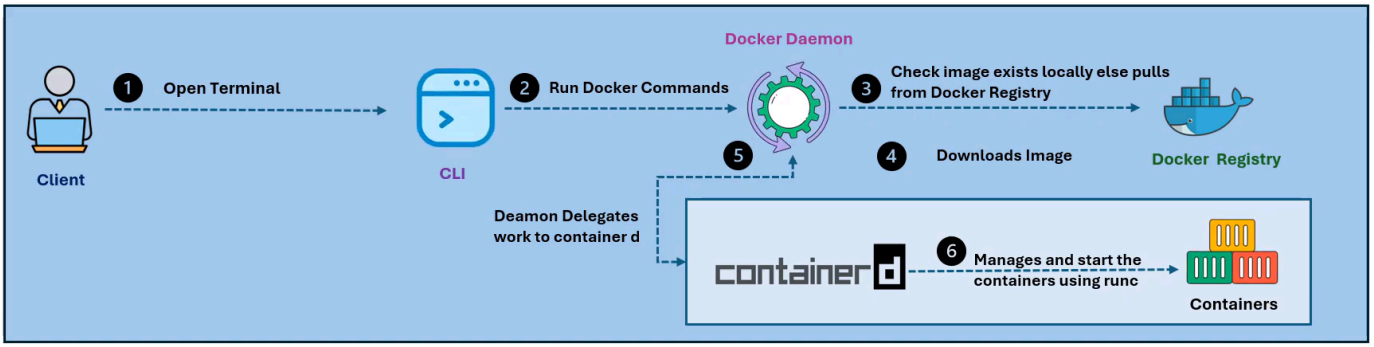
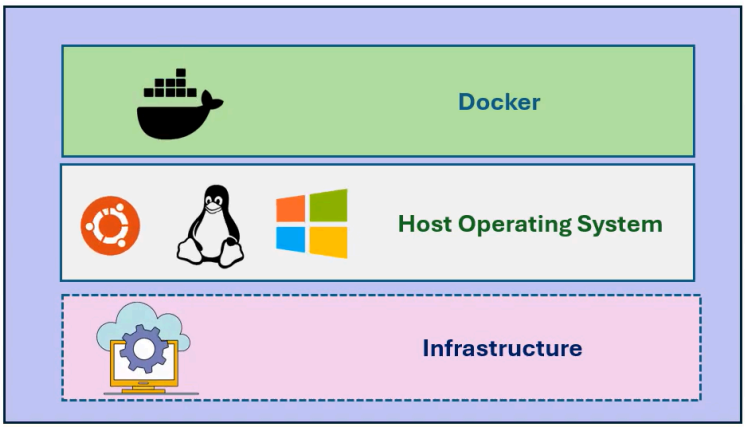
Docker Architecture



Docker Setup



- Docker Daemon Validates Request and checks image exists locally and also prepares container configuration.
- Docker Registry is a central hub to store image
- Container d responsible for image management, container lifecycle and storage.
- Runc is the lower level container execution runtime which actually starts the container.

Docker Set up:

1) Needs an infrastructure either it should be a PC or an VM
2) On top of it should have any Host Operating system linux - ubuntu redhat centos , Windows
3) Once Host Os is ready, enable virtualisation and install docker on it.

Docker Architecture:

A client or user opens a terminal and uses the docker cli to issue docker commands (build run push) ,these commands are passed to docker daemon

Docker daemon validates the request and checks image exist locally and also prepare container configuration
Docker daemon checks the image locally or not else it will pulls from docker registry

Docker registry is central hub to store all the images

Once an image is available in Docker registry it will pull from docker registry to docker host.

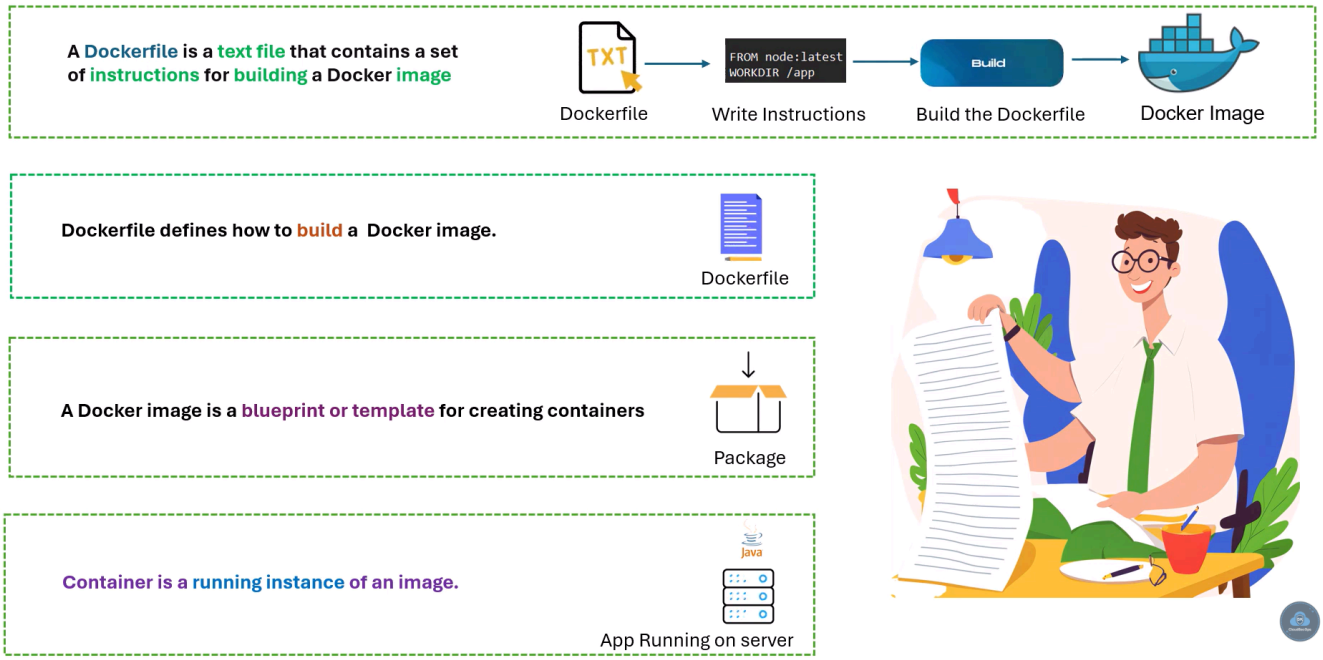
Once the docker image is successfully downloaded to docker host,
docker daemon delegates the work to **container D.**

Container D is the special service which is responsible for image management, container lifecycle and storage

Container D manages and start the container by using **runc**

Runc is the lower level container execution runtime which actually starts the container

What is Docker file and its Instructions

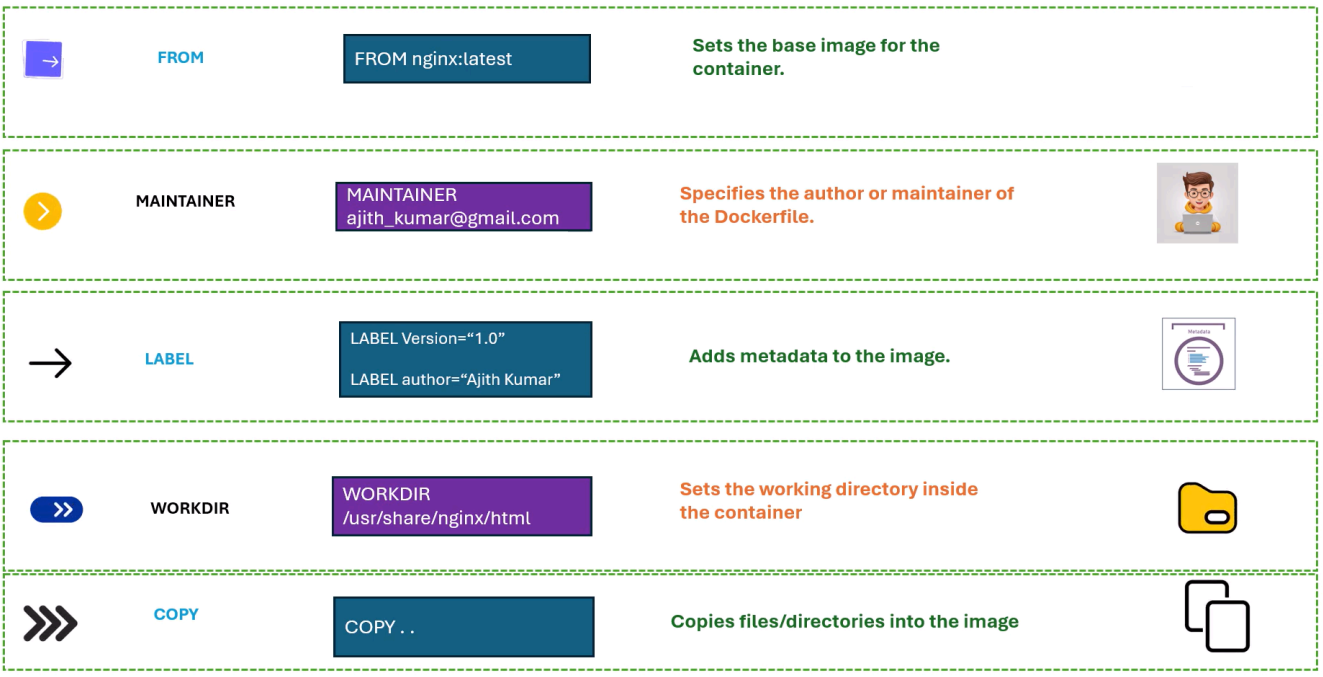


Dockerfile is an text file ,it's contain a set of instructions to build the docker image
Dockerfile defines how to build an docker image

A docker image is a blue print or read only template for creating containers

Container is an running instance of an image.

Dockerfile Instructions - Part 1



FROM : for any application to run need an parent image which is readily available in the docker hub

MAINTAINER : it is to specify the author or the MAINTAINER of this dockerfile

LABEL : it's mainly used to add the metadata to the image

Metadata is nothing but additional information related to the docker image
version of the image: 1.0
Author name : jagadish

WORKDIR :

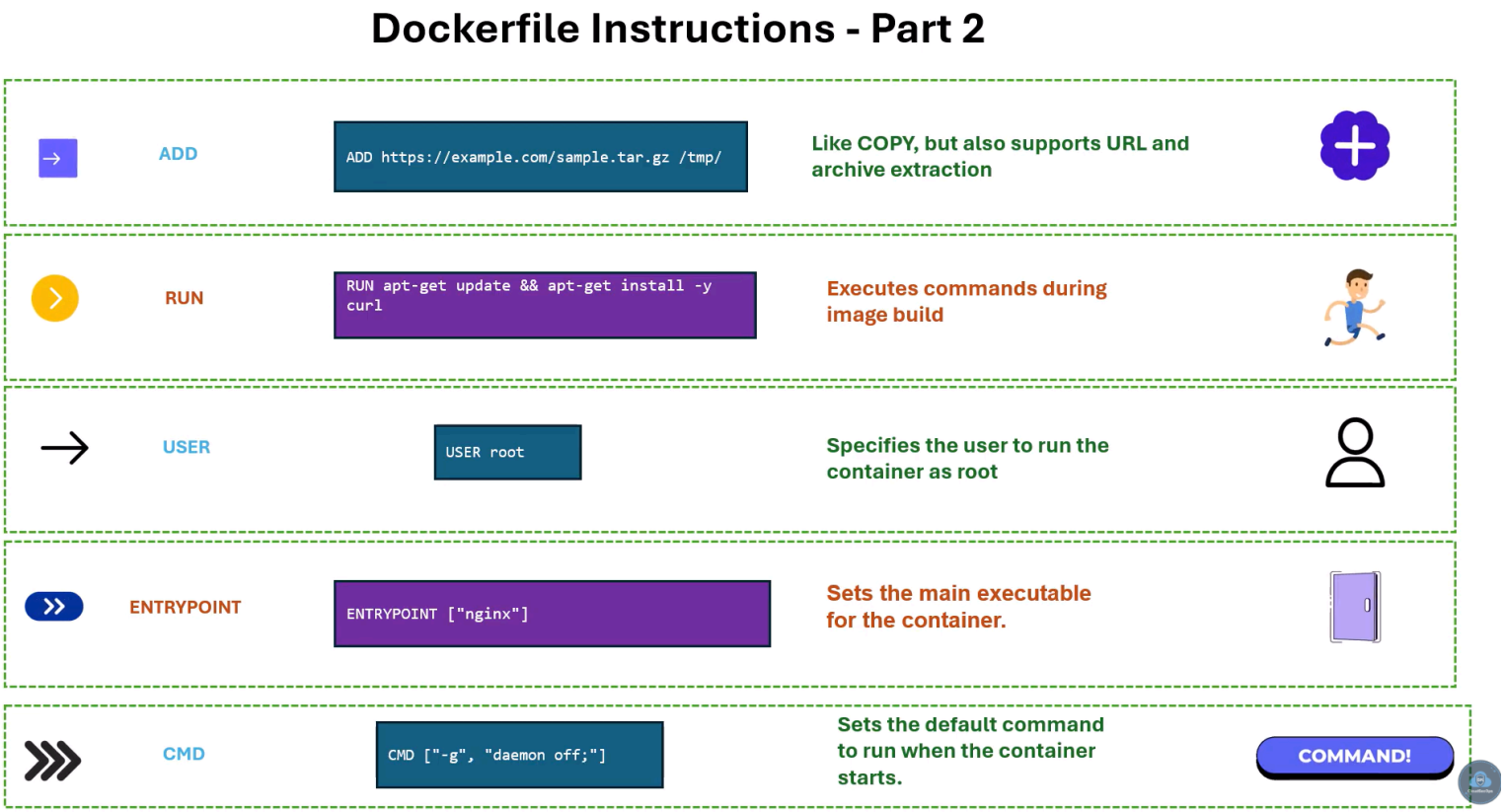
/usr/share/nginx/html/

Sets the directory inside in the container

Any subsequent operations or command will perform in this directory

COPY : Copy command is useful to copying the files and directories from the host machine to container

. . —} first dot represent the path of the host machine that is current directory of server or a path in git bash
Second dot represent the container path -' container working directory ex :- /usr/share/nginx/html



Difference between CMD and Entrypoint

CMD

- Specifies the `Command` to run when the container starts.
- **Can be overridden** when starting a container
- Provides **Default arguments** for the Entrypoint or the **default command** to run

Entrypoint

- Specifies the main `Command` that always runs.
- **Can't be overridden** when starting a container
- **Fixed entry gate** for container's execution.

Dockerfile

```
FROM ubuntu
ENTRYPOINT ["echo"]
CMD ["Hello, World!"]
```

- `FROM ubuntu` → Base image
- `ENTRYPOINT ["echo"]` → Main executable command
- `CMD ["Hello, World!"]` → Can be overriden

1) `docker run myapp` → O/P: Hello World

`docker build –t myapp .`

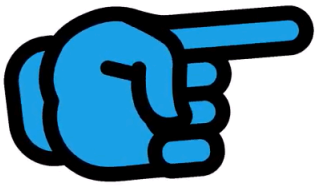
2) `docker run myapp Hi` → O/P: Hi

Difference between CMD and Entrypoint

Scenario 1: CMD as Default command (can be fully overriden)

Dockerfile

```
FROM ubuntu

CMD ["echo","Hello from CMD"]
```



Build & Run Flow:

- `docker build -t myapp .`
- `docker run myapp`

  O/P: Hello from CMD
- `docker run myapp echo "Hello override"`

  O/P: Hello override

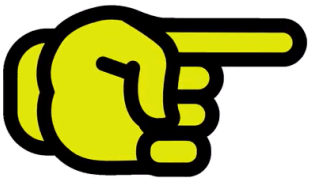
Use Case: When you want to provide a default command that the user can completely override at runtime.

Difference between CMD and Entrypoint

❖ **Scenario 2: ENTRYPOINT as fixed command (Appends Arguments)**

Dockerfile

```
FROM ubuntu

ENTRYPOINT ["echo", "Fixed Entry"]
```



Build & Run Flow:

- docker build -t myapp .
- docker run myapp

O/P: Fixed Entry

- docker run myapp "Extra"

O/P: Fixed Entry Extra

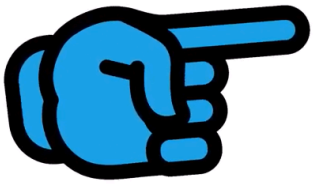
Use Case: When the container should always execute a specific binary, with user inputs added as arguments.

Difference between CMD and Entrypoint

❖ **Scenario3 : ENTRYPOINT with CMD as default arguments**

Dockerfile

```
FROM ubuntu

ENTRYPOINT ["echo"]

CMD ["Hello from CMD"]
```



Build & Run Flow:

- docker build -t myapp .
- docker run myapp

O/P: Hello from CMD

- docker run myapp "Overridden"

O/P: Overridden

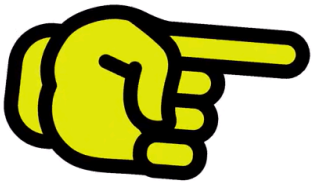
Use Case: when the base command should remain constant, but default arguments can be overridden by the user.

Difference between CMD and Entrypoint

❖ **Scenario4: Script execution with ENTRYPOINT and CMD**

Dockerfile

```
FROM ubuntu
.
.
.
COPY script.sh /script.sh
RUN chmod +x /script.sh
ENTRYPOINT ["/script.sh"]
CMD ["DefaultArg"]
```



Build & Run Flow:

- docker build -t myapp .
- docker run myapp

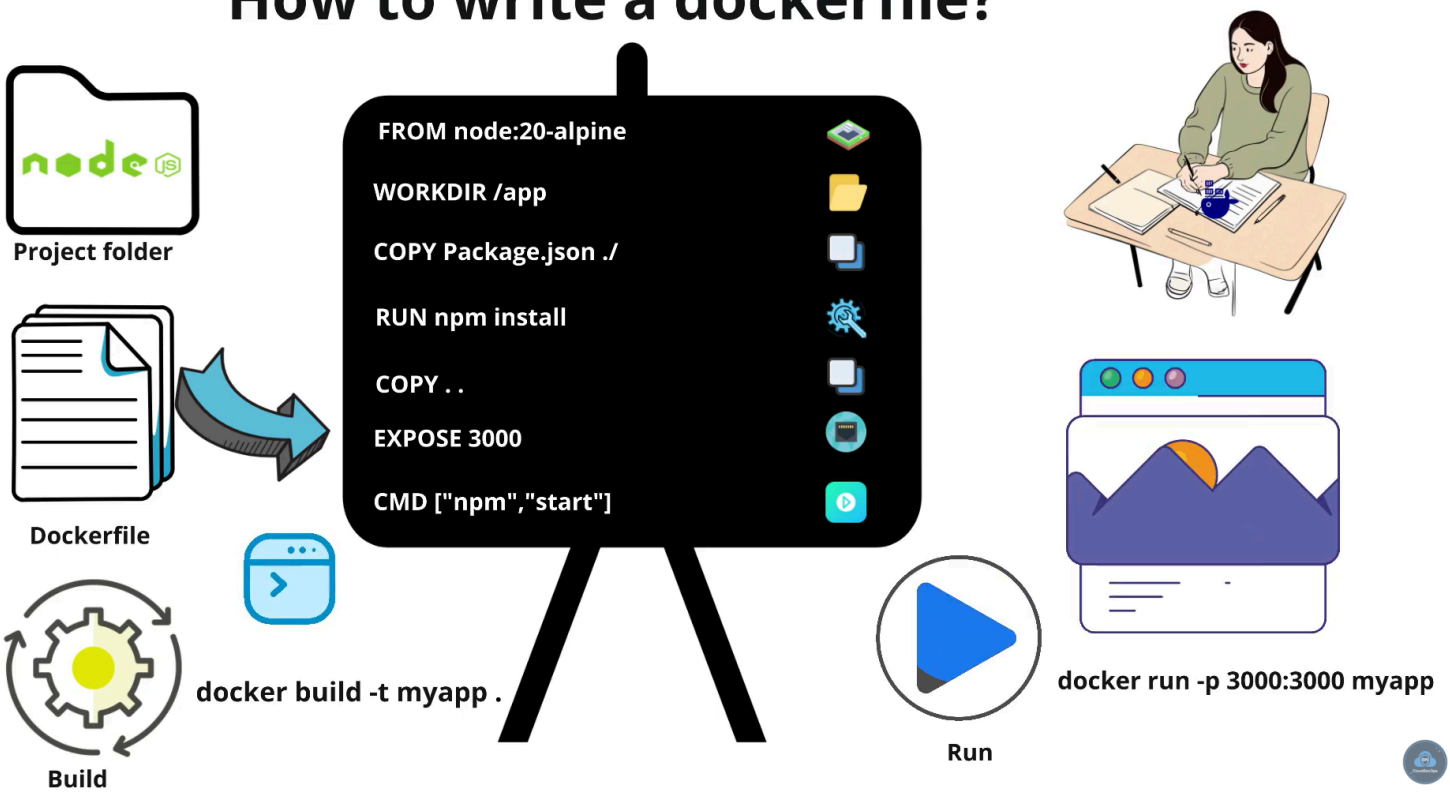
O/P: DefaultArg

- docker run myapp CustomArg

O/P: CustomArg

Use Case: Use a script for complex logic and keep it flexible for runtime inputs

How to write a dockerfile?



Overlay

Bridge

MacvLan

Network Drivers

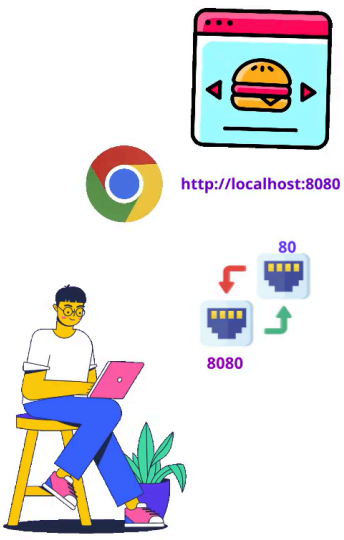
Host

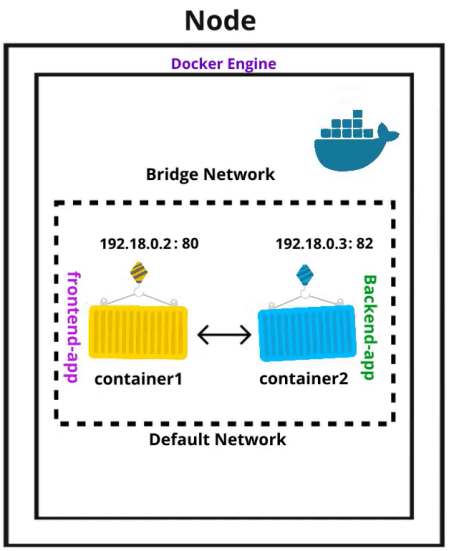
Custom

None

ttps://www.youtube.com/watch?v=xrUGEoUpa3s

Bridge Network





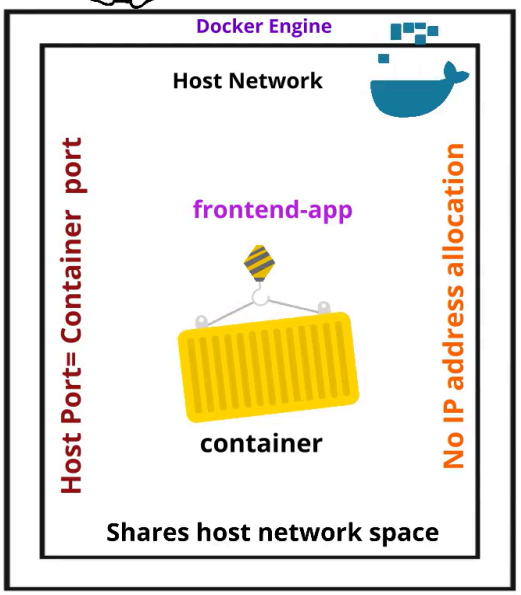
Network Drivers

```
docker network create -d bridge \
myapp-bridge

docker run -d -p 8080:80 --name frontend-app \
--network myapp-bridge nginx

docker run -d  -p 8081:82--name backend-app \
--network myapp-bridge  alpine \
sh -c "sleep 3000"

Docker exec -it backend-app /bin/bash

Ping 192.18.0.2

Curl http://frontend-app
```

Definition: Default network connect type connects two or more containers on the same host, Creates internal network on the host so that containers can communicate each other through using private ip's.

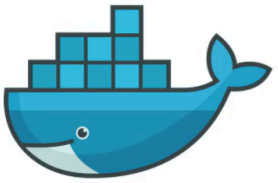
Host Network

http://localhost:80



Network Drivers

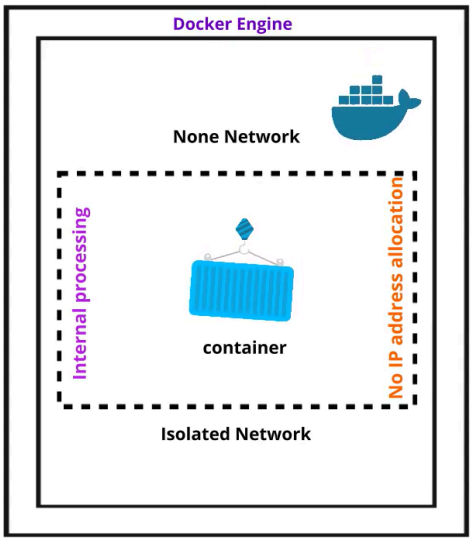
```
docker run -d  name frontend-app \
--network host nginx
```



Docker Desktop

Definition: Containers dont have any IP address they will be directly created in the system network which will remove isolation between the host and containers.

None Network



Network Drivers

```
docker run -d --network none --name \
isolated-container alpine
```

Definition: IP addresses won't be assigned to containers. These containers are not accessible to us from the outside or from any other container.

Kubernetes - Orchestration Tool