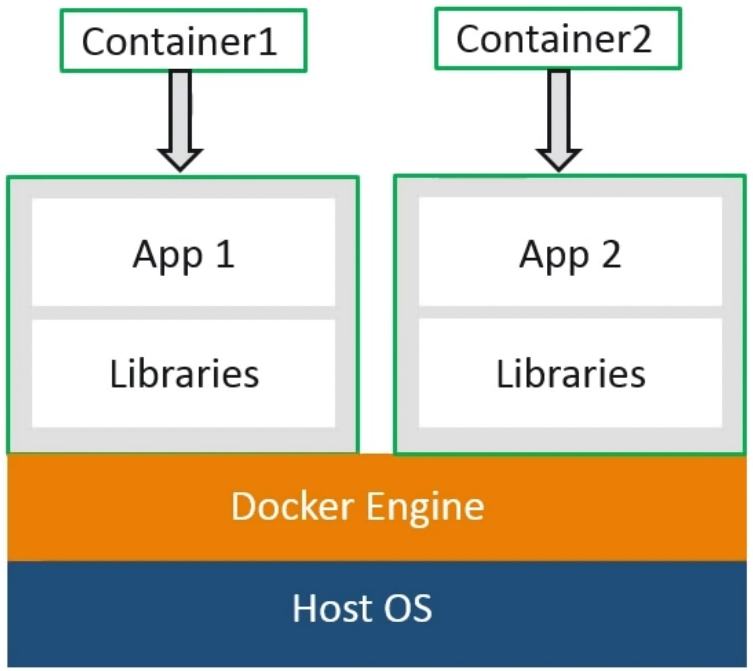
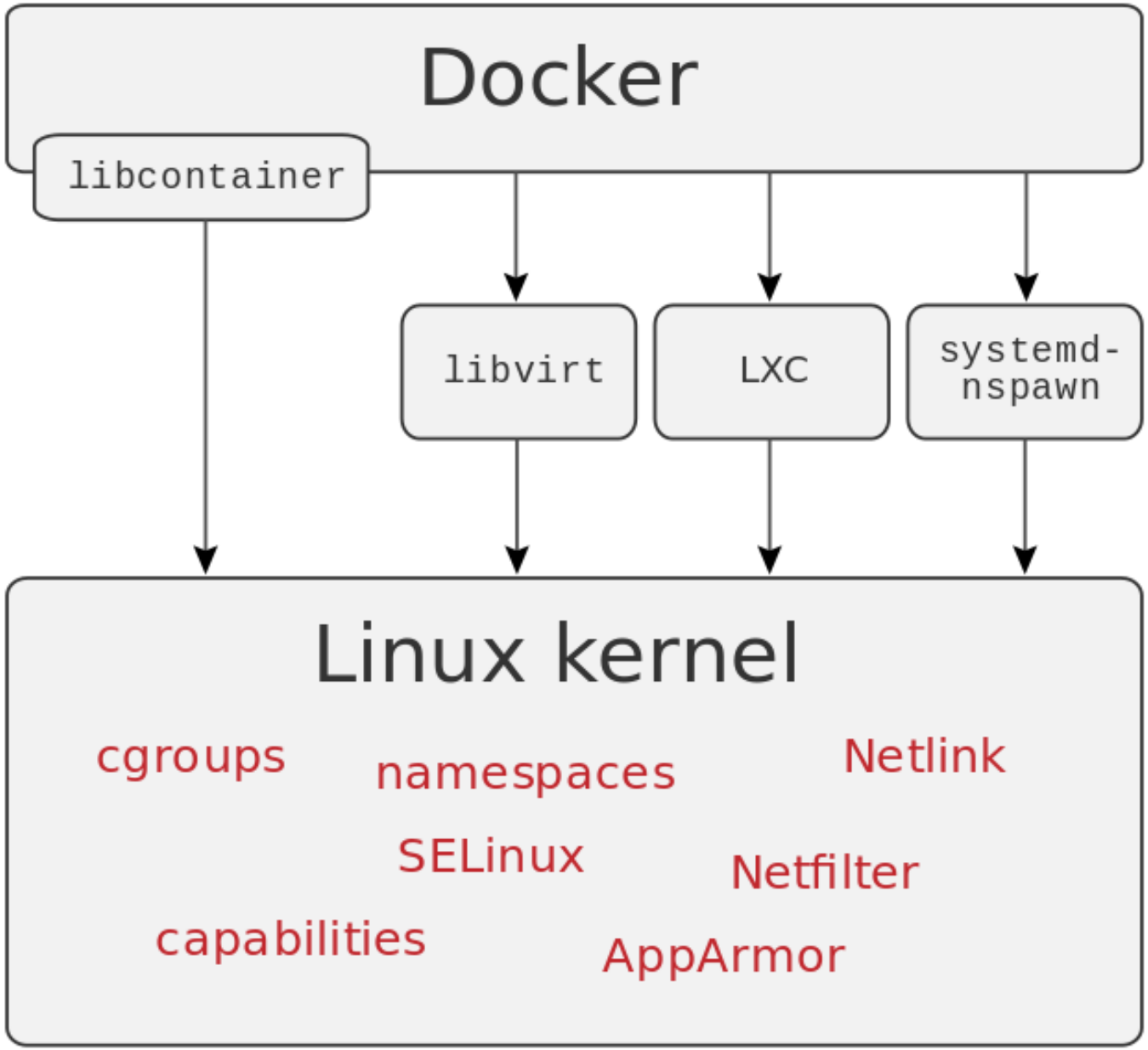
GUEST LECTURE

31.10.2023 – Mr. Dazzle B

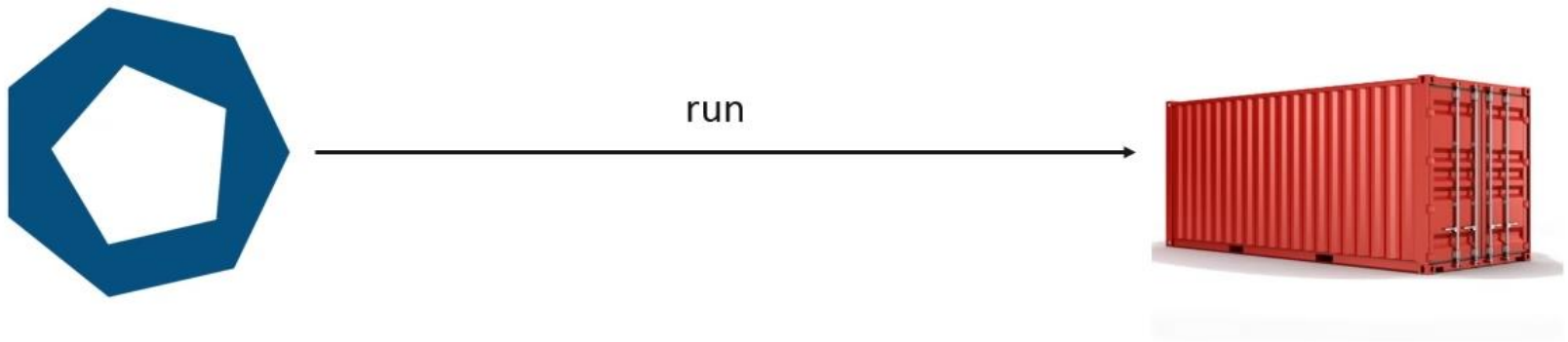
What is Docker?

- Docker makes it easier to create, deploy and run applications by using containers
- Docker containers are lightweight alternatives to Virtual Machines and it uses the host OS
- You don't have to pre-allocate any RAM in containers




Docker
libcontainer
libvirt
LXC
systemd-
nspawn
Linux kernel
cgroups
namespaces
Netlink
SELinux
Netfilter
capabilities
AppArmor

Docker Image & Container



Docker Image

Docker Container

- Read Only Template Used To Create Containers
- Built By Docker Users
- Stored In Docker Hub Or Your Local Registry

- Isolated Application Platform
- Contains Everything Needed To Run The Application
- Built From One Or More Images

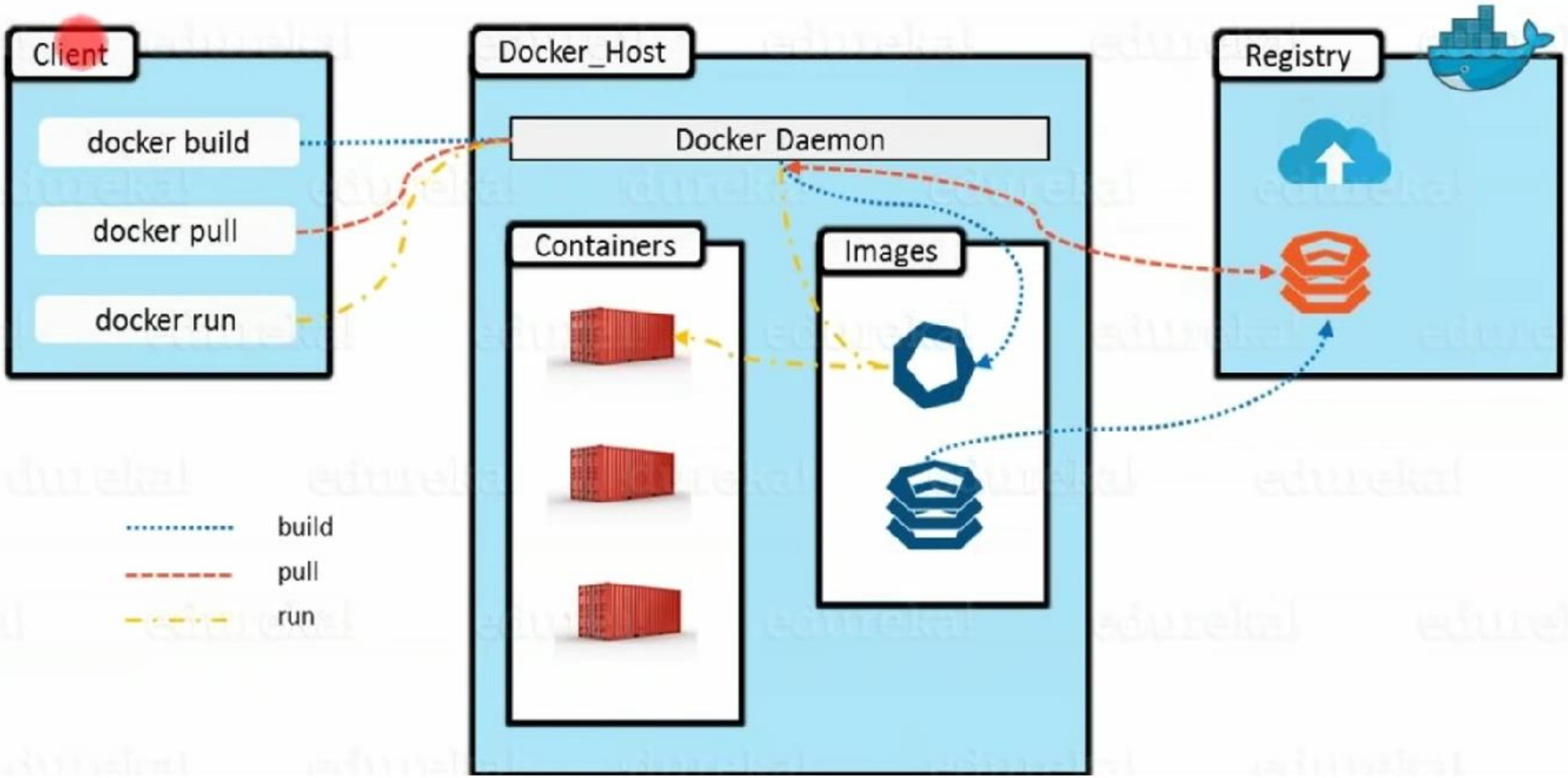
Docker Registry

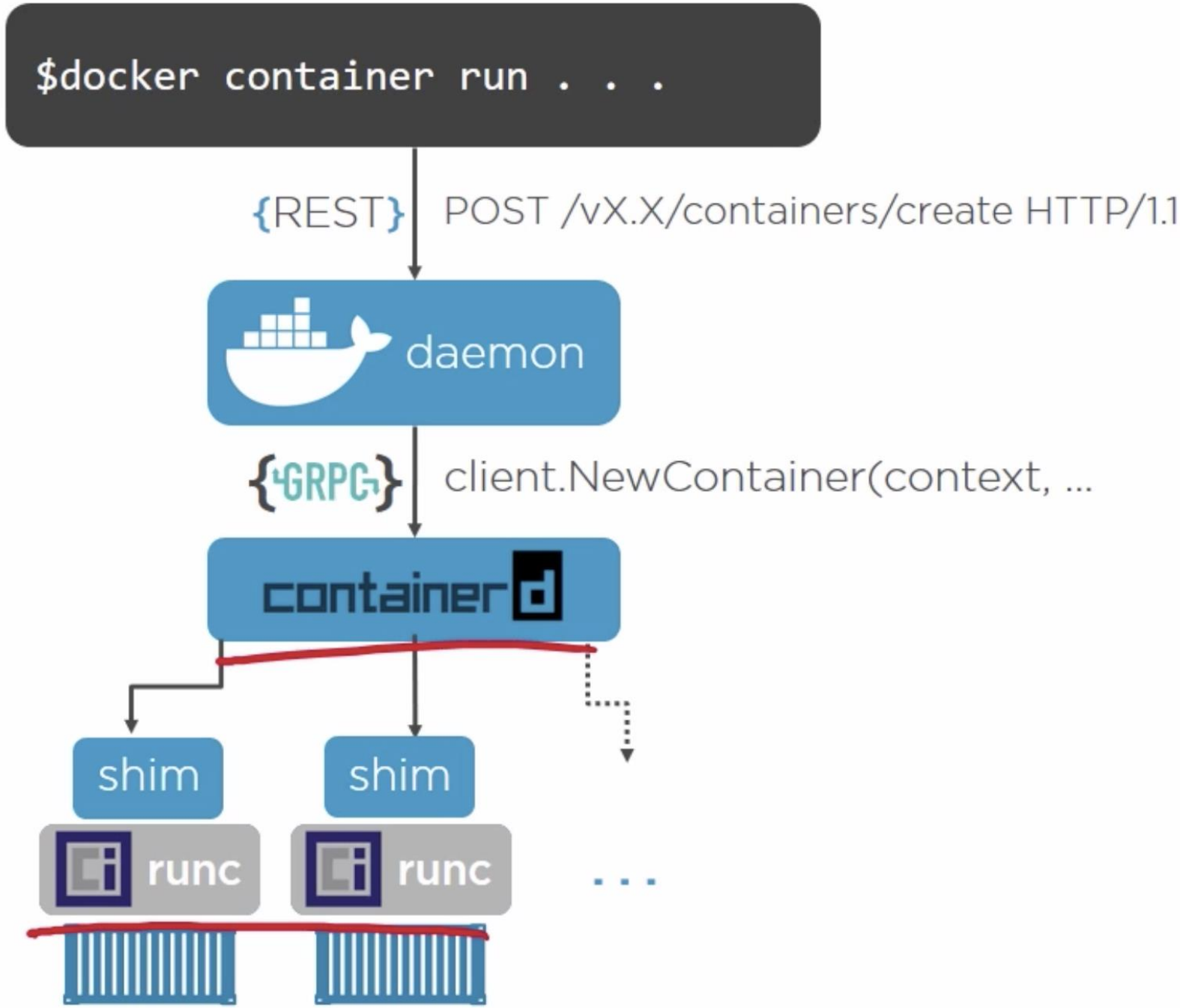
- Docker Registry is a storage component for Docker Images
- We can store the Images in either Public / Private repositories
- DockerHub is Docker's very own cloud repository

Why Use Docker Registries?

- Control where your images are being stored
- Integrate image storage with your in-house development workflow

Docker Architecture




$docker container run . . .
{REST}
POST /vX.X/containers/create HTTP/1.1
daemon
{GRPC}
client.NewContainer(context, ...
containerd
shim
shim
runc
runc
. . .

Most Used Docker Commands

```
docker --version
docker --help
docker pull
docker run
docker build
docker login
docker push
```

```
docker ps
docker images
docker stop
docker kill
docker rm
docker rmi
docker exec
```

```
docker commit
docker import
docker export
docker container
docker compose
docker swarm
docker service
```

Basic Docker Commands

docker pull

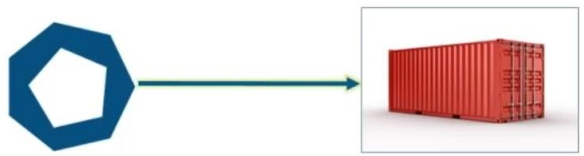
```
$ docker pull ubuntu
```

This command pulls a new Docker image from the Docker Hub

docker run

```
$ docker run ubuntu
```

This command executes a Docker image on your local repo & creates a running Container out of it



docker images

```
$ docker images
```

This command lists down all the images in your local repo

docker build

```
$ docker build –t MyUbuntuImage .
```

This command is used to compile the Dockerfile, for building custom Docker images based on the

docker container

This command is used to perform various operations on the container. Refer to www.docs.docker.com *for more info.*

```
$ docker container logs
```

```
$ docker container kill
```

```
$ docker container rm
```

```
$ docker container run
```

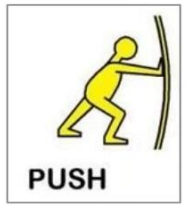
```
$ docker container start
```

And so on..

docker push

```
$ docker push vardhanns/MyUbuntuImage
```

This command pushes a Docker image on your local repo to the Docker Hub



docker stop

```
$ docker stop fe6e370a1c9c
```

This command shuts down the container whose Container ID is specified in arguments. Container is shut down gracefully by waiting for other dependencies to shut

docker ps

```
$ docker ps
```

```
$ docker ps -a
```

This command lists all the running containers in the host
If '–a' flag is specified, shutdown containers are also displayed

docker kill

```
$ docker kill fe6e370a1c9c
```

This command kills the container by stopping its execution immediately. Its similar to force kill

- #docker pull centos
- #docker container run -itd --name linuxcon100 centos
- #docker container exec -it linuxcon100 bash

- #docker container run -itd --name linuxcon100 -p 8007:80 nginx
- if we give -P , it will assign random port
- #docker container run -itd --name linuxcon100 -P nginx

- to inspect
- #docker inspect image nginx
- #docker inspect container containerName

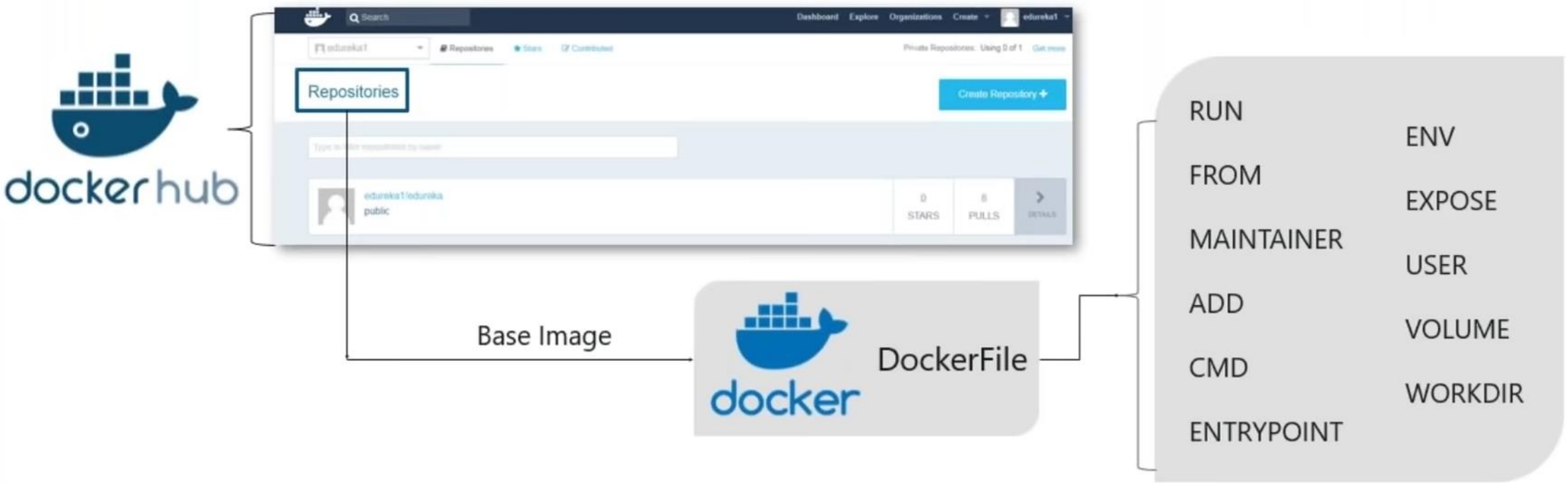
- #docker container stop cont.ID
- #docker container rm --force cont.ID
- if we give force the following incident will happen
- container kill-container die-container network disconnect-container destroy

- #docker container start
- #docker info

- docker location /var/lib/docker

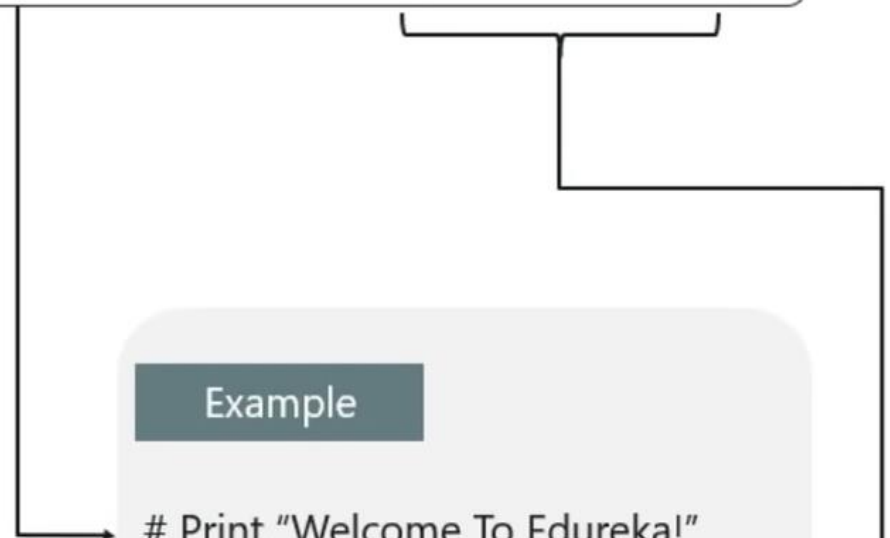
How To Build Docker Images? – Using DockerFile

Dockerfile is a script, composed of various commands (instructions) and arguments listed successively to automatically perform actions on a base image in order to create (or form) a new one



DockerFile Syntax

Dockerfile syntax consists of two kind of main line blocks: **comments** and **commands** + **arguments**



Syntax

```
# Line blocks used for commenting

command argument argument1...
```

Example

```
# Print "Welcome To Edureka!"

RUN echo "Welcome To Edureka!"
```

Apache2 Web Server. Apache is the most commonly used Web server on Linux systems. Web servers are used to serve Web pages requested by client computers

```
FROM ubuntu:14.04

MAINTAINER edureka

RUN apt-get update
RUN apt-get install -y nginx
ADD index.html /usr/share/nginx/html/index.html

ENTRYPOINT ["/usr/sbin/nginx", "-g", "daemon off;"]
EXPOSE 80
```

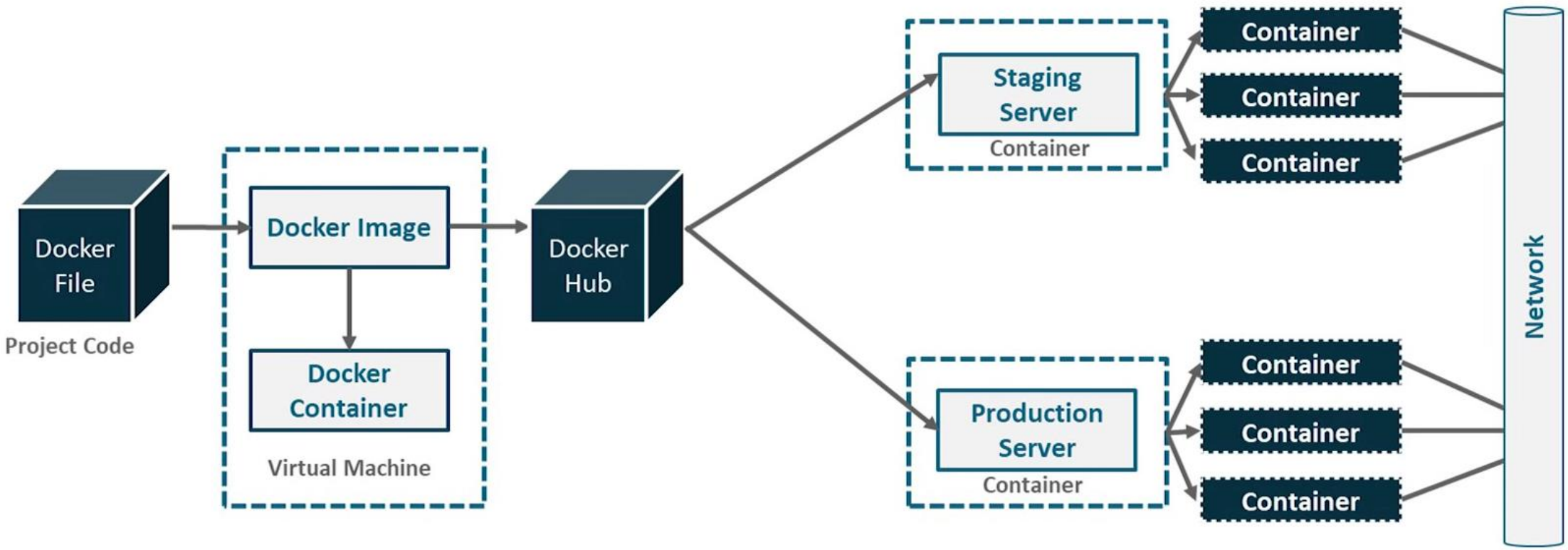
#vi Dockerfile

- FROM openjdk:8
- LABEL version="1.0."
- RUN apt-get update -Y
- RUN apt-get install git -Y
- RUN git clone https://github.com/spring-projects
- WORKDIR /spring-petclinic
- RUN ./mvnw package
- EXPOSE 8080
- CMD ["java","-jar, "target/*.jar"]

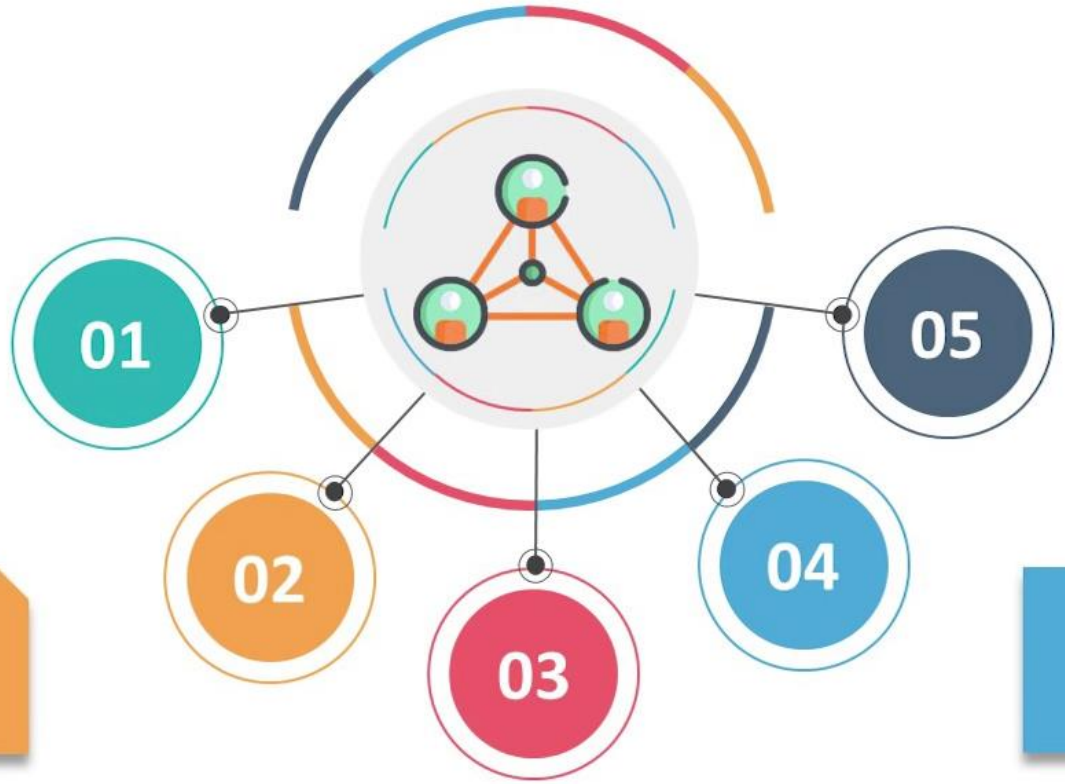
Docker Networking

Docker Networking

Docker containers and services do not need to be aware that they are deployed on Docker, or whether their peers are also Docker workloads or not, and this introduces the concept of Docker Networking.



Network Drivers



Bridge

Host

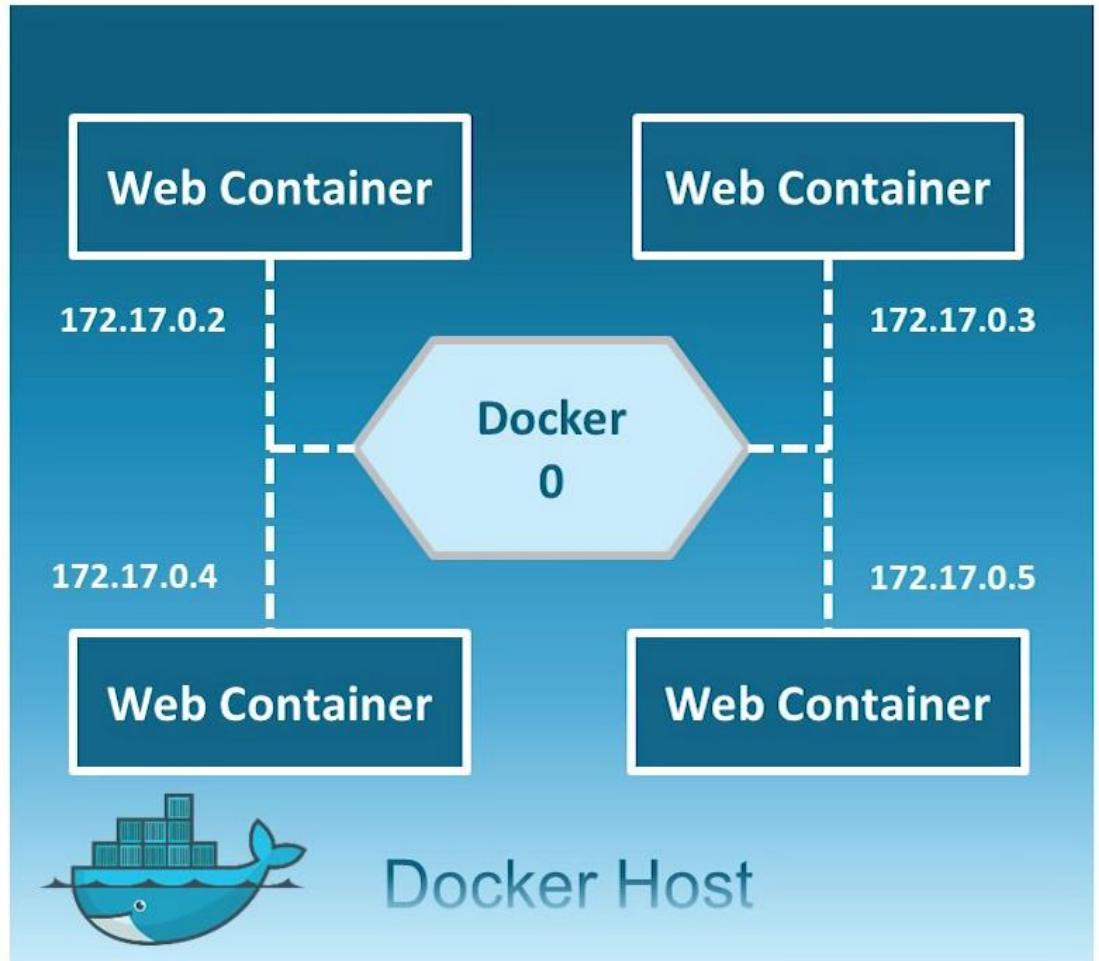
None

Overlay

Macvlan

Network Drivers: Bridge

The default network driver. If you don't specify a driver, this is the type of network you are creating. Bridge networks are usually used when your applications run in standalone containers that need to communicate.

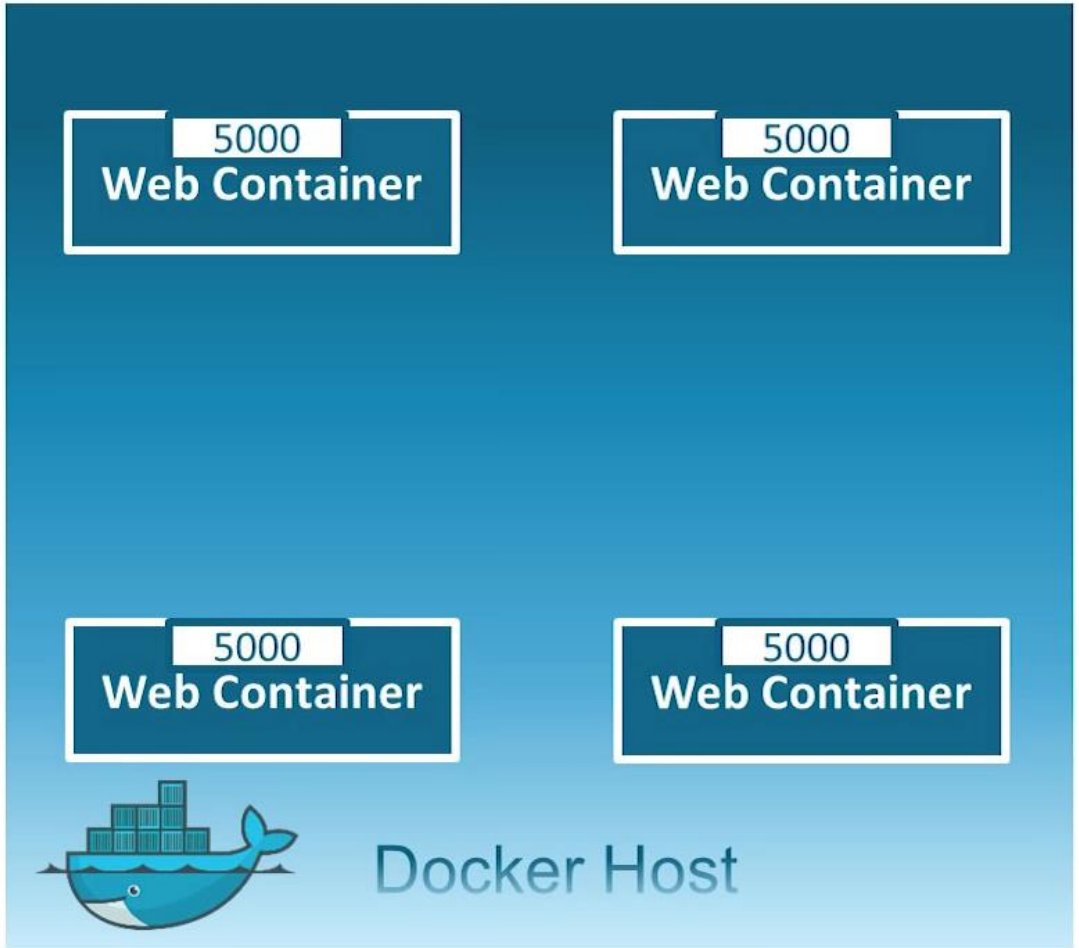


- BRIDGE: bridge n/w is used to multiple containers to communicate on the same docker host
- but OVERLAY n/w are best when you need containers running on different docker hosts.
- #docker network ls
- #docker network inspect networkid

- Network create
- #docker network create --driver bridge mynetwork-brg02 --subnet 10.0.0.0/16 --gateway 10.0.0.1
- create a container with our network bridge
- #docker container run -itd --name linux03 --network mynetwork-brg02 alpine
- connect 2 different network containers
- #docker network connect mynetwork-brg02 linux02

Network Drivers: Host

For standalone containers, removes network isolation between the container and the Docker host, and uses the host's networking directly.



- HOST: Linux base host network namespace will inherit to the container like IP
- HOST n/w are best when the n/w stack should not be isolated from the docker host
- but you want other aspects of the container to be isolated.
- NONE: disable all the network (it will check only itself -loopback)
- #docker container run -it --name noipcon --network none ubuntu

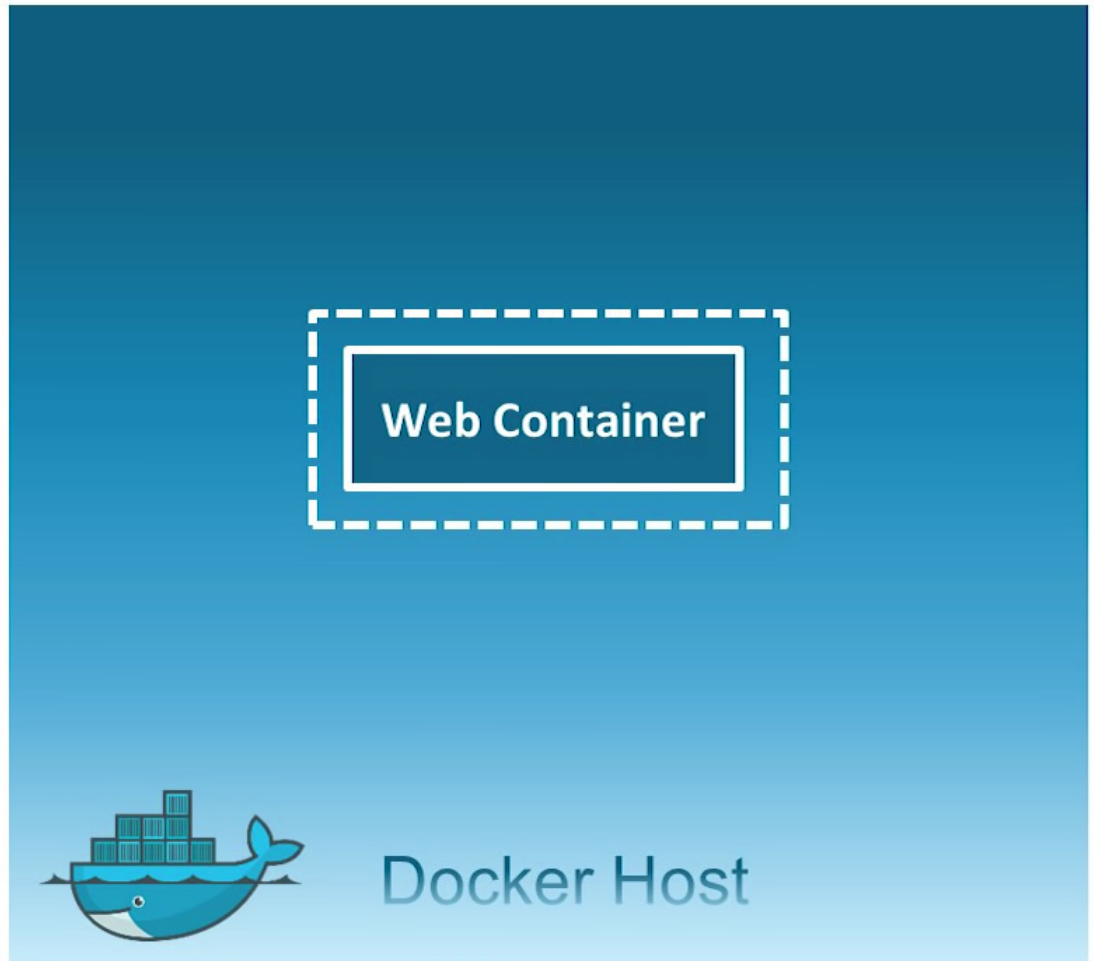
- overlay network : & macvlan network
- OVERLAY n/w are best when you need containers running on different docker hosts.

NOTE: we can create many nos of bridge network driver but there is only one "HOST"& "NULL" network driver.

Network Drivers: None

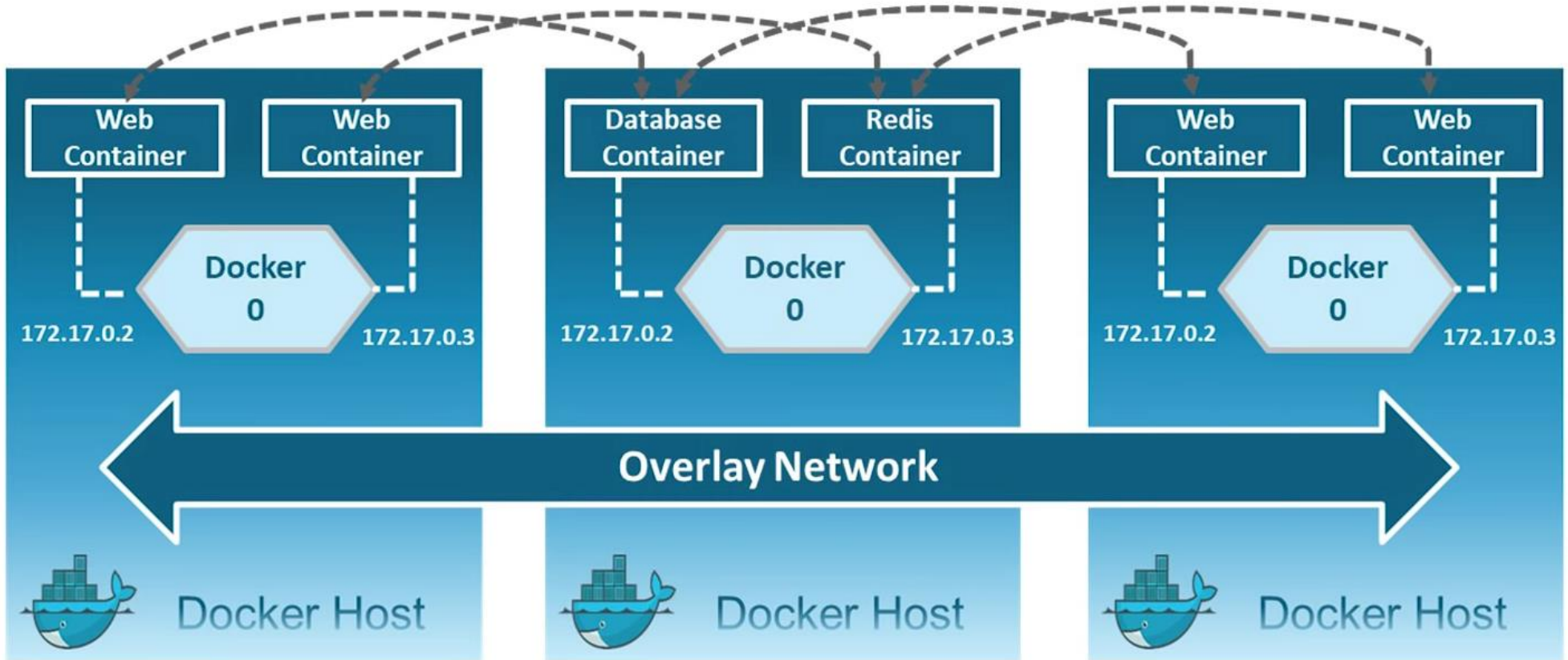
Disables all networking for containers.

Usually used in conjunction with a custom network driver.



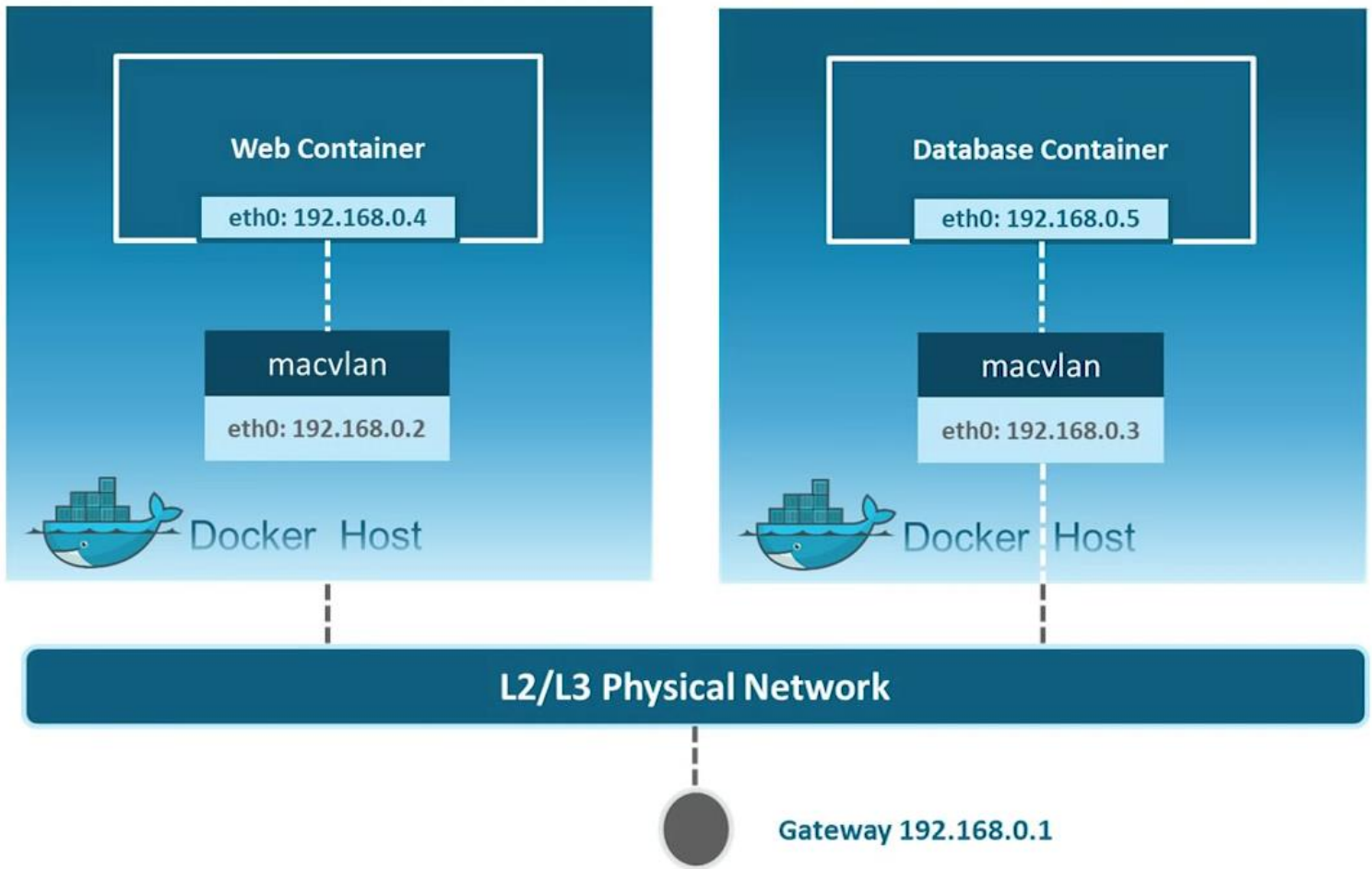
Network Drivers: Overlay

Overlay networks connect multiple Docker daemons together and enable swarm services to communicate with each other daemons.



Network Drivers: Macvlan

Macvlan networks allow you to assign a MAC address to a container, making it appear as a physical device on your network. The Docker daemon routes traffic to containers by their MAC addresses.



https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/containerd

https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/containerd#runhcs

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers

> docker volume //get information
> docker volume create
> docker volume ls
> docker volume inspect
> docker volume rm
> docker volume prune

Use of Volumes
==========
Decoupling container from storage
Share volume (storage/data) among different containers
Attach volume to container
On deleting container volume does not delete

- In container host we have to create one folder for volume

  dockerhost900#mkdir -p /test/data
- #docker container run -itd --name mycon03 - /test/data:/test/containerdata ubuntu
- We may give volume name also & map the container to that volume (default path is /var/lib/docker/volume)
- dockerhost900#docker volume create myvolumetest
- #docker container run -itd --name mycon06 -v myvolumetest:/test/mydata ubuntu
- BIND VOLUME - it will use or mount the base Linux mounted volume like /etc/nsswitch.conf
- dockerhost900#docker container run -itd --name testcon100 --mount type=bind /etc/nsswitch.conf:/etc/nsswitch.conf imagename

2 July 2018 at 2:30 PM

Docker Basic Commands

Step by Step for Beginners

Basic

> docker version
> docker -v
> docker info
> docker --help
> docker login

Images

> docker images
> docker pull
> docker rmi

Containers

> docker ps
> docker run
> docker start
> docker stop

System

> docker stats
> docker system df
> docker system prune

Developer

Dockerfile

Docker image

Docker Container

Docker Hub

Pull image

Docker image

Docker Container

Staging Environment

Pull image

Docker image

Docker Container

Test Environment

This resolves the issue of app working on one platform and not on other

- ETCD -- is a Database which has a key-value format about container->NodeLabel
- Scheduler -- identifies the right node to schedule a container(based on container requirement,worker node capacity,any other policies,taints&toleration,etc)
- Controller -- to take care of different areas,-Node ctl take care of Nodes

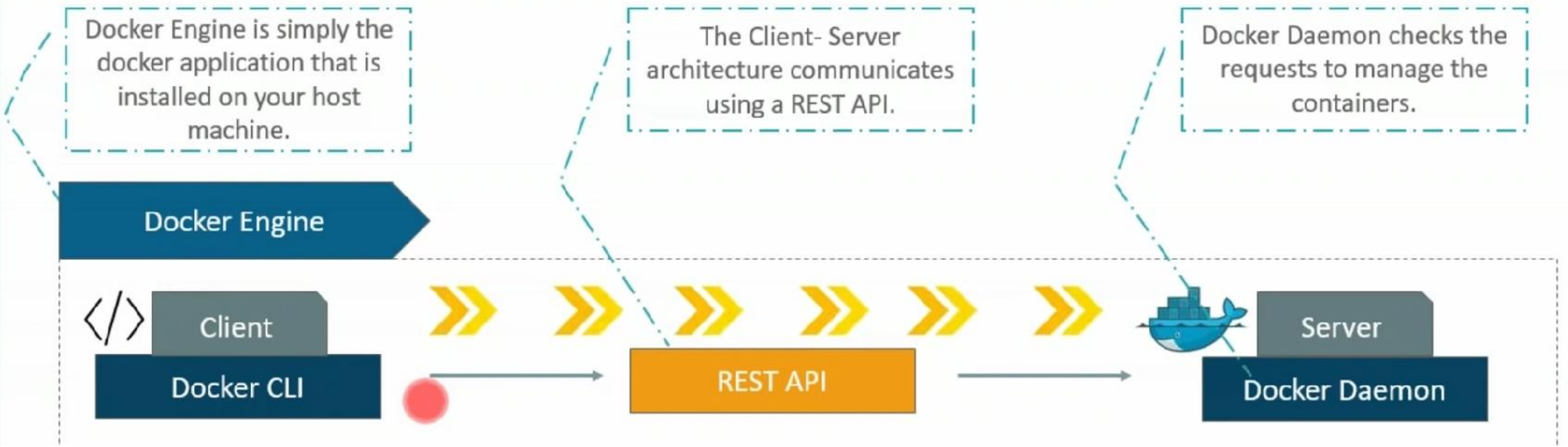
  -Replication controller take care of desired container running all times,
- KUBE-APIserver -- is PRIMARY management of orchestrating all operation within the cluster.
- KUBELET -- is captain of the ship.kubelet is an engine runs on all the nodes& its listens an instructions from the API server.

  like deploy or destroy--,. API fetches periodical status report from KUBELET.
- KUBEPROXY -- ensures neccessary rules the worker nodes allows the container to communicate which is running on worker nodes

  in simple words kubeproxy enabling communication between services within the cluster.

How Does Docker Work?

Docker Engine uses a Client Server Architecture.



Docker Architecture

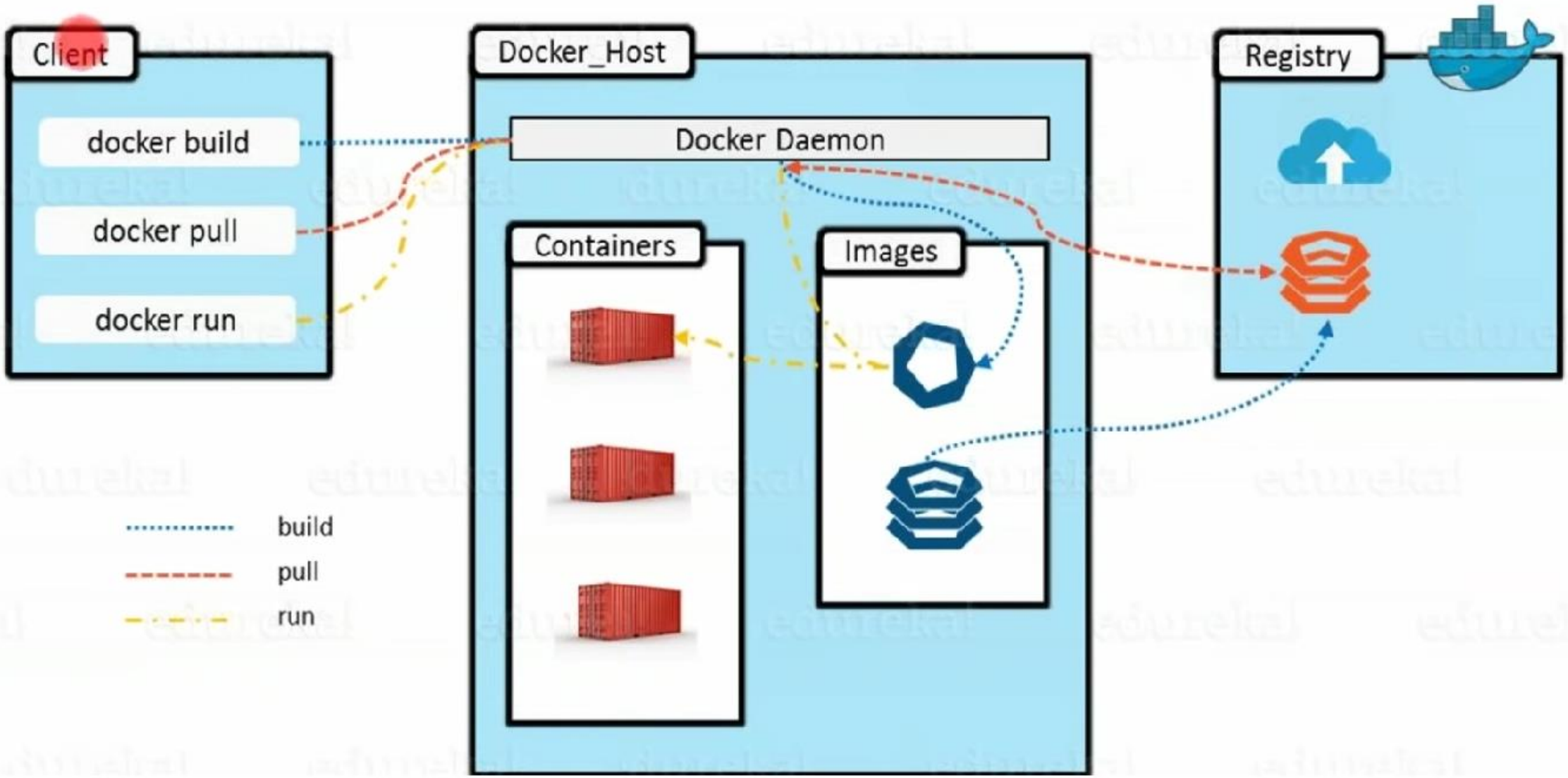


Image build through Dockerfile
[#mkdir test100
vi Dockerfile
FROM ubuntu:latest
LABEL webimage=v1.0
RUN apt-get update -y
RUN apt-get install tree -y && apt-get install git -y]

Then build
#docker image build -t webappimage:V1.0.0 .

- BRIDGE: bridge n/w is used to multiple containers to communicate on the same docker host
- but OVERLAY n/w are best when you need containers running on different docker hosts.
- #docker network ls
- #docker network inspect networkid

- Network create
- #docker network create --driver bridge mynetwork-brg02 --subnet 10.0.0.0/16 --gateway 10.0.0.1
- create a container with our network bridge
- #docker container run -itd --name linux03 --network mynetwork-brg02 alpine
- connect 2 different network containers
- #docker network connect mynetwork-brg02 linux02

- HOST: Linux base host network namespace will inherit to the container like IP
- HOST n/w are best when the n/w stack should not be isolated from the docker host
- but you want other aspects of the container to be isolated.
- NONE: disable all the network (it will check only itself -loopback)
- #docker container run -it --name noipcon --network none ubuntu

- overlay network : & macvlan network
- OVERLAY n/w are best when you need containers running on different docker hosts.

NOTE: we can create many nos of bridge network driver but there is only one "HOST"& "NULL" network driver.

3 July 2018 at 1:38 PM

Docker Volumes

Step by Step for Beginners

Today we will learn:

1. What are Volumes
2. How to create / list / delete volumes
3. How to attach volume to a container
4. How to share volume among containers
5. What are bind mounts

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers

```
> docker volume  //get information
> docker volume create
> docker volume ls
> docker volume inspect
> docker volume rm
> docker volume prune
```

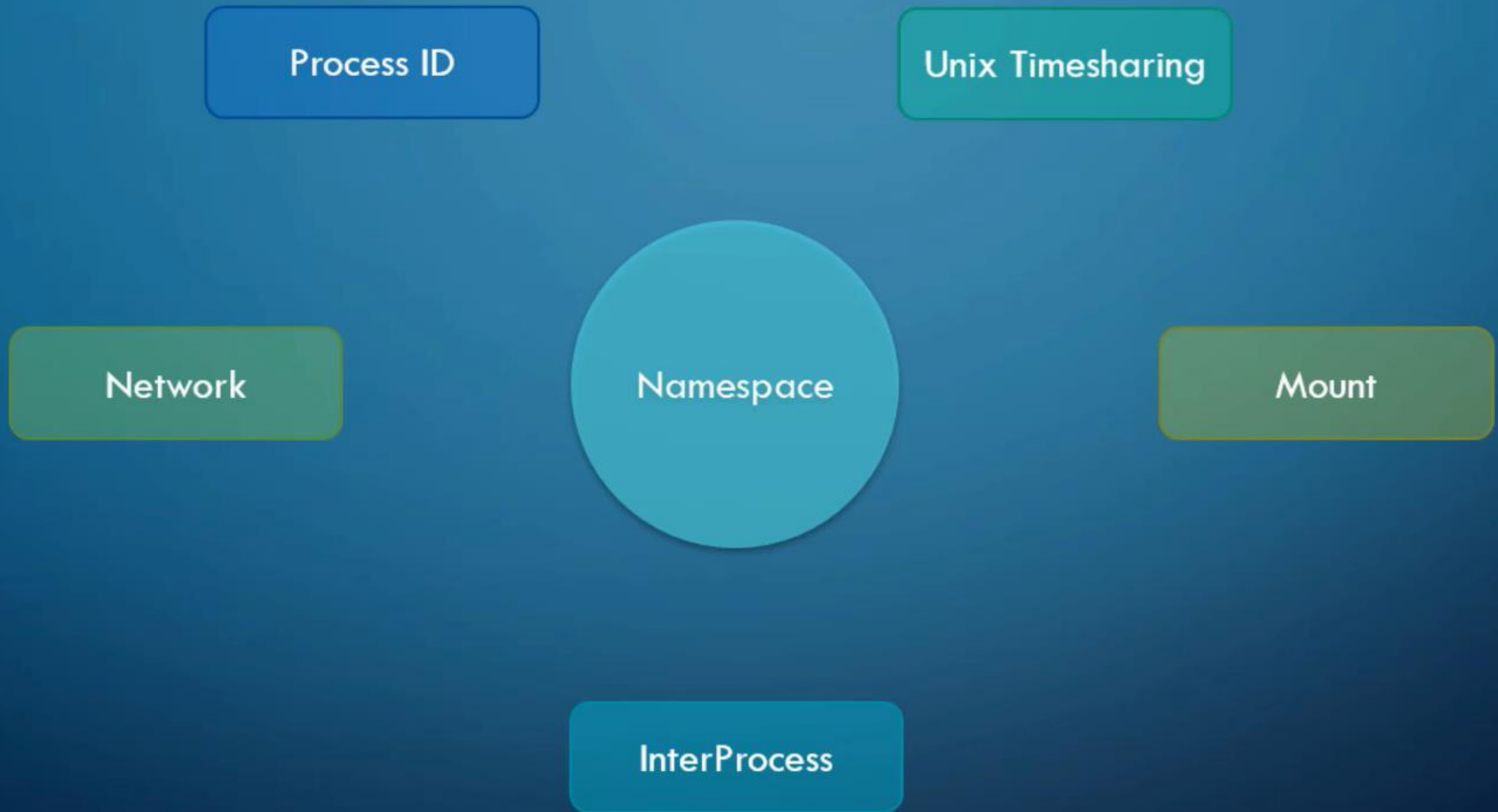
Volumes are the preferred mechanism for persisting data generated by and used by Docker containers

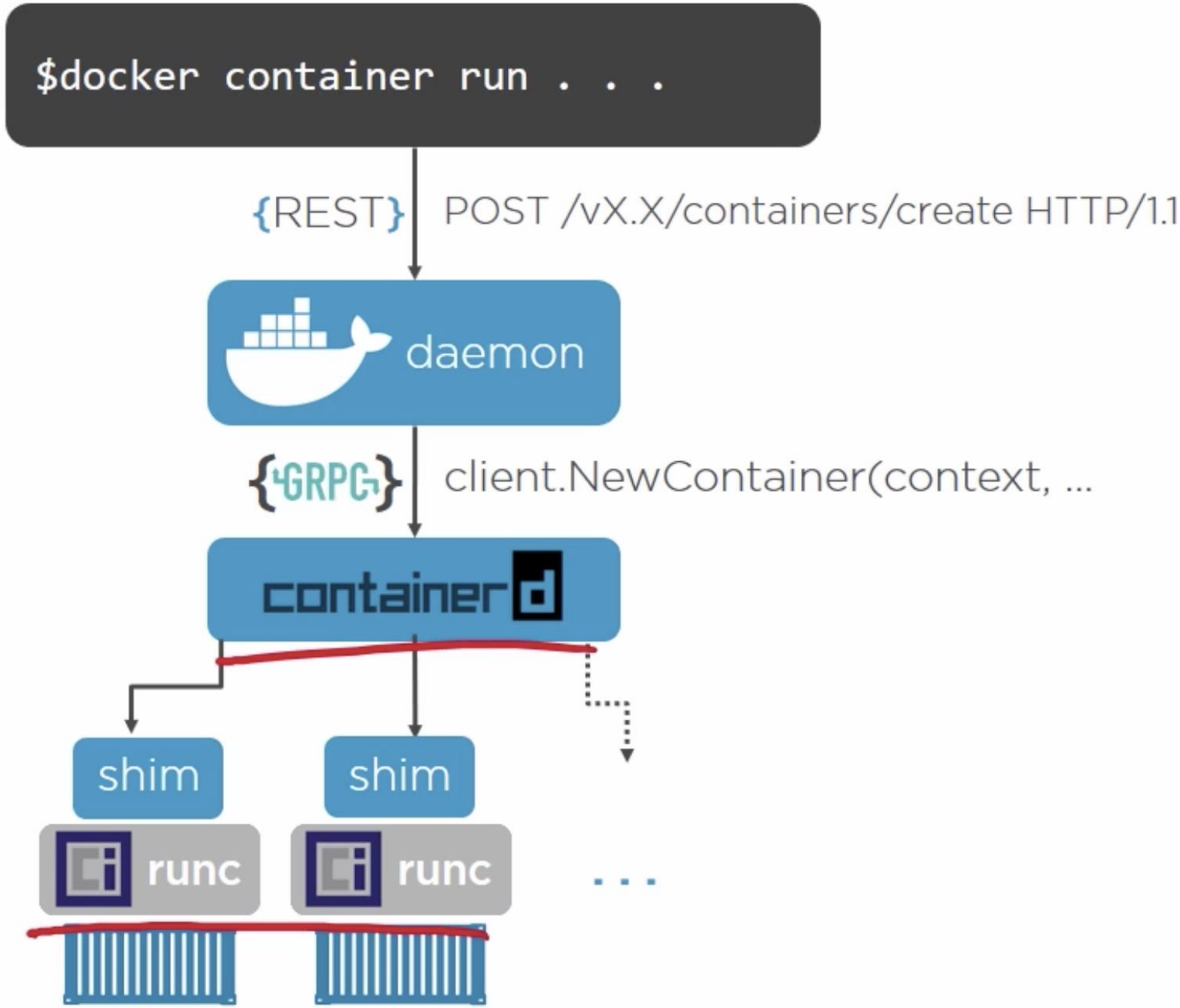
> docker volume //get information
> docker volume create
> docker volume ls
> docker volume inspect
> docker volume rm
> docker volume prune

Use of Volumes
==========
Decoupling container from storage
Share volume (storage/data) among different containers
Attache volume to container
On deleting container volume does not delete

```
raghav$ docker run --name MyJenkins1 -v myvol1:/var/jenkins_home -p 8080:8080 -p
 50000:50000 jenkins
```

CONTAINERIZATION




$docker container run . . .
{REST}
POST /vX.X/containers/create HTTP/1.1
daemon
{GRPC}
client.NewContainer(context, ...
containerd
shim
shim
runc
runc
. . .