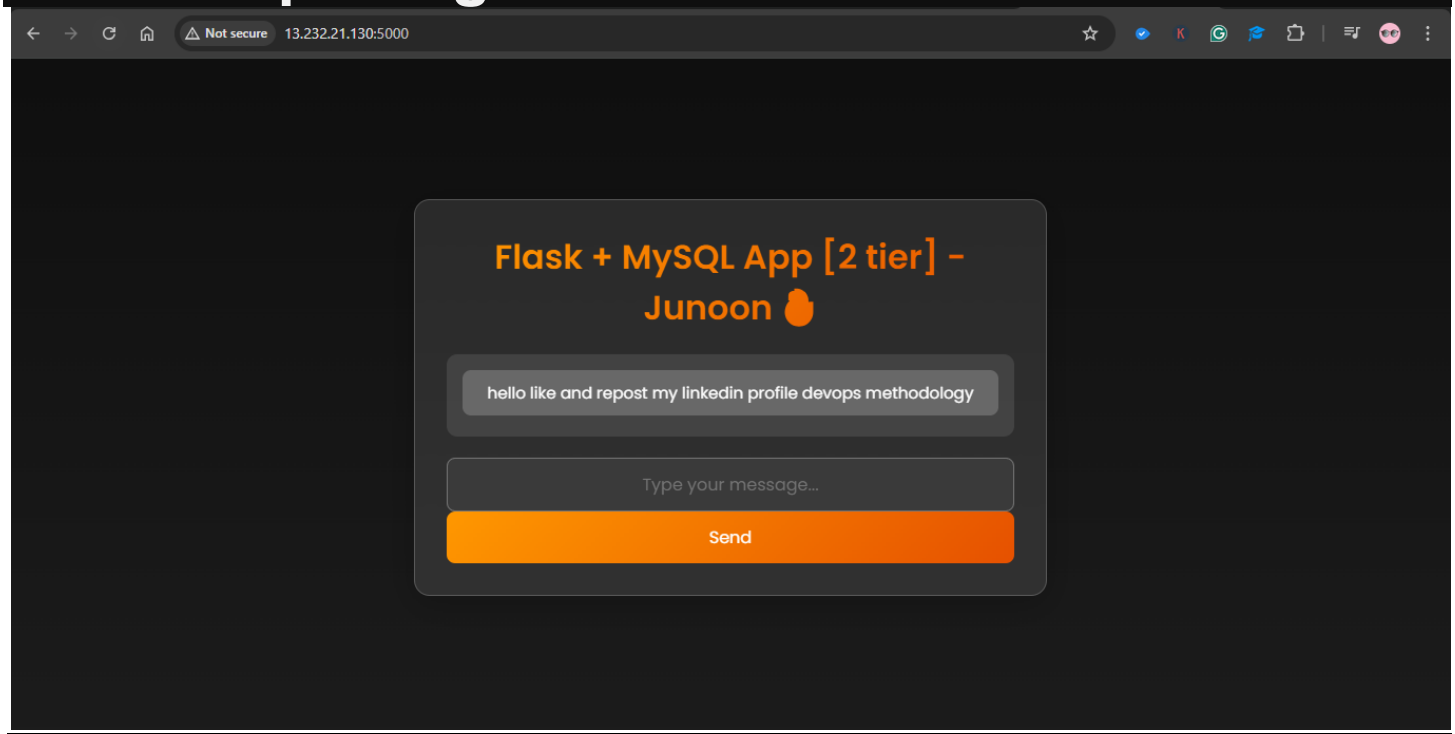
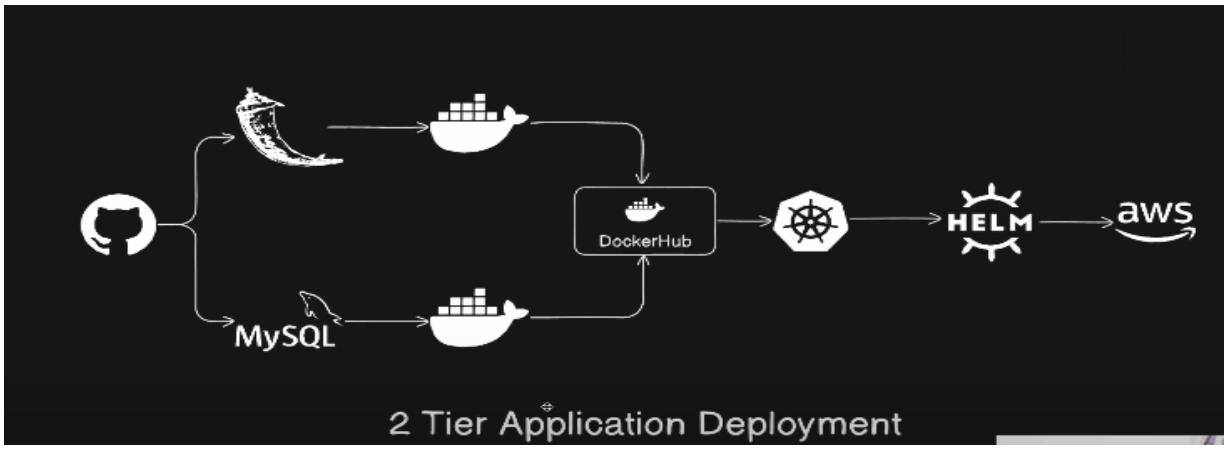
2-Tier Application Deployment Project Series for DevOps Engineers



EP-1

INTRODUCTION TO TWO TIER APPLICATION SERIES

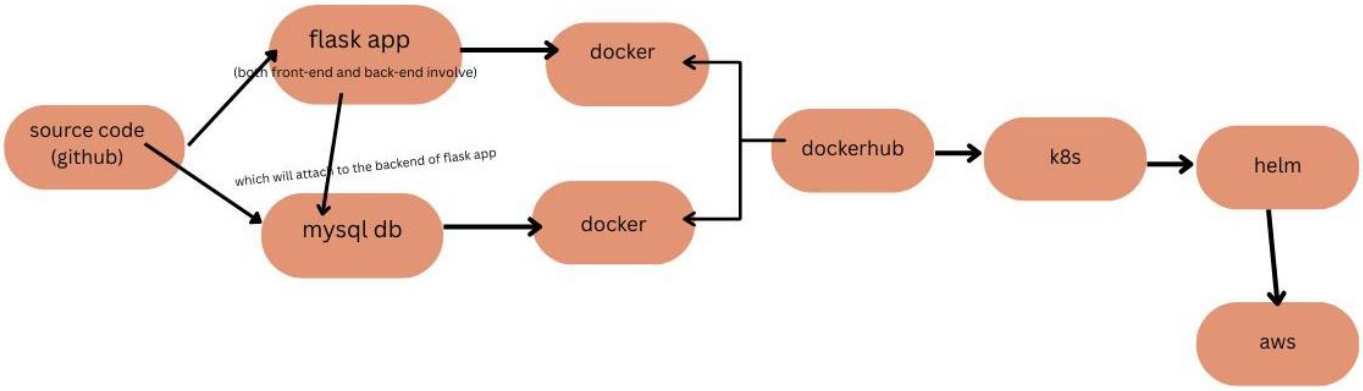


WHAT IS A TWO TEIR APPLICATION?

Means in this project we will deploy two tier 1/logic tier i.e flask app(where both front end and backend is involved)

2/database tier i.e mysql database which will be connected to backend of the flask app.

Lets go deep dive into the project



Explanation:

Accord to the project the code in the github will clone and will create a two tier application 1st tier is flask app where both front end and backend is available and the 2nd tier is mysql data base tier where the backend of the flask app is connected with the mysql database tier.After that we dockerize both the flask app and mysql and where we get an image which will push to dockerhub and pulled by k8s i.e kubernetes and then it will package by helm and deploy in aws.

EP-2

DOCKERIZE THE APPLICATION

It's a python based framework application.

app.py- from this devops engineer see the code and accord to that we can relate to an application by dockerizing it.

From this code app.py we actually create the dockerfile.

What is docker?

Docker is an open source tool which packages the application with its dependencies, libraries, Os Frameworks and other requirements and can deploy any linux server.

In this project we will create a Dockerfile from that we can build an image and from that image we can create a container.

Dockerfile-----→**image**-----→**Container**

Image-it's a readymade static template with application code and dependencies and the running instance of that image is called Container. Where the application is about to run.

In this project we are doing a flask app where in the flask app there is backend and frontend and the backend will connect with database i.e mysql. So we will create mysql database container also and as it is a python app we will also do a python app container .

From the app.py

two-tier-flask-app / app.py

Ubuntu Added some tweaks

Code Blame 48 lines (40 loc) · 1.41 KB Raw

```python
import os
from flask import Flask, render_template, request, redirect, url_for, jsonify
from flask_mysqldb import MySQL

app = Flask(__name__)

# Configure MySQL from environment variables
app.config['MYSQL_HOST'] = os.environ.get('MYSQL_HOST', 'localhost')
app.config['MYSQL_USER'] = os.environ.get('MYSQL_USER', 'default_user')
app.config['MYSQL_PASSWORD'] = os.environ.get('MYSQL_PASSWORD', 'default_password')
app.config['MYSQL_DB'] = os.environ.get('MYSQL_DB', 'default_db')

# Initialize MySQL
mysql = MySQL(app)

def init_db():
    with app.app_context():
        cur = mysql.connection.cursor()
        cur.execute('''
        CREATE TABLE IF NOT EXISTS messages (
            id INT AUTO_INCREMENT PRIMARY KEY,
            message TEXT
```

What we have discovered:

The app is flask application ,so here we need flask libraries

Flask app using mysql –so we also need mysql libraries

We also need environment variables

<u>Environment Variables:</u>

In this code there is some environment variable also why to collect mysql data we are using environment variables and we have to connect both of them that is mysql and flask.

```python
def init_db():
    with app.app_context():
        cur = mysql.connection.cursor()
        cur.execute('''
        CREATE TABLE IF NOT EXISTS messages (
            id INT AUTO_INCREMENT PRIMARY KEY,
            message TEXT
        );
        ''')
        mysql.connection.commit()
        cur.close()

@app.route('/')
def hello():
    cur = mysql.connection.cursor()
    cur.execute('SELECT message FROM messages')
    messages = cur.fetchall()
    cur.close()
    return render_template('index.html', messages=messages)

@app.route('/submit', methods=['POST'])
def submit():
    new_message = request.form.get('new_message')
    cur = mysql.connection.cursor()
    cur.execute('INSERT INTO messages (message) VALUES (%s)', [new_message])
    mysql.connection.commit()
    cur.close()
    return jsonify({'message': new_message})
```

Def hello

cur = mysql.connection.cursor both are functions which are making connection with mysql they both are mainly APIs.

new message

how api works simply-it goes to client receive the request goes to the database about the response

and mainly the api's that is hello and submit were written in flask.

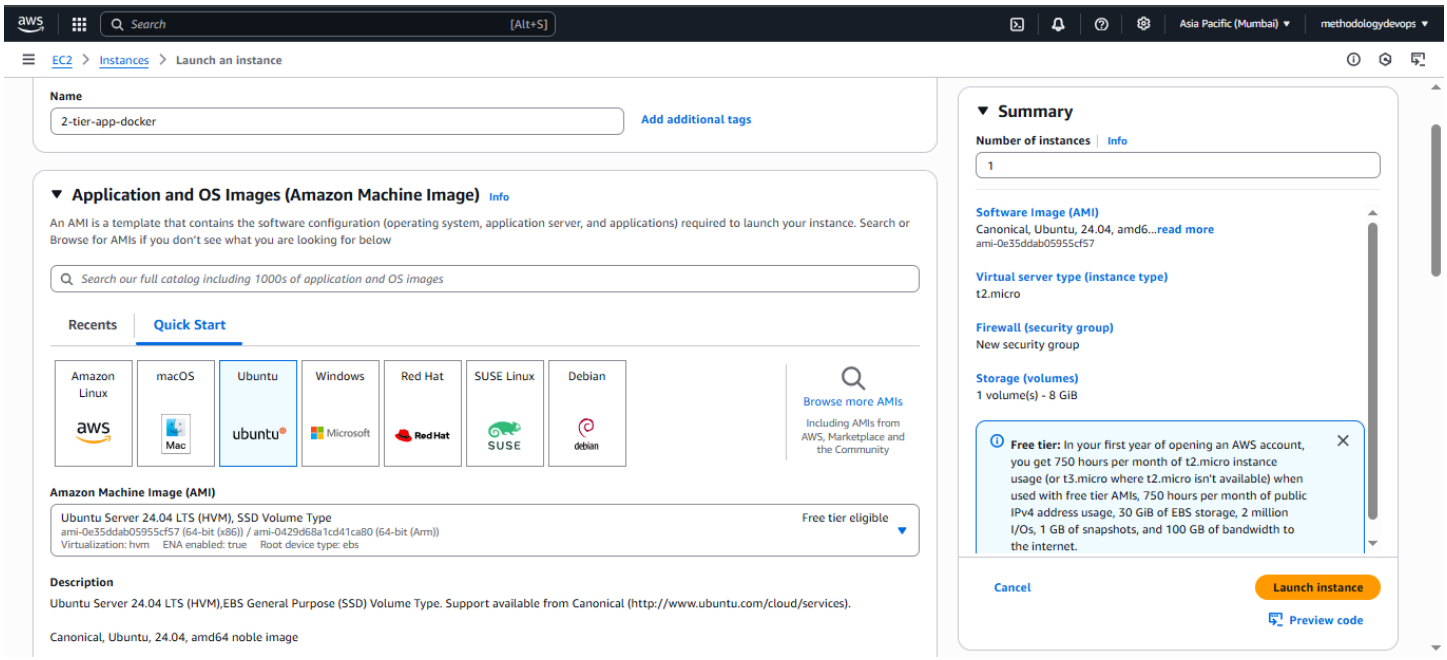
Application is working on port no -5000---noted down

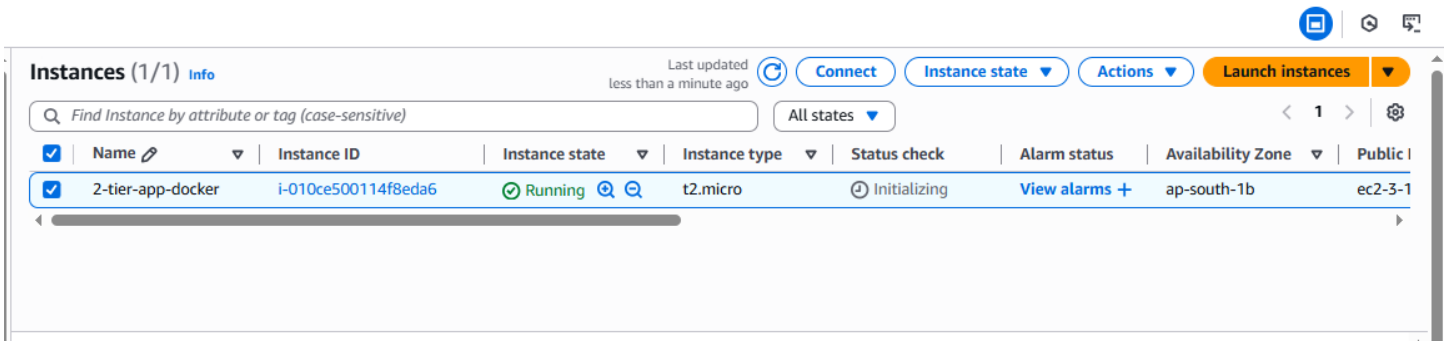
Practical.

Ubuntu

T2 micro

2-tier-app-docker





Then connect with the machine I have just used mobaxterm –its simple and useful.

```
2. 3.110.147.110 (ubuntu)

 ▸ SSH session to ubuntu@3.111.53.249
   • Direct SSH      :  ✔
   • SSH compression :  ✔
   • SSH-browser     :  ✔
   • X11-forwarding  :  ✔  (remote display is forwarded through SSH)

 ▸ For more info, ctrl+click on help or visit our website.

Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Jun  4 07:42:15 UTC 2025

  System load:  0.38               Processes:             112
  Usage of /:   25.0% of 6.71GB    Users logged in:       0
  Memory usage: 23%                IPv4 address for enX0: 172.31.15.60
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

TThe list of available updates is more than a week old.TTo check for new updates run: sudo apt update

/usr/bin/xauth:  file /home/ubuntu/.Xauthority does not exist
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-15-60:~$
```

COMMANDS:

sudo apt update

sudo apt install docker.io –y

```
Get:56 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [380 B]
Fetched 34.4 MB in 27s (1286 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
101 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-15-60:~$ sudo apt install docker.io –y
```

check the docker is running or not

```
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-15-60:~$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.47/containers/json": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-15-60:~$ sudo usermod -aG docker $USER
ubuntu@ip-172-31-15-60:~$ newgrp docker
ubuntu@ip-172-31-15-60:~$
```

for that we use the command

docker ps

for the error resolve

sudo usermod –aG docker $USER(this command is usermodification done and add docker to the current user that ubuntu ,so that docker has the required permission to do the task)

newgrp docker(to sudden effect instead restarting the server its easy)

docker ps

```
ubuntu@ip-172-31-15-60:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-15-60:~$
```

then git clone https://github.com/devops-methodology/two-tier-flask-app.git

```
ubuntu@ip-172-31-15-60:~$ git clone https://github.com/devops-methodology/two-tier-flask-app.git
Cloning into 'two-tier-flask-app'...
remote: Enumerating objects: 451, done.
remote: Total 451 (delta 0), reused 0 (delta 0), pack-reused 451 (from 1)
Receiving objects: 100% (451/451), 115.17 KiB | 8.23 MiB/s, done.
Resolving deltas: 100% (224/224), done.
ubuntu@ip-172-31-15-60:~$
```

ls then create Dockerfile

as it is python based framework so for that

```
2. 3.110.147.110 (ubuntu)
FROM python:3.9-slim

WORKDIR /app

RUN apt-get update -y \
   && apt-get upgrade -y \
   && apt-get install -y gcc default-libmysqlclient-dev pkg-config \
   && rm -rf /var/lib/apt/lists/*

COPY requirements.txt .

RUN pip install -r mysqlclient
RUN pip install -r requirements.txt

COPY . .

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

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                    17,15         All
```

we will use python based image

FROM python:3.9-slim (we use slim to reduce the image)

(it will create an os where it will already installed the python image and all dependencies and libraries)

Then we want the application to run in an folder for that we will use /WORKDIR in an folder named as app

WORKDIR /app

After that we want our sytem to update how we have to run it that's why we will use

RUN command

**RUN apt-get update –y **

**&& apt-get upgrade –y **

&& apt-get install -y gcc default-libmysqlclient-dev pkg-config \(as we are using mysql to run we need libraries for that purpose we run this command and mysql client)

&& rm -rf /var/lib/apt/lists/*(means during installation some unnecessary lists and temp files will be created we have to delete that)

After that in app.py there is requirements.txt

In there there were some name that has to be installed and run the packages

So first we have to copy it then we will run it

COPY requirements.txt .

Internally python app access mysql for that we will also run the mysql client

RUN pip install mysqlclient

RUN pip install –r requirements.txt

Then we want the code inside the container for that

We will use

COPY . . (1/. Is for the source and the 2/ . is for the in the container we want to copy)

After that to run the this app

CMD [“python”,”app”]

After that we have to build that

docker build –t flaskapp .

```
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$ docker build -t flaskapp .
```

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
flaskapp     latest    baad870daaab   17 seconds ago   392MB
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker images -a
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
flaskapp     latest    baad870daaab   30 seconds ago   392MB
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

We can check by using this command

docker ps

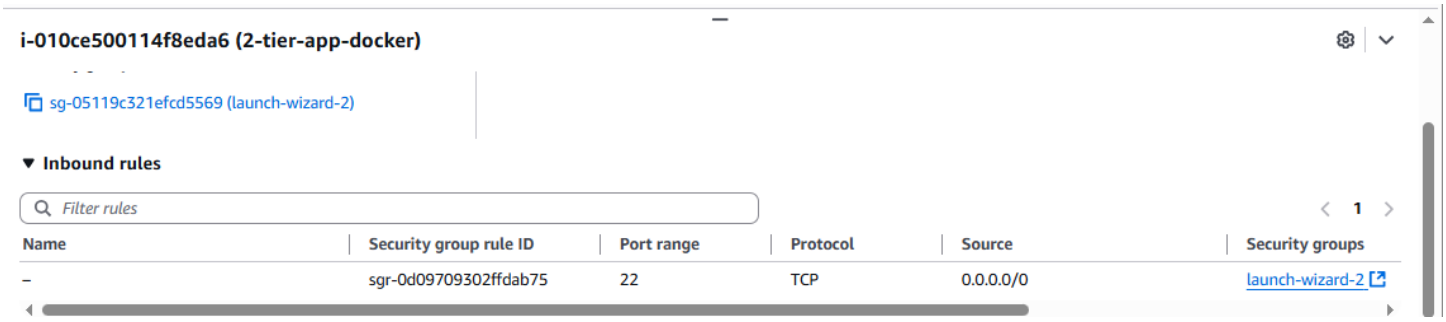
```
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$ docker build -t flaskapp .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  239.1kB
Step 1/8 : FROM python:3.9-slim
3.9-slim: Pulling from library/python
61320b01ae5e: Pull complete
2481a58f9b3d: Pull complete
1692d37168f6: Pull complete
a0684e18c375: Pull complete
Digest: sha256:aff2066ec8914f7383e115bbbcde4d24da428eac377b0d4bb73806de992d240f
Status: Downloaded newer image for python:3.9-slim
 ---> 1be4b628ef55
Step 2/8 : WORKDIR /app
 ---> Running in 377226895de9
 ---> Removed intermediate container 377226895de9
 ---> 63b96a265ddf
Step 3/8 : RUN apt-get update     && apt-get upgrade -y     && apt-get install -y gcc default-libmysqlclient-dev pkg-config     && rm -rf
 /var/lib/apt/lists/*
 ---> Running in 8e822b53379a
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8793 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [512 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [265 kB]
Fetched 9313 kB in 1s (6482 kB/s)
Reading package lists...
```

after that we can check docker images

```
Step 7/8 : COPY . .
 ---> de42ef31de56
Step 8/8 : CMD ["python", "app.py"]
 ---> Running in afb397af3a72
 ---> Removed intermediate container afb397af3a72
 ---> a47ec2cd4287
Successfully built a47ec2cd4287
Successfully tagged flaskapp:latest
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$ docker images
REPOSITORY   TAG        IMAGE ID       CREATED          SIZE
flaskapp     latest     a47ec2cd4287   4 seconds ago    392MB
python       3.9-slim   1be4b628ef55   7 weeks ago      126MB
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$
```

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker run -d -p 5000:5000 flaskapp:latest
2b4dcbd8a9155807f3a905d6fcb9961a0aa46807db269333bb62488273dbdf09
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

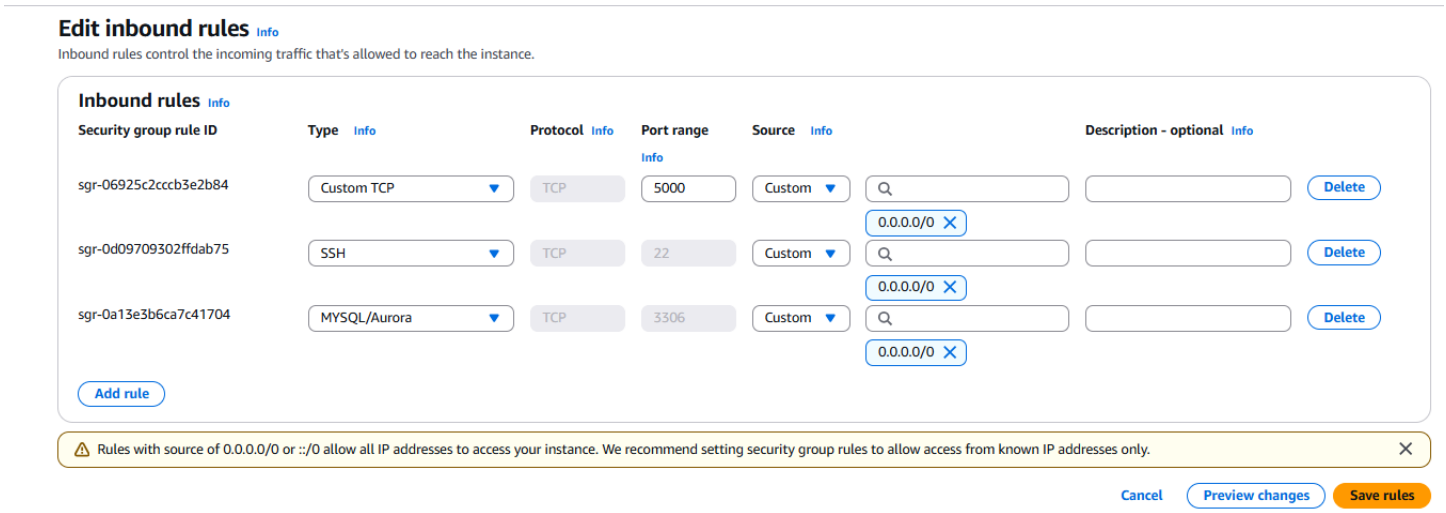
security group



Some ports to be added

5000- where the flask-app will run

3306- for mysql database



After that we have to create a container from that image

For that we will write a command

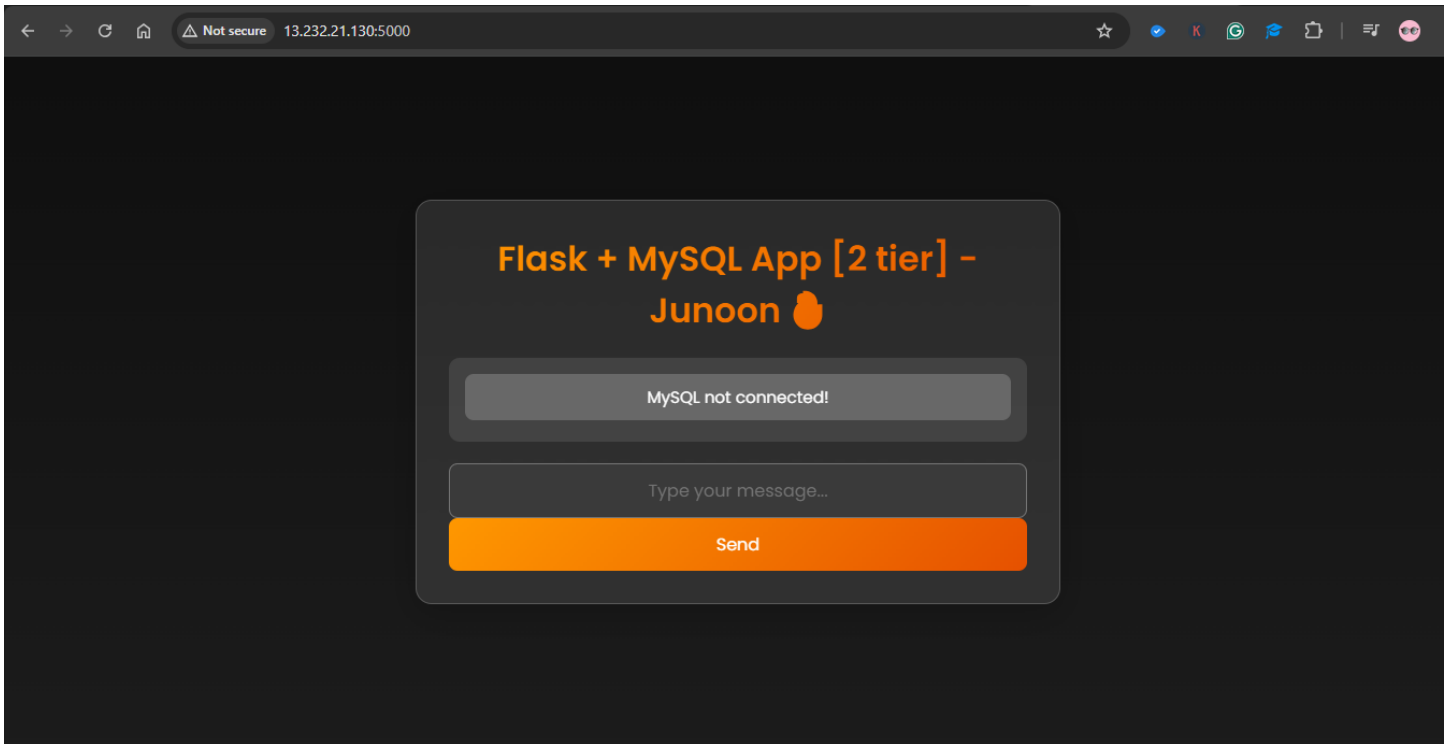
docker run –d –p 5000:5000 flaskapp:latest

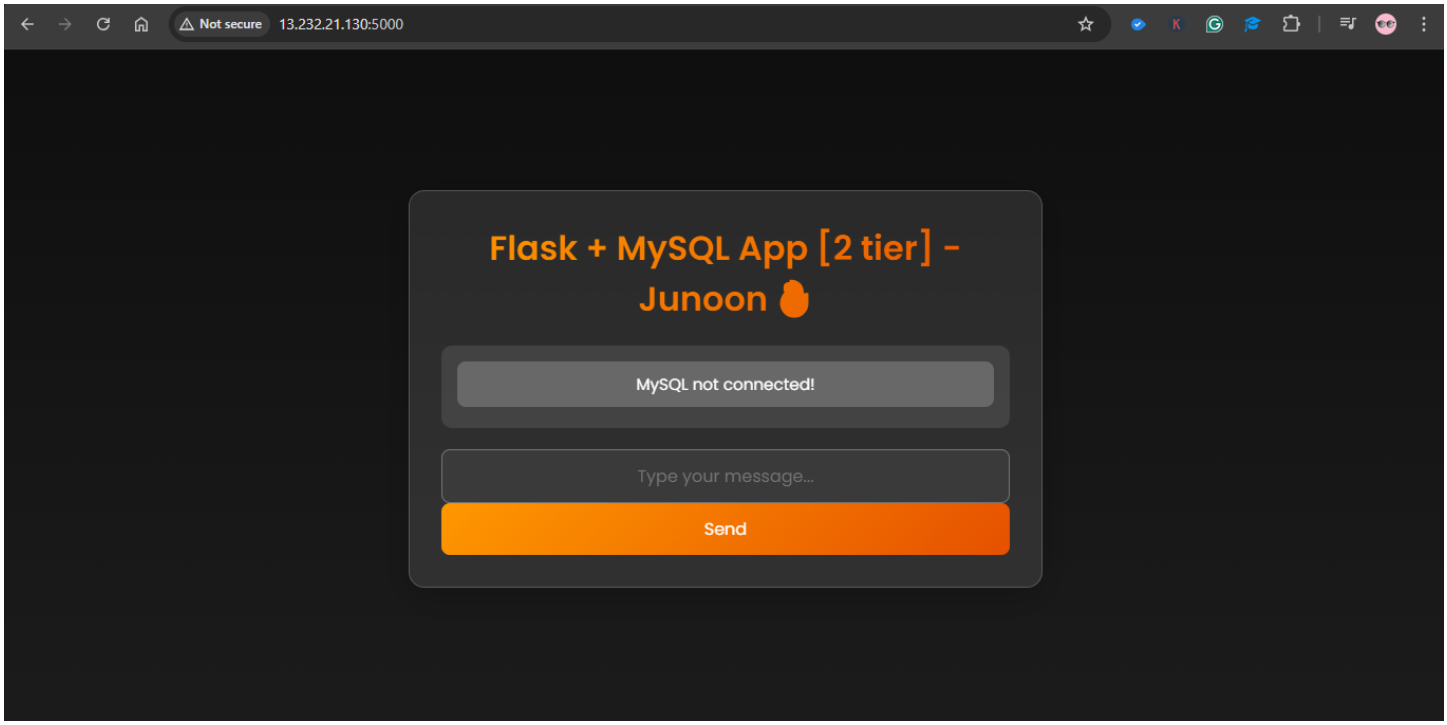
```
flaskapp     latest    92eeb2338c6f   44 seconds ago   564MB
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker run -d -p 5000:5000 --name flaskapp flaskapp:latest
b714108088cabce12ea758eda8fa1bb3b9697f376052744e2531132aeffddd91
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker ps
CONTAINER ID   IMAGE             COMMAND           CREATED         STATUS         PORTS                                          NAMES
b714108088ca   flaskapp:latest   "python app.py"   3 seconds ago   Up 3 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   flaskapp
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker ps
CONTAINER ID   IMAGE             COMMAND           CREATED          STATUS          PORTS                                          NAMES
b714108088ca   flaskapp:latest   "python app.py"   12 seconds ago   Up 12 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   flaskap
p
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
flaskapp     latest    92eeb2338c6f   2 minutes ago   564MB
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ cat requirements.txt
Flask==2.3.2
Flask-MySQLdb==1.0.1

ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

I have just changed requirements.txt and app.py and docker run command as it previously throwing error.

We can check the flask-app though instance ip 13.232.21.130:5000 (flask-app is running) but clearly showing mysql is not connected.





```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
ad8b1f65ec8a   bridge    bridge    local
b98b97f9f83f   host      host      local
bf76eb947d17   none      null      local
1bea45977176   twotier   bridge    local
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

After running it will got an error as an flask app want an database to store that's why it will ask for mysql container so we have to create mysql container for that.

```
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$ docker run -d \
    --name mysql \
    -v mysql-data:/var/lib/mysql \
    --network=twotier \
    -e MYSQL_DATABASE=mydb \
    -e MYSQL_ROOT_PASSWORD=admin \
    -p 3306:3306 \
    mysql:5.7
Unable to find image 'mysql:5.7' locally
5.7: Pulling from library/mysql
20e4dcae4c69: Extracting [==================================>              ]   36.7MB/50.5MB
1c56c3d4ce74: Download complete
e9f03a1c24ce: Download complete
68c3898c2015: Download complete
6b95a940e7b6: Download complete
90986bb8de6e: Download complete
ae71319cb779: Download complete
ffc89e9dfd88: Download complete
43d05e938198: Download complete
064b2d298fba: Download complete
df9a4d85569b: Download complete
```

So we will create mysql container but to connect both of them we have to create a network which will connect to both of them.

```
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$ docker network create twotier
ce1e23512b0216b1a5878bd540dec2dca84a10fd154cef272c9ad78d283ec31f
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$
```

docker create network twotier

after that we have to give network to both of the container.

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker run -d -p 5000:5000 --network=twotier -e MYSQL_HOST=mysql -e MYSQL_USER=admin -e MYSQ
L_PASSWORD=admin -e MYSQL_DB=myDb flaskapp:latest
e8e7b4093d2534940015dfffbc068954a48ec73f288b4ca4d285b10e99a65e19
```

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker run -d -p 3306:3306 --network=twotier -e MYSQL_DB=myDb -e MYSQL_USER=admin -e MYSQL_P
ASSWORD=admin -e MYSQL_ROOT_PASSWORD=admin mysql:5.7
c55f33a7036ff82df5eef71d78d47bc2c6cb8ae4285651374457e15802b45a3a
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

For mysql container

**docker run -d **

**--name mysql **

**-v mysql-data:/var/lib/mysql **

**--network=twotier **

**-e MYSQL_DATABASE=mydb **

**-e MYSQL_ROOT_PASSWORD=admin **

**-p 3306:3306 **

mysql:5.7

<u>for backend container</u>

**docker run -d **

**--name flaskapp **

**--network=twotier **

**-e MYSQL_HOST=mysql **

**-e MYSQL_USER=root **

**-e MYSQL_PASSWORD=admin **

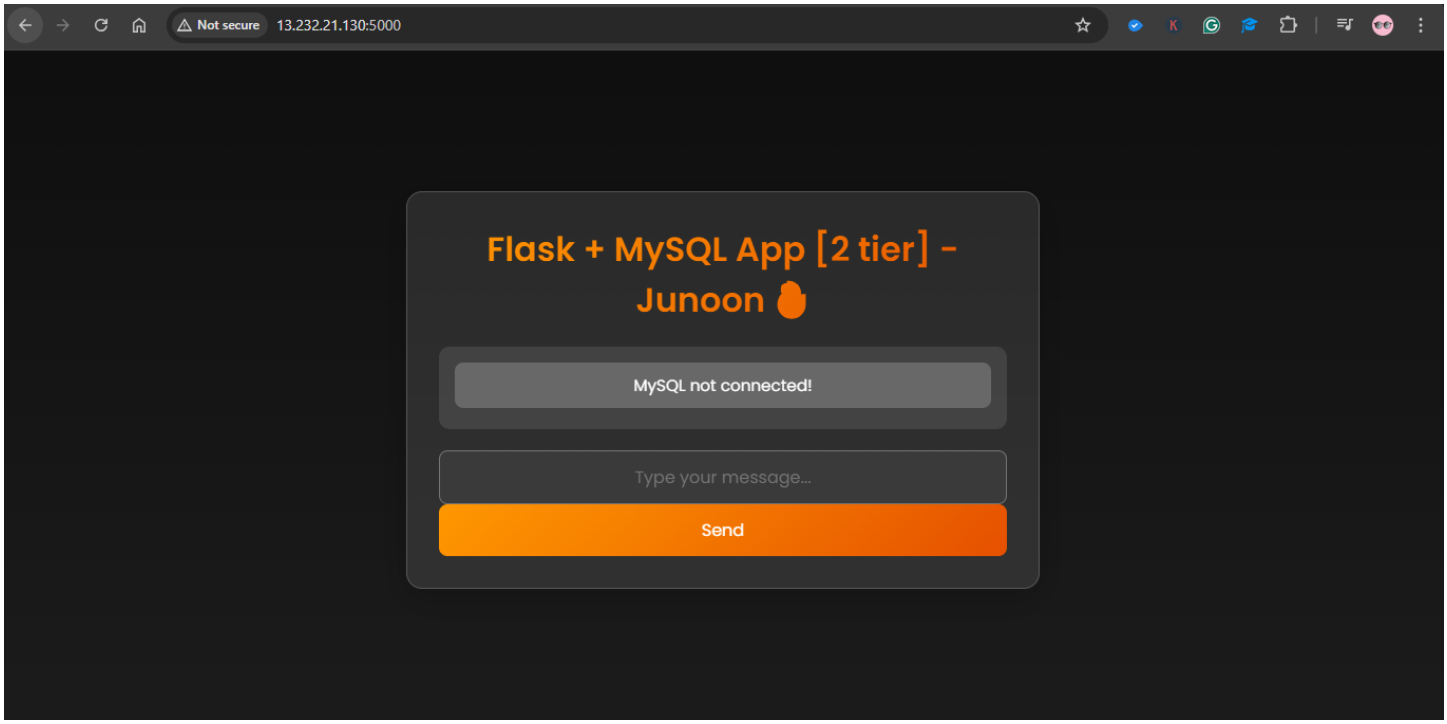
**-e MYSQL_DB=mydb **

**-p 5000:5000 **

flaskapp:latest

```
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$ docker run -d -p 5000:5000 --network=twotier -e MYSQL_HOST=mysql -e MYSQL_USER=admin -e MYSQ
L_PASSWORD=admin -e MYSQL_DB=myDb flaskapp:latest
77377a60b13a92c1597920b4d60c1e900c64eff4c81e15cf33394fb52659fbf8
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$ docker run -d -p 3306:3306 --network=twotier -e MYSQL_HOST=mysql -e MYSQL_DATABASE=myDb -e M
YSQL_USER=admin -e MYSQL_PASWORD=admin -e MYSQL_ROOT_PASSWORD=admin mysql:5.7
adc50ed1edc5f68b536083fcccca7f376b9879b8cf6764901afa11851499657c
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$
```

```
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS          PORTS
           NAMES
c55f33a7036f   mysql:5.7         "docker-entrypoint.s…"   45 seconds ago   Up 45 seconds   0.0.0.0:3306->3306/tcp, [::]:3306->3306/tcp, 3
3060/tcp   brave_chebyshev
e8e7b4093d25   flaskapp:latest   "python app.py"          3 minutes ago    Up 3 minutes    0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
           sweet_elgamal
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```



after that we can check for that the both the container is running in the same container or not

for that we will use

docker network ls

```
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
1078c3d9e5d0   bridge    bridge    local
919ff4aedf41   host      host      local
8ebb5a896cf7   none      null      local
ce1e23512b02   twotier   bridge    local
ubuntu@ip-172-31-15-60:~/two-tier-flask-app$
```

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker network inspect twotier
```

We have to check the docker network-actually both flask-app and mysql container is in the same network or not

For that we will use

docker network inspect twotier

```
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "c55f33a7036ff82df5eef71d78d47bc2c6cb8ae4285651374457e15802b45a3a": {
                "Name": "brave_chebyshev",
                "EndpointID": "04829a8774b5f1439739912e223d04e357316d4e983cf3a5509b362bcccbaa82",
                "MacAddress": "fa:44:87:68:36:87",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
            "e8e7b4093d2534940015dfffbc068954a48ec73f288b4ca4d285b10e99a65e19": {
                "Name": "sweet_elgamal",
                "EndpointID": "856e10eb004145c87fc1c887bf6e416855afd874ba48303fda04bbed7c063e9e",
                "MacAddress": "f6:7d:1c:3a:0e:91",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
```

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker kill e8e7b4093d25
e8e7b4093d25
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker run -d -p 3306:3306 --network=twotier -e MYSQL_DB=myDb -e MYSQL_USER=admin -e MYSQL_P
ASSWORD=admin -e MYSQL_ROOT_PASSWORD=admin --name=mysql mysql:5.7
docker: Error response from daemon: Conflict. The container name "/mysql" is already in use by container "29963bfff36eb540d24c14afd7e3a29
250f3c522d2872a75df269bd5639cd215". You have to remove (or rename) that container to be able to reuse that name.

Run 'docker run --help' for more information
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker rm 29963bfff36
29963bfff36
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker run -d -p 3306:3306 --network=twotier -e MYSQL_DB=myDb -e MYSQL_USER=admin -e MYSQL_P
ASSWORD=admin -e MYSQL_ROOT_PASSWORD=admin --name=mysql mysql:5.7
d7bdd5e550dc55798bc902a8b20cf37a3da6561d0806a9d5784556a3bfa3b609
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker run -d -p 5000:5000 --network=twotier -e MYSQL_HOST=mysql -e MYSQL_USER=admin -e MYSQ
L_PASSWORD=admin -e MYSQL_DB=myDb --name=flaskapp flaskapp:latest
docker: Error response from daemon: Conflict. The container name "/flaskapp" is already in use by container "b714108088cabce12ea758eda8fa
1bb3b9697f376052744e2531132aeffddd91". You have to remove (or rename) that container to be able to reuse that name.

Run 'docker run --help' for more information
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker rm b714108088ca
b714108088ca
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker run -d -p 5000:5000 --network=twotier -e MYSQL_HOST=mysql -e MYSQL_USER=admin -e MYSQ
L_PASSWORD=admin -e MYSQL_DB=myDb --name=flaskapp flaskapp:latest
80d0be4811257090 1b5151ae513892d2095a3088d842e78d990392b2ef102b14
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

so to check the network

command will be

docker network inspect twotier

you can check it

after that we have to give unique name for that 1st we have to kill the container

then same cli but only name added.

For sql

```
docker run -d \
  --name mysql \
  -v mysql-data:/var/lib/mysql \
  --network=twotier \
  -e MYSQL_DATABASE=mydb \
  -e MYSQL_ROOT_PASSWORD=admin \
  -p 3306:3306  --name=mysql\
  mysql:5.7
```

for backend container

```
docker run -d \
   --name flaskapp \
   --network=twotier \
   -e MYSQL_HOST=mysql \
   -e MYSQL_USER=root \
   -e MYSQL_PASSWORD=admin \
   -e MYSQL_DB=mydb \
   -p 5000:5000 --name-flaskapp \
   flaskapp:latest
```

after creating container when we refresh the message it will again show the error we have to create a mysql table message for that….

For that we will create this table

```
CREATE TABLE messages (
  id INT AUTO_INCREMENT PRIMARY KEY,
  message TEXT
);
```

Bt we have write this command inside the container

We have to go inside the container

For that

docker exec –it d7bdd5e550dc bash

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker exec -it d7bdd5e550dc bash
bash-4.2#
```

ls

mysql –u root –p

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker exec -it d7bdd5e550dc bash
bash-4.2# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
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>
```

admin

show databases;

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

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| myDb               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> use myDb
```

use myDb;(in that paste the table message)

CREATE TABLE messages (

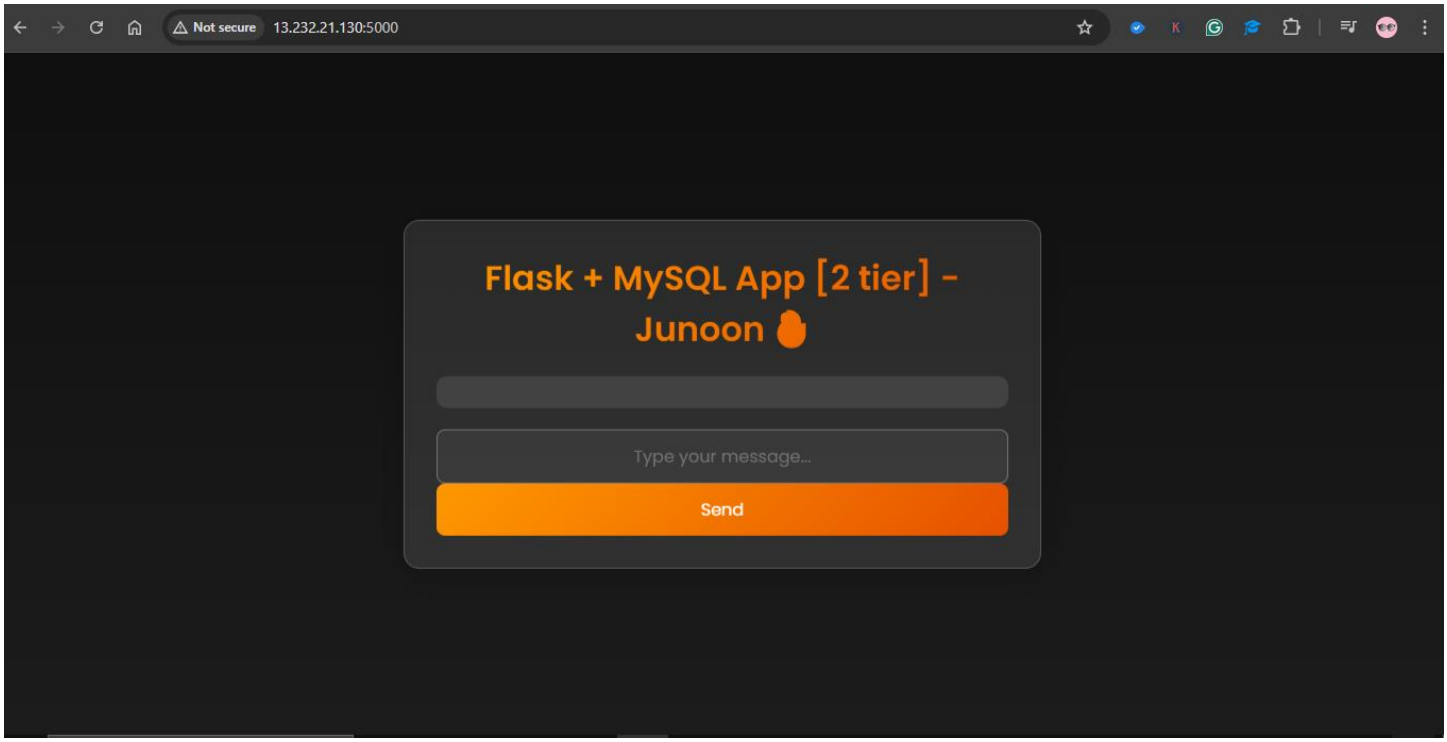
id INT AUTO_INCREMENT PRIMARY KEY,

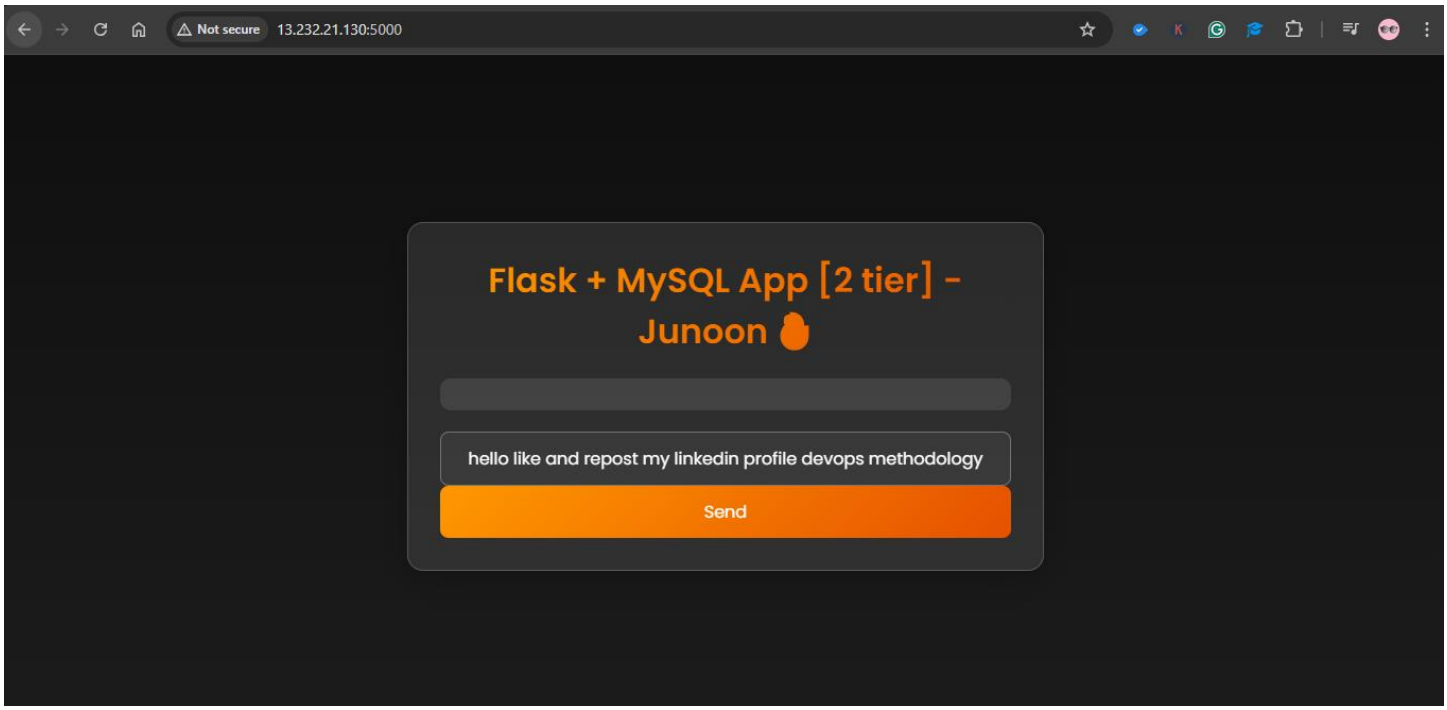
message TEXT

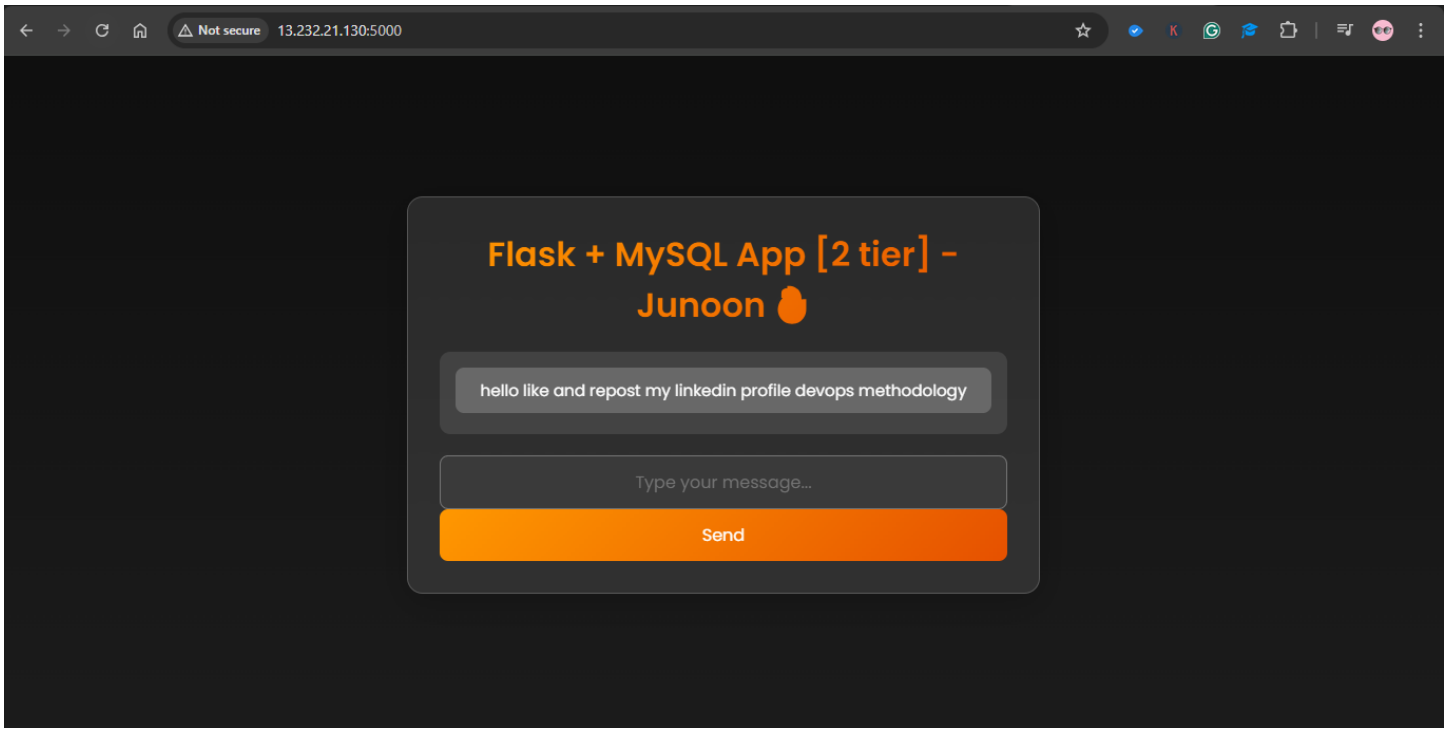
);

```
mysql> use myDb;
Database changed
mysql> CREATE TABLE messages (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     message TEXT
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Then just refresh the page you will find the flaskapp







Actually there is some error in app.py instead of message its written as new_message that's why its showing error I have just rectified it.

*Select * from messages;*

You can check in the browser the messages which I have written its showing in mysql server.

```
mysql> select * from messages;
+----+-----------------------------------------------------------------------+
| id | message                                                               |
+----+-----------------------------------------------------------------------+
|  1 | hello like and repost my linkedin profile devops methodology          |
|  2 | hello                                                                 |
|  3 | just share the post as i have rectified it  and it tooks a lot of time |
+----+-----------------------------------------------------------------------+
3 rows in set (0.00 sec)

mysql>
```

(Means select message all from that …every detail you have in the messages)

Ep-2

Lets say you have to do multiple container its hard to manage and literally time taken to individually write for the docker container .we will use for that docker-compose

For that we have to push to docker image to docker hub for that we have to tag and push the image

1st we have to login

docker login – <user-name>

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker login -u premd91

i Info → A Personal Access Token (PAT) can be used instead.
         To create a PAT, visit https://app.docker.com/settings


Password:

WARNING! Your credentials are stored unencrypted in '/home/ubuntu/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

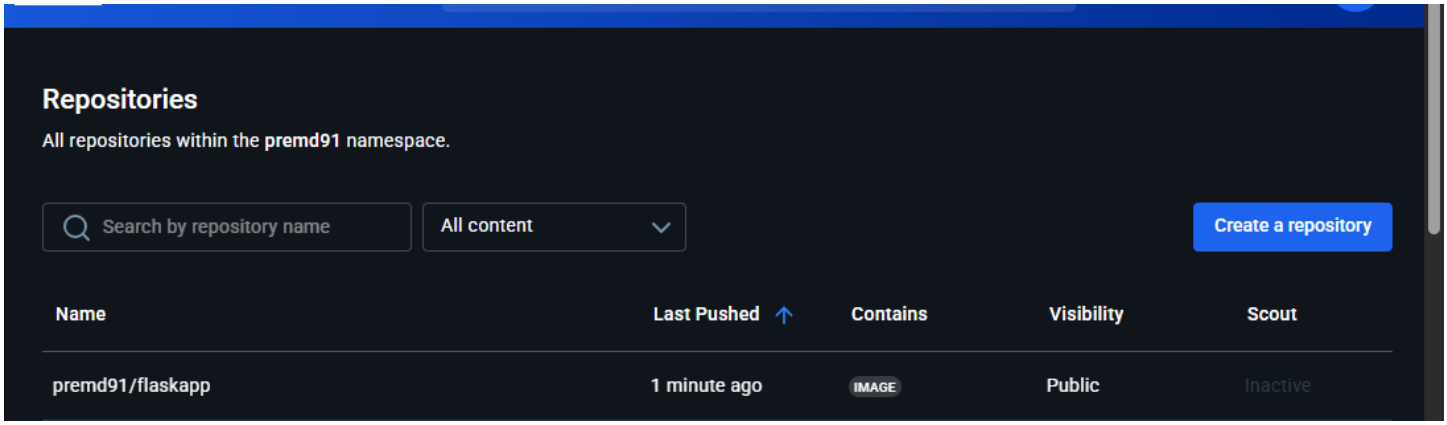
docker tag flaskapp:latest premd91/flaskapp:latest

then push it

docker push premd91/flaskapp:latest

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
flaskapp            latest    71743ceb56f2   20 minutes ago   564MB
premd91/flaskapp    latest    71743ceb56f2   20 minutes ago   564MB
mysql               5.7       5107333e08a8   18 months ago    501MB
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker push premd91/flaskapp:latest
The push refers to repository [docker.io/premd91/flaskapp]
f9a097d14c05: Pushed
0982232ec683: Pushed
54ad45cfe73f: Pushed
9877e254a8f7: Pushed
9a7855eacc0a: Pushed
978f260c1369: Mounted from library/python
9b5482944372: Mounted from library/python
9ad43ba78452: Mounted from library/python
ace34d1d784c: Mounted from library/python
latest: digest: sha256:db1f8bb689c28cf90607b7290655e8b44e68a35bba8717b4ca80b05f3b8857ab size: 2206
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

we can use anytime bu pulling this image anytime and anywhere



doing manually creating container and running the command we can do this simultaneously in short time.

So will use docker compose for that

We have to install it

sudo apt install docker-compose –y

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker-compose --version
docker-compose version 1.29.2, build unknown
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

to do this simultaneously running container we use docker-compose file

vi docker-compose.yaml

version: "3"

services:

backend:

image: premd91/flaskapp:latest(from dockerhub)

ports:

- **“5000:5000”**

environment variables:

- **MYSQL_HOST: “mysql”**
- **MYSQL_PASSWORD: “admin”**
- **MYSQL_USER: “admin”**
- **MYSQL_DB: “myDb”**
  depends_on
- **mysql**

mysql:

image: mysql:5.7

environments:

- **MYSQL_HOST: “mysql”**
- **MYSQL_PASSWORD: “admin”**
- **MYSQL_USER: “admin”**
- **MYSQL_ROOT_PASSWORD: “admin”**

Ports:

- **“3306:3306”**

volumes:

- **/message.sql:/docker-entrypoint-initdb.d/message.sql**
- **mysql-data:/var/lib/mysql**

volumes:

mysql-data

after that we have to kill the containers if there is some containers is running

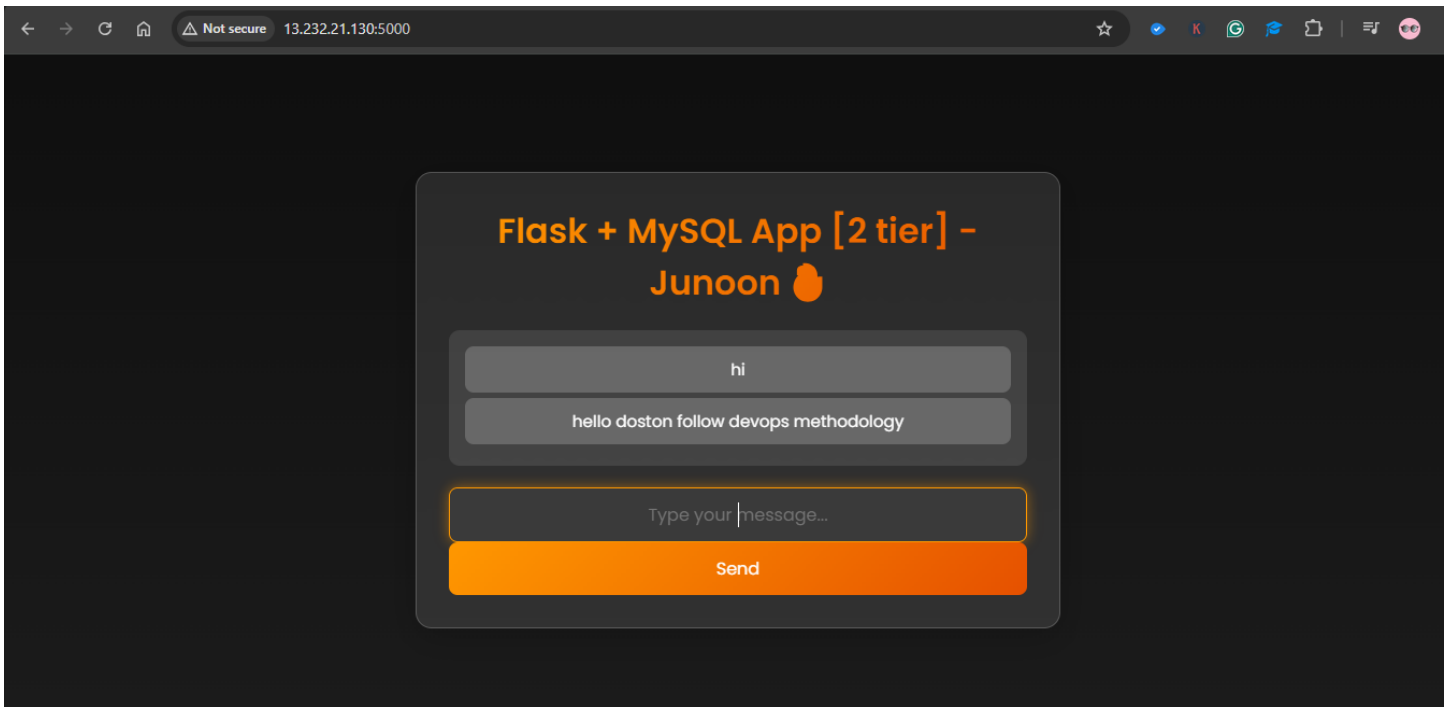
docker-compose up –d

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker-compose up -d
Creating network "two-tier-flask-app_default" with the default driver
Creating volume "two-tier-flask-app_mysql-data" with default driver
Creating two-tier-flask-app_mysql_1 ... done
Creating two-tier-flask-app_backend_1 ... done
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

docker compose –d down

After that again up

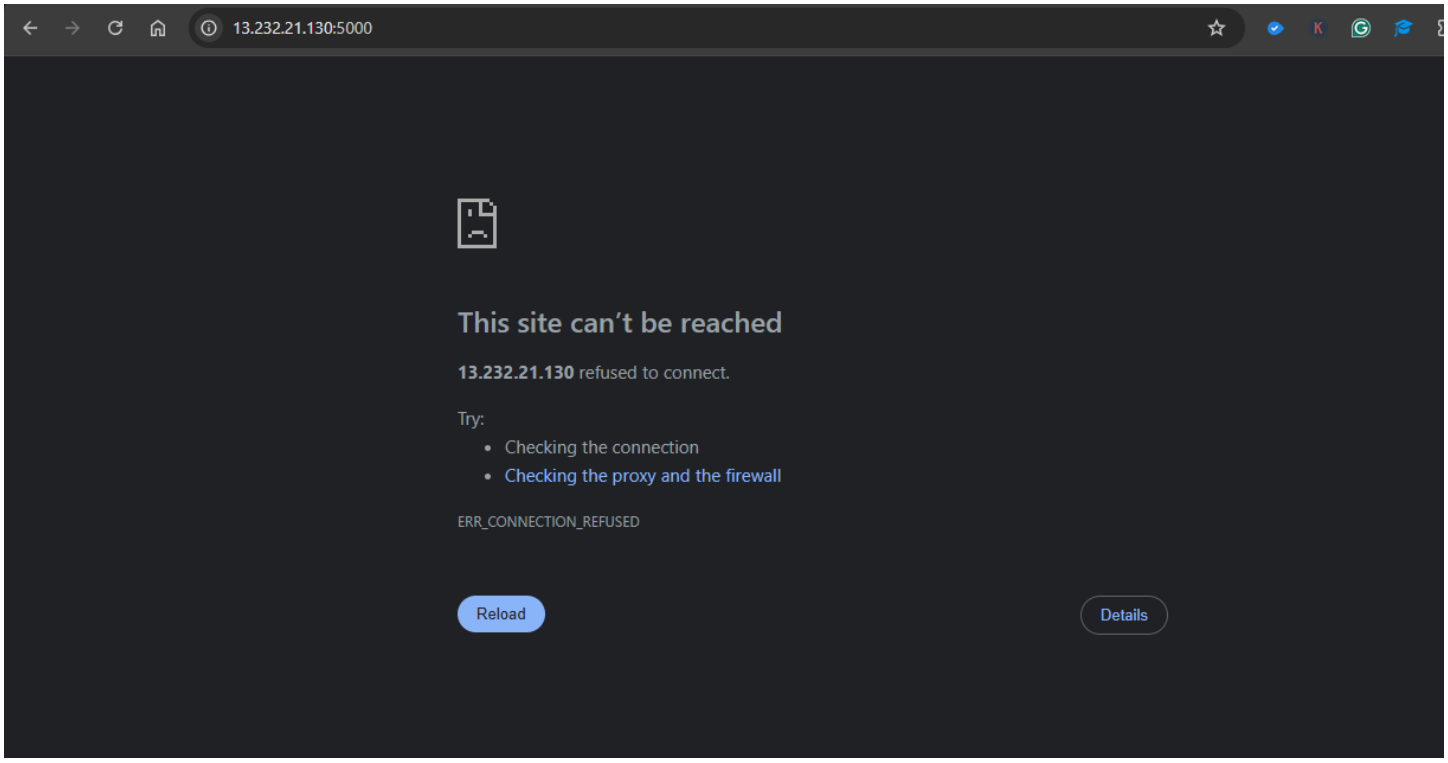
```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker compose up -d
WARN[0000] /home/ubuntu/two-tier-flask-app/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it
to avoid potential confusion
[+] Running 4/4
 ✔ Container two-tier-flask-app_mysql_1    Recreated                                                                    2.0s
 ✔ Container two-tier-flask-app_backend_1  Recreated                                                                    0.4s
 ✔ Container two-tier-flask-app-mysql-1    Started                                                                      0.4s
 ✔ Container two-tier-flask-app-backend-1  Started                                                                      0.4s
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```



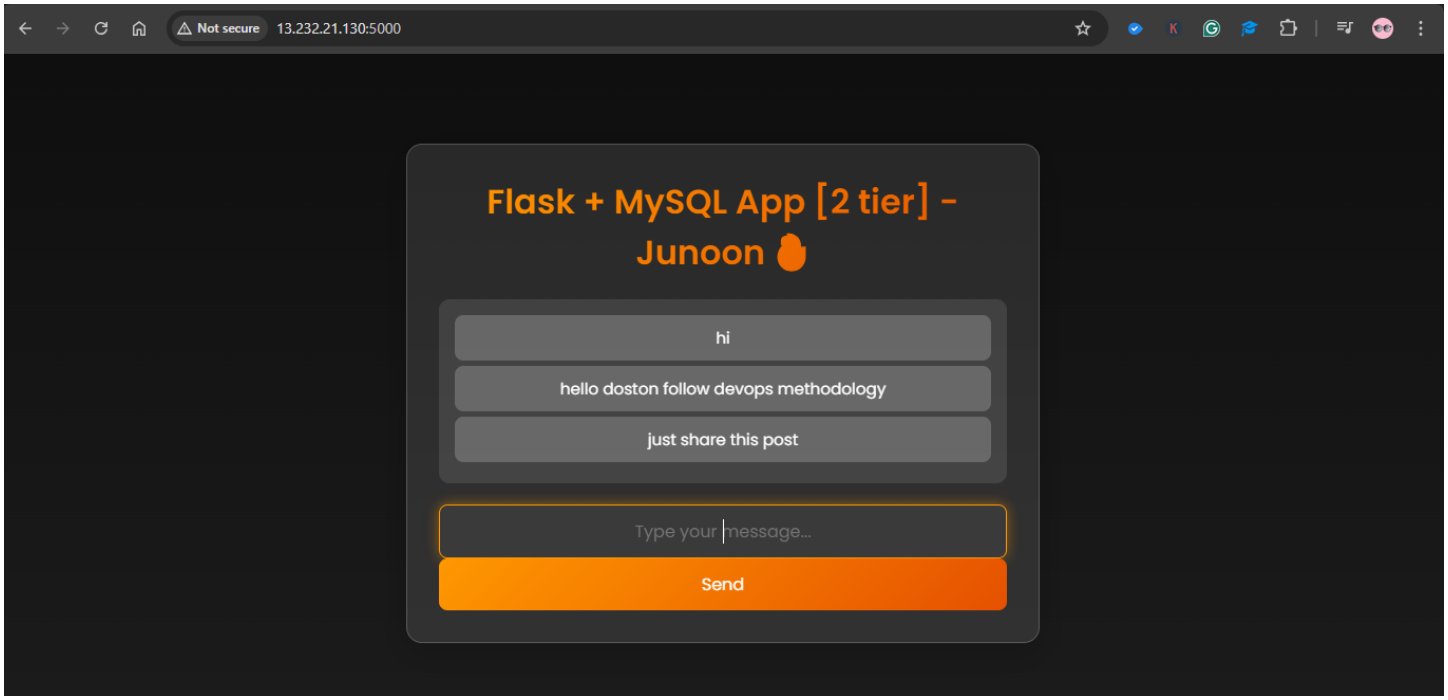

```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker-compose down
Stopping two-tier-flask-app-backend-1 ... done
Stopping two-tier-flask-app-mysql-1   ... done
Removing two-tier-flask-app-backend-1 ... done
Removing two-tier-flask-app-mysql-1   ... done
Removing network two-tier-flask-app_default
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```

When we use docker compose down

With in a second its not working so how we use docker compose for that.




```
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$ docker-compose up -d
Creating network "two-tier-flask-app_default" with the default driver
Creating two-tier-flask-app_mysql_1 ... done
Creating two-tier-flask-app_backend_1 ... done
ubuntu@ip-172-31-10-89:~/two-tier-flask-app$
```



EP-3 Kubernetes Architecture and Cluster Setup (Kubeadm) For DevOps | Episode 3

About kubernetes architecture?

Why we are using it?

In the age of era of micro-services where we work on different services for different environment for different services.

Instead of writing code and deploying manually in one place and when other services needed we have to do it manually and there is no option for auto-scaling when the business is up and also lets say one or two services crashed we have to resolve it manually so no auto healing features ,so to overcome these issues we use a feature or component called as kubernetes.

KUBERNETES(k8s):

It is an open-source tool basically famous for container orchestration tool means lets say to make an app or service we use docker container,but when lots of container needed we can't do it manually a lot of effort needed so doing manually we can use docker-compose or kubernetes but in docker-compose its hard to maintain autoscalaeble and auto-healing feature so for doing many microservices to maintain it we use kubernetes.

Kubernetes Architecture:

When more than one server is included we called as cluster.

In kubernetes as we manage a lot of services we needed a one or more than one cluster.

Where in a cluster accord to the requirement there will be one master node and one or two worker node will be there.

In some cases for a vast application we have more than one master node and worker node may be a lot.

So lets come to master node:

MASTER-NODE:

Basically for the kubernetes we tell this master node or control-plane as it controls the k8s processes.

In master node:

We have an API-server

Controller-manager

Scheduler/kube-scheduler

Ectd

API-server: in the master-node with in the cluster api-server is the entry-point to the cluster. When a user/devops engineer process a request it comes to api-server through(cli,api call and may be UI),it's the one point to the cluster for the entry,so it is safe to work on k8s.

After processing the request it goes to controller-manager it ensures that the state will always maintains i.e desired state=current state.

So after accord to the **controller manager** ----- it tells the kube-scheduler to maintain the state by maintaining the pod replacement. So **kube-scheduler** maintains/ensures the pod replacement…how it knows through after telling from **controller-manager**.**controller-manager** knows accord to the cpu memory usage,nodes(worker-node)capacity ---this information was gathered by **etcd**(stores in key value pair)information. It stores all the information of the cluster so lets say any pod is crashed is one of the nodes it suddenly knows and tell controller-manager to maintain the state and **kube-scheduler** ensures about the pod redplacement.

WORKER-NODES:

Actually all the work is done by worker node.That's why mainly cpu,memory storage in vats quantity is given to the worker nodes..

Master nodes only tell to the worker-nodes to do the required work

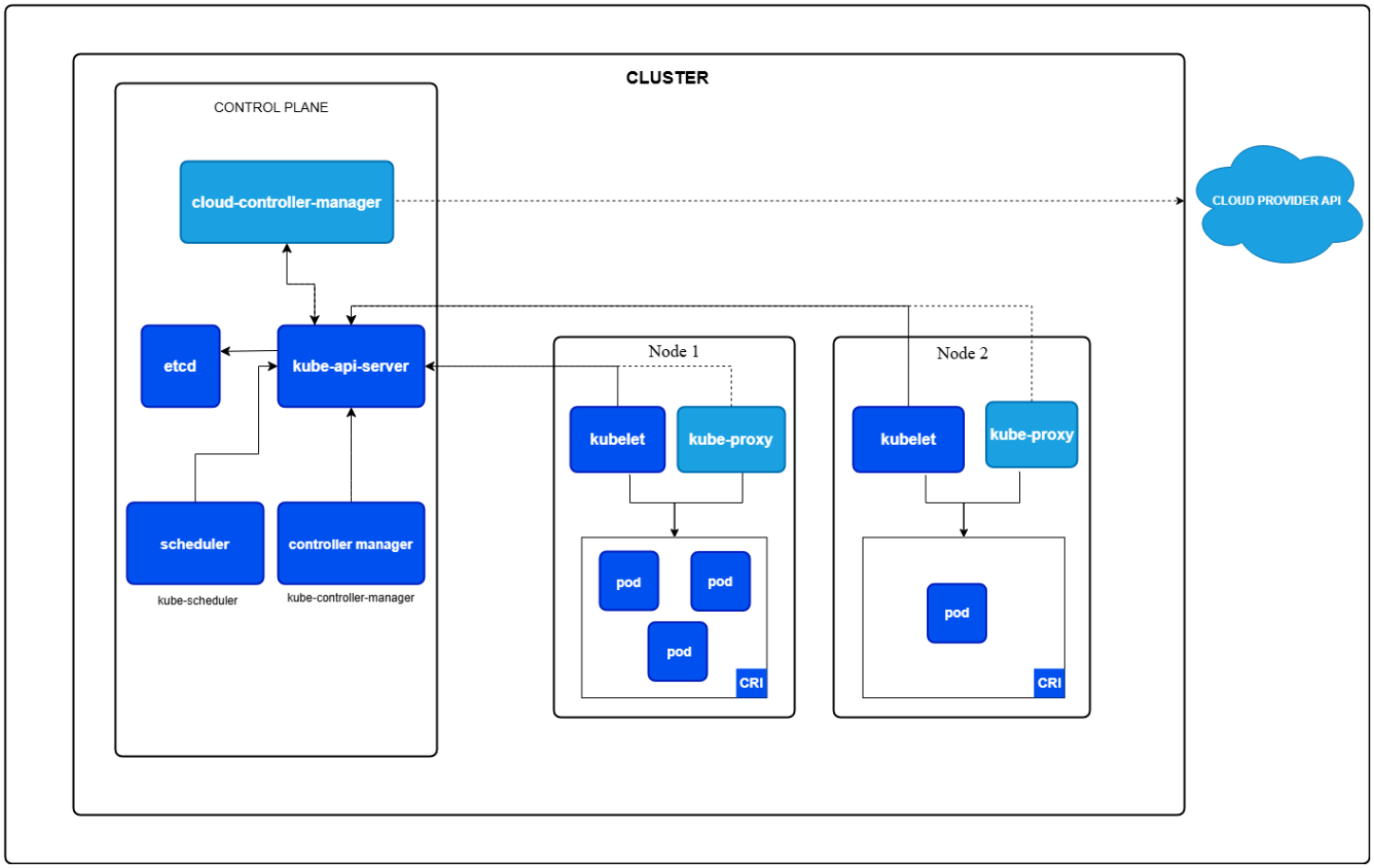
Where the application will work inside the container that is inside the pod.

*Where **POD** is the smallest unit of the cluster.*

Pod is responsible for the application working because pod is the abstraction layer over container all the service, all the thing is applied to pod.

*The work which is done by worker–node actually listened by **kubelet**..they talked with each other*

For running an application.



Actually kubelet is an agent which runs in each worker node.

It makes sure that container is running in each pod .

KUBE-PROXY:

It maintains the network rules in each nodes..we can use any network plugin instead of this like we are using ***calico*** *(network plugin) in this project.*

Kube-proxy *is responsible for the communication of the pod from inside or outside of the cluster.*

Then we use virtual-network that is container run time(CRT) in container runtime interface(CRI) which is responsible for making the cluster a robust system and also responsible for communication of master node and worker node talk to each other.

Practical

We are going to do kubeadm installation which we will create a control plane for k8s

For that

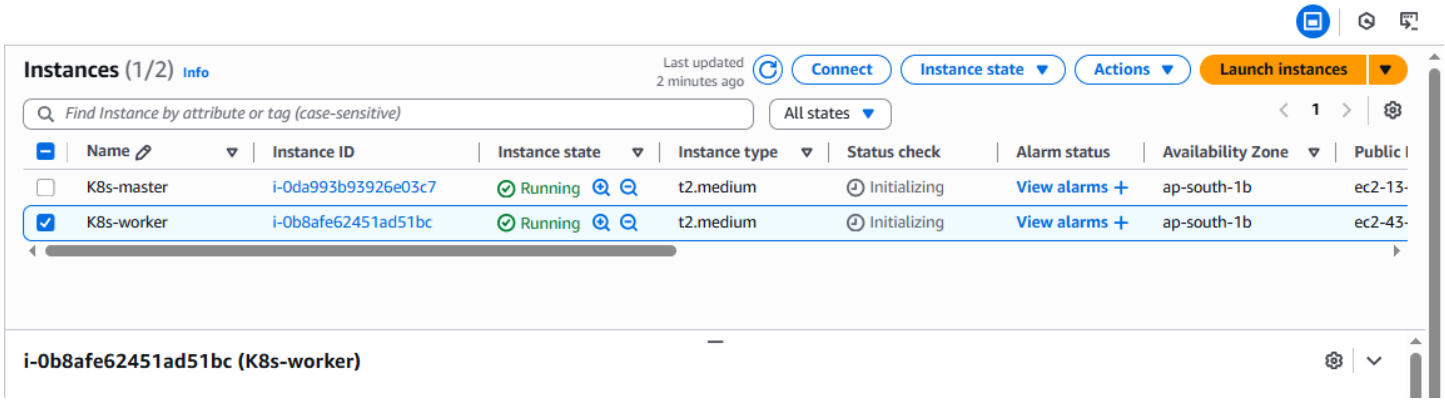
Launch ec2 instance – 2 nos(1 master node and 1 worker-node)

T2 medium

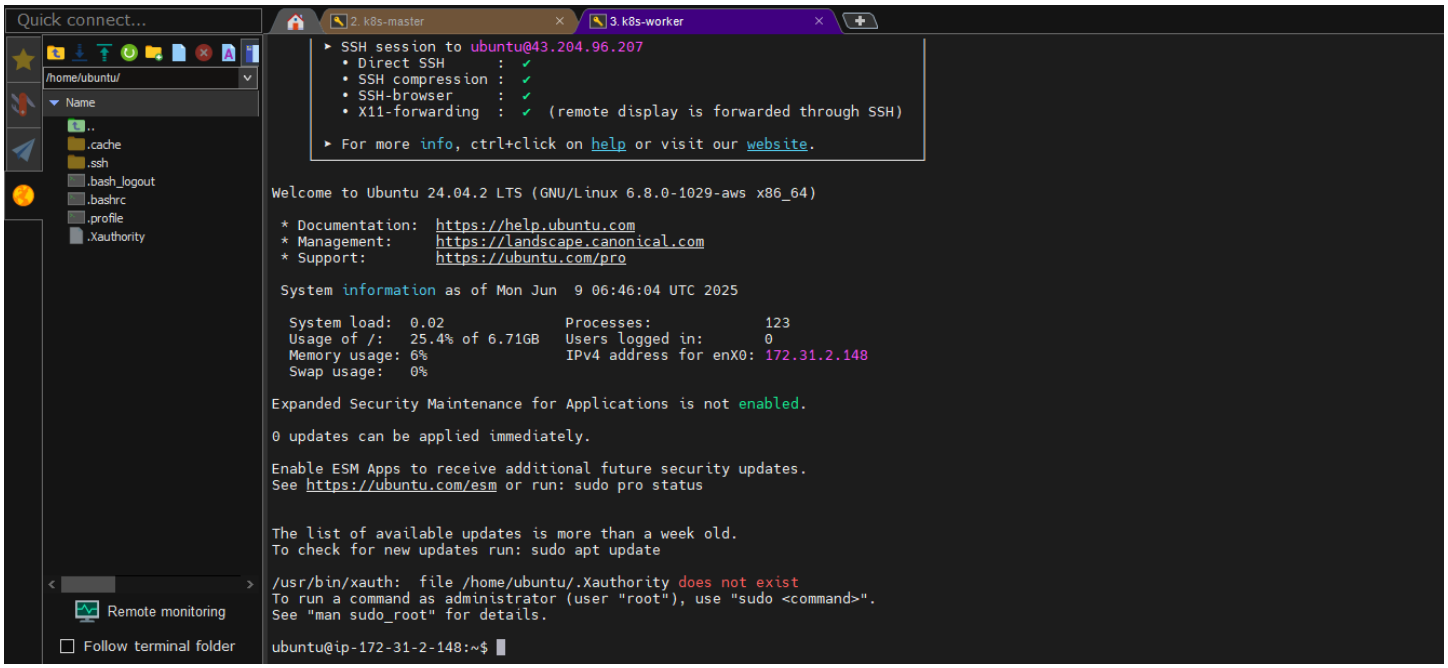
Ubuntu 22.04 lts

Storage :8 gb

K8s-master-name of the server



Then I have just launched the instance and connect to the mobaxterm, as shown below.



Go for the github (I have forked whole branch)as the main branch code is not working

This is the rectified version-----

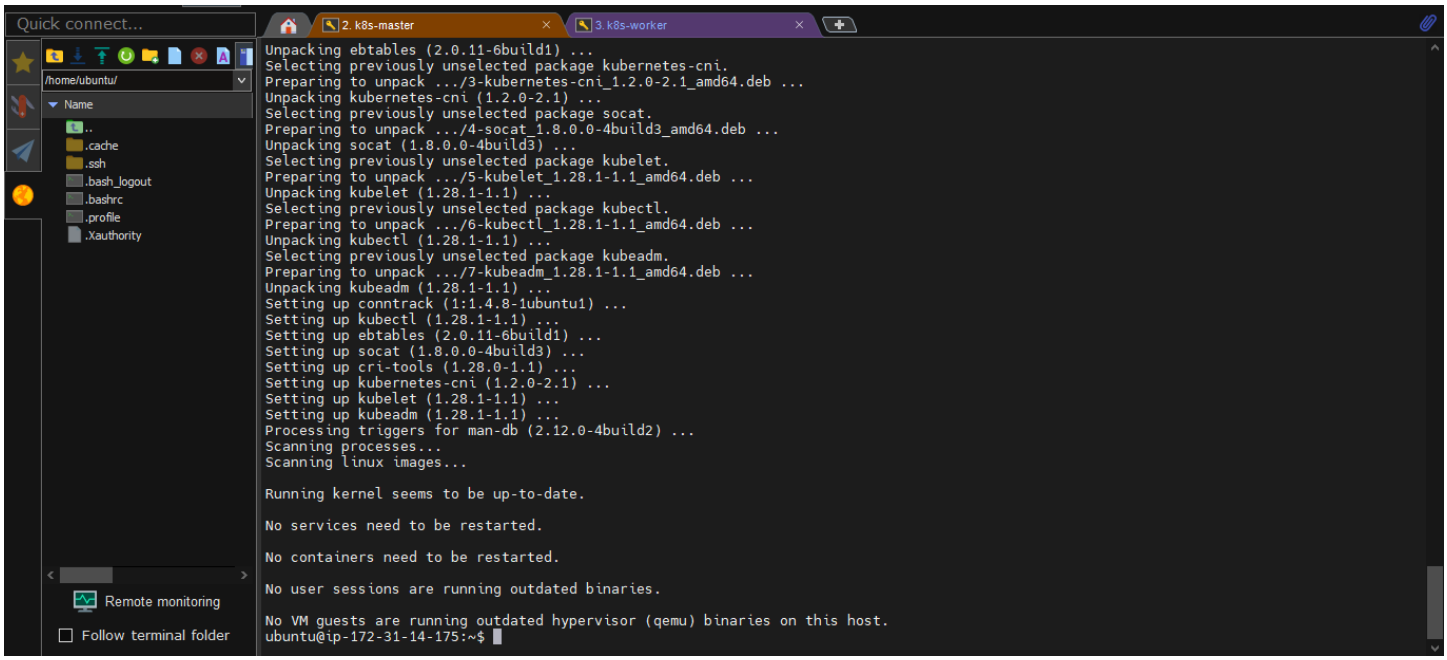
https://github.com/devops-methodology/kubestarter/tree/DevMadhup-patch-2

just go to DevMadhup-patch-2----it's a rectified version

Both Master & Worker Node

Run the following commands on both the master and worker nodes to prepare them for kubeadm.

```
# using 'sudo su' is not a good practice.
sudo apt update
sudo apt-get install -y apt-transport-https ca-certificates curl
sudo apt install docker.io -y

sudo systemctl enable --now docker # enable and start in single command.

# Adding GPG keys.
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernet

# Add the repository to the sourcelist.
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' |

sudo apt update
sudo apt install -y kubeadm=1.28.1-1.1 kubelet=1.28.1-1.1 kubectl=1.28.1-1.1 -y
```



Paste the command both in worker node and master node

 # disable swap

sudo swapoff -a

 # Create the .conf file to load the modules at bootup

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay

br_netfilter

EOF

sudo modprobe overlay

sudo modprobe br_netfilter

 # sysctl params required by setup, params persist across reboots

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ip6tables = 1

net.ipv4.ip_forward = 1

```
EOF

# Apply sysctl params without reboot
sudo sysctl --system

## Install CRIO Runtime
sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri-o.list

sudo apt-get update -y
sudo apt-get install -y cri-o

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service

echo "CRI runtime installed successfully"

# Add Kubernetes APT repository and install required packages
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update -y
```

sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"

sudo apt-get update -y

sudo apt-get install -y jq

sudo systemctl enable --now kubelet

sudo systemctl start kubelet

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.14.175:6443 --token cuyx0i.4qyza0bbw4t2wyq5 \
        --discovery-token-ca-cert-hash sha256:aff85552c28003c6cb0fe796204a08f9effd16b390d31409f77b1104b8f0fbd6
ubuntu@ip-172-31-14-175:~$  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-14-175:~$
```

```
ubuntu@ip-172-31-14-175:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE   VERSION
ip-172-31-14-175    NotReady   control-plane   10m   v1.28.1
ubuntu@ip-172-31-14-175:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE   VERSION
ip-172-31-14-175    NotReady   control-plane   10m   v1.28.1
ubuntu@ip-172-31-14-175:~$
```

Then in master node paste the command

sudo kubeadm config images pull

sudo kubeadm init

mkdir -p "$HOME"/.kube

sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config

sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config

 # Network Plugin = calico

kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

kubeadm token create --print-join-command

then

```
ubuntu@ip-172-31-14-175:~$ kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
ubuntu@ip-172-31-14-175:~$
```

After that in worker node paste the command

sudo kubeadm reset pre-flight checks –v=5 (In worker node) in case you have write the command in worker node as

sudo kubeadm init

```
ubuntu@ip-172-31-2-148:~$ sudo kubeadm reset pre-flight checks --v=5
Found multiple CRI endpoints on the host. Please define which one do you wish to use by setting the 'criSocket' field in the kubeadm c
iguration file: unix:///var/run/containerd/containerd.sock, unix:///var/run/crio/crio.sock
k8s.io/kubernetes/cmd/kubeadm/app/util/runtime.detectCRISocketImpl
        cmd/kubeadm/app/util/runtime/runtime.go:167
k8s.io/kubernetes/cmd/kubeadm/app/util/runtime.DetectCRISocket
        cmd/kubeadm/app/util/runtime/runtime.go:175
k8s.io/kubernetes/cmd/kubeadm/app/util/config.SetResetDynamicDefaults
        cmd/kubeadm/app/util/config/resetconfiguration.go:47
k8s.io/kubernetes/cmd/kubeadm/app/util/config.DefaultedResetConfiguration
        cmd/kubeadm/app/util/config/resetconfiguration.go:154
k8s.io/kubernetes/cmd/kubeadm/app/util/config.LoadOrDefaultResetConfiguration
        cmd/kubeadm/app/util/config/resetconfiguration.go:74
k8s.io/kubernetes/cmd/kubeadm/app/cmd.newResetData
        cmd/kubeadm/app/cmd/reset.go:111
k8s.io/kubernetes/cmd/kubeadm/app/cmd.newCmdReset.func2
        cmd/kubeadm/app/cmd/reset.go:240
k8s.io/kubernetes/cmd/kubeadm/app/cmd/phases/workflow.(*Runner).InitData
        cmd/kubeadm/app/cmd/phases/workflow/runner.go:183
k8s.io/kubernetes/cmd/kubeadm/app/cmd.newCmdReset.func1
        cmd/kubeadm/app/cmd/reset.go:207
github.com/spf13/cobra.(*Command).execute
        vendor/github.com/spf13/cobra/command.go:940
github.com/spf13/cobra.(*Command).ExecuteC
        vendor/github.com/spf13/cobra/command.go:1068
github.com/spf13/cobra.(*Command).Execute
        vendor/github.com/spf13/cobra/command.go:992
k8s.io/kubernetes/cmd/kubeadm/app.Run
        cmd/kubeadm/app/kubeadm.go:50
main.main
        cmd/kubeadm/kubeadm.go:25
runtime.main
        /usr/local/go/src/runtime/proc.go:267
runtime.goexit
        /usr/local/go/src/runtime/asm_amd64.s:1650
ubuntu@ip-172-31-2-148:~$
```

```
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
I0609 07:19:01.713234    4176 kubeproxy.go:55] attempting to download the KubeProxyConfiguration from ConfigMap "kube-proxy"
I0609 07:19:01.716125    4176 kubelet.go:74] attempting to download the KubeletConfiguration from ConfigMap "kubelet-config"
I0609 07:19:01.720279    4176 initconfiguration.go:114] skip CRI socket detection, fill with the default CRI socket unix:///var/run/cont
inerd/containerd.sock
I0609 07:19:01.720400    4176 interface.go:432] Looking for default routes with IPv4 addresses
I0609 07:19:01.720408    4176 interface.go:437] Default route transits interface "eth0"
I0609 07:19:01.720506    4176 interface.go:209] Interface eth0 is up
I0609 07:19:01.720558    4176 interface.go:257] Interface "eth0" has 2 addresses :[172.31.7.134/20 fe80::86c:72ff:fe94:6421/64].
I0609 07:19:01.720568    4176 interface.go:224] Checking addr  172.31.7.134/20.
I0609 07:19:01.720576    4176 interface.go:231] IP found 172.31.7.134
I0609 07:19:01.720583    4176 interface.go:263] Found valid IPv4 address 172.31.7.134 for interface "eth0".
I0609 07:19:01.720589    4176 interface.go:443] Found active IP 172.31.7.134
I0609 07:19:01.724135    4176 preflight.go:104] [preflight] Running configuration dependant checks
I0609 07:19:01.724152    4176 controlplaneprepare.go:225] [download-certs] Skipping certs download
I0609 07:19:01.724175    4176 kubelet.go:121] [kubelet-start] writing bootstrap kubelet config file at /etc/kubernetes/bootstrap-kubelet
conf
I0609 07:19:01.724628    4176 kubelet.go:136] [kubelet-start] writing CA certificate at /etc/kubernetes/pki/ca.crt
I0609 07:19:01.725028    4176 kubelet.go:157] [kubelet-start] Checking for an existing Node in the cluster with name "ip-172-31-7-134" a
d status "Ready"
I0609 07:19:01.727038    4176 kubelet.go:172] [kubelet-start] Stopping the kubelet
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
I0609 07:19:02.936066    4176 cert_rotation.go:137] Starting client certificate rotation controller
I0609 07:19:02.936619    4176 kubelet.go:220] [kubelet-start] preserving the crisocket information for the node
I0609 07:19:02.936630    4176 patchnode.go:31] [patchnode] Uploading the CRI Socket information "unix:///var/run/crio/crio.sock" to the
ode API object "ip-172-31-7-134" as an annotation

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

After that the join command which was generated in master node just copy it with sudo (command) –v=5

sudo your-token --v=5

```
ubuntu@ip-172-31-3-198:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-3-198     Ready    control-plane   2m33s   v1.29.0
ip-172-31-7-134     Ready    <none>          49s     v1.29.0
ubuntu@ip-172-31-3-198:~$
```

On master node check

kubectl get nodes

```
ubuntu@ip-172-31-3-198:~$ kubectl run nginx --image=nginx
pod/nginx created
ubuntu@ip-172-31-3-198:~$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          8s
ubuntu@ip-172-31-3-198:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   7m54s
ubuntu@ip-172-31-3-198:~$ kubectl get all
NAME        READY   STATUS    RESTARTS   AGE
pod/nginx   1/1     Running   0          68s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   8m46s
ubuntu@ip-172-31-3-198:~$
```

```
ubuntu@ip-172-31-3-198:~$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          8s
ubuntu@ip-172-31-3-198:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   7m54s
ubuntu@ip-172-31-3-198:~$ kubectl get all
NAME        READY   STATUS    RESTARTS   AGE
pod/nginx   1/1     Running   0          68s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   8m46s
ubuntu@ip-172-31-3-198:~$ kubectl expose deployment nginx --port=80 --type=NodePort
Error from server (NotFound): deployments.apps "nginx" not found
ubuntu@ip-172-31-3-198:~$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
ubuntu@ip-172-31-3-198:~$ kubectl expose deployment nginx --port=80 --type=NodePort
service/nginx exposed
ubuntu@ip-172-31-3-198:~$ kubectl get svc nginx
NAME    TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
nginx   NodePort   10.106.100.228   <none>        80:30700/TCP   8s
ubuntu@ip-172-31-3-198:~$ curl ifconfig.me
13.203.200.180ubuntu@ip-17kubectl get pods -o widet pods -o wide
NAME                    READY   STATUS    RESTARTS   AGE     IP               NODE               NOMINATED NODE   READINESS GATES
nginx                   1/1     Running   0          5m15s   192.168.36.65    ip-172-31-7-134    <none>           <none>
nginx-7854ff8877-kgv2p  1/1     Running   0          2m2s    192.168.36.66    ip-172-31-7-134    <none>           <none>
ubuntu@ip-172-31-3-198:~$ kubectl get nodes
NAME             STATUS   ROLES           AGE   VERSION
ip-172-31-3-198  Ready    control-plane   15m   v1.29.0
ip-172-31-7-134  Ready    <none>          13m   v1.29.0
ubuntu@ip-172-31-3-198:~$ kubectl get svc nginx
NAME    TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
nginx   NodePort   10.106.100.228   <none>        80:30700/TCP   6m47s
ubuntu@ip-172-31-3-198:~$ kubectl get pods -o wide
NAME                    READY   STATUS    RESTARTS   AGE     IP               NODE               NOMINATED NODE   READINESS GATES
nginx                   1/1     Running   0          10m     192.168.36.65    ip-172-31-7-134    <none>           <none>
nginx-7854ff8877-kgv2p  1/1     Running   0          7m15s   192.168.36.66    ip-172-31-7-134    <none>           <none>
ubuntu@ip-172-31-3-198:~$
```

As I have used crio so these command refers to this instead of docker commands used this in episode-4 (as the docker command which was in the repository throwing some error).

For external reference:

Task	CRI-O Command

Task	CRI-O Command
List containers	`sudo crictl ps -a`
View image list	`sudo crictl images`
View logs for container	`sudo crictl logs <container-id>`
Run exec in container	`sudo crictl exec -it <id> sh`

As it is running through cri

Kubernetes Deployment of 2-Tier Application for DevOps Engineers | Episode 4

To run an application we need 5 most important things from that

1/Pod:

Why we are using pod lets say we have multiple services to deploy if we are deploying manually mainly it takes time unconditionally and may be some manual error and to check or maintain its too difficult and lets say we have to sudden scale up or scale down we can't that where pod come into play we can create application inside k8s cluster and scale up and scale down easily with easily maintaining it .

Pod is the smallest unit of k8s cluster.

In the pod k8s internally uses CRI(container runtime interface) so we can use any type of container inside k8s cluster.

Pod is the abstraction layer of container.

We can create a multiple pod –which we called scaling.

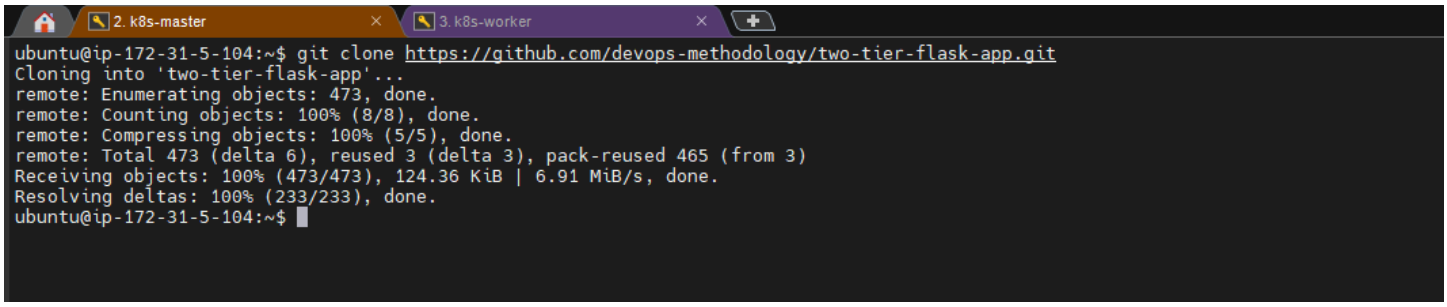
Deployment :*it's a blueprint of creating of pod.*

Pod is epiphermal in nature it can easily destroyed if there is some wrong and to connect the pod with another pod it has some ip but after the pod is destroyed the ip will be gone .

So here the service come into play where it has static ip wheather the pod is deleted it has the service ip which is will not affect in the communication.

Anything we want to do we have to create a yaml(yet another markup language) file or manifest file.

git clone <> in master node



We have to create two-tier-app-pod.yml

```
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$ vi two-tier-app-pod.yml
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$ kubectl apply -f two-tier-app-pod.yml
pod/two-tier-app-pod created
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$
```

Just created two-tier-pod.yml

```
apiVersion: v1

kind: Pod

metadata:

  name: two-tier-app-pod

spec:

  containers:

  - name: two-tier-app-pod

    image: trainwithshubham/flaskapp:latest

    env:

      - name: MYSQL_HOST

        value: "10.98.19.211"        # this is your mysql's service clusture IP, Make sure to change it with yours

      - name: MYSQL_PASSWORD

        value: "admin"

      - name: MYSQL_USER

        value: "root"

      - name: MYSQL_DB

        value: "mydb"

    ports:

      - containerPort: 5000

    imagePullPolicy: Always
```

```
pod/two-tier-app-pod created
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
two-tier-app-pod    1/1     Running   0          19s
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: two-tier-app-pod
spec:
  containers:
  - name: two-tier-app-pod
    image: premd91/flaskapp:latest
    env:
      - name: MYSQL_HOST
        value: "10.98.19.211"          # this is your mysql's service clusture IP, Make sure to change it with yours
      - name: MYSQL_PASSWORD
        value: "admin"
      - name: MYSQL_USER
        value: "root"
      - name: MYSQL_DB
        value: "mydb"
    ports:
      - containerPort: 5000
    imagePullPolicy: Always

~
~
~
~
~
~
~
~
~
```

```
ubuntu@ip-172-31-3-137:~/two-tier-flask-app/k8s$ kubectl get pods
NAME                READY   STATUS              RESTARTS   AGE
two-tier-app-pod    0/1     ContainerCreating   0          4s
ubuntu@ip-172-31-3-137:~/two-tier-flask-app/k8s$
```

sudo crictl ps

```yaml
kind: Pod
metadata:
  name: two-tier-app-pod

spec:
  containers:
    - name: two-tier-app-ctr
      image: trainwithshubham/flaskapp:latest
      env:
        - name: MYSQL_HOST
          value: "mysql"
        - name: MYSQL_USER
          value: "root"
        - name: MYSQL_PASSWORD
          value: "admin"
        - name: MYSQL_DB
          value: "mydb"
      ports:
        - containerPort: 5000
      imagePullPolicy: Always
```

kubectl apply –f two-tier-app-pod.yml

now we have to scale this pod so we will create deployment file

controllers/nginx-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Accord to this deployment file we tend to know that 1/ if we want to scale app: nginx then we can use.

*2/lets say there are lots of pods means different application is working its hard to know to the kubernetes that which service or resource it will take into consideration in that case **labels** and **selector** comes into play. Accord to that it will pod will map with his service or any other resources **matchLabels**.*

*3/there is also an **template** which is refer to the pod template it will go for that app likewise **lables** app:nginx*

*4/if **metadata name app:nginx** and **template name app:ginx** then **matchLabels app:nginx***

5/accord to that in which pod the matchLabels will same it will automatic replicas:3 will be done in that pod or auto scale up accord to that.

After deployment if one pod is crashed then it automatically startsup that's called autohealing and we can easily scale up and scale down through cli.

kubectl scale deployment two-tier-app-deployment –replicas=2

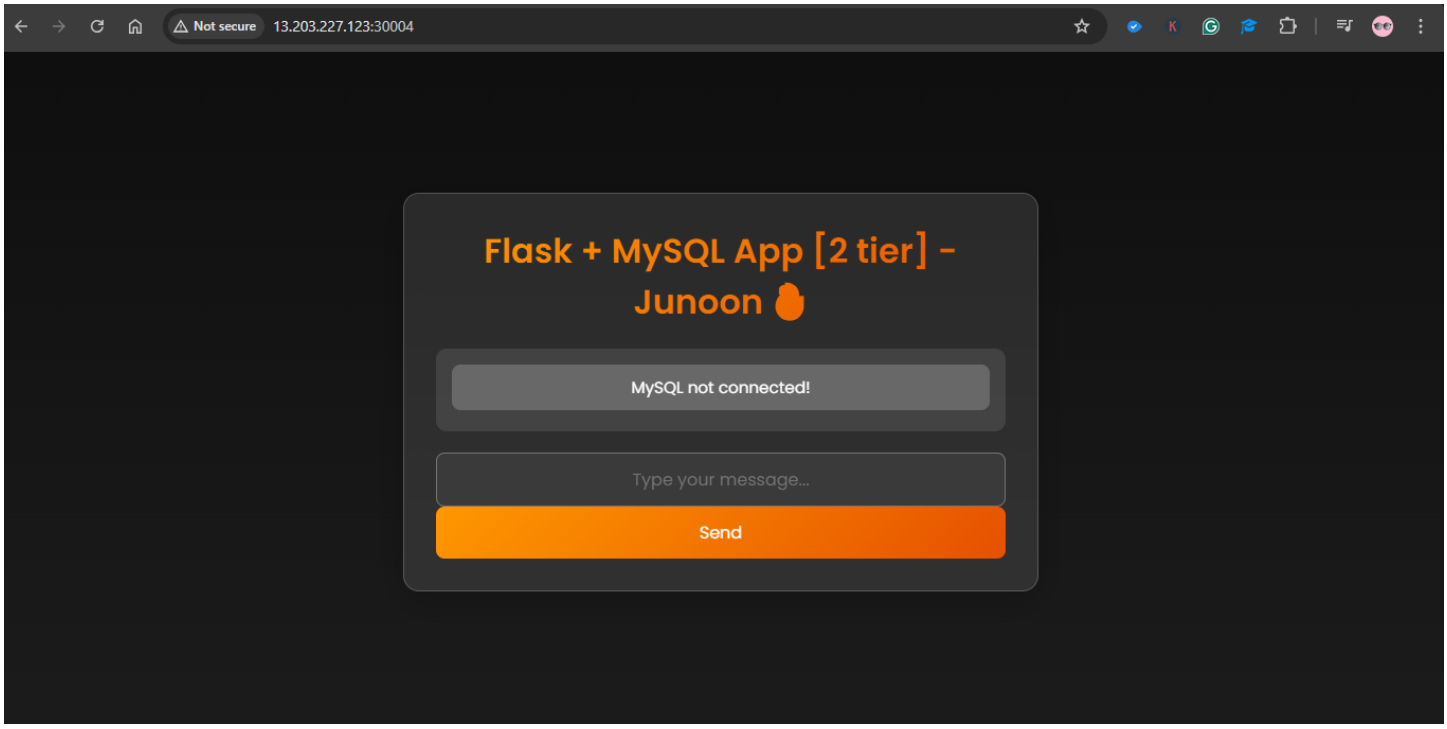
if we want to access as a user we can't in which port we can access,,,,we can access then we have to go through ip and etc but its not the properway..when we visit an website we don't go for ip address and visit service will do that…

lets say in k8s cluster two pod is communicating through ip address lets say one pod is crashed then the ip address also vanished we have to again configure that ip with that respective pod ,…instead of doing that we can use service which has static ip when ever a pod is crashed or anything failed automatically new pod will be created but the static ip of that pod provided by service will be remain there.

*In Kubernetes, a **Service** is an abstraction that exposes a set of Pods as a network service, enabling communication between components or external clients, with types like **ClusterIP** (default, internal access), **NodePort** (exposes service on a static port on each node), and **LoadBalancer** (provisions an external load balancer)*

```
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$ kubectl get svc
NAME                  TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes            ClusterIP   10.96.0.1      <none>        443/TCP        40m
two-tier-app-service  NodePort    10.107.9.35    <none>        80:30004/TCP   6m17s
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$
```

```
deployment.apps/two-tier-app created
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
hello-world-pod               1/1     Running   0          13m
two-tier-app-ff6b57c8c-7b6d7  1/1     Running   0          15s
two-tier-app-pod              1/1     Running   0          29m
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$ vi two-tier-app-deployment.yml
```



Create mysql showing flaskapp is not connected with mysql

```
ubuntu@ip-172-31-3-137:~/two-tier-flask-app/k8s$ vi mysql-deployment.yml
ubuntu@ip-172-31-3-137:~/two-tier-flask-app/k8s$ kubectl apply -f mysql-deployment.yml
deployment.apps/mysql created
ubuntu@ip-172-31-3-137:~/two-tier-flask-app/k8s$
```

Mysql deployment

```
ubuntu@ip-172-31-3-137:~/two-tier-flask-app/k8s$ kubectl get deployment
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
mysql           0/1     1            0           50s
two-tier-app    1/1     1            1           10m
ubuntu@ip-172-31-3-137:~/two-tier-flask-app/k8s$
```


Before that we have to create directory in two-tier-flask-app location

Persistent volume-from the file system it will give some storage and will give to the database.

So before creating mysql deployment we have to create persistent volume and pvc and then mysql service

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv
spec:
  capacity:
    storage: 256Mi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: /home/ubuntu/two-tier-flask-app/mysqldata       #This is your host path where your data will be stored. Make sure to create my
sqldata directory in mentioned path
~
```

2. k8s-master 3. k8s-worker

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 256Mi
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

```
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$ kubectl get svc
NAME                   TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes             ClusterIP   10.96.0.1      <none>        443/TCP        50m
mysql                  ClusterIP   10.106.228.4   <none>        3306/TCP       5s
two-tier-app-service   NodePort    10.107.9.35    <none>        80:30004/TCP   15m
ubuntu@ip-172-31-5-104:~/two-tier-flask-app/k8s$ vi two-tier-app-deployment.yml
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: two-tier-app
  labels:
    app: two-tier-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: two-tier-app
  template:
    metadata:
      labels:
        app: two-tier-app
    spec:
      containers:
        - name: two-tier-app
          image: premd91/flaskapp:latest
          env:
            - name: MYSQL_HOST
              value: "10.106.228.4"          # this is your mysql's service clusture IP, Make sure to change it with yours
            - name: MYSQL_PASSWORD
              value: "admin"
            - name: MYSQL_USER
              value: "root"
            - name: MYSQL_DB
              value: "mydb"
          ports:
            - containerPort: 5000
          imagePullPolicy: Always
~
~
~
~
~
~
"two-tier-app-deployment.yml" 31L, 771B                                                                                    22,34
```

```
ubuntu@ip-172-31-3-137:~/two-tier-flask-app/k8s$ kubectl apply -f two-tier-app-deployment.yml
deployment.apps/two-tier-app configured
ubuntu@ip-172-31-3-137:~/two-tier-flask-app/k8s$
```

sudo crictl ps

```
ubuntu@ip-172-31-8-125:~$ sudo crictl ps
CONTAINER           IMAGE                                                                                                   CREATED
    STATE               NAME                ATTEMPT             POD ID              POD
bb2acdf8f66e4       docker.io/premd91/flaskapp@sha256:db1f8bb689c28cf90607b7290655e8b44e68a35bba8717b4ca80b05f3b8857ab   2 minutes ago
    Running             two-tier-app        0                   bd1c604627887       two-tier-app-6d784c7955-65nfb
c98d518826ab9       docker.io/library/mysql@sha256:6432d412b5f7132d4a5a4a1cf5d7cba9aa4cbe0ae4da5e11ab37c65fbe5ae702      9 minutes ago
    Running             mysql               0                   951c3b0c0c800       mysql-5479cbccb8-2vswn
186cbbbf29591       docker.io/premd91/flaskapp@sha256:db1f8bb689c28cf90607b7290655e8b44e68a35bba8717b4ca80b05f3b8857ab   32 minutes ago
    Running             two-tier-app-pod    1                   ded9d538bc2cd       two-tier-app-pod
74bd2e79b5f6c       44f52c09dececf0d842450cfbdcf6f1ce1e6eaf2d7183d643b9fbf77dde03a38                                      46 minutes ago
    Running             calico-node         0                   a9f00d944f57e       calico-node-m9cnq
76eac8c68a738       registry.k8s.io/kube-proxy@sha256:f6074f465fb3700456dccc5915340df4b59a7960c591693182fbd297cbe72b53   46 minutes ago
    Running             kube-proxy          0                   810341e28c60c       kube-proxy-x788x
ubuntu@ip-172-31-8-125:~$ kubectl exec -it mysql-5479cbccb8-2vswn /bin/bash
```

To go inside the mysql container

sudo crictl exec –it <container id> /bin/bash (as I am using container run time not docker run time)

```
ubuntu@ip-172-31-8-125:~$ sudo crictl exec -it c98d51882 /bin/sh
sh-5.1#
```

Mysql –u root –p

Password: admin

Then inside

show databases;

use mydb;

create the table message for unwanted error.

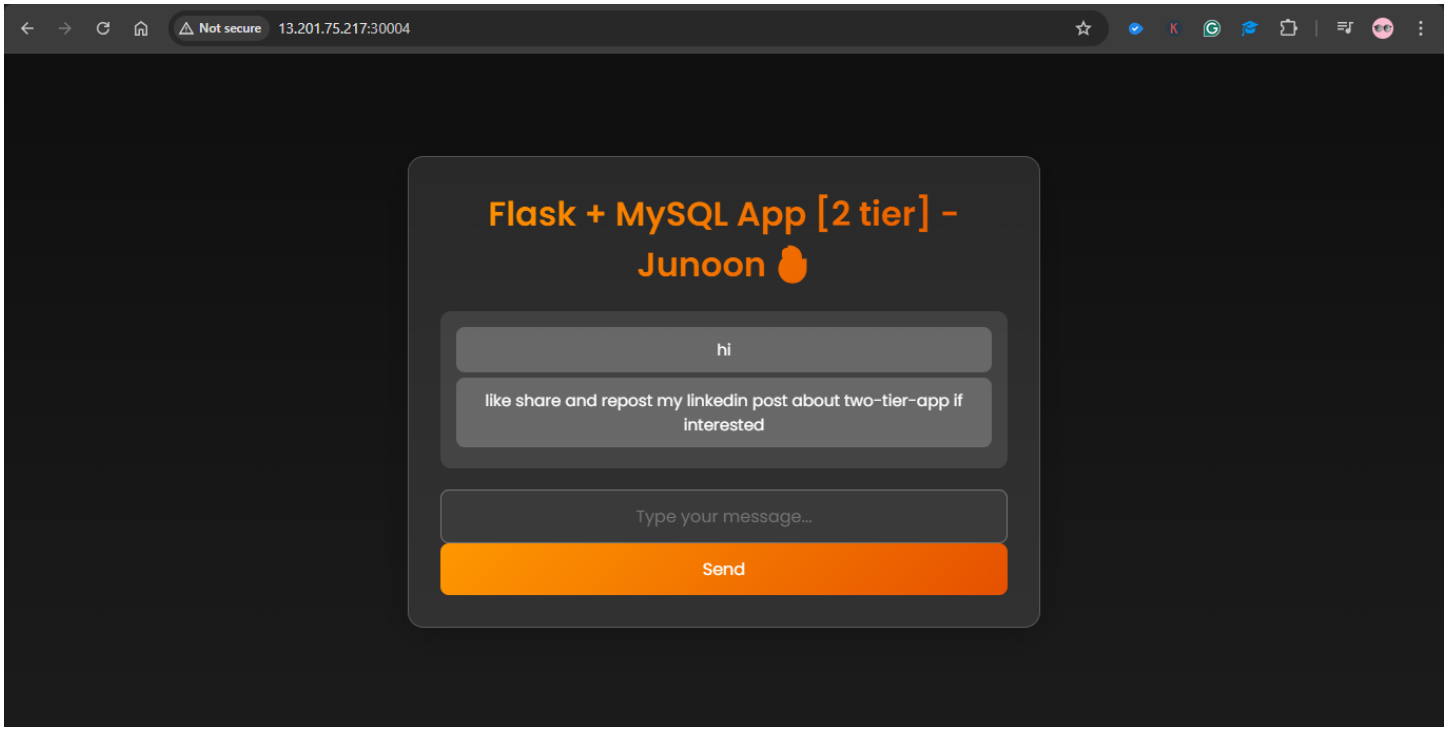
```
CREATE TABLE messages (

   id INT AUTO_INCREMENT PRIMARY KEY,

   message TEXT

);
```

```
ubuntu@ip-172-31-8-125:~$ sudo crictl exec -it c98d51882 /bin/sh
sh-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 9.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, 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>
```

```
ubuntu@ip-172-31-8-125:~$ sudo crictl exec -it c98d51882 /bin/sh
sh-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 9.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, 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> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mydb               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.004 sec)

mysql> use mydb;
Database changed
mysql>
```

```
5 rows in set (0.004 sec)

mysql> use mydb;
Database changed
mysql> CREATE TABLE messages (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     message TEXT
    -> );
Query OK, 0 rows affected (0.028 sec)

mysql>
```

Then just worker node instance ip :30004 you will find the flask-app with its backend is connected with mysql



You can check the messages in mysql server

By using this cli

Select * from messages;

```
mysql> use mydb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from messages;
+----+-------------------------------------------------------------------------+
| id | message                                                                 |
+----+-------------------------------------------------------------------------+
|  1 | hi                                                                      |
|  2 | like share and repost my linkedin post about two-tier-app if interested |
+----+-------------------------------------------------------------------------+
2 rows in set (0.000 sec)

mysql>
```

HELM Packaging of Two-Tier Applications for DevOps Engineers | Episode 5

When we do deployments through kubernetes we have to do lots of manifests files for deployments, services, databases etc..its hard to manage all this so we use helm as a package manager which will deploy all these yaml files one files so that we have not to worry to do individually.

*Through **helm** we can create template so that later we can use also.*

How to Install helm in Ubuntu

```
curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null
sudo apt-get install apt-transport-https --yes
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable
sudo apt-get update
sudo apt-get install helm
```

```
ubuntu@ip-172-31-2-94:~$ helm
The Kubernetes package manager

Common actions for Helm:

- helm search:    search for charts
- helm pull:      download a chart to your local directory to view
- helm install:   upload the chart to Kubernetes
- helm list:      list releases of charts

Environment variables:

| Name                                | Description
|
|-------------------------------------|-----------------------------------------------------------------------------------------
---------|
| $HELM_CACHE_HOME                    | set an alternative location for storing cached files.
|
| $HELM_CONFIG_HOME                   | set an alternative location for storing Helm configuration.
|
| $HELM_DATA_HOME                     | set an alternative location for storing Helm data.
|
| $HELM_DEBUG                         | indicate whether or not Helm is running in Debug mode
|
| $HELM_DRIVER                        | set the backend storage driver. Values are: configmap, secret, memory, sql.
|
| $HELM_DRIVER_SQL_CONNECTION_STRING | set the connection string the SQL storage driver should use.
|
| $HELM_MAX_HISTORY                   | set the maximum number of helm release history.
|
| $HELM_NAMESPACE                     | set the namespace used for the helm operations.
|
| $HELM_NO_PLUGINS                    | disable plugins. Set HELM_NO_PLUGINS=1 to disable plugins.
|
| $HELM_PLUGINS                       | set the path to the plugins directory
|
```

***Helm** is also referred as package manager for kubernetes. It enables you to define, install and manage k8s applications. **Helm** uses a packaging format called charts, which include all the resources needed to run the application.*

In chart -chart.yml

- *charts*
- *templates*
- *values.yml we will find*

-

```
ubuntu@ip-172-31-2-94:~$ ls
nginx-chart
ubuntu@ip-172-31-2-94:~$ cd nginx-chart
ubuntu@ip-172-31-2-94:~/nginx-chart$ ls
Chart.yaml  charts  templates  values.yaml
ubuntu@ip-172-31-2-94:~/nginx-chart$
```

chart-details about the application regarding version, names, app version its all about metadata.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "nginx-chart.fullname" . }}
  labels:
    {{- include "nginx-chart.labels" . | nindent 4 }}
spec:
  {{- if not .Values.autoscaling.enabled }}
  replicas: {{ .Values.replicaCount }}
  {{- end }}
  selector:
    matchLabels:
      {{- include "nginx-chart.selectorLabels" . | nindent 6 }}
  template:
    metadata:
      {{- with .Values.podAnnotations }}
      annotations:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      labels:
        {{- include "nginx-chart.labels" . | nindent 8 }}
        {{- with .Values.podLabels }}
```

This is the template of deployment file.

Also there is values.yaml when we change in values.yaml this values will direct inject into template.

Practical

```
ubuntu@ip-172-31-2-94:~$ mkdir two-tier-app
ubuntu@ip-172-31-2-94:~$ ls
two-tier-app
ubuntu@ip-172-31-2-94:~$ cd two-tier-app/
ubuntu@ip-172-31-2-94:~/two-tier-app$ ls
ubuntu@ip-172-31-2-94:~/two-tier-app$ helm create mysql-chart
Creating mysql-chart
ubuntu@ip-172-31-2-94:~/two-tier-app$
```

mkdir two-tier-app

cd <>

helm create mysql-chart

```
ubuntu@ip-172-31-2-94:~/two-tier-app$ helm create mysql-chart
Creating mysql-chart
ubuntu@ip-172-31-2-94:~/two-tier-app$ ls
mysql-chart
ubuntu@ip-172-31-2-94:~/two-tier-app$ cd mysql-chart/
ubuntu@ip-172-31-2-94:~/two-tier-app/mysql-chart$ ls
Chart.yaml  charts  templates  values.yaml
ubuntu@ip-172-31-2-94:~/two-tier-app/mysql-chart$
```

cd mysql-chart/

by default nginx-chart is there we have to change the values in values.yaml

we have changed in values.yaml

image:mysql

tag:latest

service port:3306

```
2. k8s-master    3. k8s-worker
  # Specifies whether a service account should be created
  create: true
  # Automatically mount a ServiceAccount's API credentials?
  automount: true
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name: ""

# This is for setting Kubernetes Annotations to a Pod.
# For more information checkout: https://kubernetes.io/docs/concepts/overview/working-with-objects/annotat
podAnnotations: {}
# This is for setting Kubernetes Labels to a Pod.
# For more information checkout: https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/
podLabels: {}

podSecurityContext: {}
  # fsGroup: 2000

securityContext: {}
  # capabilities:
  #   drop:
  #   - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000

# This is for setting up a service more information can be found here: https://kubernetes.io/docs/concepts/
service:
  # This sets the service type more information can be found here: https://kubernetes.io/docs/concepts/serv
shing-services-service-types
  type: ClusterIP
  # This sets the ports more information can be found here: https://kubernetes.io/docs/concepts/services-ne
orts
  port: 3306

-- INSERT --
```

after that we can't use fully mysql we have to use env variables for mysql as we have done in previous project

```
2. k8s-master    3. k8s-worker
      labels:
        {{- include "mysql-chart.labels" . | nindent 8 }}
        {{- with .Values.podLabels }}
        {{- toYaml . | nindent 8 }}
        {{- end }}
    spec:
      {{- with .Values.imagePullSecrets }}
      imagePullSecrets:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      serviceAccountName: {{ include "mysql-chart.serviceAccountName" . }}
      {{- with .Values.podSecurityContext }}
      securityContext:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      containers:
        - name: {{ .Chart.Name }}
          {{- with .Values.securityContext }}
          securityContext:
            {{- toYaml . | nindent 12 }}
          {{- end }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: "admin"
            - name: MYSQL_DATABASE
              value: "mydb"
            - name: MYSQL_USER
              value: "admin"
            - name: MYSQL_PASSWORD
              value: "admin"
          ports:
            - name: http
              containerPort: {{ .Values.service.port }}
              protocol: TCP
          {{- with .Values.livenessProbe }}
-- INSERT --                                                    51,29          38%
```

We don’t have to hardcode in values.yaml

```
image:
  repository: mysql
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVer
  tag: "lates"

env:
  mysqlrootpw: admin
  mysqldb: mydb
  mysqluser: admin
  mysqlpass: admin

imagePullSecrets: []
nameOverride: ""
fullnameOverride: ""

serviceAccount:
  # Specifies whether a service account should be created
  create: true
  # Automatically mount a ServiceAccount's API credentials?
  automount: true
  # Annotations to add to the service account
```

The values we have created under values.yaml we have to inject in that deployment.yaml file

```
2. k8s-master    3. k8s-worker
# Default values for mysql-chart.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

# This will set the replicaset count more information can be found here: https://kubernetes.io/docs/concepts/workloads/controllers/replic
aset/
replicaCount: 1

# This sets the container image more information can be found here: https://kubernetes.io/docs/concepts/containers/images/
image:
  repository: mysql
  # This sets the pull policy for images.
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  tag: "latest"
env:
  mysqlrootpw: admin
  mysqldb: mydb
  mysqluser: admin
  mysqlpass: admin
# This is for the secrets for pulling an image from a private repository more information can be found here: https://kubernetes.io/docs/t
asks/configure-pod-container/pull-image-private-registry/
imagePullSecrets: []
# This is to override the chart name.
nameOverride: ""
fullnameOverride: ""

# This section builds out the service account more information can be found here: https://kubernetes.io/docs/concepts/security/service-ac
counts/
serviceAccount:
  # Specifies whether a service account should be created
  create: true
  # Automatically mount a ServiceAccount's API credentials?
  automount: true
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
-- INSERT --                                                          19,14          Top
```

{{.Values.env.mysqlrootpw}} –like this wise so in deployment.yaml so that when ever we want to change any values it automatically inject the required values in deployment.yaml file.

```
securityContext:
  {{- toYaml .Values.podSecurityContext | nindent 8 }}
containers:
  - name: {{ .Chart.Name }}
    securityContext:
      {{- toYaml .Values.securityContext | nindent 12 }}
    image: "{{ .Values.image.repository }}:{{ .Values.image.

    imagePullPolicy: {{ .Values.image.pullPolicy }}
    env:
      - name: MYSQL_ROOT_PASSWORD
        value: {{ .Values.env.mysqlrootpw }}
      - name: MYSQL_DATABASE
        value: "mydb"
      - name: MYSQL_USER
        value: "admin"
      - name: MYSQL_PASSWORD
        value: "admin"
    ports:
      - name: http
        containerPort: {{ .Values.service.port }}
        protocol: TCP
```

In the top pic was by TWS –shubham sir,,,,…but in my case it was bit different,I have just screenshot in below you can see the image……..

```
        {{- toYaml . | nindent 8 }}
        {{- end }}
    spec:
      {{- with .Values.imagePullSecrets }}
      imagePullSecrets:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      serviceAccountName: {{ include "mysql-chart.serviceAccountName
      {{- with .Values.podSecurityContext }}
      securityContext:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      containers:
        - name: {{ .Chart.Name }}
          {{- with .Values.securityContext }}
          securityContext:
            {{- toYaml . | nindent 12 }}
          {{- end }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.ta
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: {{ .Values.env.mysqlrootpw }}
            - name: MYSQL_DATABASE
              value: {{ .Values.env.mysqldb" }}
            - name: MYSQL_USER
              value: {{ .Values.env.mysqluser" }}
            - name: MYSQL_PASSWORD
              value: {{ .Values.env.mysqlpass" }}
          ports:
            - name: http
              containerPort: {{ .Values.service.port }}
              protocol: TCP
          {{- with .Values.livenessProbe }}
          livenessProbe:
            {{- toYaml . | nindent 12 }}
          {{- end }}
-- INSERT --
```

After that

helm package mysql-chart

```
ubuntu@ip-172-31-2-94:~/two-tier-app$ helm package mysql-chart
Successfully packaged chart and saved it to: /home/ubuntu/two-tier-app/mysql-chart-0.1.0.tgz
ubuntu@ip-172-31-2-94:~/two-tier-app$
```

helm install mysql-chart ./mysql-chart

```
ubuntu@ip-172-31-13-204:~/two-tier-app$ ls
mysql-chart  mysql-chart-0.1.0.tgz
ubuntu@ip-172-31-13-204:~/two-tier-app$ helm install mysql-chart ./mysql-chart
NAME: mysql-chart
LAST DEPLOYED: Fri Jun 13 08:54:41 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=mysql-chart,app.kubernetes.io/instance=mysql-chart" -
o jsonpath="{.items[0].metadata.name}")
  export CONTAINER_PORT=$(kubectl get pod --namespace default $POD_NAME -o jsonpath="{.spec.containers[0].ports[0].containerPort}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace default port-forward $POD_NAME 8080:$CONTAINER_PORT
ubuntu@ip-172-31-13-204:~/two-tier-app$
```

```
ubuntu@ip-172-31-13-204:~/two-tier-app$ kubectl get all
NAME                                READY   STATUS             RESTARTS      AGE
pod/mysql-chart-5c878b76cb-29k7w    0/1     CrashLoopBackOff   4 (20s ago)   3m1s

NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/kubernetes     ClusterIP   10.96.0.1       <none>        443/TCP    19m
service/mysql-chart    ClusterIP   10.111.251.15   <none>        3306/TCP   3m1s

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mysql-chart    0/1     1            0           3m1s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/mysql-chart-5c878b76cb    1         1         0       3m1s
ubuntu@ip-172-31-13-204:~/two-tier-app$
```

But there is some crashloopbackoff error –it repeatedly restarting why because using the liveness and readiness probe which uses the http as a api call so in this case we don't want that so I have just commented it as you can see below in the pic.

```
    {{/* with .Values.livenessProbe */}}
    livenessProbe:
      {{- toYaml . | nindent 12 }}
    {{/* end */ }}
    {{/* with .Values.readinessProbe */}}
    readinessProbe:
      {{- toYaml . | nindent 12 }}
    {{/* end */}}
```

After changing I have uninstalled it

helm uninstall mysql-chart

then again

helm package mysql-chart ./mysql-chart

```
ubuntu@ip-172-31-12-181:~/two-tier-app/mysql-chart$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                   STATUS          CHART                   APP VERS
ON
mysql-chart     default         1               2025-06-13 13:35:57.646739662 +0000 UTC deployed        mysql-chart-0.1.0       1.16.0

ubuntu@ip-172-31-12-181:~/two-tier-app/mysql-chart$ helm uninstall mysql-chart
release "mysql-chart" uninstalled
ubuntu@ip-172-31-12-181:~/two-tier-app/mysql-chart$
```

```
ubuntu@ip-172-31-12-181:~/two-tier-app$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                   STATUS  CHART                   APP VERSION
mysql-chart     default         1               2025-06-13 14:01:12.390074602 +0000 UTC failed  mysql-chart-0.1.0       1.16.0
ubuntu@ip-172-31-12-181:~/two-tier-app$ helm uninstall mysql-chart
release "mysql-chart" uninstalled
ubuntu@ip-172-31-12-181:~/two-tier-app$ helm package mysql-chart
Successfully packaged chart and saved it to: /home/ubuntu/two-tier-app/mysql-chart-0.1.0.tgz
ubuntu@ip-172-31-12-181:~/two-tier-app$ helm install mysql-chart ./mysql-chart
NAME: mysql-chart
LAST DEPLOYED: Fri Jun 13 14:06:39 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=mysql-chart,app.kubernetes.io/instance=mysql-chart" -
o jsonpath="{.items[0].metadata.name}")
  export CONTAINER_PORT=$(kubectl get pod --namespace default $POD_NAME -o jsonpath="{.spec.containers[0].ports[0].containerPort}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace default port-forward $POD_NAME 8080:$CONTAINER_PORT
ubuntu@ip-172-31-12-181:~/two-tier-app$ kubectl get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/mysql-chart-84c99c66b4-vkwwv     1/1     Running   0          36s

NAME                 TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP    45m
service/mysql-chart  ClusterIP   10.105.1.109   <none>        3306/TCP   36s

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mysql-chart     1/1     1            1           36s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/mysql-chart-84c99c66b4     1         1         1       36s
ubuntu@ip-172-31-12-181:~/two-tier-app$
```

Successfully done you can see the pod is running and to go inside the container 1st I want the container id for that I have use **sudo crictl ps** then from the container id I have gone inside the container

```
ubuntu@ip-172-31-6-182:~$ sudo crictl ps
CONTAINER           IMAGE                                                                                                    CREATED
     STATE               NAME                ATTEMPT             POD ID              POD
97cae490e69b0       6af67d37072da6db21cf28ad7413ea19e6ed01ea78e839d7364641518d4eb863                                         About a minu
go   Running             mysql-chart         0                   41bfd94906b74       mysql-chart-84c99c66b4-vkwwv
e370fd01416e4       44f52c09dececf0d842450cfbdcf6f1ce1e6eaf2d7183d643b9fbf77dde03a38                                         45 minutes a
     Running             calico-node         0                   9aa7d9c10a8d7       calico-node-5mfjd
e899bf70a8b93       registry.k8s.io/kube-proxy@sha256:f6074f465fb3700456dccc5915340df4b59a7960c591693182fbd297cbe72b53   45 minutes a
     Running             kube-proxy          0                   cd654590d34d5       kube-proxy-m2dmr
ubuntu@ip-172-31-6-182:~$
```

crictl exec -it <container-id> /bin/bash

After that

mysql –u root –p

password :admin

show databases;

use mydb;

create the table:

CREATE TABLE messages (

id INT AUTO_INCREMENT PRIMARY KEY,

message TEXT

);

```
ubuntu@ip-172-31-6-182:~$ crictl exec -it 97cae490e69b0 /bin/bash
FATA[0000] validate service connection: validate CRI v1 runtime API for endpoint "unix:///var/run/crio/crio.sock": rpc error: code = Unav
ailable desc = connection error: desc = "transport: Error while dialing: dial unix /var/run/crio/crio.sock: connect: permission denied"
ubuntu@ip-172-31-6-182:~$ sudo crictl exec -it 97cae490e69b0 /bin/bash
bash-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, 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> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mydb               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.004 sec)

mysql> use mydb;
Database changed
mysql>
```

Mysql is done

Then we have to create flask-app

So we will create flask-app chart

By writing the cli

helm create flask-app-chart

```
ubuntu@ip-172-31-12-181:~/two-tier-app$ helm create flask-app-chart
Creating flask-app-chart
ubuntu@ip-172-31-12-181:~/two-tier-app$
```

Go to the flask-app-chart

Then same mysql we have to write the env variables in values.yaml and the inject in under deployment.yaml file

```
    imagePullPolicy: {{ .Values.image.pullPolicy }}
    env:
      - name: MYSQL_HOST
        value: {{ .Values.env.mysqlhost }}
      - name: MYSQL_PASSWORD
        value: {{ .Values.env.mysqlpw }}
      - name: MYSQL_USER
        value: {{ .Values.env.mysqluser }}
      - name: MYSQL_DB
        value: {{ .Values.env.mysqldb }}
    ports:
      - name: http
        containerPort: {{ .Values.service.targetPort }}
        protocol: TCP
    {{- with .Values.resources }}
    resources:
      {{- toYaml . | nindent 12 }}
    {{- end }}
    {{- with .Values.volumeMounts }}
    volumeMounts:
      {{- toYaml . | nindent 12 }}
    {{- end }}
{- with .Values.volumes }}
olumes:
 {{- toYaml . | nindent 8 }}
{- end }}
{- with .Values.nodeSelector }}
```

But in service.yaml file of flask-app chart we have to change

The ports

1/port:80 where container port is exposed

2/Target port: 5000(where the application port is exposed)

3/nodePort:where all the pod will grouped and exposed in one port of the node and can be acess to the outer-world through this port.

So we also only give the values as an env

Like wise:

{{ .Values.service.port }}

{{ .Values.service.targetPort }}

{{ .Values.service.nodeport }}

```
2. k8s-master    3. k8s-worker
apiVersion: v1
kind: Service
metadata:
  name: {{ include "flask-app-chart.fullname" . }}
  labels:
    {{- include "flask-app-chart.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: {{ .Values.service.targetPort }}
      nodePort: {{ .Values.service.nodeport }}
      protocol: TCP
      name: http
  selector:
    {{- include "flask-app-chart.selectorLabels" . | nindent 4 }}
~
~
~
~
~
~
~
~
~
~
~
~
~
```

After that kubectl get svc

The mysql-service ip will be copied and paste in deployment file in values under MYSQL_HOST where the app will access the mysql through this service ip: i.e 10.105.1.109

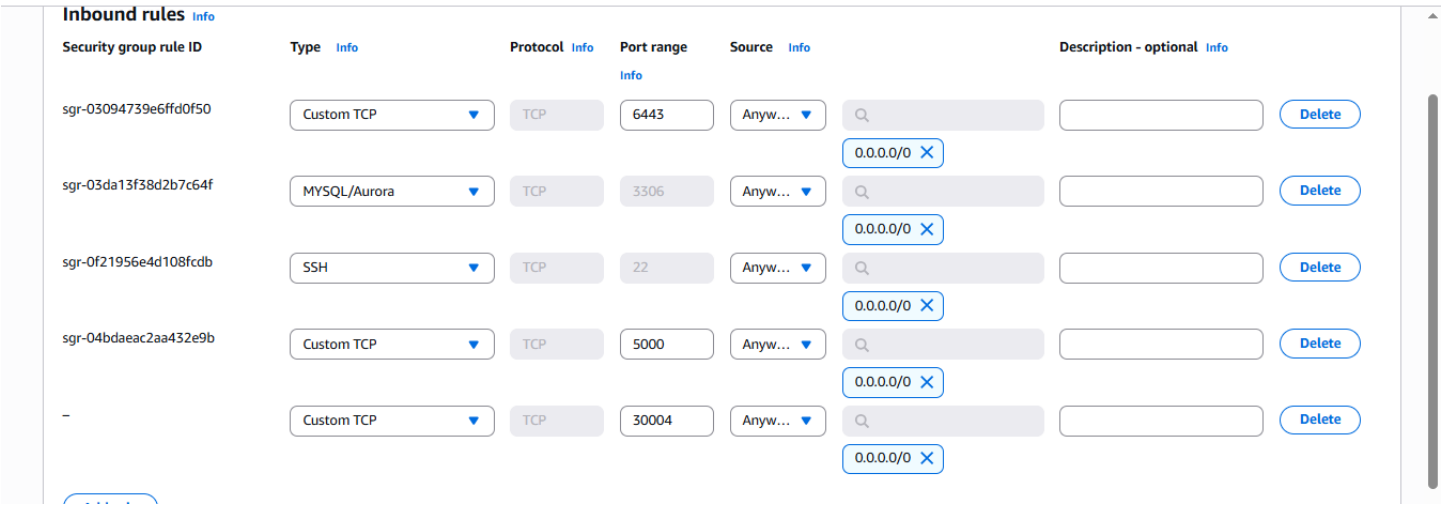
```
ubuntu@ip-172-31-12-181:~/two-tier-app/flask-app-chart$ kubectl get svc
NAME          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes    ClusterIP   10.96.0.1      <none>        443/TCP    53m
mysql-chart   ClusterIP   10.105.1.109   <none>        3306/TCP   9m3s
ubuntu@ip-172-31-12-181:~/two-tier-app/flask-app-chart$
```

Then ***helm package flask-app-chart***

helm install flask-app-chart ./flask-app-chart

after that you can see the port forward option

we have to check the SG the required port is opened or not or else it will not work only change in master-node it will automatically change in worker node.



After that check

Kubectl get all

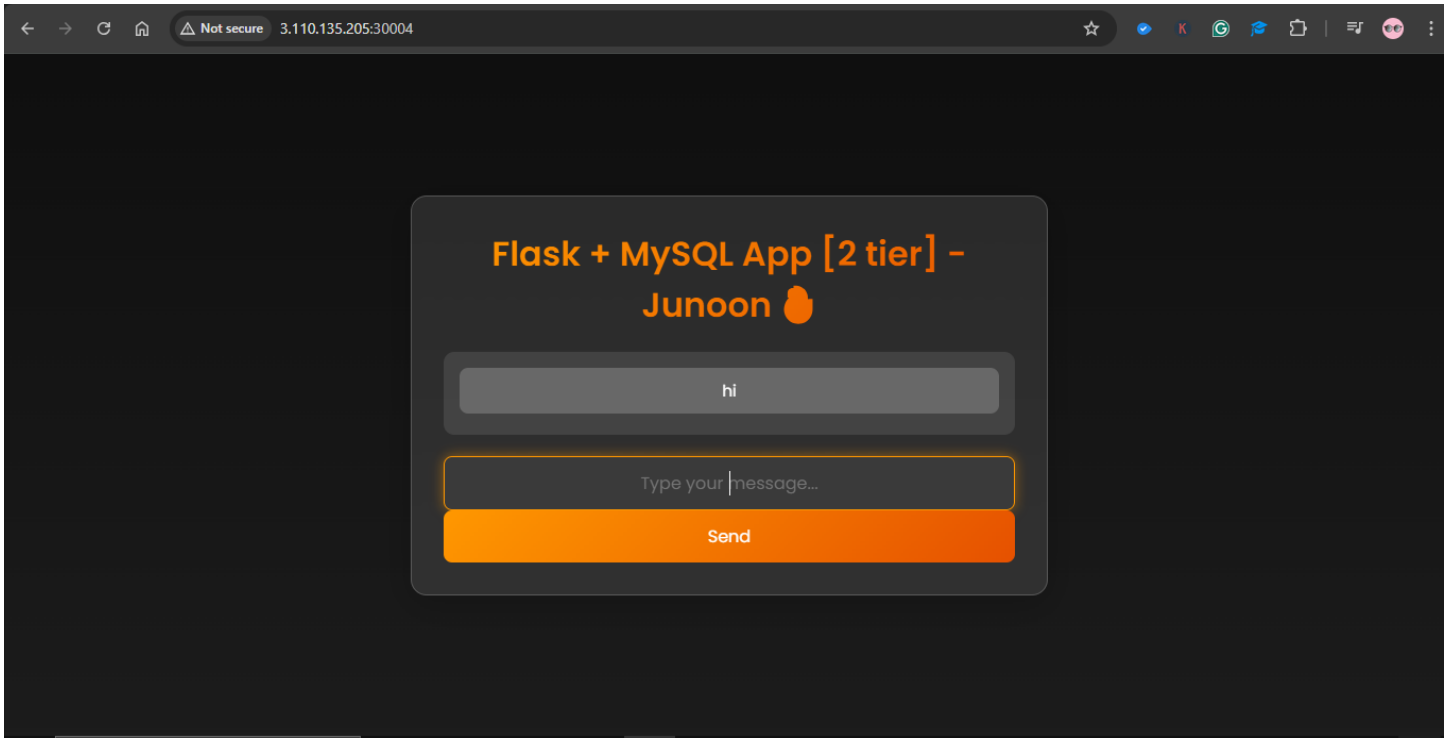
```
ubuntu@ip-172-31-12-181:~/two-tier-app$ helm package flask-app-chart/
Successfully packaged chart and saved it to: /home/ubuntu/two-tier-app/flask-app-chart-0.1.0.tgz
ubuntu@ip-172-31-12-181:~/two-tier-app$ helm install flask-app-chart ./flask-app-chart
NAME: flask-app-chart
LAST DEPLOYED: Fri Jun 13 14:31:44 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services flask-app-chart)
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
ubuntu@ip-172-31-12-181:~/two-tier-app$
```

```
ubuntu@ip-172-31-12-181:~/two-tier-app$ kubectl get all
NAME                                    READY   STATUS              RESTARTS   AGE
pod/flask-app-chart-76b9d6c76b-7ccz7    0/1     ContainerCreating   0          14s
pod/mysql-chart-84c99c66b4-vkwwv        1/1     Running             0          25m

NAME                      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/flask-app-chart   NodePort    10.110.153.68   <none>        80:30004/TCP   15s
service/kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP        69m
service/mysql-chart       ClusterIP   10.105.1.109    <none>        3306/TCP       25m

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/flask-app-chart    0/1     1            0           15s
deployment.apps/mysql-chart        1/1     1            1           25m

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/flask-app-chart-76b9d6c76b    1         1         0       14s
replicaset.apps/mysql-chart-84c99c66b4        1         1         1       25m
ubuntu@ip-172-31-12-181:~/two-tier-app$ kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/flask-app-chart-76b9d6c76b-7ccz7    1/1     Running   0          26s
pod/mysql-chart-84c99c66b4-vkwwv        1/1     Running   0          25m

NAME                      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/flask-app-chart   NodePort    10.110.153.68   <none>        80:30004/TCP   27s
service/kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP        70m
service/mysql-chart       ClusterIP   10.105.1.109    <none>        3306/TCP       25m

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/flask-app-chart    1/1     1            1           27s
deployment.apps/mysql-chart        1/1     1            1           25m

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/flask-app-chart-76b9d6c76b    1         1         1       26s
replicaset.apps/mysql-chart-84c99c66b4        1         1         1       25m
ubuntu@ip-172-31-12-181:~/two-tier-app$
```

After that I have check all container have been created …..

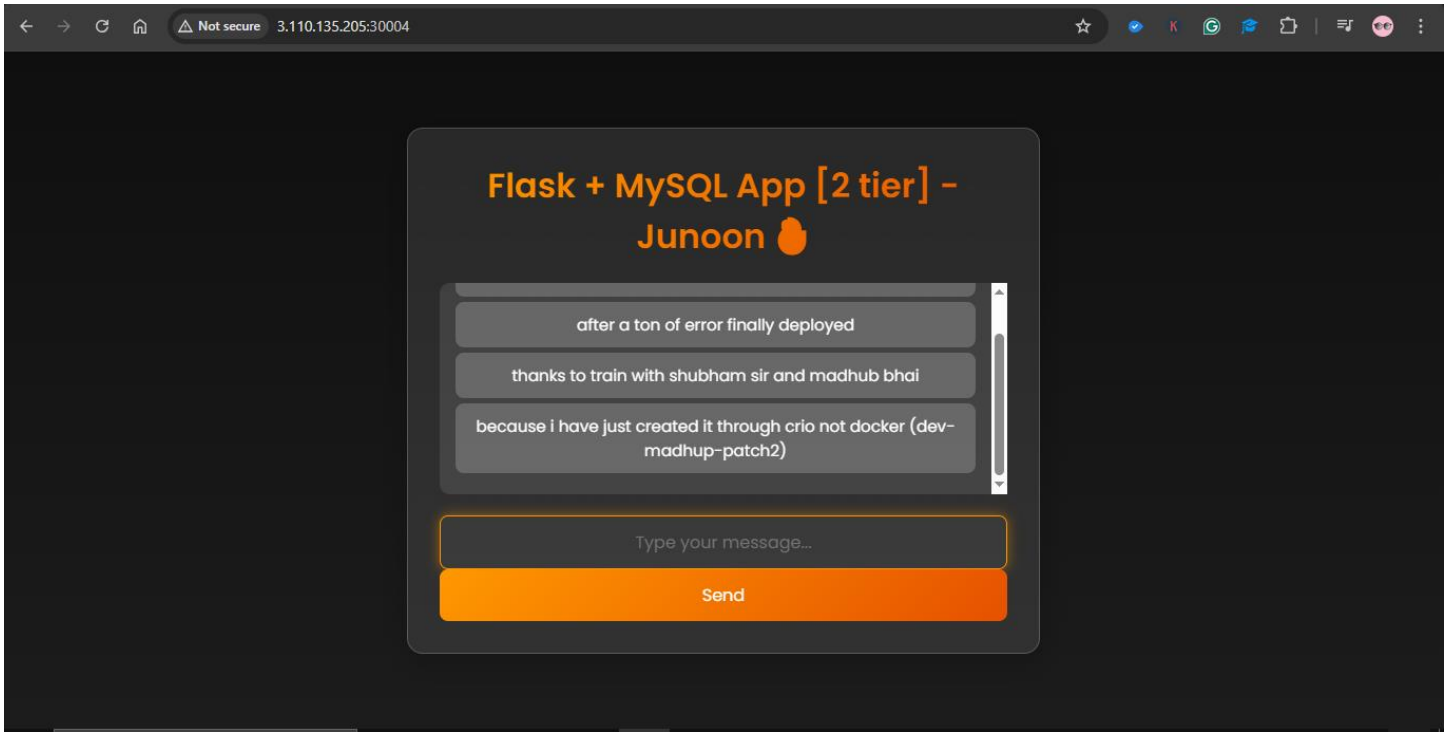
You can check worker node ip :30004

3.110.135.205:30004



Just check the messages in flask-app that I have written

A lot of error I have after that I have successfully deployed.



To check the messages

*SELECT * FROM messages;*

```
mysql> SELECT * FROM messages;
+----+-------------------------------------------------------------------------+
| id | message                                                                 |
+----+-------------------------------------------------------------------------+
|  1 | hi                                                                      |
|  2 | after a ton of error finally deployed                                   |
|  3 | thanks to train with shubham sir and madhub bhai                        |
|  4 | because i have just created it through crio not docker (dev-madhup-patch2) |
|  5 | hi                                                                      |
+----+-------------------------------------------------------------------------+
5 rows in set (0.001 sec)

mysql>
```

Then you can uninstall ***helm uninstall mysql-chart flask-app-chart***

```
ubuntu@ip-172-31-12-181:~/two-tier-app/flask-app-chart$ cd ..
ubuntu@ip-172-31-12-181:~/two-tier-app$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                   STATUS          CHART                   APP VERSI
ON
flask-app-chart default         1               2025-06-13 14:31:44.812217469 +0000 UTC deployed        flask-app-chart-0.1.0   1.16.0

mysql-chart     default         1               2025-06-13 14:06:39.715453937 +0000 UTC deployed        mysql-chart-0.1.0       1.16.0

ubuntu@ip-172-31-12-181:~/two-tier-app$ helm uninstall flask-app-chart mysql-chart
release "flask-app-chart" uninstalled
release "mysql-chart" uninstalled
ubuntu@ip-172-31-12-181:~/two-tier-app$
```