Docker 101

Workshop

MAERSK SEALAND
ARTHUR MÆRSK
RØNNE
Kelvin Davies
©kelvin@kelvindavies.co.uk

A brief tour of Docker

By the end of this session you will understand:

- What is a container and why you may want one
- How to create your own containers
- How to share your containers
- How to create multi-container applications

What the why now?

If docker is the answer, what is the question?

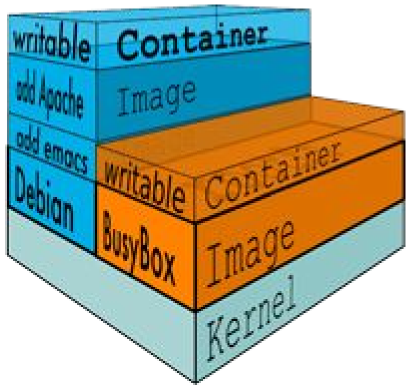
Docker is a platform

Docker is a platform for developing, shipping and running applications using container technology

The Docker Platform consists of multiple products/tools:

- Docker Engine
- Docker Hub
- Docker Trusted Registry
- Docker Machine
- Docker Swarm
- Docker Compose
- Kitematic

Dependency management



Docker provides a means to package and application with all its **dependencies** into standardized unit for software development

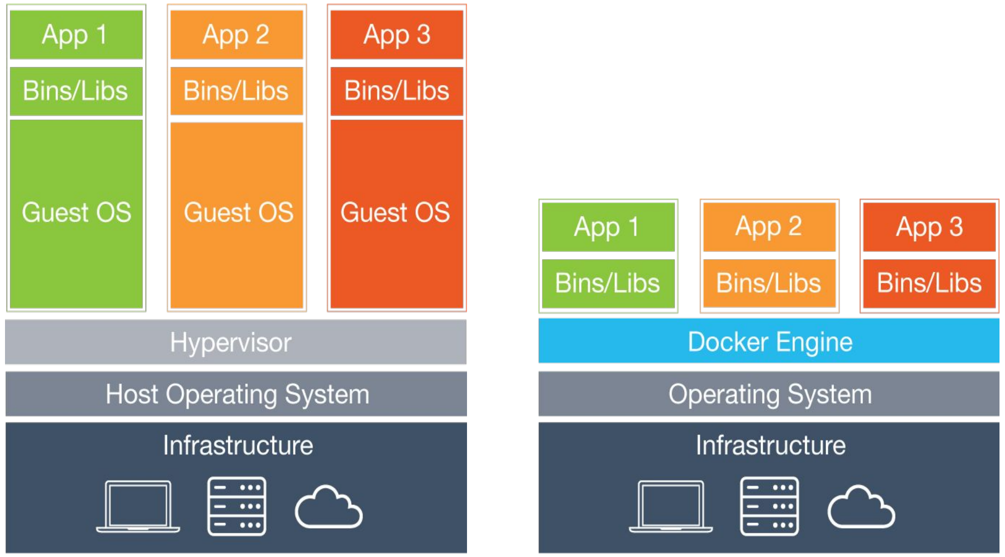
It provides **isolation**, so applications on the same host and stack can avoid dependency conflict

It is **portable**, so you can be sure to have exactly the same dependencies at runtime during development, testing and in production

From https://www.docker.com/what-docker

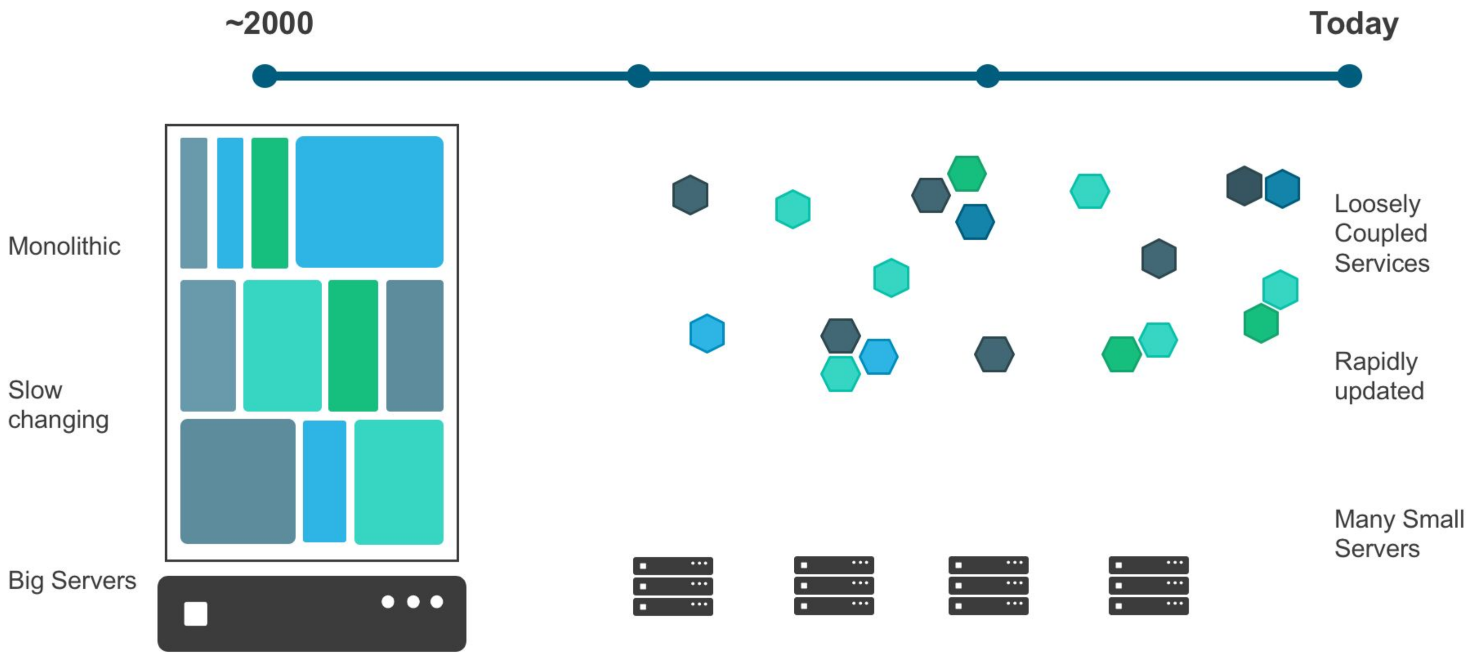
Resource Utilization

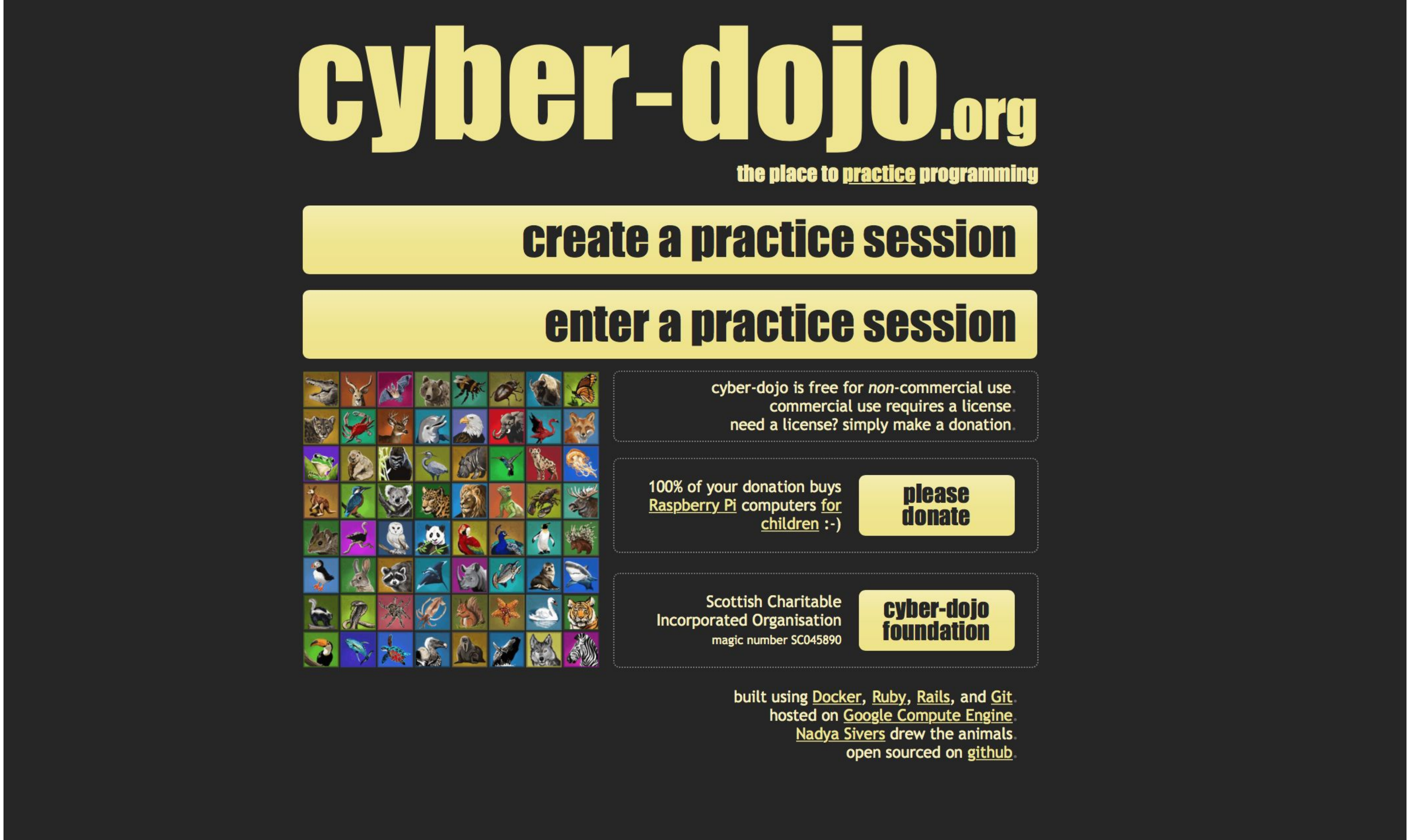
Better utilization, more portable, shared operating system



From https://www.docker.com/what-docker

Transforming the Application Landscape





cyber-dojo.org

the place to practice programming

create a practice session

enter a practice session

cyber-dojo is free for *non*-commercial use.
commercial use requires a license.
need a license? simply make a donation.

100% of your donation buys Raspberry Pi computers for children :-)

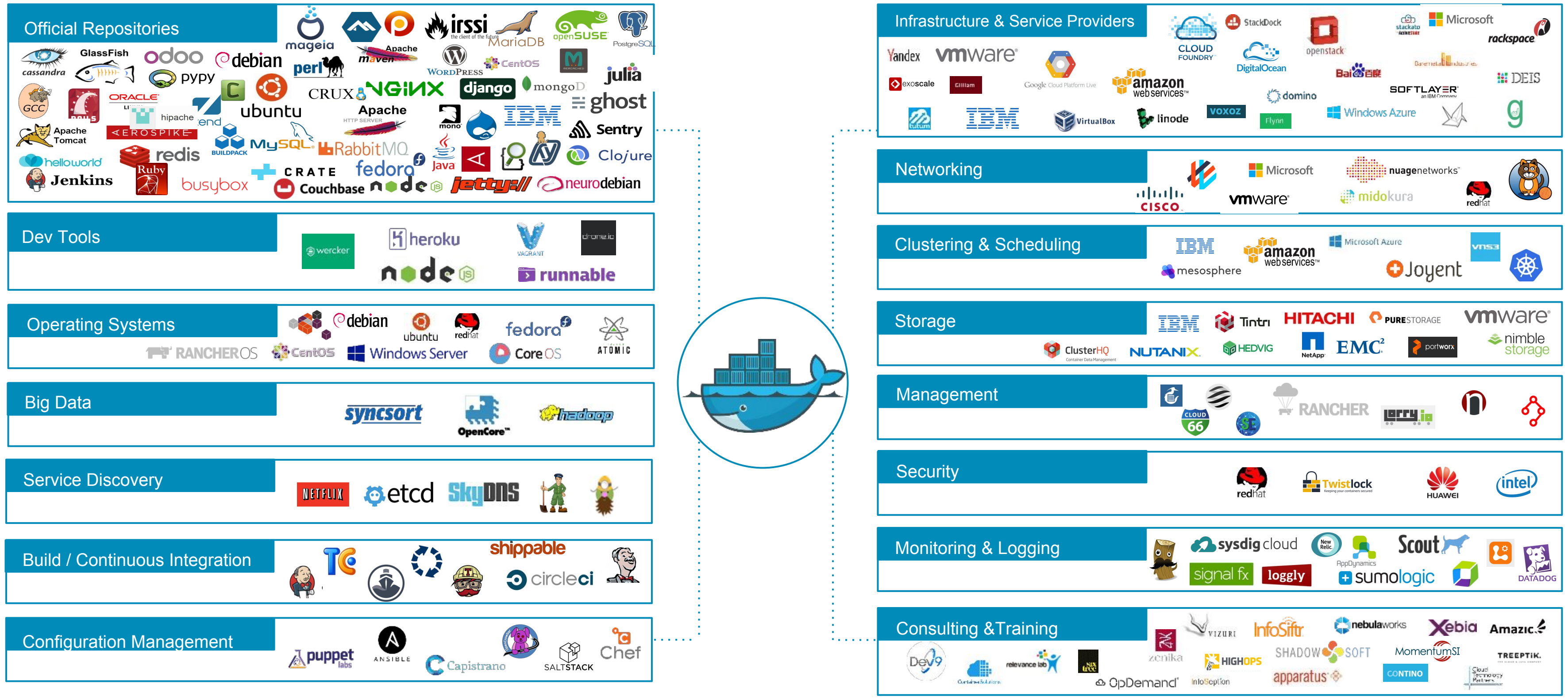
please donate

Scottish Charitable Incorporated Organisation
magic number SC045890

cyber-dojo foundation

built using Docker, Ruby, Rails, and Git.
hosted on Google Compute Engine.
Nadya Sivers drew the animals.
open sourced on github.

The Docker ecosystem

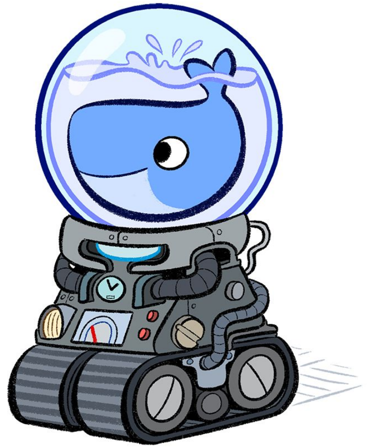


PRAQMA

Install docker now!

docker

http://docs.docker.com/



Sidetrack for those of us not on linux...

Docker toolbox is the simplest way to get started running containers on mac and windows systems

It uses virtualbox to create linux virtual machines for running containers

It can also be used to create docker environments on cloud providers such as amazon, google, and digitalocean

You will also need a working git

Are we there yet?

```
$   docker info
```

Are we there yet?

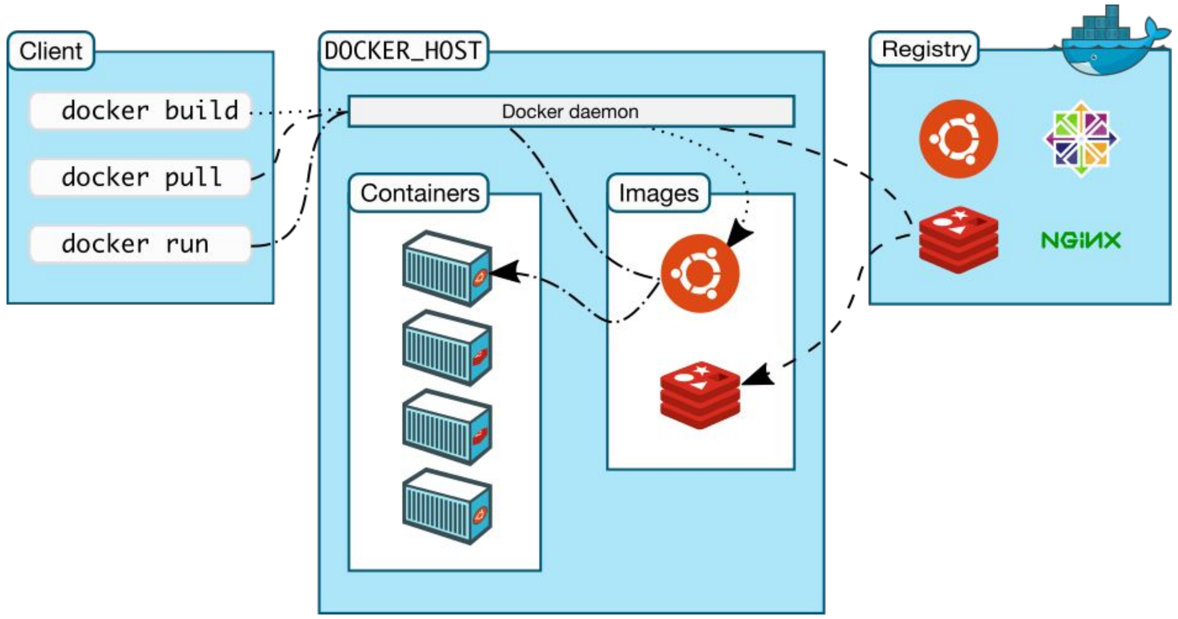
```
$   docker version
```

Let’s create some containers!

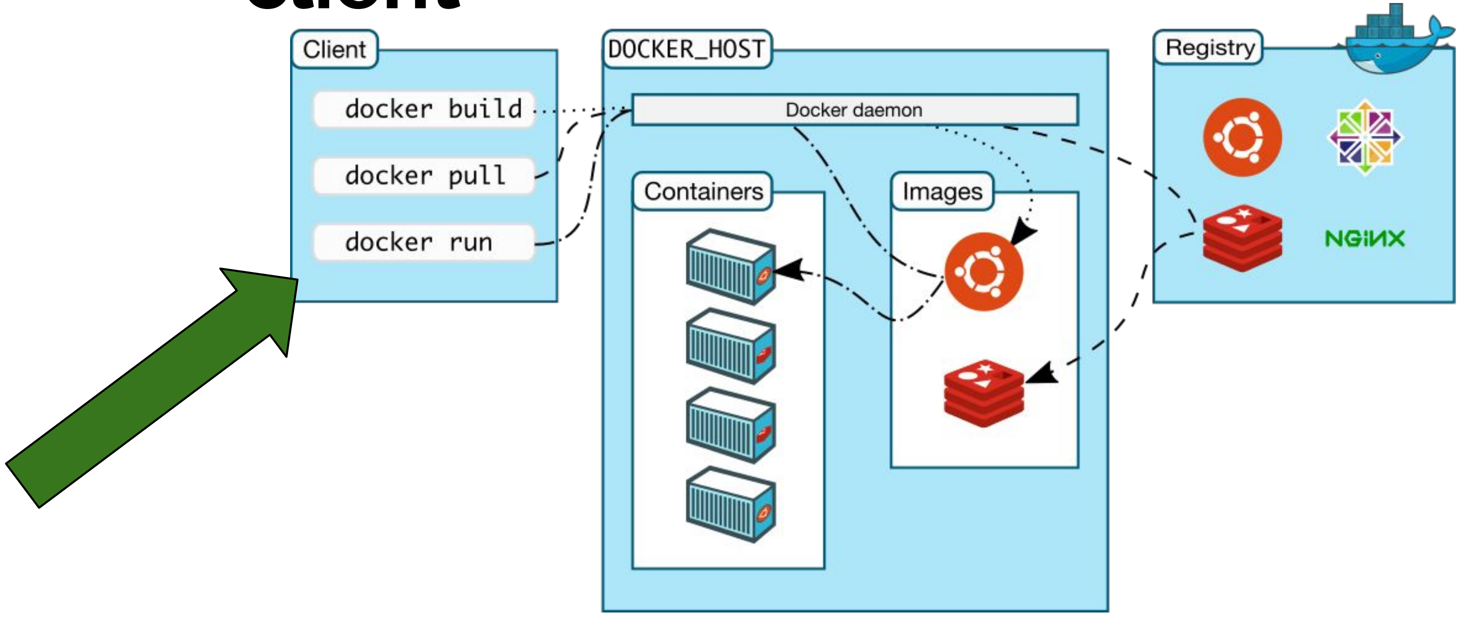
Hello, ACCU!

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

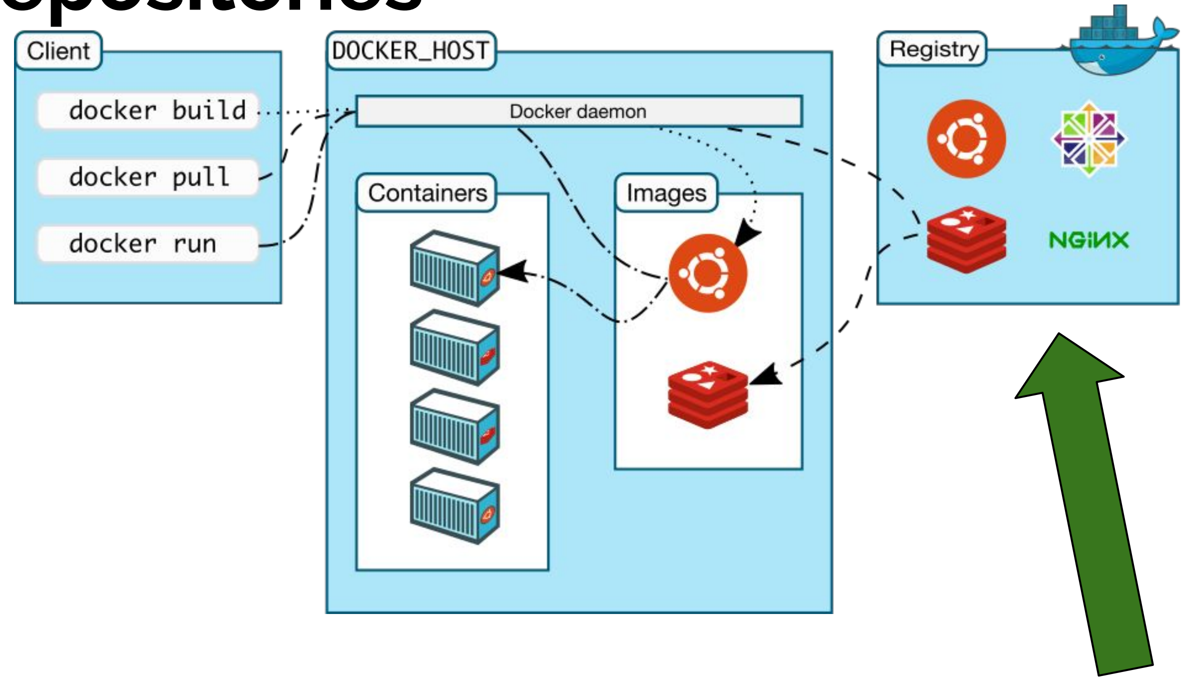
What just happened there then?



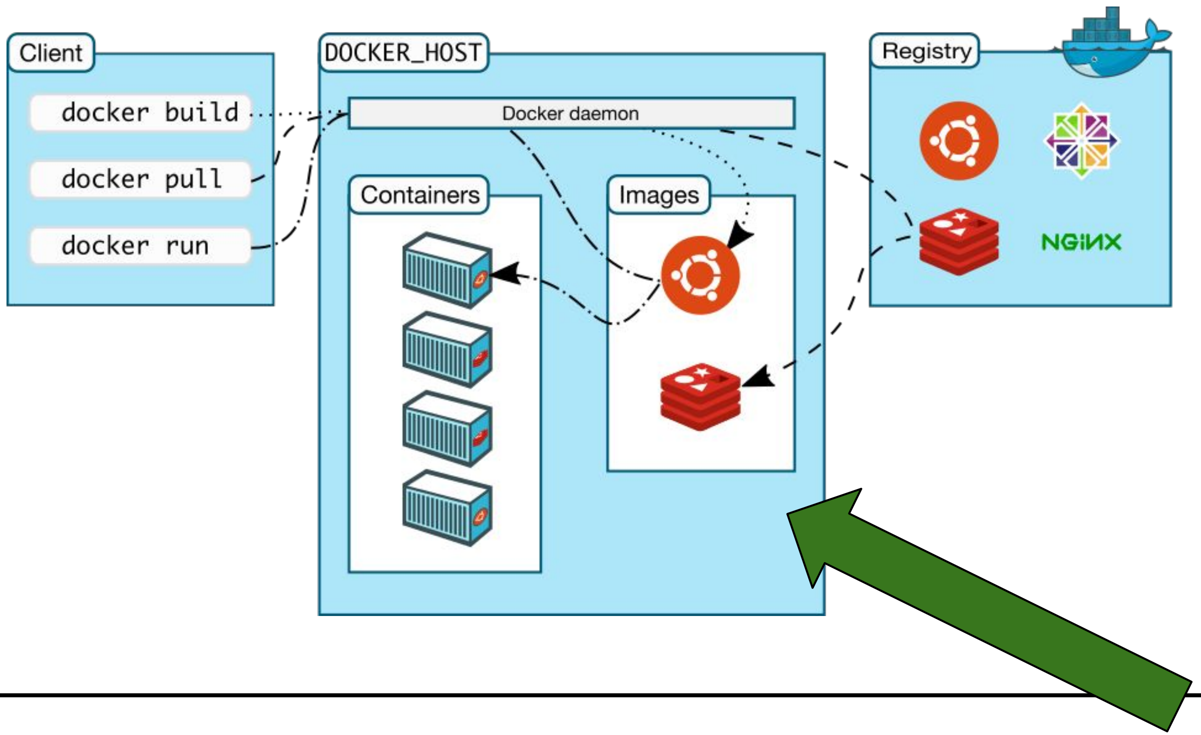
Commands are executed on the client



Images are pulled from repositories



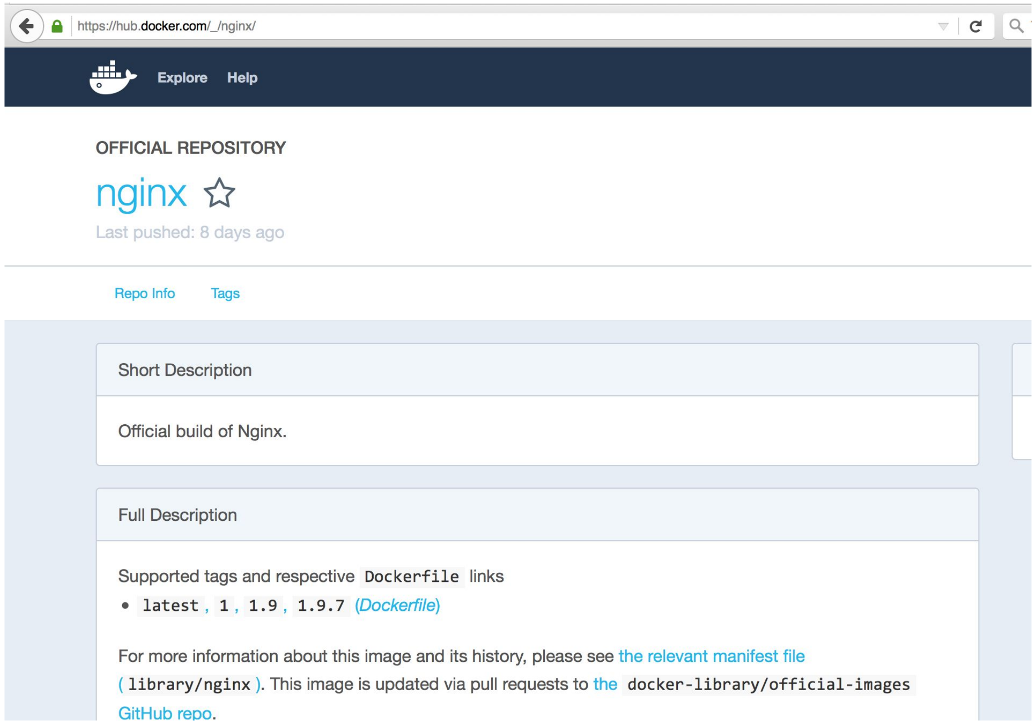
Containers are run from images



An container is...

- an isolated and secure application platform
- run, started, stopped, moved, and deleted
- created from a Docker image

Docker hub



OFFICIAL REPOSITORY

nginx

Last pushed: 8 days ago

Repo Info Tags

Short Description

Official build of Nginx.

Full Description

Supported tags and respective `Dockerfile` links

- `latest` , `1` , `1.9` , `1.9.7` *(Dockerfile)*

For more information about this image and its history, please see the relevant manifest file (`library/nginx`). This image is updated via pull requests to the `docker-library/official-images` GitHub repo.

Find out what images you have

```
# docker images
```

Docker will attempt to use local image first
Will look to hub if not found

Image Tags

`Images are specified by repository:tag`

`Default tag is latest`

Let’s saturate the network!

```
$ docker run ubuntu:14.04 echo “hello
world”
$ docker run ubuntu:14.04 ps aux
```

The second run should be faster because there is no download

Let’s run a container with a terminal

```
$ docker run -i -t ubuntu:14.04 /bin/bash
```

-i flag tells docker to connect to STDIN on the container
-t flag specifies to get a pseudo-terminal

Let’s add something to our container

```
$ apt-get update
$ apt-get install vim
$ vim test.txt
$ exit
```

Container processes

```
$ docker run ubuntu:14.04 echo “hello”
$ docker run -ti ubuntu:14.04 /bin/bash
root@1234dfs:/# ps -ef
CTRL + P + Q
$ ps -ef
```

A container only runs as long as it's process
Your command's process is always PID 1 in the container

Look at our running containers

```
$ docker ps -a
```

List running containers
Use the -a flag to include stopped containers
Containers have ID's and Names

Getting back in

```
$ docker attach <container-id>
```

Containers have ID's and Names
Either can be used

Use detached mode to run a container in the background

```
$ docker run -d ubuntu:14.04 ping 127.0.0.1 -c 50
```

Use `docker logs [containerID]` *to get the output*

`-f` *is a useful flag*

Time for a web server!

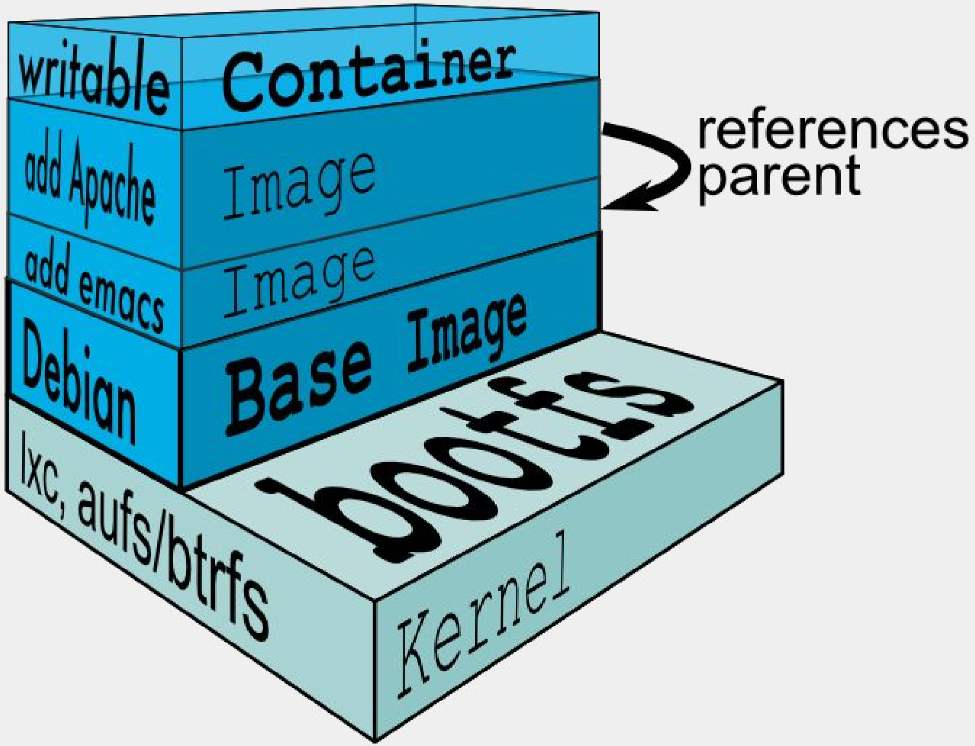
```
$ docker run -d -P nginx
```

Use `the public DNS of your AWS instance`

Use `docker ps` *to get the nginx port mapping*

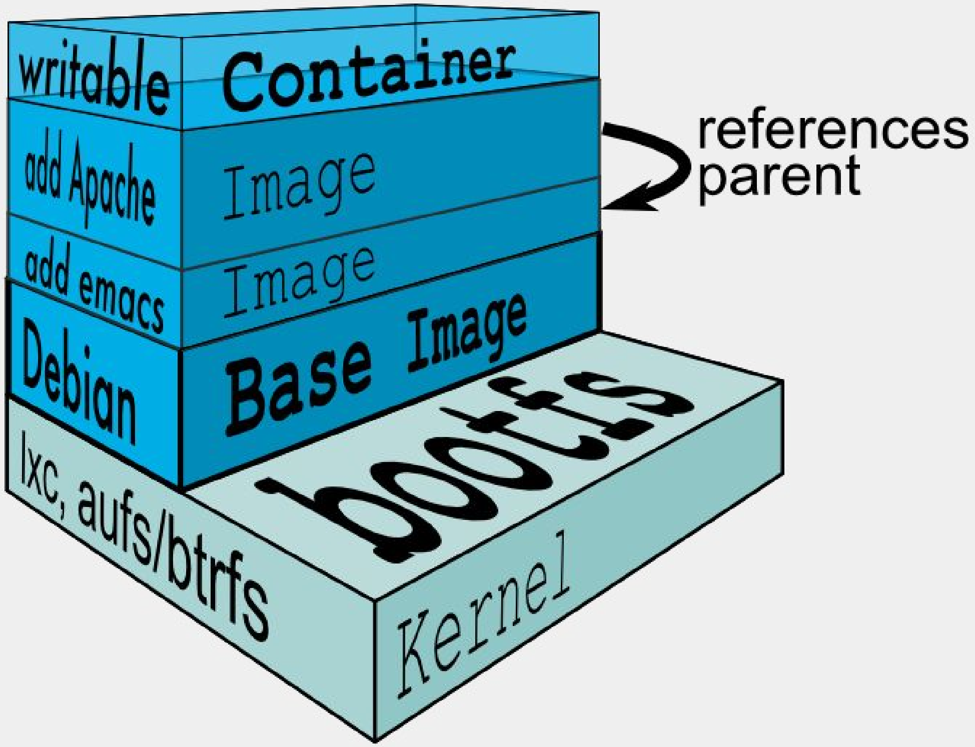
Images

An image is...

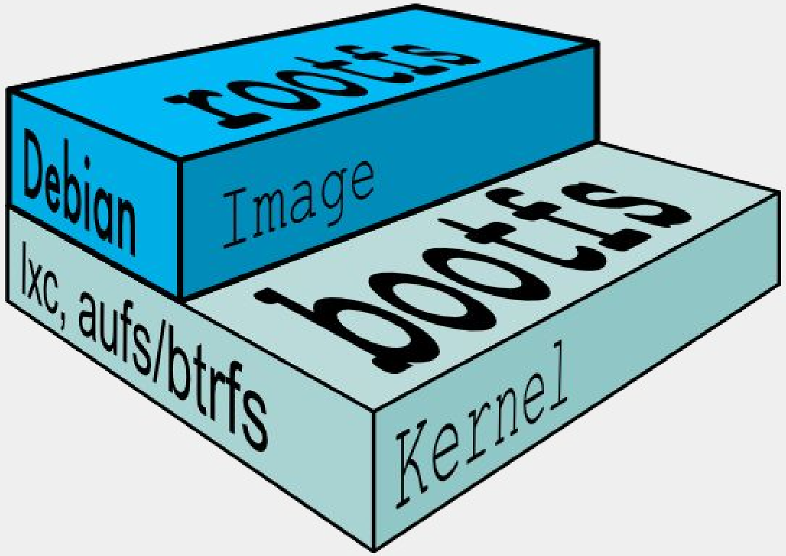


- A read-only template for creating containers
- The **build** component of docker
- Stored in registries
- Can be created by yourself distributed by others

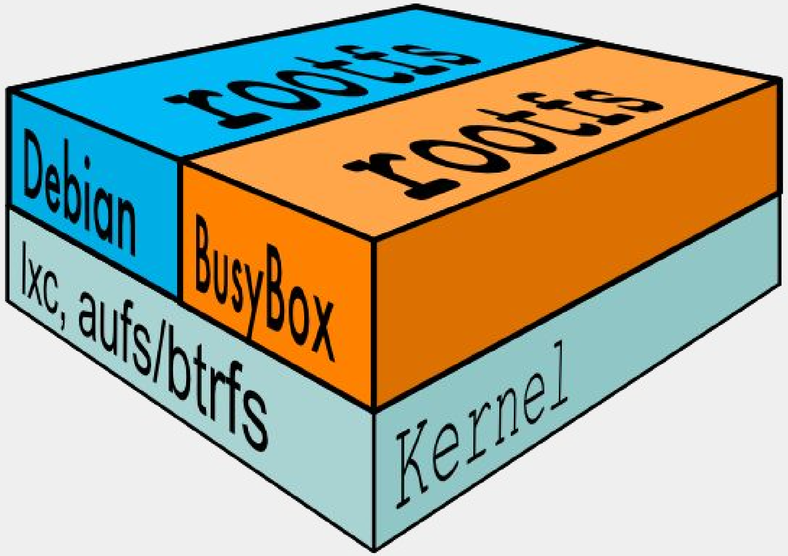
Images are layered read-only filesystems



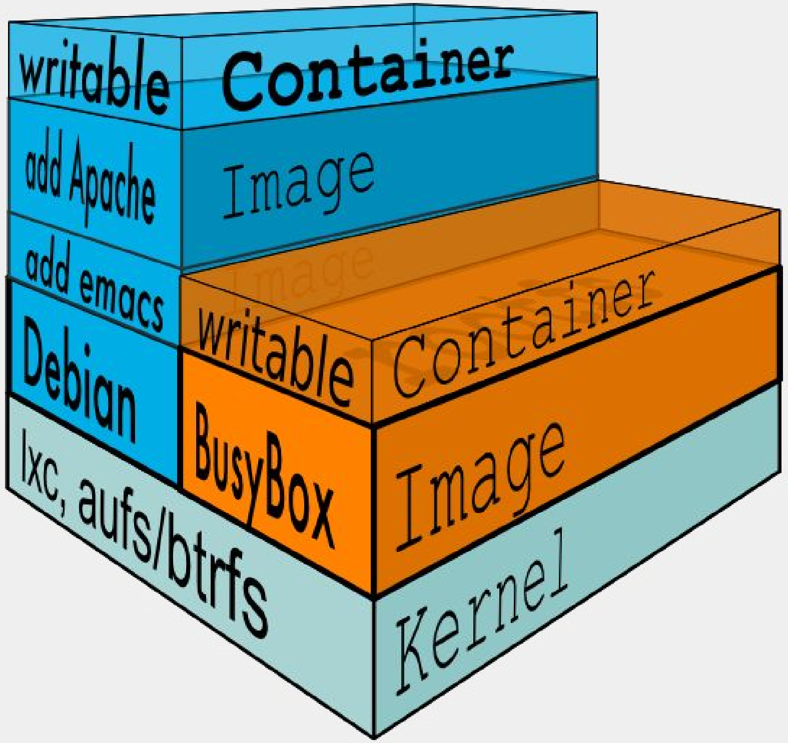
Images have base layers



Multiple root file systems per host are normal



When an image is run, a writable layer is added



Downloading an image with pull

```
$ docker pull busybox
```

Let’s make an image

Docker commit saves changes in a container as a new image

```
$ docker commit 234d3ea32 jkrag/simple:1.0
```

Let’s run our new image

```
$ docker run -ti jkrag/simple:1.0 bash
root@2343245:/# curl 127.0.0.1
```

The Dockerfile

The Dockerfile

A **Dockerfile** is a configuration file that allows us to specify instructions on how to build an image

It enables **configuration as code**

More **effective** than using `commit`

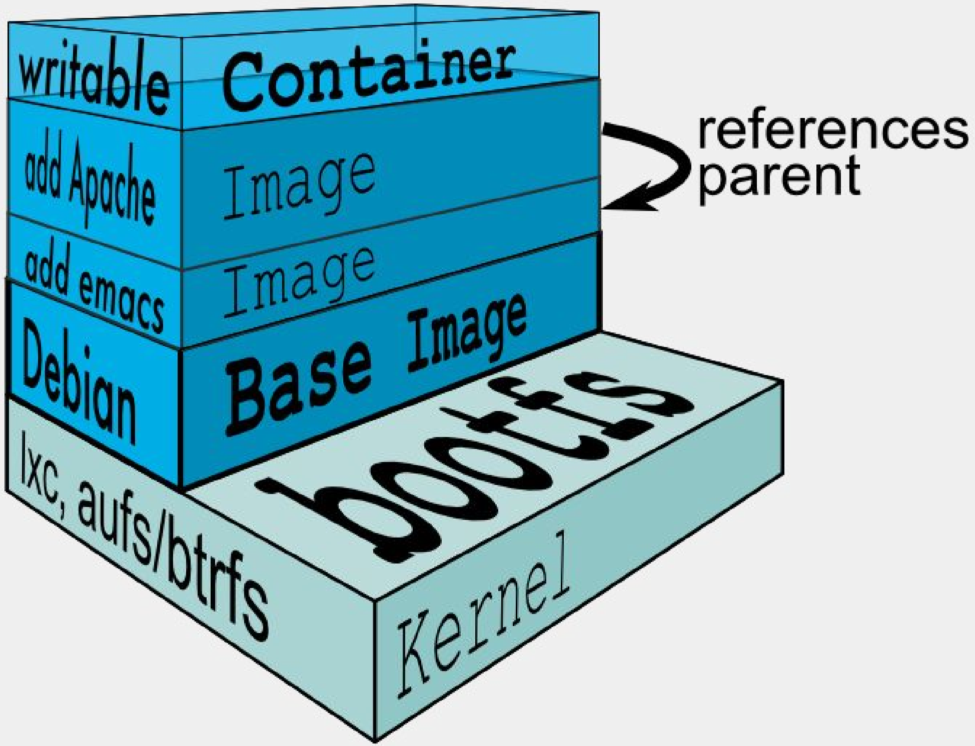
- Share the configuration rather than image
- Supports continuous integration
- Easier to review
- Easier to update

Dockerfile instructions

```
# Dockerfile for myapp
FROM ubuntu:14.04
RUN apt-get update
RUN apt-get install curl
RUN apt-get install vim
```

The default name for the file is `Dockerfile`

Run instructions are executed in the top writable layer



Aggregating RUN instructions to reduce layers

```
# Dockerfile for myapp
FROM ubuntu:14.04
RUN apt-get update && apt-get install -y \
  curl \
  vim
```

Building an image from a Dockerfile

```
$ docker build -t simple:1.1 .
```

The build command takes a build context on the filesystem

-f flag can be used to specify a different location for the Dockerfile

Go ahead and make your image

The CMD instruction

```
# Dockerfile for myapp
FROM ubuntu:14.04
RUN apt-get install curl
RUN apt-get install vim
CMD [“PING”, “127.0.0.1”, “-c”, “10”]
```

Can only be defined once
Can be overridden at run time

Run your new image with and without a command

The ENTRYPOINT instruction

```
# Dockerfile for myapp
FROM ubuntu:14.04
...
ENTRYPOINT [“PING”]
```

Cannot be overridden at run time
Can have a CMD in addition

Other notable Dockerfile commands

```
# Dockerfile for myapp
EXPOSE 80
ENV JAVA_HOME /usr/bin/java
COPY index.html /var/www
ADD robots.txt /var/www
```

Dockerfile best practices

Containers should be ephemeral

Use a `.dockerignore` file to exclude unnecessary files from the build context

Avoid including unnecessary packages and dependencies

Run only one process per container

Minimize the number of layers

Use the build cache to your advantage

Managing Containers

Other notable commands

```
$ docker run -d nginx
$ docker stop [CONTAINER_ID]
$ docker start [CONTAINER_ID]
```

Getting terminal access to a container

```
$ docker exec -it [CONTAINER_ID] bash
```

Removing containers

```
$ docker rm [CONTAINER_ID]
```

Can only remove stopped containers

Deleting images

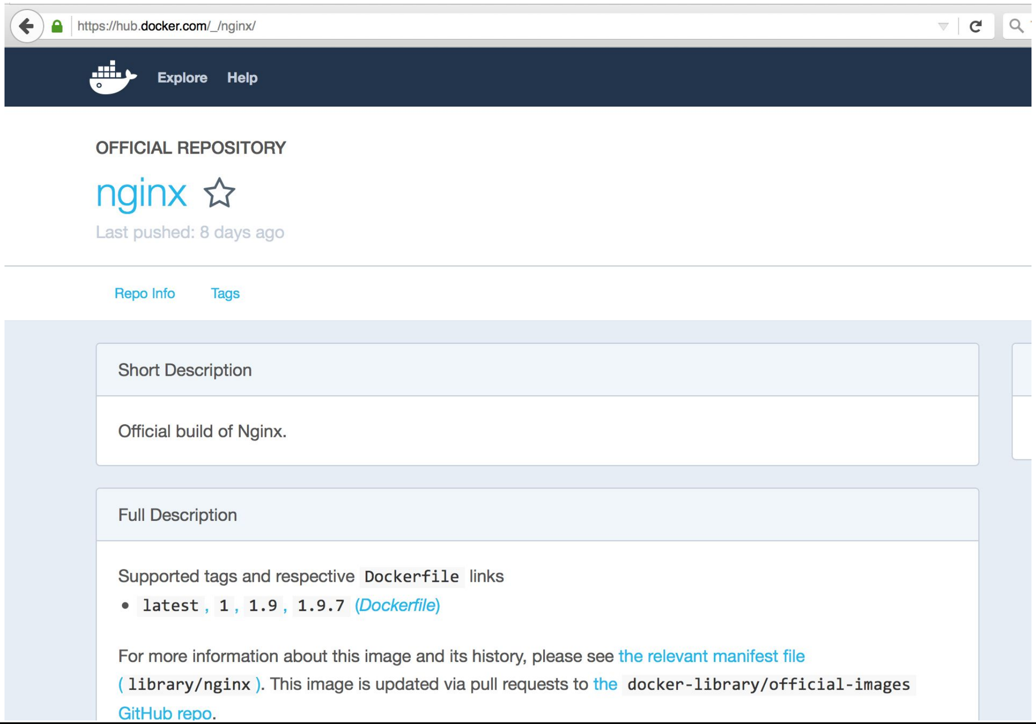
```
$ docker rmi jkrag/simple:1.0
```

Wipe em all out

```
$ docker rm -f $(docker ps -a -q)
```

Sharing containers

Let's add our repository on hub



OFFICIAL REPOSITORY

nginx

Last pushed: 8 days ago

Repo Info Tags

Short Description

Official build of Nginx.

Full Description

Supported tags and respective `Dockerfile` links

- `latest`, `1`, `1.9`, `1.9.7` *(Dockerfile)*

For more information about this image and its history, please see the relevant manifest file (`library/nginx`). This image is updated via pull requests to the `docker-library/official-images` GitHub repo.

Make a tag that matches our repository on hub

```
$ docker tag jkrag/simple:1.0 jkrag/aarhusdemo:1.0
```

Push to hub

```
$ docker push jkrag/aarhusdemo:1.0
```

Docker volumes

A volume is a directory in a container used for persistence

- Survive beyond the lifetime of a container
- Can be mapped to a host folder
- Can be shared amongst containers

A volume is a directory in a container used for persistence

```
$ docker run -d -P -v /tmp/myapp/html/:/www/website
nginx
$ docker exec -ti [ID] bash
$ ls /var/www/html
```

You can also add volumes in the Dockerfile

```
# create a volume
VOLUME /myvol

# multiple volumes
VOLUME /myvol1 /logs

# json syntax
VOLUME [“myvol1”,”myvol2”]
```

Volume best practices

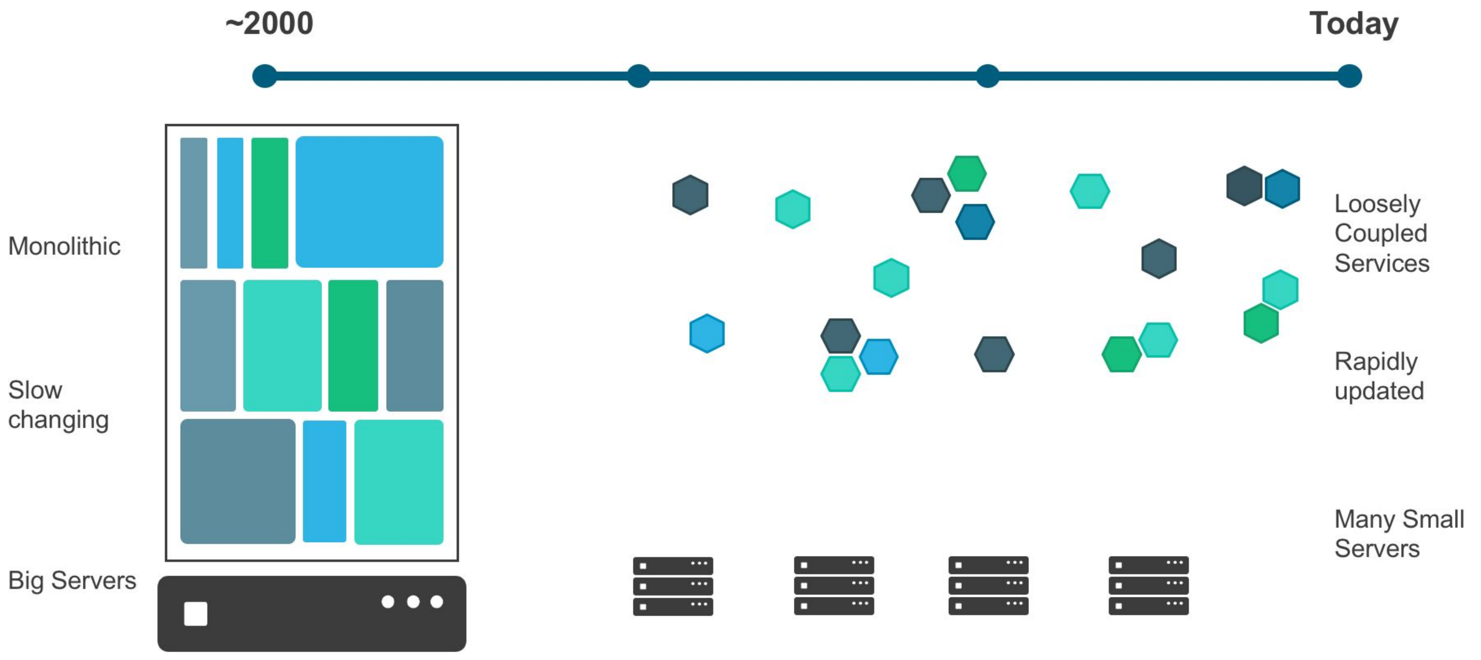
Containers should be ephemeral

Avoid mounting directories from the host in production

Data containers are recommended

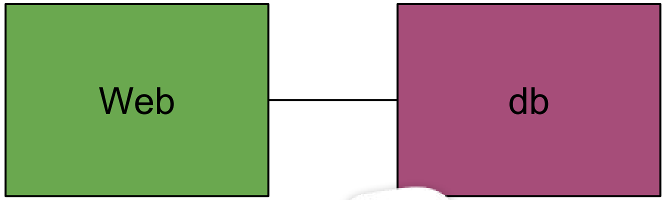
Docker compose

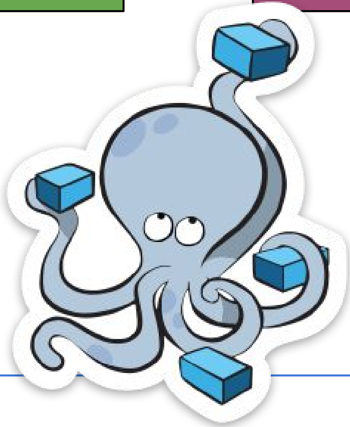
Transforming the Application Landscape



Using docker-compose to create multi-container apps

```
web:
  build: .
  ports:
   - "5000:5000"
  volumes:
   - .:/code
  links:
   - redis
redis:
  image: redis
```





Using docker-compose

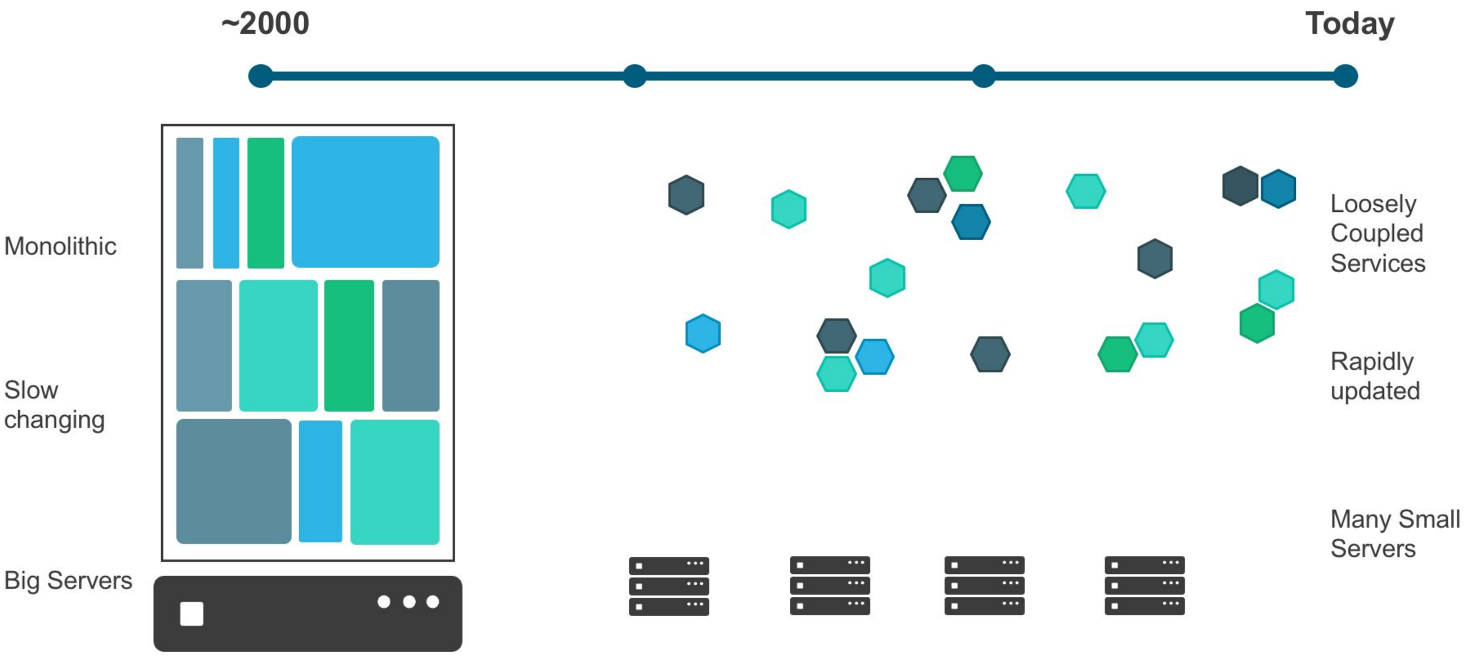
```
$ docker-compose up
$ docker-compose -d up
$ docker ps
$ docker-compose ps
$ docker-compose start <service name>
$ docker-compose stop <service name>
$ docker-compose rm <-v> <service name>
```

Using docker-compose continued...

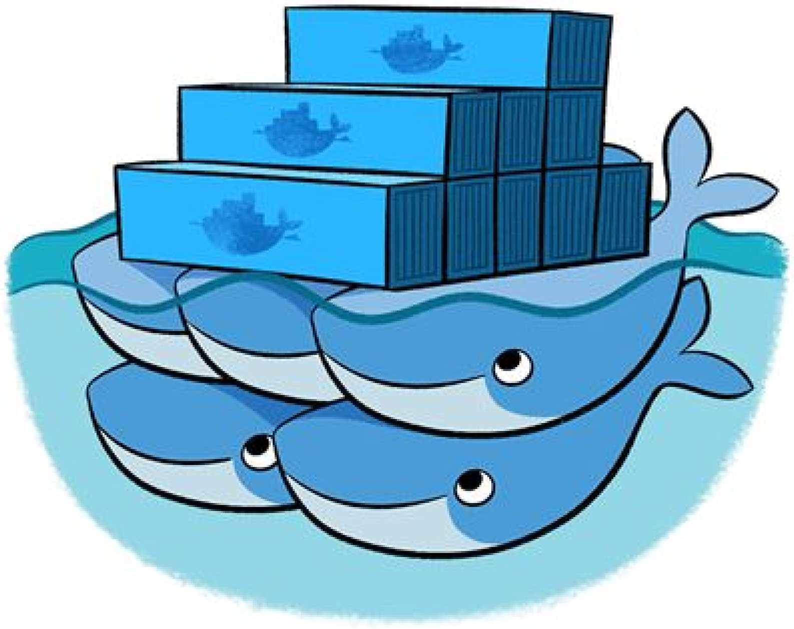
```
$ docker-compose logs
$ docker-compose scale
$ docker-compose -f compose-net.yml
--x-networking up -d
```

Multi-host applications

Transforming the Application Landscape



Using docker-swarm to create multi-host apps



Cluster technology for containers

Integrated networking and volumes

High availability options

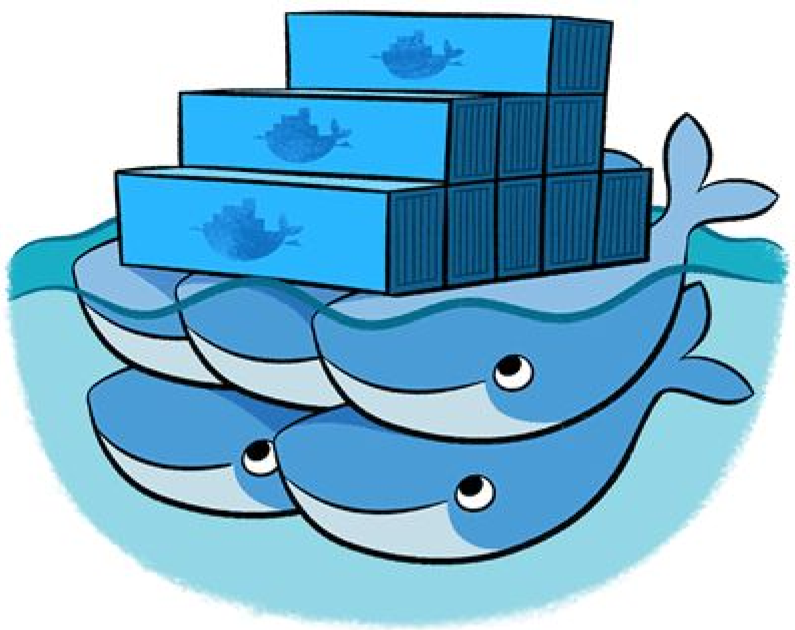
Pluggable schedulers and node discovery

Set up a docker-swarm using docker-machine

https://docs.docker.com/swarm/install-w-machine/

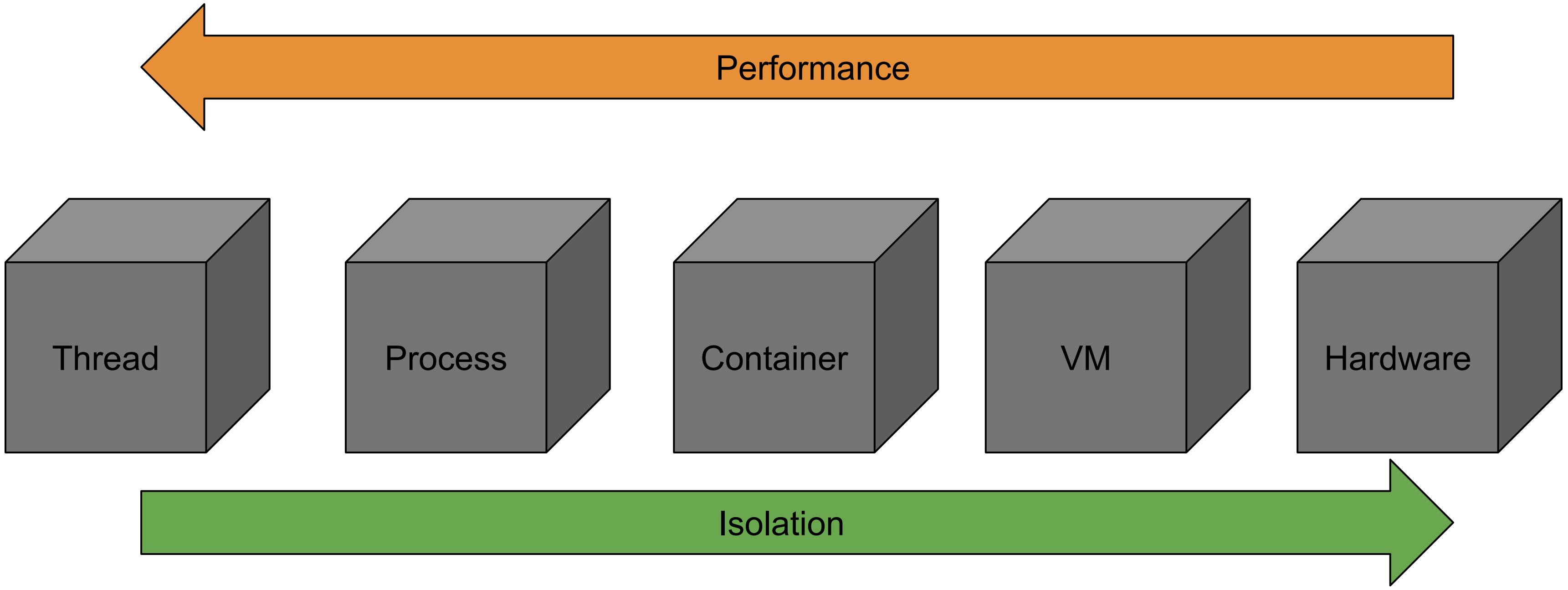
```
$ eval $(docker-machine env --swarm
swarm-master)
$ docker ps -a
```

A tour of swarm



https://docs.docker.com/swarm/install-w-machine/
https://gist.github.com/meekrosoft/f4f345331aaee2c917c44e78699c29ef

Where are we now?


Performance
Thread
Process
Container
VM
Hardware
Isolation

A brief tour of Docker

By the end of this workshop you will understand:

- What is a container and why you may want one
- How to create your own containers
- How to share your containers
- How to create multi-container applications
- How to create multi-host applications

Docker 101

Workshop