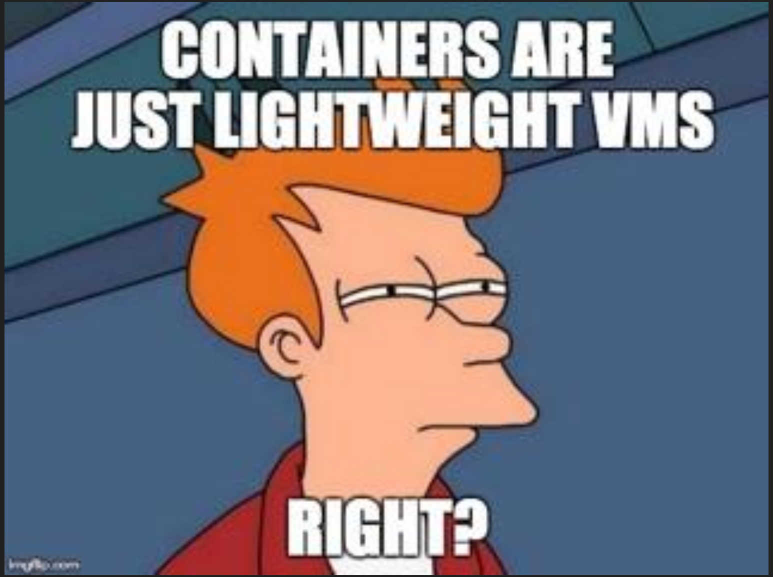
Docker
A Story of Containers
The Road to Orchestration

brad.marshall@gmail.com

IN THE FUTURE
EVERYTHING IS DOCKERIZED
imgflip.com

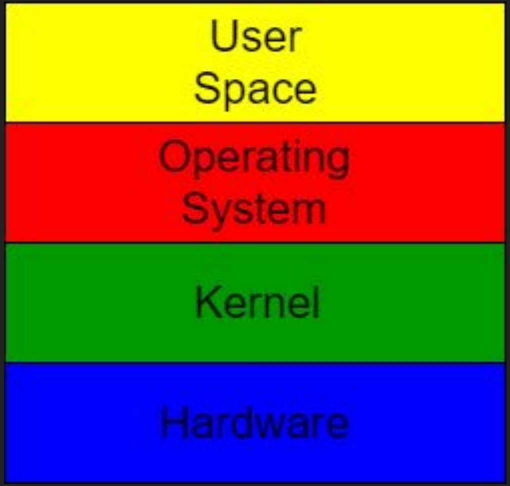
Some background

Before we start talking about containers need to first understand how a normal system works

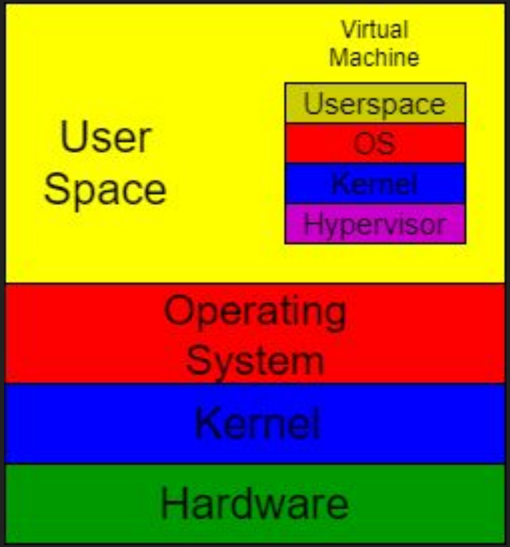


… well, no.

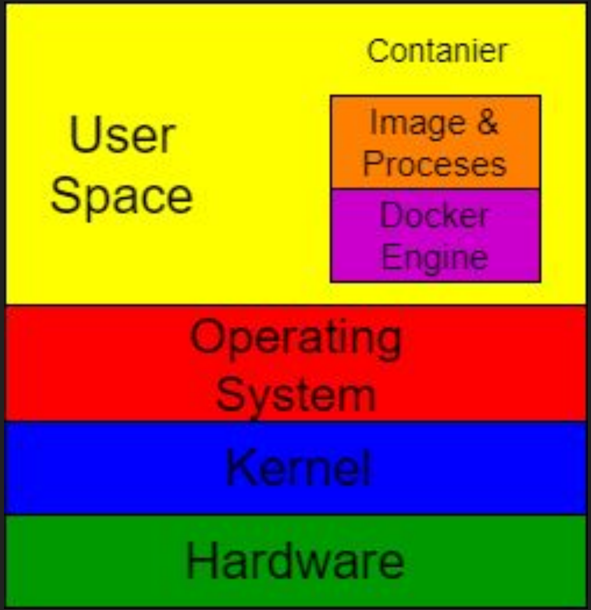
Standard System Architecture



Virtual Machine Architecture



Container Architecture



MIGRATING CONTAINERS TO CLOUD
imgflip.com

What are containers

- Similar to VMs in design
- Isolate and container an application to a self contained unit that can run where you want
- Remove the dependency on physical hardware
- Both have private space to run processes, own IP address, run things as root
- Biggest difference is that containers share the host kernel
  - Means you have to run the same operating system - no Windows running on Linux
- Containers are much lighter weight that VMs
- Makes it easy to build an application on laptop and move to production very simply

IT WORKS ON MY MACHINE
THEN WE'LL SHIP YOUR MACHINE
AND THAT IS HOW DOCKER WAS BORN

Container technical details

- Namespaces + Cgroups + UnionFS
- Namespaces are like chroot, but for network config, firewall rules, processes, mounts, IPC etc - can be shared among processes
- Cgroups are control groups - limits resource allocation to processes
- UnionFS allows images to be build in layers
- Doesn’t need hardware emulation or cpu flags exposed
- Each container runs its own process

System containers

- Process is one that could serve as init process on the host
- Normally systemd, upstart or SysVinit
- Spawns subprocesses like sshd etc
- Usually run in a user namespace, so root process in container is user process on host
- Example is LXC/LXD

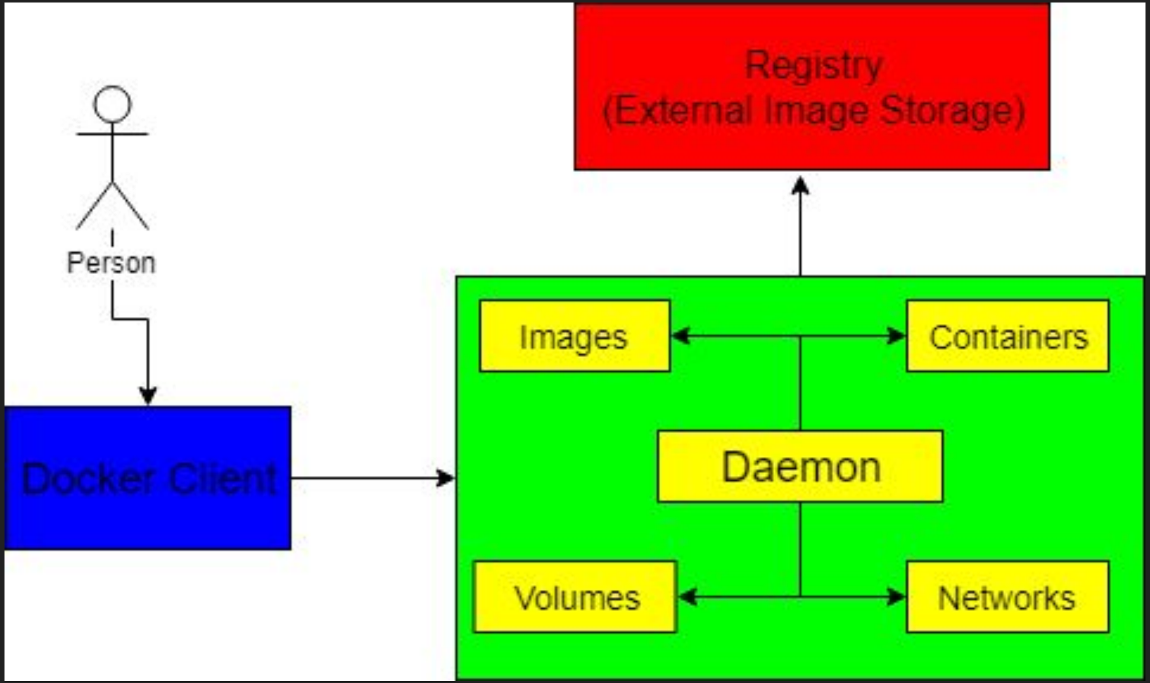
Application containers

- Can run any process
- Uses fewer resources as it runs less than a system container
- Has its own private filesystem, network stack etc
  - Completely isolated from other containers
- Own filesystem means own copy of libraries and dependencies
  - Both good and bad - means you have to maintain multiple versions of things
  - Obvious security implications here - have to know every version of things you're running
- Examples are:
  - Docker - oldest one
  - Podman - compatible with Docker, but no mgmt daemon
  - rkt

Docker

- 3 main parts
  - Command line
  - REST API
  - Daemon to manage containers / images / networks
- Images from registry, including Docker Hub
- Needs volumes mounted for persistent data
  - Directories mounted from host system into container is simplest
- Applications are usually exposed by mapping ports from host system

Docker Architecture



SAY DOCKER
ONE MORE TIME

Installing Docker

Ubuntu

- apt install docker.io

RedHat variants

- yum install docker

Or you can use upstream version - see https://docs.docker.com

Running Docker containers

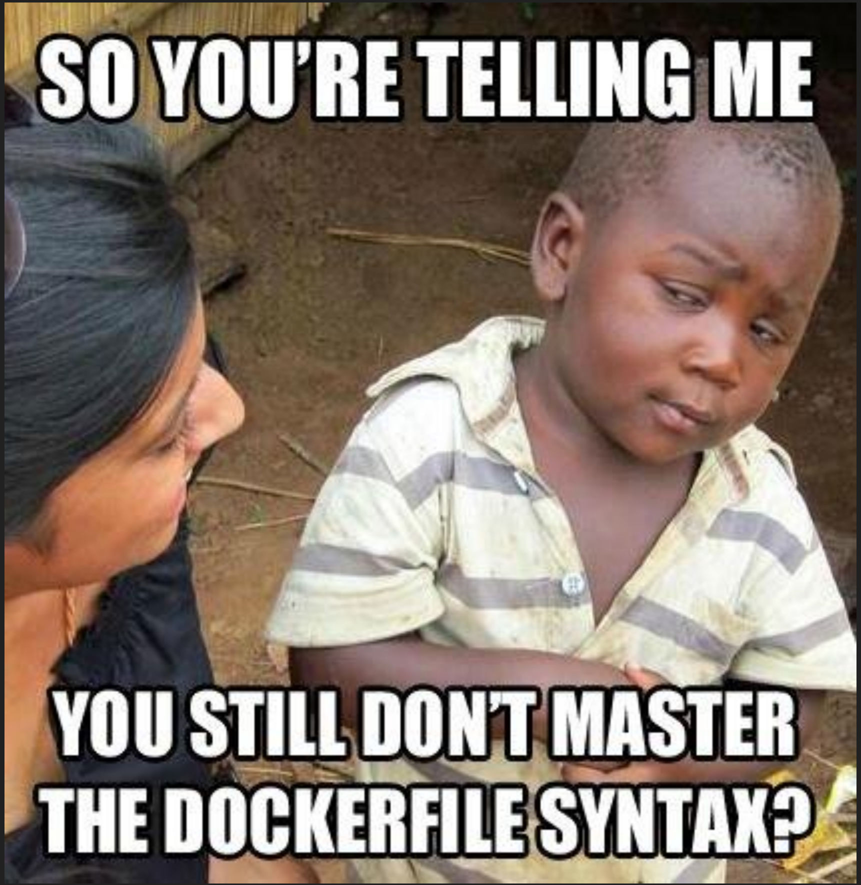
```
$ sudo docker run -d -p 8080:80 --name httpd httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
1ab2bdfe9778: Pull complete
174a8e3bca83: Pull complete
c8e4c9e94892: Pull complete
4568916ecf2d: Pull complete
533f5cf513cb: Pull complete
Digest: sha256:98caed3e3a90ed9db8d25dcbb98eebe0ce56358a9dbbc940d7eb66a8e2b88252
Status: Downloaded newer image for httpd:latest
B33229ad604998f9e1d50104a368e60ff646562b8d18302245e078fe4eec3b7d

$ curl http://localhost:8080
<html><body><h1>It works!</h1></body></html>
```

Docker Status

```
$ sudo docker ps
CONTAINER ID        IMAGE                    COMMAND               CREATED             STATUS
PORTS                    NAMES
b33229ad6049        httpd                    "httpd-foreground"   7 minutes ago       Up 7
minutes        0.0.0.0:8080->80/tcp   httpd

$ sudo docker images
REPOSITORY          TAG                      IMAGE ID            CREATED             SIZE
httpd               latest                   7d85cc3b2d80        2 weeks ago         154MB
```


SO YOU'RE TELLING ME
YOU STILL DON'T MASTER THE DOCKERFILE SYNTAX?

Building Docker containers

Dockerfile:

```
FROM ubuntu:18.04

RUN apt-get update
RUN apt-get install -y nginx

CMD [“/usr/sbin/nginx”, “-g”, “daemon off;”]
```

Build Docker containers cont

```
$ docker build -t bradm/httpd .

$ docker run -d -p 8080:80 -d --name brad-httpd bradm/httpd
```

Docker Persistent Data

- Bind mounts
  - Create directory on docker host and mount into container

```
docker run -it --name ubuntu --mount type=bind,source=/mnt/srv,target=/srv

 ubuntu:18.04
```

- Docker volume
  - Volume managed by docker mounted into container

```
docker run -it --name ubuntu --mount source=mysrv,target=/srv ubuntu:18.04
```

Docker Volumes

- See the volumes

```
$ docker volume ls
```

- See details on a specific volume

```
$ docker volume inspect mysrv
```

- Create a new volume

```
$ docker volume create --label mytest
```

docker-compose up
I'M SUCH A HACKER!

Docker-compose

- Used to define multiple containers in a project
  - Example is Wordpress with reverse proxy frontend and db
- Defines multiple containers and relationships between them
- Sets up volumes to store persistent data
- Handles port mapping from external ports to docker ports
- Only restarts changed containers
- Variables allow usage between dev/test/prod etc
- Defined in a yaml file

Using Docker-compose

- Define the containers, relationships, ports, volumes in the yaml file

```
# Bring the services up

$ docker-compose up -d

# Shut the services down and delete the volumes

$ docker-compose down -v
```

Wordpress docker-compose example

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: rootwp
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress
```

Wordpress docker-compose example cont

```
   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
     volumes:
       - wordpress:/var/www/html
volumes:
    db_data: {}
    wordpress: {}
```

Demo time!

References

- https://linuxcontainers.org/
- https://hub.docker.com
- https://www.portainer.io/
- https://podman.io/
- https://coreos.com/rkt/
- https://rancher.com/