Kubernetes Project - 03

DEPLOY A MULTI-TIER WEB APPLICATION ON KUBERNETES

1. Setup Kubernetes Cluster

If we don't have a Kubernetes cluster, set up Minikube:

minikube start

kubectl cluster-info

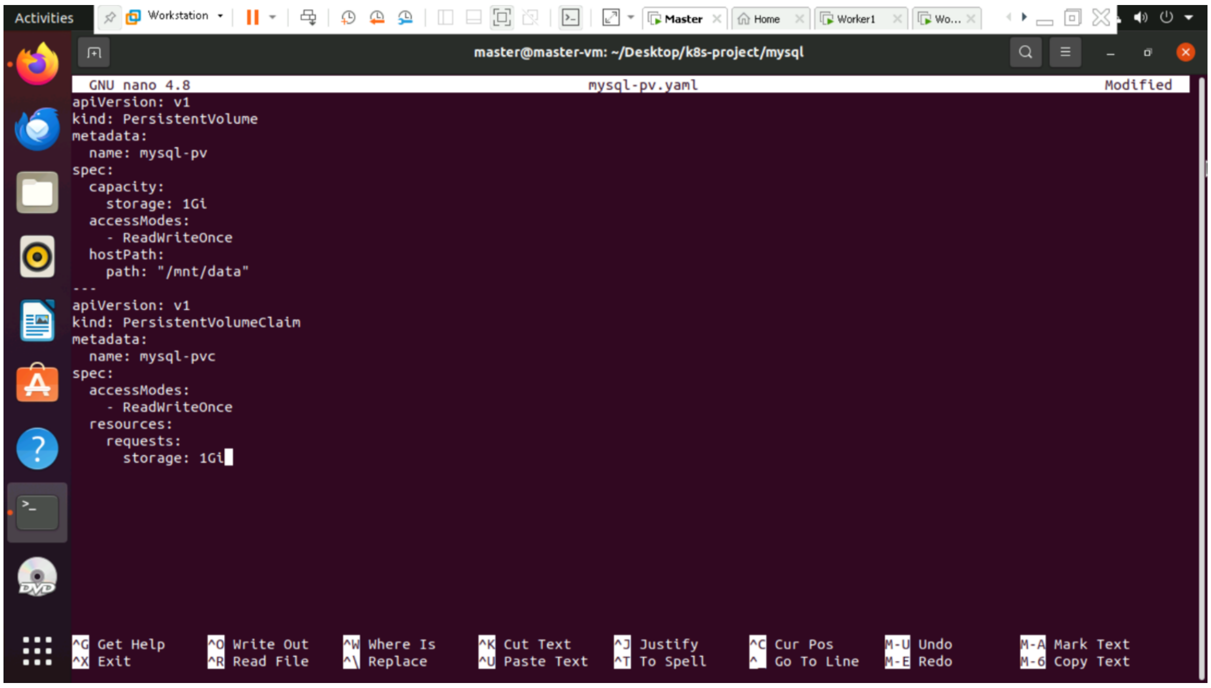
kubectl get nodes

---

2. Deploy MySQL Database (StatefulSet)

Step 1: Create a Persistent Volume for MySQL

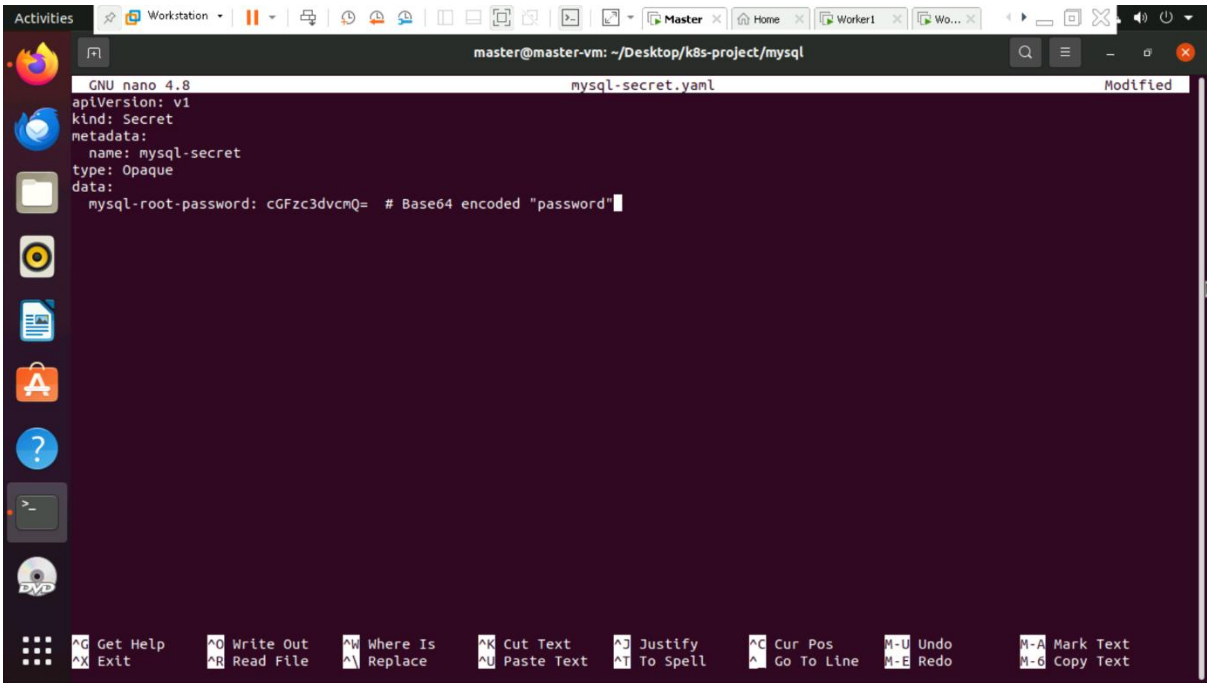
# mysql-pv.yaml



```
  GNU nano 4.8                          mysql-pv.yaml                          Modified
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Step 2: Create MySQL Secret for Password Storage

# mysql-secret.yaml

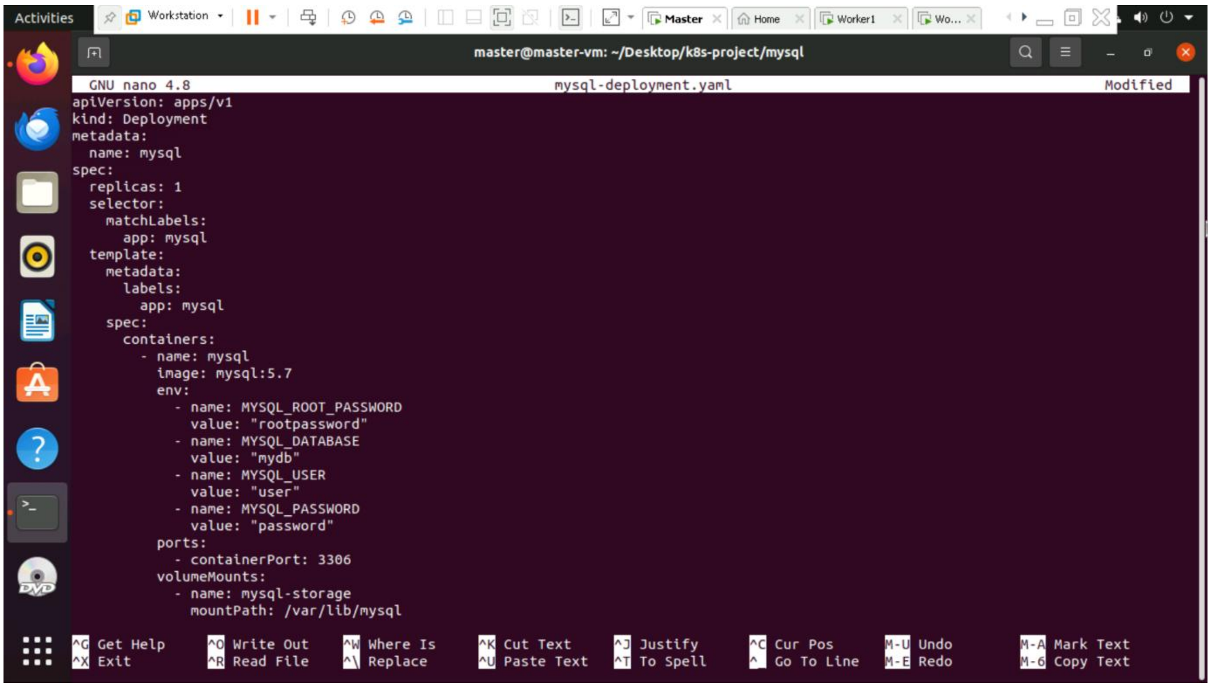


```
  GNU nano 4.8                         mysql-secret.yaml                         Modified
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
type: Opaque
data:
  mysql-root-password: cGFzc3dvcmQ=  # Base64 encoded "password"
```

```
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Step 3: Deploy MySQL as a StatefulSet

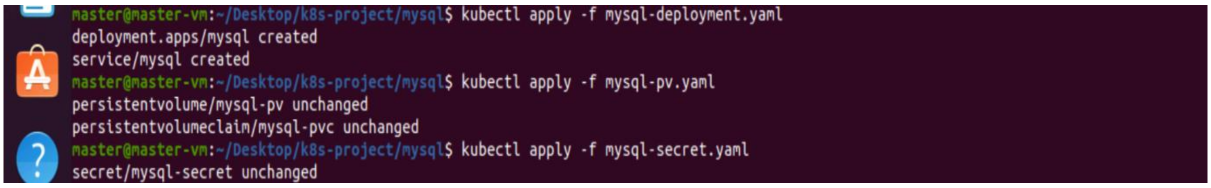
# mysql-deployment.yaml



```
  GNU nano 4.8                       mysql-deployment.yaml                       Modified
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql:5.7
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: "rootpassword"
            - name: MYSQL_DATABASE
              value: "mydb"
            - name: MYSQL_USER
              value: "user"
            - name: MYSQL_PASSWORD
              value: "password"
          ports:
            - containerPort: 3306
          volumeMounts:
            - name: mysql-storage
              mountPath: /var/lib/mysql
```

```
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

- **kubectl apply -f mysql-deployment.yaml**
- **kubectl apply -f mysql-pv.yaml**
- **kubectl apply -f mysql-secret.yaml**



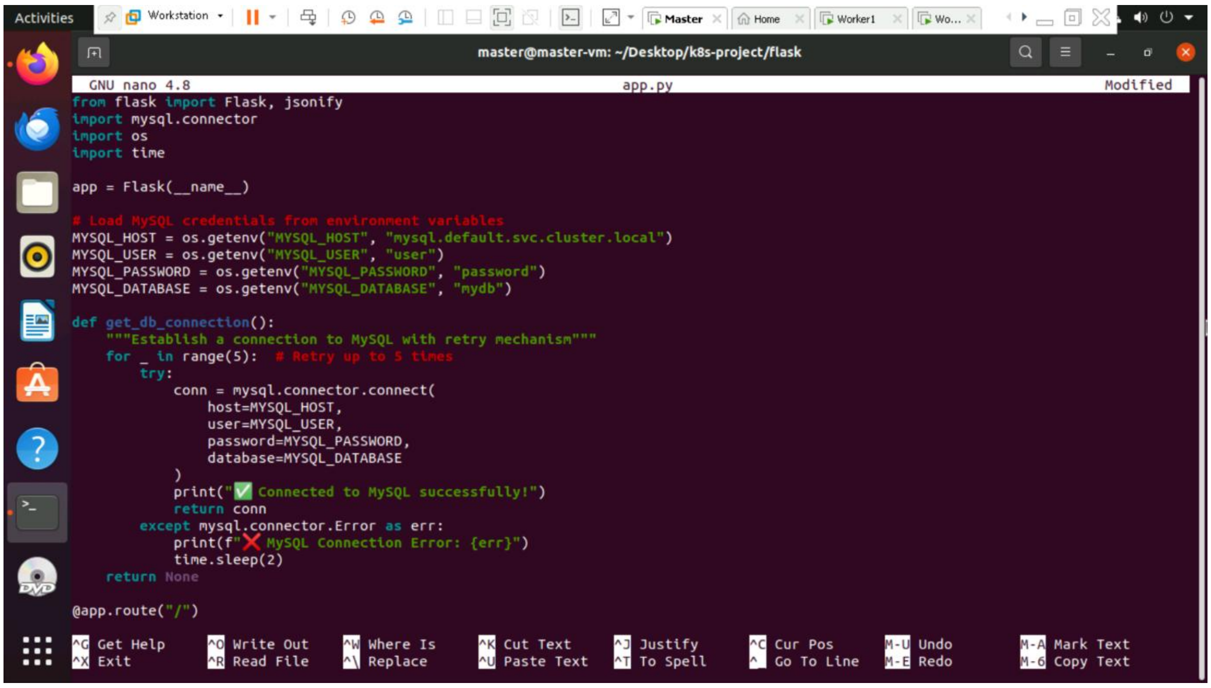
```
master@master-vm:~/Desktop/k8s-project/mysql$ kubectl apply -f mysql-deployment.yaml
deployment.apps/mysql created
service/mysql created
master@master-vm:~/Desktop/k8s-project/mysql$ kubectl apply -f mysql-pv.yaml
persistentvolume/mysql-pv unchanged
persistentvolumeclaim/mysql-pvc unchanged
master@master-vm:~/Desktop/k8s-project/mysql$ kubectl apply -f mysql-secret.yaml
secret/mysql-secret unchanged
```

---

3. Deploy Flask Backend

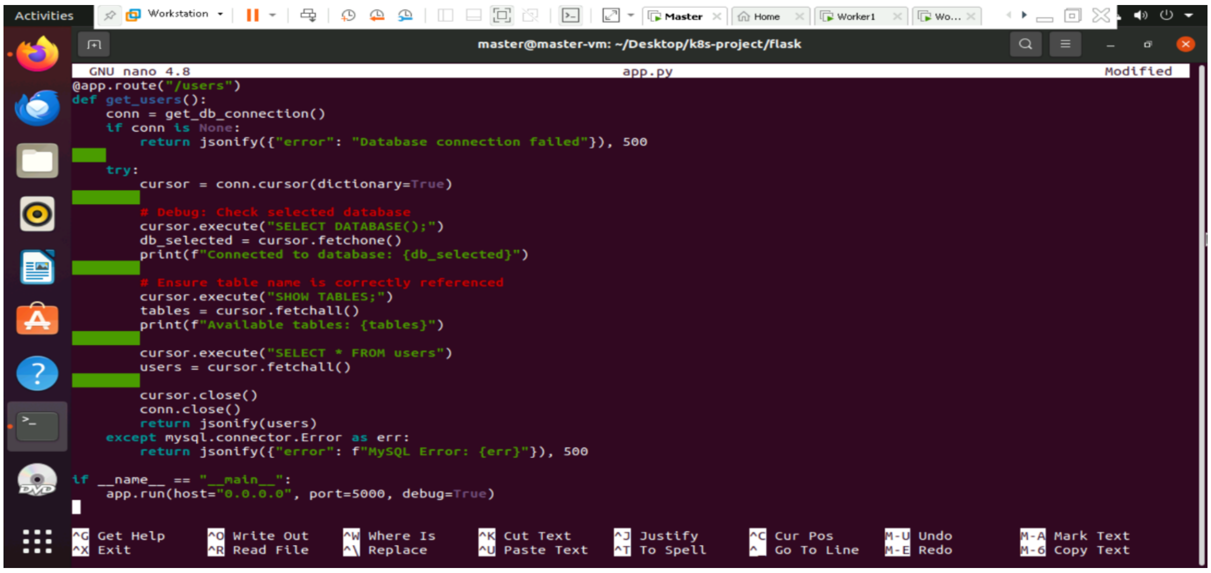
Step 1: Create a Flask App (Dockerized)

# app.py



master@master-vm: ~/Desktop/k8s-project/flask

```
  GNU nano 4.8                          app.py                          Modified
from flask import Flask, jsonify
import mysql.connector
import os
import time

app = Flask(__name__)

# Load MySQL credentials from environment variables
MYSQL_HOST = os.getenv("MYSQL_HOST", "mysql.default.svc.cluster.local")
MYSQL_USER = os.getenv("MYSQL_USER", "user")
MYSQL_PASSWORD = os.getenv("MYSQL_PASSWORD", "password")
MYSQL_DATABASE = os.getenv("MYSQL_DATABASE", "mydb")

def get_db_connection():
    """Establish a connection to MySQL with retry mechanism"""
    for _ in range(5):  # Retry up to 5 times
        try:
            conn = mysql.connector.connect(
                host=MYSQL_HOST,
                user=MYSQL_USER,
                password=MYSQL_PASSWORD,
                database=MYSQL_DATABASE
            )
            print("✅ Connected to MySQL successfully!")
            return conn
        except mysql.connector.Error as err:
            print(f"❌ MySQL Connection Error: {err}")
            time.sleep(2)
    return None

@app.route("/")

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

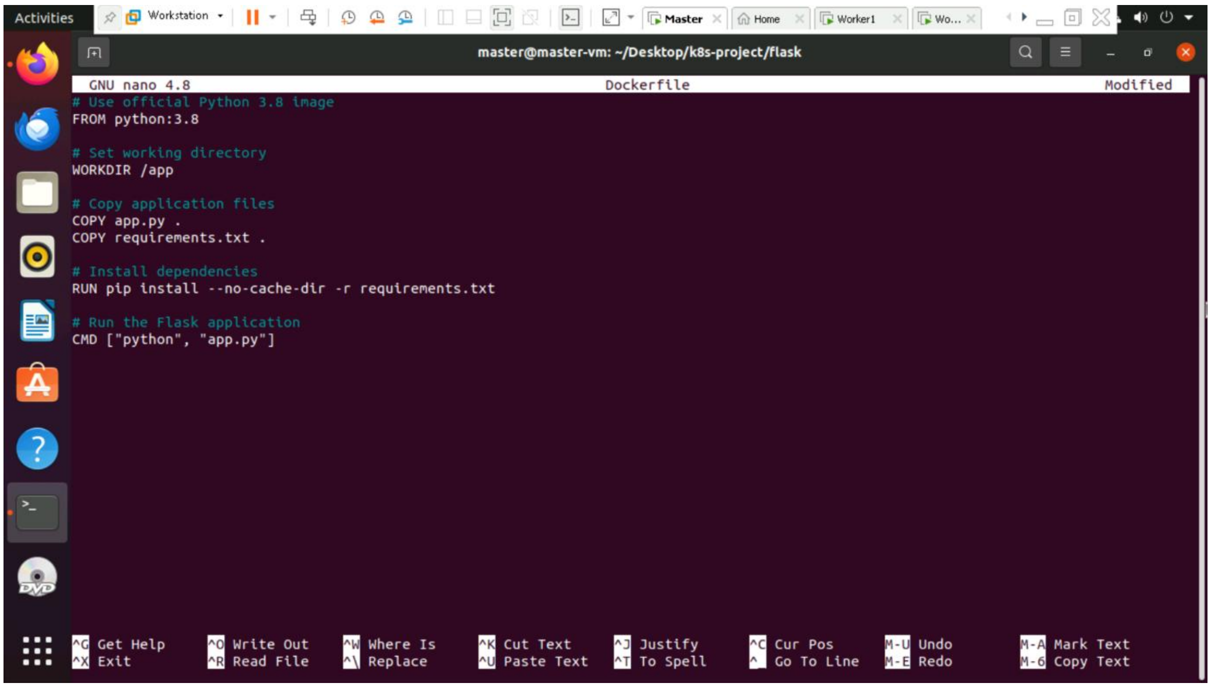


master@master-vm: ~/Desktop/k8s-project/flask

```
  GNU nano 4.8                          app.py                          Modified
@app.route("/users")
def get_users():
    conn = get_db_connection()
    if conn is None:
        return jsonify({"error": "Database connection failed"}), 500

    try:
        cursor = conn.cursor(dictionary=True)

        # Debug: Check selected database
        cursor.execute("SELECT DATABASE();")
        db_selected = cursor.fetchone()
        print(f"Connected to database: {db_selected}")

        # Ensure table name is correctly referenced
        cursor.execute("SHOW TABLES;")
        tables = cursor.fetchall()
        print(f"Available tables: {tables}")

        cursor.execute("SELECT * FROM users")
        users = cursor.fetchall()

        cursor.close()
        conn.close()
        return jsonify(users)
    except mysql.connector.Error as err:
        return jsonify({"error": f"MySQL Error: {err}"}), 500

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Step 2: Create a Dockerfile

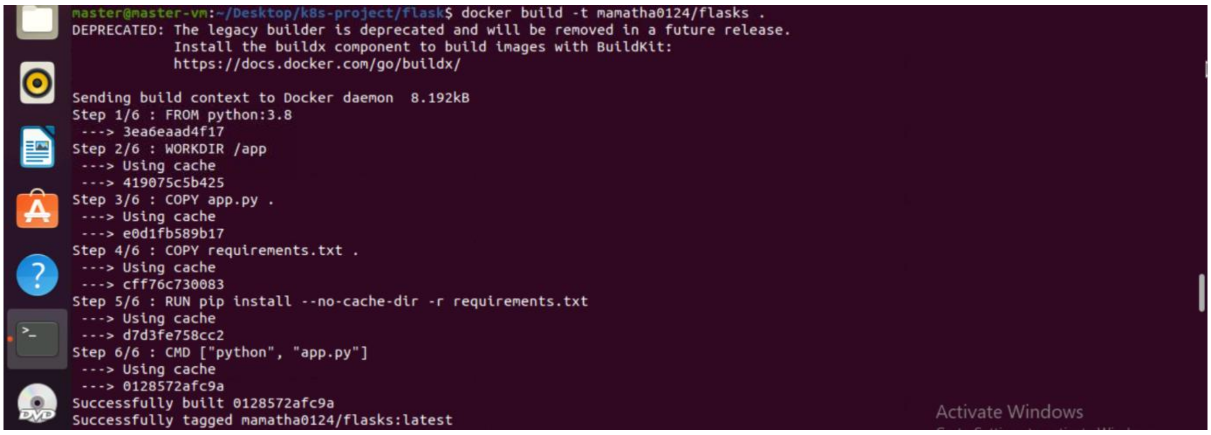
# Dockerfile



```
  GNU nano 4.8                           Dockerfile                           Modified
# Use official Python 3.8 image
FROM python:3.8

# Set working directory
WORKDIR /app

# Copy application files
COPY app.py .
COPY requirements.txt .

# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Run the Flask application
CMD ["python", "app.py"]
```

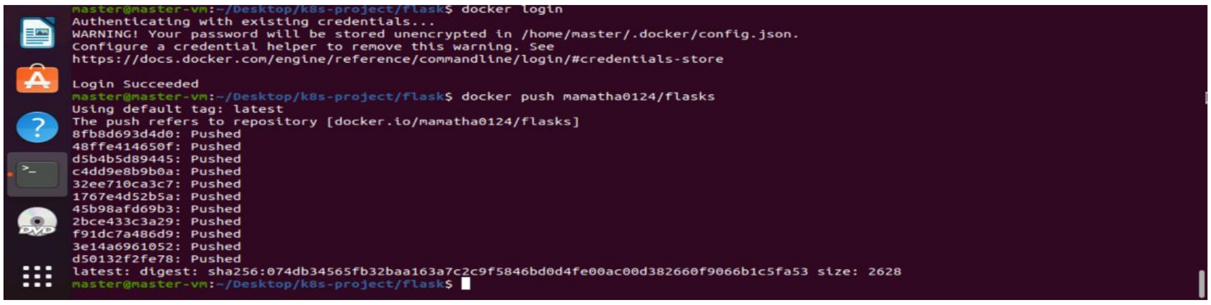
Build & push Docker image:

docker build -t mamatha0124/flasks



```
master@master-vm:~/Desktop/k8s-project/flask$ docker build -t mamatha0124/flasks .
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  8.192kB
Step 1/6 : FROM python:3.8
 ---> 3ea6eaad4f17
Step 2/6 : WORKDIR /app
 ---> Using cache
 ---> 419075c5b425
Step 3/6 : COPY app.py .
 ---> Using cache
 ---> e0d1fb589b17
Step 4/6 : COPY requirements.txt .
 ---> Using cache
 ---> cff76c730083
Step 5/6 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Using cache
 ---> d7d3fe758cc2
Step 6/6 : CMD ["python", "app.py"]
 ---> Using cache
 ---> 0128572afc9a
Successfully built 0128572afc9a
Successfully tagged mamatha0124/flasks:latest
```

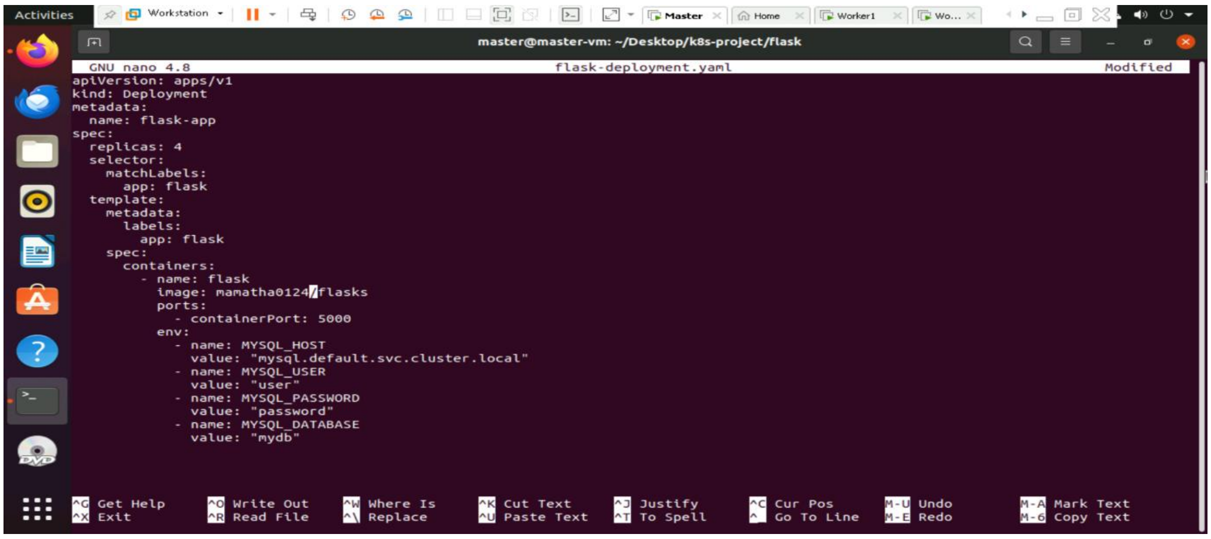
docker push mamatha0124/flasks



```
master@master-vm:~/Desktop/k8s-project/flask$ docker login
Authenticating with existing credentials...
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:~/Desktop/k8s-project/flask$ docker push mamatha0124/flasks
Using default tag: latest
The push refers to repository [docker.io/mamatha0124/flasks]
8fb8d693d4d0: Pushed
48ffe414650f: Pushed
d5b4b5d89445: Pushed
c4dd9e8b9b0a: Pushed
32ee710ca3c7: Pushed
1767e4d52b5a: Pushed
45b98afd69b3: Pushed
2bce433c3a29: Pushed
f91dc7a486d9: Pushed
3e14a6961052: Pushed
d50132f2fe78: Pushed
latest: digest: sha256:074db34565fb32baa163a7c2c9f5846bd0d4fe00ac00d382660f9066b1c5fa53 size: 2628
master@master-vm:~/Desktop/k8s-project/flask$
```

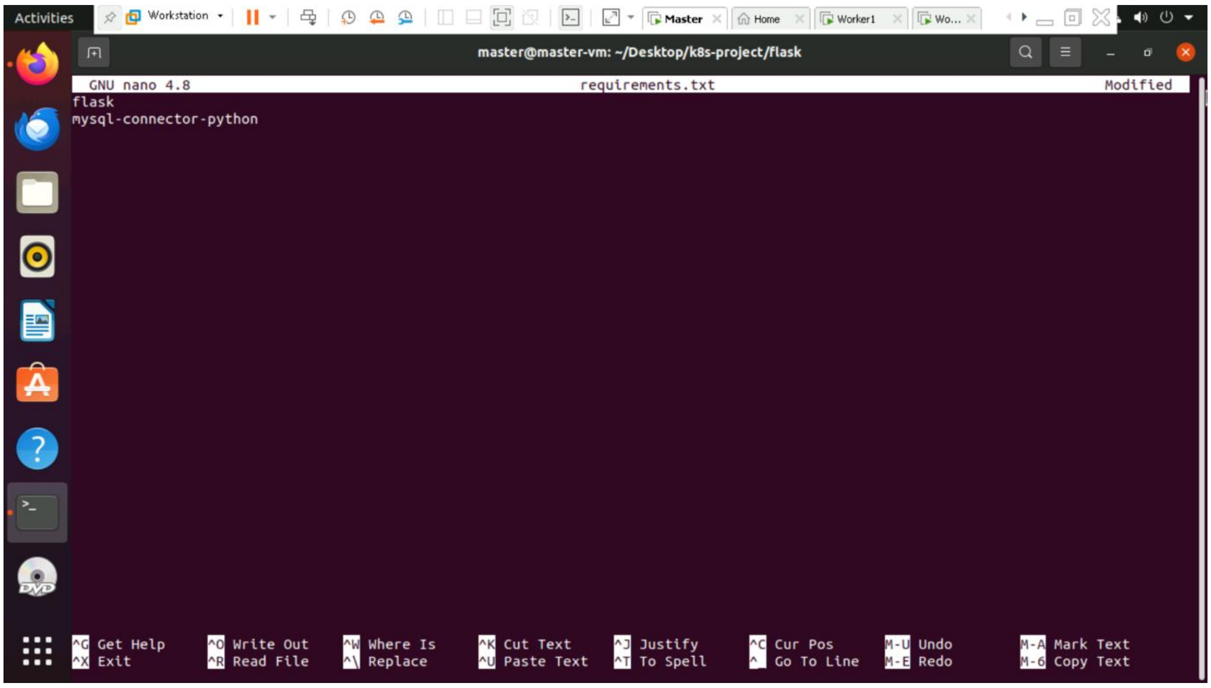
Step 3: Deploy Flask App on Kubernetes

# flask-deployment.yaml



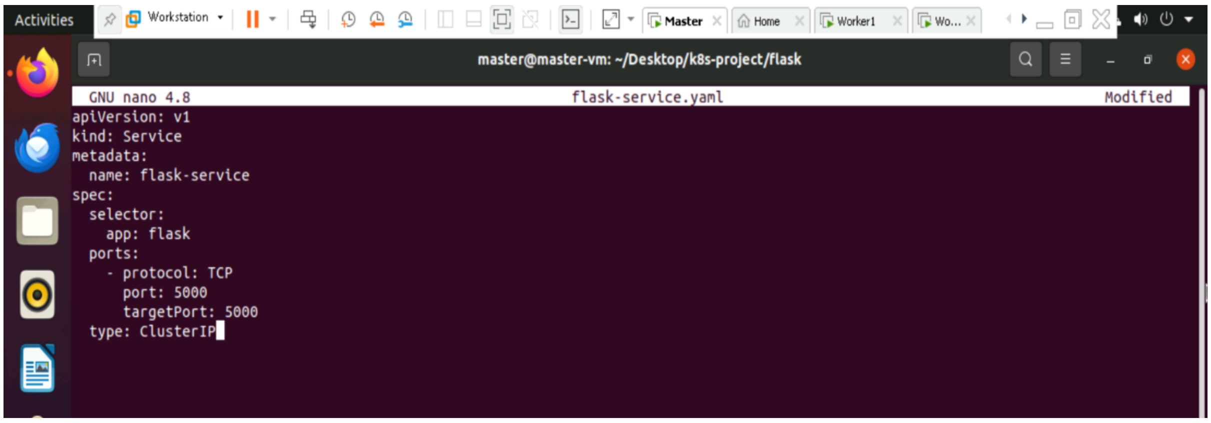
```
master@master-vm: ~/Desktop/k8s-project/flask

  GNU nano 4.8                    flask-deployment.yaml                    Modified
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
spec:
  replicas: 4
  selector:
    matchLabels:
      app: flask
  template:
    metadata:
      labels:
        app: flask
    spec:
      containers:
        - name: flask
          image: mamatha0124/flasks
          ports:
            - containerPort: 5000
          env:
            - name: MYSQL_HOST
              value: "mysql.default.svc.cluster.local"
            - name: MYSQL_USER
              value: "user"
            - name: MYSQL_PASSWORD
              value: "password"
            - name: MYSQL_DATABASE
              value: "mydb"

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Step 4: Create a requirements.txt file



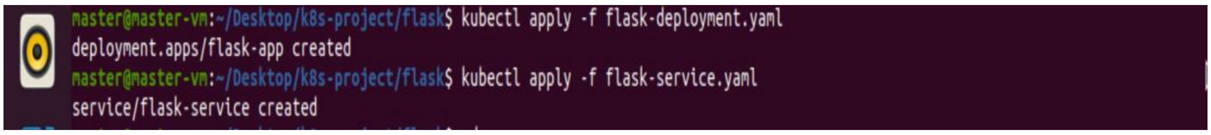
```
master@master-vm: ~/Desktop/k8s-project/flask

  GNU nano 4.8                    requirements.txt                    Modified
flask
mysql-connector-python

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Step 5: Create Flask Service file in Kubernetes



```
  GNU nano 4.8                     flask-service.yaml                     Modified
apiVersion: v1
kind: Service
metadata:
  name: flask-service
spec:
  selector:
    app: flask
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
  type: ClusterIP
```

```
kubectl apply -f flask-deployment.yaml

kubectl apply -f nginx-service.yaml
```

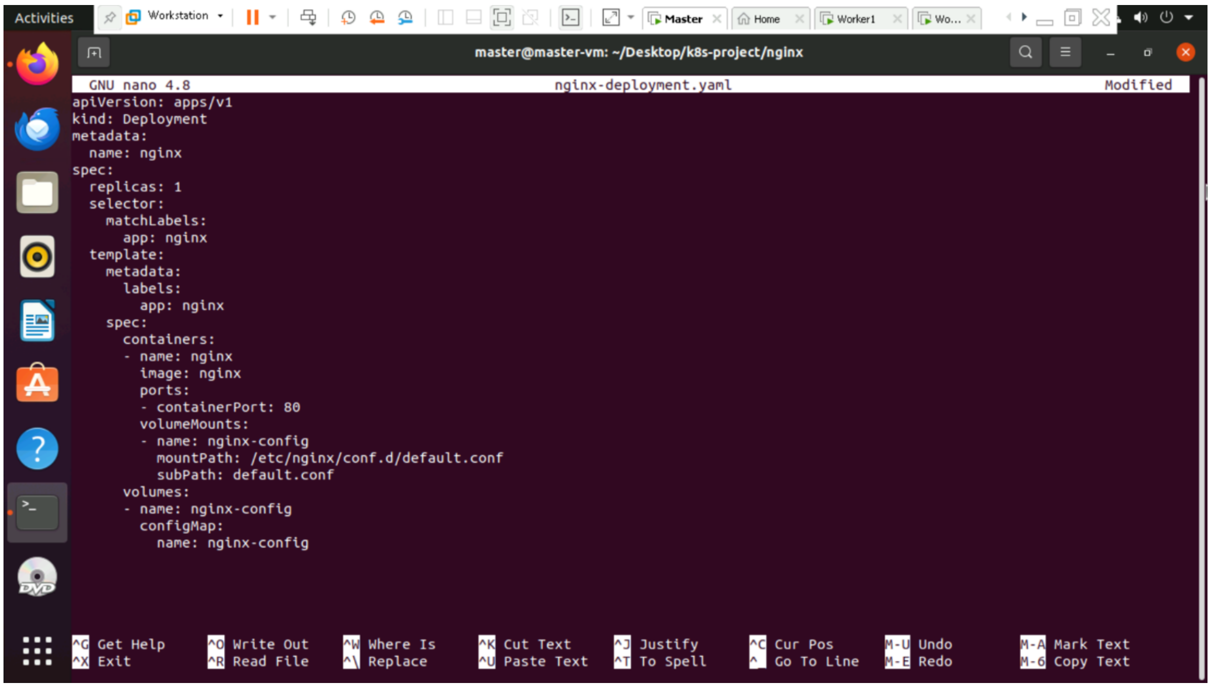


```
master@master-vm:~/Desktop/k8s-project/flask$ kubectl apply -f flask-deployment.yaml
deployment.apps/flask-app created
master@master-vm:~/Desktop/k8s-project/flask$ kubectl apply -f flask-service.yaml
service/flask-service created
```

---

4. Deploy Nginx as Frontend

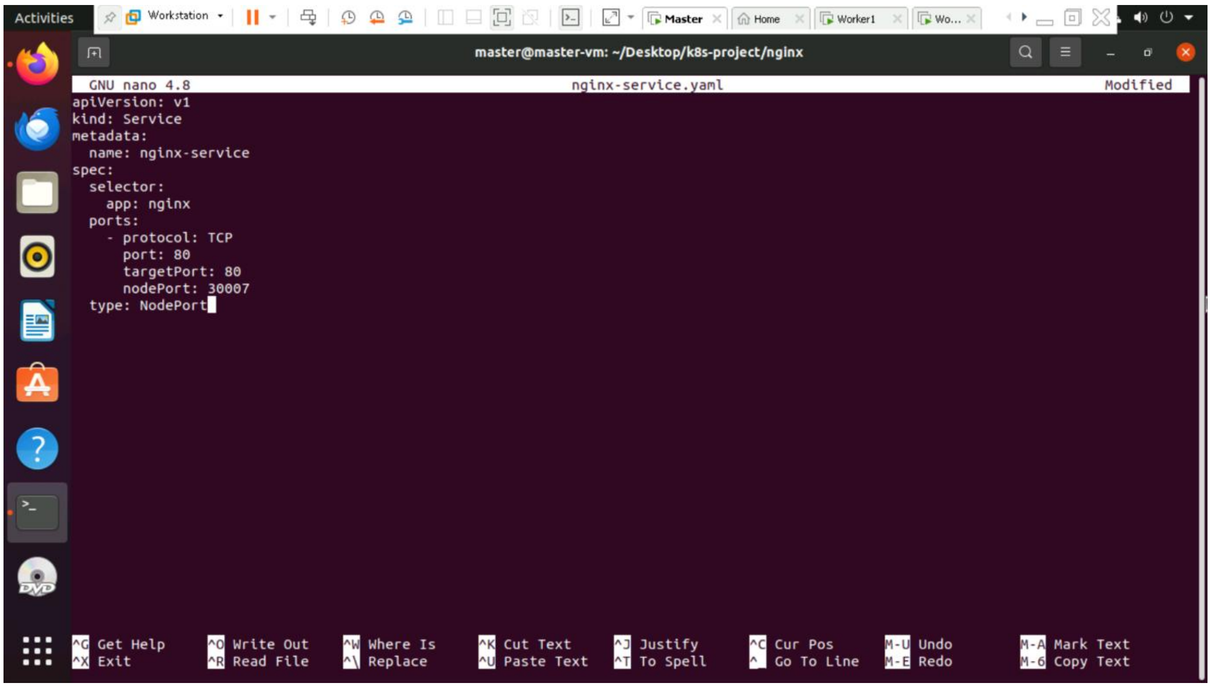
# nginx-deployment.yaml



```
  GNU nano 4.8                     nginx-deployment.yaml                     Modified
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
        volumeMounts:
        - name: nginx-config
          mountPath: /etc/nginx/conf.d/default.conf
          subPath: default.conf
      volumes:
      - name: nginx-config
        configMap:
          name: nginx-config

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo    M-A Mark Text
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo    M-6 Copy Text
```

Expose Nginx via NodePort:

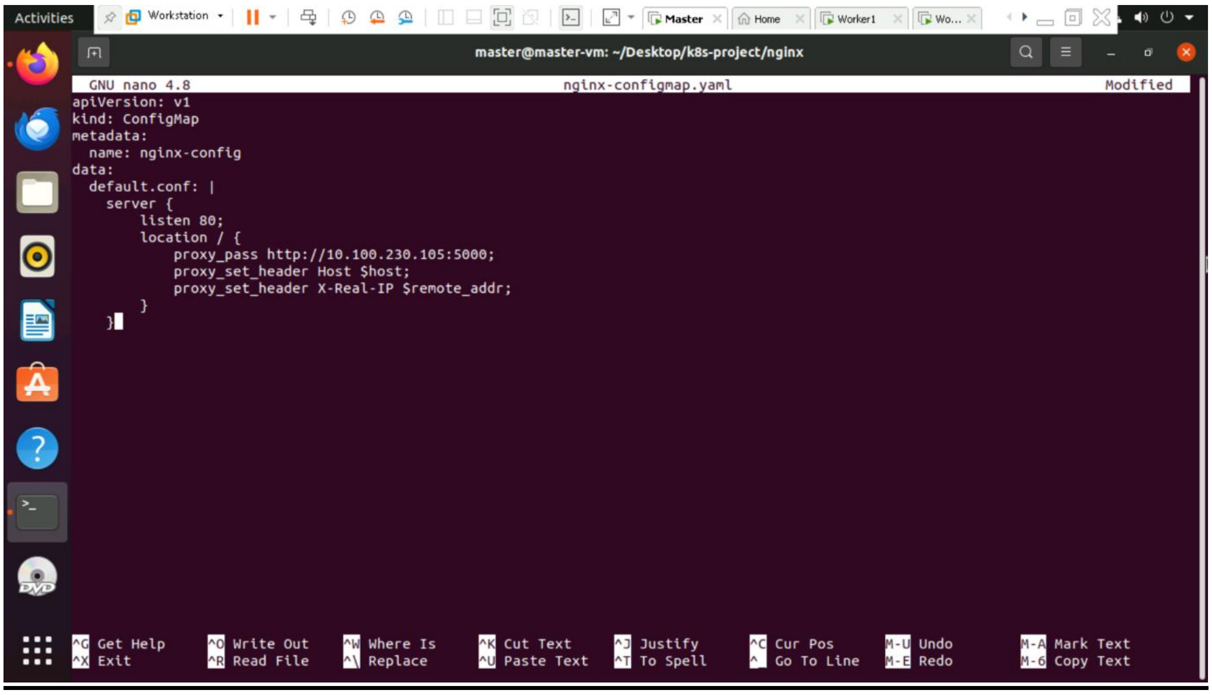
# nginx-service.yaml



```
  GNU nano 4.8                    nginx-service.yaml                    Modified
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30007
  type: NodePort
```

Activities
Workstation
Master
Home
Worker1
Wo...
master@master-vm: ~/Desktop/k8s-project/nginx
^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos   M-U Undo   M-A Mark Text
^X Exit   ^R Read File   ^\ Replace   ^U Paste Text   ^T To Spell   ^_ Go To Line   M-E Redo   M-6 Copy Text

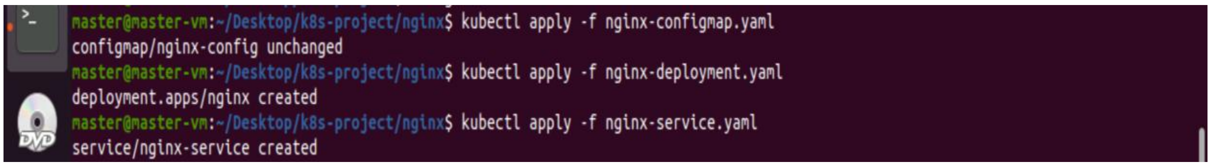
Create a Nginx Configmap file



```
  GNU nano 4.8                    nginx-configmap.yaml                    Modified
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-config
data:
  default.conf: |
    server {
        listen 80;
        location / {
            proxy_pass http://10.100.230.105:5000;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }
    }
```

Activities
Workstation
Master
Home
Worker1
Wo...
master@master-vm: ~/Desktop/k8s-project/nginx
^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos   M-U Undo   M-A Mark Text
^X Exit   ^R Read File   ^\ Replace   ^U Paste Text   ^T To Spell   ^_ Go To Line   M-E Redo   M-6 Copy Text

```
- kubectl apply -f nginx-configmap.yaml
- kubectl apply -f nginx-deployment.yaml
- kubectl apply -f nginx-service.yaml
```



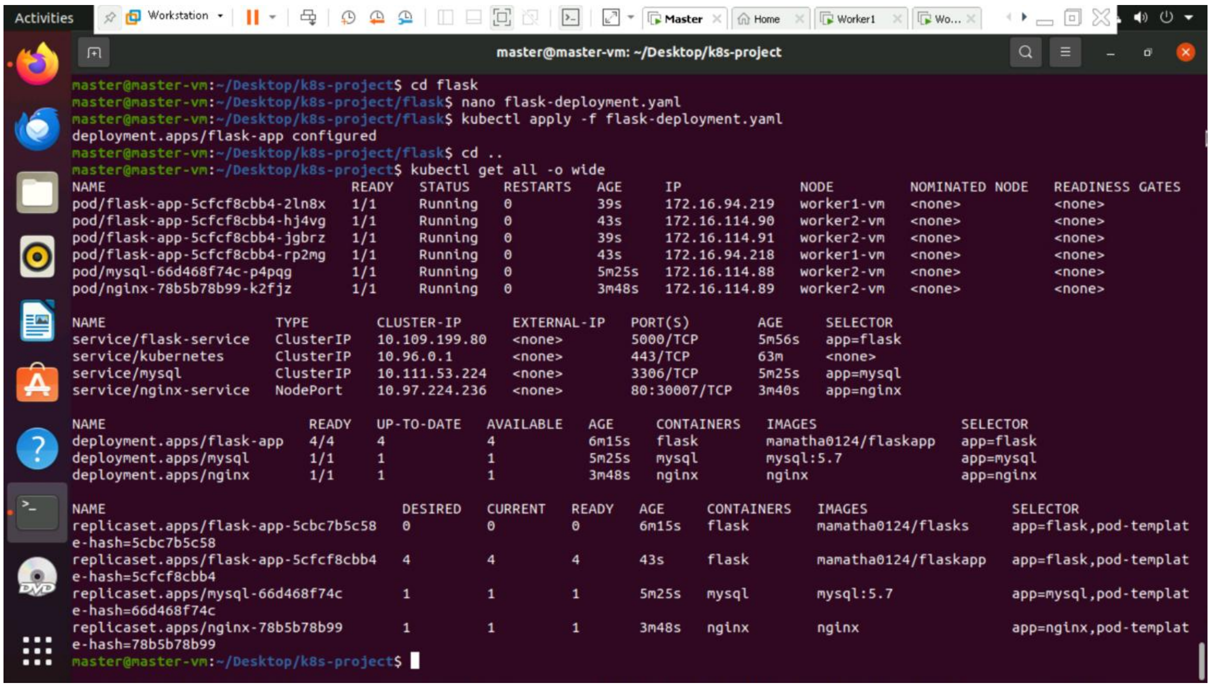
```
master@master-vm:~/Desktop/k8s-project/nginx$ kubectl apply -f nginx-configmap.yaml
configmap/nginx-config unchanged
master@master-vm:~/Desktop/k8s-project/nginx$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx created
master@master-vm:~/Desktop/k8s-project/nginx$ kubectl apply -f nginx-service.yaml
service/nginx-service created
```

---

5. Verify and Test Application

Check Kubernetes pods,deployments,services running status

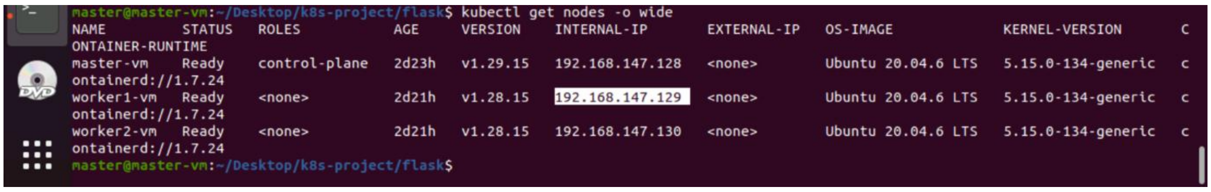
- kubectl get all -o wide



```
master@master-vm:~/Desktop/k8s-project$ cd flask
master@master-vm:~/Desktop/k8s-project/flask$ nano flask-deployment.yaml
master@master-vm:~/Desktop/k8s-project/flask$ kubectl apply -f flask-deployment.yaml
deployment.apps/flask-app configured
master@master-vm:~/Desktop/k8s-project/flask$ cd ..
master@master-vm:~/Desktop/k8s-project$ kubectl get all -o wide
NAME                             READY   STATUS    RESTARTS   AGE     IP              NODE         NOMINATED NODE   READINESS GATES
pod/flask-app-5cfcf8cbb4-2ln8x   1/1     Running   0          39s     172.16.94.219   worker1-vm   <none>           <none>
pod/flask-app-5cfcf8cbb4-hj4vg   1/1     Running   0          43s     172.16.114.90   worker2-vm   <none>           <none>
pod/flask-app-5cfcf8cbb4-jgbrz   1/1     Running   0          39s     172.16.114.91   worker2-vm   <none>           <none>
pod/flask-app-5cfcf8cbb4-rp2mg   1/1     Running   0          43s     172.16.94.218   worker1-vm   <none>           <none>
pod/mysql-66d468f74c-p4pqg       1/1     Running   0          5m25s   172.16.114.88   worker2-vm   <none>           <none>
pod/nginx-78b5b78b99-k2fjz       1/1     Running   0          3m48s   172.16.114.89   worker2-vm   <none>           <none>

NAME                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE     SELECTOR
service/flask-service   ClusterIP   10.109.199.80   <none>        5000/TCP       5m56s   app=flask
service/kubernetes      ClusterIP   10.96.0.1       <none>        443/TCP        63m     <none>
service/mysql           ClusterIP   10.111.53.224   <none>        3306/TCP       5m25s   app=mysql
service/nginx-service   NodePort    10.97.224.236   <none>        80:30007/TCP   3m40s   app=nginx

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES                SELECTOR
deployment.apps/flask-app   4/4     4            4           6m15s   flask        mamatha0124/flaskapp   app=flask
deployment.apps/mysql       1/1     1            1           5m25s   mysql        mysql:5.7             app=mysql
deployment.apps/nginx       1/1     1            1           3m48s   nginx        nginx                 app=nginx

NAME                                   DESIRED   CURRENT   READY   AGE     CONTAINERS   IMAGES                 SELECTOR
replicaset.apps/flask-app-5cbc7b5c58   0         0         0       6m15s   flask        mamatha0124/flasks     app=flask,pod-templat
e-hash=5cbc7b5c58
replicaset.apps/flask-app-5cfcf8cbb4   4         4         4       43s     flask        mamatha0124/flaskapp   app=flask,pod-templat
e-hash=5cfcf8cbb4
replicaset.apps/mysql-66d468f74c       1         1         1       5m25s   mysql        mysql:5.7              app=mysql,pod-templat
e-hash=66d468f74c
replicaset.apps/nginx-78b5b78b99       1         1         1       3m48s   nginx        nginx                  app=nginx,pod-templat
e-hash=78b5b78b99
master@master-vm:~/Desktop/k8s-project$ 
```

To access the application:

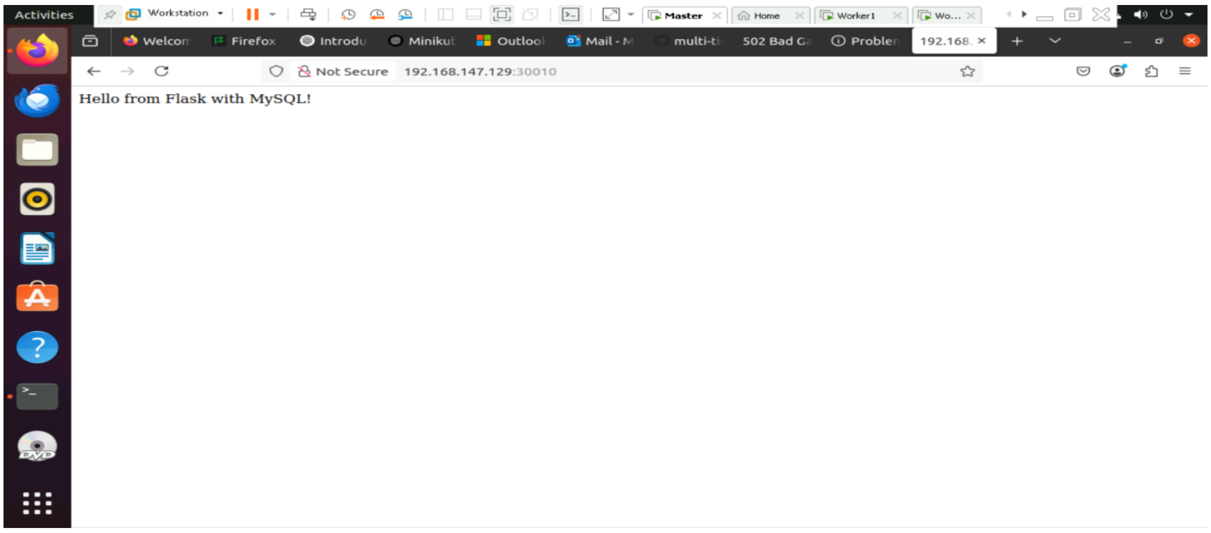
- **Kubectl get nodes**



```
master@master-vm:~/Desktop/k8s-project/flask$ kubectl get nodes -o wide
NAME         STATUS   ROLES           AGE     VERSION    INTERNAL-IP       EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       C
ONTAINER-RUNTIME
master-vm    Ready    control-plane   2d23h   v1.29.15   192.168.147.128   <none>        Ubuntu 20.04.6 LTS   5.15.0-134-generic   c
ontainerd://1.7.24
worker1-vm   Ready    <none>          2d21h   v1.28.15   192.168.147.129   <none>        Ubuntu 20.04.6 LTS   5.15.0-134-generic   c
ontainerd://1.7.24
worker2-vm   Ready    <none>          2d21h   v1.28.15   192.168.147.130   <none>        Ubuntu 20.04.6 LTS   5.15.0-134-generic   c
ontainerd://1.7.24
master@master-vm:~/Desktop/k8s-project/flask$
```

Copy worker 1 IP or worker 2 IP and in browser enter

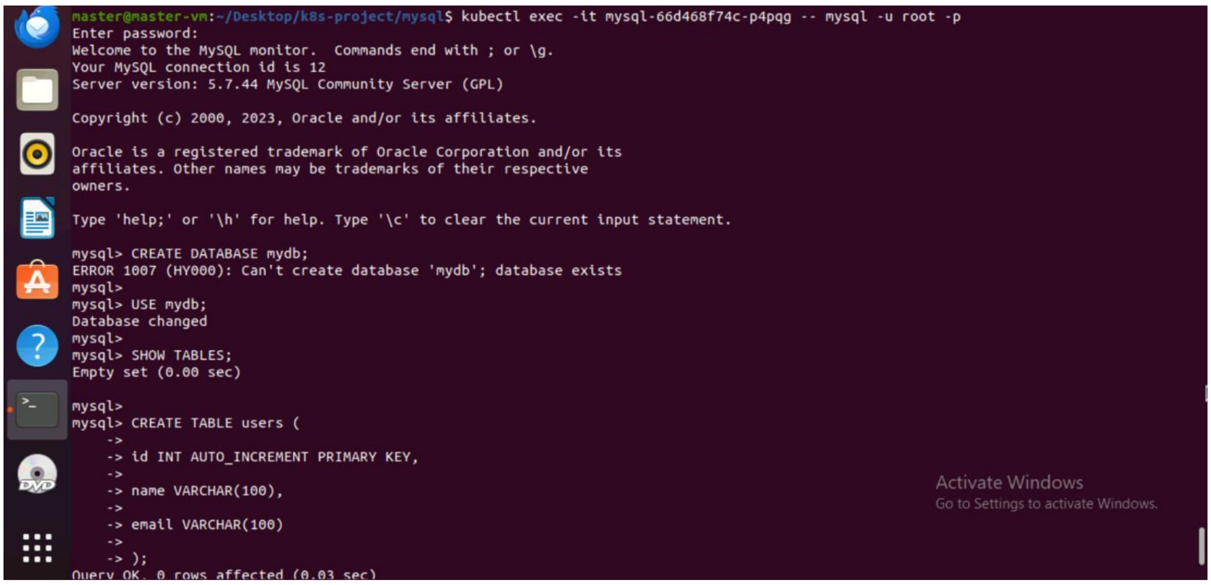
http://192.168.147.129:300010/



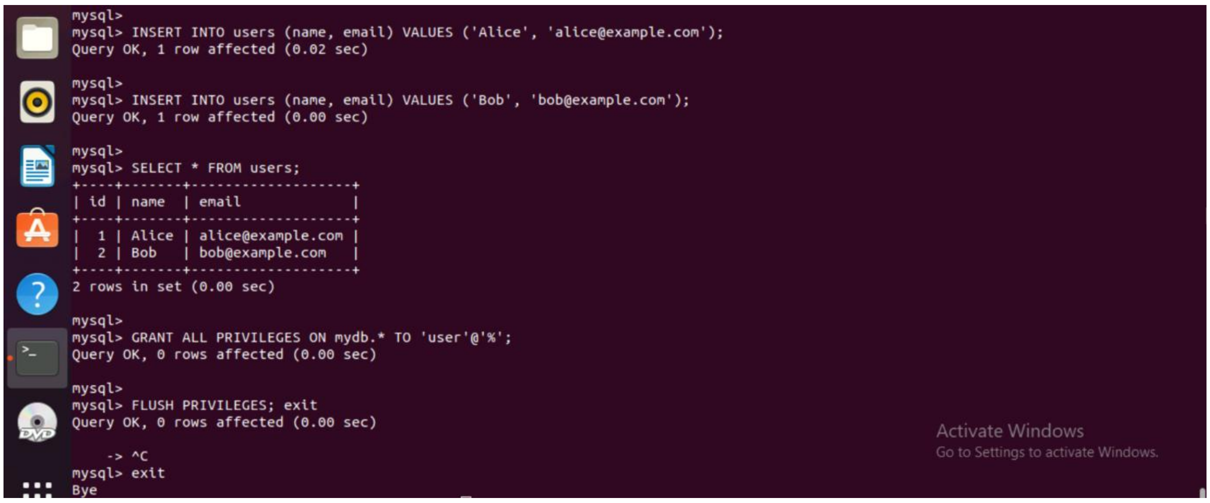
Hello from Flask with MySQL!

Access MySQL Inside the Pod and add the data

kubectl exec -it mysql-0 -- mysql -u root -p

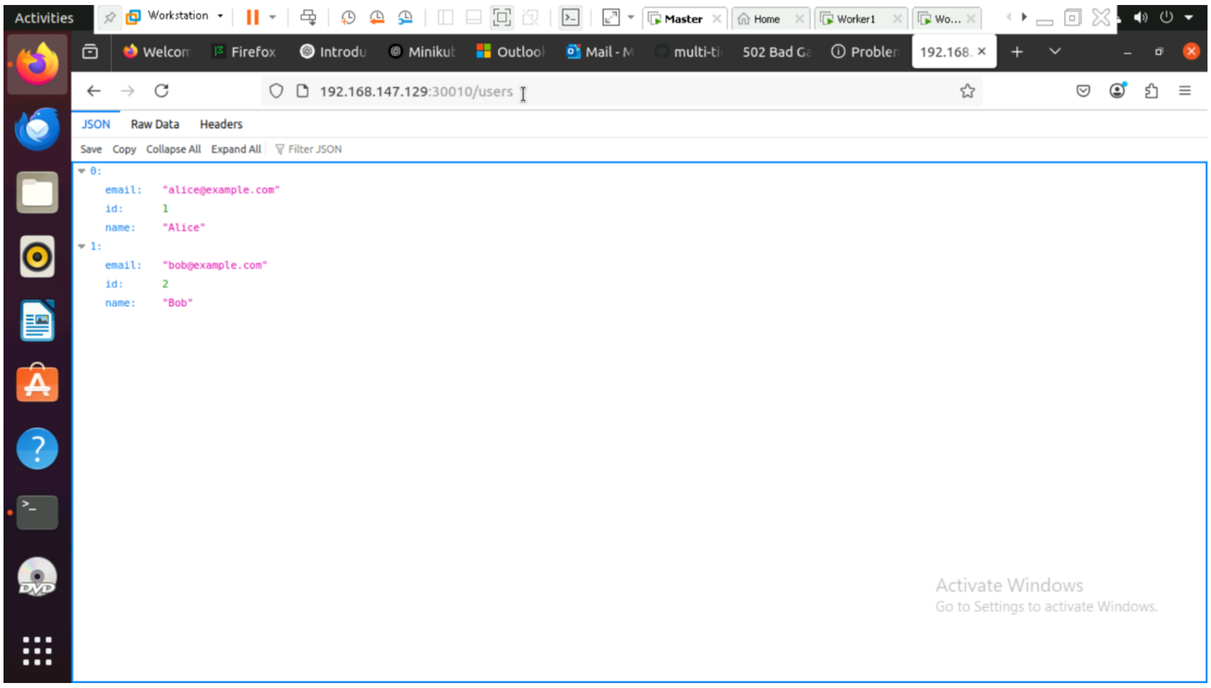


```
master@master-vm:~/Desktop/k8s-project/mysql$ kubectl exec -it mysql-66d468f74c-p4pqg -- mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.7.44 MySQL Community Server (GPL)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE mydb;
ERROR 1007 (HY000): Can't create database 'mydb'; database exists
mysql>
mysql> USE mydb;
Database changed
mysql>
mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql>
mysql> CREATE TABLE users (
    ->
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    ->
    -> name VARCHAR(100),
    ->
    -> email VARCHAR(100)
    ->
    -> );
Query OK, 0 rows affected (0.03 sec)
```



```
mysql>
mysql> INSERT INTO users (name, email) VALUES ('Alice', 'alice@example.com');
Query OK, 1 row affected (0.02 sec)

mysql>
mysql> INSERT INTO users (name, email) VALUES ('Bob', 'bob@example.com');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> SELECT * FROM users;
+----+-------+-------------------+
| id | name  | email             |
+----+-------+-------------------+
|  1 | Alice | alice@example.com |
|  2 | Bob   | bob@example.com   |
+----+-------+-------------------+
2 rows in set (0.00 sec)

mysql>
mysql> GRANT ALL PRIVILEGES ON mydb.* TO 'user'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> FLUSH PRIVILEGES; exit
Query OK, 0 rows affected (0.00 sec)

    -> ^C
mysql> exit
Bye
```

To access users http://192.168.147.129:300010/users



```
0:
  email:  "alice@example.com"
  id:     1
  name:   "Alice"
1:
  email:  "bob@example.com"
  id:     2
  name:   "Bob"
```