Major Docker Components

Docker Engine

Images

Containers

Registries and Repositories

Docker Engine

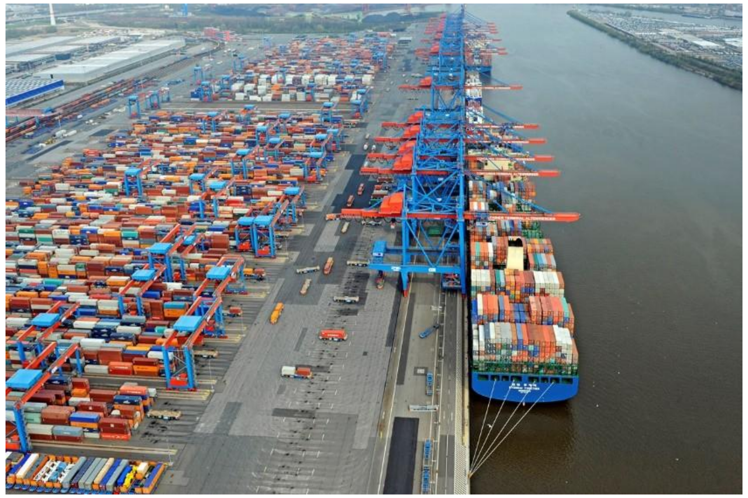
(Shipping Yard)

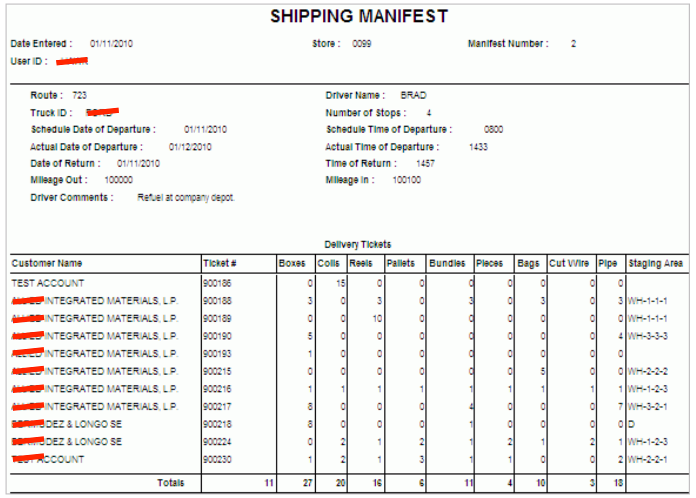
Docker Images

(Manifests)

Docker Containers

(Shipping Containers)





SHIPPING MANIFEST

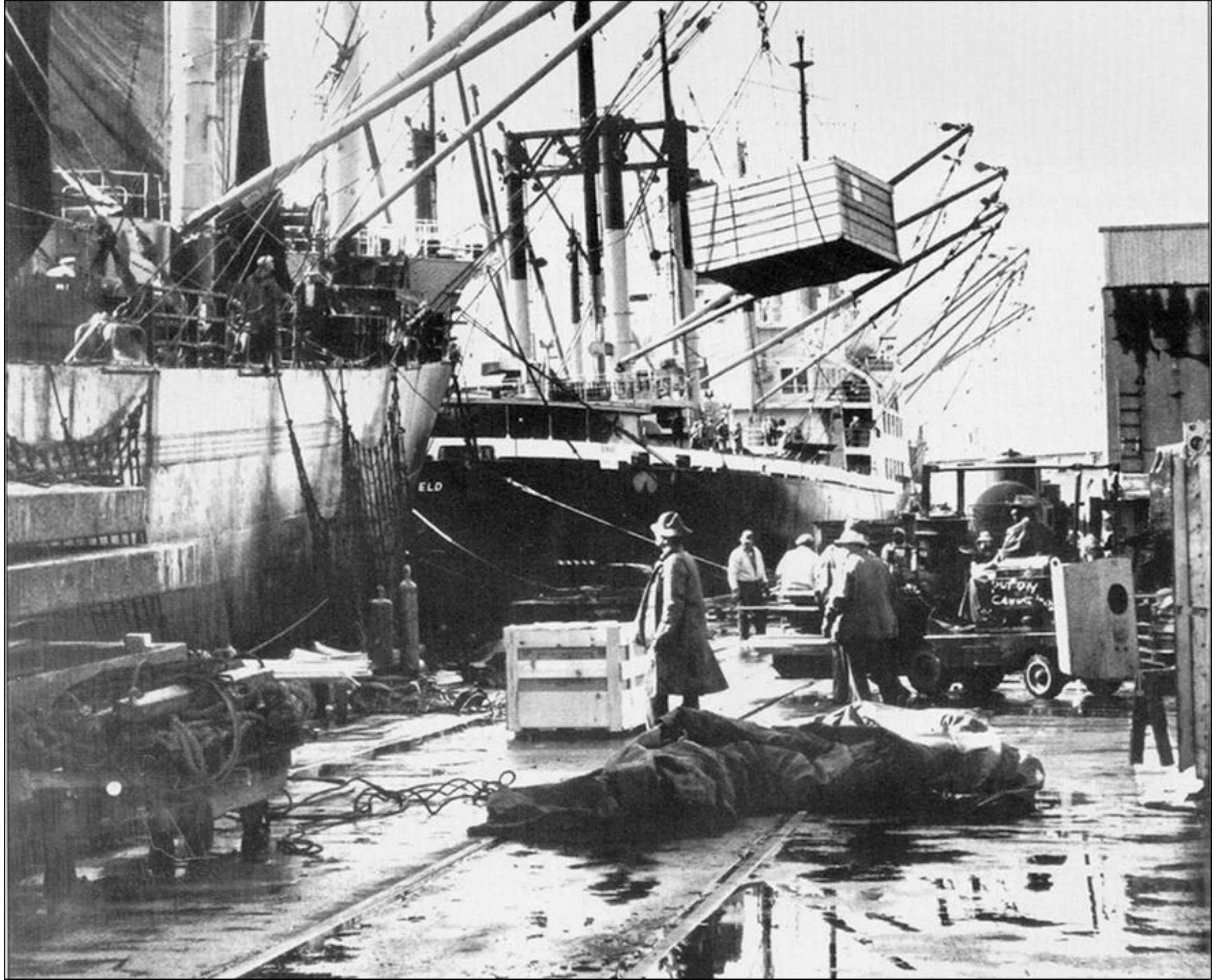
Date Entered : 01/11/2010 Store : 0099 Manifest Number : 2
User ID : [redacted]

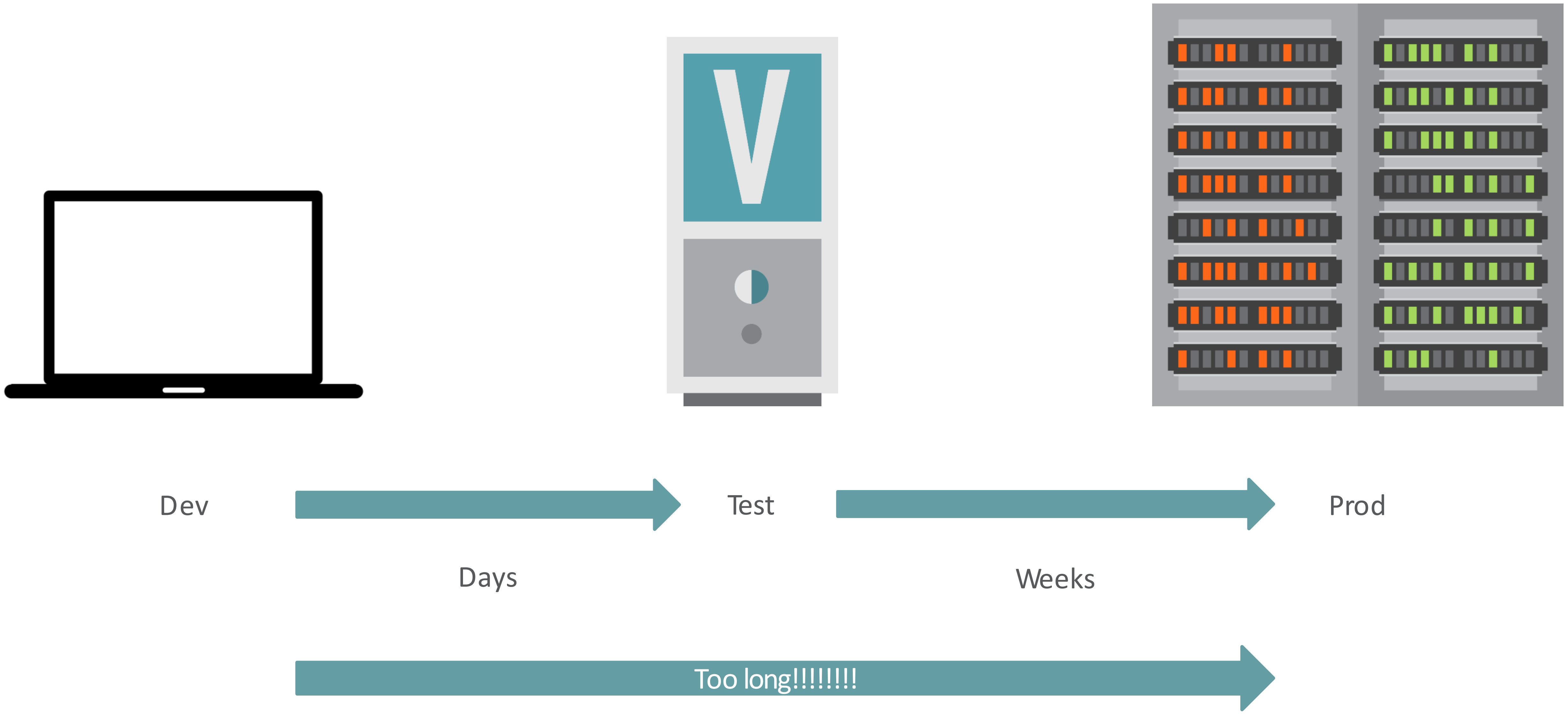
Route : 723 Driver Name : BRAD
Truck ID : [redacted] Number of Stops : 4
Schedule Date of Departure : 01/11/2010 Schedule Time of Departure : 0800
Actual Date of Departure : 01/12/2010 Actual Time of Departure : 1433
Date of Return : 01/11/2010 Time of Return : 1457
Mileage Out : 100000 Mileage In : 100100
Driver Comments : Refuel at company depot.

Delivery Tickets

Customer Name	Ticket #	Boxes	Coils	Reels	Pallets	Bundles	Pieces	Bags	Cut Wire	Pipe	Staging Area
TEST ACCOUNT	900186	0	15	0	0	0	0	0	0	0	
[redacted] INTEGRATED MATERIALS, L.P.	900188	3	0	3	0	3	0	3	0	3	WH-1-1-1
[redacted] INTEGRATED MATERIALS, L.P.	900189	0	0	10	0	0	0	0	0	0	WH-1-1-1
[redacted] INTEGRATED MATERIALS, L.P.	900190	5	0	0	0	0	0	0	0	4	WH-3-3-3
[redacted] INTEGRATED MATERIALS, L.P.	900193	1	0	0	0	0	0	0	0	0	
[redacted] INTEGRATED MATERIALS, L.P.	900215	0	0	0	0	0	0	5	0	0	WH-2-2-2
[redacted] INTEGRATED MATERIALS, L.P.	900216	1	1	1	1	1	1	1	1	1	WH-1-2-3
[redacted] INTEGRATED MATERIALS, L.P.	900217	8	0	0	0	4	0	0	0	7	WH-3-2-1
[redacted] DEZ & LONGO SE	900218	8	0	0	0	1	0	0	0	0	D
[redacted] DEZ & LONGO SE	900224	0	2	1	2	1	2	1	2	1	WH-1-2-3
[redacted] ACCOUNT	900230	1	2	1	3	1	1	0	0	2	WH-2-2-1
Totals	11	27	20	16	6	11	4	10	3	18	

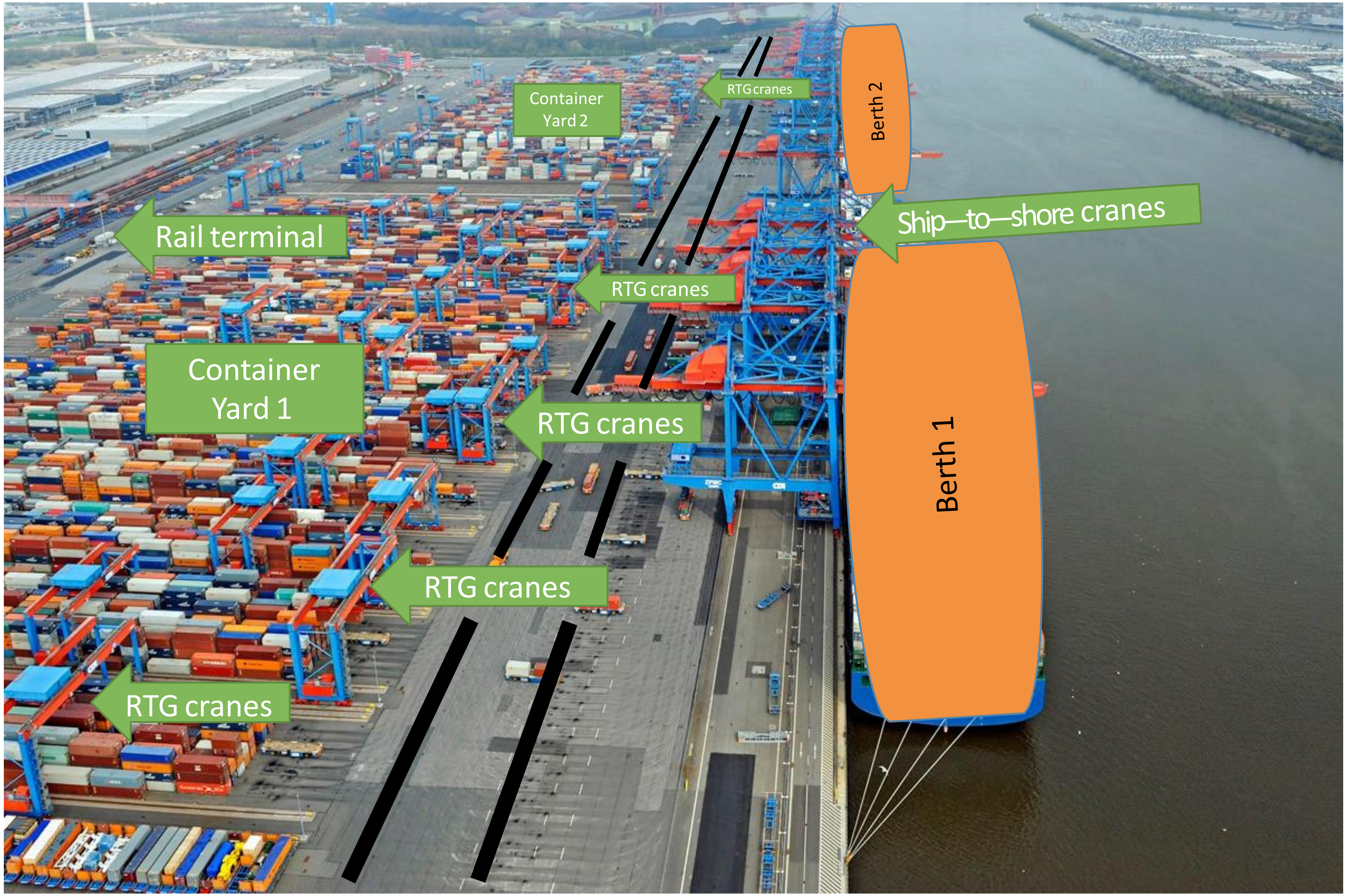
Docker Engine a.k.a. Docker Daemon, or Docker Runtime…..

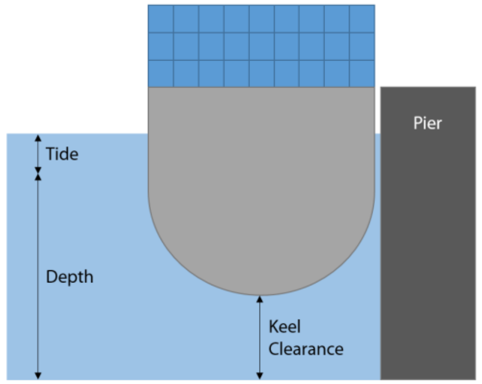

ELD

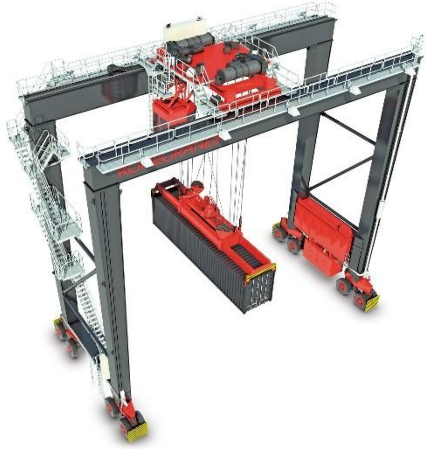

Dev
Test
Prod
Days
Weeks
Too long!!!!!!!!

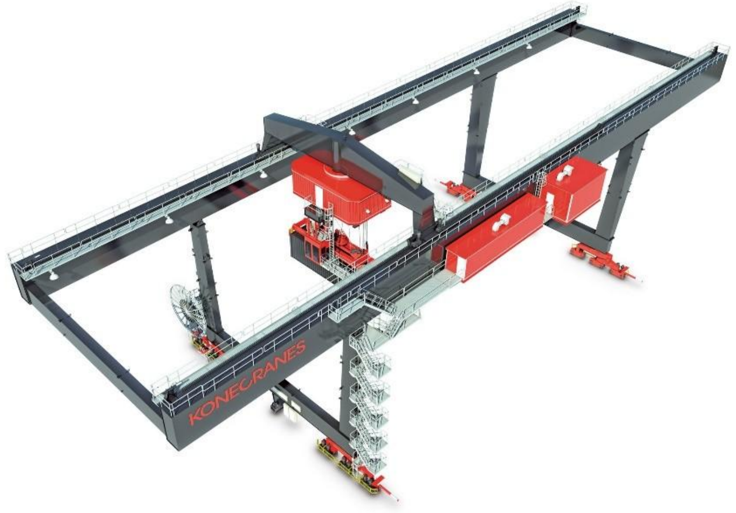
docker

Docker Engine


Container Yard 2
RTG cranes
Berth 2
Ship–to–shore cranes
Rail terminal
RTG cranes
Container Yard 1
RTG cranes
Berth 1
RTG cranes
RTG cranes

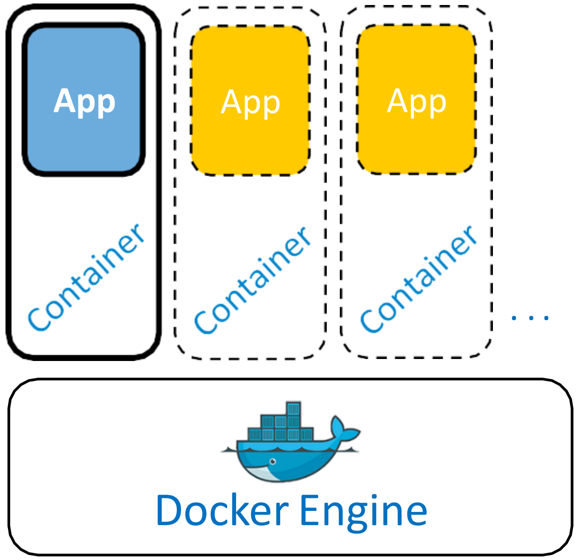


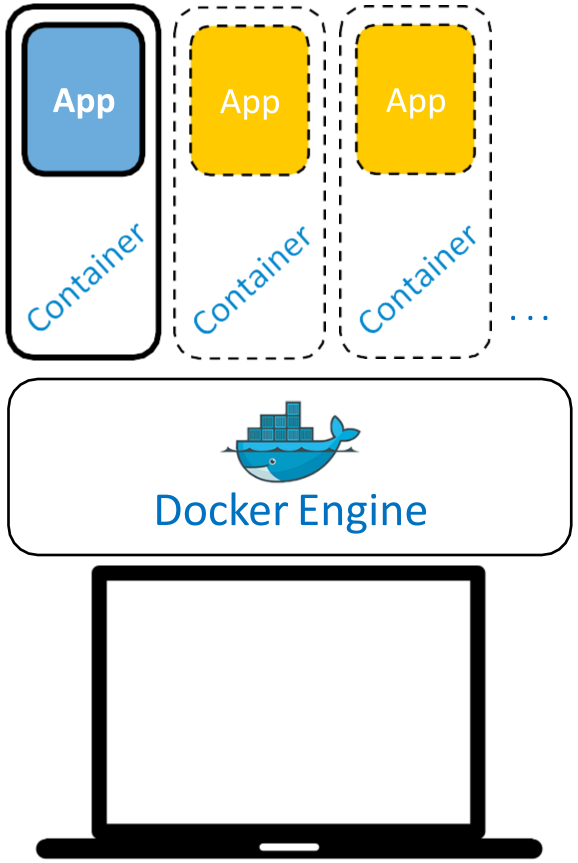


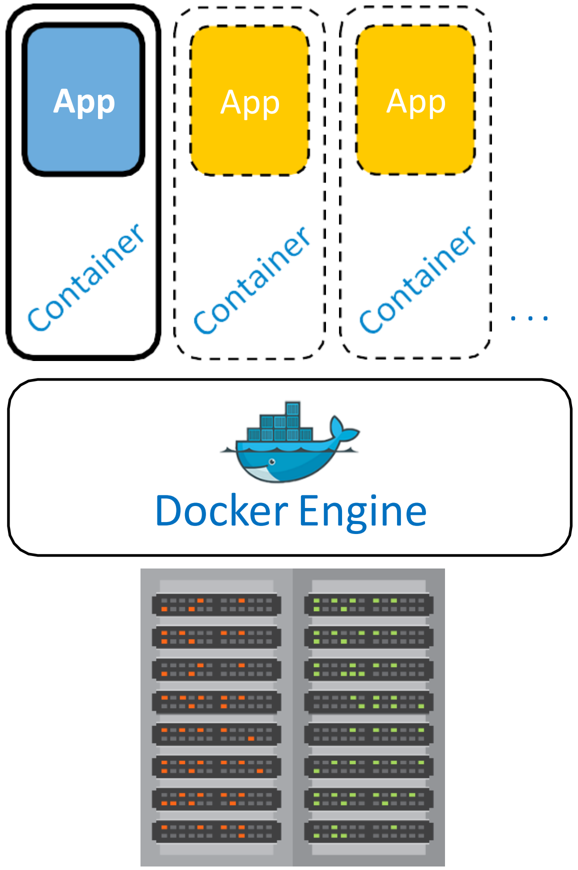


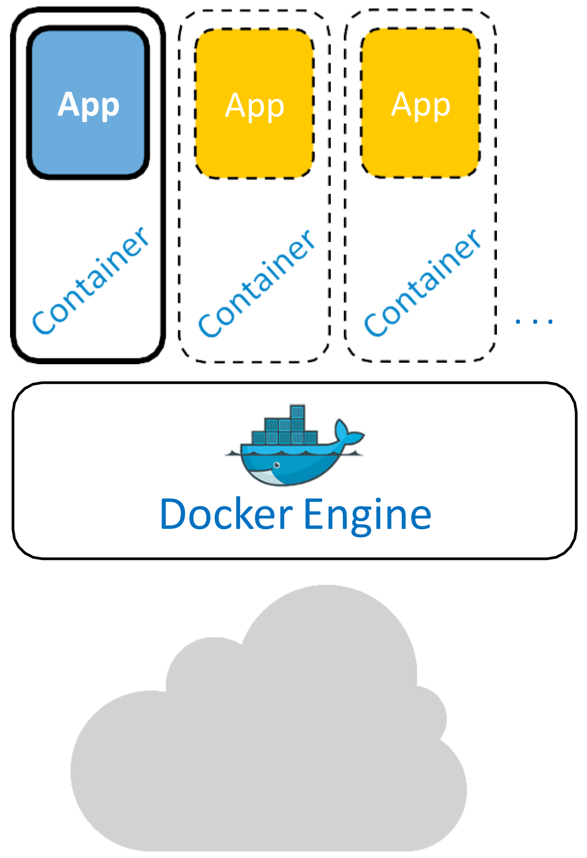
Equipment images from www.konecranes.com and www.terex.com

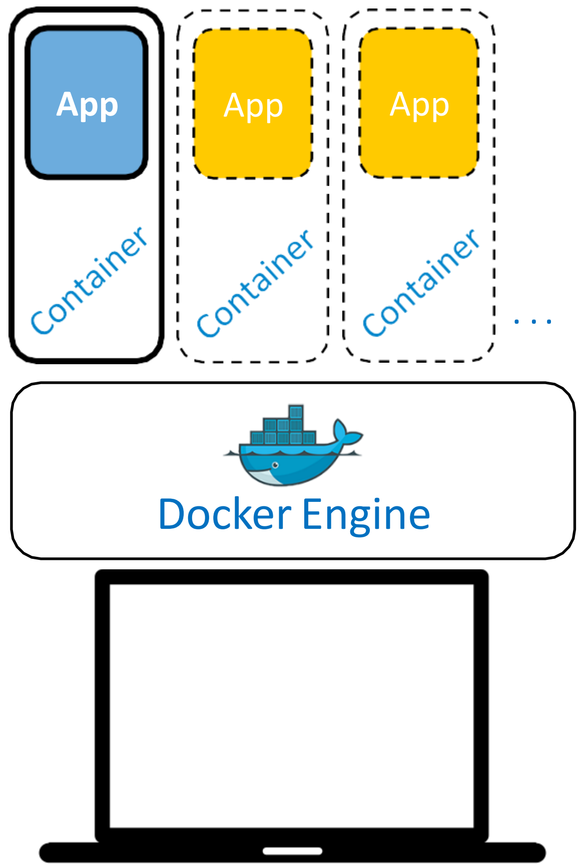
Docker Engine

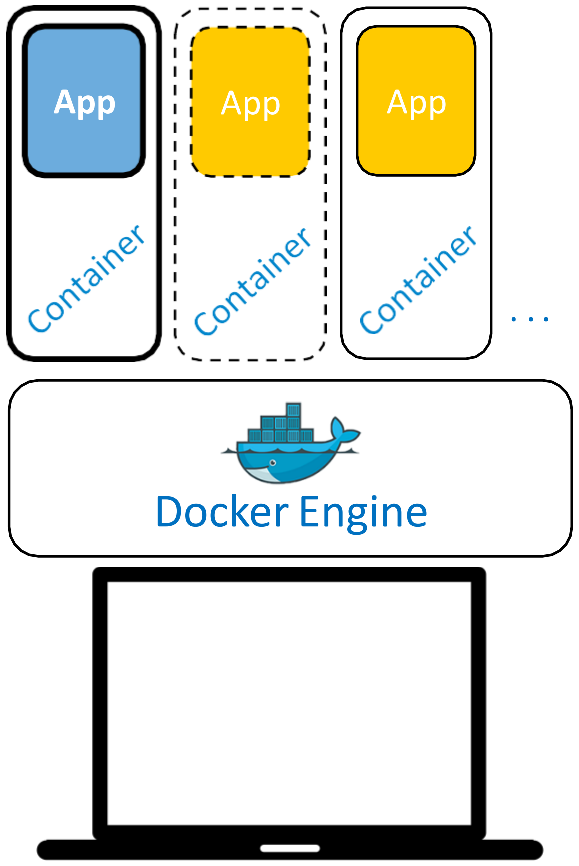

App
Container
App
Container
App
Container
. . .
Docker Engine

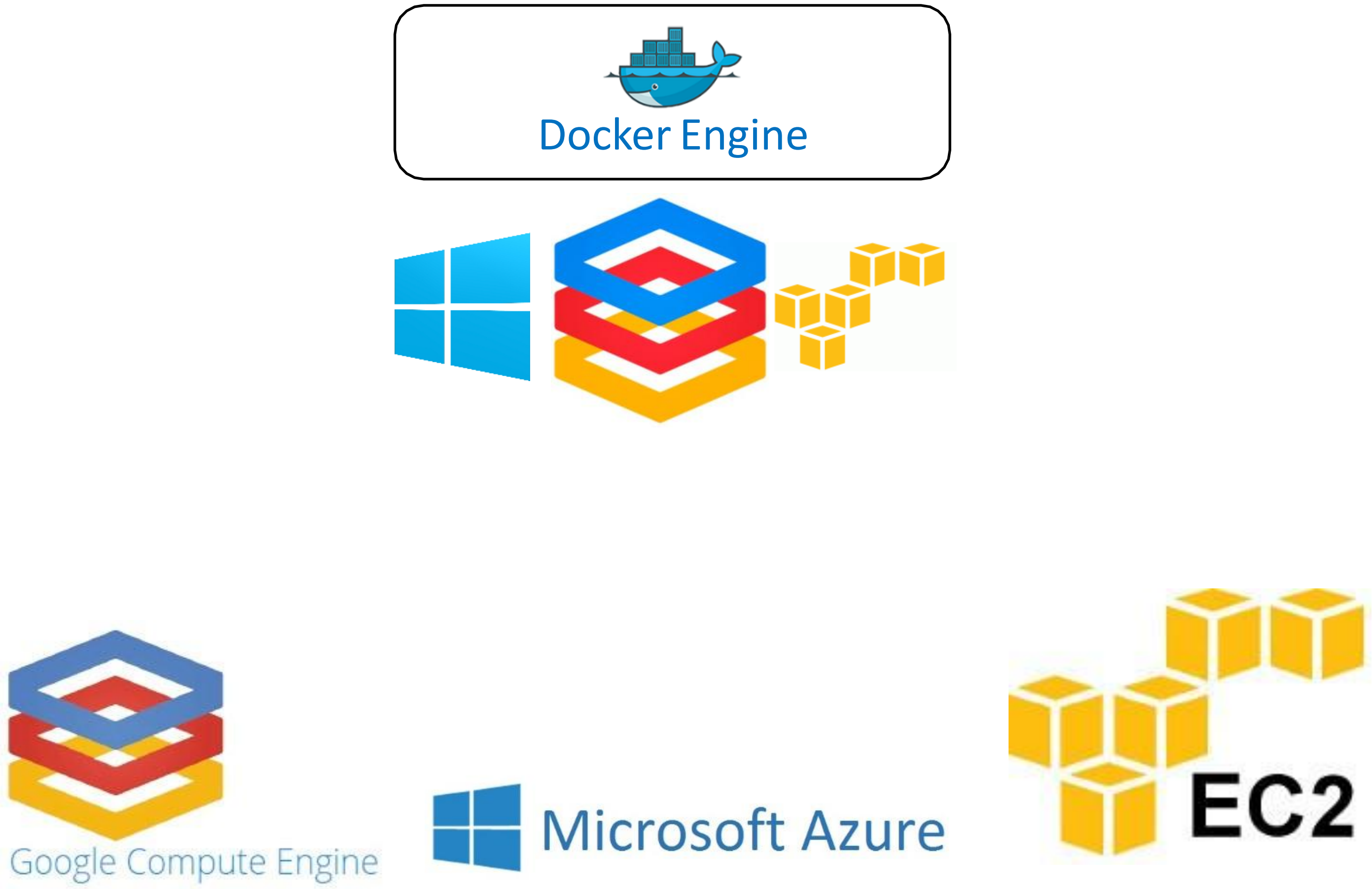

App
App
App
Container
Container
Container
. . .
Docker Engine


App
App
App
Container
Container
Container
. . .
Docker Engine


App
App
App
Container
Container
Container
. . .
Docker Engine


App
App
App
Container
Container
Container
. . .
Docker Engine

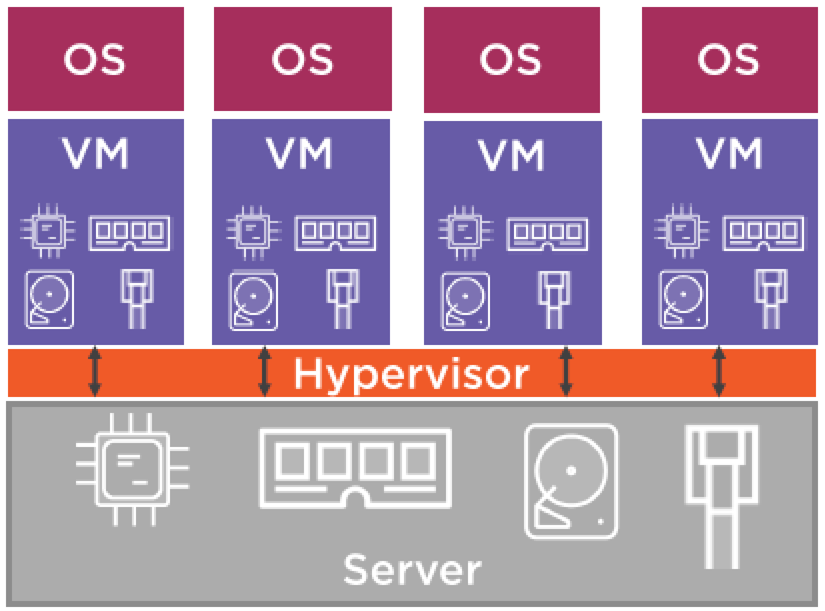

App
App
App
Container
Container
Container
. . .
Docker Engine

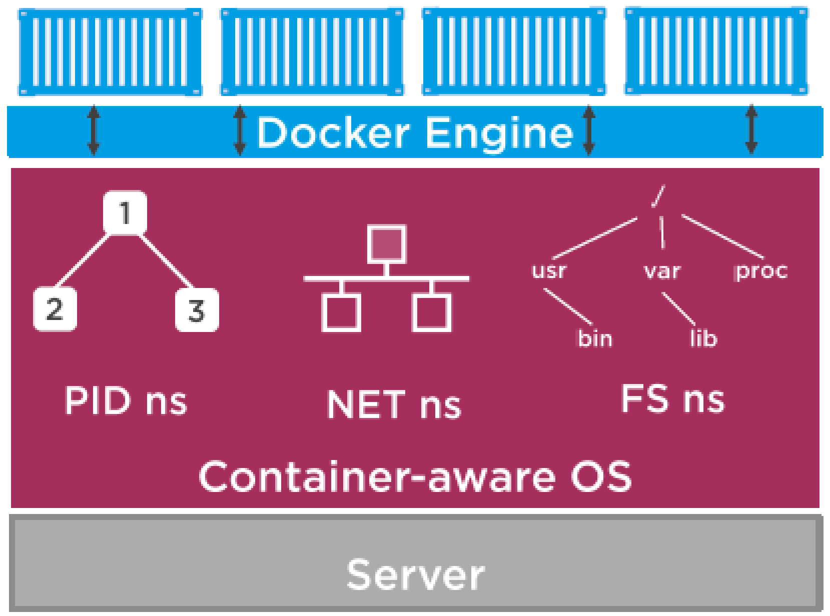

Docker Engine
Google Compute Engine
Microsoft Azure
EC2

Docker Images

VM Model

Container Model





-

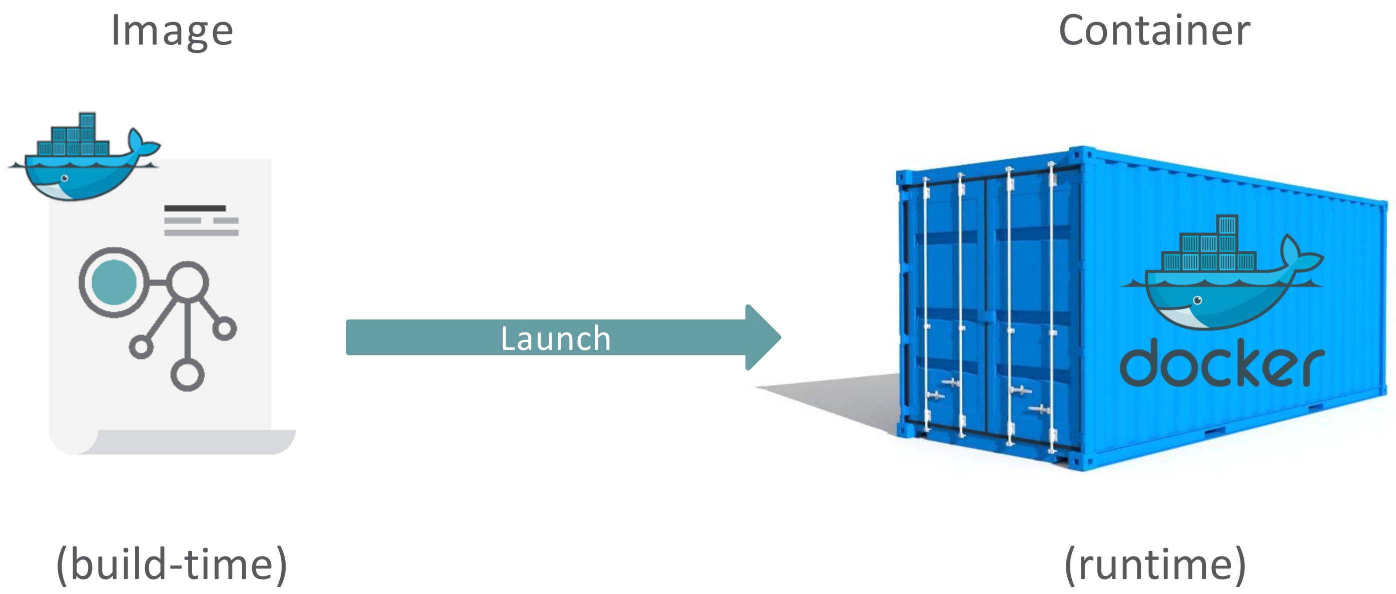


Image
Container
Launch
docker
(build-time)
(runtime)

Images vs Container

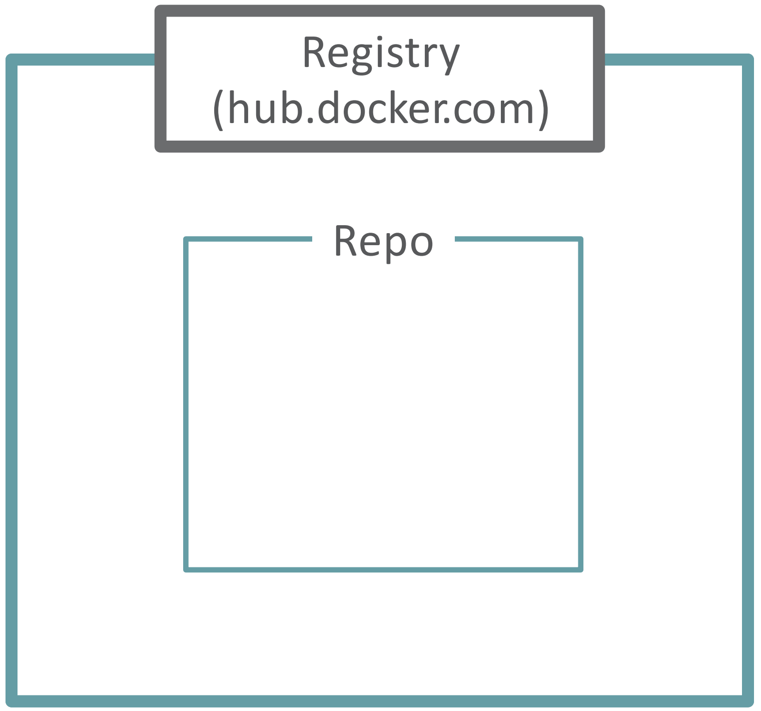
An instance of an image is called container. If you start this image, you have a running container of this image. You can have many running containers of the same image. You can see all your images with docker images whereas you can see your running containers with docker ps (and you can see all containers with docker ps -a).

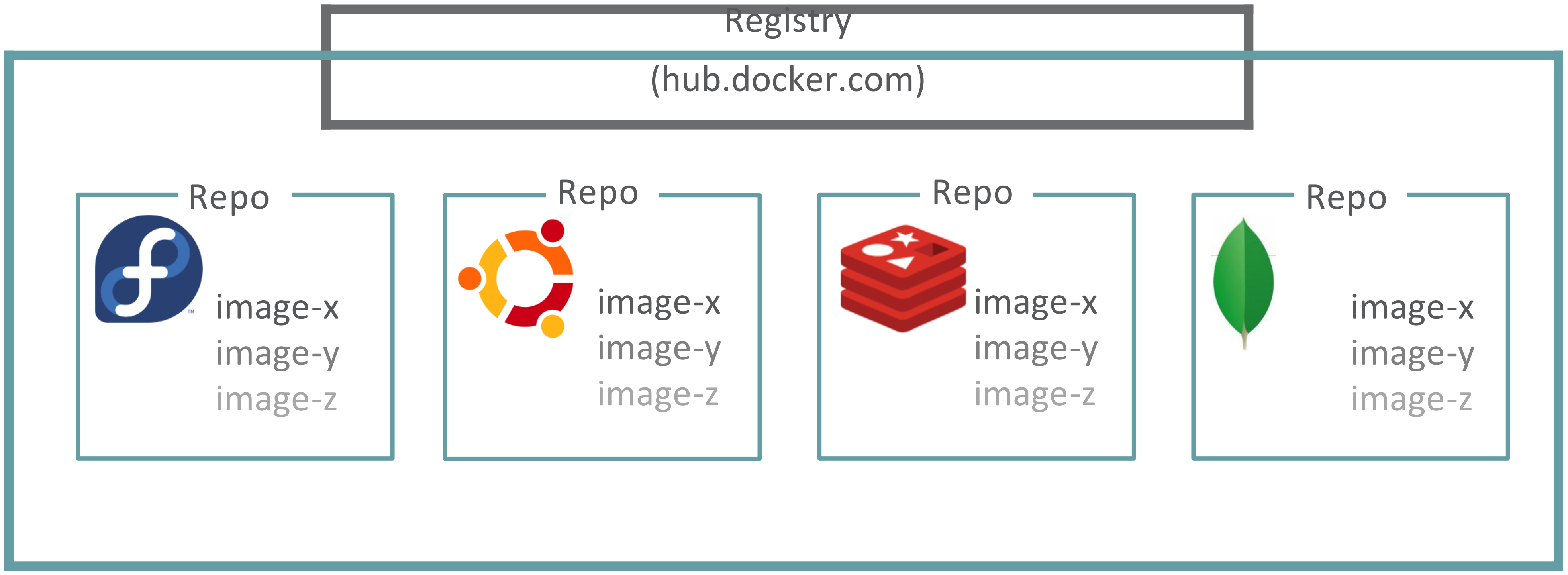
So a running image is a container.

Reference
http://stackoverflow.com/questions/23735149/docker-image-vs-container

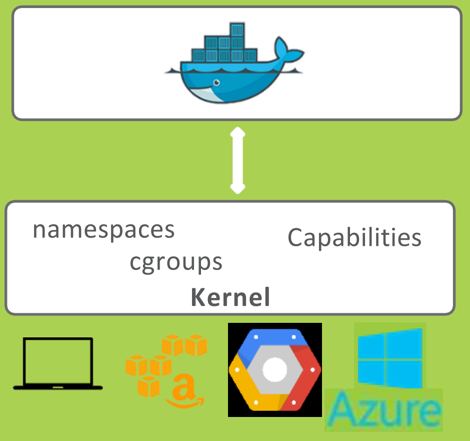
Registries and Repositories

A Quick Look


Registry
(hub.docker.com)
Repo

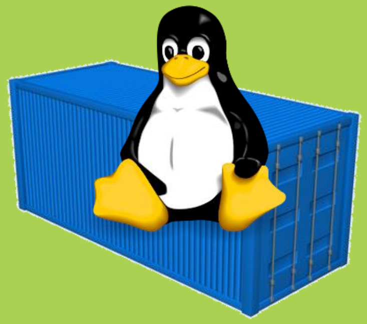

Registry
(hub.docker.com)
Repo
image-x
image-y
image-z
Repo
image-x
image-y
image-z
Repo
image-x
image-y
image-z
Repo
image-x
image-y
image-z

Module Recap



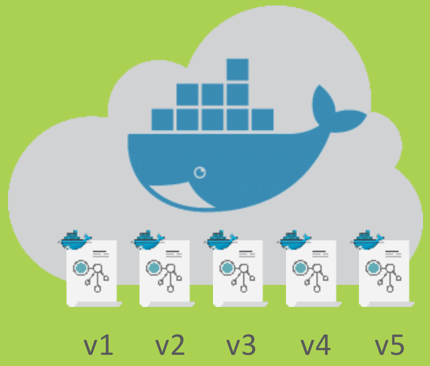
Docker Engine

Images



(runtime)

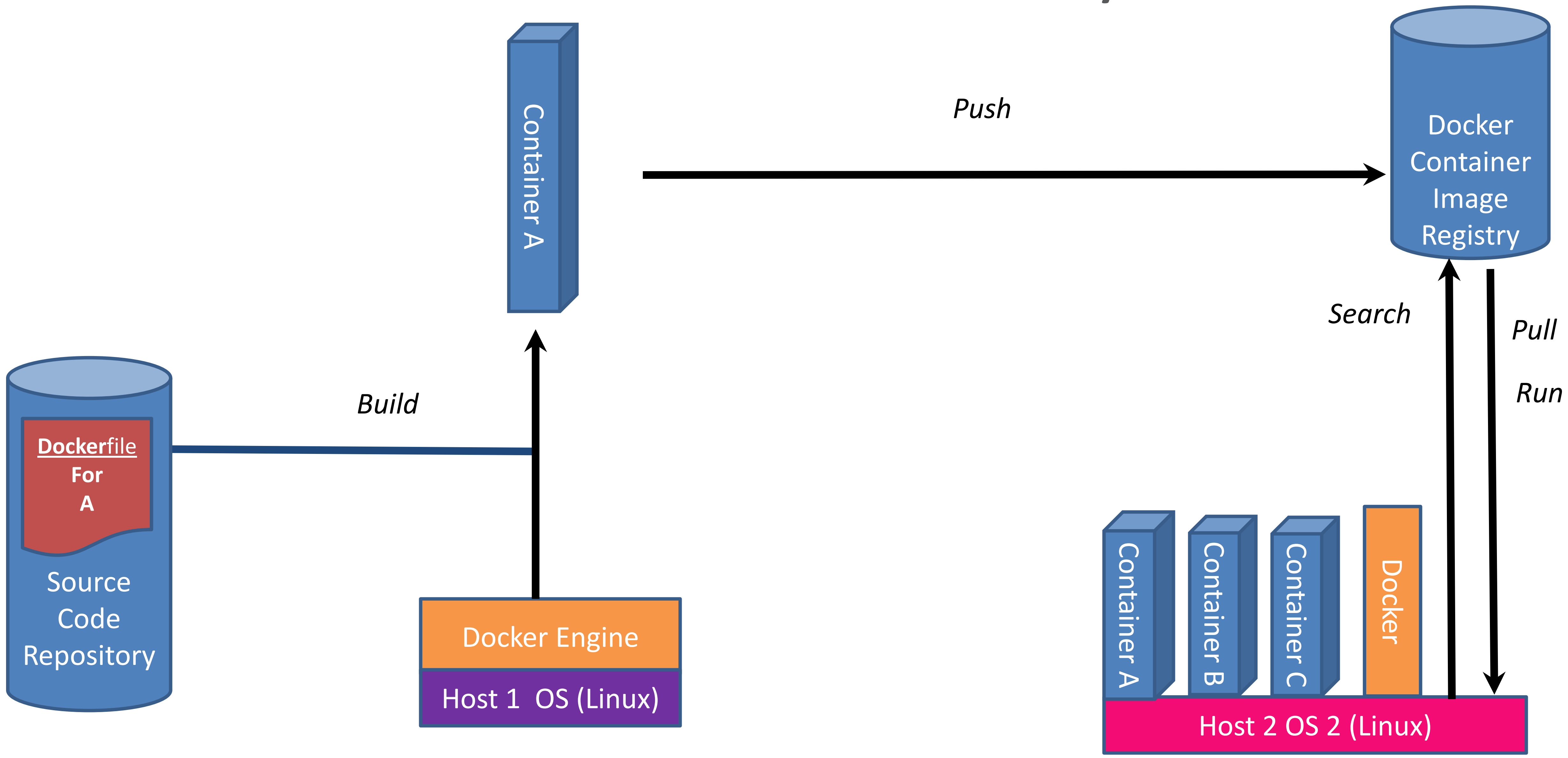
Containers



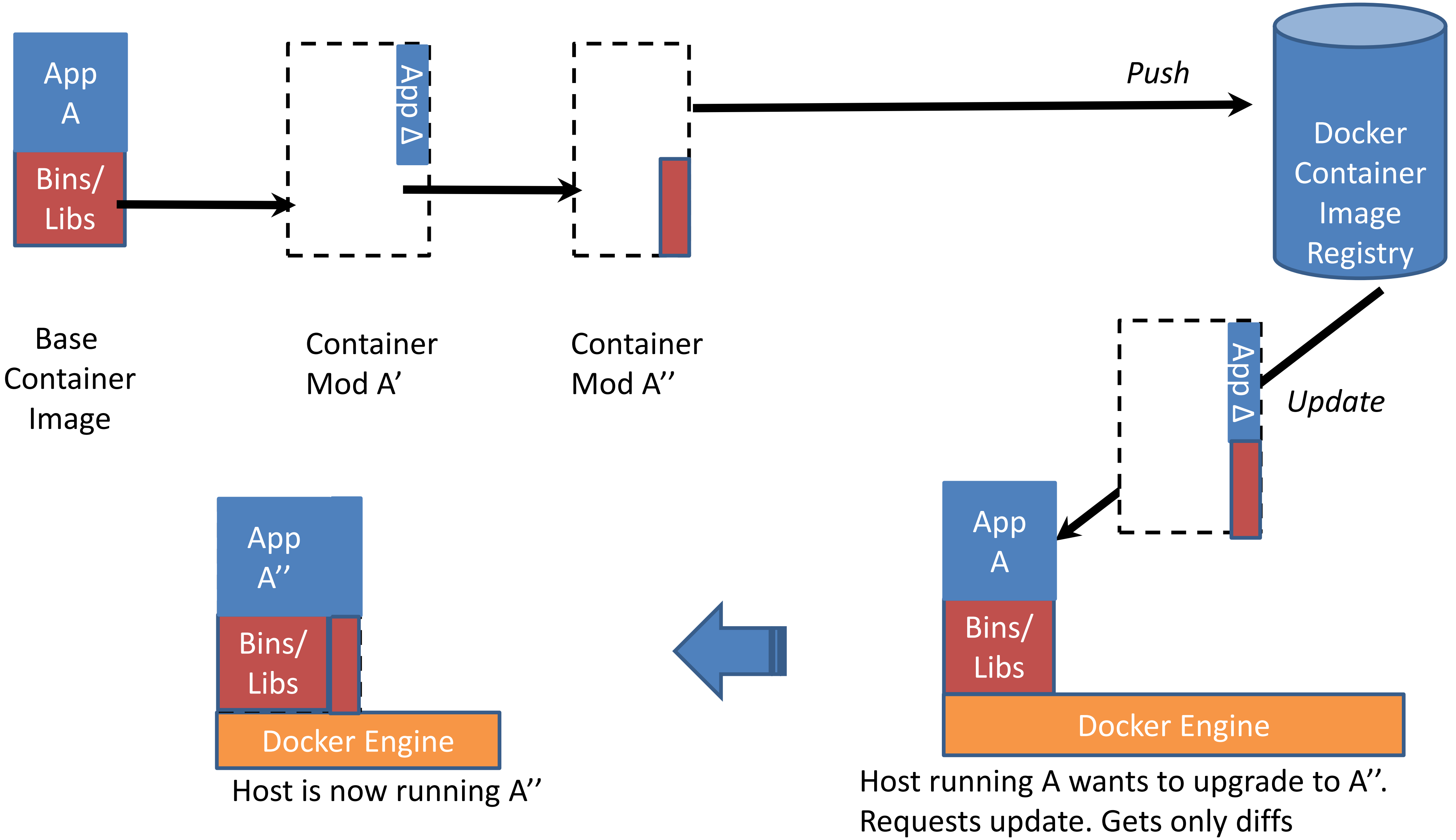
Registries & Repos

Docker Basic Workflow

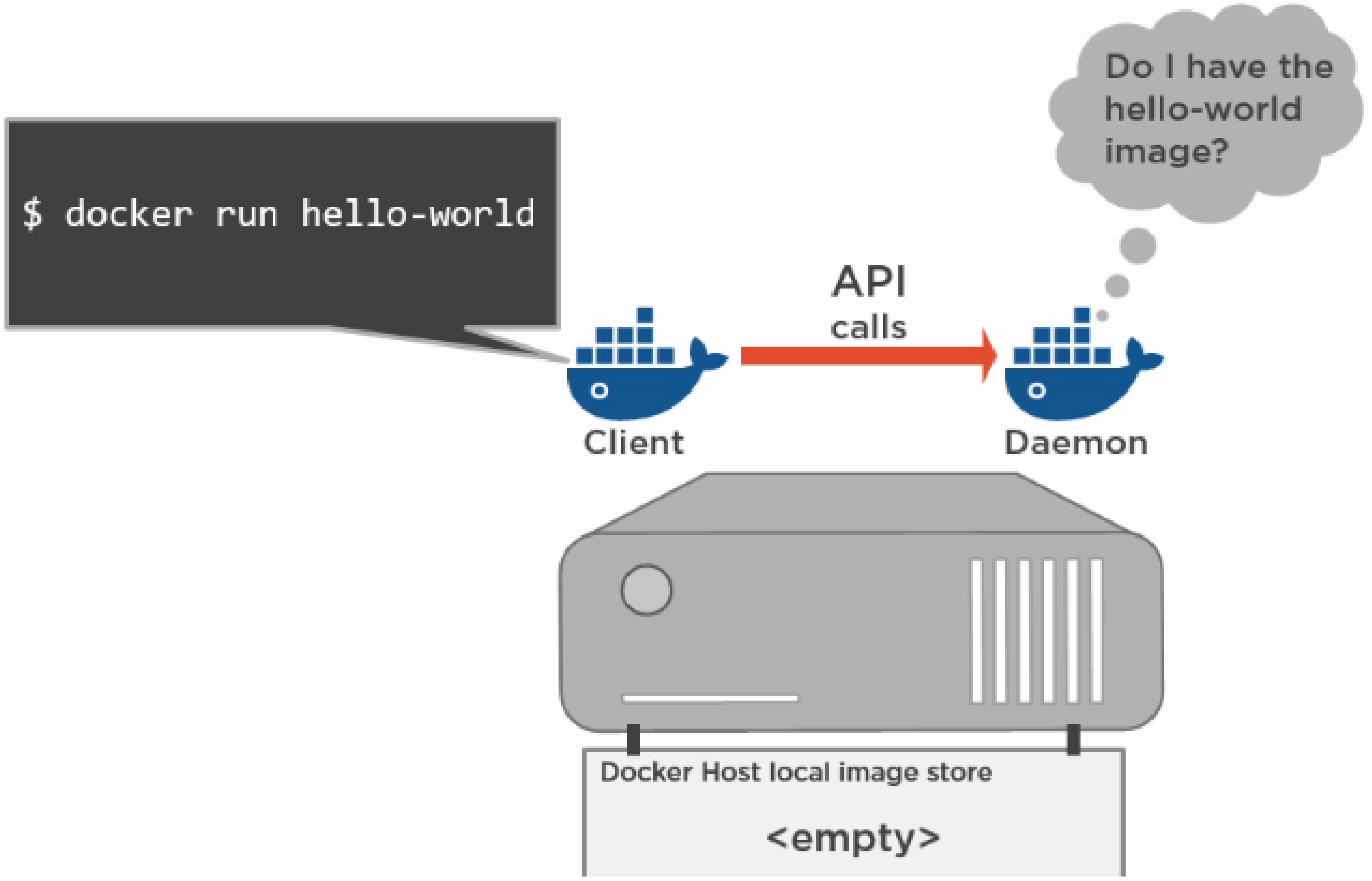
What are the basics of the Docker system?



Changes and Updates



Docker run Lifecycle



```
$ docker run hello-world
```

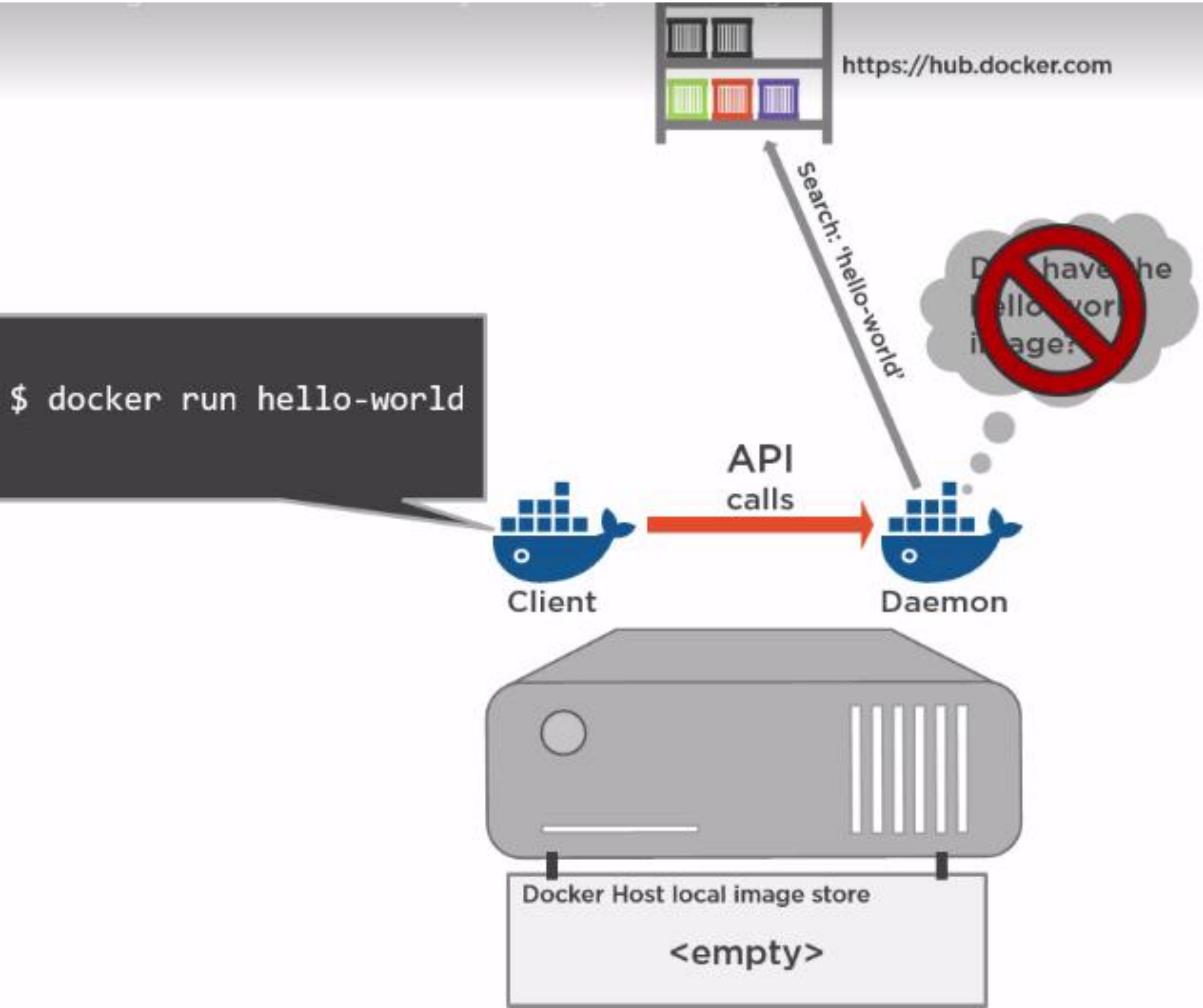
API calls
Client
Daemon
Do I have the hello-world image?
Docker Host local image store
<empty>

Installing Docker gives you the **client** and **daemon**

Client makes API calls to daemon

Daemon implements the *Docker Remote API*

`docker run` starts a new container



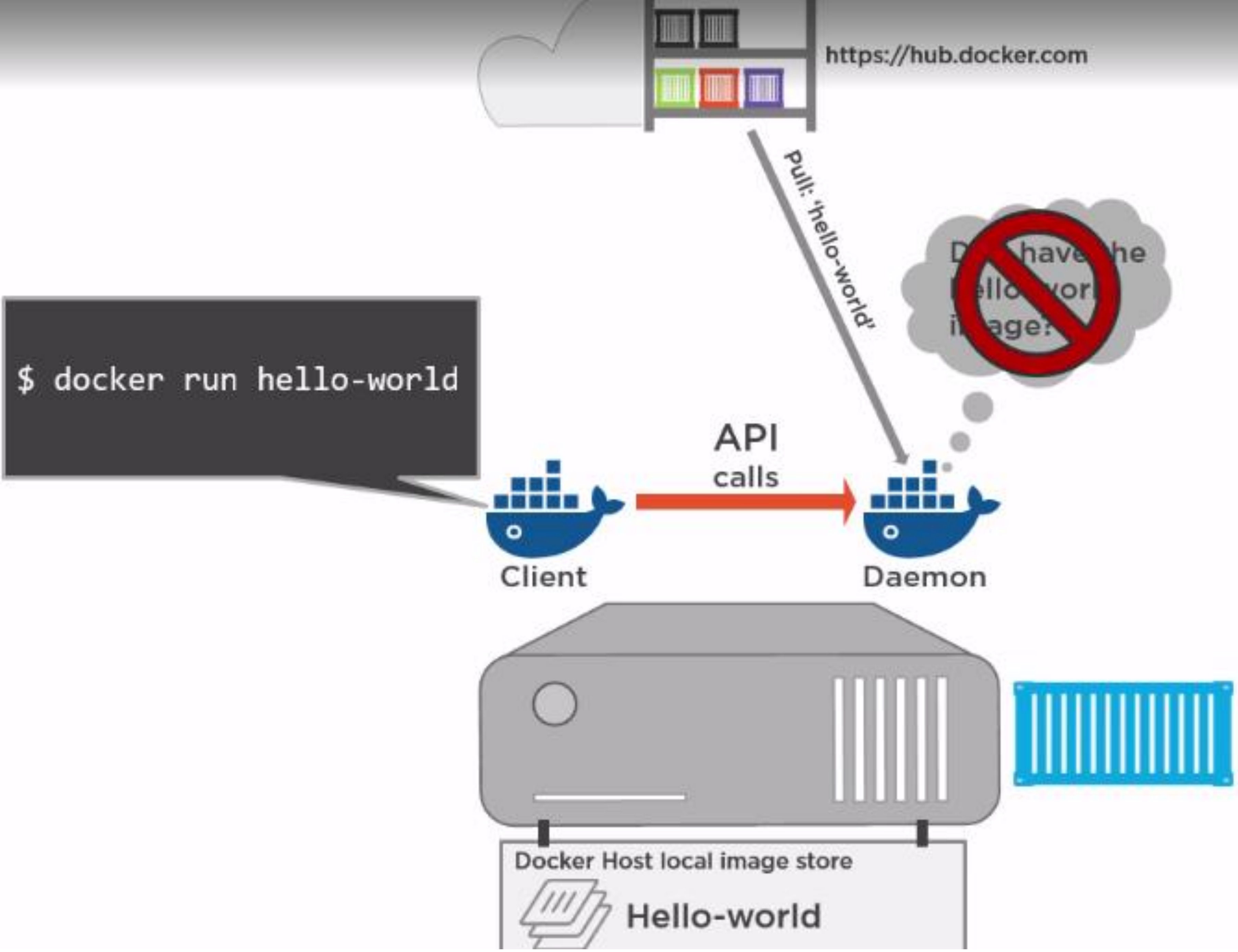
```
$ docker run hello-world
```

Installing Docker gives you the **client** and **daemon**

Client makes API calls to daemon

Daemon implements the *Docker Remote API*

`docker run` starts a new container


```
$ docker run hello-world
```

Installing Docker gives you the **client** and **daemon**

Client makes API calls to daemon

Daemon implements the *Docker Remote API*

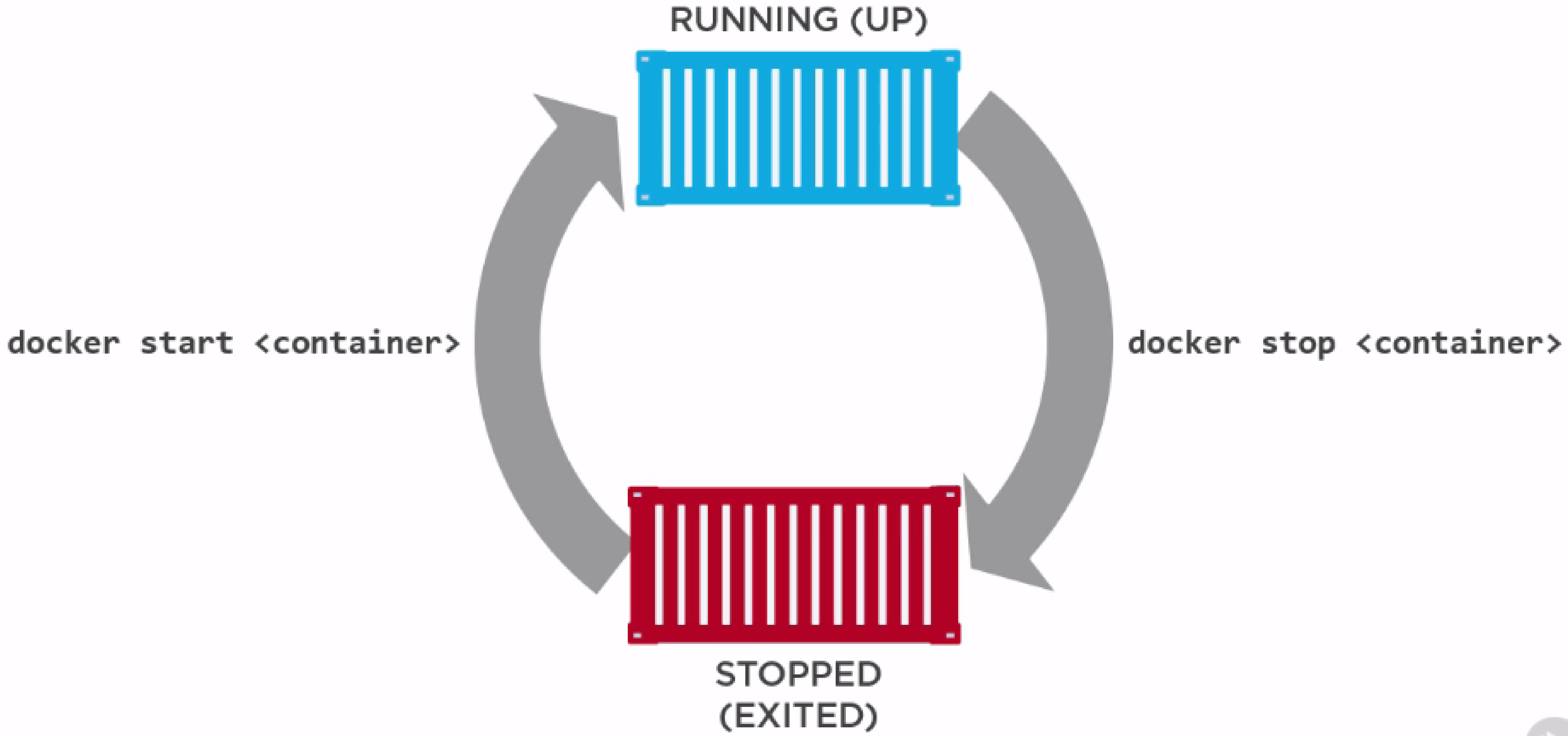
`docker run` starts a new container

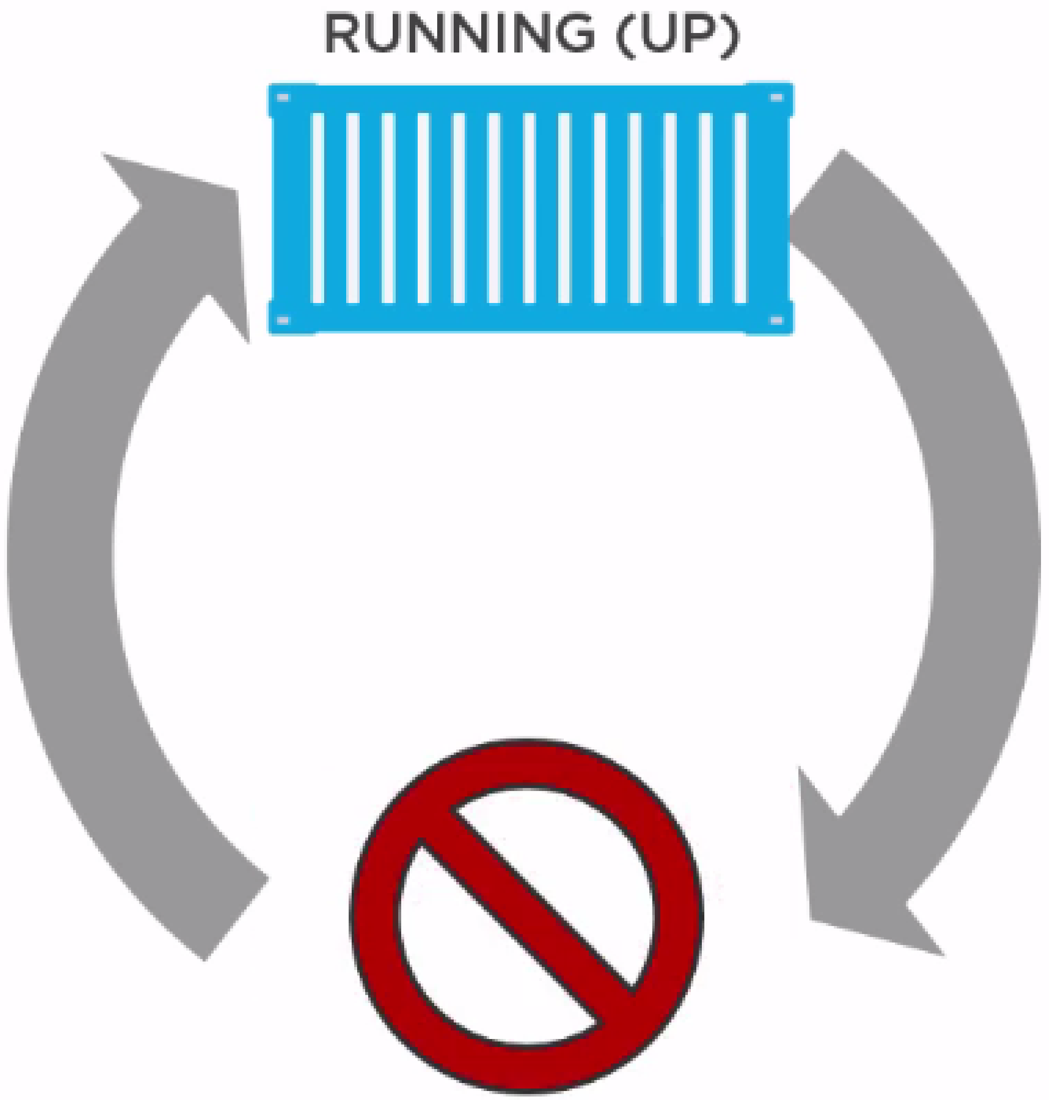
Docker Hub is the default public registry

The daemon will *pull* images that it doesn't already have

Containers and Images

Images	~	Stopped containers
Containers	~	Running Images


RUNNING (UP)
docker start <container>
docker stop <container>
STOPPED
(EXITED)

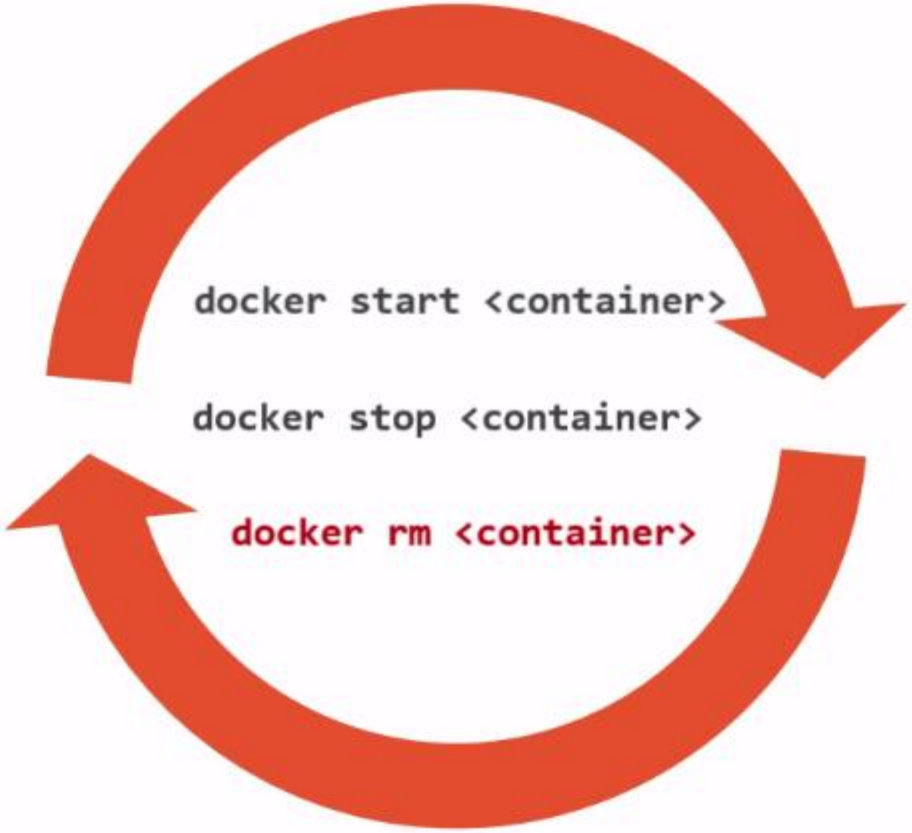


```
docker start <container>
```

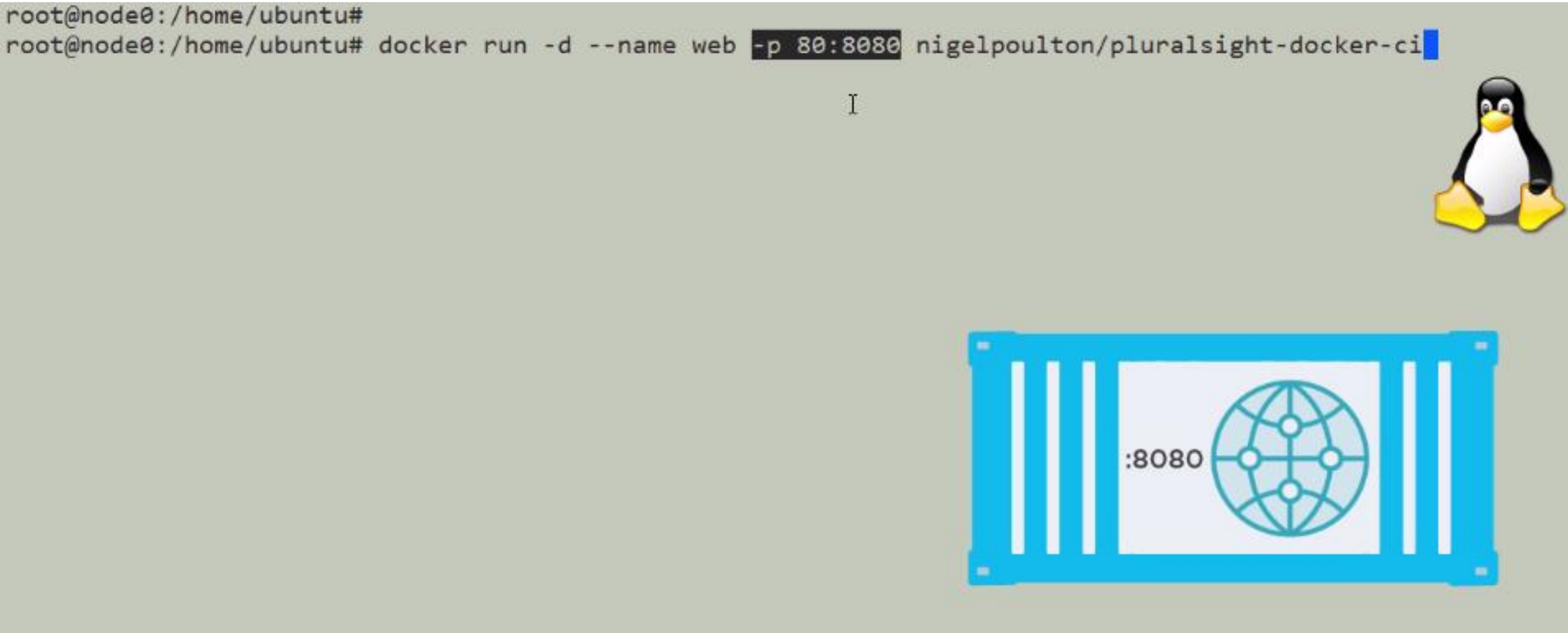
```
docker stop <container>
```

```
docker rm <container>
```

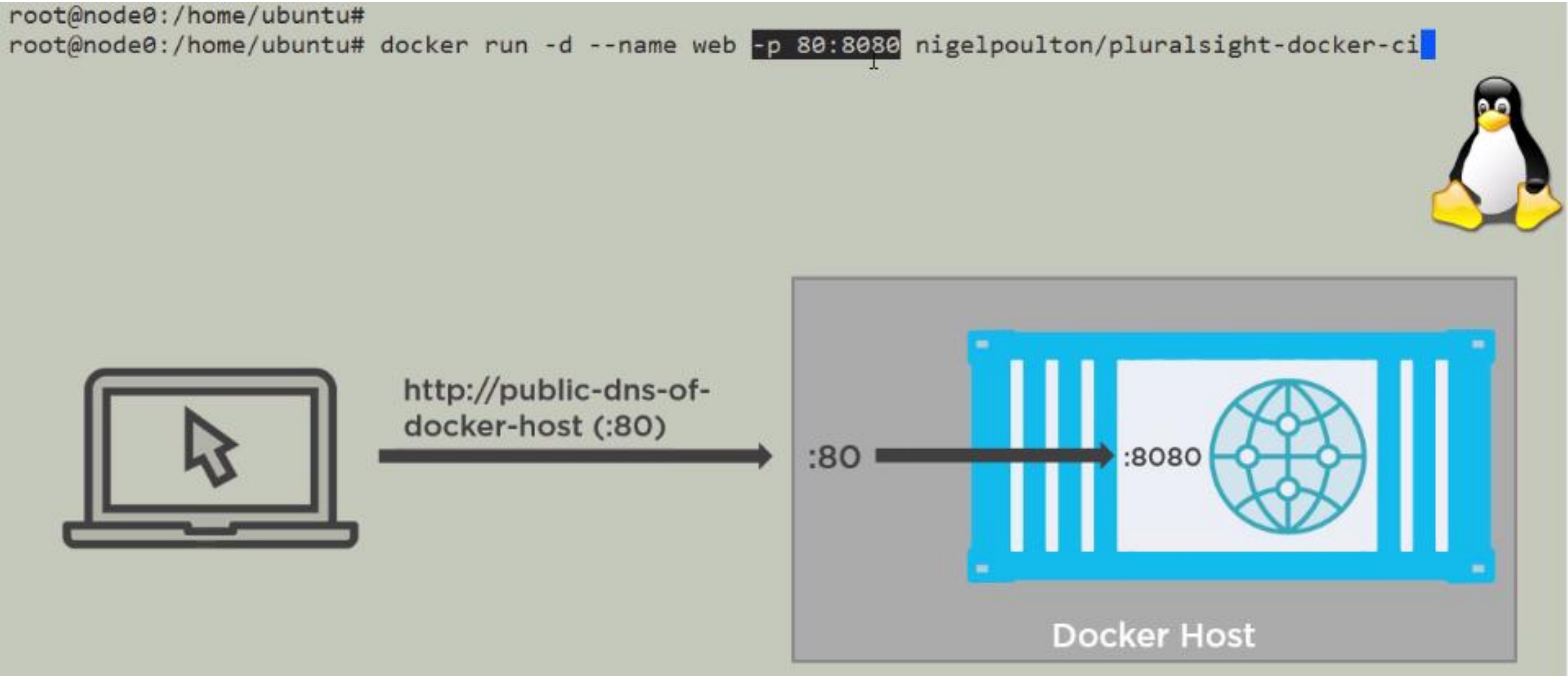
Container lifecycle ~ VM lifecycle



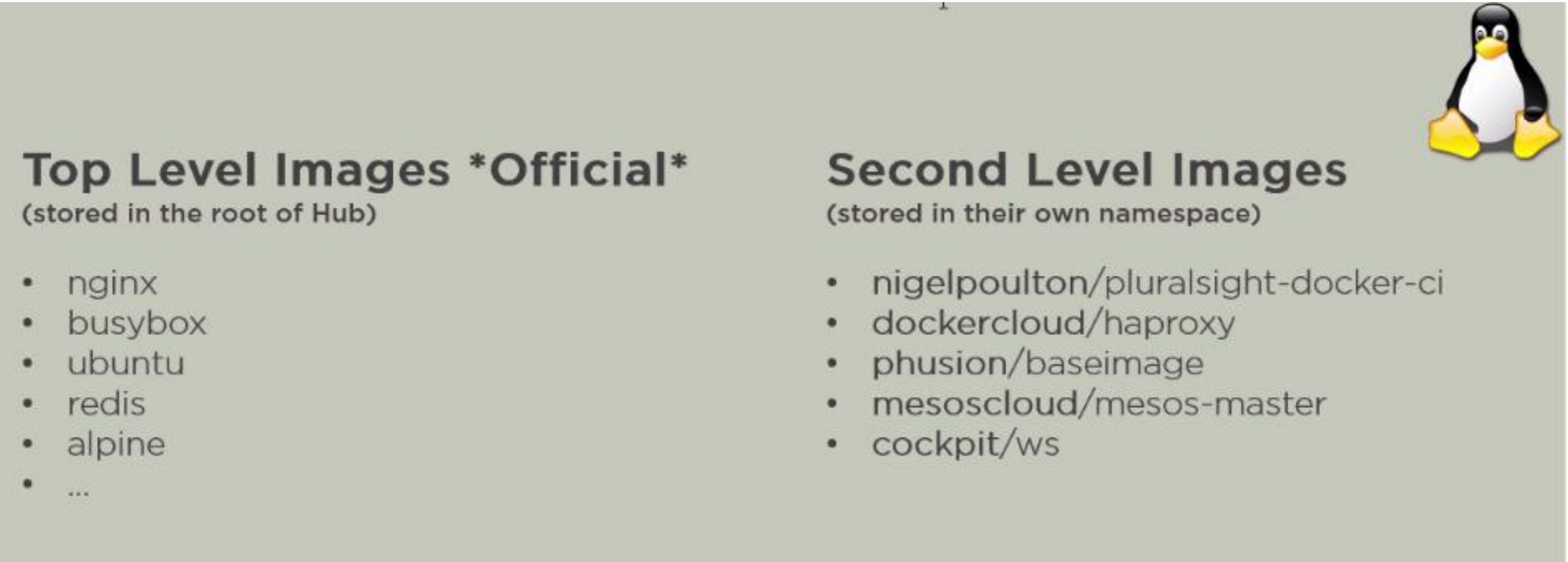
```
docker start <container>
docker stop <container>
docker rm <container>
```



```
root@node0:/home/ubuntu#
root@node0:/home/ubuntu# docker run -d --name web -p 80:8080 nigelpoulton/pluralsight-docker-ci
```



```
root@node0:/home/ubuntu#
root@node0:/home/ubuntu# docker run -d --name web -p 80:8080 nigelpoulton/pluralsight-docker-ci
```



Top Level Images *Official*

(stored in the root of Hub)

- nginx
- busybox
- ubuntu
- redis
- alpine
- ...

Second Level Images

(stored in their own namespace)

- nigelpoulton/pluralsight-docker-ci
- dockercloud/haproxy
- phusion/baseimage
- mesoscloud/mesos-master
- cockpit/ws

How to verify the version of docker?

```
> docker -v
> docker version
```

How to know Docker running?

```
> service docker.io status
> systemctl status docker.service
```

How to check details of Docker clients, deamon, containers, images, drivers, etc?

```
> docker info
```

Update Docker version

```
> wget -q0- https://get.docker.com/gpg | apt-key add -
> echo deb http://get.docker.com/ubantu docker main > /etc/apt/sources.list.d/docker.list
> apt-get update
> apt-get install lxc-docker
> docker version
```

Adding Users to the Docker Group (Docker Config (Need root to work)

```
> docker run -it ubuntu /bin/bash (as a non-root)
[ permission denied]
> cat /etc/group
> sudo gpasswd -a username docker
> cat /etc/group
> docker run -it ubuntu /bin/bash (as a non-root)
> logout
> login username
```

Setup Network to Docker Container

```
> docker -v
> netstat -tlp
> service docker stop
> docker -H ipaddress:port -d &
> netstat -tlp

> export DOCKER_HOST="tcp://ipaddress:port" (from another machine)
> docker version
```

Docker Images

- ➢docker pull -a fedora
- ➢Docker info

> docker run -it fedora /bin/bash
> docker images fedora

[Images are stored under /var/lib/docker/<storage drivers>

Docker Containers

```
> docker run -it ubuntu /bin/bash
> docker images
> docker ps
> docker attach <container_id>
> docker ps -a
```

Docker Registries and Repositories

hub.docker.com

Setup Jenkins Using Docker

Pull the official jenkins image from Docker repository.

```
> docker pull jenkins
```

Next, run a container using this image and map data directory from the container to the host; e.g in the example below /var/jenkins_home from the container is mapped to jenkins/ directory from the current path on the host. Jenkins 8080 port is also exposed to the host as 49001.

```
> docker run -d -p 49001:8080 -v $PWD/jenkins:/var/jenkins_home -t jenkins
```

Other commands

```
> docker run -p 8080:8080 jenkins
> docker create -v /var/jenkins_home --name jenkins-dv jenkins


> docker run -d -p 8080:8080 --volumes-from jenkins-dv --name myjenkins jenkins
> http://localhost:8080


> docker run -d -p 8080:8080 --volumes-from jenkins-dv --name myjenkins2 jenkins
```

Questions