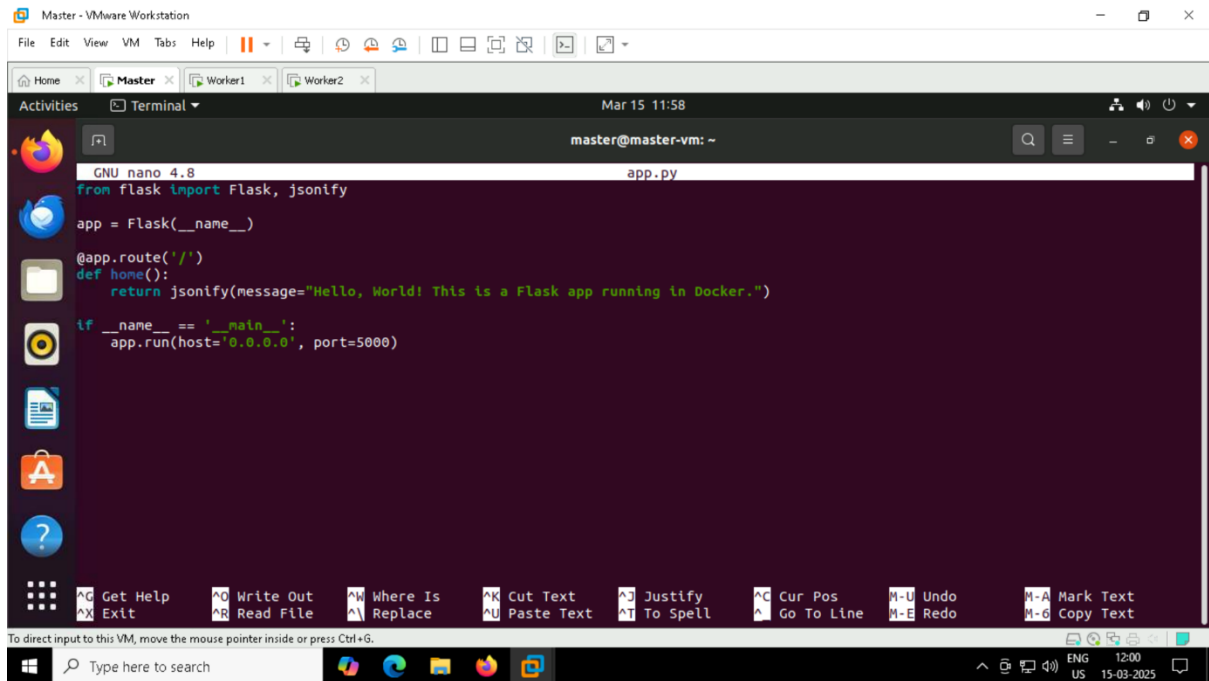


Kubernetes Project -1

Deploying a Flask Application on Kubernetes with Auto-Scaling

Step 1: Building and Containerizing the Flask Application

- Flask Application (app.py)



The screenshot shows a terminal window titled 'Master - VMware Workstation' with tabs for 'Master', 'Worker1', and 'Worker2'. The terminal is running 'GNU nano 4.8' and editing a file named 'app.py'. The code in the file is as follows:

```
from flask import Flask, jsonify

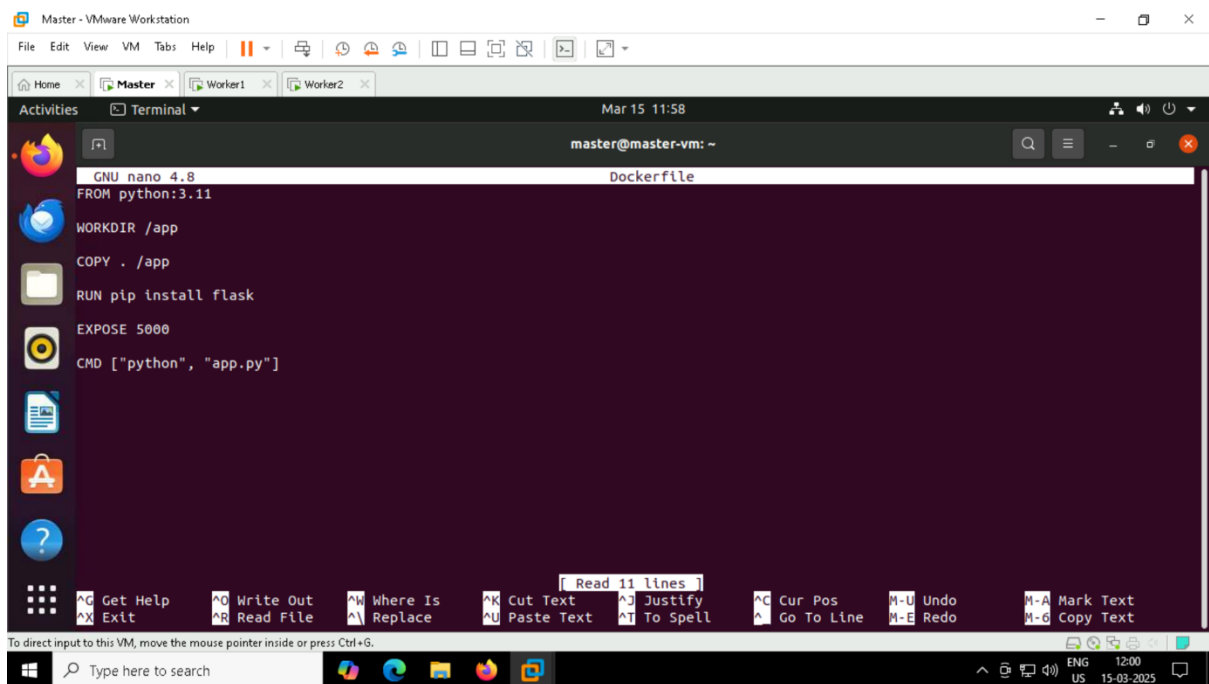
app = Flask(__name__)

@app.route('/')
def home():
    return jsonify(message="Hello, World! This is a Flask app running in Docker.")

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

The terminal window also shows a status bar at the bottom with various keyboard shortcuts and a system tray on the right indicating the date and time as 'Mar 15 11:58'.

- Create a Dockerfile



The screenshot shows a terminal window titled 'Master - VMware Workstation' with tabs for 'Master', 'Worker1', and 'Worker2'. The terminal is running 'GNU nano 4.8' and editing a file named 'Dockerfile'. The code in the file is as follows:

```
FROM python:3.11

WORKDIR /app

COPY . /app

RUN pip install flask

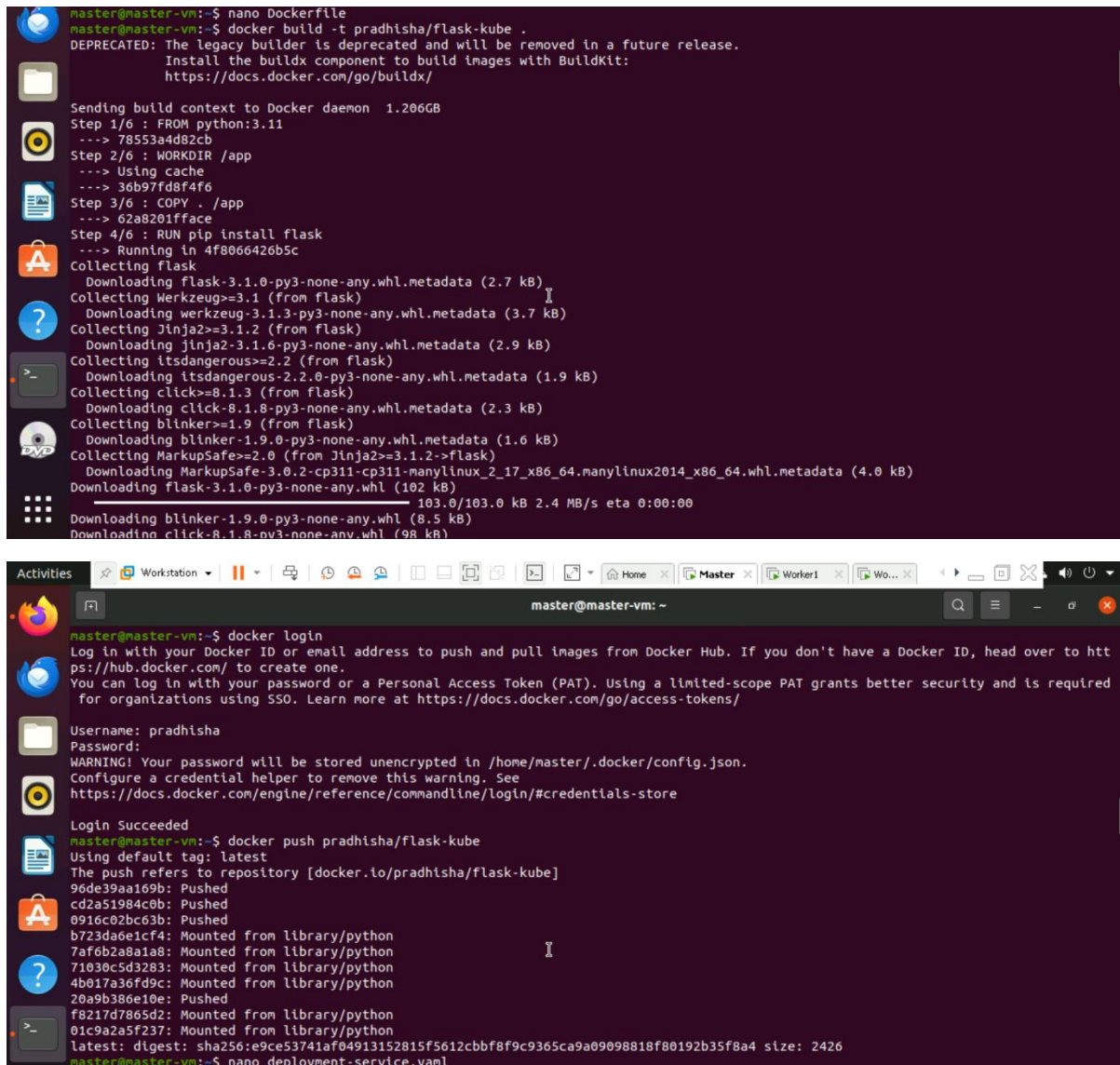
EXPOSE 5000

CMD ["python", "app.py"]
```

The terminal window also shows a status bar at the bottom with various keyboard shortcuts and a system tray on the right indicating the date and time as 'Mar 15 11:58'.

Step 2: Build and Push the Image

- `docker build -t pradhisha/flask-kube .`
- `docker push pradhisha/flask-kube`



The first terminal screenshot shows the execution of `docker build -t pradhisha/flask-kube .`. It displays the build context being sent to the Docker daemon, followed by a series of steps: FROM python:3.11, WORKDIR /app, COPY . /app, and RUN pip install flask. The build process then proceeds to collect and download various dependencies, including flask, werkzeug, Jinja2, itsdangerous, click, and blinker, along with their respective metadata. The final step shows the download of the flask wheel file.

```
master@master-vm:~$ nano Dockerfile
master@master-vm:~$ docker build -t pradhisha/flask-kube .
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 1.206GB
Step 1/6 : FROM python:3.11
--> 78553a4d82cb
Step 2/6 : WORKDIR /app
--> Using cache
--> 36b97fd8f4f6
Step 3/6 : COPY . /app
--> 62a8201fface
Step 4/6 : RUN pip install flask
--> Running in 4f8066426b5c
Collecting flask
  Downloading flask-3.1.0-py3-none-any.whl.metadata (2.7 kB)
Collecting Werkzeug>=3.1 (from flask)
  Downloading werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Collecting Jinja2>=3.1.2 (from flask)
  Downloading Jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting itsdangerous>=2.2 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Collecting blinker>=1.9 (from flask)
  Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->flask)
  Downloading MarkupSafe-3.0.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.0 kB)
Downloading flask-3.1.0-py3-none-any.whl (102 kB)
103.0/103.0 kB 2.4 MB/s eta 0:00:00
Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Downloading click-8.1.8-py3-none-any.whl (98 kB)
```

The second terminal screenshot shows the execution of `docker login` and `docker push pradhisha/flask-kube`. It displays the login process, including the username (pradhisha) and password, and the push process, which shows the image being pushed to the repository [docker.io/pradhisha/flask-kube]. The push process also shows the image being mounted from the library/python base image.

```
master@master-vm:~$ 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@master-vm:~$ docker push pradhisha/flask-kube
Using default tag: latest
The push refers to repository [docker.io/pradhisha/flask-kube]
96de39aa169b: Pushed
cd2a51984c0b: Pushed
0916c02bc63b: Pushed
b723da6e1cf4: Mounted from library/python
7af6b2a8a1a8: Mounted from library/python
71030c5d3283: Mounted from library/python
4b017a36fd9c: Mounted from library/python
20a9b386e10e: Pushed
f8217d7865d2: Mounted from library/python
01c9a2a5f237: Mounted from library/python
latest: digest: sha256:e9ce53741af04913152815f5612cbbf8f9c9365ca9a09098818f80192b35f8a4 size: 2426
master@master-vm:~$ nano deployment-service.yaml
```

Step 3: Deploying Flask App on Kubernetes

- Create Deployment & Service YAML (deployment-service.yaml)

```
GNU nano 4.8 deployment-service.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
        - name: flask-container
          image: kpk25/flask-kube:latest
          ports:
            - containerPort: 5000
          resources:
            requests:
              cpu: "100m"
            limits:
              cpu: "250m"
          imagePullSecrets:
            - name: docker-secret
---
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: flask-service
spec:
  selector:
    app: flask-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5000
  type: NodePort
```

- Apply Deployment
- Patch Default Service Account

```
master@master-vm: ~$ kubectl apply -f deployment-service.yaml
deployment.apps/flask-app created
service/flask-service created
master@master-vm: ~$ kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name": "docker-secret"}]}'
serviceaccount/default patched
```

Step 4: Installing and Troubleshooting Metrics Server

```
master@master-vm: ~$ kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server-auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
```

Step 5: Enabling HPA (Horizontal Pod Autoscaler)

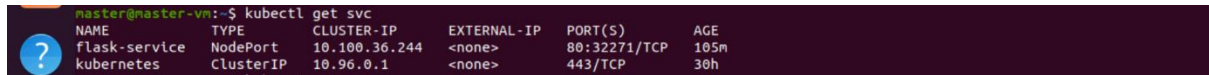
- `kubectl autoscale deployment flask-app --cpu-percent=50 --min=3 --max=10`
- `kubectl get hpa`

```
master@master-vm: ~$ kubectl autoscale deployment flask-app --cpu-percent=50 --min=3 --max=10
horizontalpodautoscaler.autoscaling/flask-app autoscaled
```

```
master@master-vm: ~$ kubectl get hpa
NAME         REFERENCE          TARGETS          MINPODS  MAXPODS  REPLICAS  AGE
flask-app    Deployment/flask-app  cpu: <unknown>/50%  3         10        3         37m
```

Step 6: Finding NodePort and Testing External Access

- `kubectl get svc`

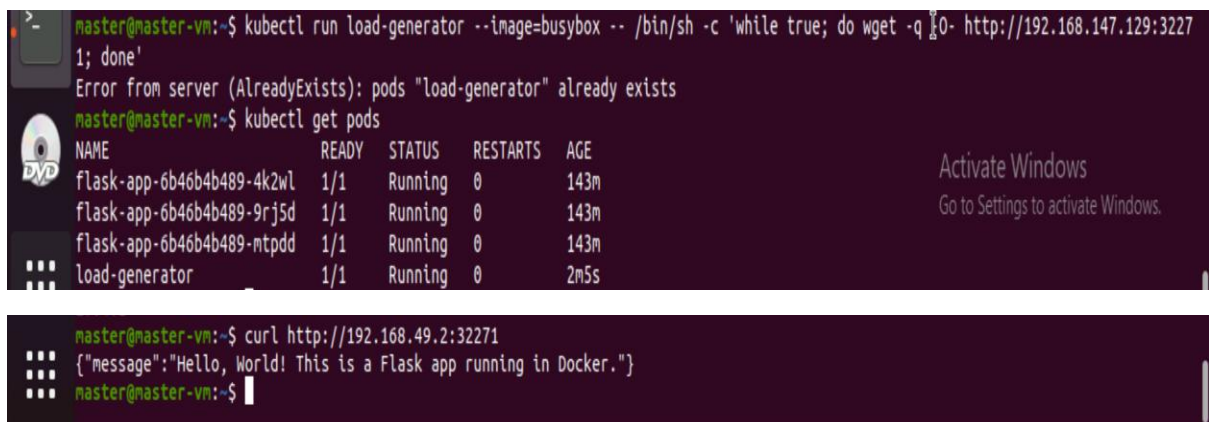


```
master@master-vn:~$ kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
flask-service	NodePort	10.100.36.244	<none>	80:32271/TCP	105m
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	30h

Step 7: Simulating Load for HPA

- `kubectl run -it --rm load-generator --image=busybox -- /bin/sh`
- `while true; do wget -q -O- http://192.168.147.129:32271; done`
- `kubectl get pods`



```
master@master-vn:~$ kubectl run load-generator --image=busybox -- /bin/sh -c 'while true; do wget -q -O- http://192.168.147.129:32271; done'
```

Error from server (AlreadyExists): pods "load-generator" already exists

```
master@master-vn:~$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
flask-app-6b46b4b489-4k2wl	1/1	Running	0	143m
flask-app-6b46b4b489-9rj5d	1/1	Running	0	143m
flask-app-6b46b4b489-ntpdd	1/1	Running	0	143m
load-generator	1/1	Running	0	2m5s

```
master@master-vn:~$ curl http://192.168.49.2:32271
```

```
{ "message": "Hello, World! This is a Flask app running in Docker." }
```

```
master@master-vn:~$
```

Step 8: View the json output in the browser by entering the IP address

