

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/

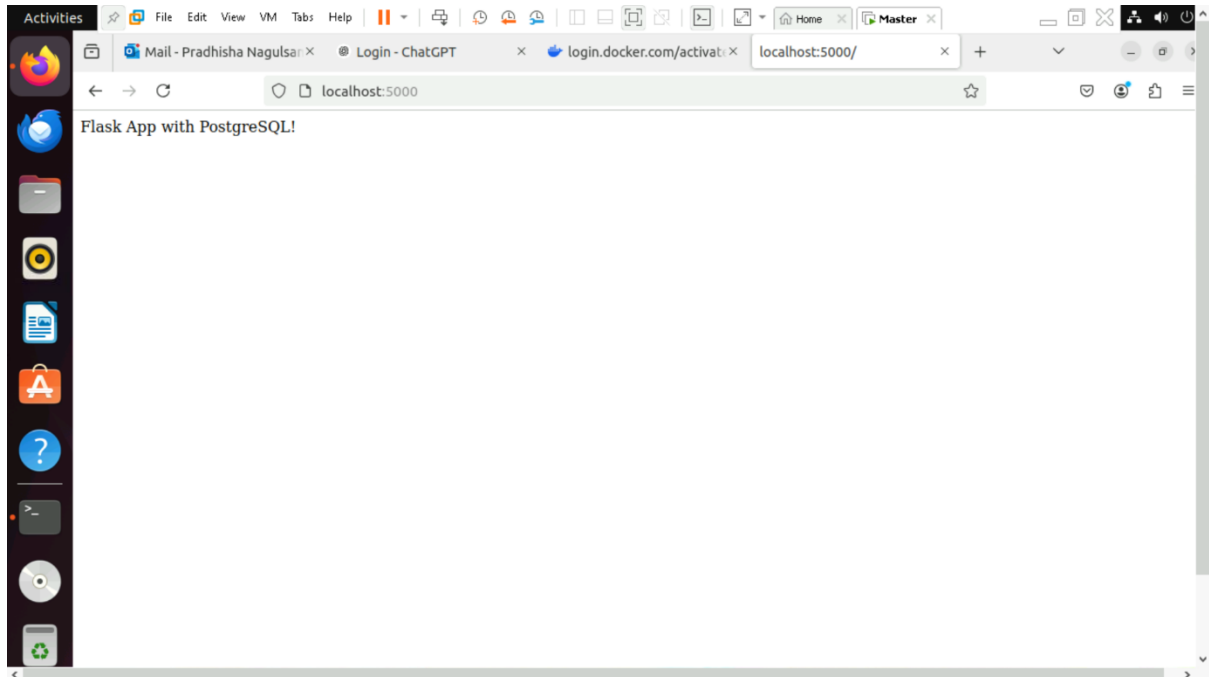
Login Succeeded
master@master-vm:~/Flask-Docker$ docker-compose up -d --build
Pulling db (postgres)...
latest: Pulling from library/postgres
7cf63256a31a: Already exists
543c6dea2e39: Pull complete
dc87fb4dbc03: Pull complete
55c54708c8e7: Pull complete
878a40f56a67: Pull complete
6424ae1ae883: Pull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

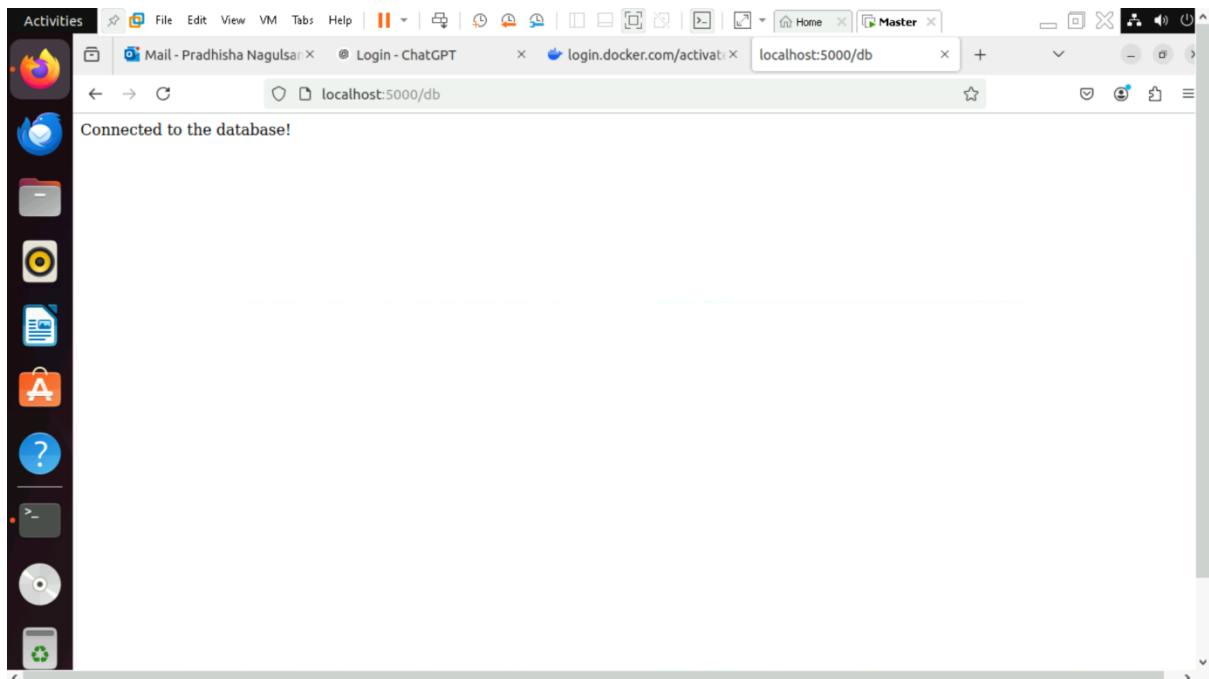
```
master@master-vm:~/Flask-Docker$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
24633d30f8c4   flask-docker_web_1   "python app.py"         35 seconds ago   Up 34 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
c6e2be65acf5   postgres   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

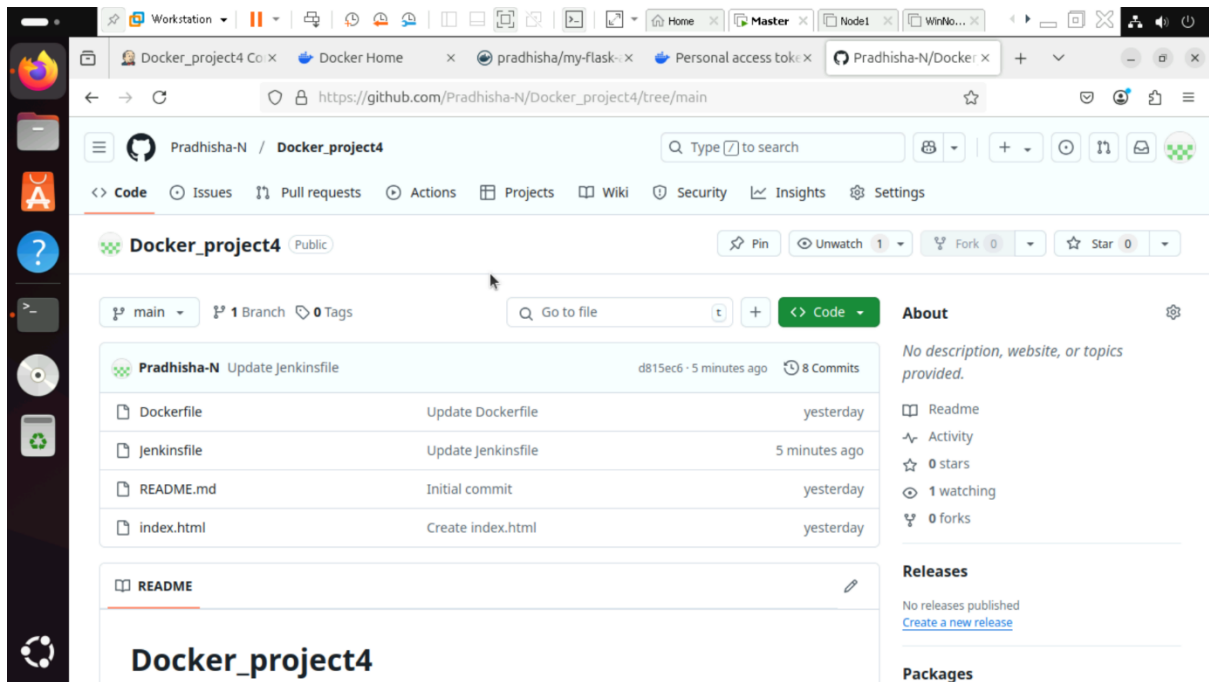
3. **Save the file and restart SSH:**

```
sudo systemctl restart ssh
```

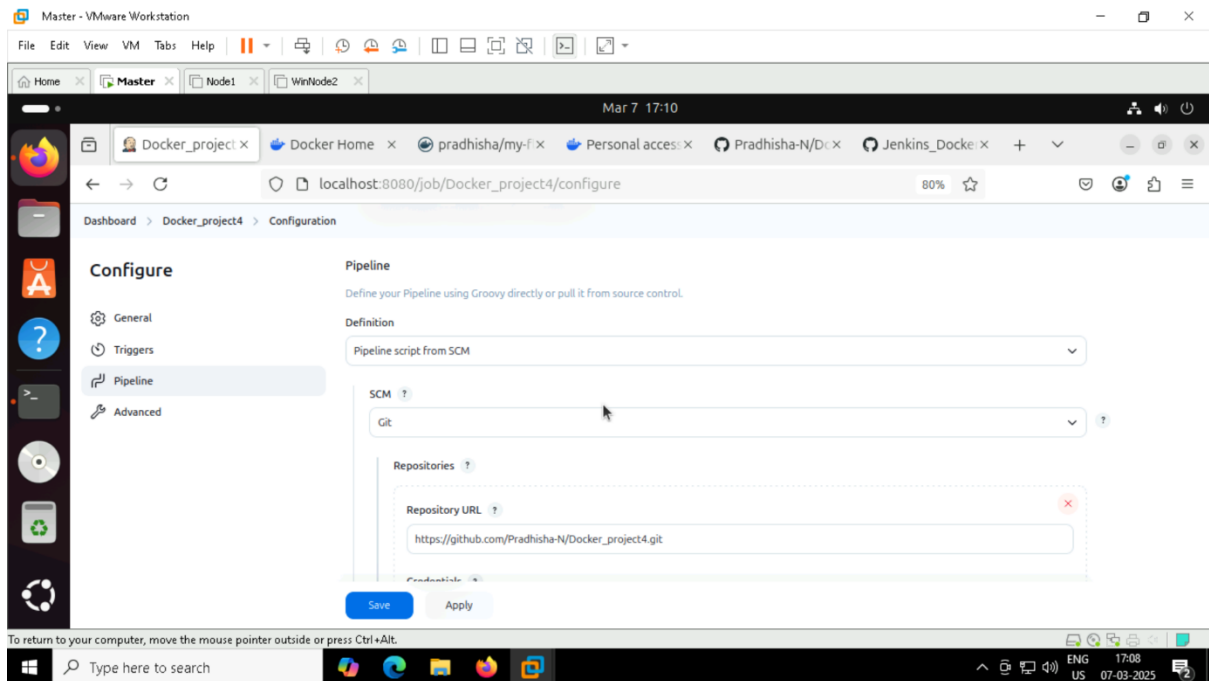
4. **Test SSH login:**

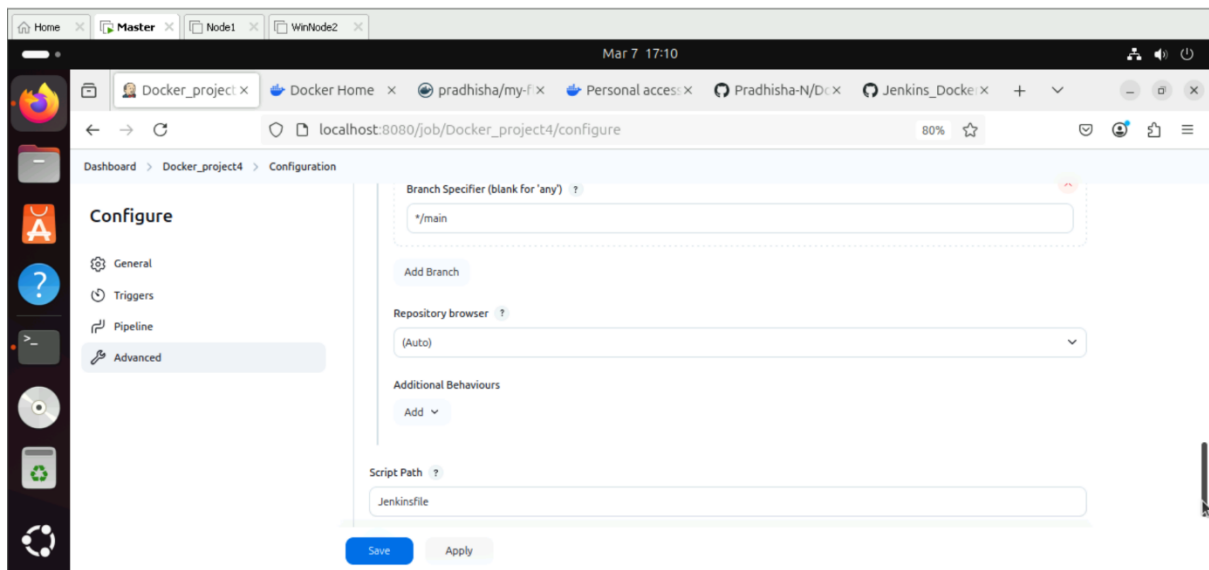
```
ssh master@192.168.203.128
```

Step 3: Create a Simple Web Application

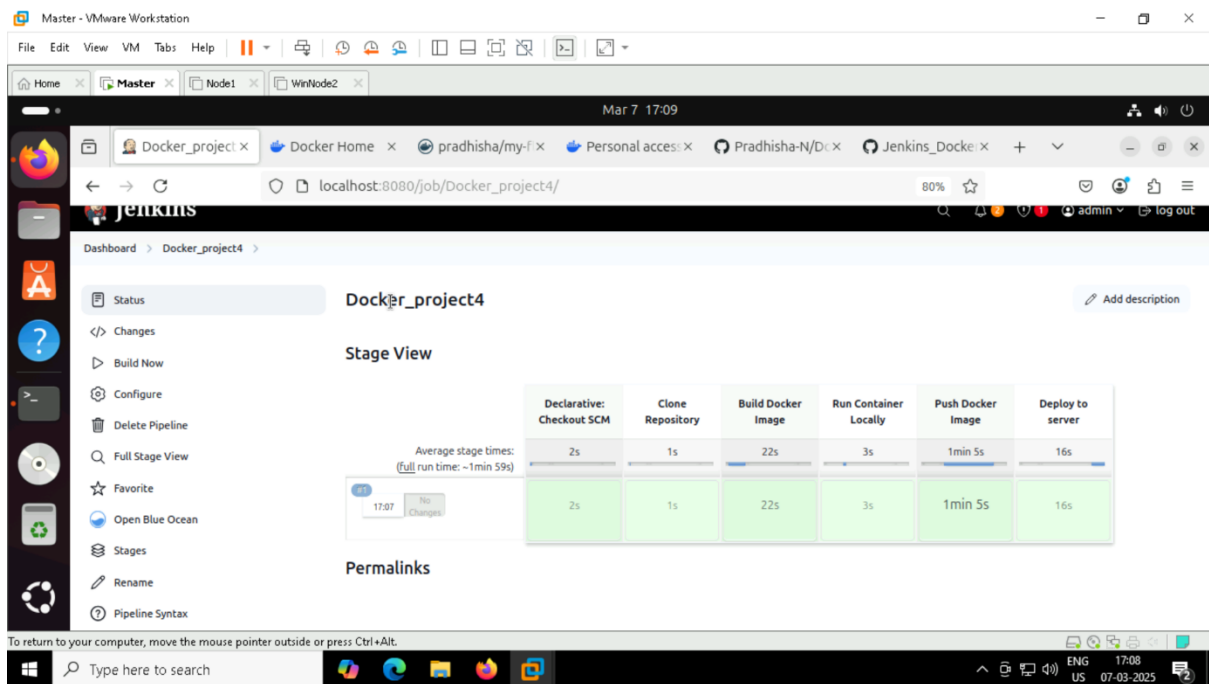


Step 4: Create a Jenkins Pipeline





Step 5: Run the Jenkins Pipeline



Deployment Successful with Jenkins and Docker!

Step 1: Install Dependencies

Step 3: Install minikube

The terminal shows the following commands and output:

```
master@mastervm: ~ - flask-ci-ud
master@mastervm:~$ curl -LO "https://dl.k8s.io/release/${curl -L -s https://dl.k8s.io/release/stable.txt}/bin/linux/amd64/kubectrl"
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left     Speed
  0     0    0     0     0      0     0 --:--:--  0      0     0     0      0     0 --:--:-- 100    138    100
138    0    0    416     0 --:--:--  --:--:--  --:--:--  416           0     0     0     0      0     0 --:--:-- 100    54.6M    100
54.6M    0    0   8355     0     5080     3:08: 26 54.6M    26 14.6M    0     0   6831k    0 0:00:100 54.6M    100
4.6M    0    0   16.9M    0 0:00:100 54.6M    100 54.6M    0     0   16.9M    0     0   0:00:03 0:00:03 --:--:-- 34.7M

master@mastervm:~$ chmod +x kubectrl
master@mastervm:~$ sudo mv kubectrl /usr/local/bin/
master@mastervm:~$ kubectrl version --client
Client Version: v1.32.2
Kustomize Version: v5.5.0

master@mastervm:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left     Speed
  0     0    0     0     0      0     0 --:--:--  0      0     0     0      0     0 --:--:--  0    119M    0
142k    0    0   129k    0 0:01:15  2 119M    2 3120k    0     0   1489k    0 0:01:18 119M    18 22.0M    0 0
7270k    0 0:00:34 119M    34 41.4M    0     0   10.1M    0 0:00:51 119M    51 60.9M    0 0 11.9M    0 0
:00:66 119M    66 79.4M    0     12.8M    0 0:00:78 119M    78 94.2M    0     0   12.8M    0 0:00:96 119M
96 114M    0     14.1M    0 0:00:100 119M    100 119M    0     0   14.3M    0 0 0:00:08 0:00:08 --:--:-- 18.4M

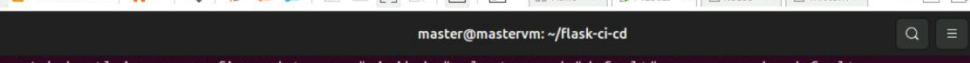
master@mastervm:~$ chmod +x minikube-linux-amd64
master@mastervm:~$ sudo mv minikube-linux-amd64 /usr/local/bin/minikube
```

Step 4: Start minikube

```
master@mastervm:~$ minikube start --driver=docker
🐵 minikube v1.35.0 on Ubuntu 24.04
🔧 Using the docker driver based on user configuration
👤 Using Docker driver with root privileges
🏠 Starting "minikube" primary control-plane node in "minikube" cluster
📥 Pulling base image v0.0.46 ...
🎗 Downloading Kubernetes v1.32.0 preload ...
> gcr.io/k8s-minikube/kicbase...: 0 B [_____] ? > gcr.io/k8s-minikube/kicbase...: 0 B [_____] ?
> gcr.io/k8s-minikube/kicbase...: 0 B [_____] ? > gcr.io/k8s-minikube/kicbase...: 0 B [_____] ?
ase...: 0 B [_____] ? > preloaded-images-k8s-v18-v1...: 30.77 KiB / 333.57 MiB [>] 0.01 > gcr.io/k8s-minikube/kicbase...: 1.61 KiB / 500.31 MiB [>.] 0.00 > preloaded-images-k8s-v18-v1...: 78.76 KiB / 333.57 MiB [>.] 0.02 > gcr.io/k8s-minikube/kicbase...: 1.61 KiB / 500.31 MiB [>.] 0.00 > preloaded-images-k8s-v18-v1...: 94.76 KiB / 333.57 MiB [>.] 0.03 > gcr.io/k8s-minikube/kicbase...: 1.61 KiB / 500.31 MiB 0.00% 2. > preloaded-images-k8s-v18-v1...: 190.76 KiB / 333.57 MiB 0.06% > gcr.io/k8s-minikube/kicbase...: 1.61 KiB / 500.31 MiB 0.00% 2. > preloaded-images-k8s-v18-v1...: 350.76 KiB / 333.57 MiB 0.10% > gcr.io/k8s-minikube/kicbase...: 113.75 KiB / 500.31 MiB 0.02% > preloaded-images-k8s-v18-v1...: 654.76 KiB / 333.57 MiB 0.19% > gcr.io/k8s-minikube/kicbase...
```

Step 5: Check status

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



```
master@mastervm: ~/flask-ci-cd
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
master@mastervm:~$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.49.2:8443
CoreDNS is running at https://192.168.49.2:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
master@mastervm:~$ mkdir flask-ci-cd
master@mastervm:~$ cd flask-ci-cd
master@mastervm:~/flask-ci-cd$ nano app.py
master@mastervm:~/flask-ci-cd$ nano Dockerfile
master@mastervm:~/flask-ci-cd$ nano requirements.txt
```


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

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.

```
master@mastervm: ~/flask-ci-cd
Successfully tagged pradhisha/flask-ci-cd:latest
master@mastervm:~/flask-ci-cd$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head
over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and
is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/
Username: pradhisha
Password:
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 pradhisha/flask-ci-cd:latest
The push refers to repository [docker.io/pradhisha/flask-ci-cd]
b262279092d3: Pushed
c82ae9943396: Pushed
cd6376737168: Pushed
9b9578f066c5: Pushed
01db3e67097a: Mounted from library/python
e49d0c94aa2a: Mounted from library/python
1c86760c5c93: Mounted from library/python
4b017a36fd9c: Mounted from library/python
20a9b386e10e: Mounted from library/python
f8217d7865d2: Mounted from library/python
01c9a2a5f237: Mounted from library/python
latest: digest: sha256:f3d34c1855e1fd0d2453bc7015c46cdc7d246c9190520416271c7cf1a912cece size: 2627
```

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

```
master@mastervm: ~/flask-ci-cd
latest: digest: sha256:f3d34c1855e1fd0d2453bc7015c46cdc7d246c9190520416271c7cf1a912cece size: 2627
master@mastervm:~/flask-ci-cd$ kubectl create secret docker-registry docker-hub-secret \
--docker-server=https://index.docker.io/v1/ \
--docker-username=pradhisha \
--docker-password=Navaladi2323 \
--docker-email=289248@ust.com
secret/docker-hub-secret created
master@mastervm:~/flask-ci-cd$ nano k8s-deployment.yaml
master@mastervm:~/flask-ci-cd$ kubectl apply -f k8s-deployment.yaml
deployment.apps/flask-app created
service/flask-service created
```

Step 11: Check if the pods are running.

```
master@mastervm:~/flask-ci-cd$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
flask-app-58b8cc8758-bbct4          1/1     Running   0           114s
flask-app-58b8cc8758-jdv4w          1/1     Running   0           114s
master@mastervm:~/flask-ci-cd$
```