

# 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/

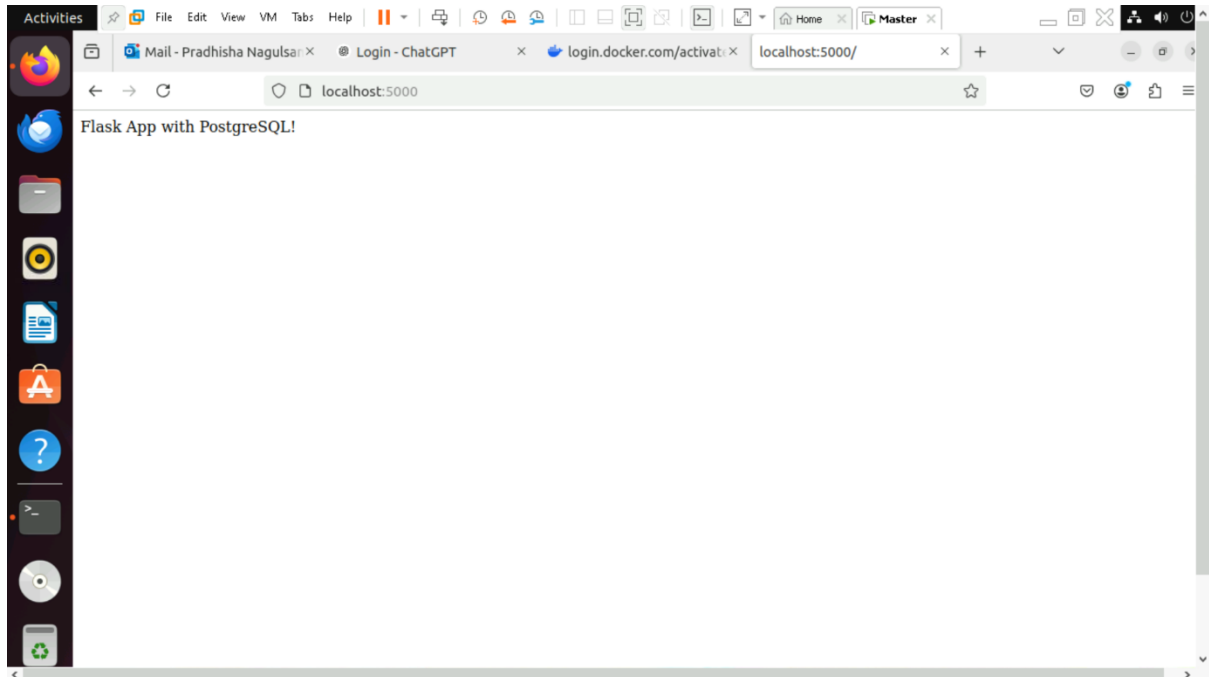
Login Succeeded
master@master-vm:~/Flask-Docker$ docker-compose up -d --build
Pulling db (postgres)...
latest: Pulling from library/postgres
7cf63256a31a: Already exists
543c6dea2e39: Pull complete
dc87fb4dbc03: Pull complete
55c54708c8e7: Pull complete
878a40f56a67: Pull complete
6424ae1ae883: Pull 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.

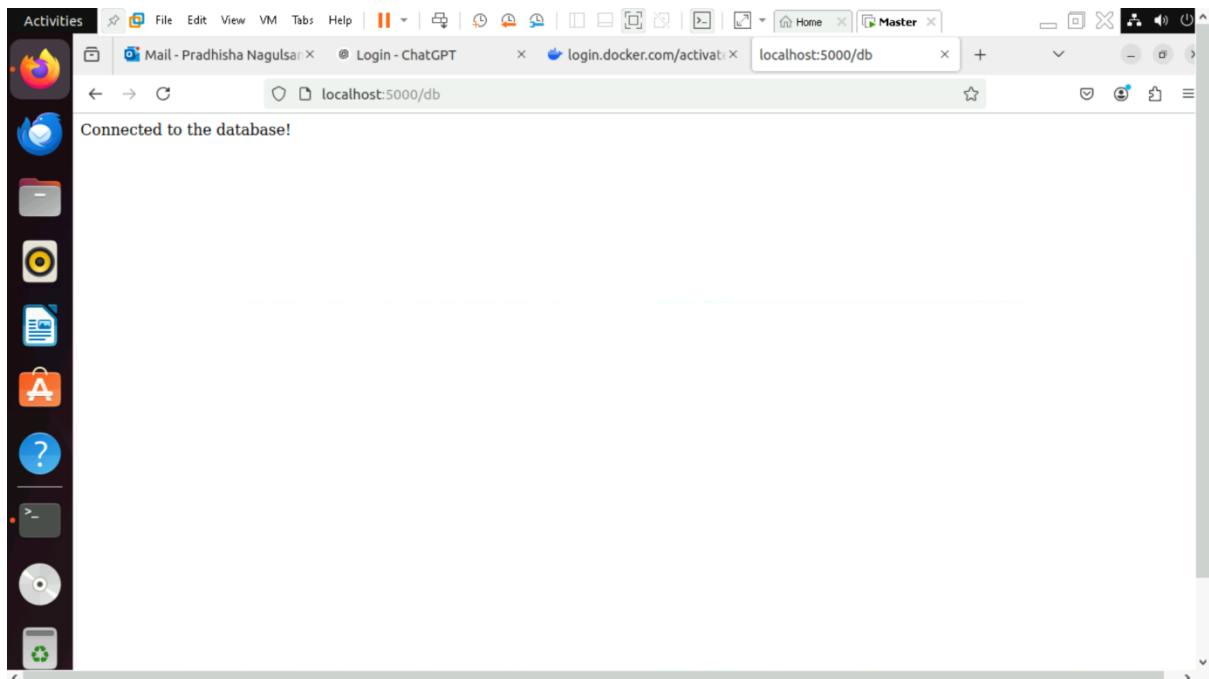
```
master@master-vm:~/Flask-Docker$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
24633d30f8c4   flask-docker_web_1   "python app.py"         35 seconds ago   Up 34 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
c6e2be65acf5   postgres        "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
```

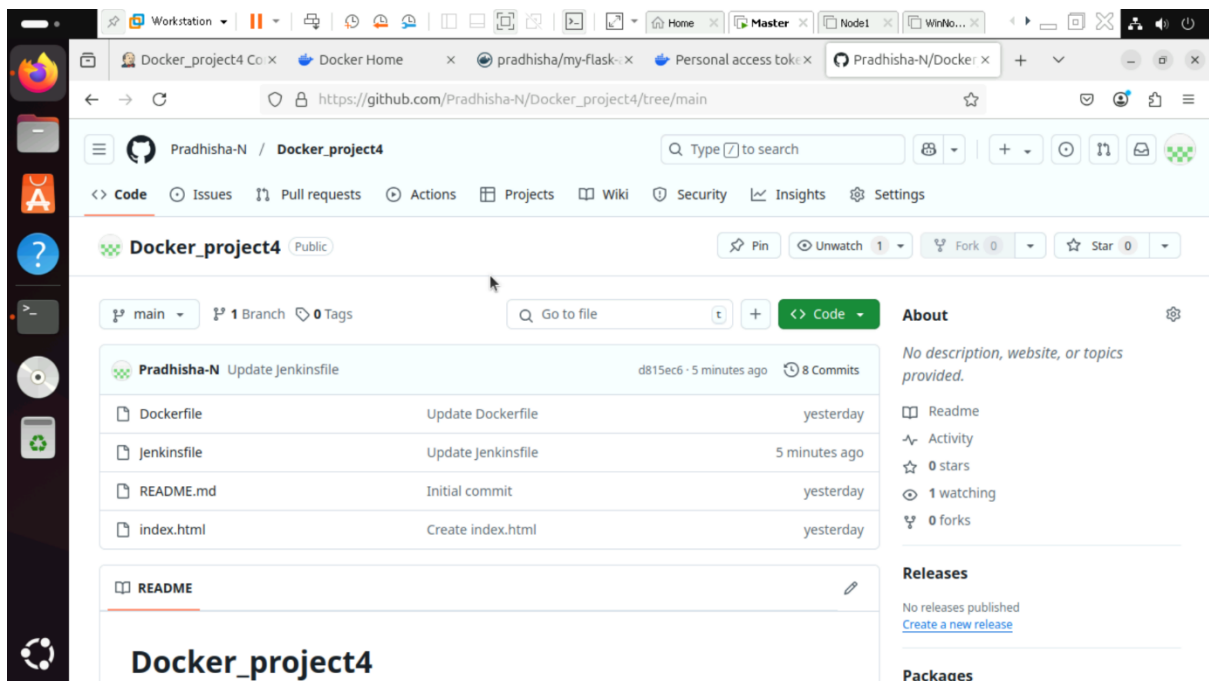
3. **Save the file and restart SSH:**

```
sudo systemctl restart ssh
```

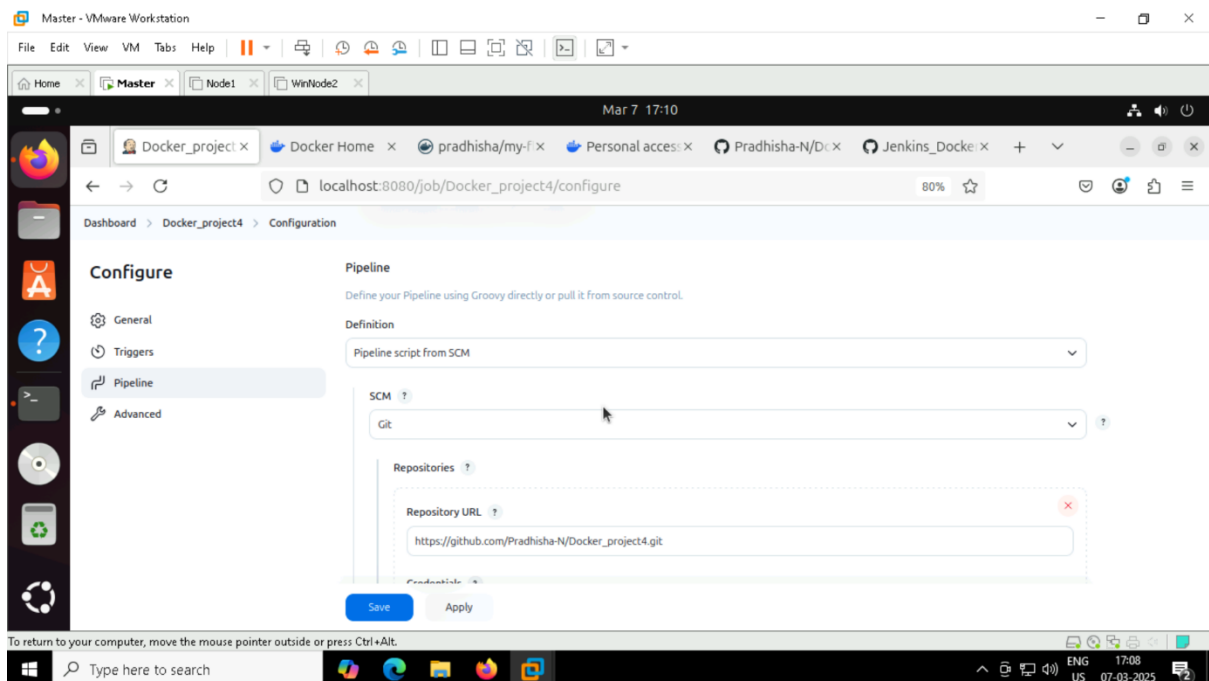
4. **Test SSH login:**

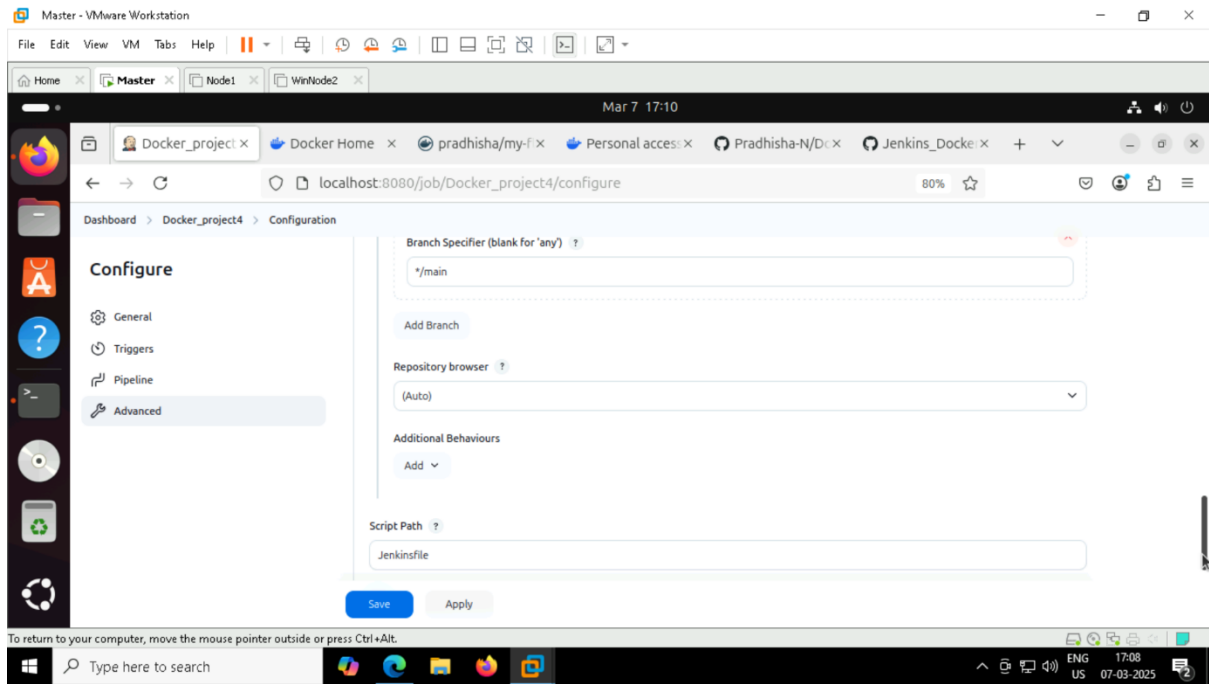
```
ssh master@192.168.203.128
```

### Step 3: Create a Simple Web Application

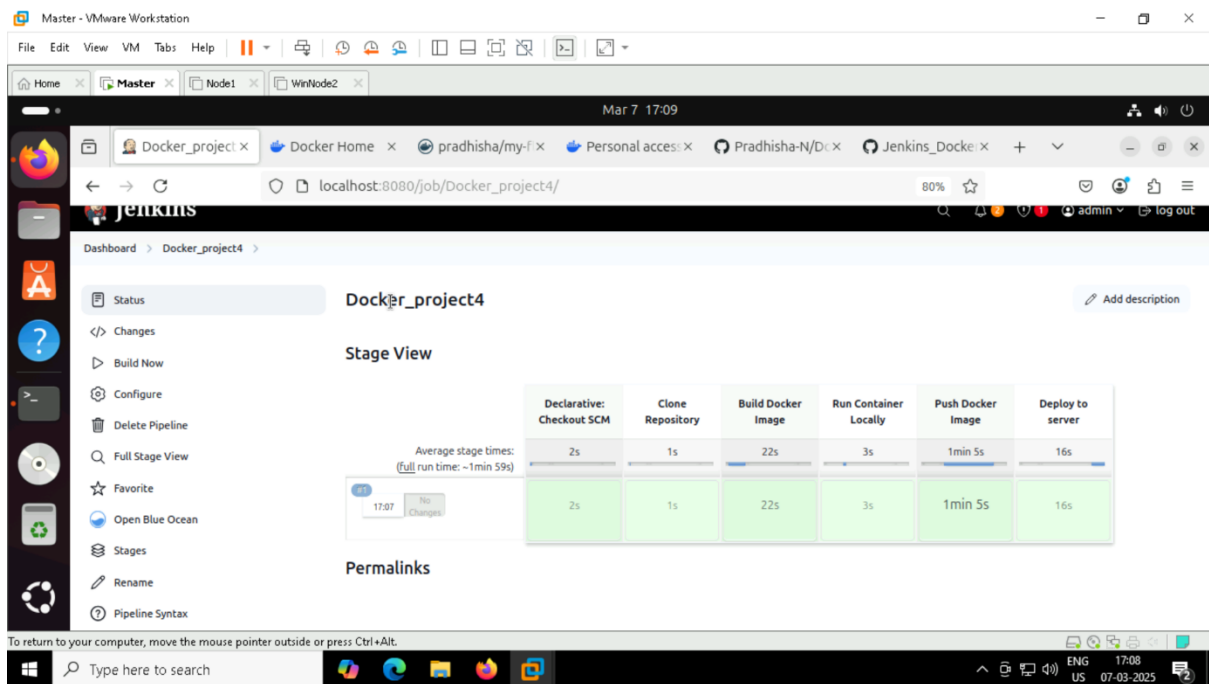


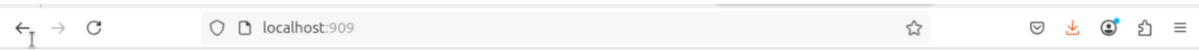
### Step 4: Create a Jenkins Pipeline





## Step 5: Run the Jenkins Pipeline





**Deployment Successful with Jenkins and Docker!**