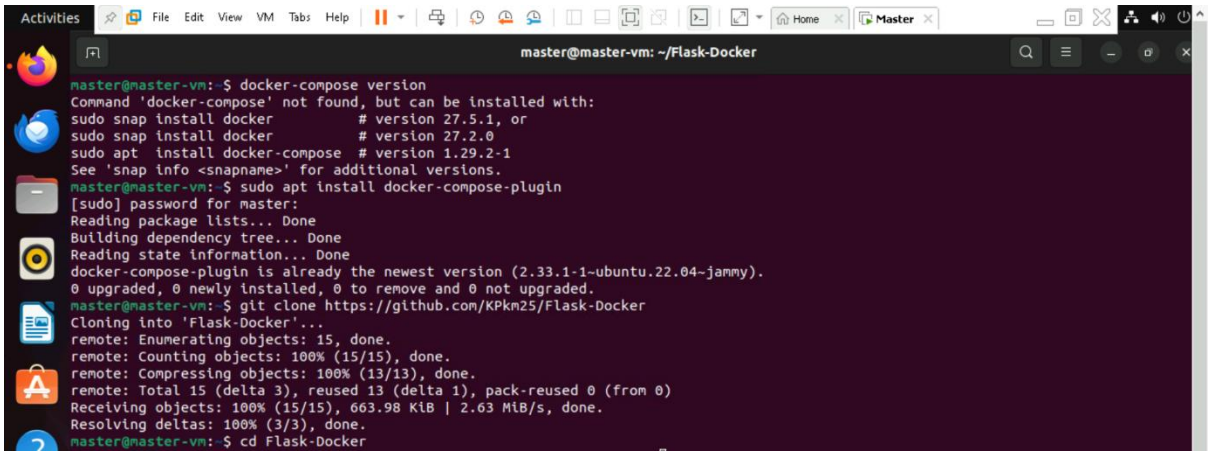
1. **Multi-Container Flask Application with PostgreSQL Using Docker Compose**

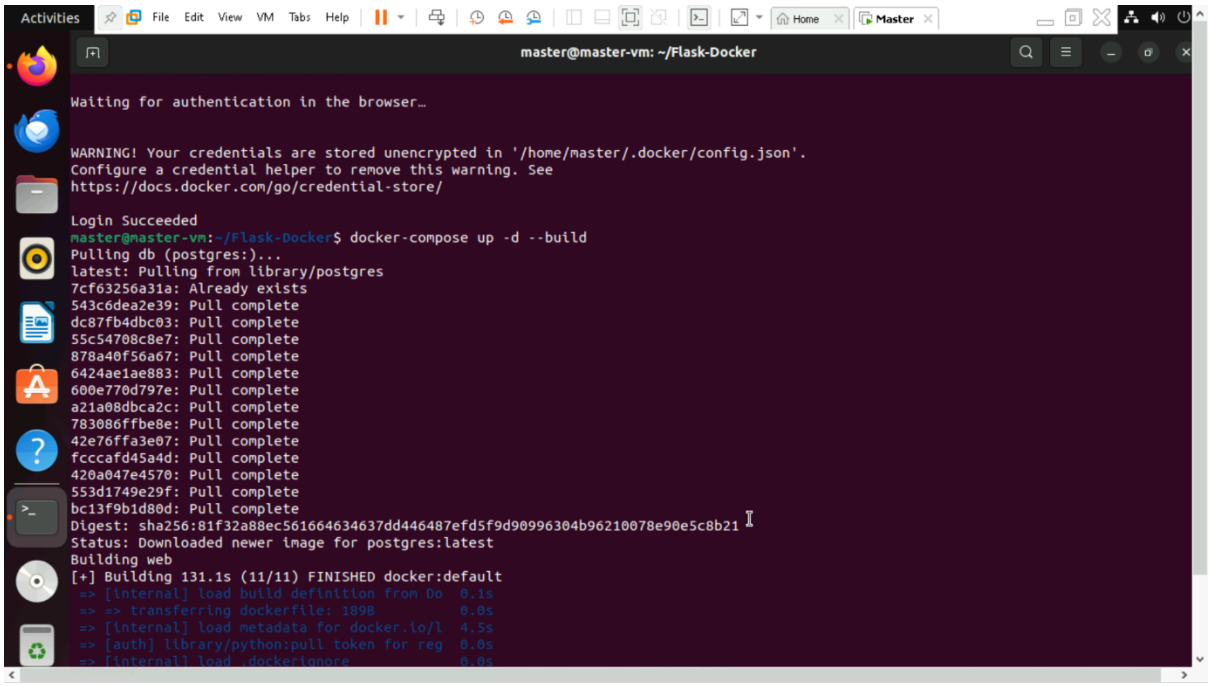
Step 1: Download Docker .

Sudo apt install docker-compose-plugin

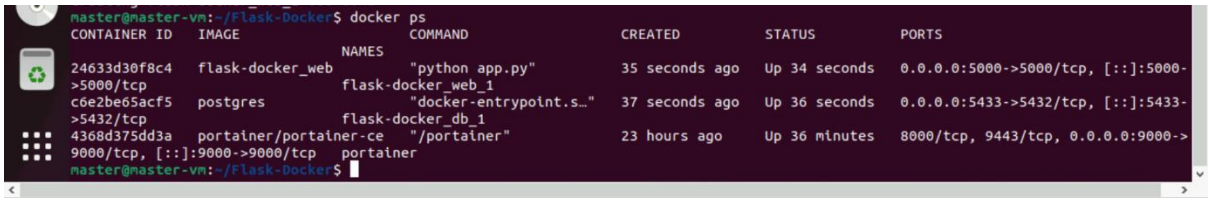
Step 2: Clone the git repository and move to directory cd Flask-Docker



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.*

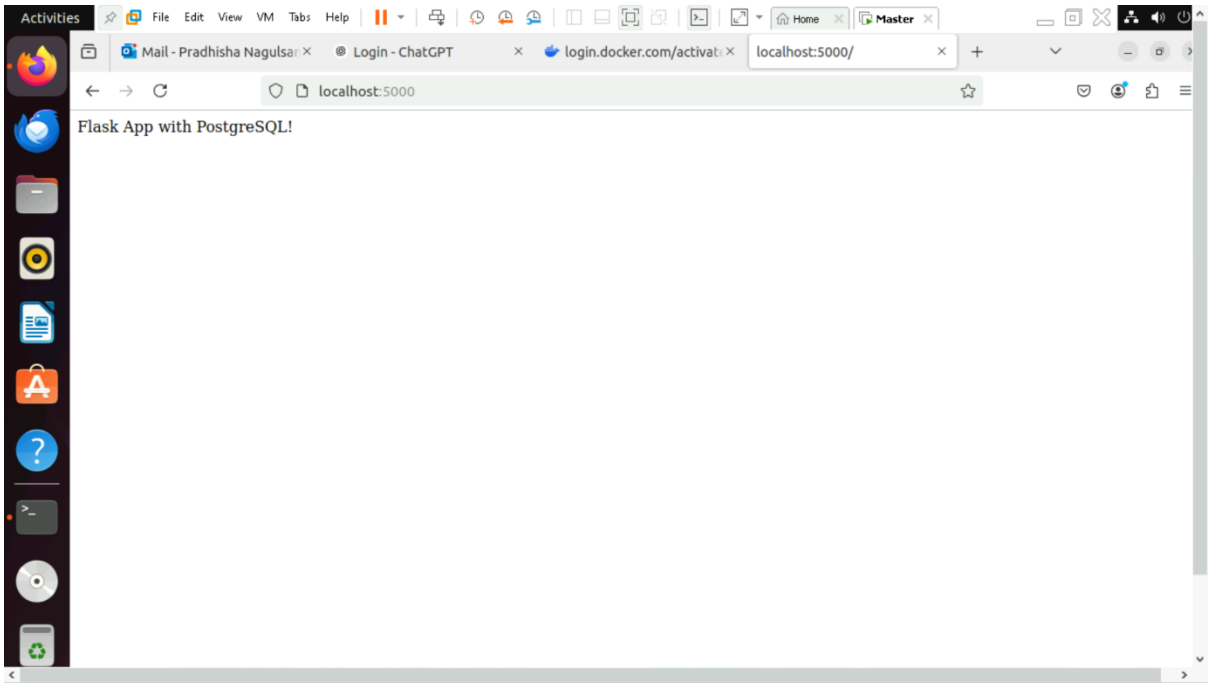


Step 4: Verify the running containers using *"docker ps"*. It used to verify whether web and db is there or not.

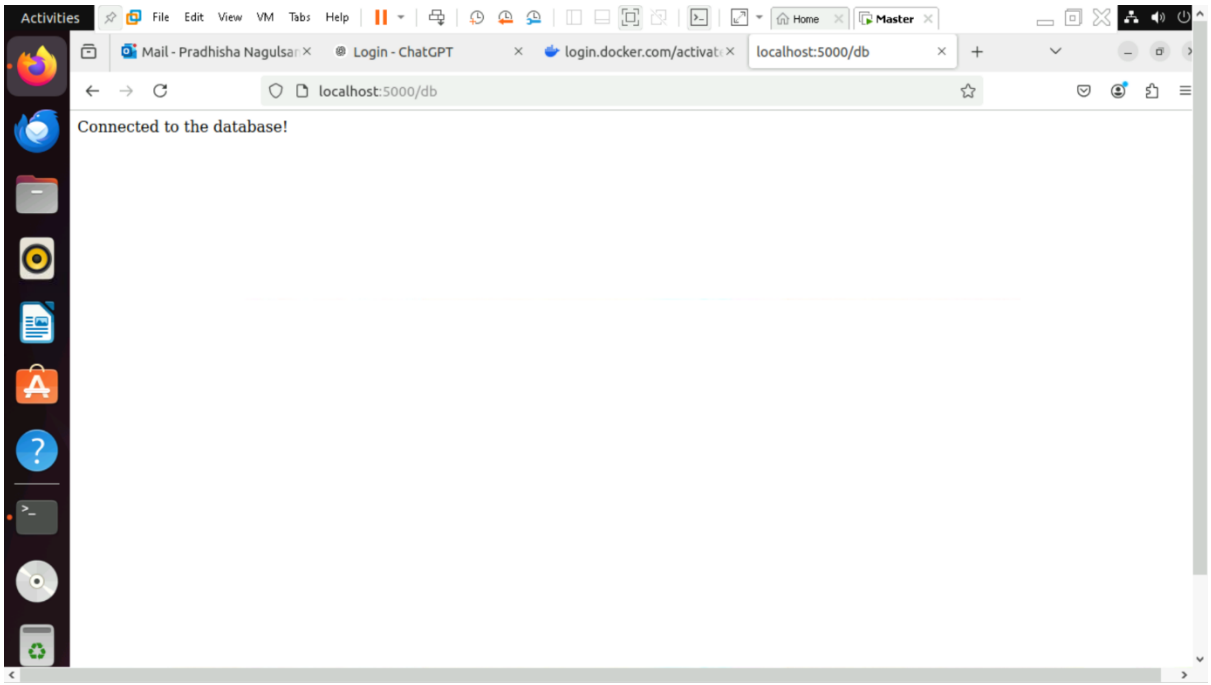


Step 5: Test the application.

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

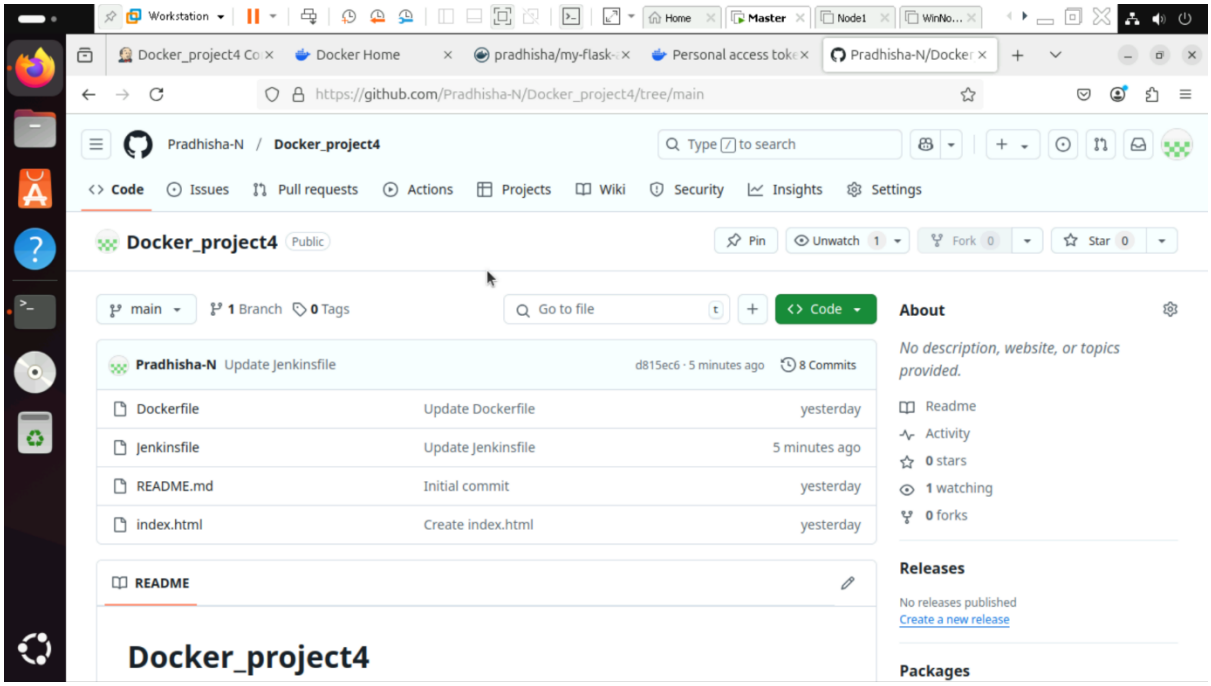
3. **Save the file and restart SSH:**

   sudo systemctl restart ssh

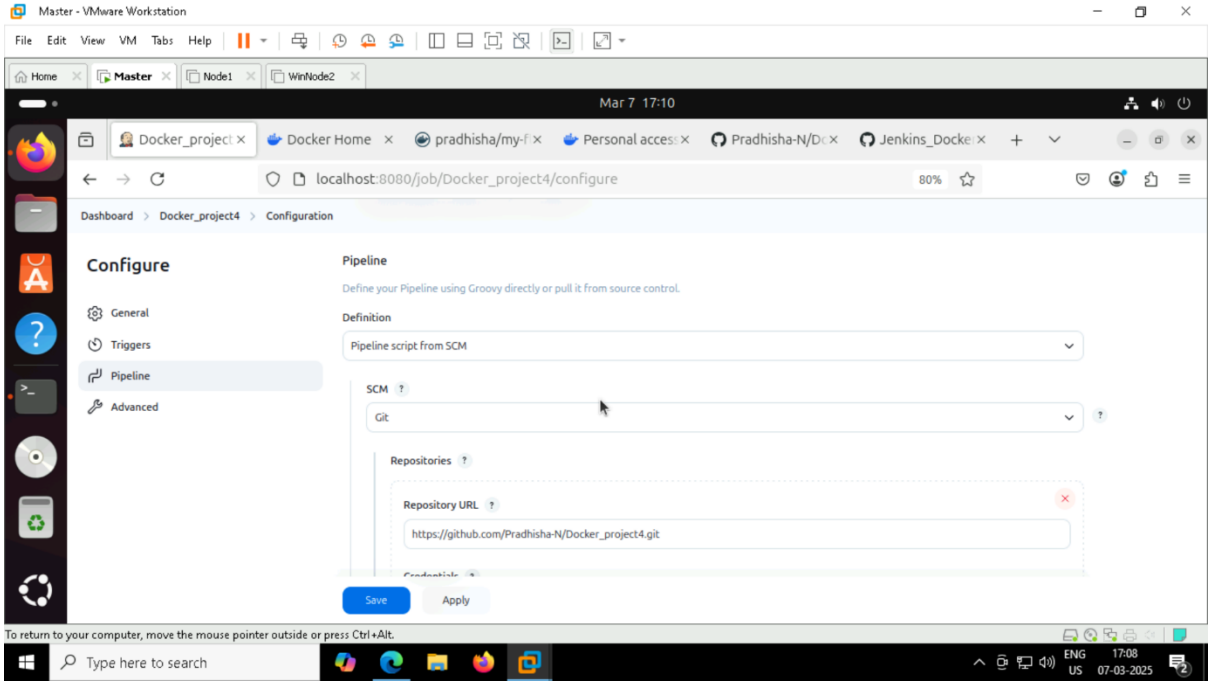
4. **Test SSH login:**

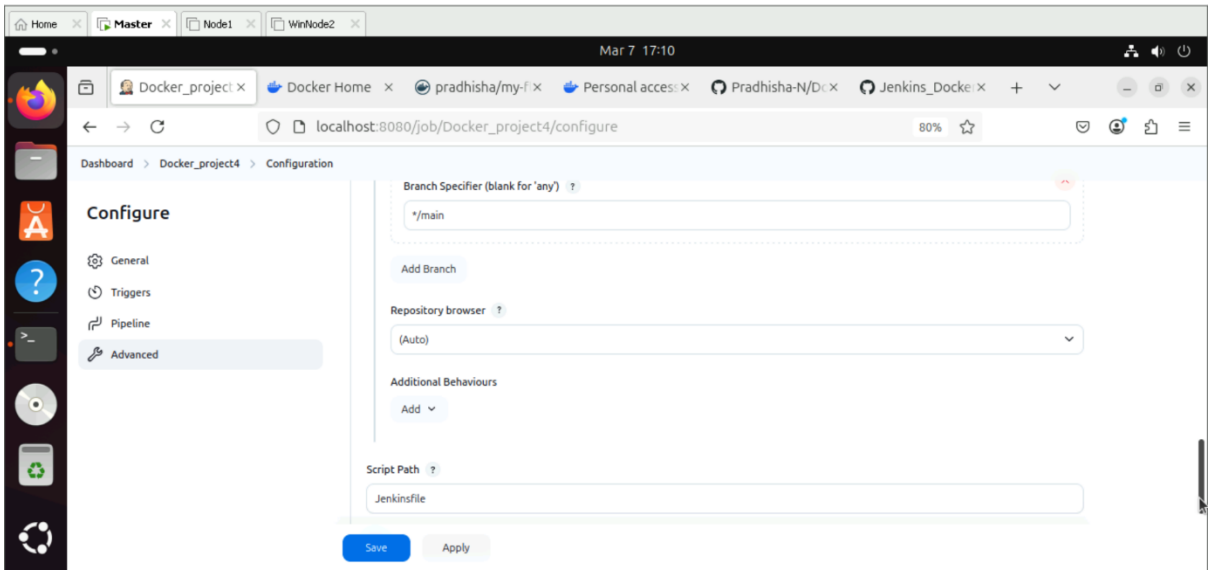
   ssh master@192.168.203.128

Step 3: Create a Simple Web Application

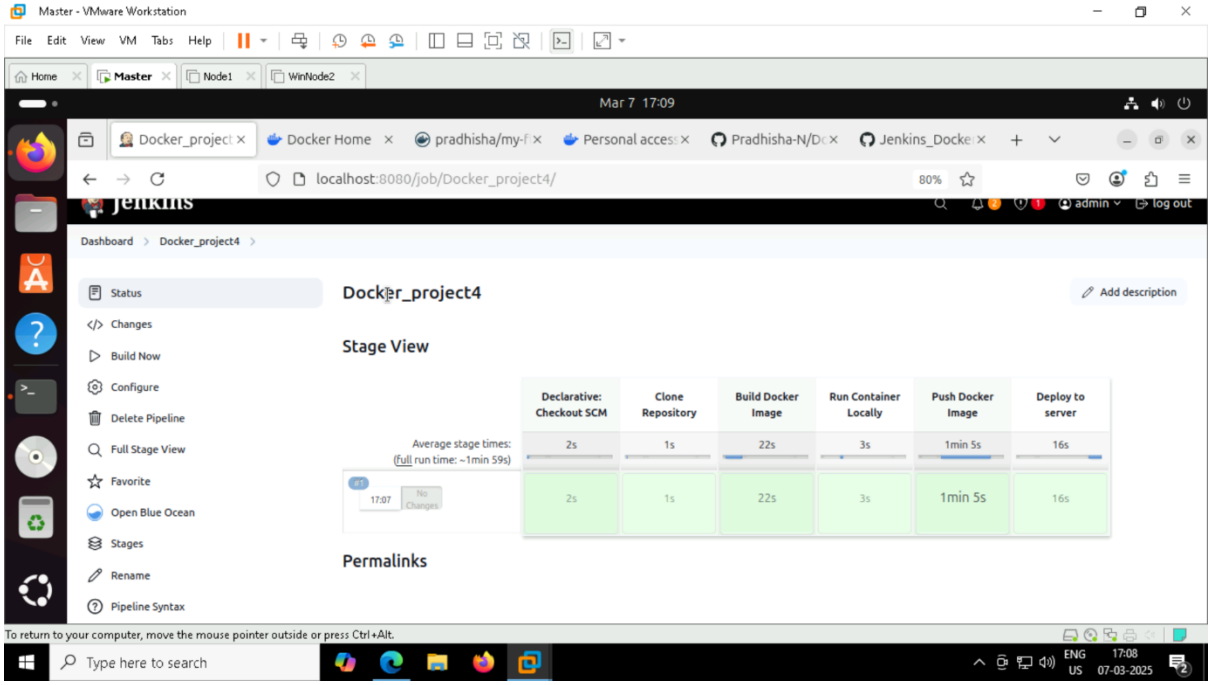


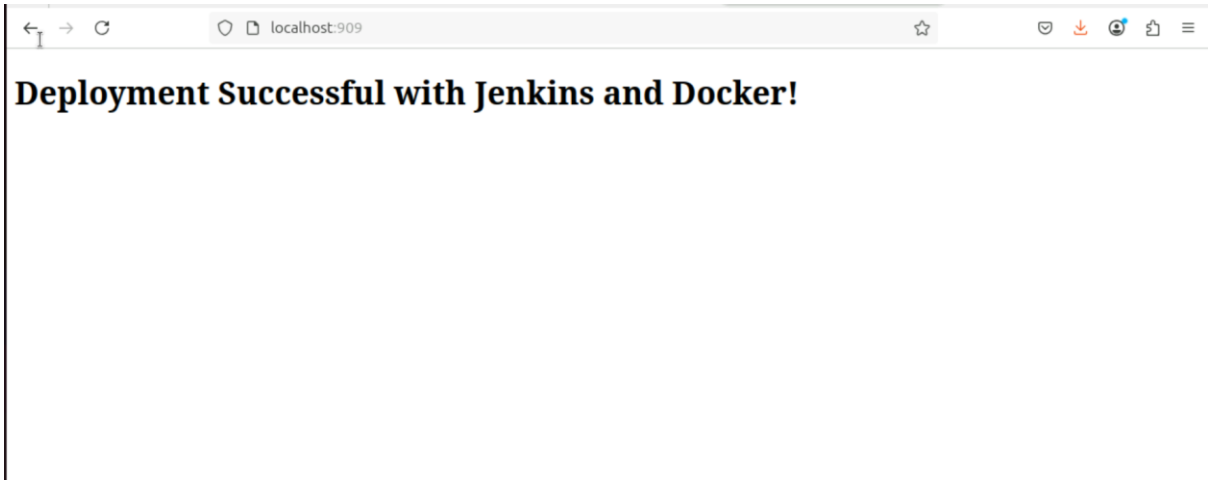
Step 4: Create a Jenkins Pipeline





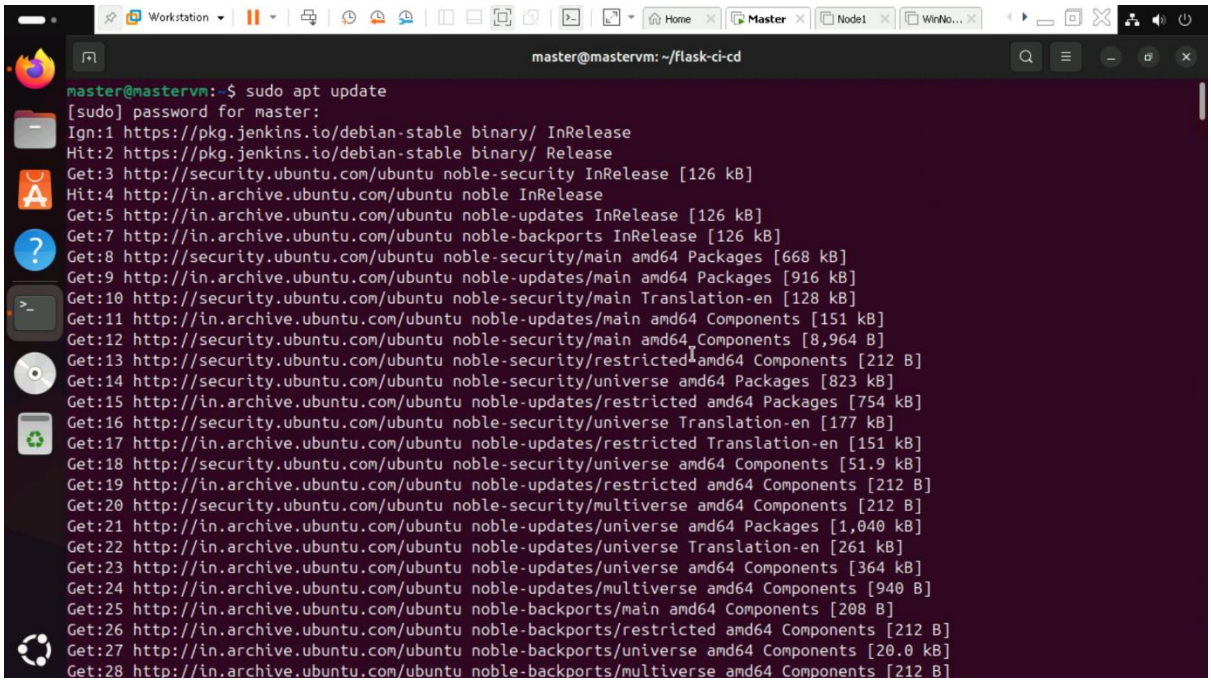
Step 5: Run the Jenkins Pipeline

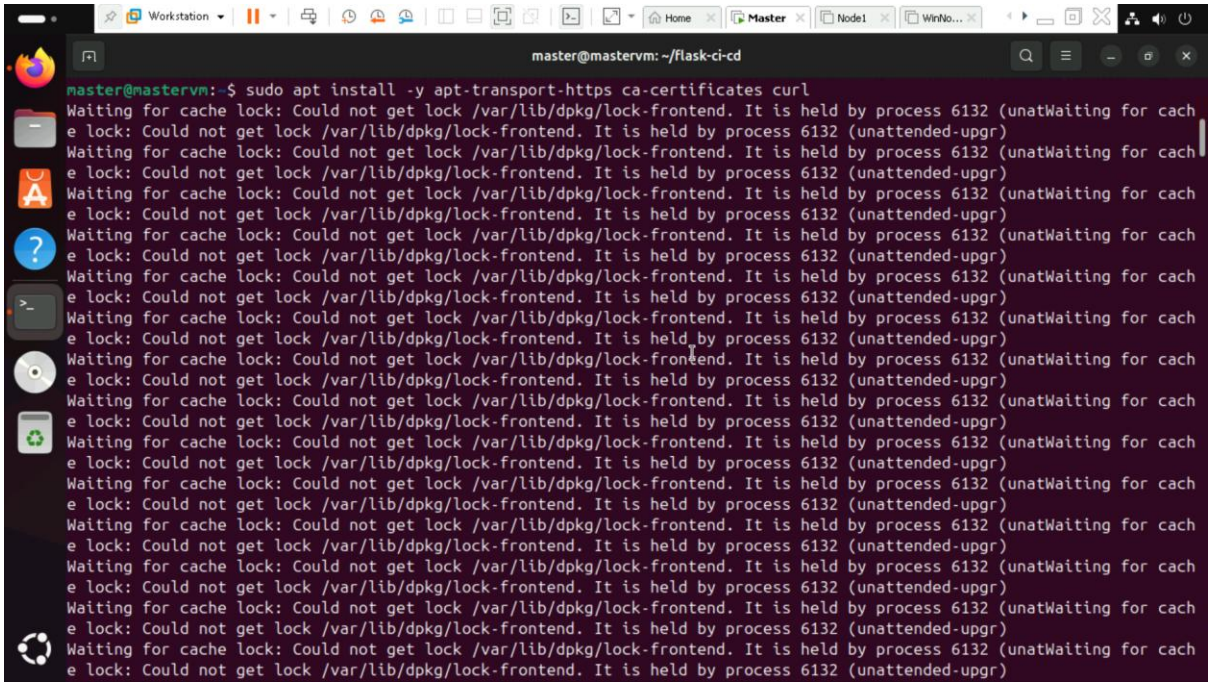




3. Dockerized CI/CD Pipeline using Jenkins and Kubernetes

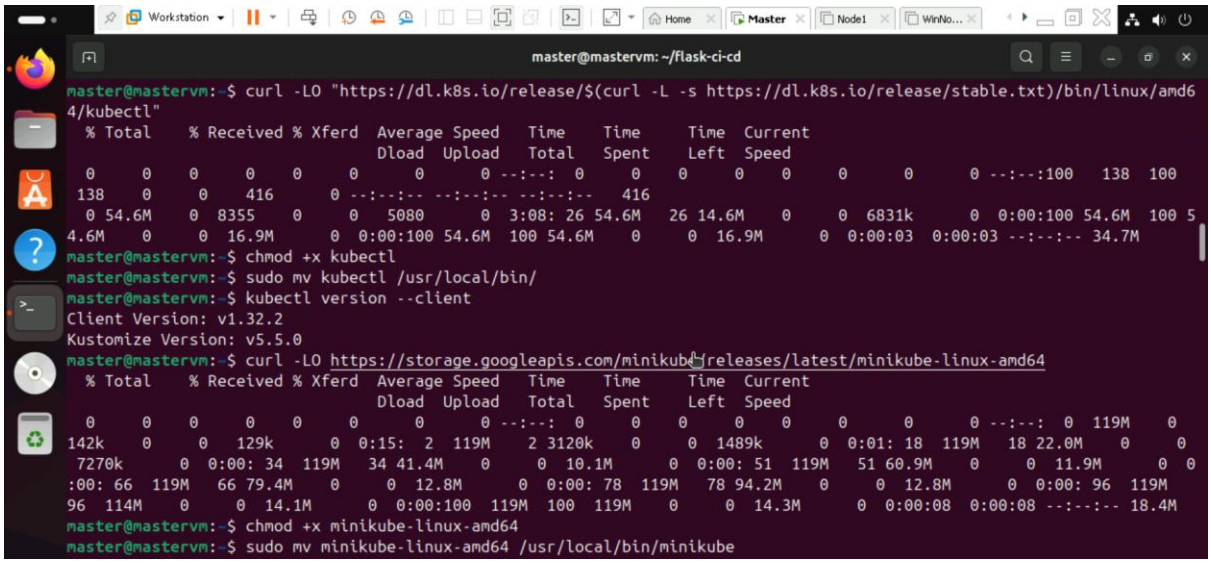
Step 1: Install Dependencies



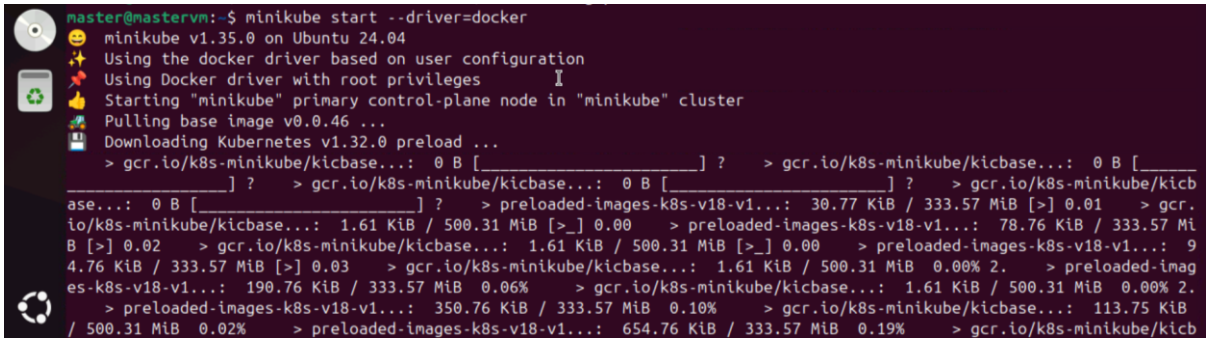


Step 2: Install kubectl

Step 3: Install minikube

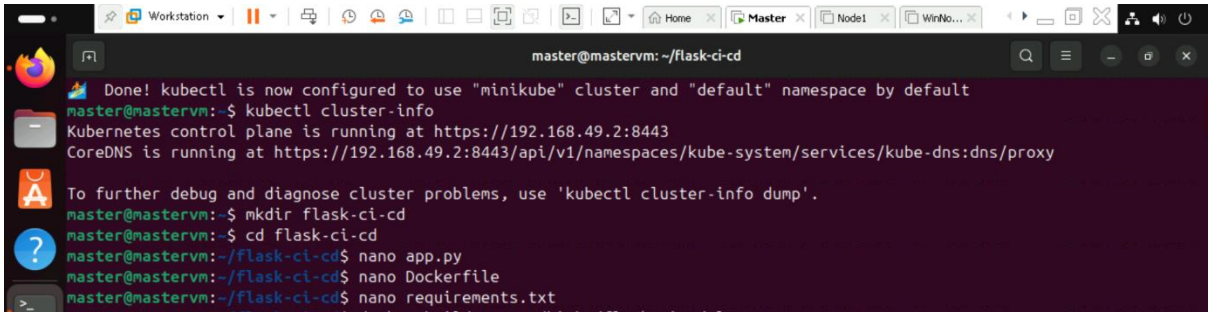


Step 4: Start minikube



Step 5: Check status

Step 6: Create app.py file, Docker file, requirements.txt file.



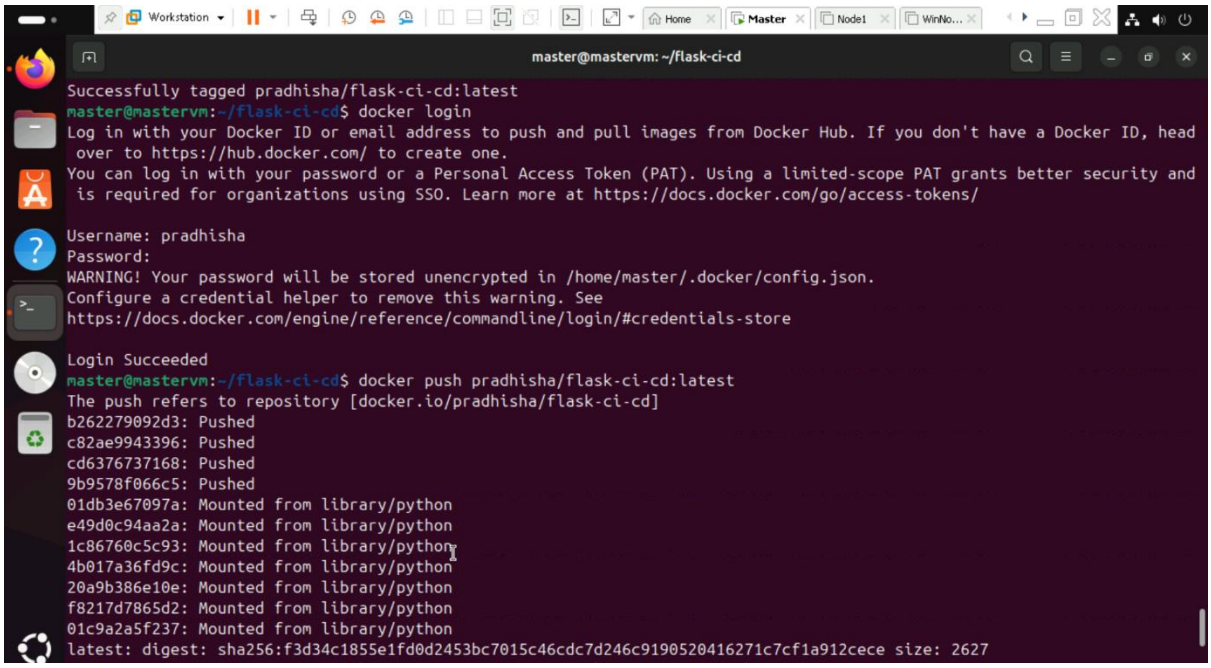
Step 7: Build the docker image.

```
master@mastervm:~/flask-ci-cd$ docker build -t pradhisha/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
Get "https://registry-1.docker.io/v2/": net/http: TLS handshake timeout
master@mastervm:~/flask-ci-cd$ docker pull python:3.9
3.9: Pulling from library/python
155ad54a8b28: Pull complete
8031108f3cda: Pull complete
1d281e50d3e4: Pull complete
447713e77b4f: Pull complete
93bee3686f31: Pull complete
95b7226c62e1: Pull complete
521cad6ddc53: Pull complete
Digest: sha256:5ea663a1c6ba266fdcac5949d1d2ea364ce30a2da92a3df95bb3c01437633ad9
Status: Downloaded newer image for python:3.9
docker.io/library/python:3.9
```

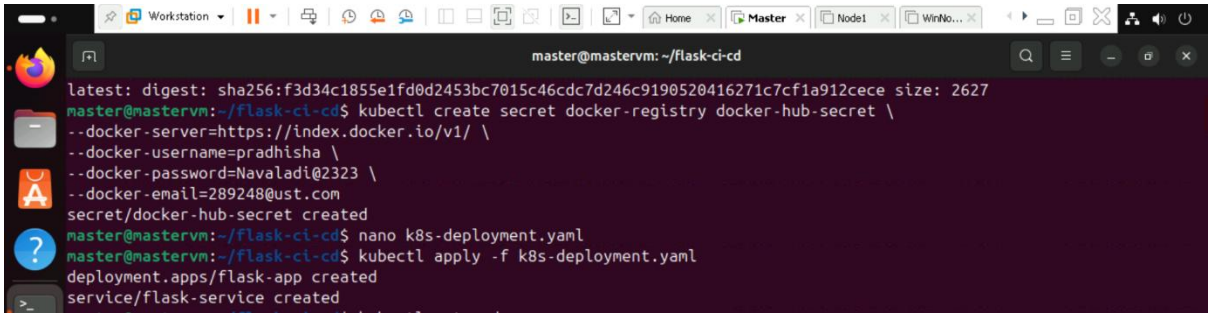
```
master@mastervm: ~/flask-ci-cd
docker.io/library/python:3.9
master@mastervm:~/flask-ci-cd$ docker build -t pradhisha/flask-ci-cd:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            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
 ---> Running in 5cbf3cd9e8ff
 ---> Removed intermediate container 5cbf3cd9e8ff
 ---> 4763963df6bf
Step 3/6 : COPY requirements.txt requirements.txt
 ---> baace9eec900
Step 4/6 : RUN pip install -r requirements.txt
 ---> Running in 69bc5e4fd33b
Collecting Flask
  Downloading flask-3.1.0-py3-none-any.whl (102 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 103.0/103.0 kB 1.7 MB/s eta 0:00:00
Collecting Jinja2>=3.1.2
  Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.9/134.9 kB 5.9 MB/s eta 0:00:00
Collecting click>=8.1.3
  Downloading click-8.1.8-py3-none-any.whl (98 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.2/98.2 kB 9.3 MB/s eta 0:00:00
Collecting blinker>=1.9
  Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Collecting Werkzeug>=3.1
```

Step 8: Login to docker.

Step 9: Push the docker image.



Step 10: Connect Kubernetes to Docker and create k8s-deployement. Apply the deployment.



Step 11: Check if the pods are running.