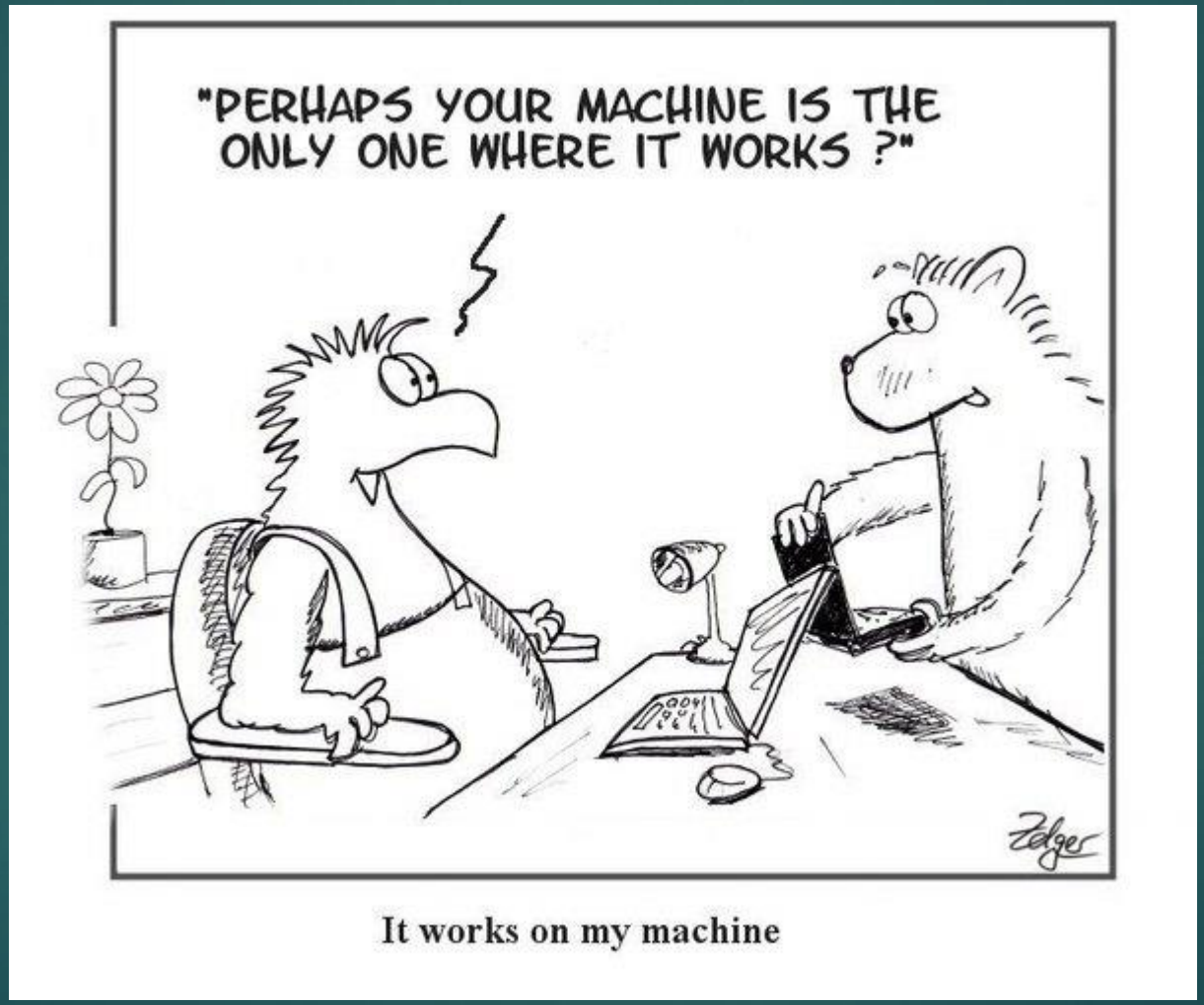
Docker for Developers

Challenges in Software Shipping

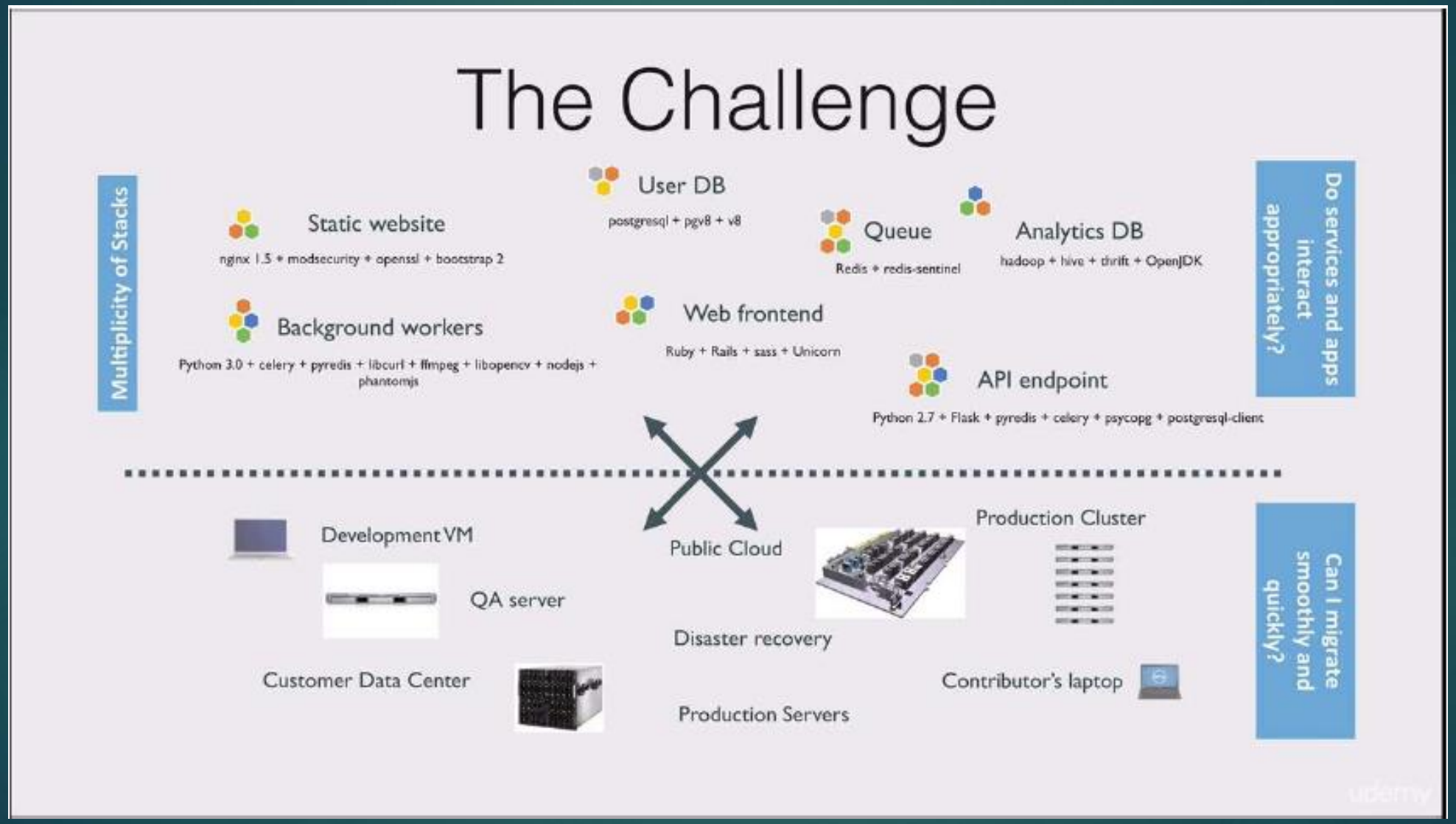
1. How to ship software?
2. How to ensure exact versioning between dev, test, and production beds?

Challenges in Software Shipping

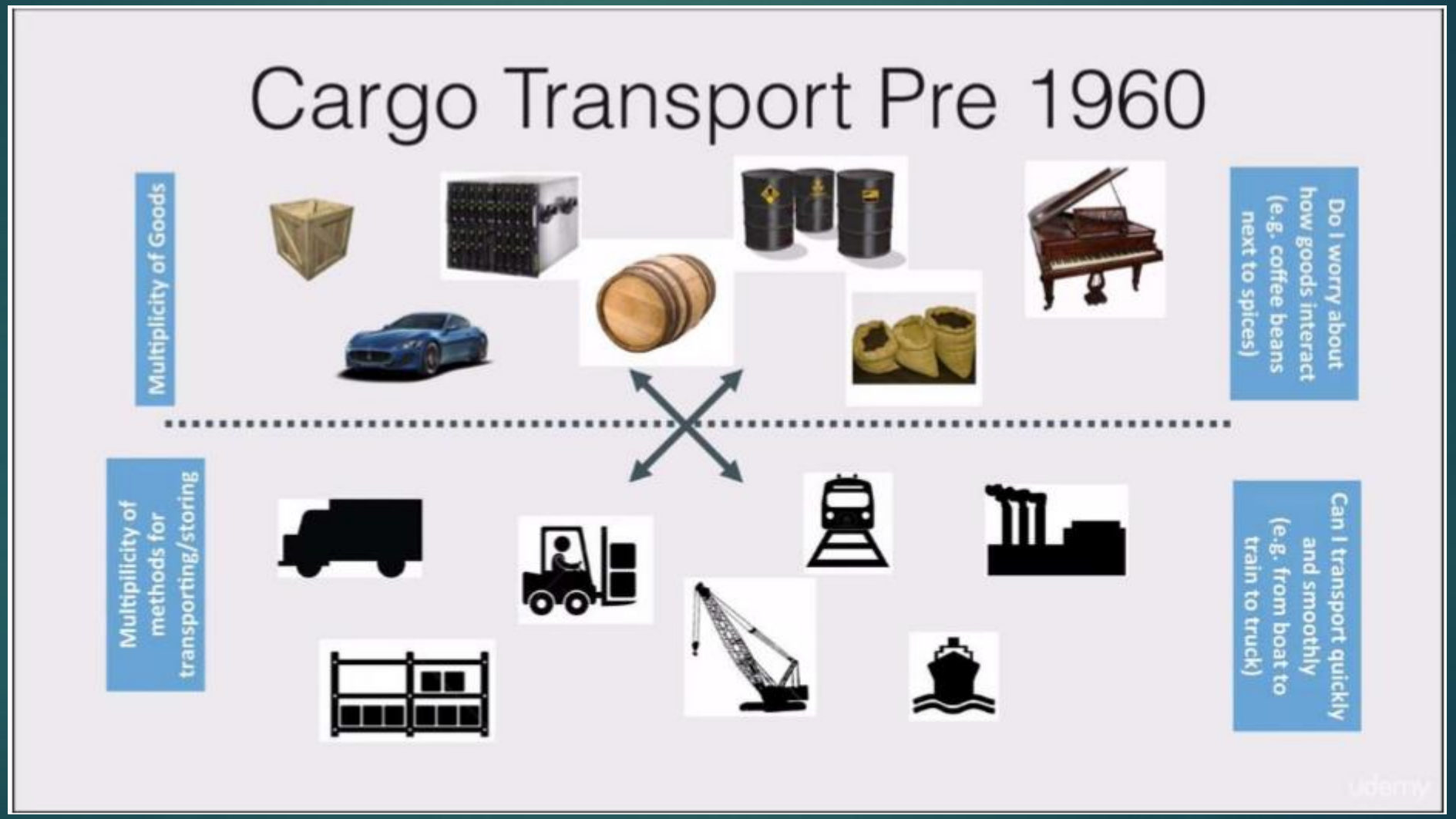


It works on my machine

The Challenge: How to Ship Software?

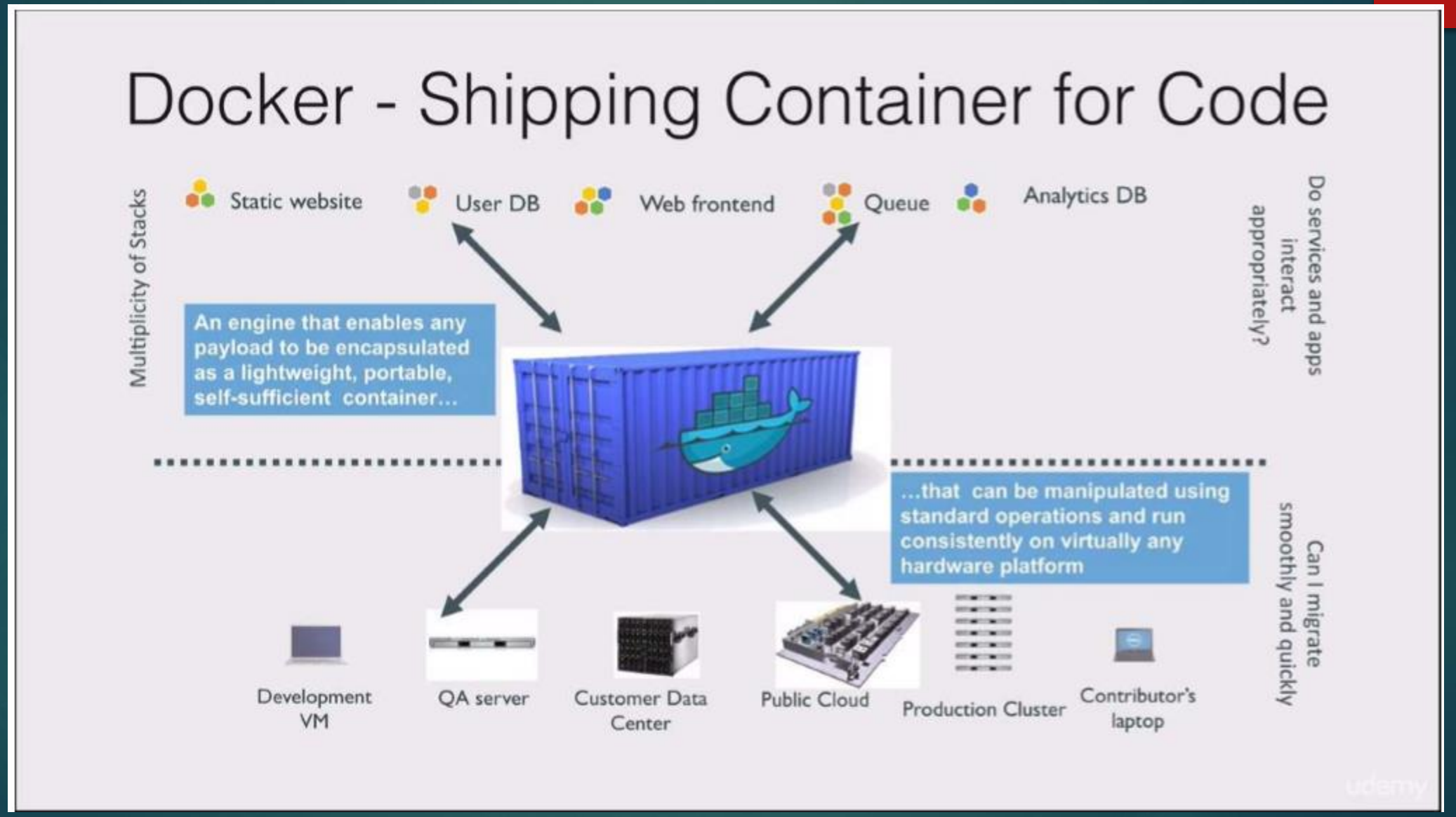


Analogy: Traditional Cargo Shipping



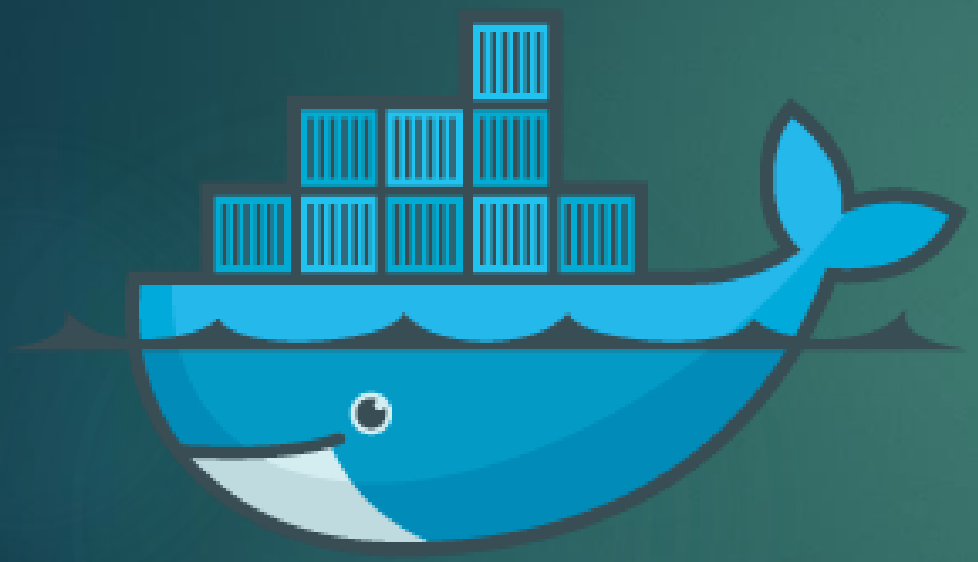
Analogy – Modern Cargo Shipping

Docker– Software Shipping Container Technology



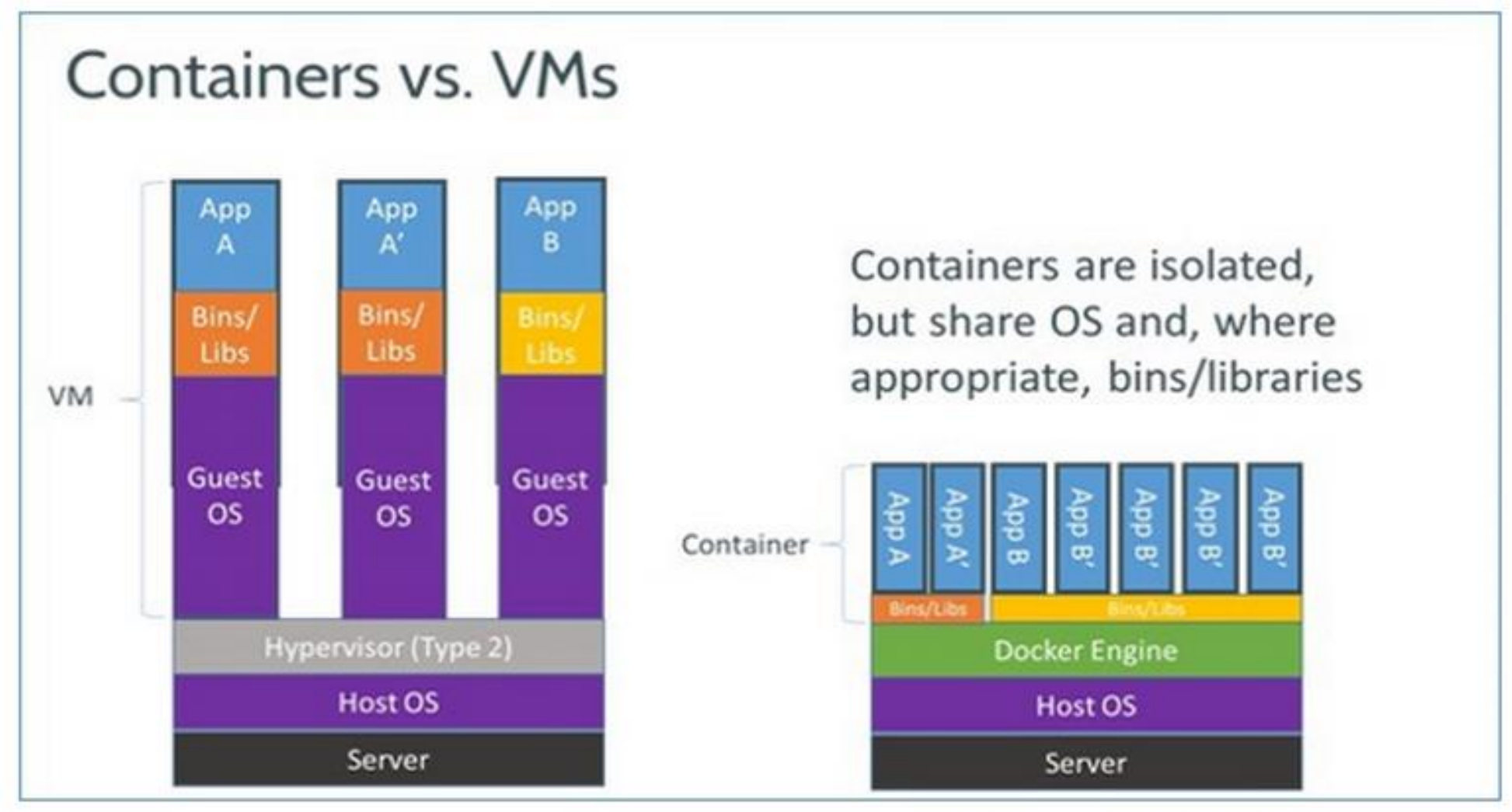
What is Docker?

Container Technology to Build, Ship, and Run Any App, Anywhere

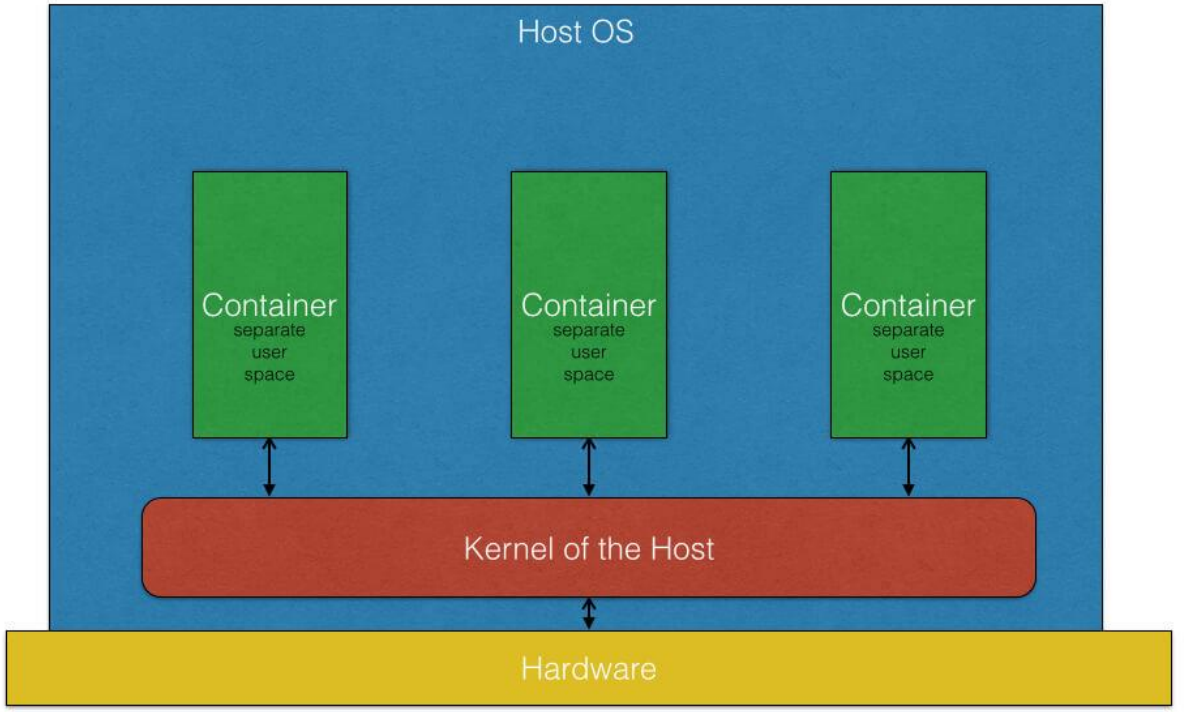


docker

Containers vs VMs



Containers vs VMs



Operating System/Container Virtualization

Just a Minute (JAM)

Btw, Docker is written in Go

Docker - Components

Image

- A read only template.
- Includes all the dependencies (such as frameworks) plus deployment and execution configuration to be used by a container runtime.
- Usually derives from multiple base images that are layers stacked on top of each other to form the container's file system.
- Is immutable once it has been created.

Container

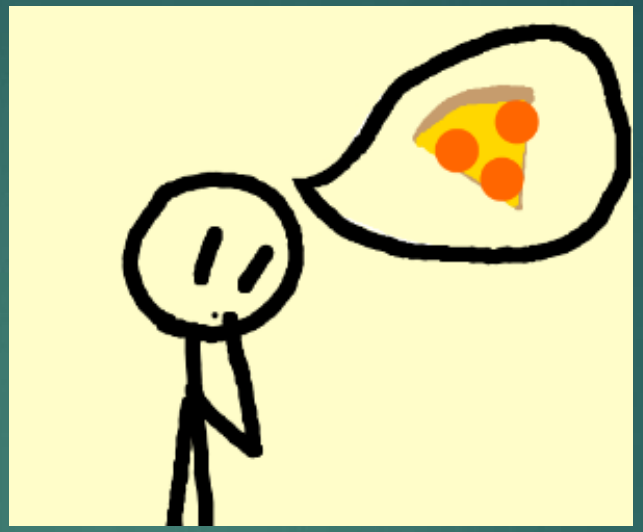
- A runnable instance of a Docker image.
- Can be created, started, stopped, and deleted.

Docker Images

What is a Docker Image?

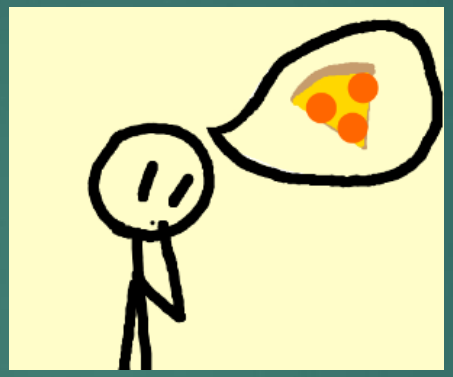
- An Image defines a Docker Container.
  - Similar in concept to a snapshot of a VM.
  - Or a class vs an instance of the class.
- Images are immutable.
  - Once built, the files making up an image do not change.

Layers in Docker Images



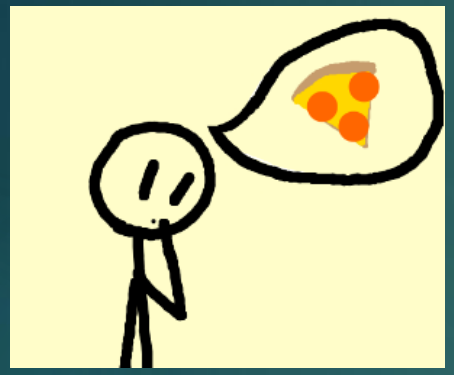
I want to eat Pizza with **Herbs**
Topping

Docker Images



I'll make the pizza

Docker Images



I'll make the pizza

My Pizza Image

Layer 5 (Herbs Topping)

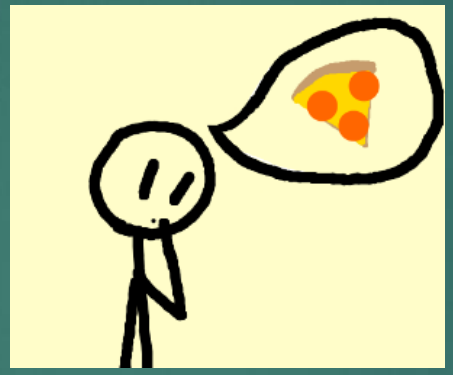
Layer 4 (Veg Topping)

Layer 3 (Ingredients)

Layer 2 (Sauce)

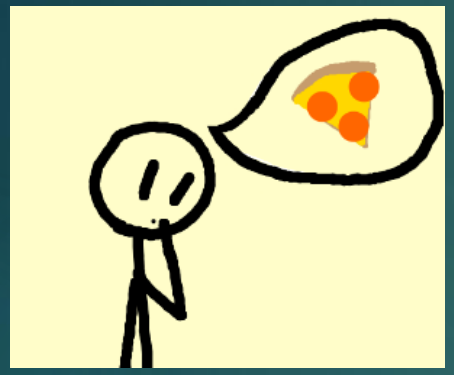
Layer 1 (Crust)

Docker Images



I'll order the pizza and add the Herbs Topping Layer

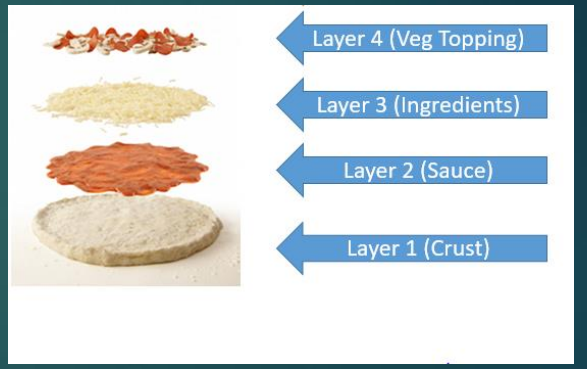
Docker Images



I'll order the pizza and add Herbs Topping Layer

My Pizza Image

Ordered Pizza Image

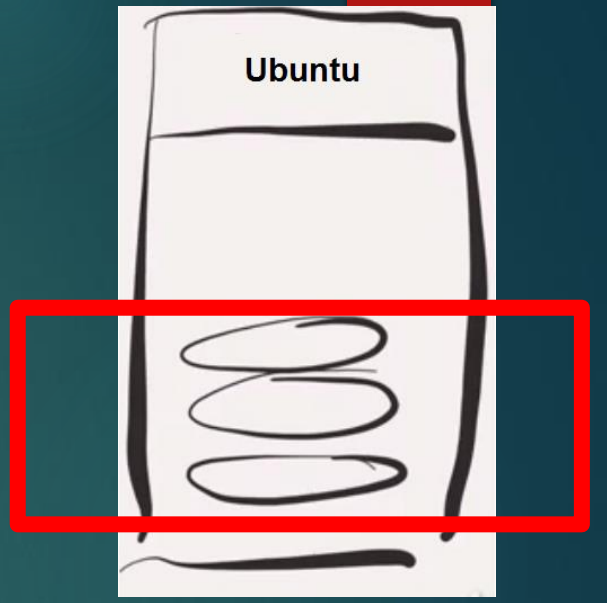


Docker Image Layers

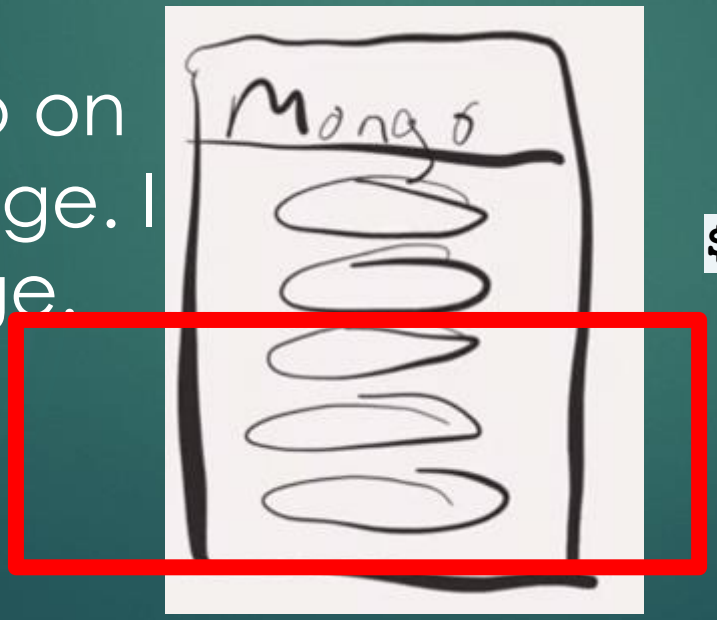
I want Mongo running
on Docker

Docker Image Layers

1. I will order(pull) the base Ubuntu image



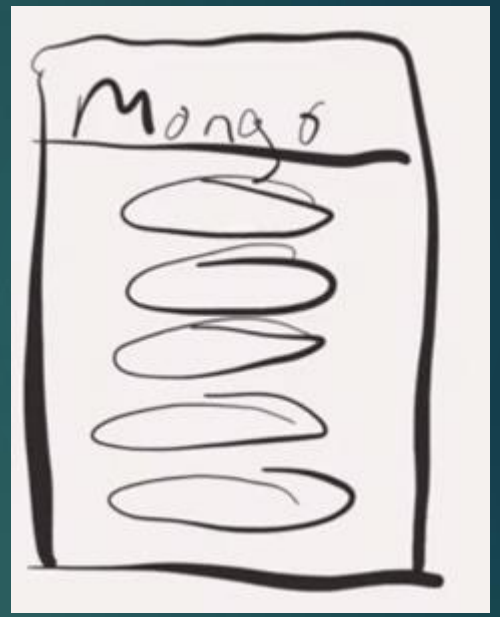
2. I will install Mongo on the base Ubuntu image. I get a Mongo Image.




```
$ sudo apt-get install -y mongodb-org
```

Docker Image Layers

2. I will order(pull) a Mongo image



Docker Images

Image Layers

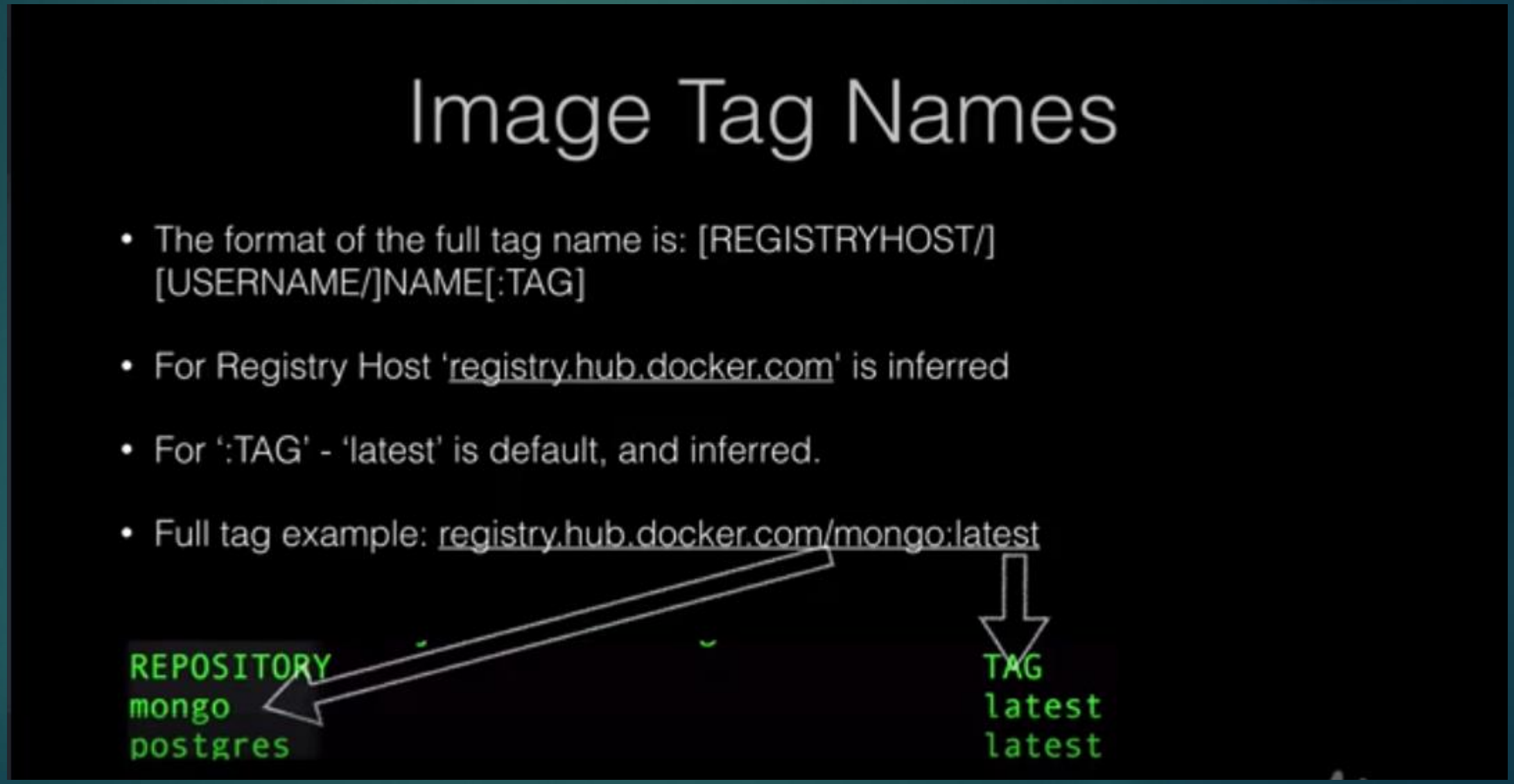
- Images are built in layers.
- Each layer is an immutable file, but is a collection of files and directories.
- Layers receive an ID, calculated via a SHA 256 hash of the layer contents.
  - Thus, if the layer contents change, the SHA 256 hash changes also.

Docker Images

Image Tag Names

- The hash values of images are referred to by ‘tag’ names.
  - This concept is very confusing at first.

Docker Images



Docker Images

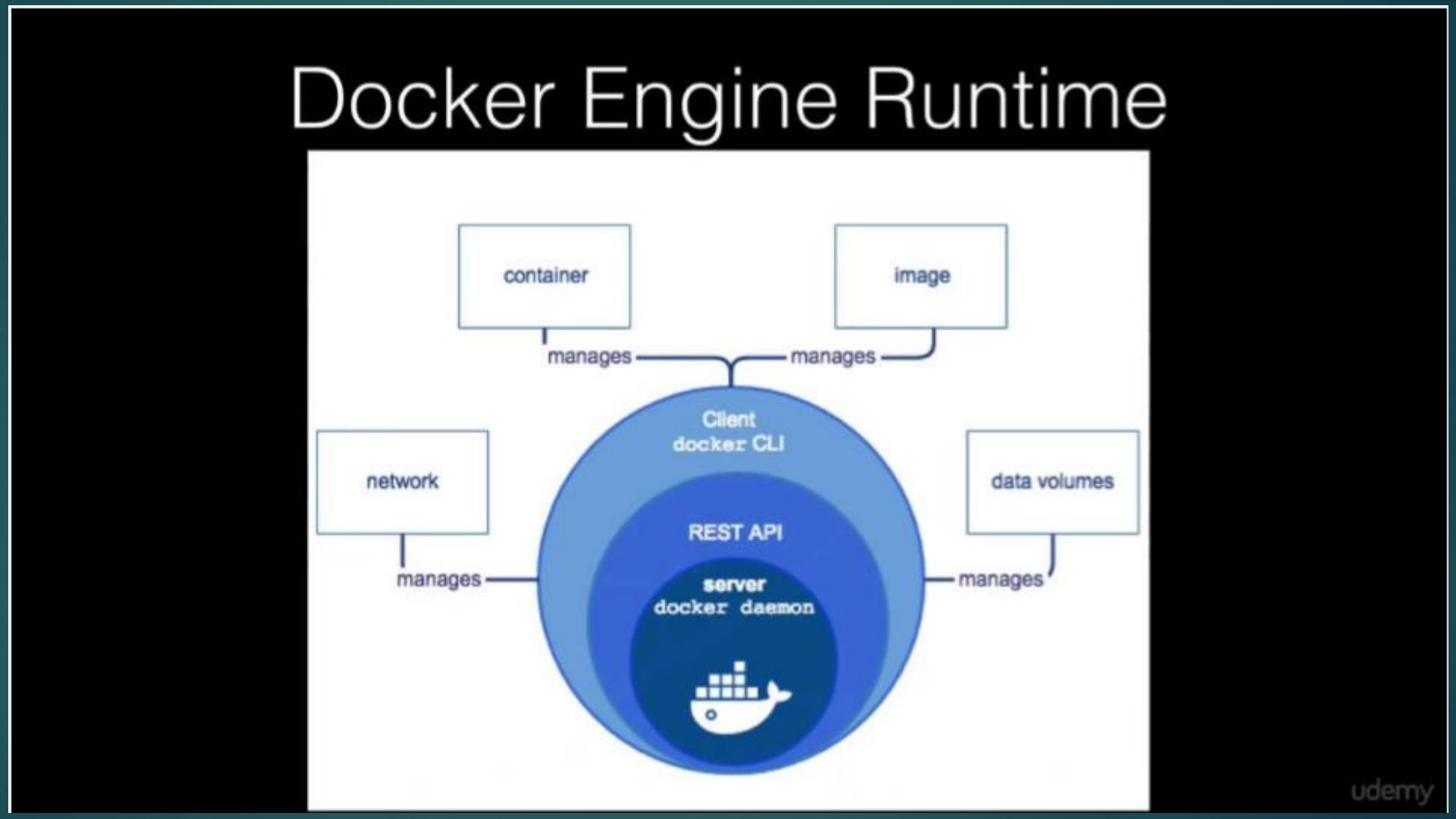
docker images

docker inspect mongo

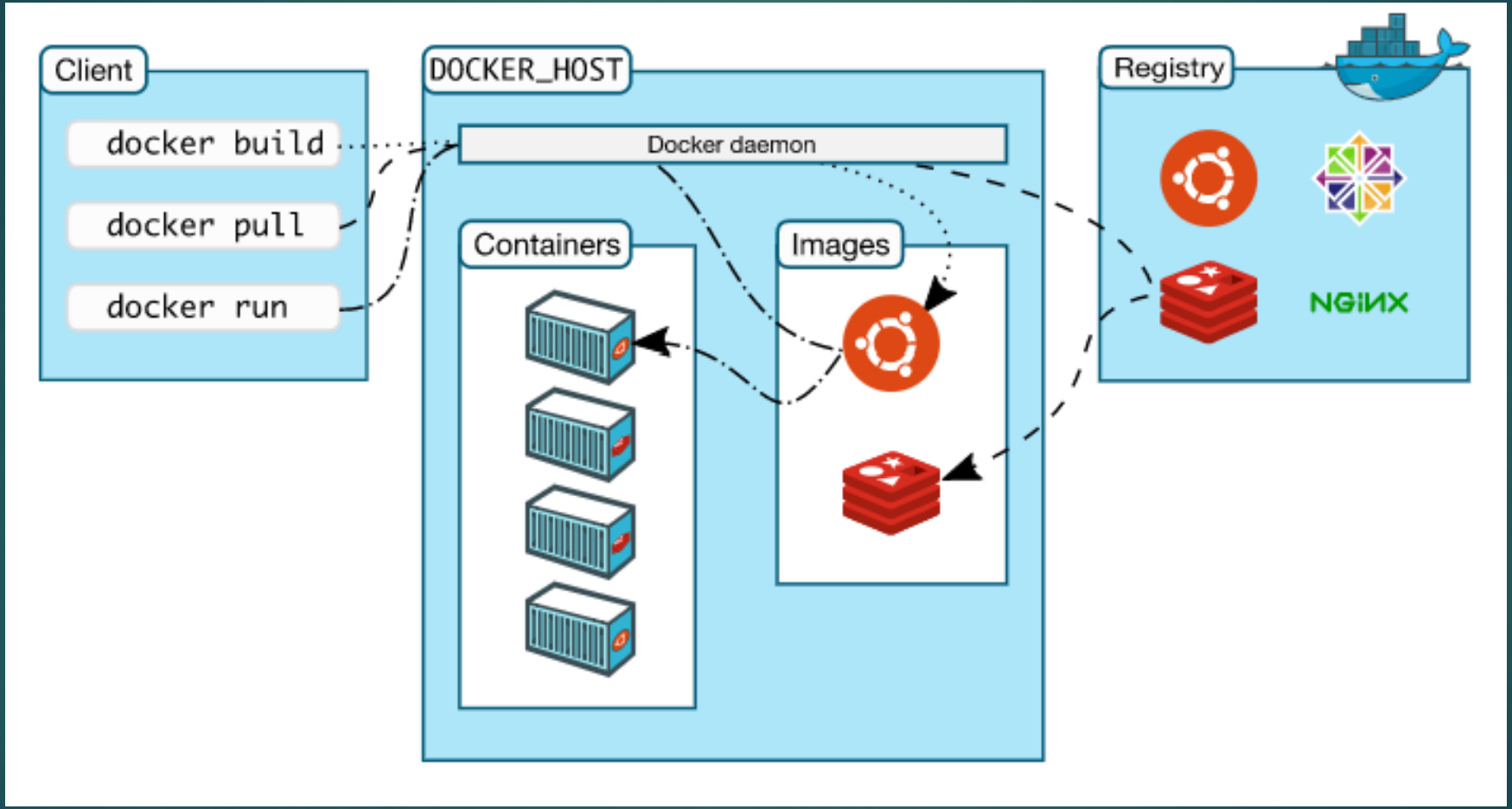
```
docker history -q webserver-image:v1 | wc -l
```

Using Docker

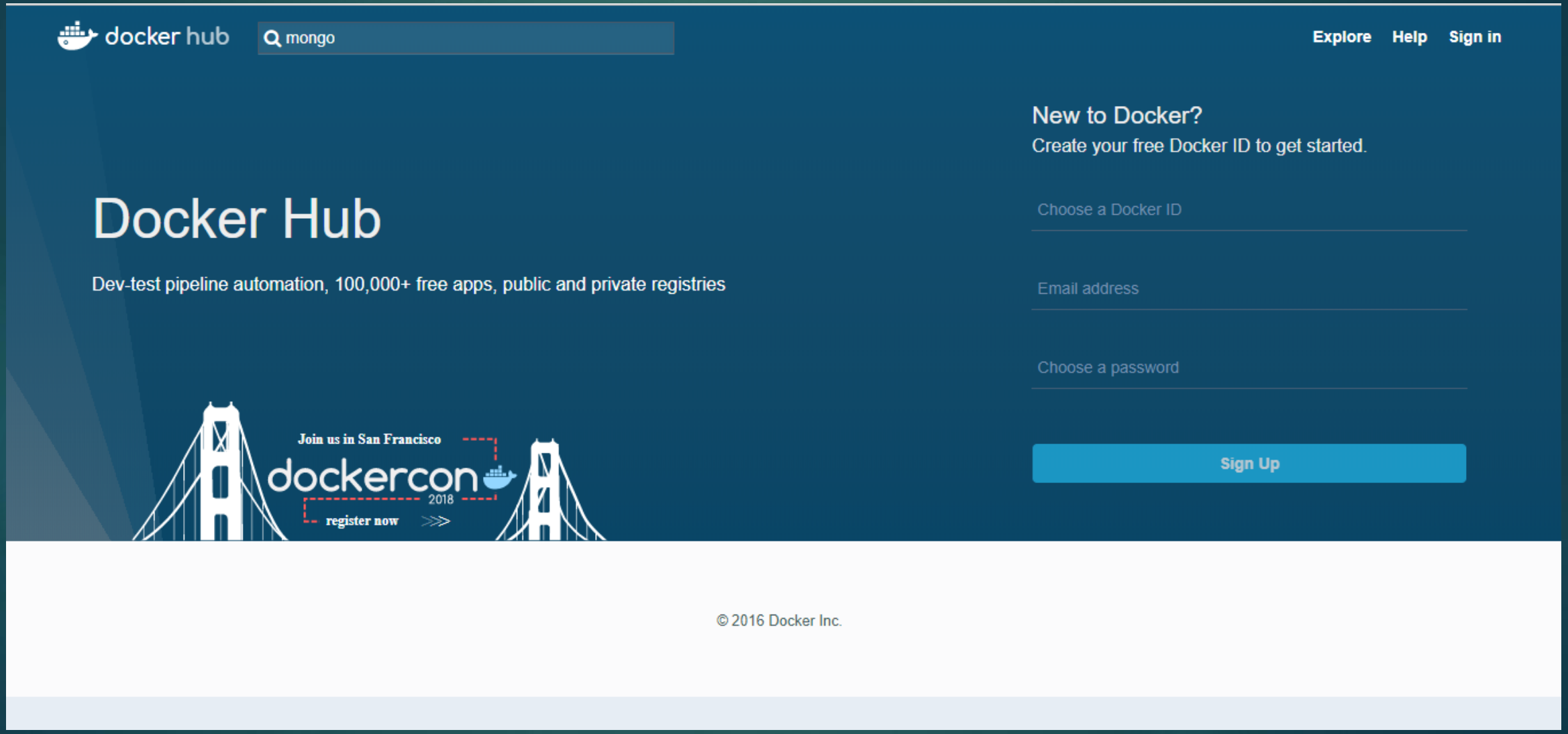
Docker– Software Shipping Container Technology



Docker– Software Shipping Container Technology



Docker Hub

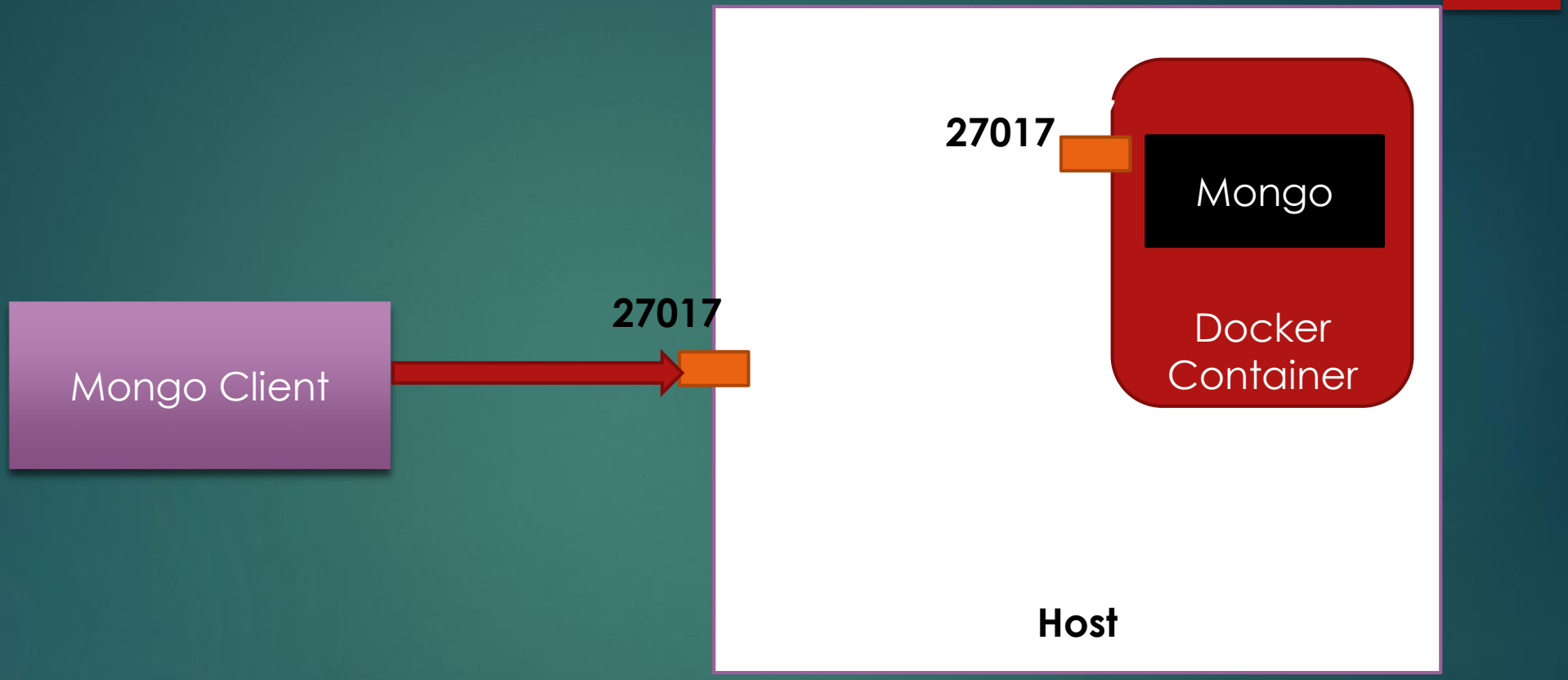

docker hub
mongo
Explore
Help
Sign in
New to Docker?
Create your free Docker ID to get started.
Docker Hub
Dev-test pipeline automation, 100,000+ free apps, public and private registries
Choose a Docker ID
Email address
Choose a password
Sign Up
Join us in San Francisco
dockercon
2018
register now
© 2016 Docker Inc.

Using Docker

docker run -d mongo

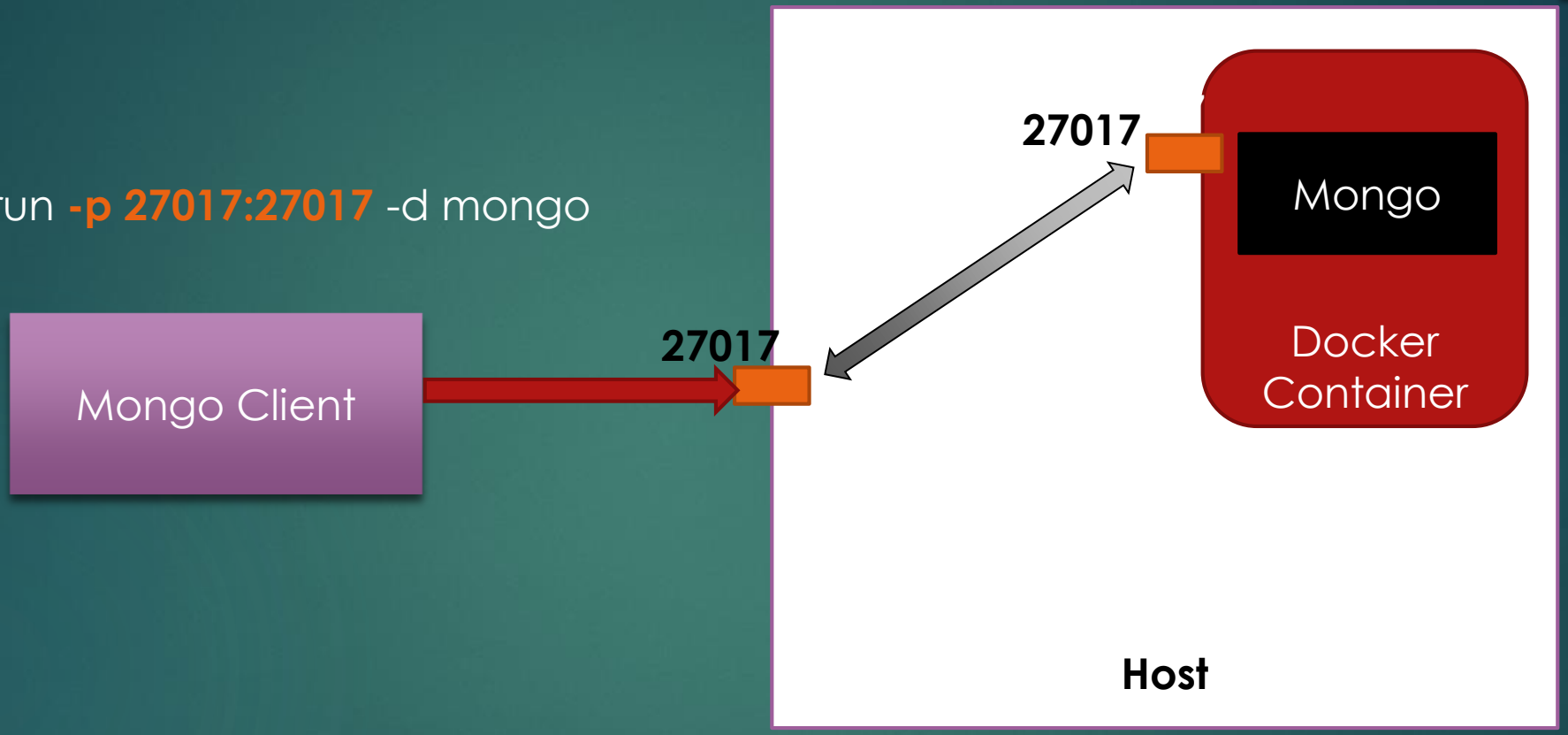
docker pull mongo
docker run –d mongo

What Happened?



Port Mapping

docker run **-p 27017:27017** -d mongo



Assigning Storage

docker run -p 27017:27017 **-v /users/data:/data/db** -d mongo

Building your Own Image

Dockerflle

```
FROM java:8-jre

ADD ./target/MovieCruiserServer-0.0.1-SNAPSHOT.jar /usr/src/MovieCruiserServer-0.0.1-SNAPSHOT.jar

WORKDIR usr/src

ENTRYPOINT ["java","-jar", "MovieCruiserServer-0.0.1-SNAPSHOT.jar"]
```

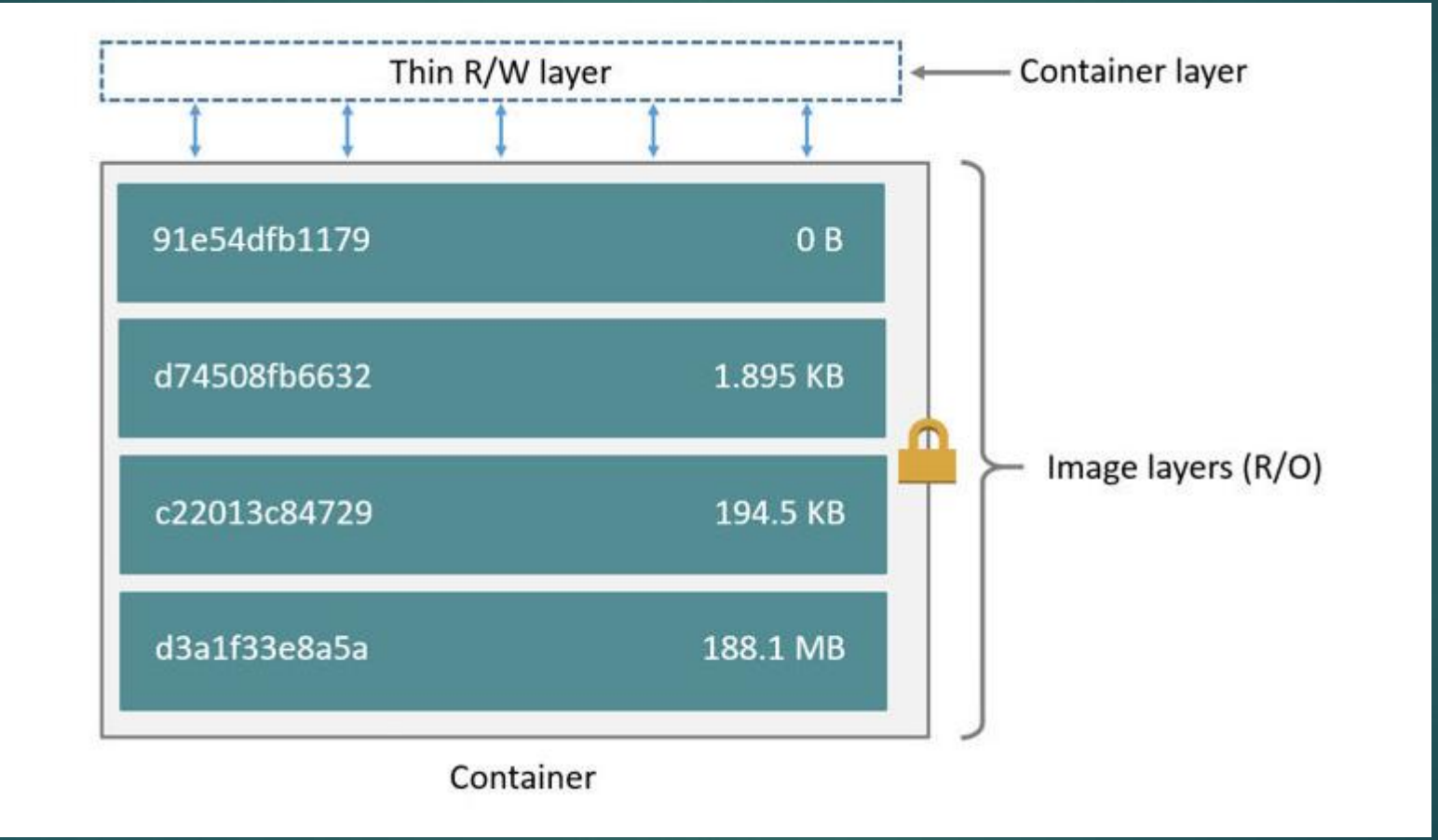
Building Image

```
docker build -t movieservice-image:v1 .
```

Running Container

```
docker run -d -p 8080:8080 movieservice
```

Running Container



House keeping

Executing Commands inside Running Container

```
docker exec -it <container_name> bash
```

Pushing to Docker Hub

```
docker login
```

```
docker tag <image_id> <dockerhub_id>/<image_name>: tag
```

```
docker push <image_name>
```

House keeping: Containers

Kill all running docker containers

```
docker kill $(docker ps -q)
```

Delete all stopped containers

```
docker rm $(docker ps –a -q)
```

House keeping: Images

Cleaning Up Images

- Remove a Docker Image
  - `docker rmi <image name>`
- Delete Untagged (dangling) Images
  - `docker rmi $(docker images -q -f dangling=true)`
- Delete All Images
  - `docker rmi $(docker images -q)`

House keeping: Volumes

Cleaning Up Volumes

- Once a volume is no longer associated with a container, it is considered ‘dangling’.
- Remove all dangling volumes
  - `docker volume rm $(docker volume ls -f dangling=true -q)`
- NOTE: Does not remove files from host system in shared volumes.

Docker Cheat Sheet

Pull Image `docker pull mvertes/alpine-mongo`	View Images `docker images`	Run Container `docker run <image name>` *To run in background use -d: `docker run -d <image name>`*
List Running Containers `docker ps`	Stop container `docker stop <container_id>/<container_name>`	View Container Logs `docker log <container_id/name>` *Use -f to view log real time*
Remove all stopped containers `docker ps -aq --no-trunc | xargs docker rm`	Remove container by name `docker rm $(docker ps -aq --filter name=myContainerName)`	
List all Exited Containers `docker ps -aq -f status=exited`		

Port Forwarding
`docker run -d -p <host_port>:<guest_port> <image name>`
`docker run -d -p 2700:2700 <image name>`		

Thank you