446487efd5f9d90996304b96210078e90e5c8b21
Status: Downloaded newer image for postgres:latest
Building web
[+] Building 131.1s (11/11) FINISHED docker:default
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 189B 0.0s
=> [internal] load metadata for docker.io/l 4.5s
=> [auth] library/python:pull token for reg 0.0s
=> [internal] load dockerignore 0.0s
```

Step 4: Verify the running containers using “docker ps”. It used to verify whether web and db is there or not.

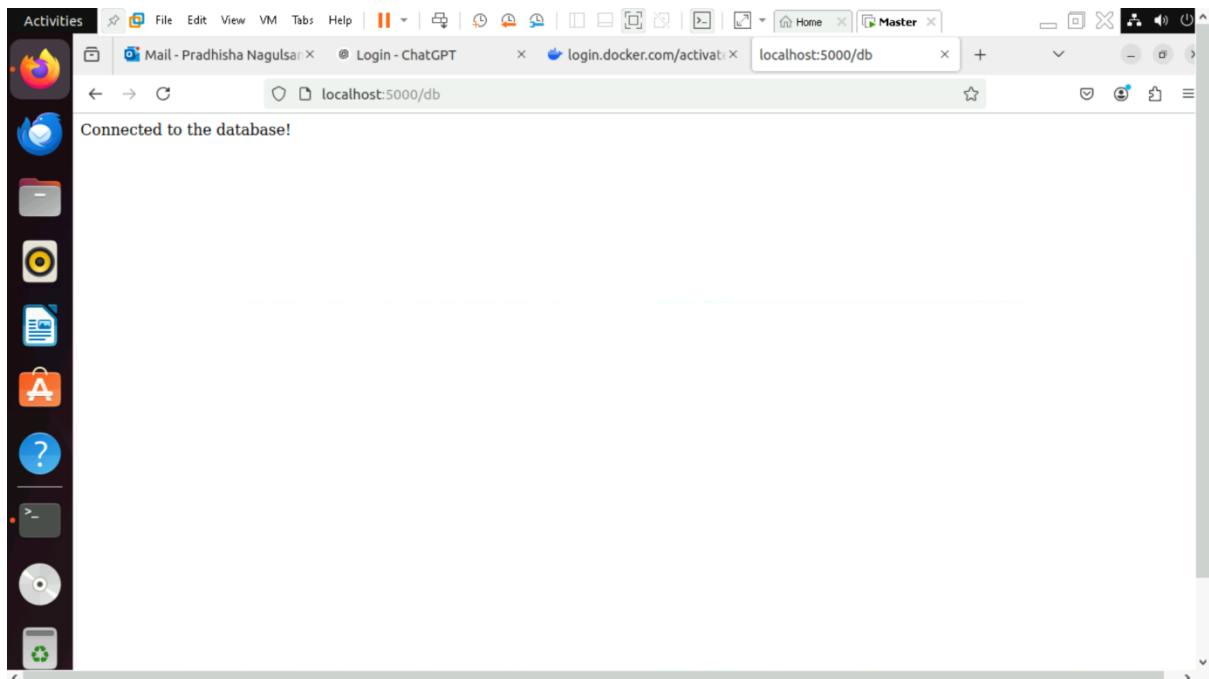
```
master@master-vm:~/Flask-Docker$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
24633d30f8c4   flask-docker_web_1   "python app.py"         35 seconds ago   Up 34 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
c6e2be65acf5   postgres        "docker-entrypoint.s..." 37 seconds ago   Up 36 seconds   0.0.0.0:5433->5432/tcp, [::]:5433->5432/tcp
4368d375dd3a   portainer/portainer-ce   "/portainer"           23 hours ago     Up 36 minutes   8000/tcp, 9443/tcp, 0.0.0.0:9000->9000/tcp, [::]:9000->9000/tcp
master@master-vm:~/Flask-Docker$
```

Step 5: Test the application.

<http://localhost:5000/> → Should return "Flask App with PostgreSQL!"



<http://localhost:5000/db> → Should confirm database connection.



2. Jenkins + Docker Pipeline Project Documentation

Step 1: Install Docker on Jenkins Server

1. **Update system packages and install Docker:**

```
sudo apt update
```

```
sudo apt install docker.io -y
```

2. **Start and enable Docker:**

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

3. **Add Jenkins user to Docker group (to allow Jenkins to run Docker commands):**

```
sudo usermod -aG docker jenkins
```

4. **Restart Jenkins to apply changes:**

```
sudo systemctl restart jenkins
```

5. **Verify Docker installation:**

```
docker --version
```

Step 2: Enable Password Authentication (If Needed)

If SSH key authentication is not set up, enable password login:

1. **Connect to the remote server and edit the SSH configuration file:**

```
sudo nano /etc/ssh/sshd_config
```

2. **Modify these lines:**

```
PasswordAuthentication yes
```

```
PermitRootLogin yes
```

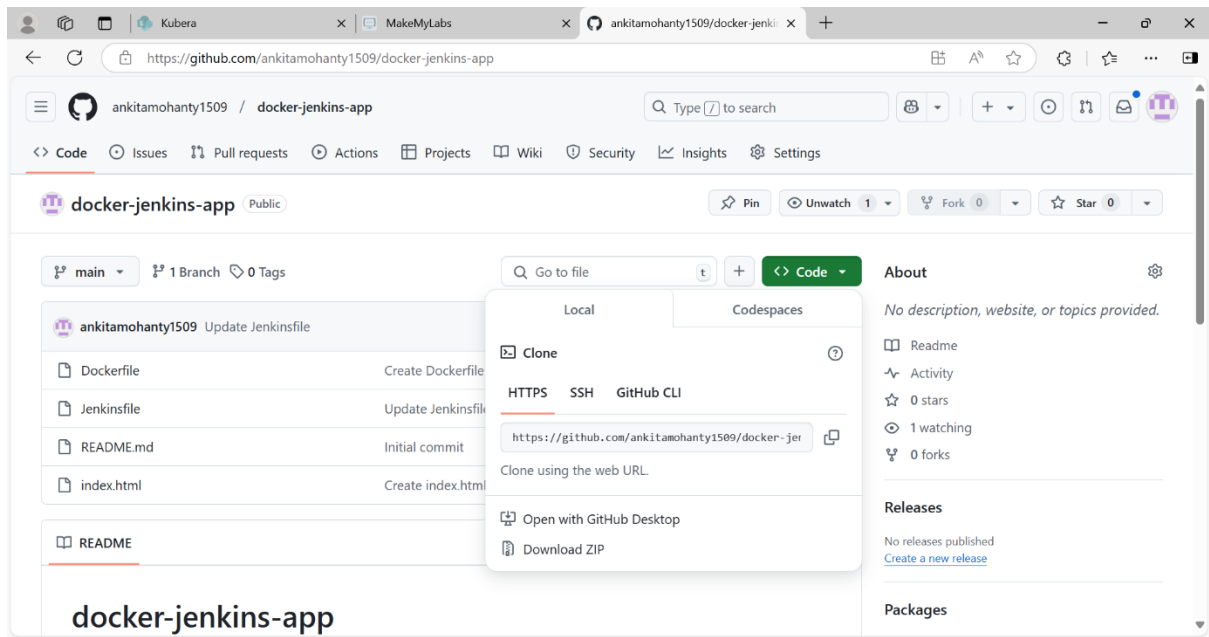
3. **Save the file and restart SSH:**

```
sudo systemctl restart ssh
```

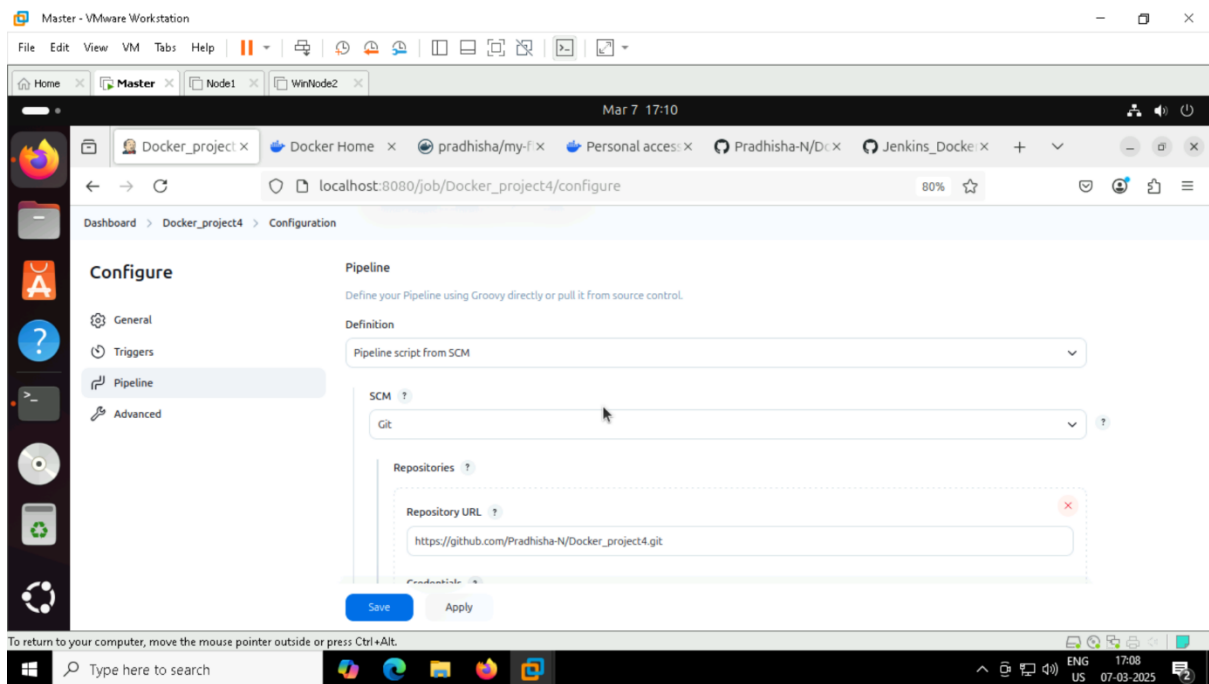
4. **Test SSH login:**

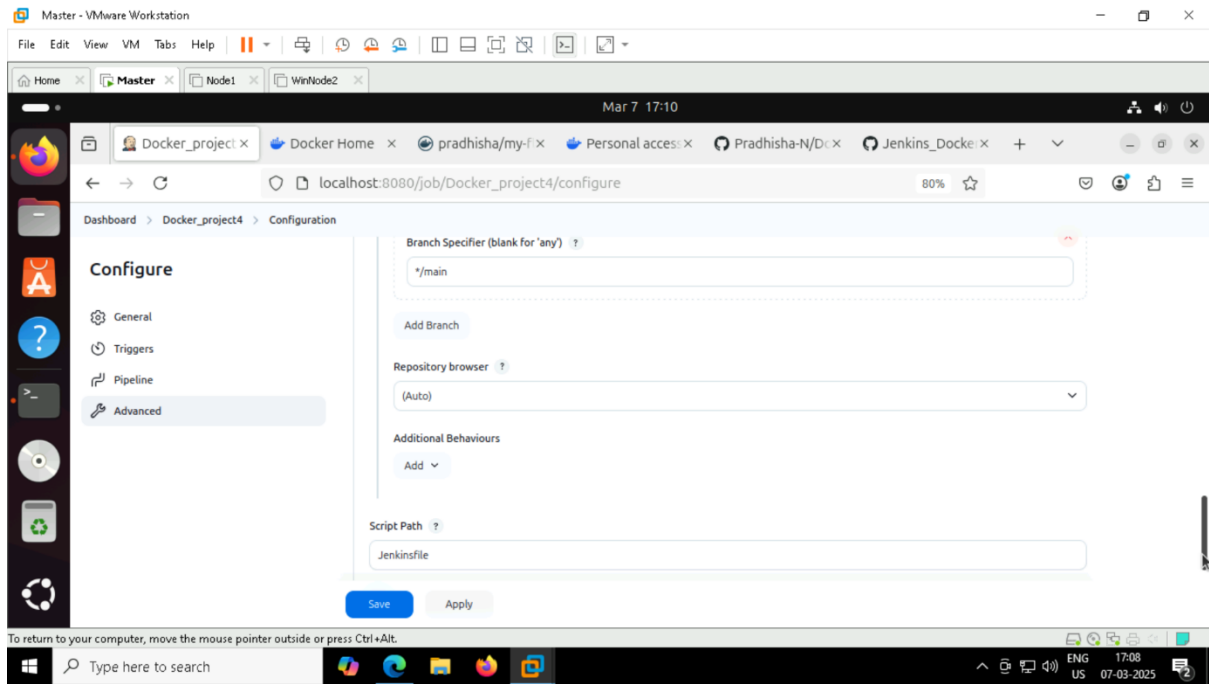
```
ssh master@192.168.203.128
```

Step 3: Create a Simple Web Application

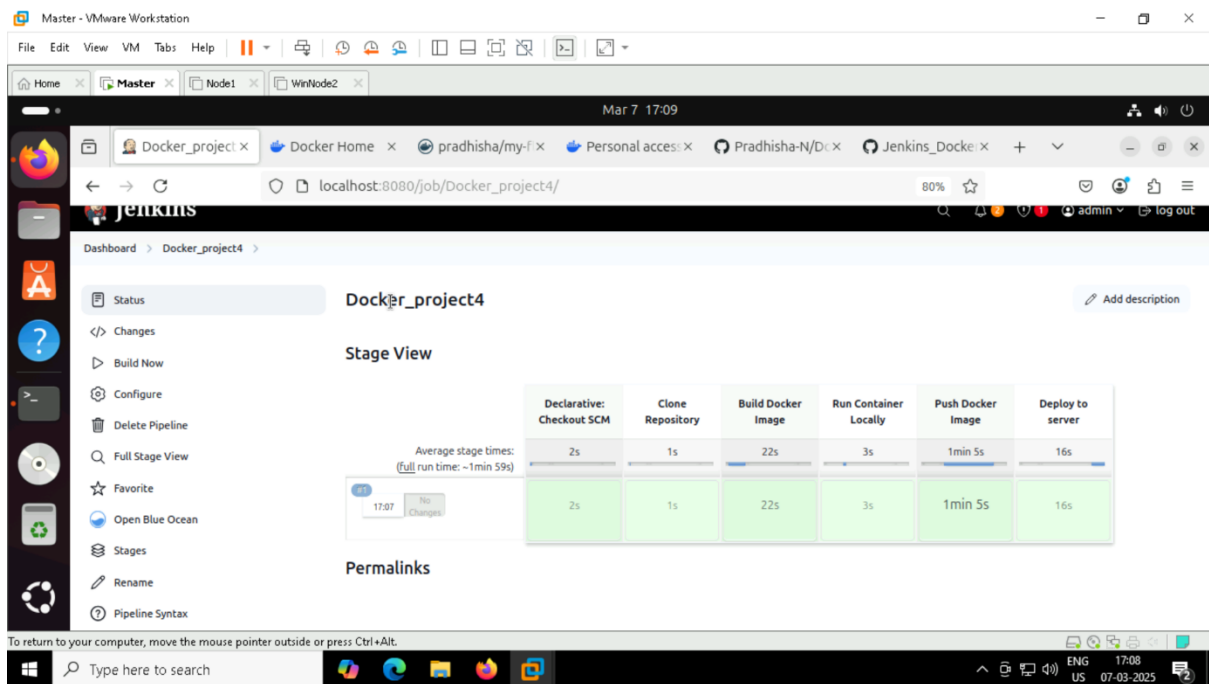


Step 4: Create a Jenkins Pipeline





Step 5: Run the Jenkins Pipeline



Deployment Successful with Jenkins and Docker!

3. Kubectl setup

1. Install Kubernetes on Ubuntu

a. Install Dependencies

```
master@mastervm: ~  
$ sudo apt update  
[sudo] password for master:  
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease  
Hit:2 https://pkg.jenkins.io/debian-stable binary/ Release  
Hit:3 http://in.archive.ubuntu.com/ubuntu noble InRelease  
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]  
Get:5 http://in.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]  
Get:7 http://in.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]  
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [668 kB]  
Get:9 http://in.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [916 kB]  
Get:10 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [128 kB]  
Get:11 http://in.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]  
Get:12 http://in.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [754 kB]  
Get:13 http://in.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [151 kB]  
Get:14 http://in.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]  
Get:15 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1,040 kB]  
Get:16 http://in.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [261 kB]  
Get:17 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [8,964 B]  
Get:18 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [364 kB]  
Get:19 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]  
Get:20 http://in.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]  
Get:21 http://in.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]  
Get:22 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [823 kB]  
Get:23 http://in.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [212 B]  
Get:24 http://in.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [20.0 kB]  
Get:25 http://in.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]  
Get:26 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [177 kB]  
Get:27 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.9 kB]  
100% [27 Components-amd64 store 0 B] [Waiting for headers] Get:28 http://security.ubuntu.com/ubuntu noble-
```



```
master@mastervm:~$ minikube start --driver=docker
minikube v1.35.0 on Ubuntu 24.04
Using the docker driver based on user configuration

Exiting due to PROVIDER_DOCKER_NEWGRP: "docker version --format <no value>:<no value>:<no value>" exit status 1: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.45/version": dial unix /var/run/docker.sock: connect: permission denied
Suggestion: Add your user to the 'docker' group: 'sudo usermod -aG docker $USER && newgrp docker'
Documentation: https://docs.docker.com/engine/install/linux-postinstall/
```

e. Check status

```
master@mastervm:~$ kubectl cluster-info
E0311 13:12:52.589497 6750 memcache.go:265] "Unhandled Error" err=<
couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3Ftimeout%3D32s' /><script id='redirect' data-redirect-url='/login?from=%2Fapi%3Ftimeout%3D32s' src='/static/4cd96940/scripts/redirect.js'></script></head><body style='background-color:white; color:white;'>
```

2. Create project

a. Create folder

```
master@mastervm:~/flask-ci-cd
master@mastervm:~$ mkdir flask-ci-cd
cd flask-ci-cd
```

b. Create app.py

```
master@mastervm:~/flask-ci-cd$ nano app.py
```

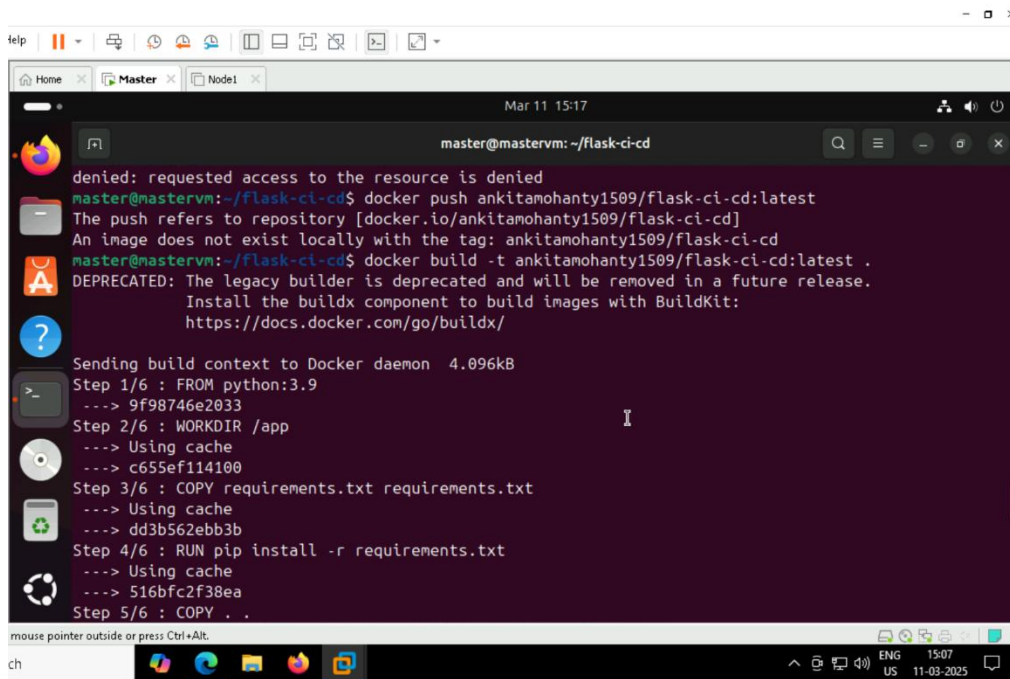
c. Create Dockerfile

```
master@mastervm:~/flask-ci-cd$ nano Dockerfile
```

d. Create requirements.txt

```
master@mastervm:~/flask-ci-cd$ nano requirements.txt
master@mastervm:~/flask-ci-cd$ docker build -t kpk25/flask-ci-cd:latest .
docker login
docker push kpk25/flask-ci-cd:latest
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/
Sending build context to Docker daemon 4.096kB
Step 1/6 : FROM python:3.9
3.9: Pulling from library/python
155ad54a8b28: Pulling fs layer
8031108f3cda: Pull complete
1d281e50d3e4: Pull complete
447713e77b4f: Pull complete
```

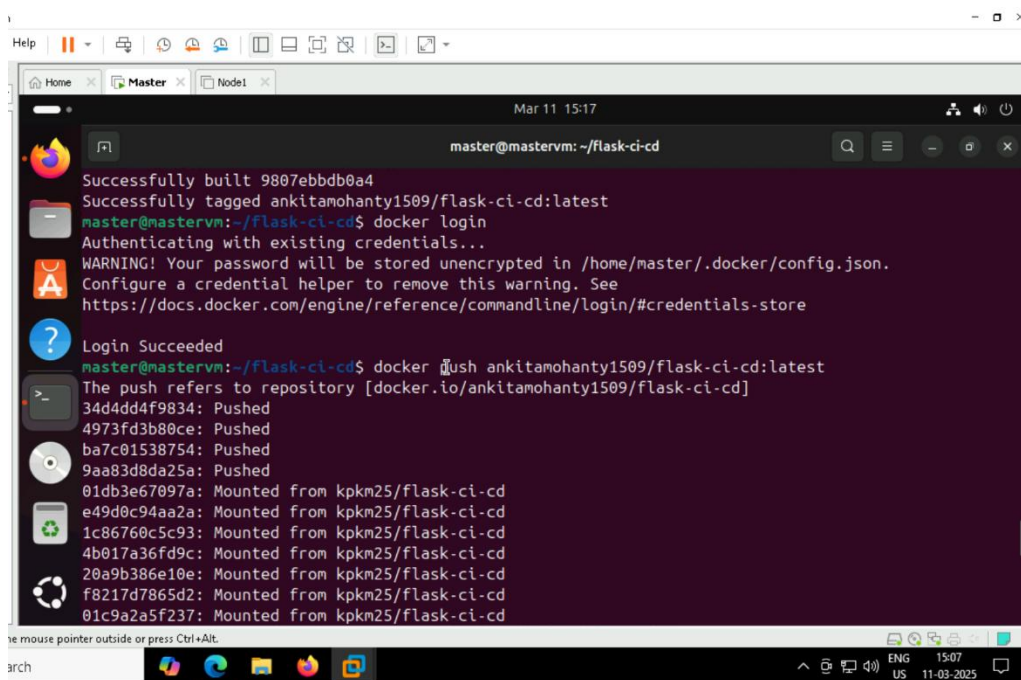
3. Build Docker Image

A terminal window titled 'master@mastervm: ~/flask-ci-cd' showing the process of building and pushing a Docker image. The user runs 'docker push ankitamohanty1509/flask-ci-cd:latest', which fails with a 'denied: requested access to the resource is denied' error. Then, the user runs 'docker build -t ankitamohanty1509/flask-ci-cd:latest .', which succeeds. The build steps are: Step 1/6: FROM python:3.9, Step 2/6: WORKDIR /app, Step 3/6: COPY requirements.txt requirements.txt, Step 4/6: RUN pip install -r requirements.txt, and Step 5/6: COPY . . The terminal also shows a warning about the deprecated legacy builder and a link to the buildx component.

```
denied: requested access to the resource is denied
master@mastervm:~/flask-ci-cd$ docker push ankitamohanty1509/flask-ci-cd:latest
The push refers to repository [docker.io/ankitamohanty1509/flask-ci-cd]
An image does not exist locally with the tag: ankitamohanty1509/flask-ci-cd
master@mastervm:~/flask-ci-cd$ docker build -t ankitamohanty1509/flask-ci-cd:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 4.096kB
Step 1/6 : FROM python:3.9
--> 9f98746e2033
Step 2/6 : WORKDIR /app
--> Using cache
--> c655ef114100
Step 3/6 : COPY requirements.txt requirements.txt
--> Using cache
--> dd3b562ebb3b
Step 4/6 : RUN pip install -r requirements.txt
--> Using cache
--> 516bfc2f38ea
Step 5/6 : COPY . .
```

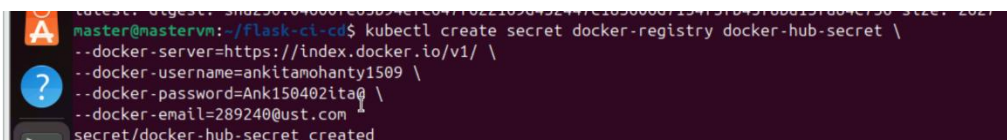
a. Push Docker Image

A terminal window titled 'master@mastervm: ~/flask-ci-cd' showing the process of logging into Docker and pushing the image. The user runs 'docker login', which succeeds. Then, the user runs 'docker push ankitamohanty1509/flask-ci-cd:latest', which succeeds. The push process shows the image being pushed to the repository and then mounted from the local image. The terminal also shows a warning about the deprecated legacy builder and a link to the buildx component.

```
Successfully built 9807ebdbb0a4
Successfully tagged ankitamohanty1509/flask-ci-cd:latest
master@mastervm:~/flask-ci-cd$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/master/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
master@mastervm:~/flask-ci-cd$ docker push ankitamohanty1509/flask-ci-cd:latest
The push refers to repository [docker.io/ankitamohanty1509/flask-ci-cd]
34d4dd4f9834: Pushed
4973fd3b80ce: Pushed
ba7c01538754: Pushed
9aa83d8da25a: Pushed
01db3e67097a: Mounted from kpk25/flask-ci-cd
e49d0c94aa2a: Mounted from kpk25/flask-ci-cd
1c86760c5c93: Mounted from kpk25/flask-ci-cd
4b017a36fd9c: Mounted from kpk25/flask-ci-cd
20a9b386e10e: Mounted from kpk25/flask-ci-cd
f8217d7865d2: Mounted from kpk25/flask-ci-cd
01c9a2a5f237: Mounted from kpk25/flask-ci-cd
```

4. Connect Kubernetes to Docker

A terminal window titled 'master@mastervm: ~/flask-ci-cd' showing the process of creating a Kubernetes secret for Docker. The user runs 'kubectl create secret docker-registry docker-hub-secret', which succeeds. The secret is created with the following values: --docker-server=https://index.docker.io/v1/, --docker-username=ankitamohanty1509, --docker-password=Ank150402ita@, and --docker-email=289240@ust.com.

```
master@mastervm:~/flask-ci-cd$ kubectl create secret docker-registry docker-hub-secret \
--docker-server=https://index.docker.io/v1/ \
--docker-username=ankitamohanty1509 \
--docker-password=Ank150402ita@ \
--docker-email=289240@ust.com
secret/docker-hub-secret created
```

5. Kubernetes Deployment

a. Create k8s-deployment.yaml

```
master@mastervm:~/flask-ci-cd$ nano k8s-deployment.yaml
master@mastervm:~/flask-ci-cd$ kubectl apply -f k8s-deployment.yaml
error when retrieving current configuration of:
Resource: "apps/v1, Resource=deployments", GroupVersionKind: "apps/v1, Kind=Deployment"
Name: "", Namespace: "default"
from server for: "k8s-deployment.yaml": resource name may not be empty
error when retrieving current configuration of:
Resource: "/v1, Resource=services", GroupVersionKind: "/v1, Kind=Service"
Name: "", Namespace: "default"
```

6. Apply the deployment

```
master@mastervm:~/flask-ci-cd$ kubectl apply -f k8s-deployment.yaml
deployment.apps/flask-app created
service/flask-service created
```

7. Check if the pods are running

```
master@mastervm:~/flask-ci-cd$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
flask-app-58b8cc8758-kj5v5	1/1	Running	0	3m48s
flask-app-58b8cc8758-wvl9m	1/1	Running	0	3m48s