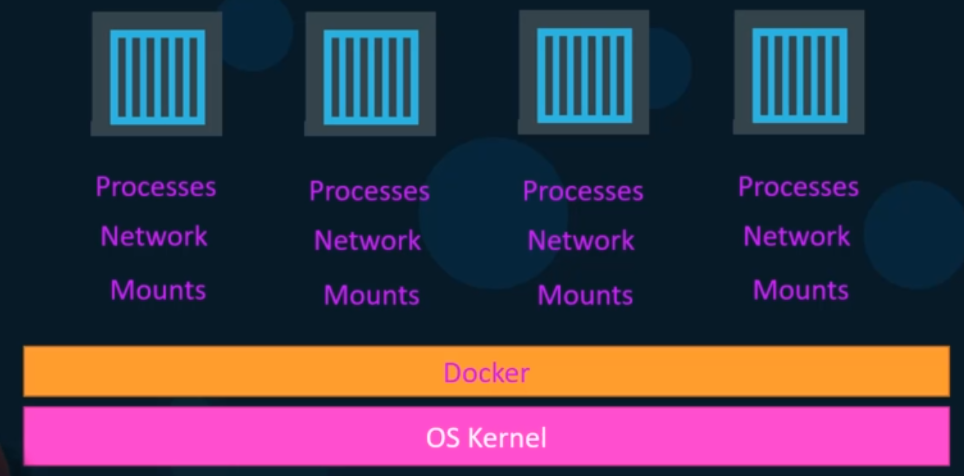
# Docker

What are Containers ?

Containers are completely isolated environments, As they can have their own processes ,services, network interfaces, mounts just like virtual machines except they all share the same OS kernel.

Docker utilizes LXC.

Different type of containers are LXC, LXD, LXCFS etc.



**Image**

An Image is a package or a template and it is used to create one or more containers.

**Containers**

Containers are running instances of image that are isolated and have their own environments and set of processes.

Containers are meant to run a specific task or process such as to host an instance of a web server or application server or a database to simple carry some kind of computation or analysis task.

Once the task is complete the container exits, a container live as long as the process inside it is alive.

## Docker Engine

The docker engine contains three components

1) Docker CLI

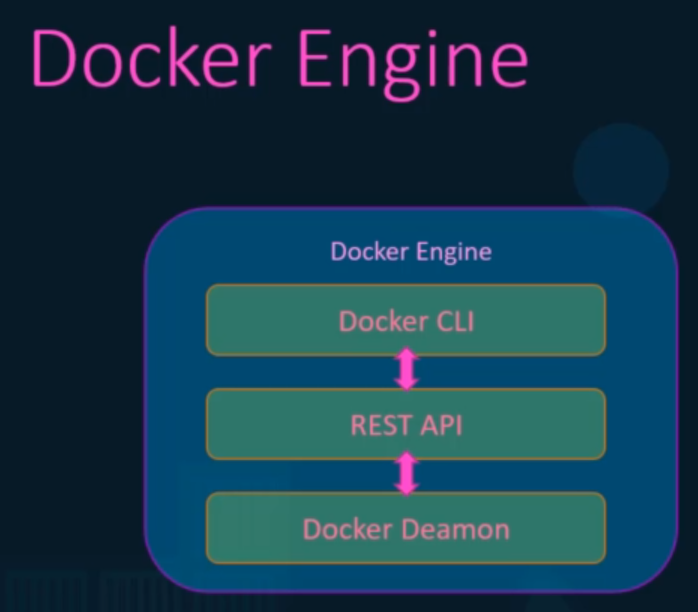
2) REST API

3) Docker Deamon

**Docker Deamon -** is a background process that manages docker objects such as images, containers, volumes and networks.

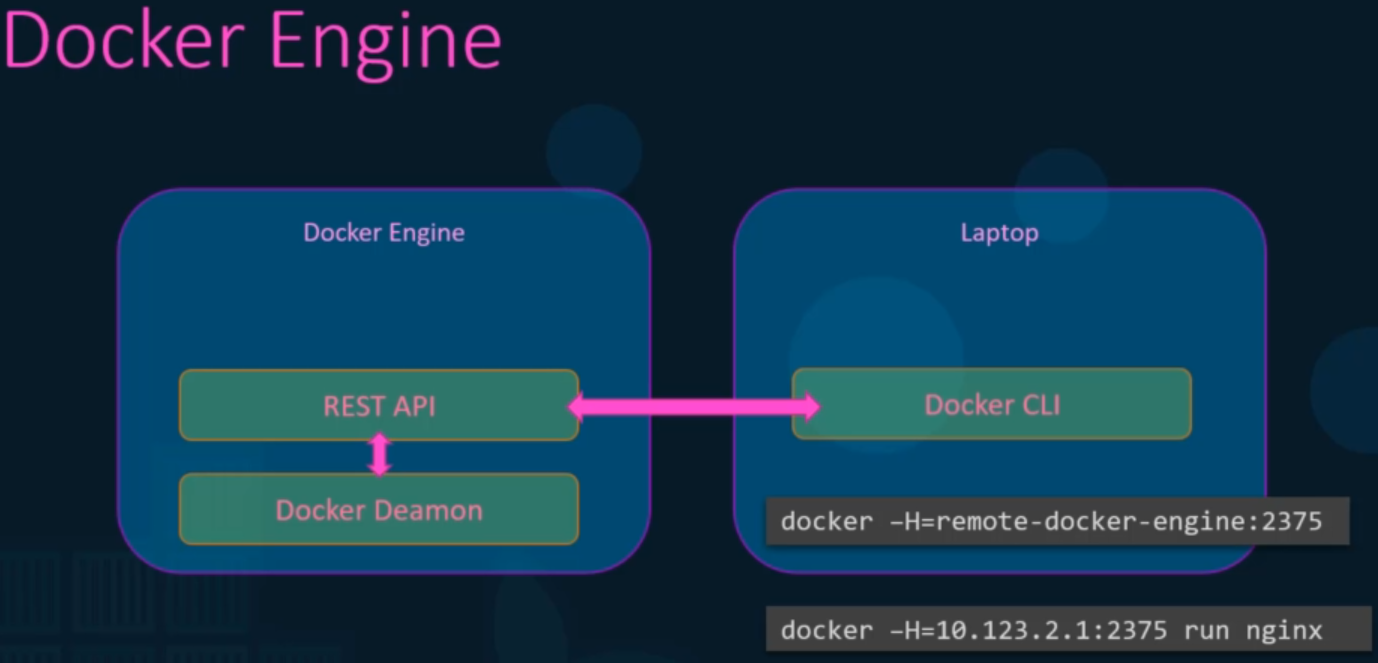
**Docker API Server –** is the API interface that programs can use to talk to the Deamon and provide instructions.

**Docker CLI –** is a command line interface used to perform actions such as running or stopinf containers. It uses the REST API to interact to the docker deamon.



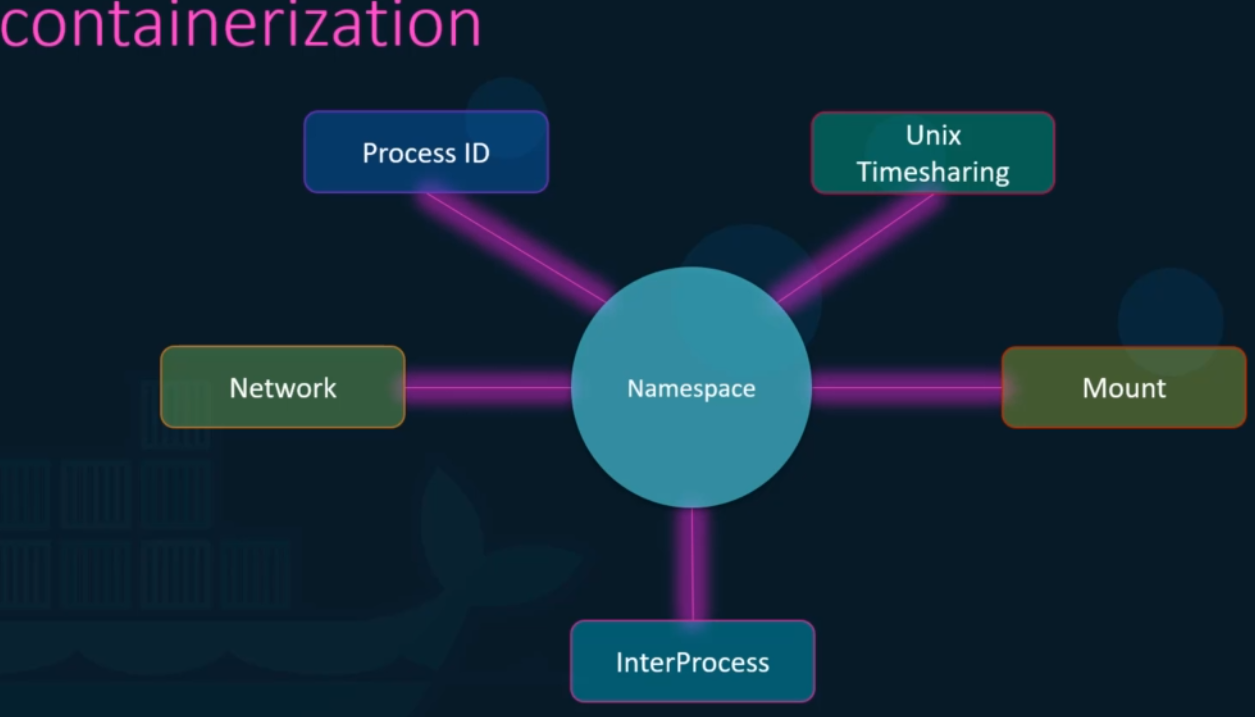
To connect to remote docker engine use command

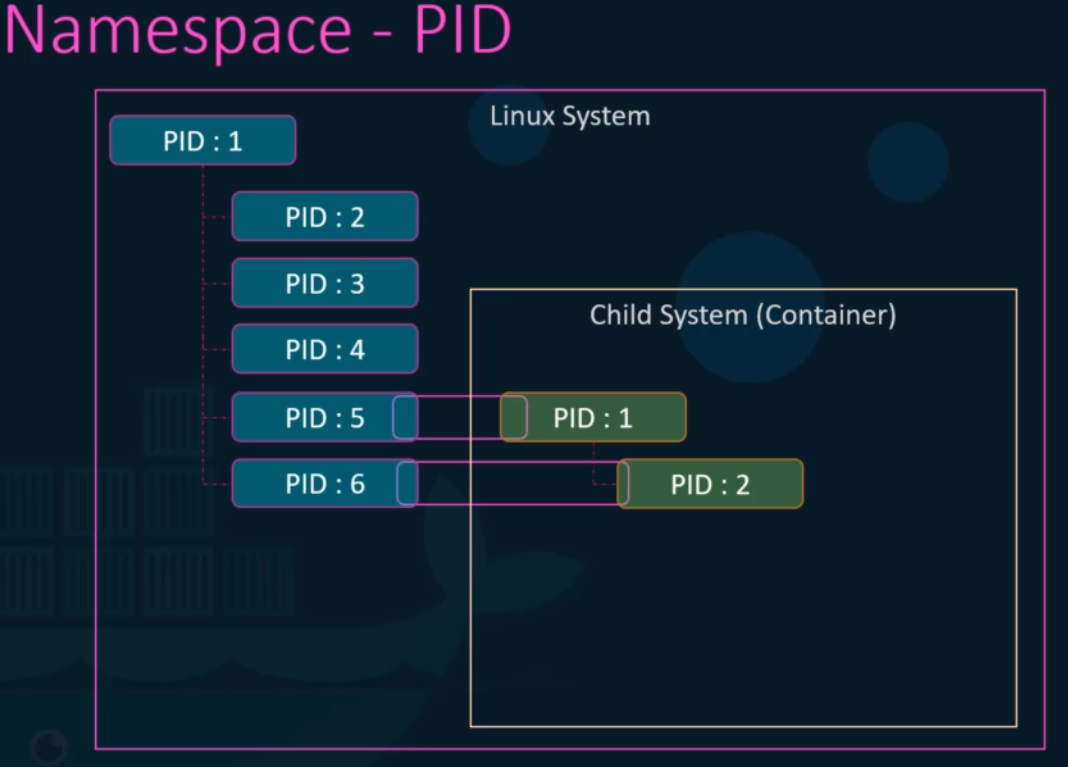
docker -H={remote-docker-engine:2375} run nginx



## Containerization

Docker uses namespace to isolate workspace process ID, Network, Interprocess communications(IPC ), Mount, Unix time sharing sytems are created in their own namespace thereby providing isolation between containers.



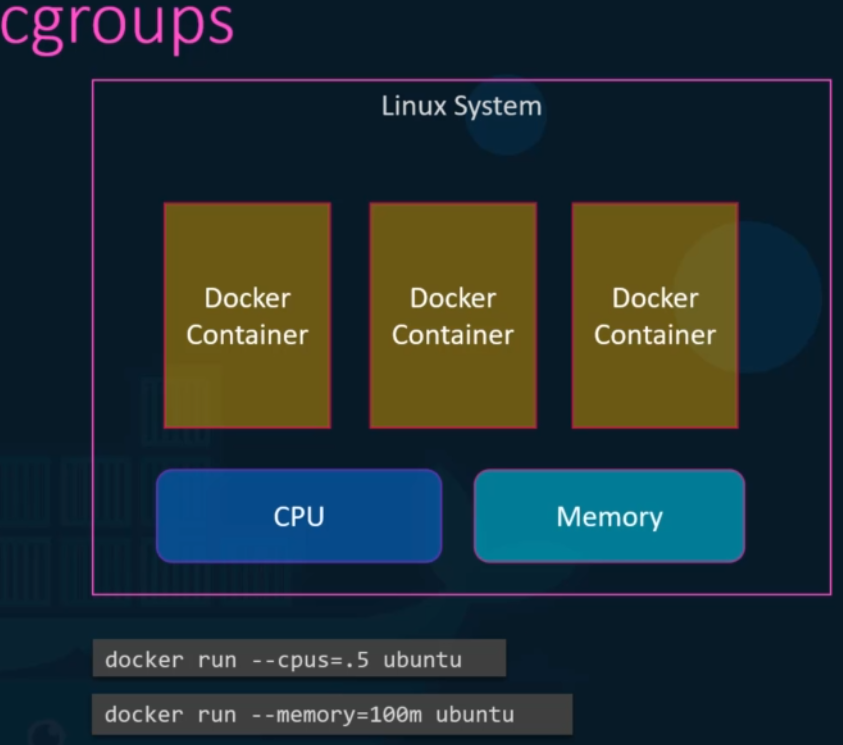


**Cgroups(Control groups)**

Cgroups restricts container utilize all the system resources such as CPU, Memory

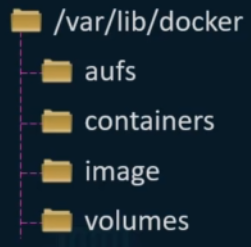
docker run –cpus=.5 ubuntu

docker run –memory=100m ubuntu



**Docker Storage**

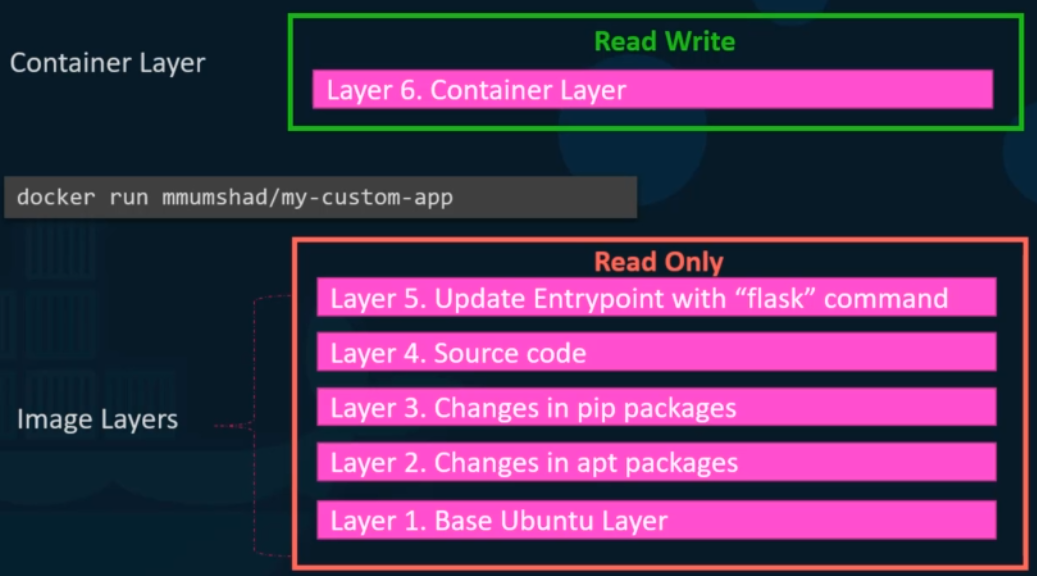
**File System**

****

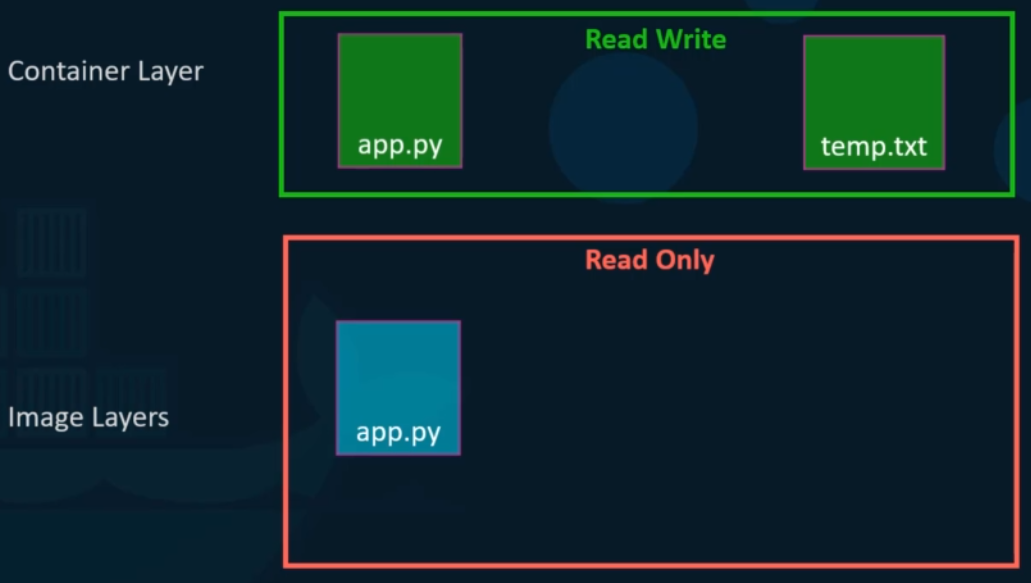
**Layered Architecture**

When docker builds images it builds these in a layered architecture.

Each line of instruction in the docker file creates a new layer in the docker image with just the changes from the previous layer.



**COPY-ON-WRITE**



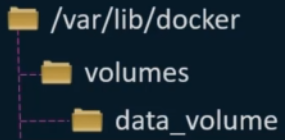
**Volumes**

docker volume create –name {volumename}

docker volume ls

docker volume inspect {volumename}

It creates a folder under volumes directory

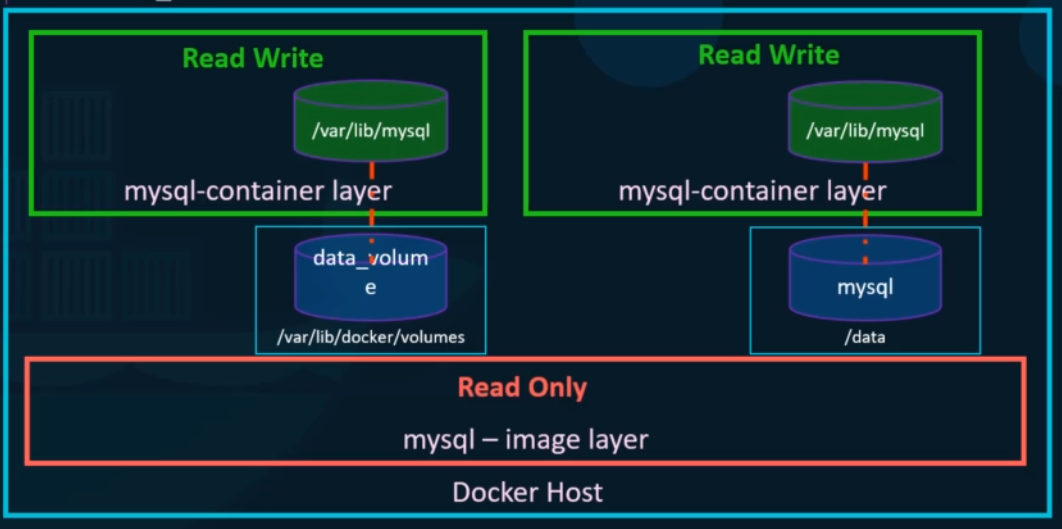


**Mounts**

There are two types of mounts

1) Volume mount

2) Bind Mount



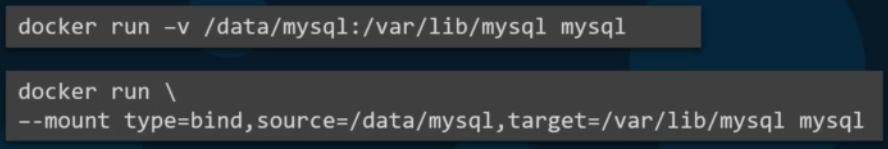
**Volume mount**

docker container run -v data\_volume:/var/lib/mysql mysql

docker container run -v data\_volume2:/var/lib/mysql mysql

if volume **data\_volume2** not created manually the docker will automatically created for you at command execution

**Bind mount**



**Storage Drivers**

Storage Drivers are responsible for the below operation.

1) Create and maintain layered architecture

2) Copy on Write

3) Mounting

The different type of storage drivers are given below.

AUFS

ZFS

NTRFS

Device Mapper

Overlay

Overlay2

**Networking**

Docker creates three networks automatically that are given below.

1. Bridge
2. None
3. Host

Bridge is the default network a container gets attached to.

**Bridge**

Bridge is the private internal network created by Docker on the host.

All containers attached to this network by default and they get an Internal IP address usually in the range 172.17 series.

The containers access each other using this internal IP if required

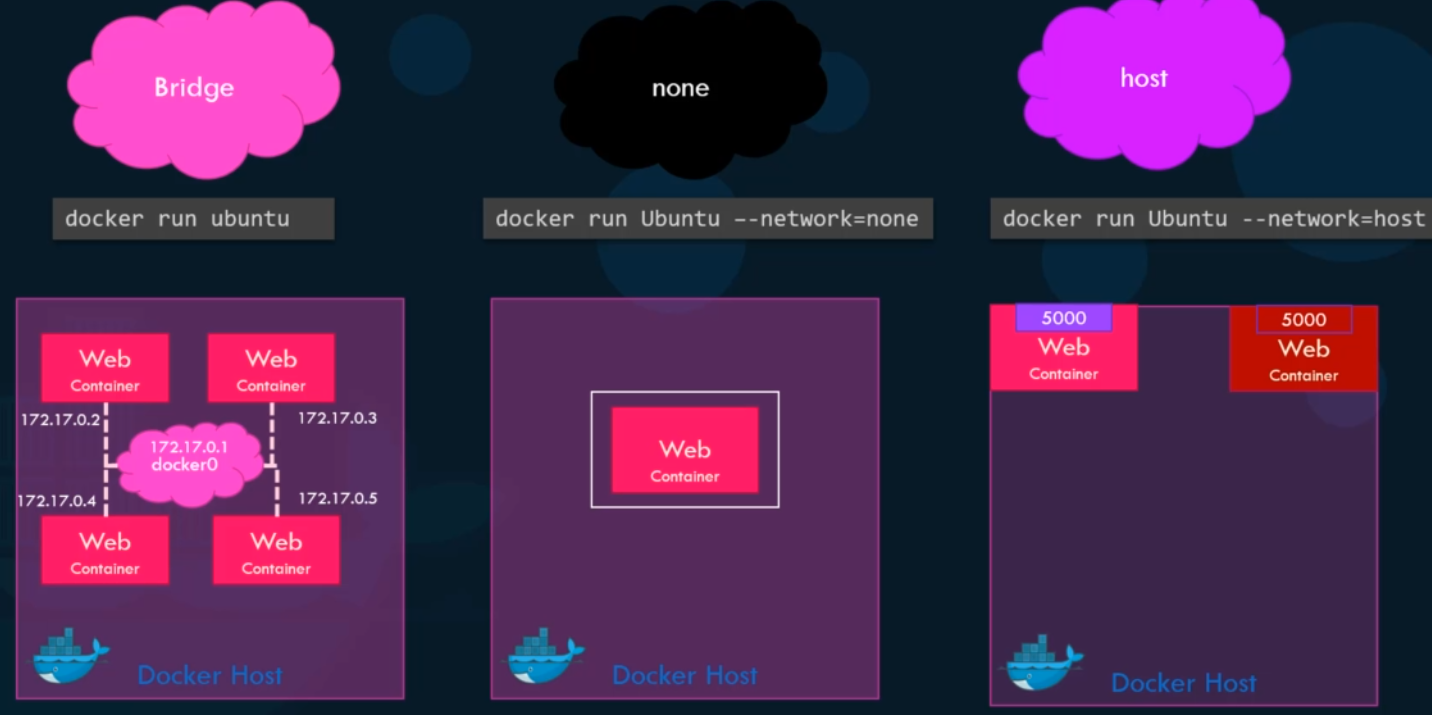
**Host**

To access the containers externally is to associate the container to the host network.

This takes out any network isolation between the docker host and the docker container.

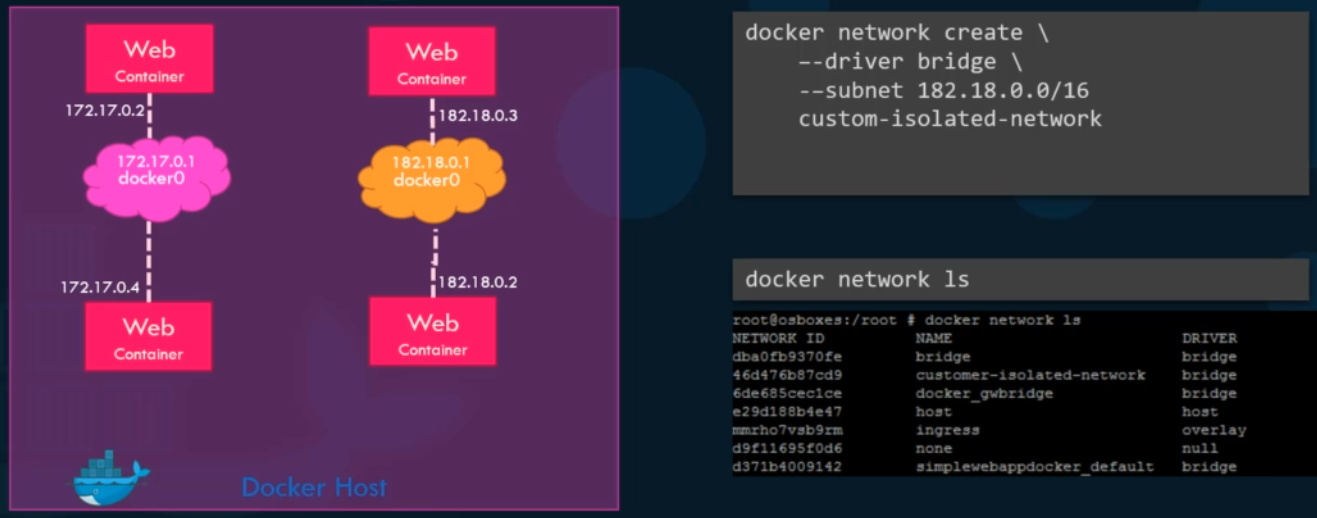
**None**

The containers are not attached to any network and doesn’t have any access to the external network or other containers they run in an isolated network.



**User-defined networks**

docker network create –driver {driverName} –subnet {182.18.0.0/16} {networkName}



**Embedded DNS**

All containers in the docker host can resolve each other with the name of the conatiner.

Docker has a built in DNS Server that helps the conatiners to resolve each other using the container name.

The built in DNS server always run at address 127.0.0.11

