Step-4 Docker

25 March 2025 11:32

Containerization:

is a form of operating system virtualization that packages an application and its dependencies into a single Image. This image is known as Container.

Container:

1. A way to package an application with all the necessary dependencies and configuration.
2. It can be easily shared.
3. It makes development and deployment efficient.
4. All the containers are independent with their separate environment and sytem requiremnt.

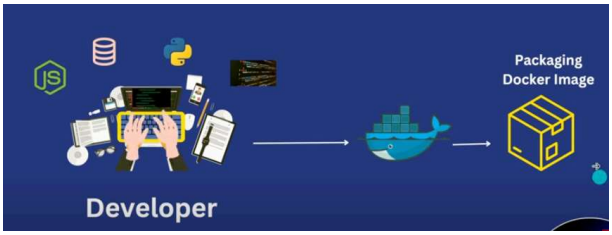
Diff Btw containers and Virtualization

1. VMs virtualize hardware, while containers virtualize the operating system.
2. Containers share the host operating system's kernel, making them lightweight and efficient. VMs include a full operating system within each virtual machine, leading to higher resource consumption.

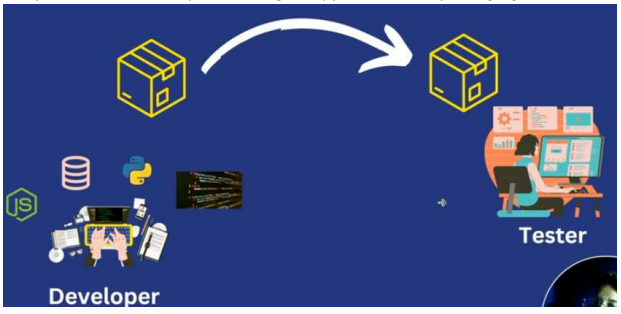
Docker Containers	VMs
Low impact on OS, very fast, low disk space usage	High impact on OS, slower, high disk space usage
Sharing, re-building and distribution is easy	Sharing, re-building and distribution is challenging
Encapsulate apps instead of whole machine	Encapsulate whole machine

Docker:

1. It is a containerization platform for developing, packaging, shipping and running application.
2. It provide us the ability to run an application in an isolated environment called a container.
3. Makes deployment and development efficient.

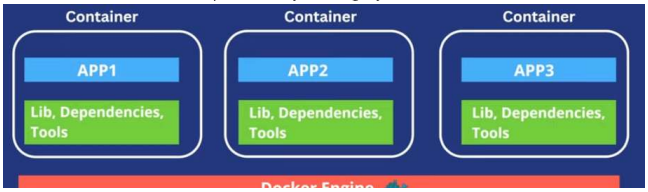


the process of a developer creating an application and packaging it into a Docker image for deployment.

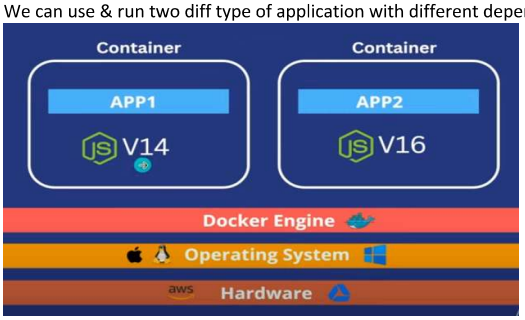


Architecture:

- Docker creates **containers** which are like independent boxes that can run any application without affecting the other containers or the underlying operating system.
- The **Docker Engine** is the tool that manages these containers.
- All of this runs on top of the **Operating System**, which in turn runs on the **Hardware**.



We can use & run two diff type of application with different dependencies over a same machine.

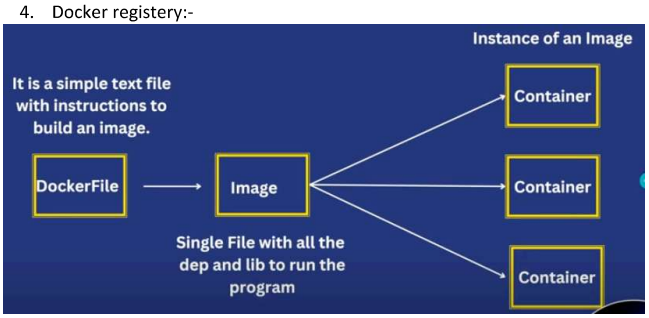


Why this is useful:

- **Isolation:** Each application runs in its own container, so they don't interfere with each other.
- **Consistency:** Containers ensure that an application runs the same way regardless of the environment (like different operating systems).
- **Efficiency:** Containers are lightweight and start quickly, making them efficient for running applications.

Components of Dockers:

1. **Docker file**
2. **Docker image:-** A Docker image is like a template or a blueprint. It contains all the code, libraries, runtime, system tools, and settings needed to run an application. It's a read-only file. You can think of it as a class in object-oriented programming.
3. **Docker container:-** A Docker container is a running instance of a Docker image. It's a lightweight, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries, and settings. You can think of it as an object created from a class.
4. Docker registery:-



The process starts with a **Docker file**. It's just a text file where you list all the instructions Docker needs to build your image.
We specify the base operating system, install any required software, copy your application code, and set up configurations.

Docker reads the Docker file and creates a **Docker image**. This image is a snapshot of your application and its entire environment, bundled together. It's like a self-contained package.

to share this image or deploy it somewhere else we use **Docker Hub or a private registry.** It's like a central library for Docker images.

In simple terms, we create a recipe (Docker file), bake the cake (image), store it in a pantry (registry), and then serve it (run)

Diff btw repository and registry
Registry:

1. Think of a registry as the overall "warehouse" or "hub" where Docker images are stored.
2. It's the top-level entity.
3. The **registry** is the building.{In simple terms}

Repository:

1. It's a collection of Docker images that are related
2. A repository is a "shelf" or "folder" within that warehouse.
3. The **repository** is a specific room inside the building.{In simple terms}

Important Commands:

1. **First create a Docker File.**

- **FROM** *node* = its means am using my base image as node
- **WORKDIR** *path* = choose a working directory {we created a folder in an empty container || now this is my working directory}
- **COPY** . . = copy all the files in this working directory.
- **RUN** *npm install* = install npm to run this file
- **EXPOSE** *3000* = defining port number where the application run.
- **CMD** *["npm", "start"]* = CMD tells Docker what to run when the container starts. {"When this Docker container starts, automatically run the command npm start, which will launch my Node.js application."}

We create a file for these commands

Dockerfile U

testapp > Dockerfile > ...

```
1  FROM node:20
2
3  WORKDIR /myapp
4
5  COPY . .
6
7  RUN npm install
8  EXPOSE 3000
9
10 CMD ["npm", "start"]
11
12
```

2. **Creating Docker Image**

- **docker build** . = instructs Docker to create a new image from the files in your current directory.
- **docker image list** = this will show us all the image of our file.

```
Usage:  docker buildx build [OPTIONS] PATH | URL | -
```

3. **Creating container from image**

- **docker run** *img_id* = this will run our image.{create and start a container from a specified Docker image.}

☆ *Docker containers run in isolated environments. They have their own file system, their own processes, and their own network. When our website runs inside a container, it's running within that container's network space, not directly on your host machine's network.*

- **docker stop** *image_name* = used to stop a running Docker container. {This img_name is a unique identifier}
- **docker ps** = show detail reading running docker container.
- **docker ps -a** = to show all container list.
- **docker run -p** *3000:3000 img_id* = this is port mapping. It tells Docker to forward network traffic from port 3000 on the host machine (your computer) to port 3000 inside the container.
- **docker rm** *image name* = to delete or drop a docker image.
- **docker logs** *container id* = to see the logs for a particular container

4. **For running docker in background so we can use terminal.**

- **docker run -d -p** *3000:3000 img_id*

```
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp> docker run -d -p 3000:3000  9b59cd4529b0
72cdfde9ead78c00ae94a668cf6d5f7ed56b84ac9524e1082ca6c7ce85d6a30b
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp>
```

We can run multiple containers from one image

```
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp> docker run -d -p 3001:3000  9b59cd4529b0
d3b02d0f3b6994861f2791eae3b53a95218ef52fd619d73c04113528f94b947a
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp> docker run -d -p 3002:3000  9b59cd4529b0
c3d3e365188c5fd4cd014c5939a2ed9913942f4e23afd76f61d49e4846d30793
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp> docker run -d -p 3003:3000  9b59cd4529b0
f27e67a1dc1265c02f93f7c53776db4627dc14f39112a75dd6834cef1ad23738
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp>
```

Here we are creating multiple docker container from one image.

these commands are creating multiple copies of the same container, each with a different port mapping. This allows you to access the same web application on different ports on your local machine.

In the example, the application inside the container consistently uses port 3000, but it's made accessible on the host via ports 3001, 3002, and 3003.

This is happening due to Docker containers, by design, are isolated environments.

- **docker run -d --rm -p** *3000:3000 img_id* = This will delete that container when we stop it.
- **docker run -d --rm --name** *"MyWeb"* **-p** *3003:3000 img_id* = this will add a speific name for that container.

```
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp> docker run -d --rm --name "MyWeb" -p 3003:3000  9b59cd4529b0
569ffc291b92522b759fb6952eb8301176fbbab7fc5986a8433a13364c96bfeb
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
569ffc291b92   9b59cd4529b0   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3003->3000/tcp   MyWeb
72cdfde9ead7   9b59cd4529b0   "docker-entrypoint.s…"   About an hour ago    Up About an hour    0.0.0.0:3000->3000/tcp   zealous_bohr
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp>
```

5. **For Image management**

- **docker image ls** = to see all the images.
- **docker build -t** *name:version .* = adding a tag to image while creating it.

```
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp> docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
<none>       <none>    40e4da8818c0   2 hours ago    2.1GB
<none>       <none>    9b59cd4529b0   2 hours ago    2.1GB
mywebapp     01        97e5cfc8d36c   2 hours ago    2.1GB
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp>
```

- **docker rmi** *img_id/img_name* = to delete image.

6. **To update our project.**

Just create a new docker image and start containers. We can use multiple version of app at a same time

7. Predefined image.

8. **Intractive containers**

Create a python program, create img, create container

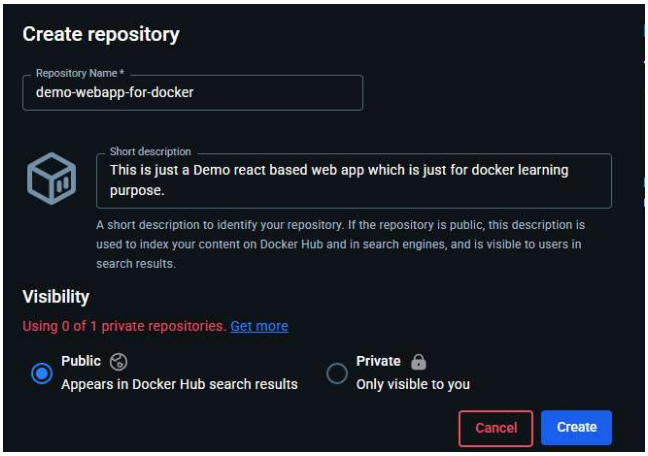
- **docker run -it** *img_id* = for interactive container.

```
(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
```

9. **Image Sharing**

We push docker image in docker hub.

First create a repo:



Pushing image to Docker hub from local system.

- **docker push** *devopskarannegi/demo-webapp-for-docker:01* = this is the name and tag for the specific image.

```
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp> docker images
REPOSITORY                                TAG       IMAGE ID       CREATED             SIZE
devopskarannegi/demo-webapp-for-docker    01        4078ba4c04af   About an hour ago   2.1GB
mywebapp                                  02        2c7804f0c6b9   About an hour ago   2.1GB
mywebapp                                  01        97e5cfc8d36c   4 hours ago         2.1GB
nginx                                     latest    124b44bfc9cc   7 weeks ago         279MB
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp> docker push devopskarannegi/demo-webapp-for-docker:01
The push refers to repository [docker.io/devopskarannegi/demo-webapp-for-docker]
1412a0c7569b: Pushed
4c9f6f19aba4: Pushed
d9be9895137: Pushed
```

```
c7c39b84cee2: Pushed
8dddd2795891: Pushed
bb871256cc0f: Pushed
8091a3a66714: Pushed
091eb8249475: Pushed
255774e0027b: Pushed
667459d47f42: Pushed
353e14e5cc47: Pushed
7cd785773db4: Pushed
01: digest: sha256:4078ba4c04af63383423b74de9a0a4a6e2fec916e3c8ecef8e367934de208e53 size: 856
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp>
```

- **docker pull** *devopskarannegi/demo-webapp-for-docker:01* = for pulling a docker image to work on it.
- **docker tag** *mywebapp:02 devopskarannegi/demo-webapp-for-docker:02* = This will change the name of image to another.

```
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp> docker push devopskarannegi/demo-webapp-for-docker:02
The push refers to repository [docker.io/devopskarannegi/demo-webapp-for-docker]
667459d47f42: Layer already exists
7cd785773db4: Layer already exists
8091a3a66714: Layer already exists
d9be90953137: Layer already exists
4c9f6f19aba4: Layer already exists
bb871256cc0f: Layer already exists
091eb8249475: Layer already exists
255774e0027b: Layer already exists
d1da9d4b4f03: Pushed
c7c39b84cee2: Layer already exists
8dddd2795891: Layer already exists
353e14e5cc47: Layer already exists
02: digest: sha256:2c7804f0c6b9ba2c80803cccb82c698fdccfa8ed708f8e1a1fe19f4fbbeeb0e9 size: 856
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\testapp>
```

When we push same image or new version docker will take only those new change and add them in into existed file.

10. **Pulling our docker image in new system.**

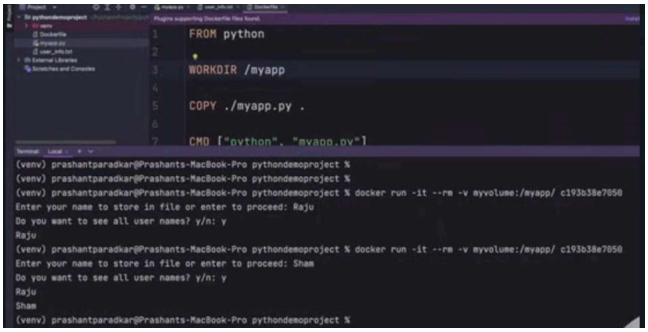
- **docker pull** *devopskarannegi/demo-webapp-for-docker:02* = this is for pulling docker img.

11. Docker Volume

☆ *Docker volumes are used to persist data generated by containers. Unlike data within a container, which is deleted when the container stops, volumes provide external storage that persists. This is useful for saving application data, sharing data between containers, and decoupling data from the container's lifecycle.*

When we running a python program{from video} locally it store all the name but when we create a Docker img it store all the data till the docker container run when it's got delete all the data is lost.

```
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\Python_Program_case\first_Senario> docker image ls
REPOSITORY                          TAG       IMAGE ID       CREATED          SIZE
volume_learn                        001       19660d2774a3   3 minutes ago    1.47GB
test                                0001      d28bc01d280a   4 hours ago      2.1GB
devopskarannegi/demo-webapp-for-docker  02        2c7804f0c6b9   7 days ago       2.1GB
mywebapp                            02        2c7804f0c6b9   7 days ago       2.1GB
devopskarannegi/demo-webapp-for-docker  01        4078ba4c04af   7 days ago       2.1GB
nginx                               latest    124b44bfc9cc   2 months ago     279MB
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\Python_Program_case\first_Senario> docker run -it --rm -v myvolume:/First_Senario/  19660d2774a3
Enter your name to store in file or enter to proceed: Karan
Do you want to see all user names? y/n:y
Karan
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\Python_Program_case\first_Senario> docker run -it --rm -v myvolume:/First_Senario/  19660d2774a3
Enter your name to store in file or enter to proceed: Puchi
Do you want to see all user names? y/n:y
Karan
Puchi
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\Python_Program_case\first_Senario>
```

- **docker run -it --rm -v** *myvolume:/First_Senario/ img_id* = myvolume:/First_Senario/ this the same path which we mentioned over the docker file. This file is managed by docker itself.
- **docker volume ls** = this is the volume created by docker and by this we can see them all
- **docker volume inspect** *volume name* = to check the details about the docker volume.
- 

When we create a volume it create a directory manage by docker which store data of docker conatiners. It is a shareable directory. Its store locally within our system.

- **docker volume ls** = to check all the list of data
- **docker volume inspect** *volume_name* = this will show all data related to that file.

Docker Mount:

When our program needs a external file to execute this. And we mount a docker container file to our local system file so that docker can use updated data.

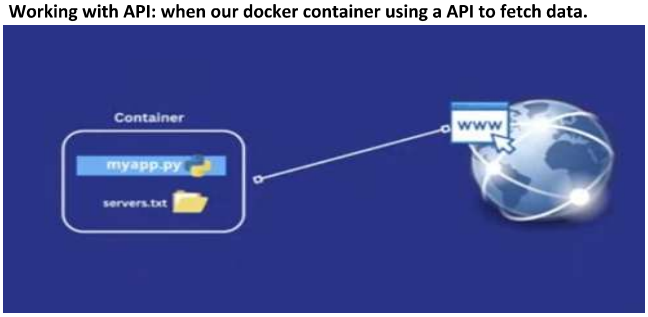
```
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\Python_Program_case\first_Senario> docker image ls
REPOSITORY                          TAG       IMAGE ID       CREATED          SIZE
volume_learn                        001       19660d2774a3   3 minutes ago    1.47GB
test                                0001      d28bc01d280a   4 hours ago      2.1GB
devopskarannegi/demo-webapp-for-docker  02        2c7804f0c6b9   7 days ago       2.1GB
mywebapp                            02        2c7804f0c6b9   7 days ago       2.1GB
devopskarannegi/demo-webapp-for-docker  01        4078ba4c04af   7 days ago       2.1GB
nginx                               latest    124b44bfc9cc   2 months ago     279MB
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\Python_Program_case\first_Senario> docker run -it --rm -v myvolume:/First_Senario/  19660d2774a3
Enter your name to store in file or enter to proceed: Karan
Do you want to see all user names? y/n:y
Karan
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\Python_Program_case\first_Senario> docker run -it --rm -v myvolume:/First_Senario/  19660d2774a3
Enter your name to store in file or enter to proceed: Puchi
Do you want to see all user names? y/n:y
Karan
Puchi
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning\Python_Program_case\first_Senario>

PS C:\Users\negid\OneDrive\Desktop\Docker_Learning> docker run -v C:\Users\negid\OneDrive\Desktop\Docker_Learning\Python_Program_case\Second_Scenario\se
```

```
rver.txt:\Second_Scenario\server.txt  125891e71c19
SERVER 1
SERVER 2
SERVER 3
SERVER 4
SERVER 5
SERVER 6
server 7
SERVER 8
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning>
```

- **docker run –v** *absolute path:/myapp/servers.txt --rm img_id* = after colon container ke ander vali location before colon is absolute path.{jab bhi hum local system ki file mei changes krenenge tab vo humare container mei bhi ho jayenge} server list ko agr hum apne local system se update krenge tab bhi humne vo changes dikhenge we mount our local system file with docker containers.
  Useful:
  When our code is depeneded on external file.
  When we are in development phase.

- .**dockerignore** = ignore those useless file which we don't want to share in a img.

12. **Docker Connections.**
1. **Working with API: when our docker container using a API to fetch data.**



We will recive a error if we will run a API image simply.

```
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning> docker run 0439128037f2
python: can't open file '/third_senario/Net_to_container': [Errno 2] No such file or directory
PS C:\Users\negid\OneDrive\Desktop\Docker_Learning>
```

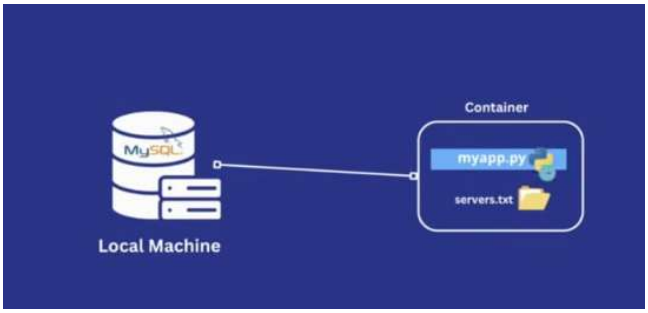
We use these commands in docker file.

```
FROM python

WORKDIR /third_senario

COPY ./Net_to_container.py .
RUN pip install requests

CMD ["python", "Net_to_container"]
```

After using run command the container will install request by using pip.

```
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject % docker run 63d5dfdf76d1
Random Cat Fact:
{"data":["Purring not always means happiness. Purring could mean a cat is in terrible pain such as during childbirth. Kitten wil
l purr to their mother to let her know they are getting enough milk while nursing. Purring is a process of inhaling and exhaling
, usually performed while the mouth is closed. But don't worry, if your cat is purring while your gently petting her and holding
 her close to you - that is a happy cat!"]}
(venv) prashantparadkar@Prashants-MacBook-Pro pythondemoproject %
(venv) prashantparadkar@Prashants-MacBook-Pro,pythondemoproject % docker run 63d5dfdf76d1
Random Cat Fact:
{"data":["Studies now show that the allergen in cats is related to their scent glands. Cats have scent glands on their faces and
 at the base of their tails. Entire male cats generate the most scent. If this secretion from the scent glands is the allergen,
allergic people should tolerate spayed female cats the best."]}
```

2. **Connect local SQL data base with docker container.**

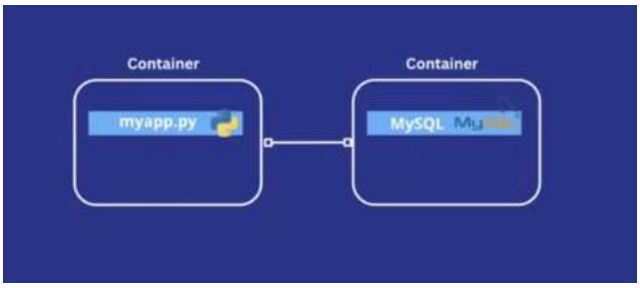


For docker file.

```
FROM python

WORKDIR /myapp

COPY ./sql_demo.py .

RUN pip install pymysql

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

docker run -it --rm *image id* = to run that docker container.
What we doing are here we want to connect our local host SQL data base to an isolated docker container.
How to handle this tell docker to target local machine where the docker is installed.

3. **Container to container**



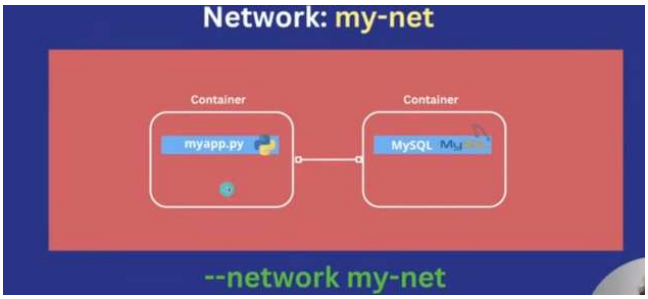
docker inspect *container id* = this is to inspect info of containers.
Now we need to target our one container to the other containers public IP. Containers also have public IP because they are independent.

Docker file content.

```
WORKDIR /myapp

COPY ./sql_demo.py .

RUN pip install pymysql
RUN pip install cryptography

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

Now for this example SQL container will run in background and we will run python container again and again. To insert data.

13. Docker Networks



First create docker network then insert both containers inside it.

- **docker network create my-net** = this will create a docker network.
- **docker network ls** = to see all the network.
- **docker run -it --rm --network** *my-net image id* = to run container on docker network.

14. Docker compose

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

- **For one Container.**
  Create a file = docker-compose.yaml

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

- **docker-compose up** = this will create an image and container both by using and following that file instructions.
- **docker-compose down** = this will stop the container but image will be there, volumes, network will be there.
- **docker-compose up -d =** run that container in background

For Multiple containers.

```
services:
```

When we use docker compose file with diff containers all the containers are in a same network.