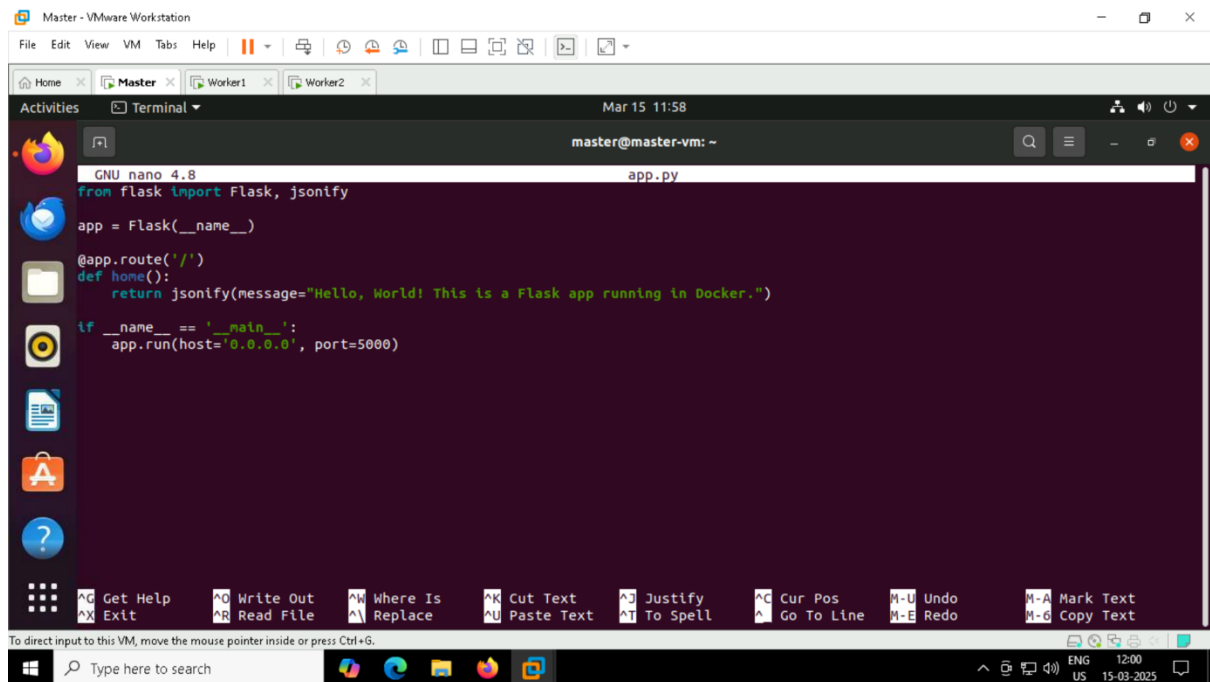


# Kubernetes Case Study

## 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@master-vm: ~' with the file 'app.py' open in nano 4.8. The code defines a Flask application with a single route '/' that returns a JSON message. The application is configured to run on host '0.0.0.0' and port '5000'.

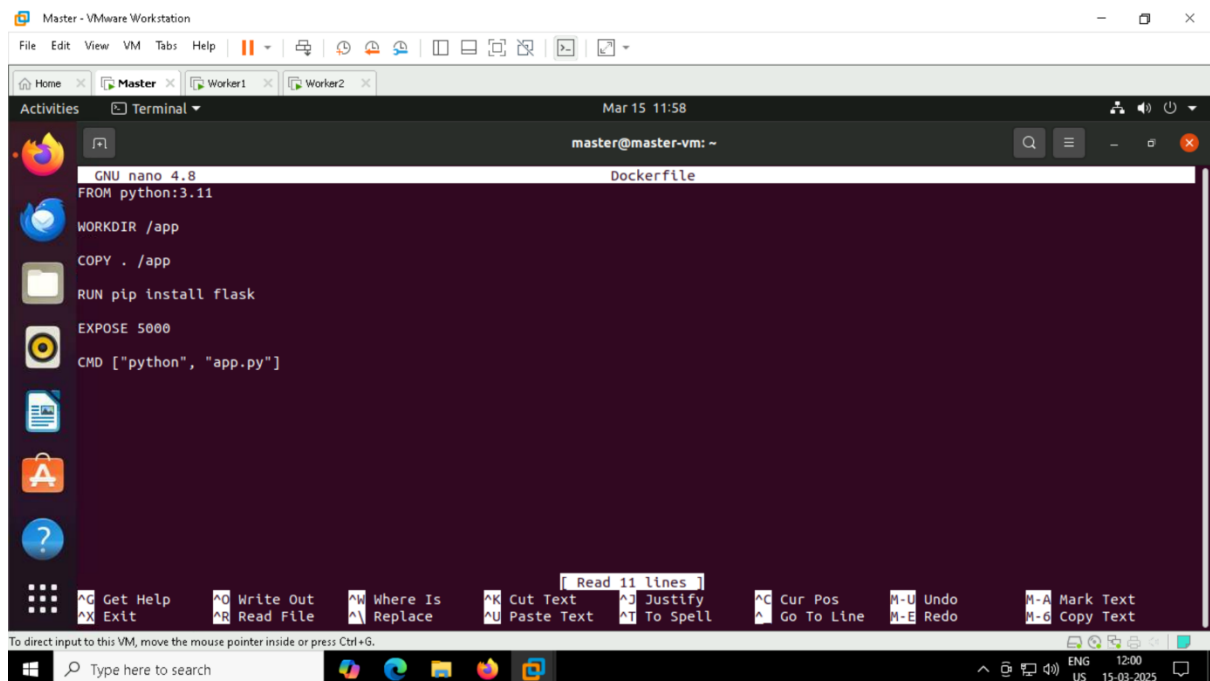
```
GNU nano 4.8 app.py
from flask import Flask, jsonify

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)
```

- Create a Dockerfile



The screenshot shows a terminal window titled 'master@master-vm: ~' with the file 'Dockerfile' open in nano 4.8. The Dockerfile contains instructions to build a container image from a Python 3.11 base image, set the working directory to /app, copy the application files, install Flask, and expose port 5000.

```
GNU nano 4.8 Dockerfile
FROM python:3.11

WORKDIR /app

COPY . /app

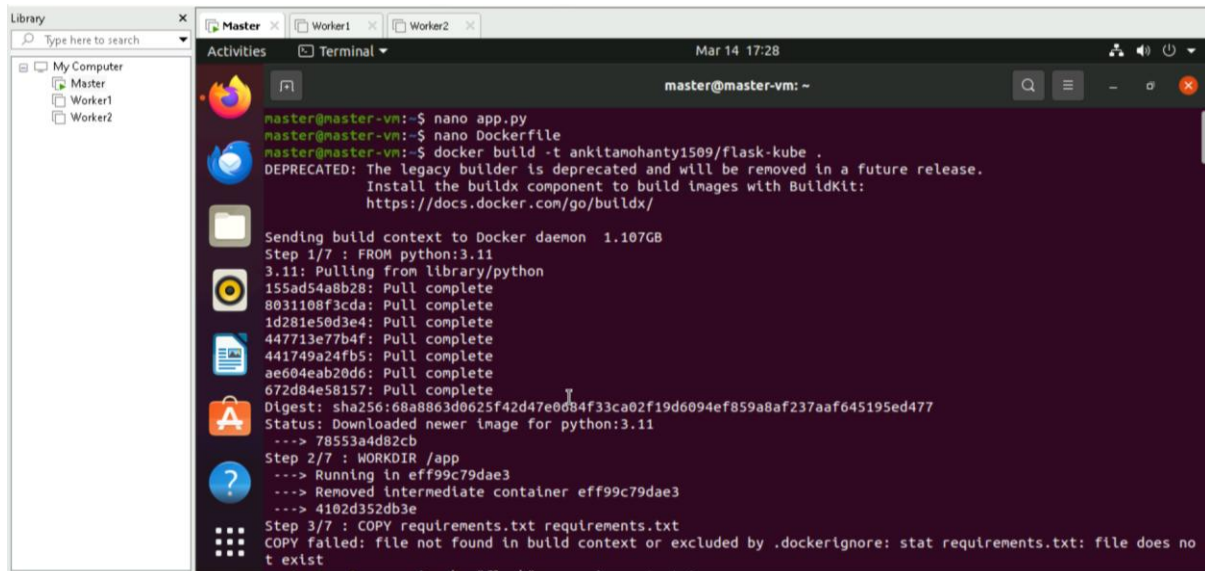
RUN pip install flask

EXPOSE 5000

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

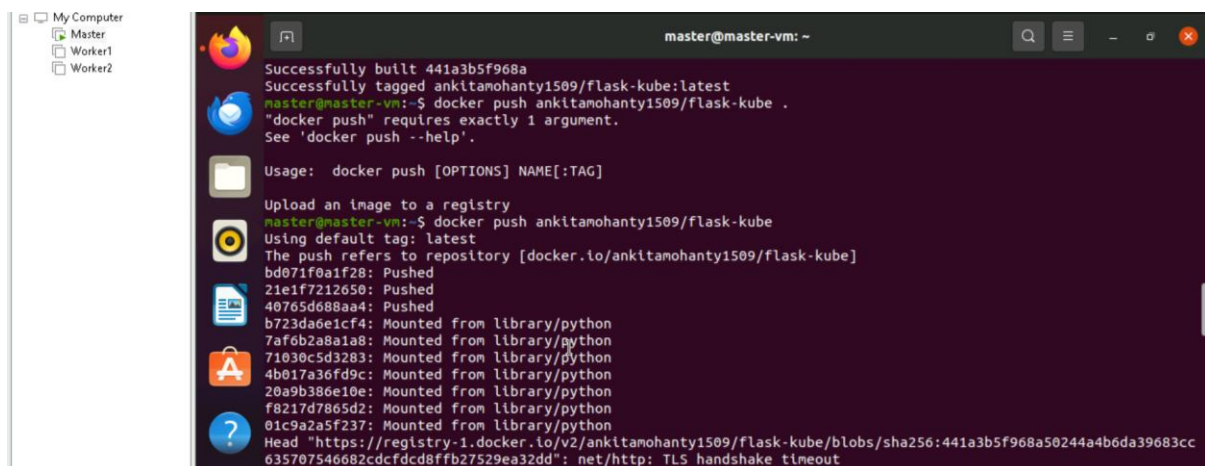
## Step 2: Build and Push the Image

- `docker build -t ankitamohanty1509/flask-kube .`
- `docker push ankitamohanty1509/flask-kube`



```
master@master-vm:~$ nano app.py
master@master-vm:~$ nano Dockerfile
master@master-vm:~$ docker build -t ankitamohanty1509/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 1.107GB
Step 1/7 : FROM python:3.11
3.11: Pulling from library/python
155ad54a8b28: Pull complete
8031108f3cda: Pull complete
1d281e50d3e4: Pull complete
447713e77b4f: Pull complete
441749a24fb5: Pull complete
ae604eab20d6: Pull complete
672d84e58157: Pull complete
Digest: sha256:68a8863d0625f42d47e0d84f33ca02f19d6094ef859a8af237aaf645195ed477
Status: Downloaded newer image for python:3.11
--> 78553a4d82cb
Step 2/7 : WORKDIR /app
--> Running in eff99c79dae3
--> Removed intermediate container eff99c79dae3
--> 4102d352db3e
Step 3/7 : COPY requirements.txt requirements.txt
COPY failed: file not found in build context or excluded by .dockerignore: stat requirements.txt: file does not exist
```



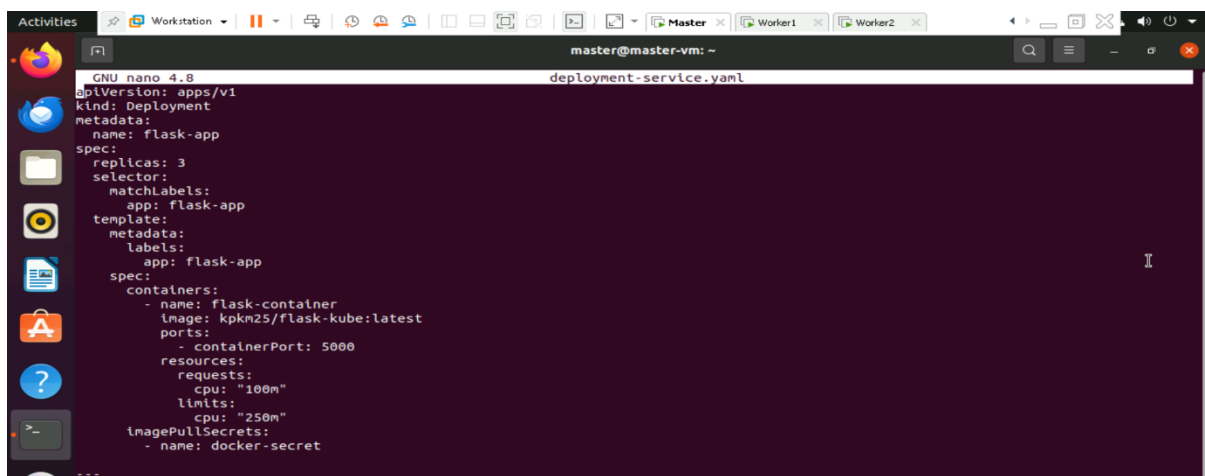
```
Successfully built 441a3b5f968a
Successfully tagged ankitamohanty1509/flask-kube:latest
master@master-vm:~$ docker push ankitamohanty1509/flask-kube
"docker push" requires exactly 1 argument.
See 'docker push --help'.

Usage:  docker push [OPTIONS] NAME[:TAG]

Upload an image to a registry
master@master-vm:~$ docker push ankitamohanty1509/flask-kube
Using default tag: latest
The push refers to repository [docker.io/ankitamohanty1509/flask-kube]
bd071f0a1f28: Pushed
21e1f7212650: Pushed
40765d688aa4: Pushed
b723da6e1cf4: Mounted from library/python
7af6b2a8a1a8: Mounted from library/python
71030c5d3283: Mounted from library/python
4b017a36fd9c: Mounted from library/python
20a9b386e10e: Mounted from library/python
f8217d7865d2: Mounted from library/python
01c9a2a5f237: Mounted from library/python
Head "https://registry-1.docker.io/v2/ankitamohanty1509/flask-kube/blobs/sha256:441a3b5f968a50244a4b6da39683cc635707546682cdcfdcd8ffb27529ea32dd": net/http: TLS handshake timeout
```

## 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: kpkm25/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: ~
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:system: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-vm:~$ 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://192.168.147.129:3227 1; 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-mtpdd         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

