

A Seminar Presentation On

Introduction to Docker



Virendra singh ruhela

Introduction to Docker



docker



DOCKER HISTORY

- A dotCloud (PAAS provider) project
- Initial commit January 18, 2013
- Docker 0.1.0 released March 25, 2013
- 18,600+ github stars, 3800+ forks, 740 Contributors.... and continues
- dotCloud pivots to docker inc. October 29, 2013



What is Docker?!!!

- Open platform for developers and sysadmins to build, ship and run distributed applications
- Can run on popular 64-bit Linux distributions with kernel 3.8 or later
- Supported by several cloud platforms including Amazon EC2, Google Compute Engine, and Rackspace.



Features....

- Light-Weight
 - Minimal overhead (*cpu & io & net & disk*)
 - Based on Linux containers
 - Uses layered filesystem to save space (AUFS/LVM)
 - Uses a copy-on-write filesystem to track changes
- Portable
 - Can run on any Linux system that supports LXC (today).
 - 0.7 release includes support for RedHat/Fedora family.
 - Raspberry pi support.
 - Future plans to support other container tools (lxcftfy, etc.)
 - Support for other operating systems (Solaris, OSX, Windows?)
- Self-sufficient
 - A Docker container contains everything it needs to run
 - Minimal Base OS
 - Libraries and frameworks
 - Application code
 - A docker container should be able to run anywhere that Docker can run.



The Challenge.....

Multiplicity of Stacks

 Static website
nginx 1.5 + modsecurity + openssl + bootstrap 2

 Background workers
Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs +
phantomjs

 User DB
postgresql + pgv8 + v8

 Web frontend
Ruby + Rails + sass + Unicorn

 Queue
Redis + redis-sentinel

 Analytics DB
hadoop + hive + thrift + OpenUDK

 API endpoint
Python 2.7 + Flask + pyredis + celery + pycppg + postgresql-client

Do services and apps
interact
appropriately?

Multiplicity of
hardware
environments

 Development VM
 QA server

Customer Data Center



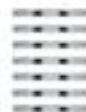
Public Cloud

Disaster recovery

Production Servers



Production Cluster




Contributor's laptop



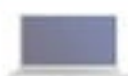
Can I migrate
smoothly and
quickly?



The Matrix From Hell.....



Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?
	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers



Cargo Transport Pre-1960.....

Multiplicity of Goods



Do I worry about
how goods interact
(e.g. coffee beans
next to spices)








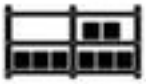





Multiplicity of
methods for
transporting/storing



Can I transport quickly
and smoothly
(e.g. from boat to train
to truck)



Also a Matrix from Hell.....

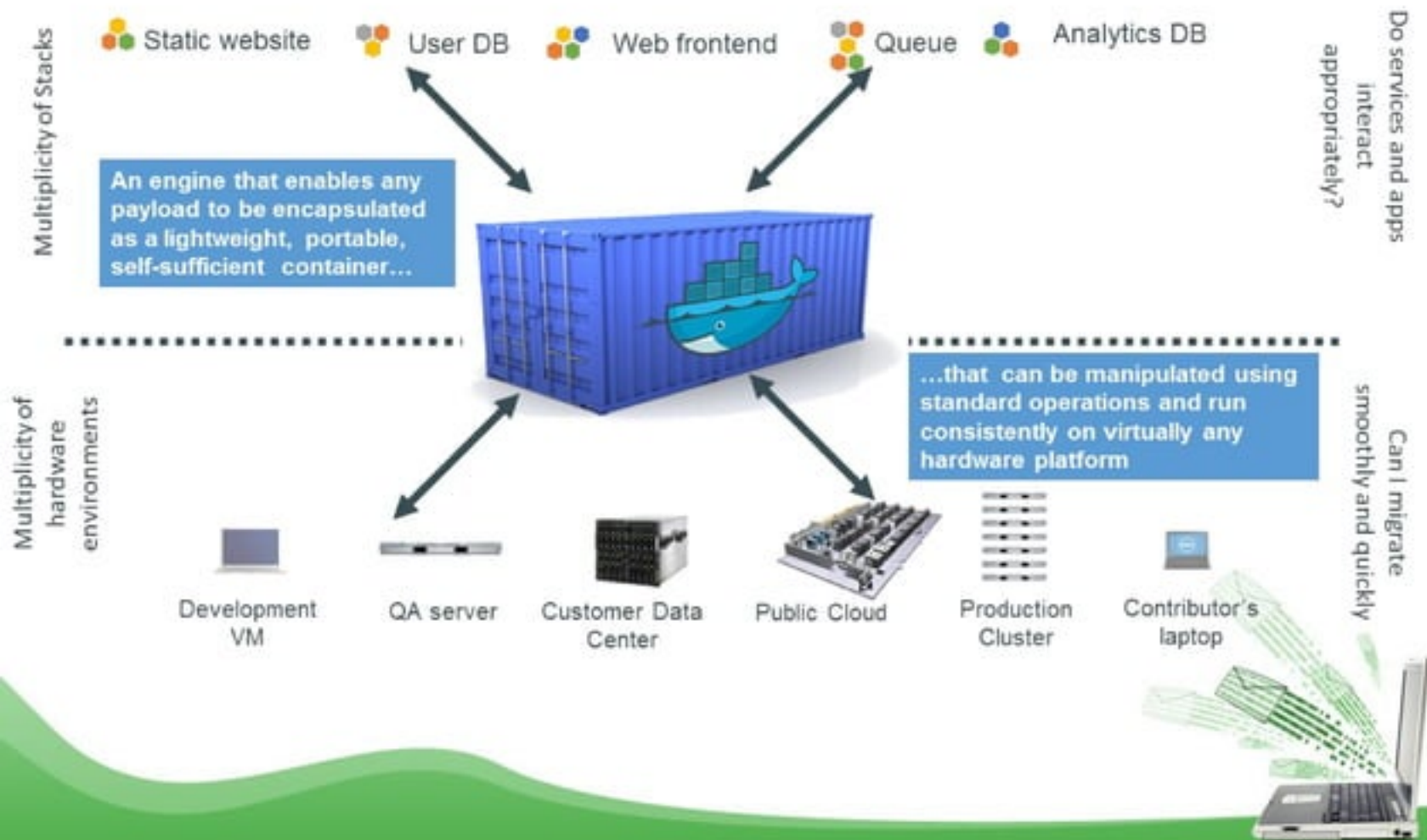
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
							



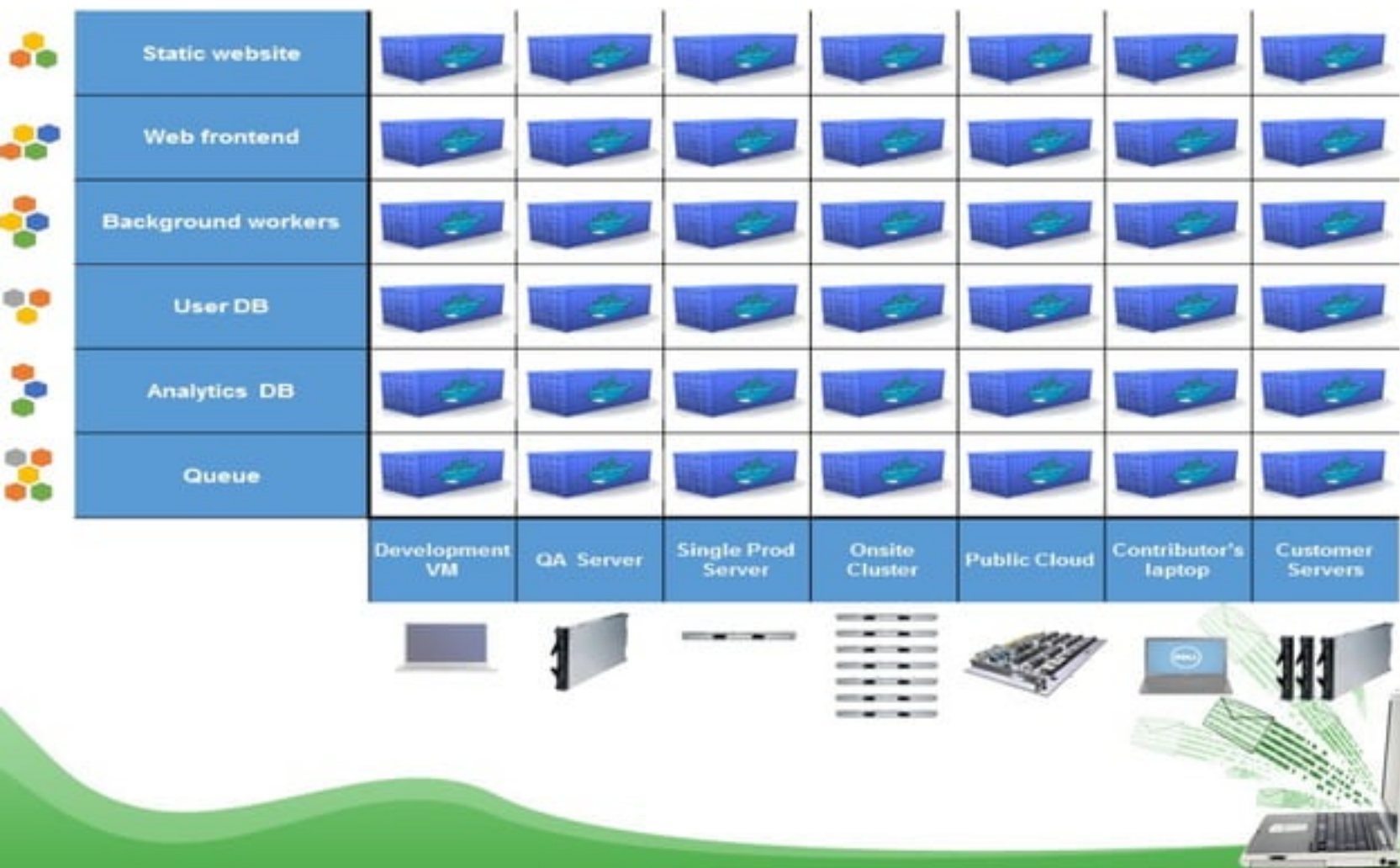
Solution: Intermodal Shipping Container.....



Docker is a Container System for Code.....



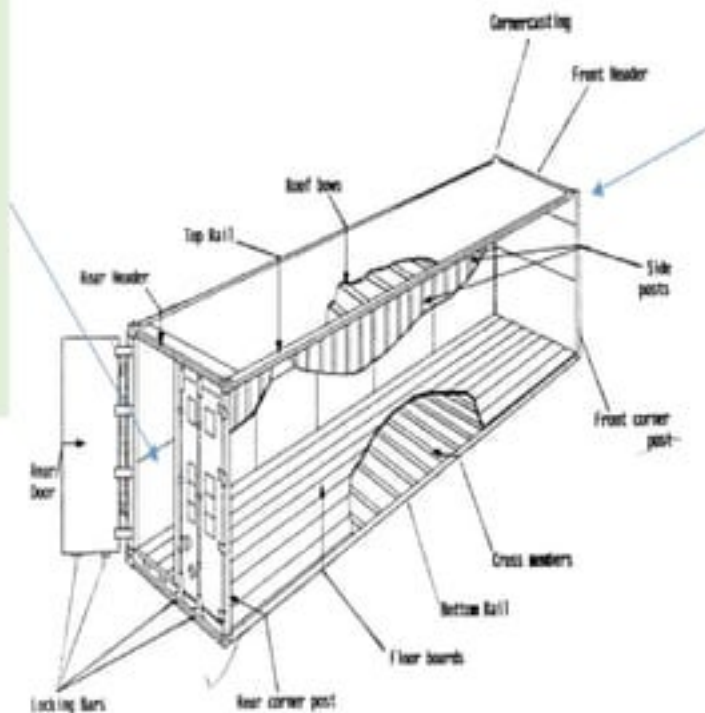
Docker Eliminates the Matrix from Hell.....



Why it Works: Separation of Concerns.....

• Dan the Developer

- Worries about what's "inside" the container
 - His code
 - His Libraries
 - His Package Manager
 - His Apps
 - His Data
- All Linux servers look the same



Major components of the container:

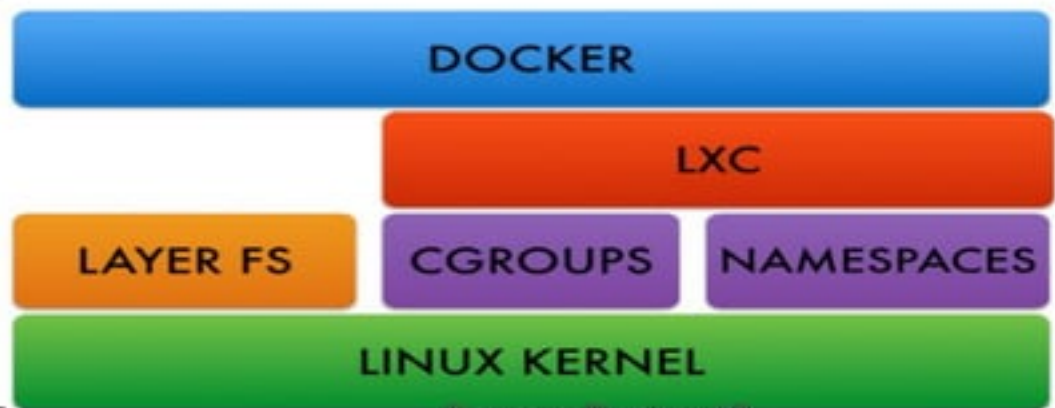
• Oscar the Ops Guy

- Worries about what's "outside" the container
 - Logging
 - Remote access
 - Monitoring
 - Network config
- All containers start, stop, copy, attach, migrate, etc. the same way



Docker Architecture.....

- Docker Engine
 - CLI
 - Docker Daemon
 - Docker Registry
- Docker Hub
 - Cloud service
 - Share Applications
 - Automate workflows
 - Assemble apps from components
- Docker images
- Docker containers

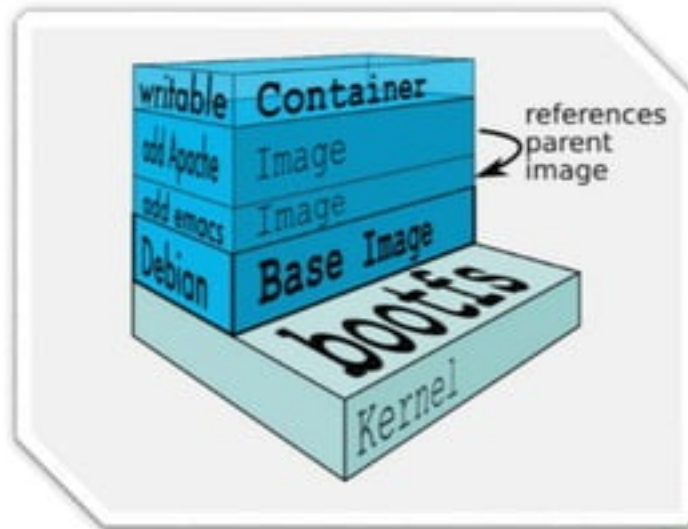
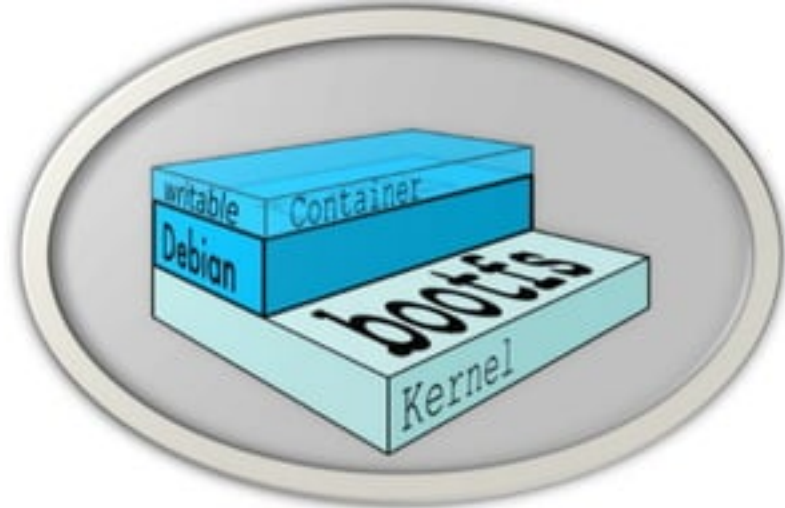


Source : Docker-IO



Docker images.....

- NOT A VHD
- NOT A FILESYSTEM
- uses a *Union File System*
- a read-only *Layer*
- do not have state
- Basically a tar file
- Has a hierarchy
 - Arbitrary depth
- Fits into the Docker Registry



Source: GitHub

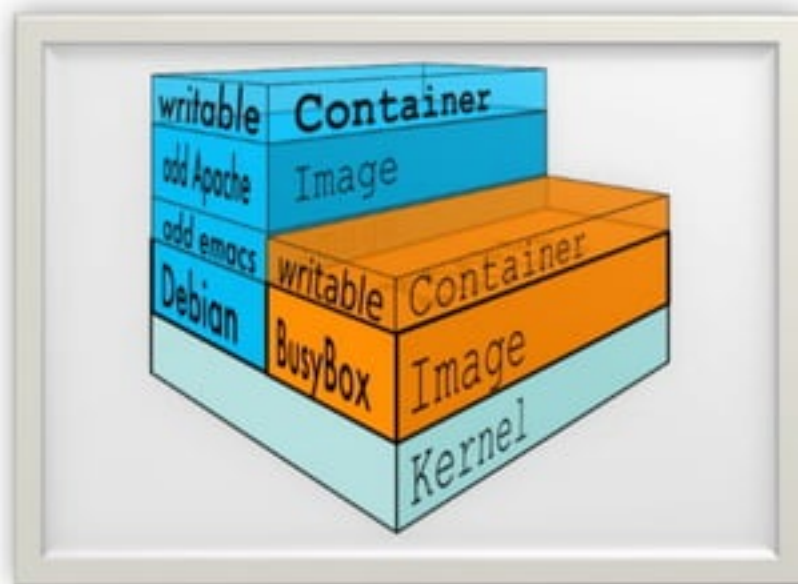


Docker Containers...

Units of software delivery (ship it!)

- run everywhere
 - regardless of kernel version
 - regardless of host distro
 - (but container and host architecture must match*)
- run anything
 - if it can run on the host, it can run in the container
 - i.e., if it can run on a Linux kernel, it can run

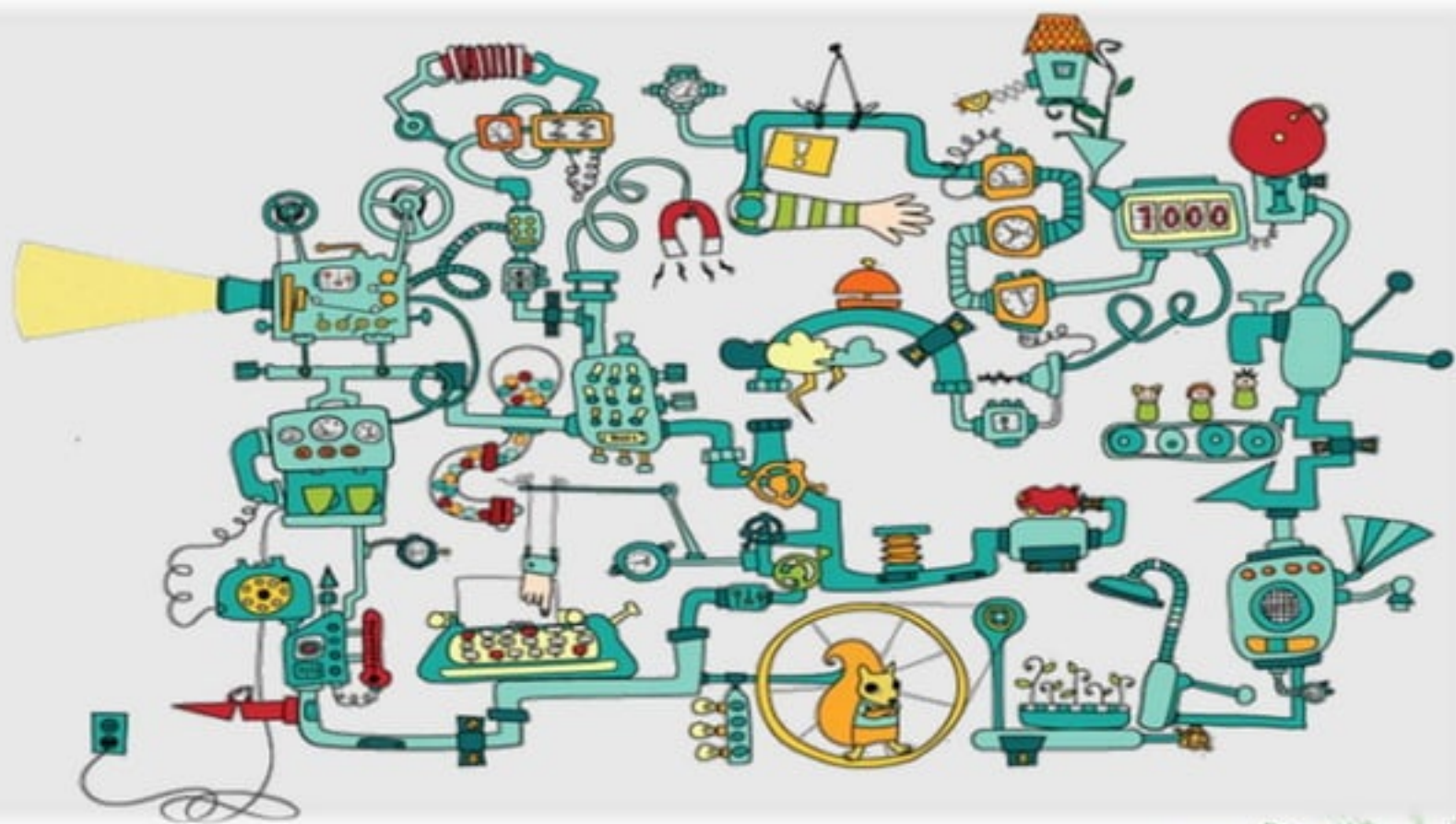
*Unless you emulate CPU with qemu and binfmt



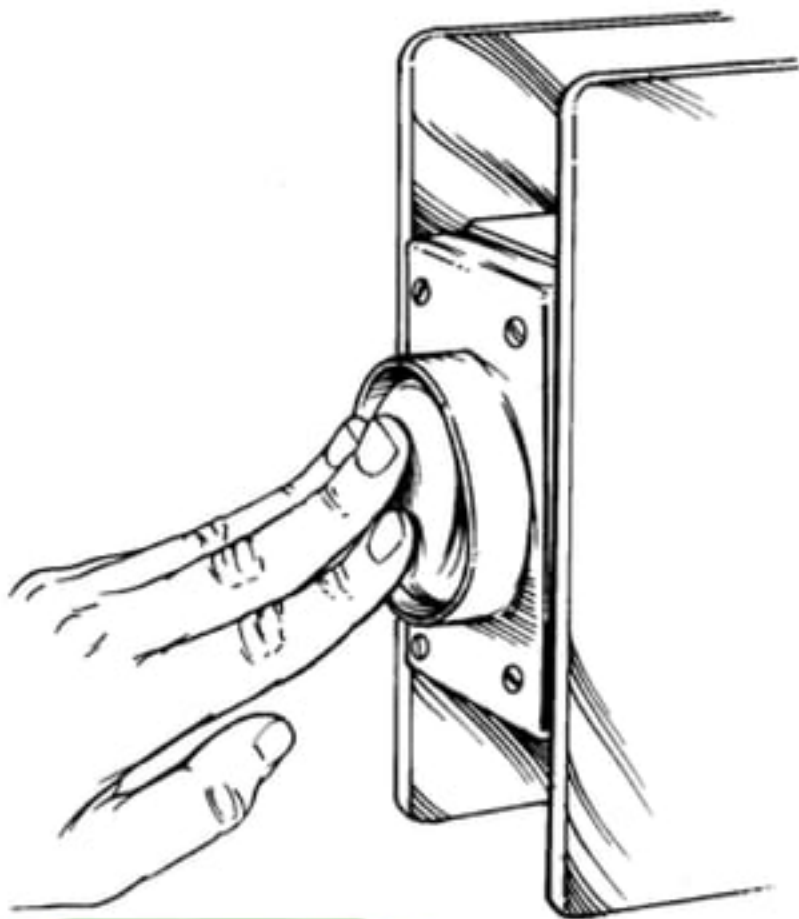
Source: Docker-IO



Containers before Docker.....



Containers after Docker

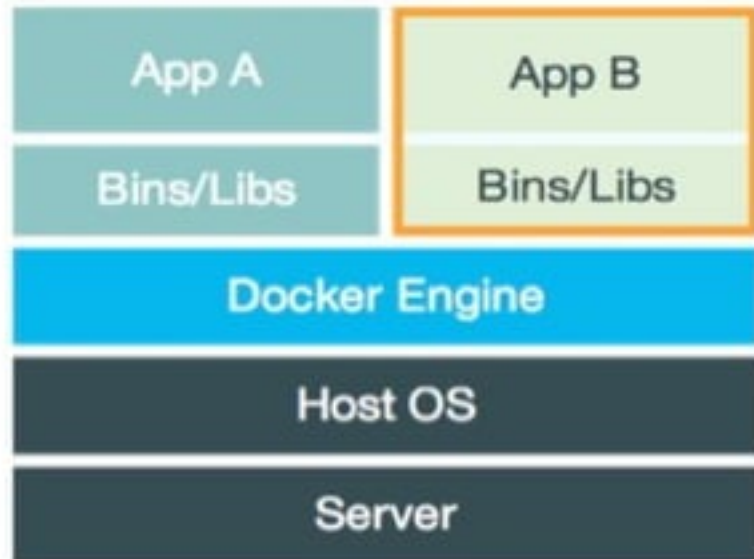
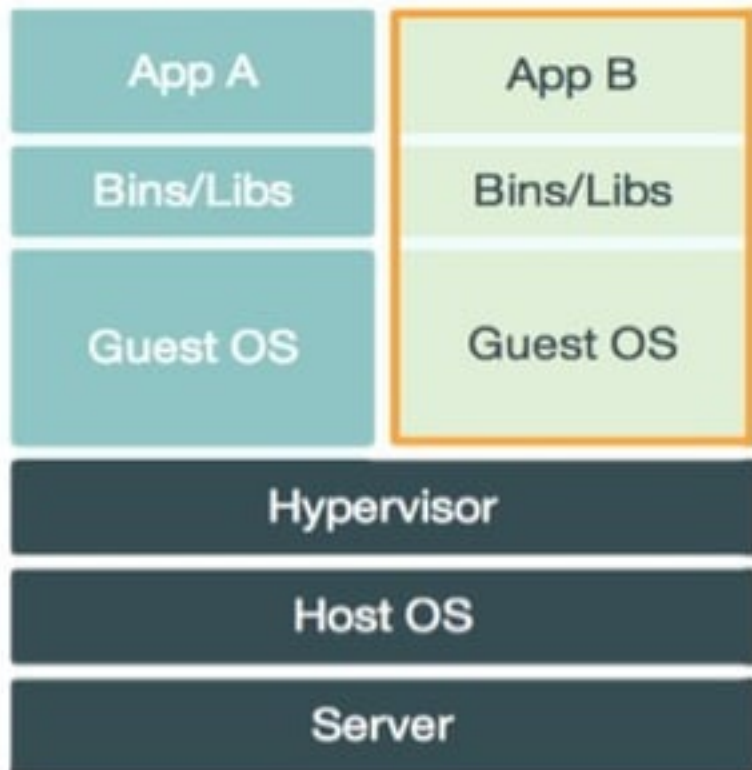


How does Docker work ?

- You can build Docker images that hold your applications
- You can create Docker containers from those Docker images to run your applications.
- You can share those Docker images via Docker Hub or your own registry

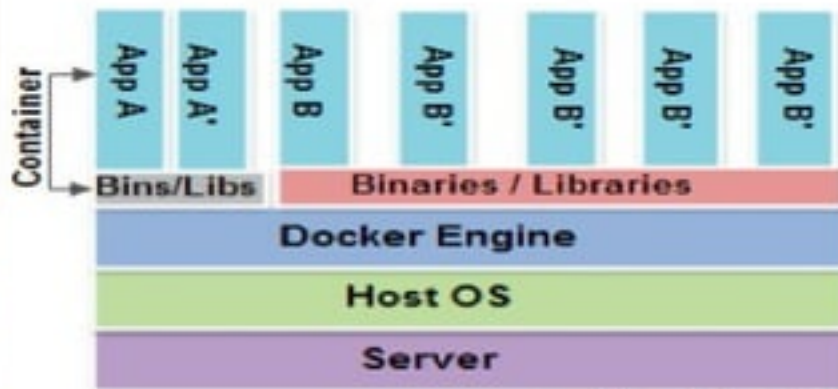
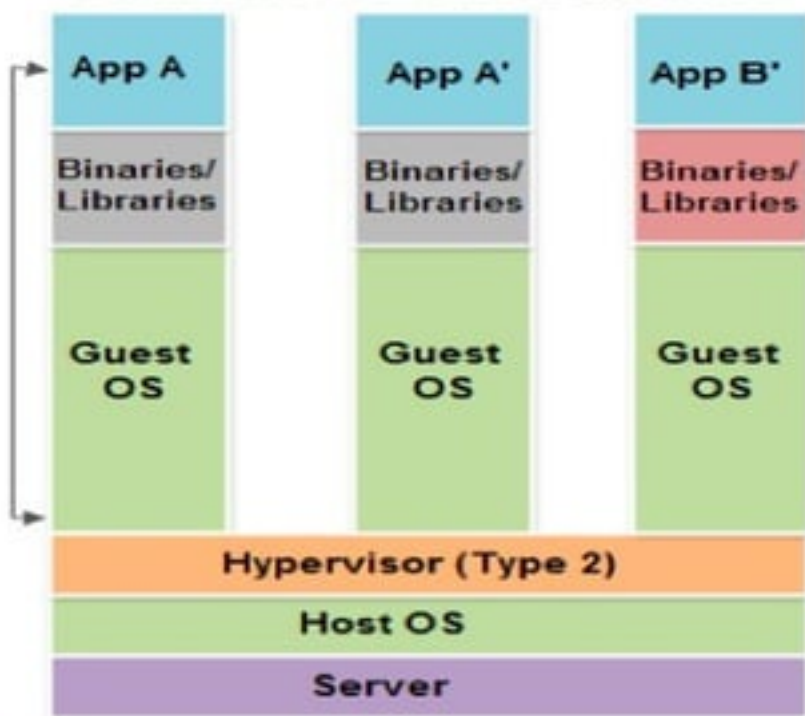


Virtual Machine Versus Container.....



Virtual Machine Versus Container.....

Containers vs Virtual Machines



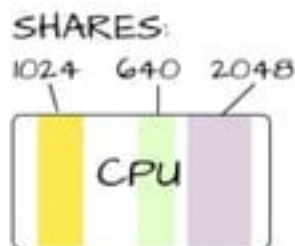
Docker Container Lifecycle

- The Life of a Container
 - Conception
 - **BUILD** an Image from a Dockerfile
 - Birth
 - **RUN** (create+start) a container
 - Reproduction
 - **COMMIT** (persist) a container to a new image
 - **RUN** a new container from an image
 - Sleep
 - **KILL** a running container
 - Wake
 - **START** a stopped container
 - Death
 - **RM** (delete) a stopped container
- Extinction
 - **RM** a container image (delete image)



Linux Cgroups

- Kernel Feature
- Groups of processes
- Control resource allocations
 - CPU
 - Memory
 - Disk
 - I/O
- May be nested



CGROUP #1

Gets half as much CPU time as cgroup #3.

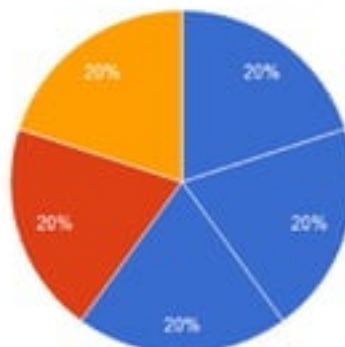
CGROUP #2

Gets the least CPU time.

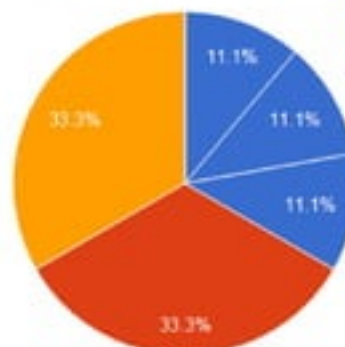
CGROUP #3

Gets the most CPU time.

CPU usage per process without cgroups



CPU usage per process with cgroups

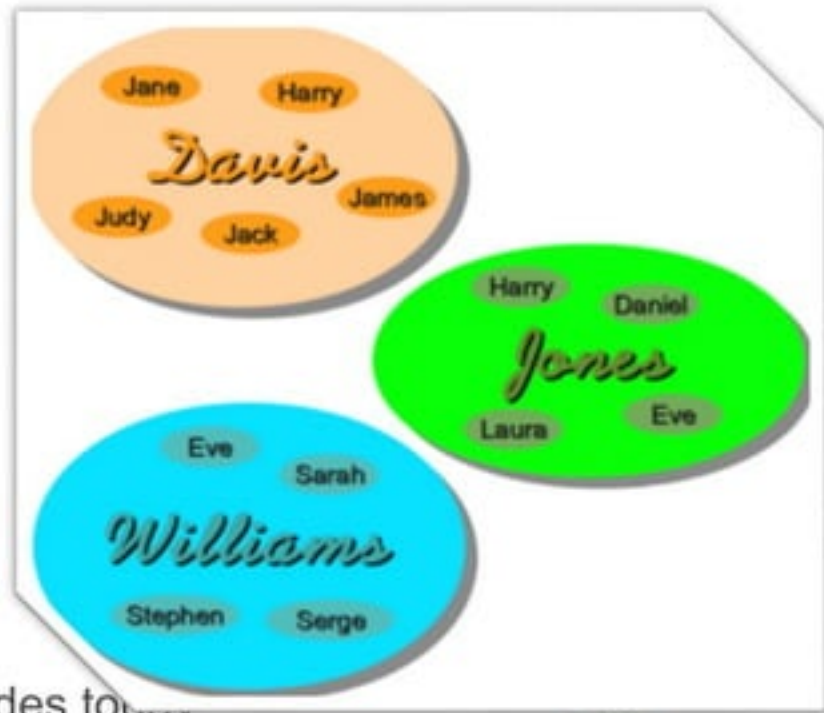


User A - process 1
User A - process 2
User A - process 3
User B - process 4
User C - process 5



Linux Kernel Namespaces

- Kernel Feature
- Restrict your view of the system
 - Mounts (CLONE_NEWNS)
 - UTS (CLONE_NEWUTS)
 - `uname()` output
 - IPC (CLONE_NEWIPC)
 - PID (CLONE_NEWPID)
 - Networks (CLONE_NEWNET)
 - User (CLONE_NEWUSER)
 - Not supported in Docker yet
 - Has privileged/unprivileged modes today
- May be nested



Dockerfile

- Like a Makefile (shell script with keywords)
- Extends from a Base Image
- Results in a new Docker Image
- Imperative, not Declarative
- A Docker file lists the steps needed to build an images
- docker build is used to run a Docker file
- Can define default command for docker run, ports to expose, etc

file 15 lines (11 sloc) 0.475 kb Open Edit Raw Blame History Delete

```
1 FROM ubuntu:12.04
2
3 RUN apt-get update
4
5 # Make it easy to install PPA sources
6 RUN apt-get install -y python-software-properties
7
8 # Install Oracle's Java (Recommended for Hadoop)
9 # Auto-accept the license
10 RUN add-apt-repository -y ppa:webupd8team/java
11 RUN apt-get update
12 RUN echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | sudo /usr/bin/debconf-set-selections
13 RUN apt-get -y install oracle-java7-installer
14 ENV JAVA_HOME /usr/lib/jvm/java-7-oracle
```



Docker CLI Commands (v 1.1.2).....

attach	Attach to a running container	pause	Pause all processes within a container
build	Build an image from a Dockerfile	ps	List containers
commit	Create new image from container's changes	pull	Pull image or repo from docker registry
cp	Copy files from containers fs to host	push	Push image or repo to docker registry
diff	Inspect changes on a container's fs	restart	Restart a running container
events	Get real time events from the server	rm	Remove one or more containers
export	Stream contents of container as tar	rmi	Remove one or more images
history	Show the history of an image	run	Run a command in a new container
images	List images	save	Save an image to a tar archive
import	Create new fs image from a tarball	search	Search for an image in the docker index
info	Display system-wide information	start	Start a stopped container
inspect	Return low-level info on a container	stop	Stop a running container
kill	Kill a running container	tag	Tag an image into a repository
load	Load an image from a tar archive	top	Lookup running processes of a container
login	Login to the docker registry server	unpause	Unpause a paused container
logs	Fetch the logs of a container	version	Show the docker version information
port	Lookup public-facing port	wait	Block and print exit code upon cont exit



Contributing to Docker

Want to hack on Docker ?

- Reporting Security Issues
- Design and Cleanup Proposals
- Reporting Issues
- Build Environment



SUMMARY.....

- Easy to build, run & share containers
- Rapidly expanding ecosystem
- Better performance vs. VMs
- Layered file system gives us git-like control of images
- Reduces complexity of system builds
- Red Hat - Project Atomic Host, and certifications - containerized applications, Geard and OpenShift.
- Google is expected to tightly integrate containers with its IaaS and PaaS offerings.



References.....

- <https://www.docker.com/what-docker>
- [https://en.wikipedia.org/wiki/Docker_\(software\)](https://en.wikipedia.org/wiki/Docker_(software))
- <https://github.com/docker/docker/blob/990a3e30fa66e7bd3df3c78c873c97c5b1310486/daemon/graphdriver/driver.go#L37-L43>
- <https://github.com/docker/distribution>
- The Docker Book/ Version: v1.10.2 (bf6b7fe)
- The Docker Book/ Version: v1.10.2 (bf6b7fe)/summary



