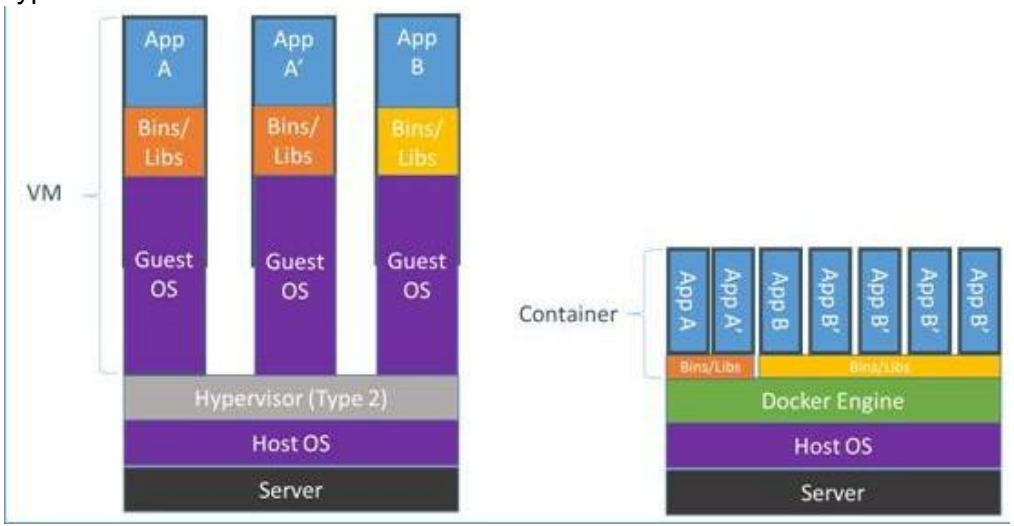
Docker for fun and profit

Solomon Hykes* about Docker:
"It uses Linux containers and the Internet won't shut up about it."
(LinuxCon 2014 keynote)

*Founder of dotcloud and creator of the Docker project

What are Linux containers or containers in general?

Hypervisor vs Containers



Hypervisors are based on Emulating Virtual Hardware

- emulate virtual hardware and BIOS, run full OS
- every instance is totally separate
- size several GB

Containers are based on Sharing the Operating System

- instance shares the kernel (limited to running only Linux container on Linux host)
- in theory the container can share everything or almost nothing with the host
- start time
- size can be couple of MB
- application container

Containers and Linux

2005 OpenVZ - first open source container technology (out of the Linux kernel source tree)

2006 Process Containers (CGroups)

2007 Google use CGroups to containerise search (Googleplex went pretty much fully containerized)

2008 LXC version 0.1.0 released

2011 Container Unification agreement on fringes of Kernel Summit
- agreed there would be one container technology in Linux
- work began on Container Unification at Kernel API level
- CGroups and Namespaces now agreed API (in Kernel source tree)
- only one underlying kernel technology for containers which is used by everybody (OpenVZ, LXC, Docker, ZeroVM….)

2013 First Linux Kernel Supporting OpenVZ with no patches (3.12) released

Containers and Linux

Namespaces isolate processes.

CGroups control resources.

There are 12 CGroups and 6 Namespaces in the kernel.

Containers can use all of these or any combination.

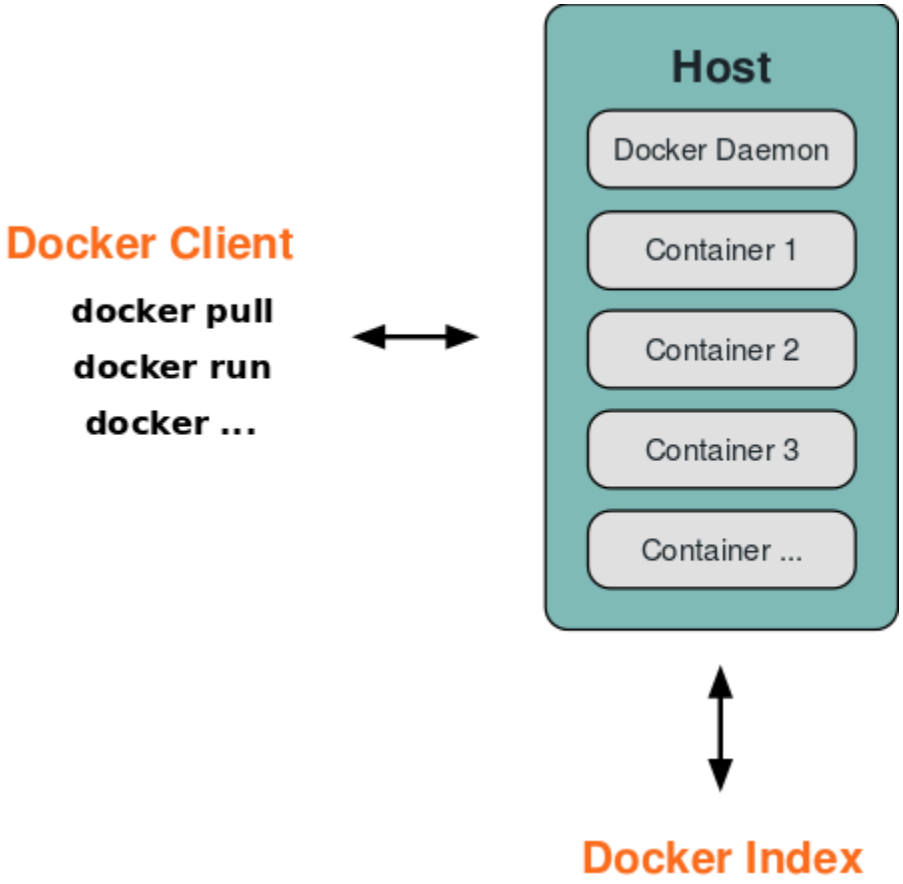
Container security:
As part of the agreement from 2011, User Namespaces became the container security mechanism.
From 2014 Distributions begin enabling User Namespaces.

What is Docker?

Docker is a tool that uses containers to create lightweight packages for applications with instant portability.

Docker components

- The Docker client and server
- Docker Images
- Registries
- Docker Containers



The Docker client and server

Docker client → Docker daemon/server → libcontainer → Host OS kernel

Docker client does not have to be on the same server.

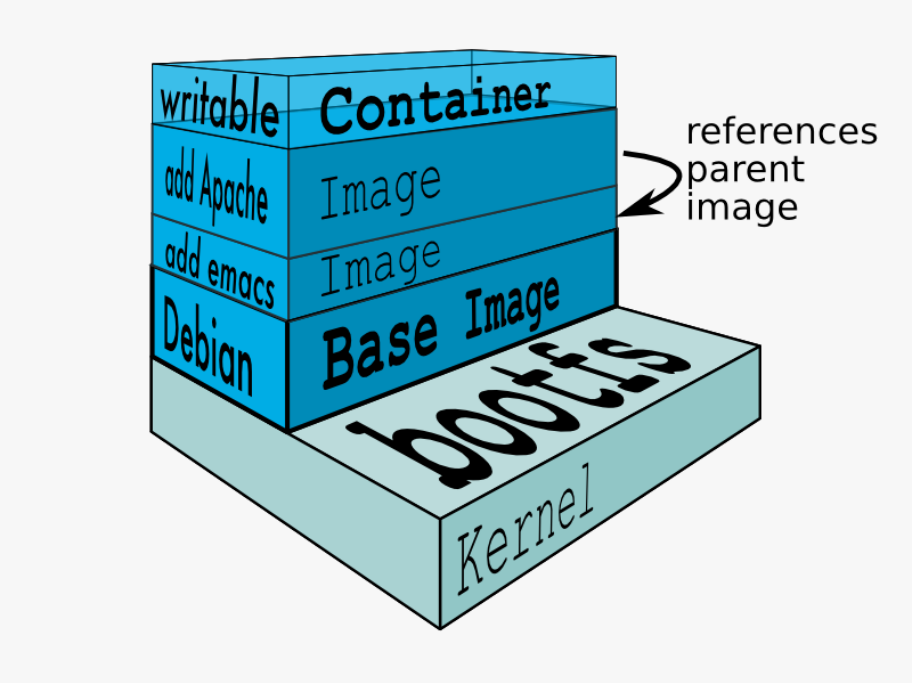
kernel >= 3.8
memory and swap accounting (optional)

Docker Images

Docker images are basically a read-only template out of which docker containers are instantiated.

Images contain all the information on a certain type of container.

These images can either be defined by Dockerfiles or by commiting a container. When trying to run a container, docker will automatically download the image you specified.



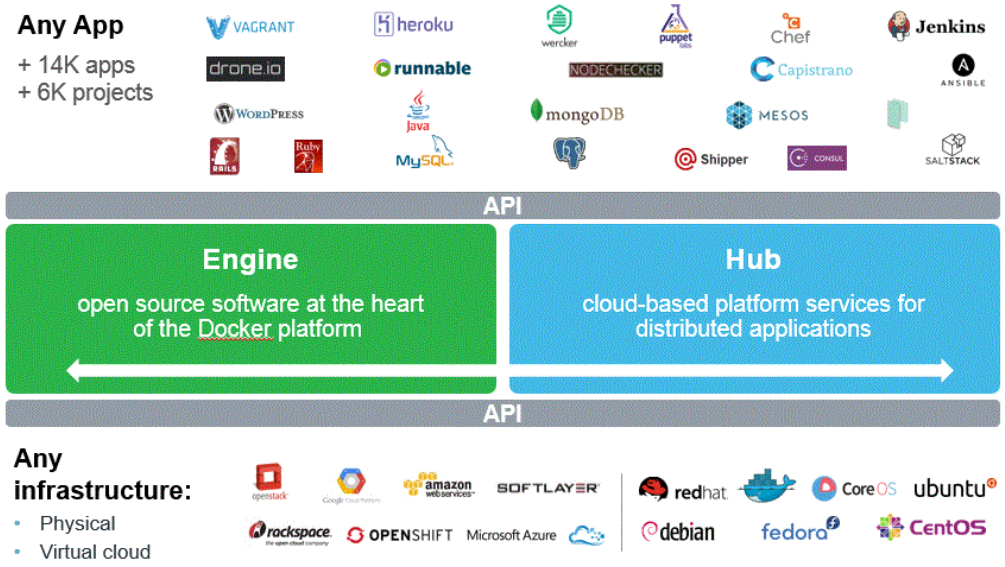
Registries

Docker stores the images you build in registries.

Two types of registries: public and private.

Docker, Inc., operates the public registry for images, called the Docker Hub. You can create an account on the Docker Hub and use it to share and store your own images.

The Docker Hub also contains over 14,000 images that other people
have built and shared.



Docker Containers

Containers are launched from images and can contain one or more running processes. You can think about images as the building or packing aspect of Docker and the containers as the running or execution aspect of Docker.

A Docker container is:
- An image format.
- A set of standard operations.
- An execution environment.

The Docker container captures the exact configuration of a version of an application. To upgrade the application in production, the container is usually replaced with a new version, which takes a few seconds. The layers of components that go into the configuration are kept separate and can be inspected and rebuilt easily.

When to use Docker?

- use it as version control system for your entire app's operating system
- when you want to distribute/collaborate on your app's operating system with a team
- use it to run your code on your laptop in the same environment as you have on your server
- whenever your app needs to go through multiple phases of development (dev/test/qa/prod)
- use it with your configuration management tool of choice

Build once, run anywhere! ~ developer
Configure once, run anything! ~ operations

Fun with Docker!

- containerize everything
- tools
- workflows and techniques

Docker Skype

Run Skype in a docker container and avoid cluttering the host with multi-arch setup.

Build the container's image directly from github (https://github.com/shofetim/docker-skype)

docker build --rm=true -t DESIRED_NAME https://raw.github.com/shofetim/docker-skype/master/Dockerfile

1. Run container docker run -d -P DESIRED_NAME
2. Get the port that SSH is nat'ed to. docker ps
3. Get the IP address. ifconfig
4. Run skype with X forwarded to your normal desktop ssh -X docker@IP -p PORT skype password will be docker.

Docker Streisand

Streisand sets up a new server running L2TP/IPsec, OpenSSH, OpenVPN, Shadowsocks, sslh, Stunnel, and a Tor bridge. It also generates custom configuration instructions for all of these services. At the end of the run you are given an HTML file with instructions that can be shared with friends, family members, and fellow activists.

https://github.com/gdoteof/docker-streisand

Trivia: The Streisand effect is the phenomenon whereby an attempt to hide, remove, or censor a piece of information has the unintended consequence of publicizing the information more widely, usually facilitated by the Internet.

Dockersh

Designed to be used as a login shell on machines with multiple interactive users.

When a user invokes dockersh, it will bring up a Docker container (if not already running), and then spawn a new interactive shell in the container's namespace.

dockersh can be used as a shell in /etc/passwd or as an ssh ForceCommand.

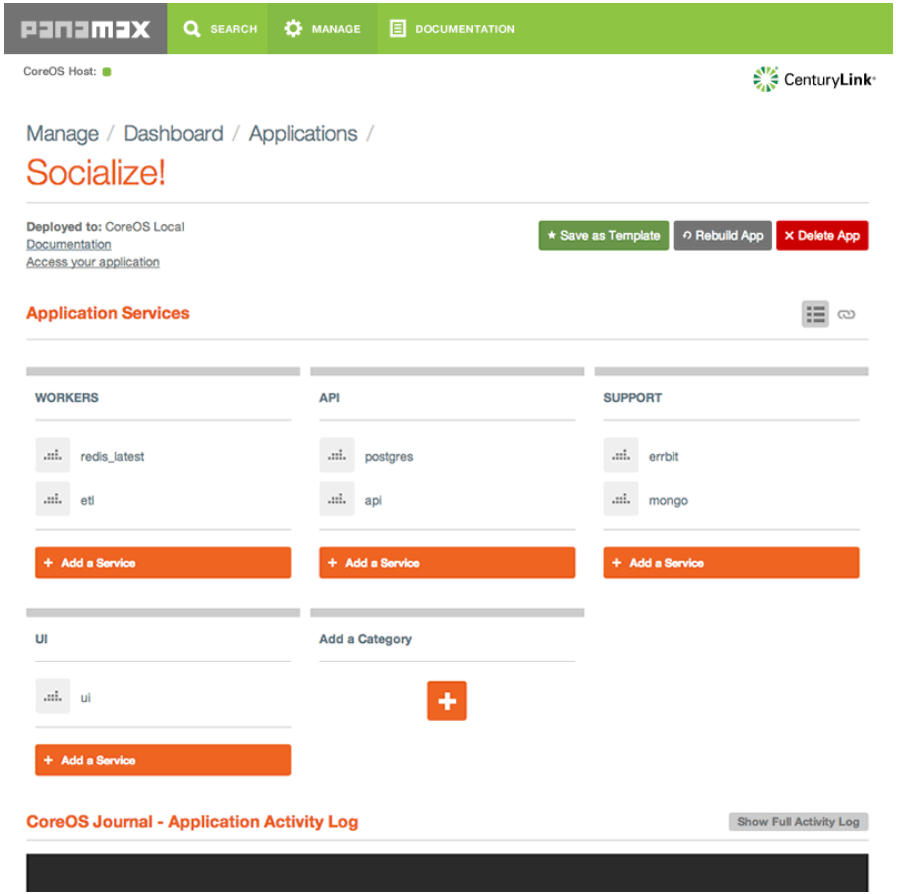
This means that the user is isolated from the rest of the system, and they can only see their own processes, and have their own network stack. This gives better privacy between users, and can also be used for more easily separating each user's processes from the rest of the system with per user constraints.

https://github.com/Yelp/dockersh

Panamax

Panamax is a containerized app creator with an open-source app marketplace hosted in GitHub. Panamax provides a friendly interface for users of Docker, Fleet & CoreOS. With Panamax, you can easily create, share and deploy any containerized app no matter how complex it might be.

http://panamax.io/



Workflows and techniques

Workflow 1

Develop inside a single running container as you would in a single VM.

Start a shell in container:
docker run -i -t ubuntu /bin/bash -v /path/to/code:/src

To use a container as a full Development Environment use phusion/baseimage:<VERSION>
https://registry.hub.docker.com/u/phusion/baseimage/

Workflow 2

Leverage containers, modularise

Workflows and techniques

Embrace Reusability in Dockerfiles.
Write general requirements early, commit and name relevant checkpoints, leave customisations last.

Add + build routine
docker add <src> <dest>
The ADD instruction copies new files from host's <src> to container's <dest>

1. Update code in local app folder (git pull?)
2. docker build your image with updated code
3. Distribute and profit!


Questions?

Thank you!