Syllabus of Docker :

- **Docker Installation on Mac, Windows & Linux**
- **Creating Demo Project on Node and Python**
- **Creating DockerFile**
- **Creating Docker Image**
- **Running Containers**
- **Pre-defined Images**
- **DockerHub**
- **Docker Volumes and Network**
- **Docker Compose**

What is a Docker?

• Docker is a containerization platform for developing, packaging,

shipping, and running applications.

• It provides the ability to run an application in an isolated

environment called a container.

• Makes deployment and development efficient.

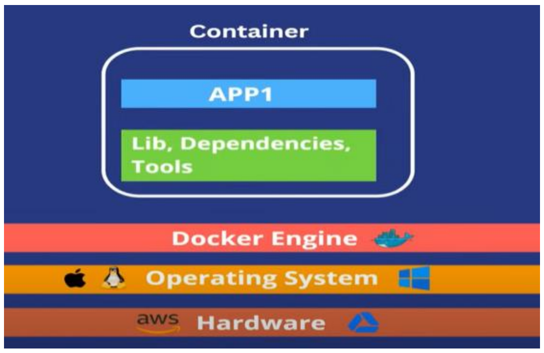
What is a Container?

• A way to package an application with all the necessary

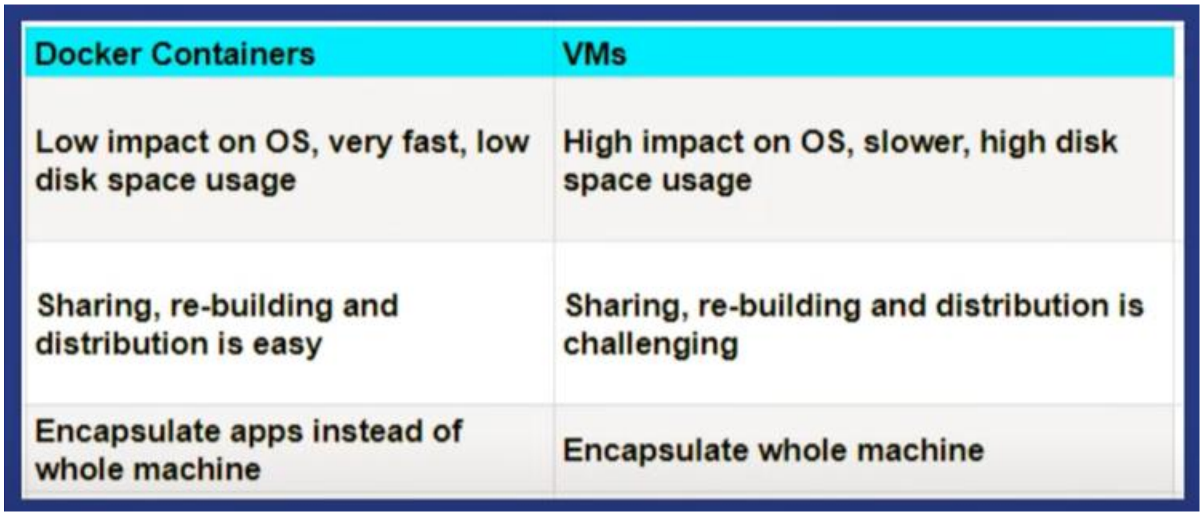
dependencies and configuration.

• It can be easily shared

• Makes deployment and development efficient.



Here multiple container are run at a same time in a isolated form.



Docker Containers	VMs
Low impact on OS, very fast, low disk space usage	High impact on OS, slower, high disk space usage
Sharing, re-building and distribution is easy	Sharing, re-building and distribution is challenging
Encapsulate apps instead of whole machine	Encapsulate whole machine

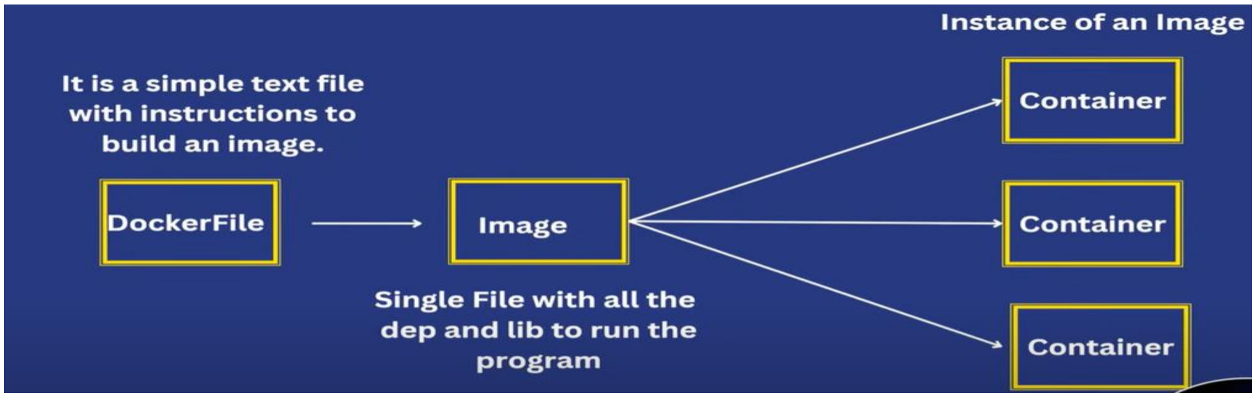
Main components of Docker

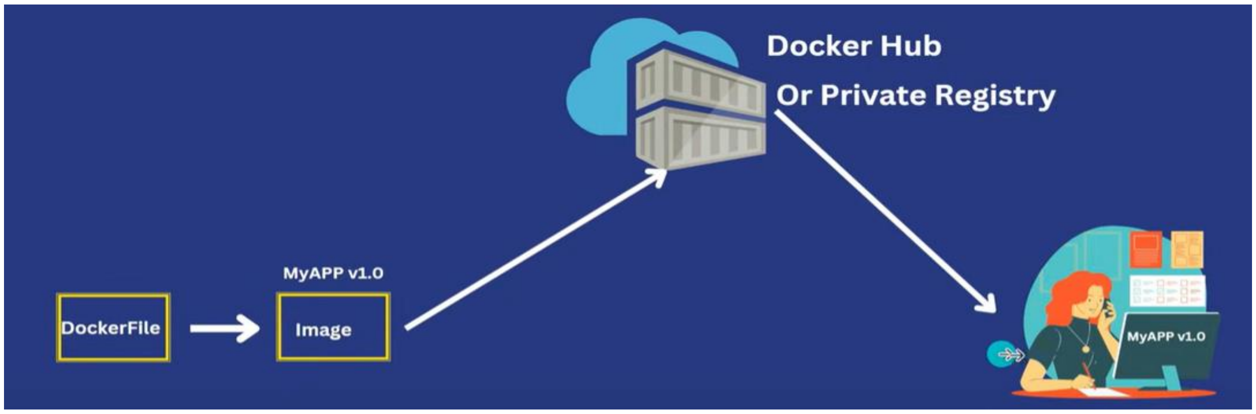
• DockerFile

• Docker Image

• Docker Container

• Docker Registry


Instance of an Image
It is a simple text file with instructions to build an image.
DockerFile
Image
Container
Container
Container
Single File with all the dep and lib to run the program

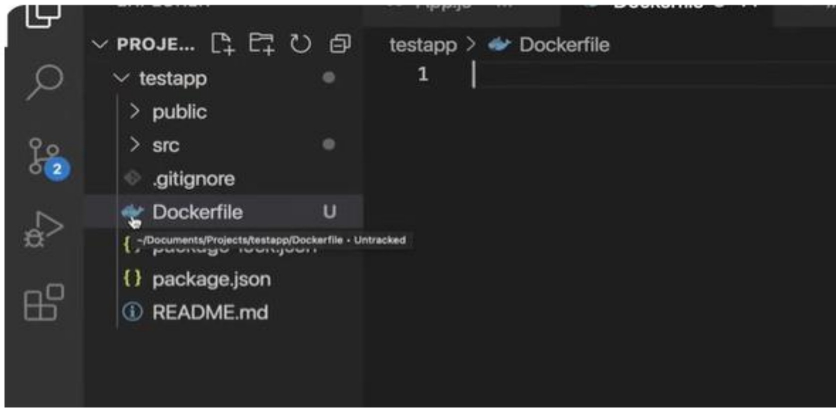


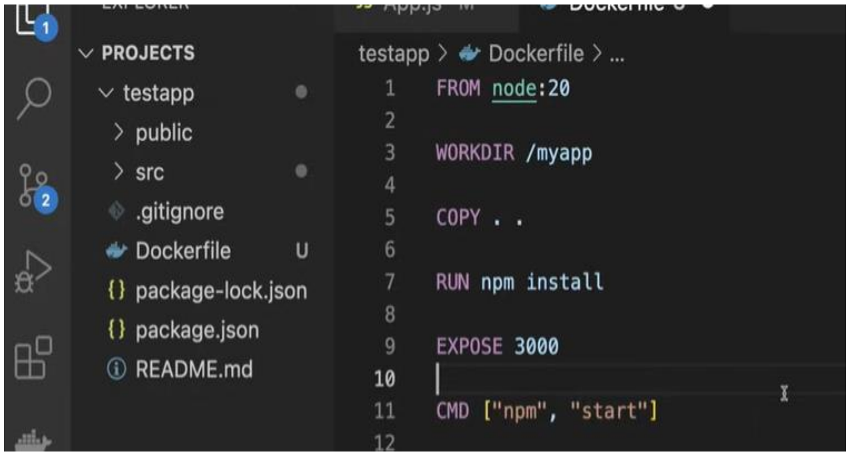
Docker Hub is a registry and inside that various versions are exist this is a repository.

Docker -v //check version

Now you create any application

1] Initially create a docker file





```
FROM node:20

WORKDIR /myapp

COPY . .

RUN npm install

EXPOSE 3000

CMD ["npm", "start"]
```

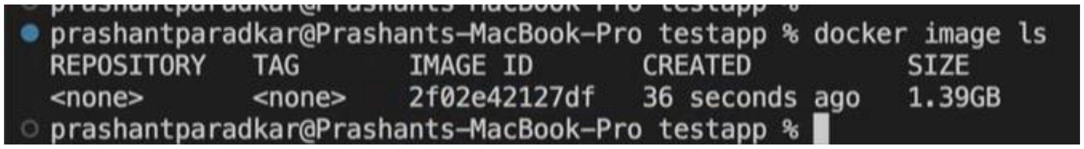
If we don’t mention node version it take latest version. Next is working directory we copy file from present dir to working dir. Exposing on 3000 port.

2] Create a docker image

In terminal :

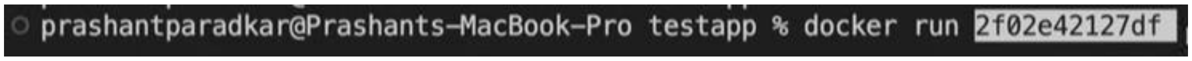
docker build .   //in present directory my docker file present. Run command with saving docker file.

docker image ls //to check list of image



```
prashantparadkar@Prashants-MacBook-Pro testapp % docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
<none>       <none>    2f02e42127df   36 seconds ago   1.39GB
prashantparadkar@Prashants-MacBook-Pro testapp %
```

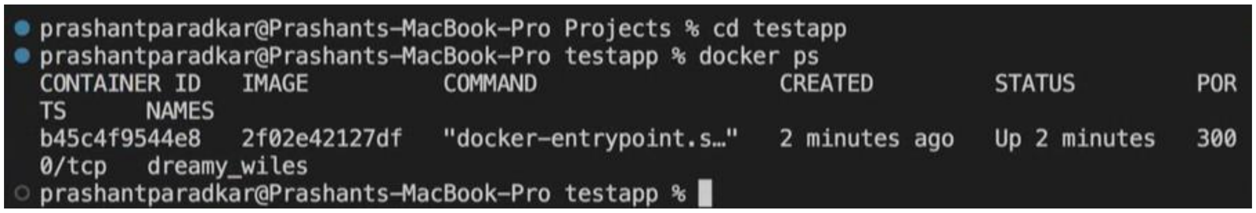
3] Run and manage docker container.



```
○ prashantparadkar@Prashants-MacBook-Pro testapp % docker run 2f02e42127df
```

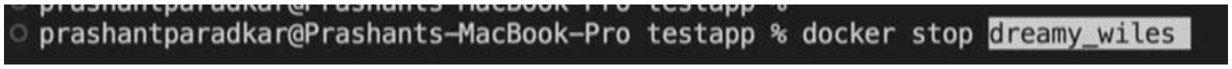
Take an image id.

Problem happen above we are able to run application inside container but not on local machine. i.e we are not able to access it outside the container.



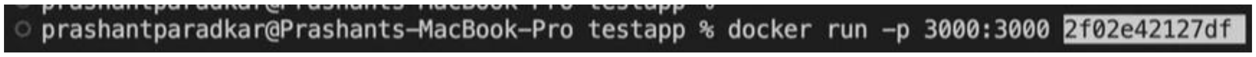
```
● prashantparadkar@Prashants-MacBook-Pro Projects % cd testapp
● prashantparadkar@Prashants-MacBook-Pro testapp % docker ps
  CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         POR
  TS       NAMES
  b45c4f9544e8   2f02e42127df   "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   300
  0/tcp   dreamy_wiles
○ prashantparadkar@Prashants-MacBook-Pro testapp % █
```

This is a running container. Docker ps is the present state of container.



```
○ prashantparadkar@Prashants-MacBook-Pro testapp % docker stop dreamy_wiles
```

Stop running container.

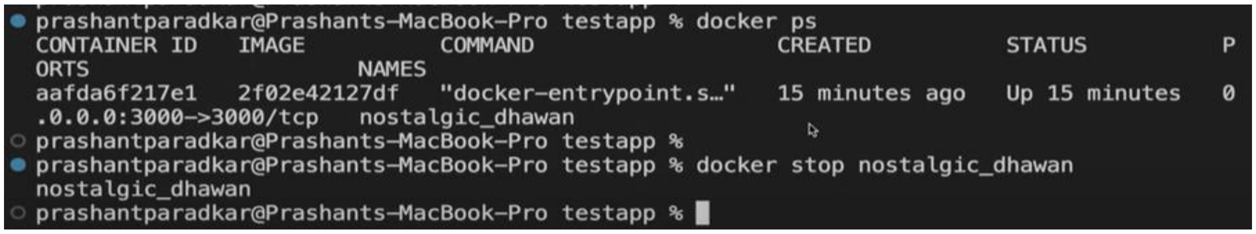


```
○ prashantparadkar@Prashants-MacBook-Pro testapp % docker run -p 3000:3000 2f02e42127df
```

Here we doing port binding i.e application run in container on 3000 port that will be bind outside container on 3000 port. Now it will be able to run in our local machine.

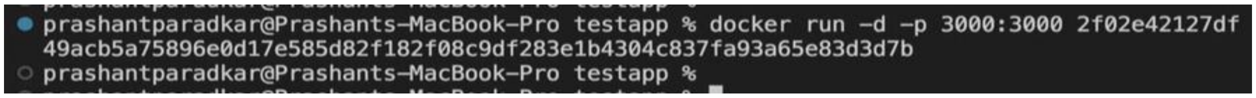
4] Running container on detached mode.

After step 3 our container run in foreground and compiler stuck but we want it run in background.



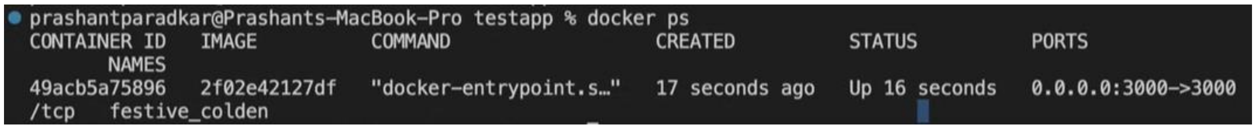
```
● prashantparadkar@Prashants-MacBook-Pro testapp % docker ps
  CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          P
  ORTS                       NAMES
  aafda6f217e1   2f02e42127df   "docker-entrypoint.s…"   15 minutes ago   Up 15 minutes   0
  .0.0.0:3000->3000/tcp   nostalgic_dhawan
○ prashantparadkar@Prashants-MacBook-Pro testapp %
● prashantparadkar@Prashants-MacBook-Pro testapp % docker stop nostalgic_dhawan
  nostalgic_dhawan
○ prashantparadkar@Prashants-MacBook-Pro testapp % █
```

Stopping our container.



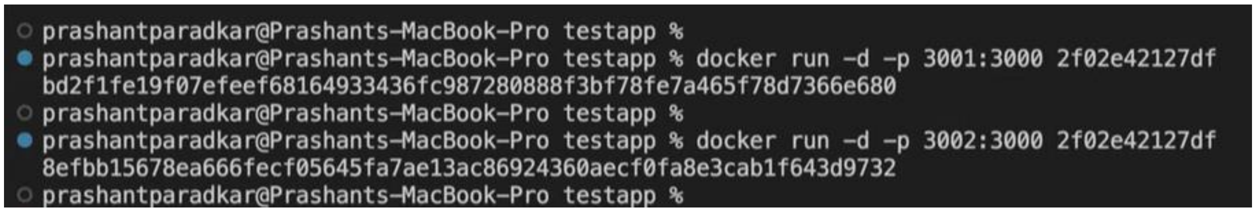
```
● prashantparadkar@Prashants-MacBook-Pro testapp % docker run -d -p 3000:3000 2f02e42127df
  49acb5a75896e0d17e585d82f182f08c9df283e1b4304c837fa93a65e83d3d7b
○ prashantparadkar@Prashants-MacBook-Pro testapp %
```

It generate and process id and terminal becomes free to use. -d detached mode. Check docker running below.



```
● prashantparadkar@Prashants-MacBook-Pro testapp % docker ps
  CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
        NAMES
  49acb5a75896   2f02e42127df   "docker-entrypoint.s…"   17 seconds ago   Up 16 seconds   0.0.0.0:3000->3000
  /tcp   festive_colden
```

5] Running Multiple container.



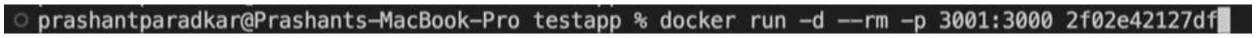
```
prashantparadkar@Prashants-MacBook-Pro testapp %
prashantparadkar@Prashants-MacBook-Pro testapp % docker run -d -p 3001:3000 2f02e42127df
bd2f1fe19f07efeef68164933436fc987280888f3bf78fe7a465f78d7366e680
prashantparadkar@Prashants-MacBook-Pro testapp %
prashantparadkar@Prashants-MacBook-Pro testapp % docker run -d -p 3002:3000 2f02e42127df
8efbb15678ea666fecf05645fa7ae13ac86924360aecf0fa8e3cab1f643d9732
prashantparadkar@Prashants-MacBook-Pro testapp %
```

Each container run on different port. All listen on 3000 port.

docker ps -a //show all running conainer with hidden also i.e run in background.

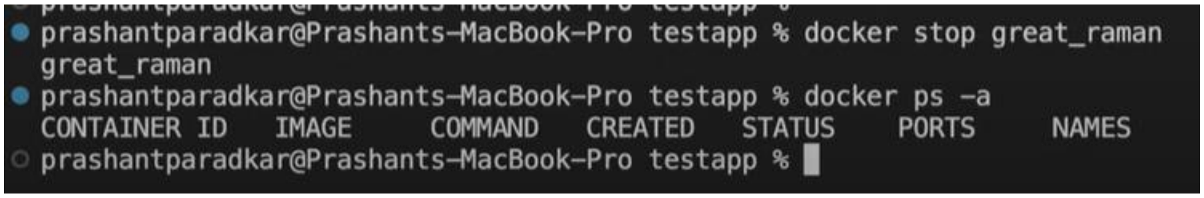
docker rm container_name //to remove a container we can also delete multiple container at same time.

//above process we can directly performed in docker desktop also.



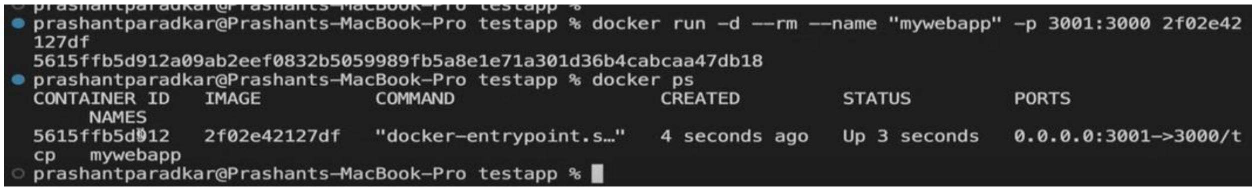
```
prashantparadkar@Prashants-MacBook-Pro testapp % docker run -d --rm -p 3001:3000 2f02e42127df
```

when we stop container it will automatically removed.



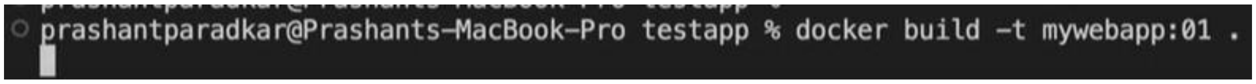
```
prashantparadkar@Prashants-MacBook-Pro testapp % docker stop great_raman
great_raman
prashantparadkar@Prashants-MacBook-Pro testapp % docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
prashantparadkar@Prashants-MacBook-Pro testapp %
```

Changing name of container.



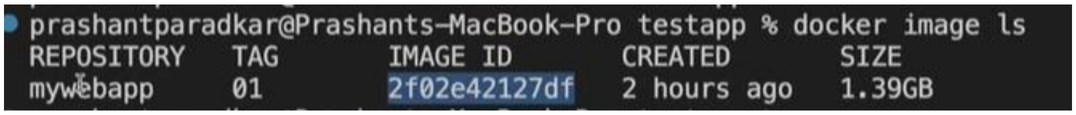
```
prashantparadkar@Prashants-MacBook-Pro testapp % docker run -d --rm --name "mywebapp" -p 3001:3000 2f02e42
127df
5615ffb5d912a09ab2eef0832b5059989fb5a8e1e71a301d36b4cabcaa47db18
prashantparadkar@Prashants-MacBook-Pro testapp % docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS
     NAMES
5615ffb5d912   2f02e42127df   "docker-entrypoint.s…"   4 seconds ago   Up 3 seconds   0.0.0.0:3001->3000/t
cp   mywebapp
prashantparadkar@Prashants-MacBook-Pro testapp %
```

Changing name of an image



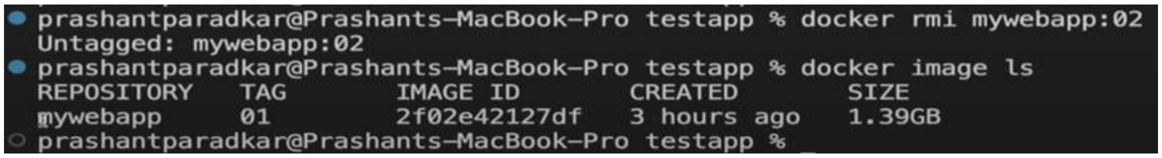
```
prashantparadkar@Prashants-MacBook-Pro testapp % docker build -t mywebapp:01 .
```

-t for tag format: name:version



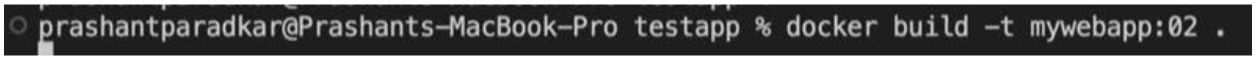
```
prashantparadkar@Prashants-MacBook-Pro testapp % docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
mywebapp     01        2f02e42127df   2 hours ago   1.39GB
```

remove an image. **rmi**



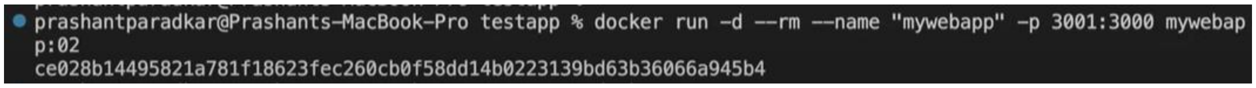
```
prashantparadkar@Prashants-MacBook-Pro testapp % docker rmi mywebapp:02
Untagged: mywebapp:02
prashantparadkar@Prashants-MacBook-Pro testapp % docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
mywebapp     01        2f02e42127df   3 hours ago   1.39GB
prashantparadkar@Prashants-MacBook-Pro testapp %
```

if I want to make changes in project.



```
prashantparadkar@Prashants-MacBook-Pro testapp % docker build -t mywebapp:02 .
```

create new version with upgradation.



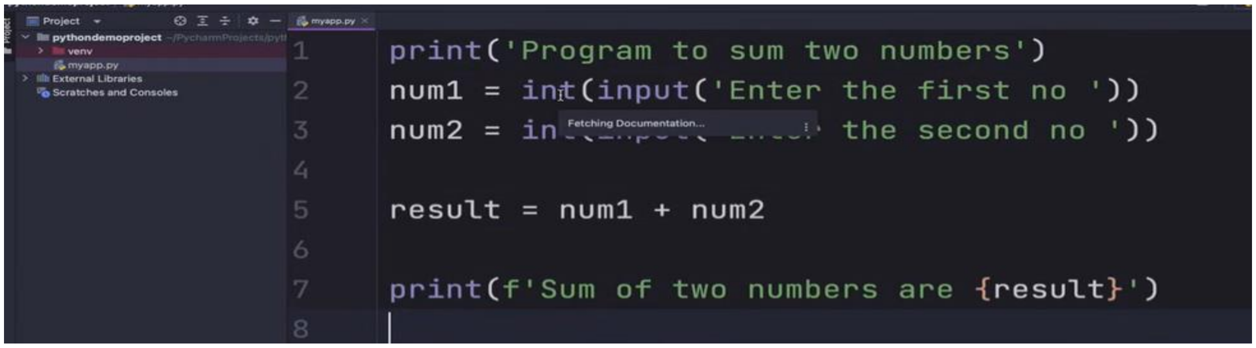
```
prashantparadkar@Prashants-MacBook-Pro testapp % docker run -d --rm --name "mywebapp" -p 3001:3000 mywebap
p:02
ce028b14495821a781f18623fec260cb0f58dd14b0223139bd63b36066a945b4
```

To pull image from dockerhub.

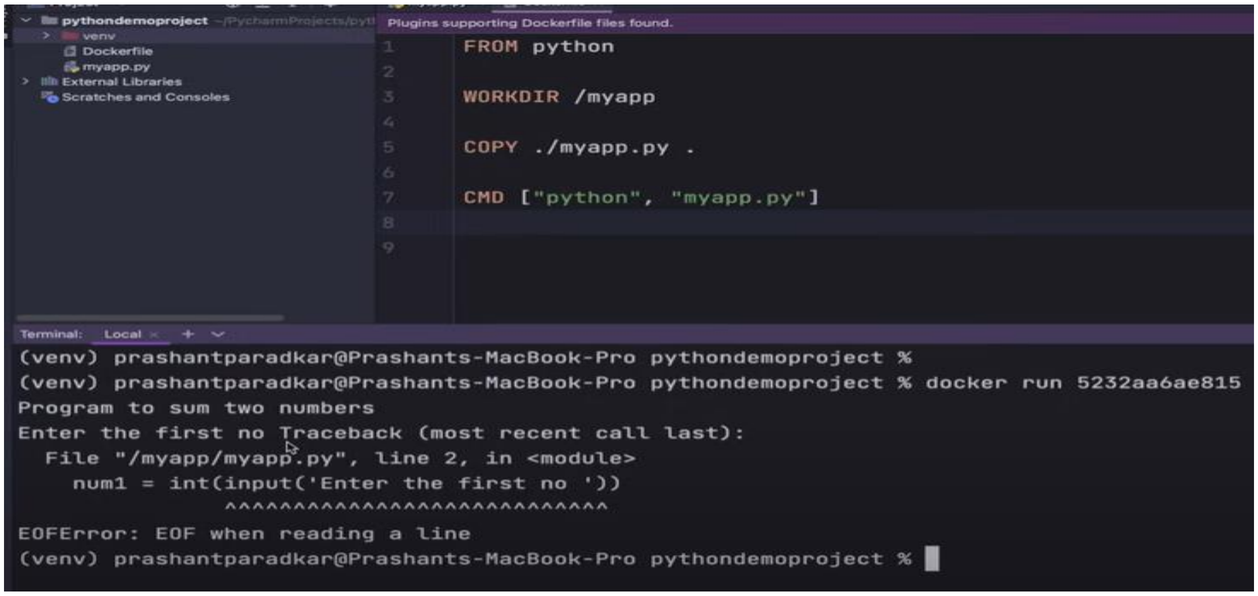
docker pull python //pull latest version if we don't specify.

ngnix run on default port 8080.

How to use docker in interactive mode i.e used pass input externally.



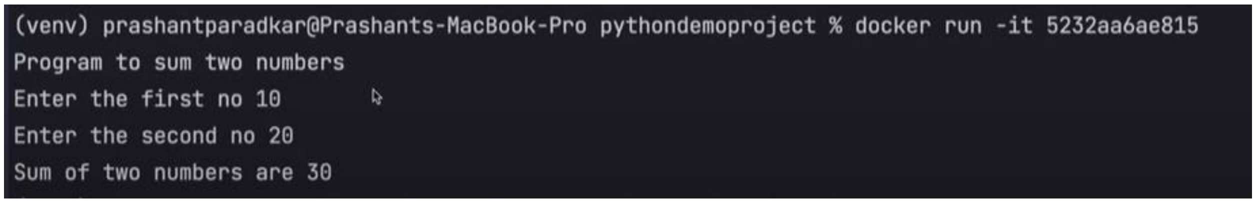
```
print('Program to sum two numbers')
num1 = int(input('Enter the first no '))
num2 = int(input('Enter the second no '))

result = num1 + num2

print(f'Sum of two numbers are {result}')
```



```
FROM python

WORKDIR /myapp

COPY ./myapp.py .

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

```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run 5232aa6ae815
Program to sum two numbers
Enter the first no Traceback (most recent call last):
  File "/myapp/myapp.py", line 2, in <module>
    num1 = int(input('Enter the first no '))
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
EOFError: EOF when reading a line
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
```

create docker file but it fail to take an input.



```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run -it 5232aa6ae815
Program to sum two numbers
Enter the first no 10
Enter the second no 20
Sum of two numbers are 30
```

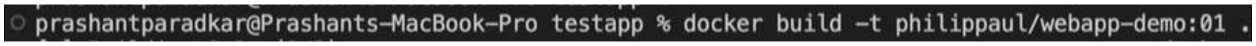
-it for running in interactive terminal.

To push docker image on dockerhub.

initially create repository in dockerhub.

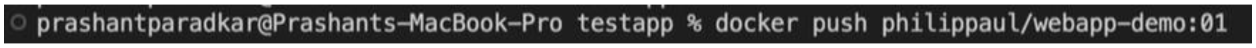
on terminal:

docker login



```
prashantparadkar@Prashants-MacBook-Pro testapp % docker build -t philippaul/webapp-demo:01 .
```

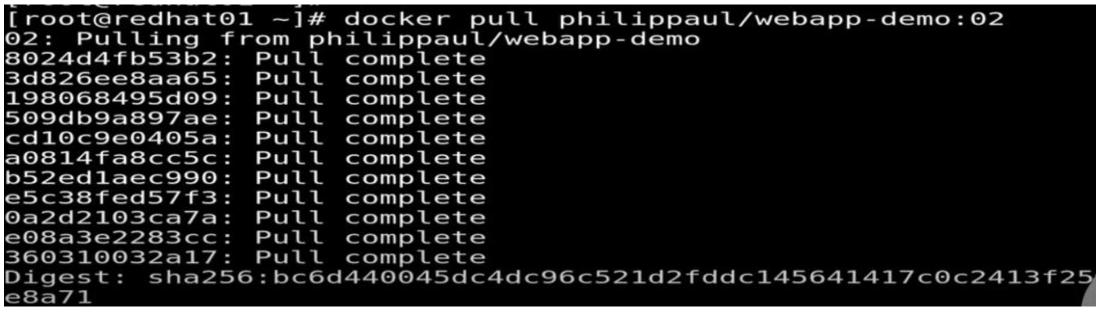
make image of same name of which repository name.



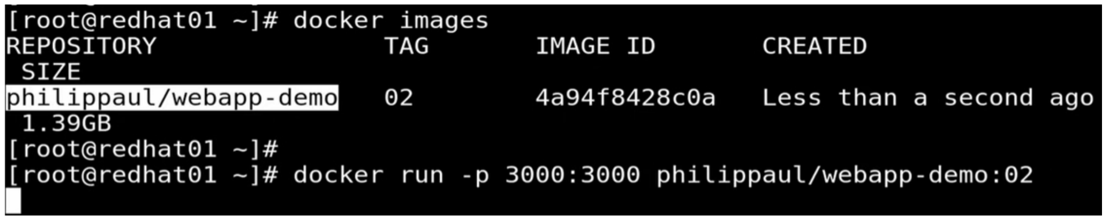
```
prashantparadkar@Prashants-MacBook-Pro testapp % docker push philippaul/webapp-demo:01
```

and then push.

How to use image on different os to check it is running or not.

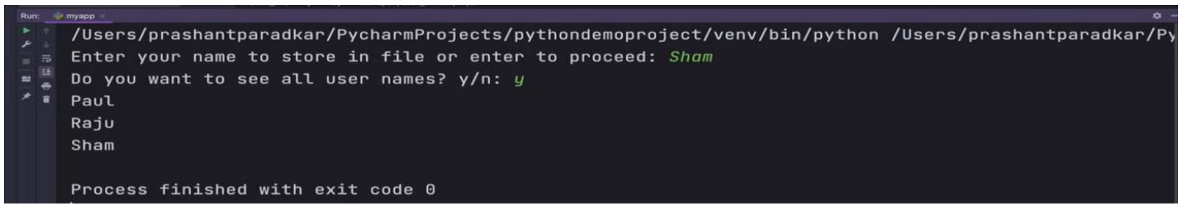


```
[root@redhat01 ~]# docker pull philippaul/webapp-demo:02
02: Pulling from philippaul/webapp-demo
8024d4fb53b2: Pull complete
3d826ee8aa65: Pull complete
198068495d09: Pull complete
509db9a897ae: Pull complete
cd10c9e0405a: Pull complete
a0814fa8cc5c: Pull complete
b52ed1aec990: Pull complete
e5c38fed57f3: Pull complete
0a2d2103ca7a: Pull complete
e08a3e2283cc: Pull complete
360310032a17: Pull complete
Digest: sha256:bc6d440045dc4dc96c521d2fddc145641417c0c2413f25
e8a71
```

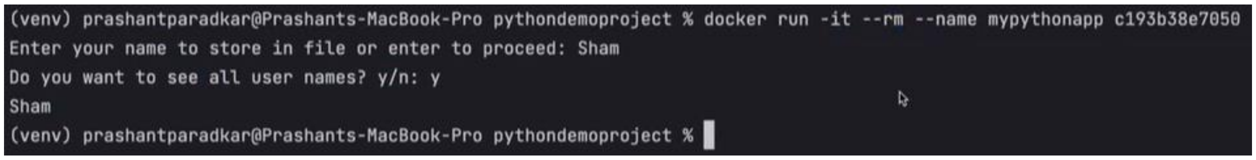


```
[root@redhat01 ~]# docker images
REPOSITORY              TAG       IMAGE ID       CREATED
 SIZE
philippaul/webapp-demo  02        4a94f8428c0a   Less than a second ago
 1.39GB
[root@redhat01 ~]#
[root@redhat01 ~]# docker run -p 3000:3000 philippaul/webapp-demo:02
```

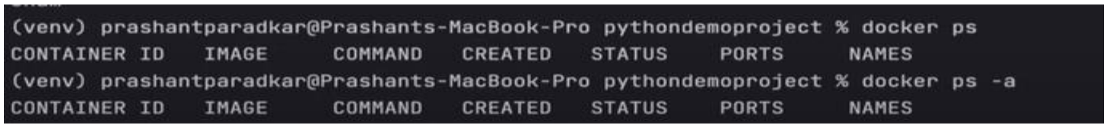
Docker Volumes : python program that store data permently



```
/Users/prashantparadkar/PycharmProjects/pythondemoproject/venv/bin/python /Users/prashantparadkar/Py
Enter your name to store in file or enter to proceed: Sham
Do you want to see all user names? y/n: y
Paul
Raju
Sham

Process finished with exit code 0
```

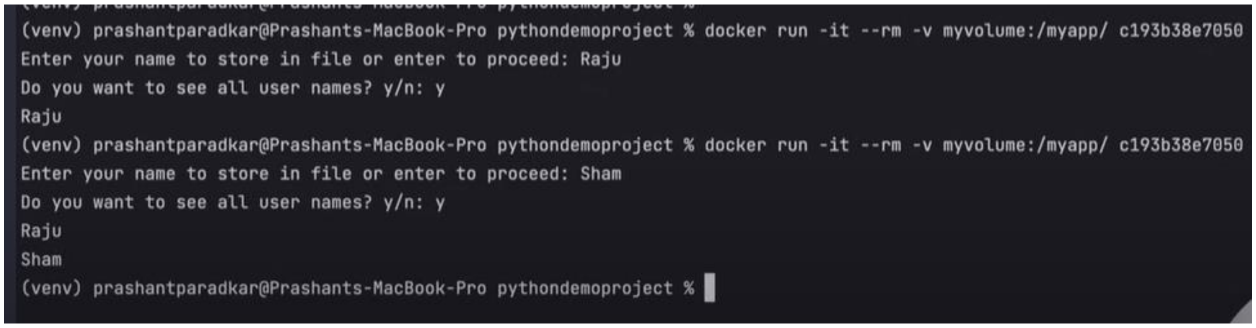


```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run -it --rm --name mypythonapp c193b38e7050
Enter your name to store in file or enter to proceed: Sham
Do you want to see all user names? y/n: y
Sham
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
```



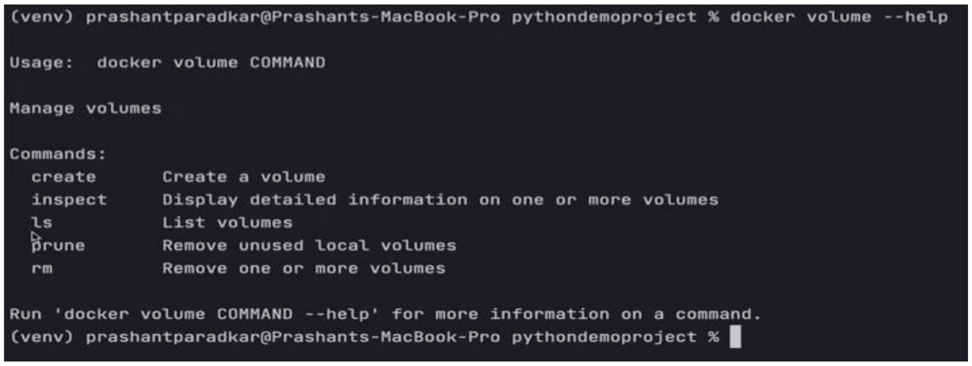
```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

here container stopped file will be removed.

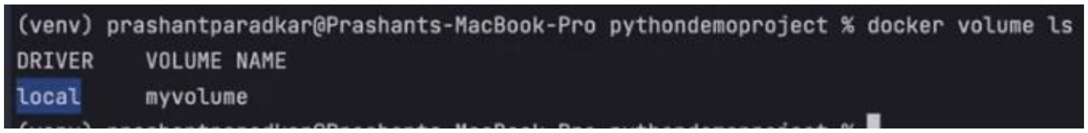


```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run -it --rm -v myvolume:/myapp/ c193b38e7050
Enter your name to store in file or enter to proceed: Raju
Do you want to see all user names? y/n: y
Raju
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run -it --rm -v myvolume:/myapp/ c193b38e7050
Enter your name to store in file or enter to proceed: Sham
Do you want to see all user names? y/n: y
Raju
Sham
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
```

-v is volume and myvolume is volume name stored this volume is same directory where your python file will be running i.e /myapp/



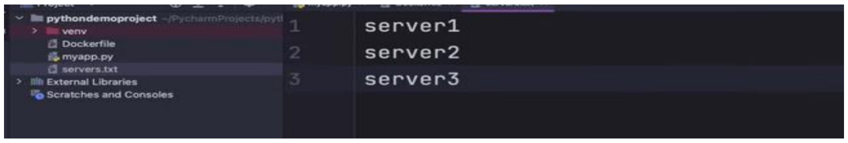
```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker volume --help

Usage:  docker volume COMMAND

Manage volumes

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove unused local volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
```



```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker volume ls
DRIVER    VOLUME NAME
local     myvolume
```

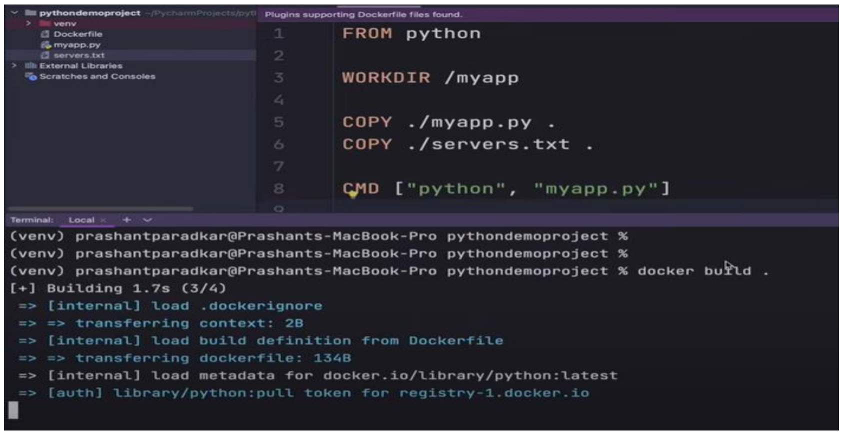
docker volume inspect myvolume //give all information.

Bind Mouts :



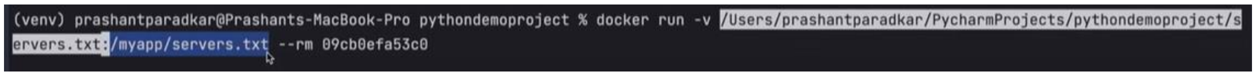
```
server1
server2
server3
```

when we add data in servers.txt file it will be displayed when program run.



```
FROM python

WORKDIR /myapp

COPY ./myapp.py .
COPY ./servers.txt .

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

```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker build .
[+] Building 1.7s (3/4)
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 134B
 => [internal] load metadata for docker.io/library/python:latest
 => [auth] library/python:pull token for registry-1.docker.io
```

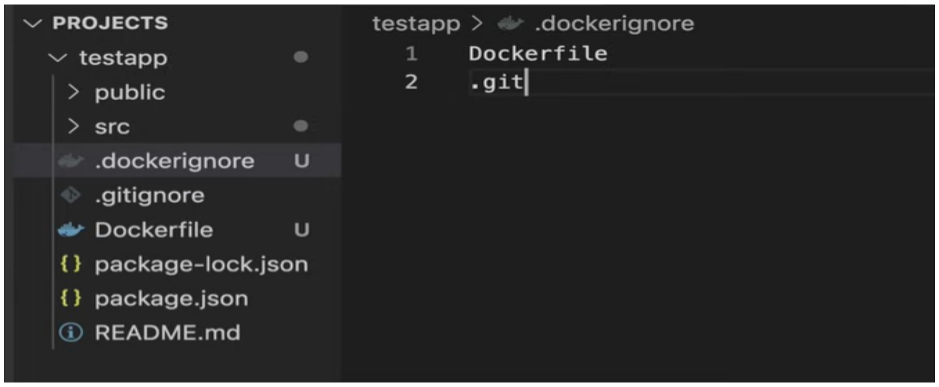


```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run -v /Users/prashantparadkar/PycharmProjects/pythondemoproject/s
ervers.txt:/myapp/servers.txt --rm 09cb0efa53c0
```

mount servers.txt in local machine with remote servers.txt in /myapp/servers.txt.

when we add data in file it will be visible. here we do not need volume.

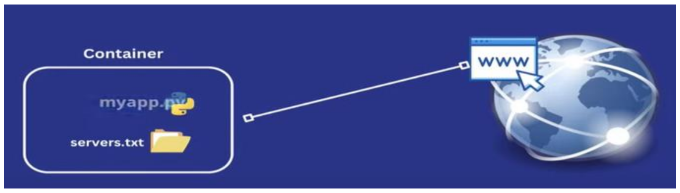
.dockerignore in docker :



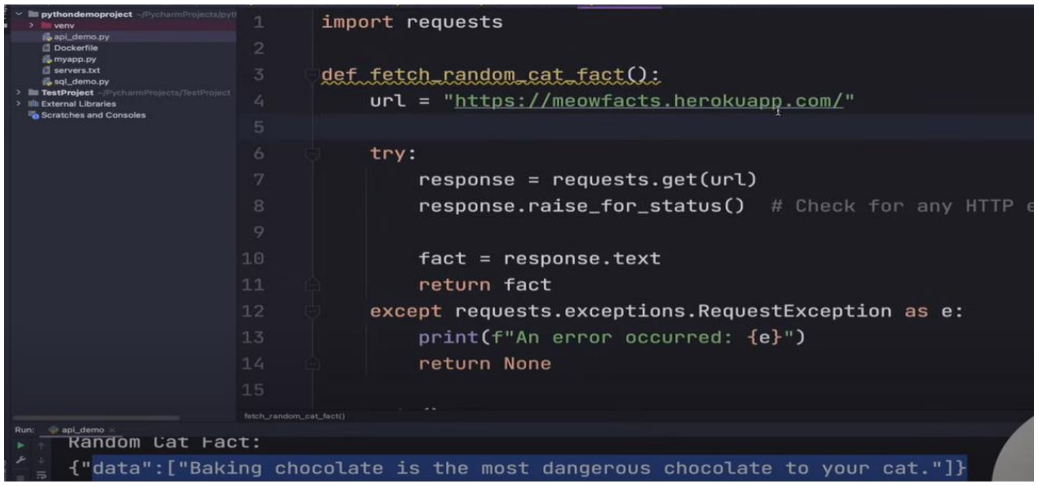
```
Dockerfile
.git
```

do not include that file that do not required.

Communication From/ To Containers : 3 cases



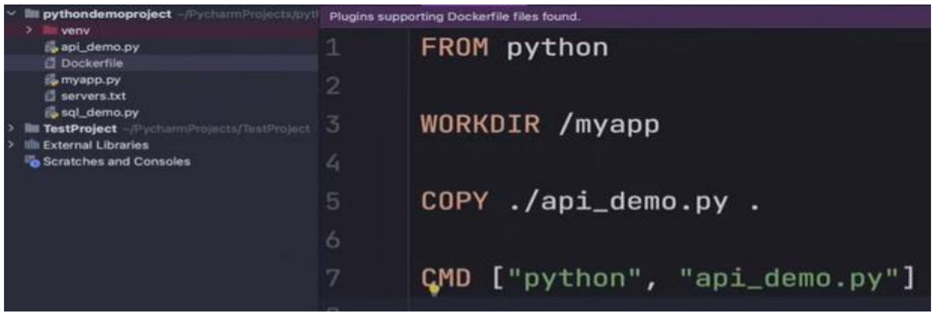
Working with API :



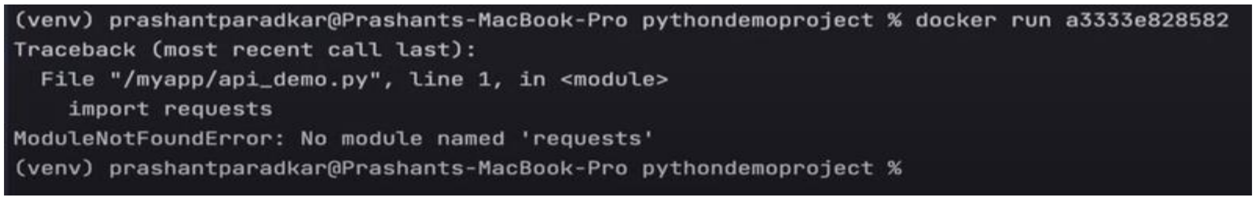
```python
import requests

def fetch_random_cat_fact():
    url = "https://meowfacts.herokuapp.com/"

    try:
        response = requests.get(url)
        response.raise_for_status()  # Check for any HTTP e

        fact = response.text
        return fact
    except requests.exceptions.RequestException as e:
        print(f"An error occurred: {e}")
        return None
```

```
Random Cat Fact:
{"data":["Baking chocolate is the most dangerous chocolate to your cat."]}
```

Here we are using API that generate different text about cat when we execute. Create docker file

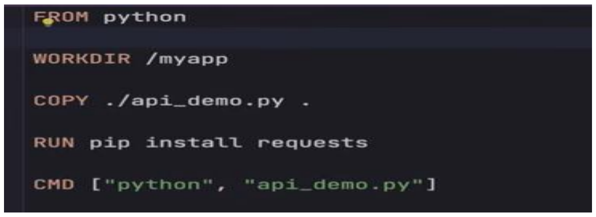


```dockerfile
FROM python

WORKDIR /myapp

COPY ./api_demo.py .

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



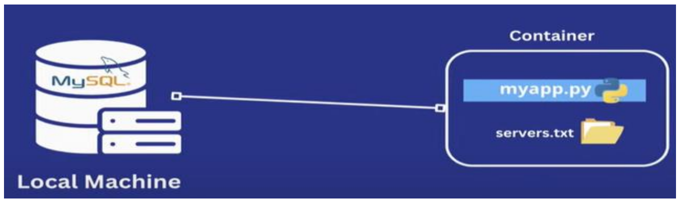
```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run a3333e828582
Traceback (most recent call last):
  File "/myapp/api_demo.py", line 1, in <module>
    import requests
ModuleNotFoundError: No module named 'requests'
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
```

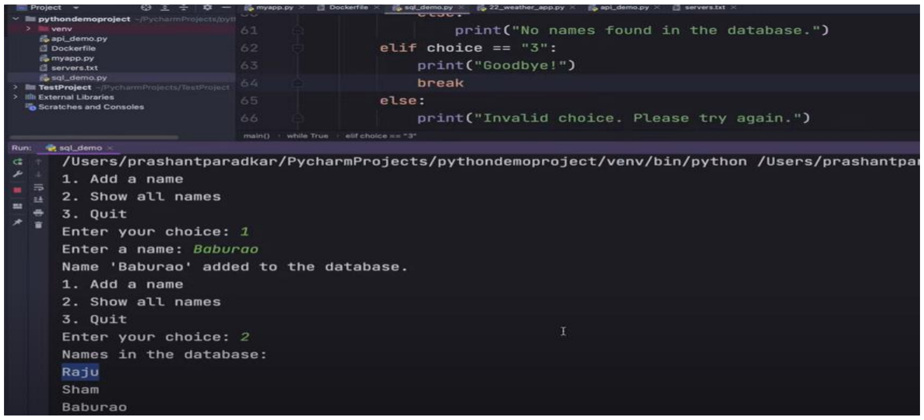
got an error for API



```dockerfile
FROM python

WORKDIR /myapp

COPY ./api_demo.py .

RUN pip install requests

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

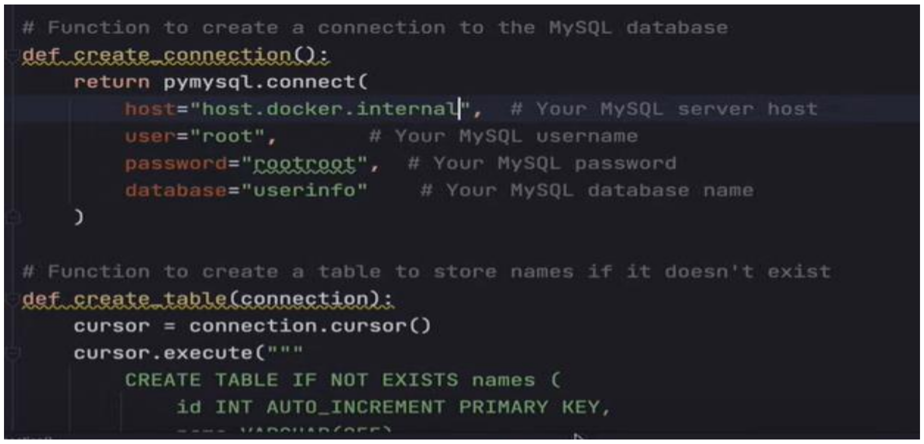
add in docker file.





```
            print("No names found in the database.")
        elif choice == "3":
            print("Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")
```

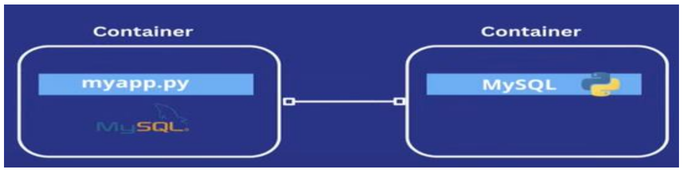
```
/Users/prashantparadkar/PycharmProjects/pythondemoproject/venv/bin/python /Users/prashantpa
1. Add a name
2. Show all names
3. Quit
Enter your choice: 1
Enter a name: Baburao
Name 'Baburao' added to the database.
1. Add a name
2. Show all names
3. Quit
Enter your choice: 2
Names in the database:
Raju
Sham
Baburao
```



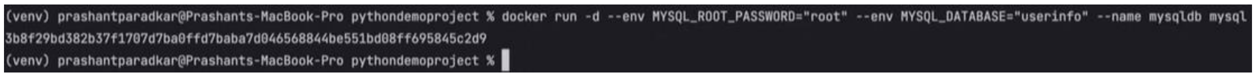
```
# Function to create a connection to the MySQL database
def create_connection():
    return pymysql.connect(
        host="host.docker.internal",  # Your MySQL server host
        user="root",         # Your MySQL username
        password="rootroot",  # Your MySQL password
        database="userinfo"     # Your MySQL database name
    )

# Function to create a table to store names if it doesn't exist
def create_table(connection):
    cursor = connection.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS names (
            id INT AUTO_INCREMENT PRIMARY KEY,
```

make change in host.

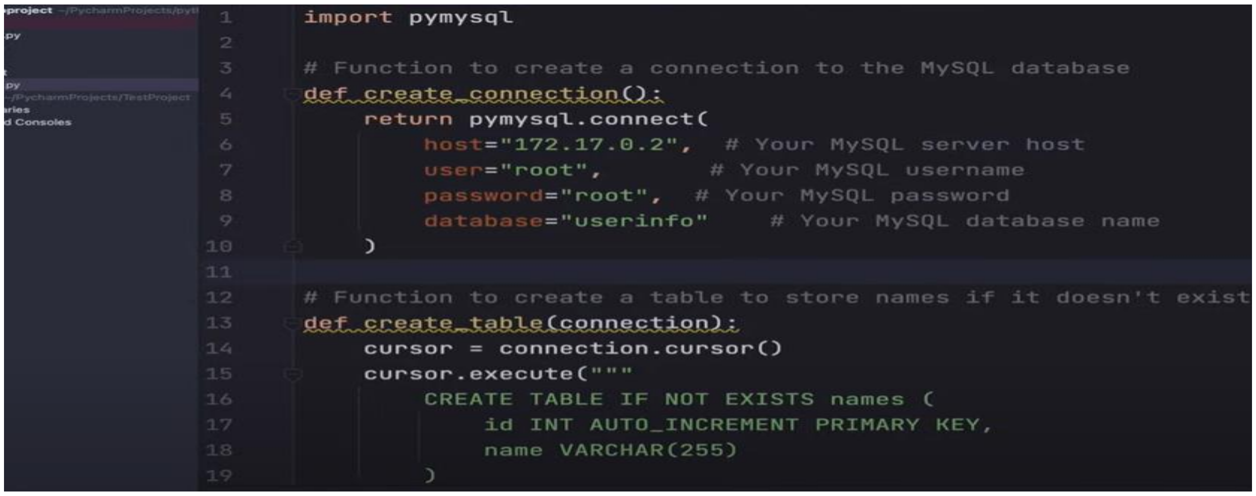
Communication between container.



docker pull mysql //make mysql container



```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run -d --env MYSQL_ROOT_PASSWORD="root" --env MYSQL_DATABASE="userinfo" --name mysqldb mysql
3b8f29bd382b37f1707d7ba0ffd7baba7d046568844be551bd08ff695845c2d9
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
```



```python
import pymysql

# Function to create a connection to the MySQL database
def create_connection():
    return pymysql.connect(
        host="172.17.0.2",  # Your MySQL server host
        user="root",        # Your MySQL username
        password="root",  # Your MySQL password
        database="userinfo"     # Your MySQL database name
    )

# Function to create a table to store names if it doesn't exist
def create_table(connection):
    cursor = connection.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS names (
            id INT AUTO_INCREMENT PRIMARY KEY,
            name VARCHAR(255)
        )
```

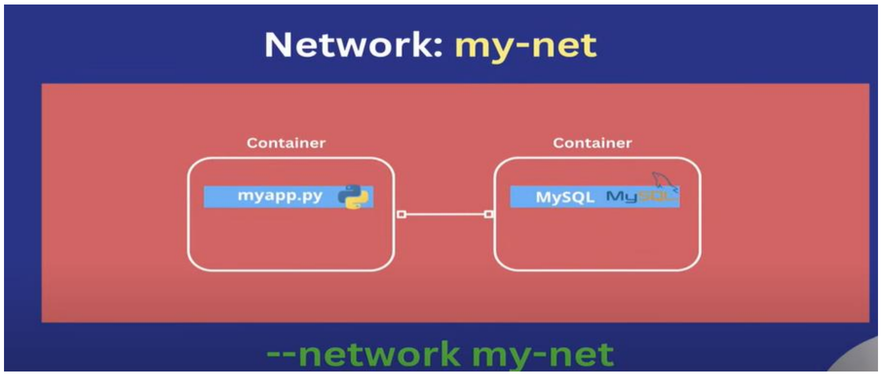
change host ip by using docker inspect mysql.

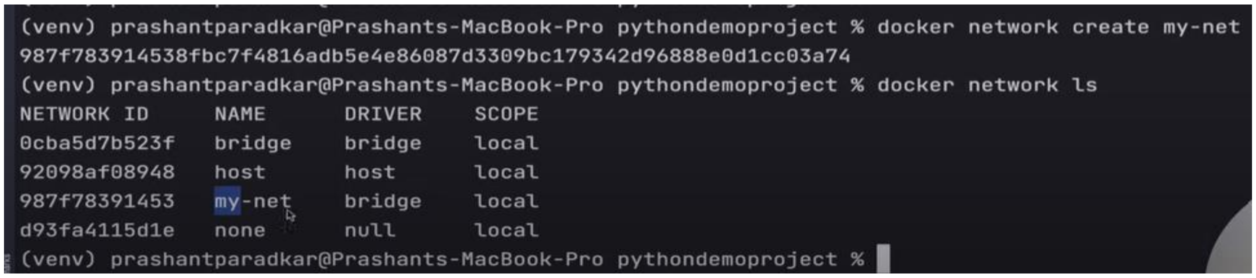
to start container : docker start mysql

Docker Network :

above we create the connection between two container but for running python container before that we need to always run mysql container mandatory.

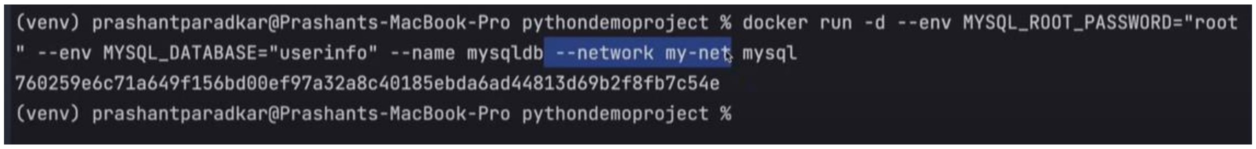
sol: Docker Network both container run in same network.

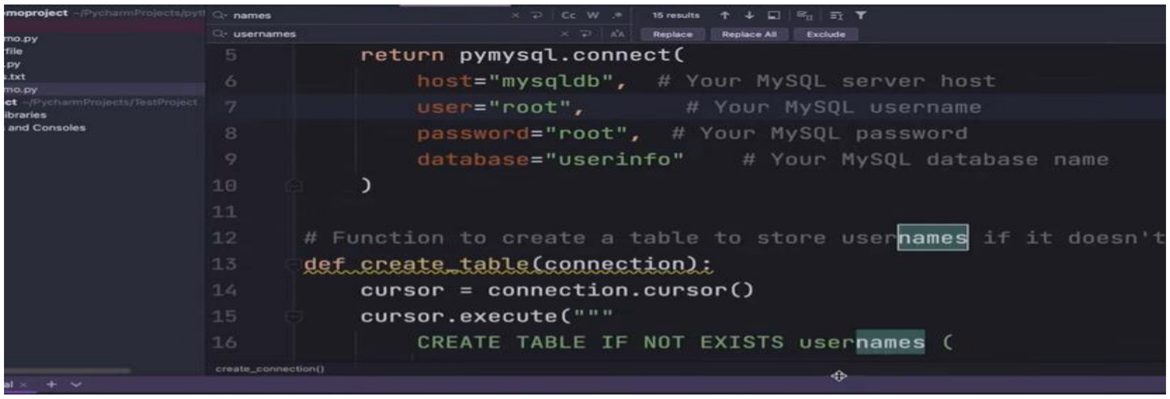




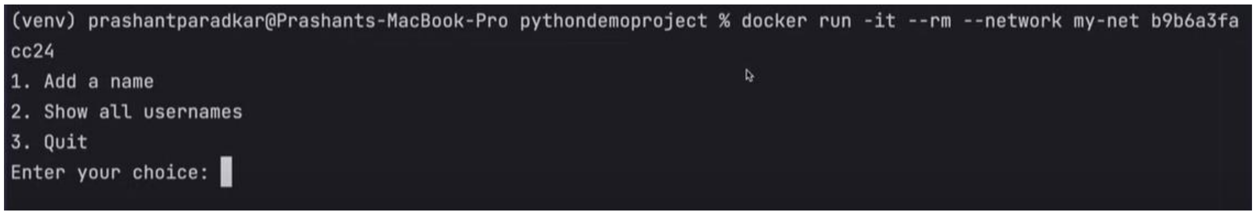
```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker network create my-net
987f783914538fbc7f4816adb5e4e86087d3309bc179342d96888e0d1cc03a74
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
0cba5d7b523f   bridge    bridge    local
92098af08948   host      host      local
987f78391453   my-net    bridge    local
d93fa4115d1e   none      null      local
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
```

Running Mysql container assign to network.


```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run -d --env MYSQL_ROOT_PASSWORD="root
" --env MYSQL_DATABASE="userinfo" --name mysqldb --network my-net mysql
760259e6c71a649f156bd00ef97a32a8c40185ebda6ad44813d69b2f8fb7c54e
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
```


```
        return pymysql.connect(
            host="mysqldb",  # Your MySQL server host
            user="root",        # Your MySQL username
            password="root",  # Your MySQL password
            database="userinfo"    # Your MySQL database name
        )

    # Function to create a table to store usernames if it doesn't
    def create_table(connection):
        cursor = connection.cursor()
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS usernames (
```

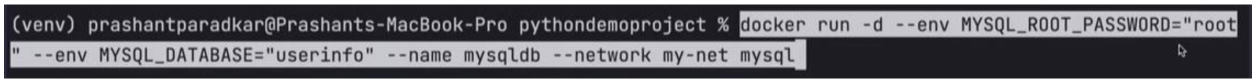
here in python code host name will be replaced with container name directly bcoz its part of network.


```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run -it --rm --network my-net b9b6a3fa
cc24
1. Add a name
2. Show all usernames
3. Quit
Enter your choice:
```

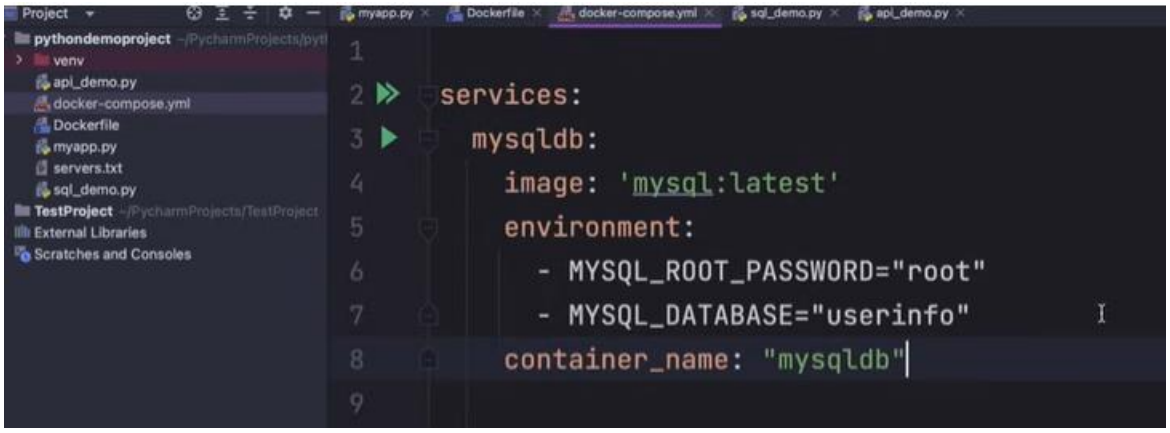
Docker Compose :

Configuration file to manage multiple containers running on same machine..

problem in normal method:


```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run -d --env MYSQL_ROOT_PASSWORD="root
" --env MYSQL_DATABASE="userinfo" --name mysqldb --network my-net mysql
```

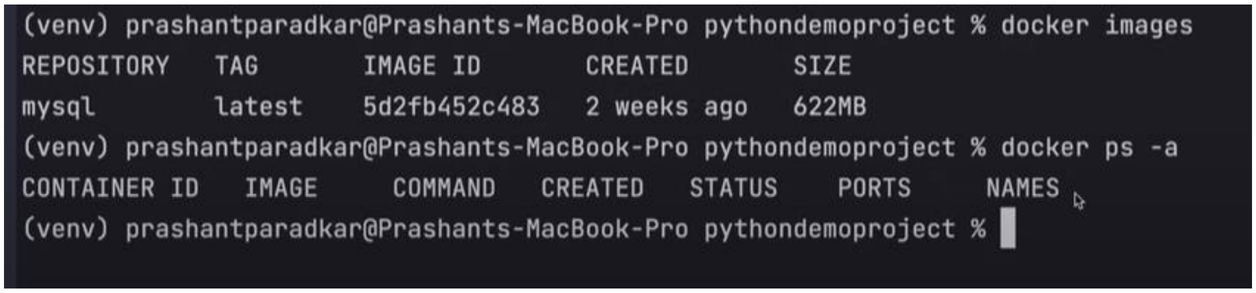
here query will be big lot of configuration will be added.


```

services:
  mysqldb:
    image: 'mysql:latest'
    environment:
      - MYSQL_ROOT_PASSWORD="root"
      - MYSQL_DATABASE="userinfo"
    container_name: "mysqldb"

```

Here we creating docker-compose.yml for one container i.e mysql

to up container : docker-compose up

to down container : docker-compose down



```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
mysql        latest    5d2fb452c483   2 weeks ago   622MB
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
```

when we down container it will automatically remove container no need to add –rm

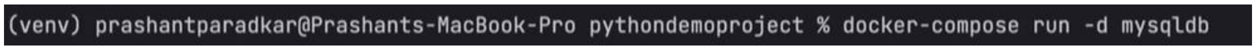
docker-compose up -d //run in detached mode

Docker Compose with multiple container :

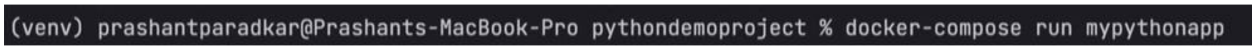
docker compose will not replace docker file still we need docker file .

```
services:
  mysqldb:
    image: 'mysql:latest'
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=userinfo
    container_name: "mysqldb"
    networks:
      - my-network
    healthcheck:
      test: ['CMD','mysqladmin','ping','-h','localhost']
      timeout: 20s
      retries: 10

  mypythonapp:
    build: ./
    container_name: mypyapp
    networks:
      - my-network
    volumes:
      - ./servers.txt:/myapp/servers.txt
    depends_on:
      mysqldb:
        condition: service_healthy
    stdin_open: true
    tty: true

networks:
  my-network:
```

adding python container in compose file given relative path ./ and it will run after entire mysql container running. Run one by one.

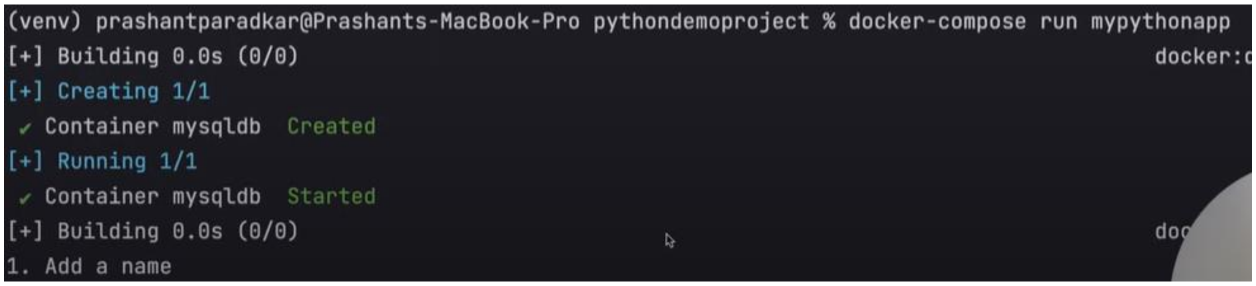


```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker-compose run -d mysqldb
```



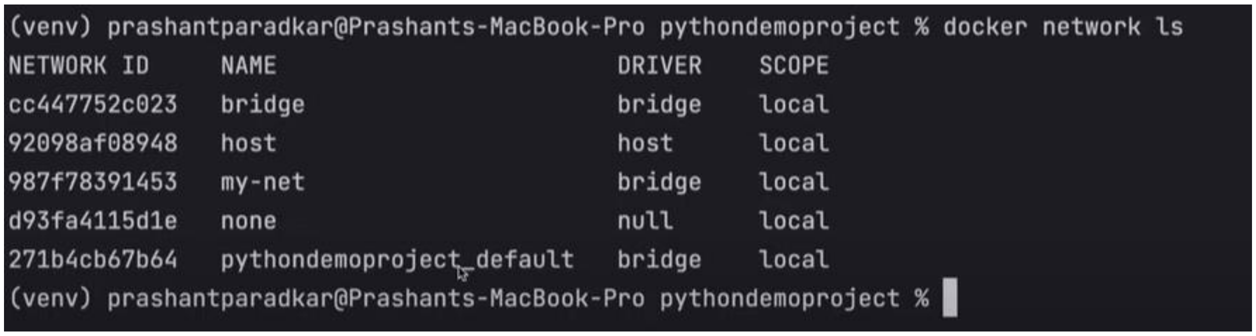
```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker-compose run mypythonapp
```

Docker compose with network :



```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker-compose run mypythonapp
[+] Building 0.0s (0/0)                                                               docker:
[+] Creating 1/1
 ✓ Container mysqldb  Created
[+] Running 1/1
 ✓ Container mysqldb  Started
[+] Building 0.0s (0/0)                                                               do
1. Add a name
```

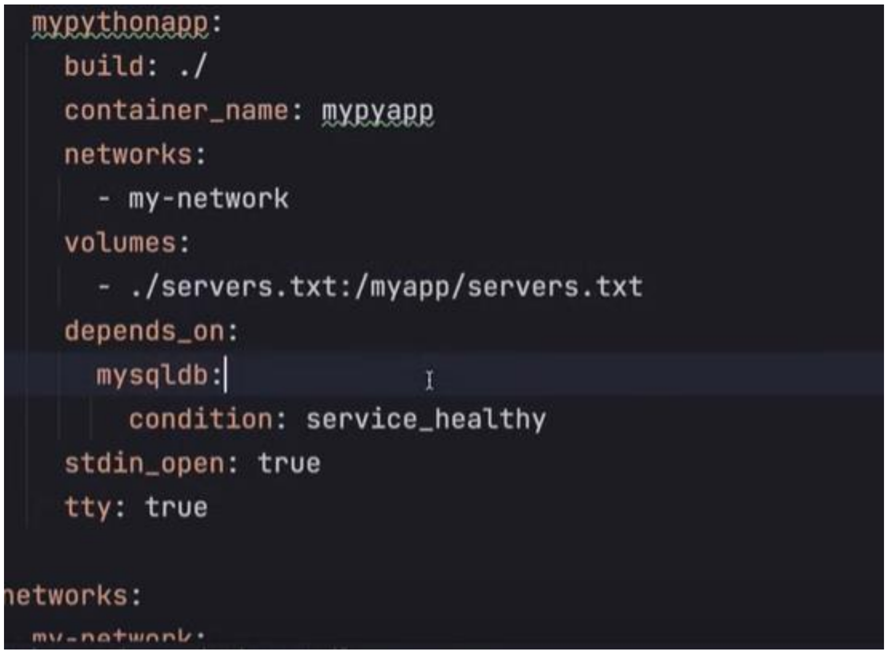
if we running python then automatically mysql container running. this happen bcoz when we add different services inside the docker compose file it will be part of single network.



```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker network ls
NETWORK ID     NAME                         DRIVER    SCOPE
cc447752c023   bridge                       bridge    local
92098af08948   host                         host      local
987f78391453   my-net                       bridge    local
d93fa4115d1e   none                         null      local
271b4cb67b64   pythondemoproject_default    bridge    local
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
```

network created automatically.

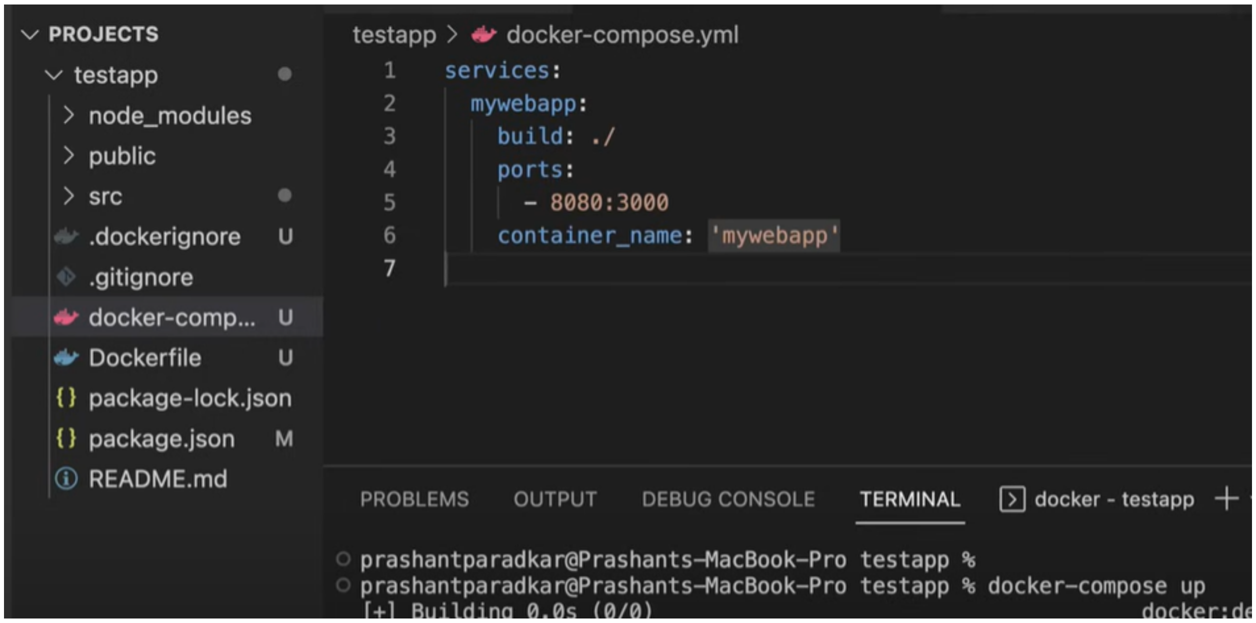
Mount Bind in compose :



```
  mypythonapp:
    build: ./
    container_name: mypyapp
    networks:
      - my-network
    volumes:
      - ./servers.txt:/myapp/servers.txt
    depends_on:
      mysqldb:
        condition: service_healthy
    stdin_open: true
    tty: true

networks:
  my-network:
```

bind server.txt on local machine to remote machine.

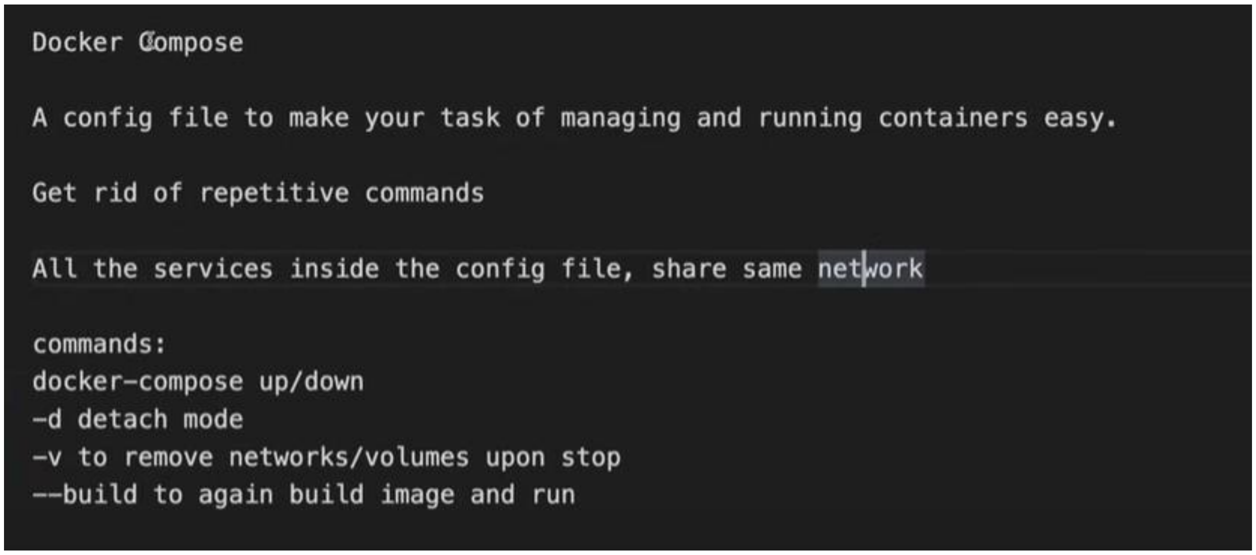
Docker Compose with Ports :



```
services:
  mywebapp:
    build: ./
    ports:
      - 8080:3000
    container_name: 'mywebapp'
```

```
prashantparadkar@Prashants-MacBook-Pro testapp %
prashantparadkar@Prashants-MacBook-Pro testapp % docker-compose up
[+] Building 0.0s (0/0)                                  docker:de
```

run on 8080 port.



```
Docker Compose

A config file to make your task of managing and running containers easy.

Get rid of repetitive commands

All the services inside the config file, share same network

commands:
docker-compose up/down
-d detach mode
-v to remove networks/volumes upon stop
--build to again build image and run
```