

# Introduction to Docker

# Contents



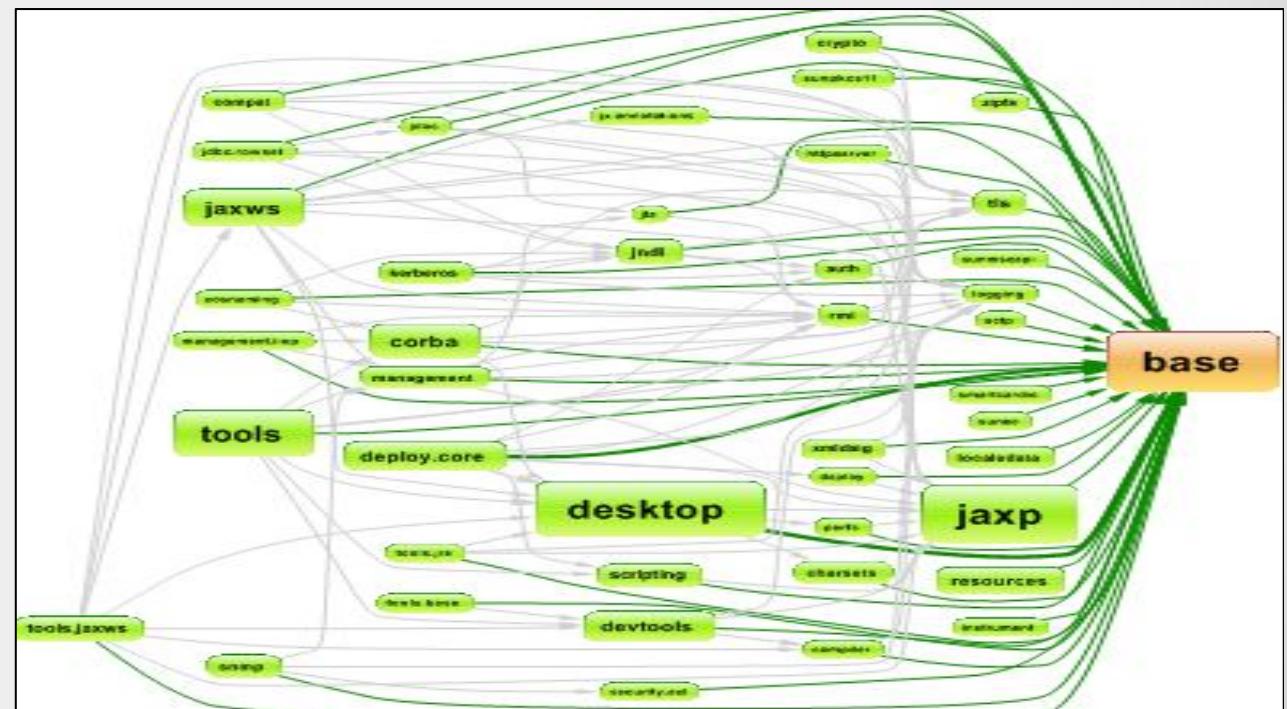
- The challenge
- The Solution
- Containers vs. VMs
- What is Docker and Why people care: Separation of Concerns
- Why They Work?
- Docker Architecture
- Docker Components
- Docker Workflow
- Docker Orchestration

“

What Is the Problem?

# The dependency hell

- Maven: Java
  - npm: Node.js
  - pip : Python
  - Gem: Ruby
  - Composer: PHP

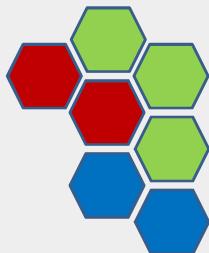


# Challenges

Thin app on mobile,  
tablet

2020

Assembled by  
developers using  
best available  
services



How to ensure services  
interact consistently, avoid  
dependency hell

Running on any  
available set of  
physical resources  
(public/private/  
virtualized)



How to migrate & scale  
quickly, ensure  
compatibility

How to avoid  $n \times n$   
different configs

# 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

 Queue  
Redis + redis-sentinel

 Analytics DB  
hadoop + hive + thrift + OpenJDK

 Web frontend  
Ruby + Rails + sass + Unicorn

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

Do services and apps interact appropriately?

Multiplicity of hardware environments

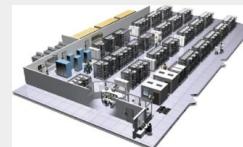
 Development VM



Customer Data Center



Public Cloud



Production Cluster



Disaster recovery

Contributor's laptop



Production Servers

Can I migrate smoothly and quickly?

# The Matrix From Hell

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

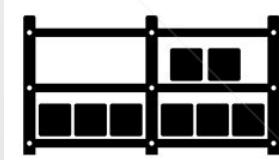
# Cargo Transport Pre-1960

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)

“

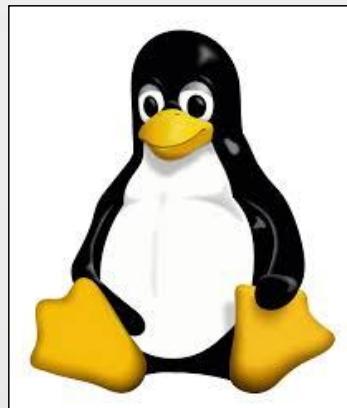
The Solution

# The Solution

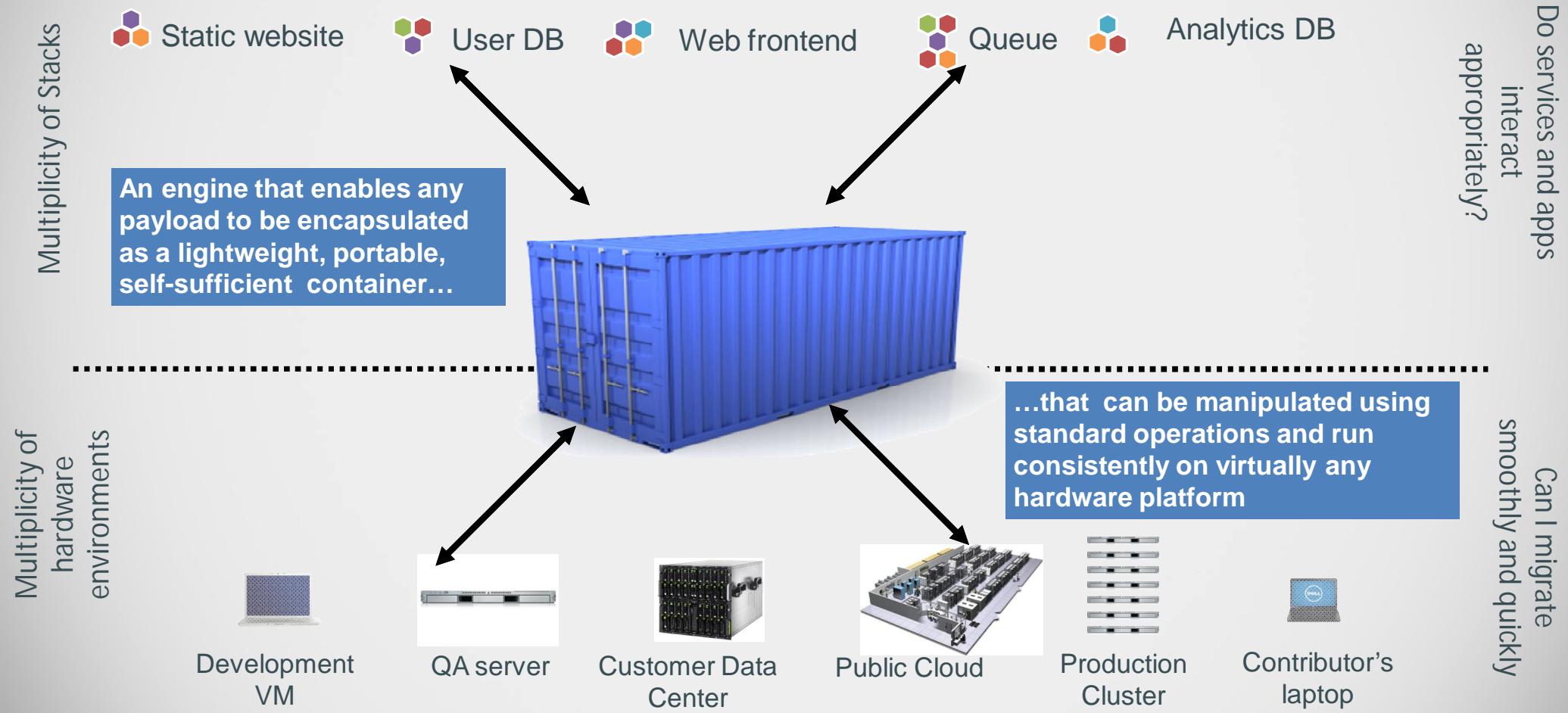


# Linux Containers

An operating system–level virtualization method for running multiple isolated Linux systems (containers) on a single control host.



# Docker is a shipping container system for code



# Docker eliminates the matrix from Hell

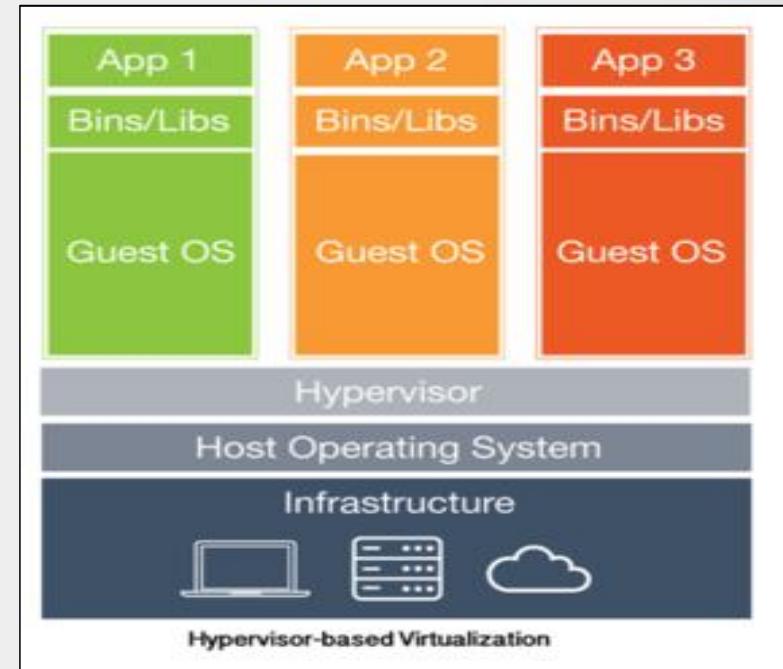


	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website							
Web frontend							
Background workers							
User DB							
Analytics DB							
Queue							
	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's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

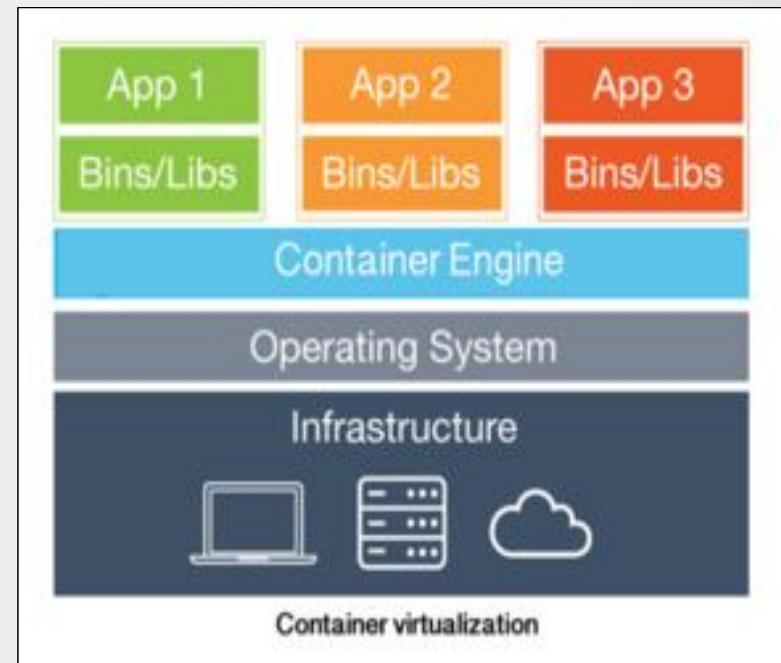
“

## VMs vs. Containers

# VMs

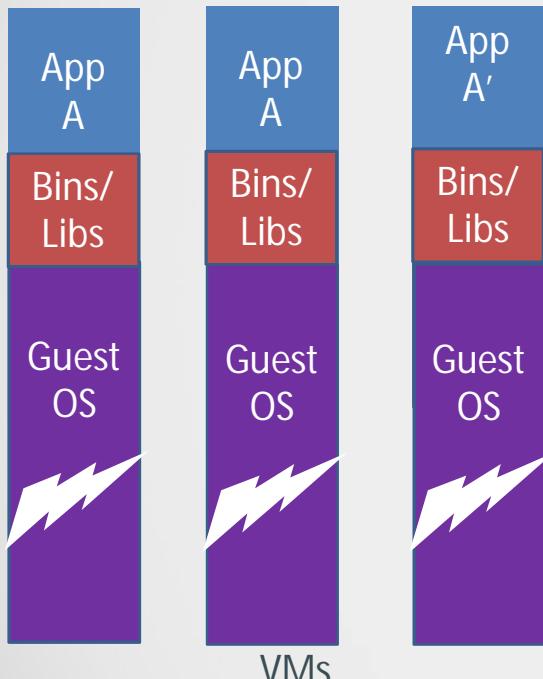


# Containers



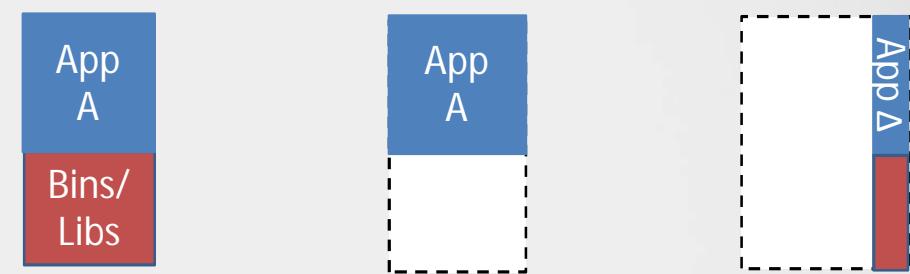
# Why are containers lightweight?

VMs



Every app, every copy of an app, and every slight modification of the app requires a new virtual server

Containers

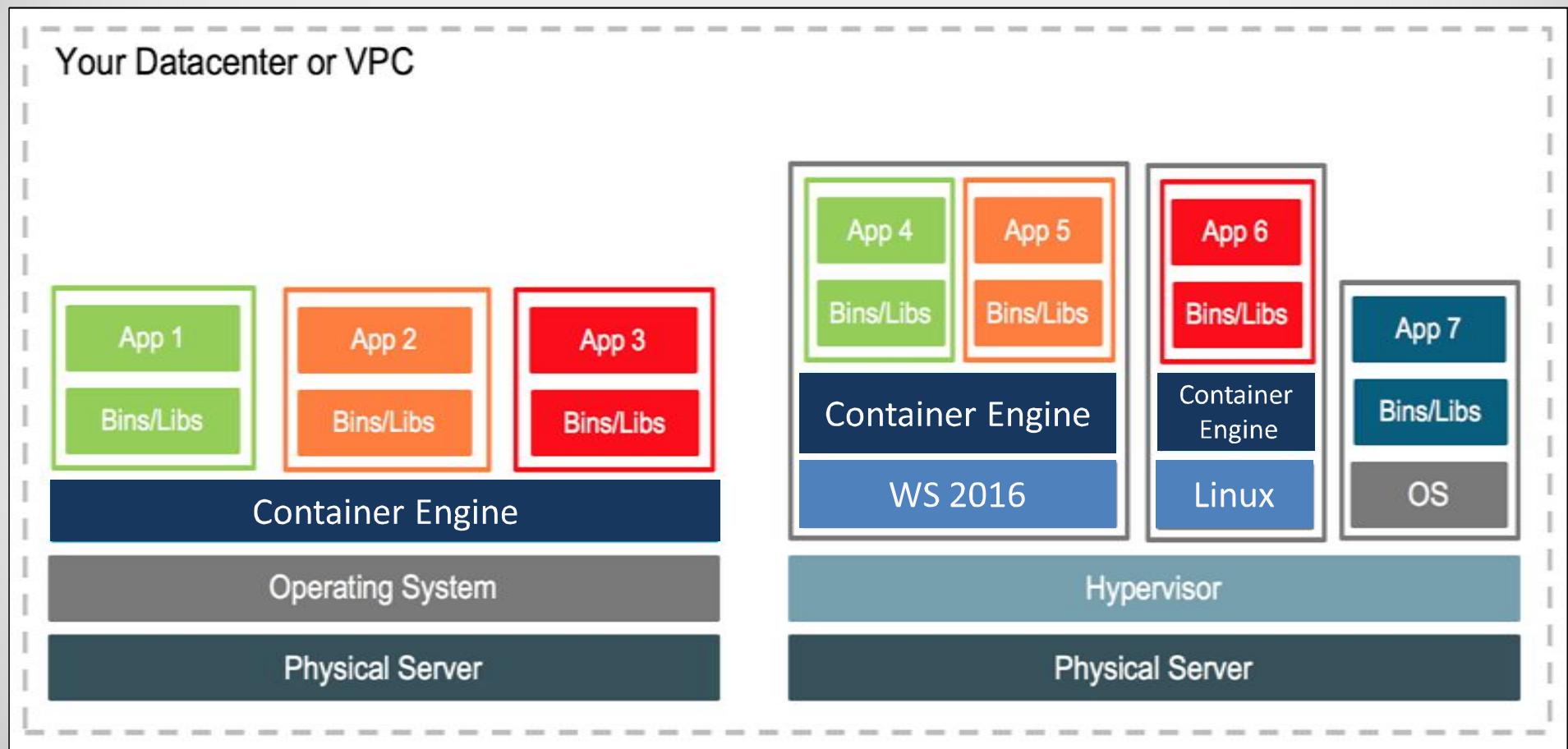


Original App  
(No OS to take up space, resources, or require restart)

Copy of App  
No OS. Can Share bins/libs

Modified App  
Copy on write capabilities allow us to only save the diffs Between container A and container A'

# They're different, not mutually exclusive



# VMs vs. Containers

	VMs	Containers
<b>Security</b>	More isolated	Less isolated
<b>Size</b>	GBs	MBs
<b>Provision</b>	Mins	Secs
<b>OS</b>	More flexible	Less flexible



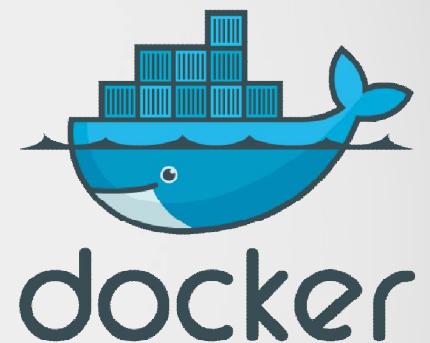
“

What is Docker and Why people care

# What is Docker?

---

- Docker is “a platform for developers and sysadmins to develop, ship, and run applications”, based on containers.
- Docker is open-source, mainly created in Go and originally on top of libvirt and LXC.
- Docker simplifies and standardizes the creation and management of containers.



# Build, Ship, Run, Any App Anywhere

From Dev



To Ops



Any App



Any OS



Windows



Linux

Anywhere



Physical



Virtual



Cloud



# Why Developers Care



- **Build once, run anywhere**

- A clean, safe, hygienic and portable runtime environment for your app.
- No worries about missing dependencies, packages and other pain points during subsequent deployments.
- Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying.
- Automate testing, integration, packaging...anything you can script.
- Reduce/eliminate concerns about compatibility on different platforms, either your own or your customers.

# Why Devops Cares?



- **Configure once, run anything**
  - Make the entire lifecycle more efficient, consistent, and repeatable.
  - Increase the quality of code produced by developers.
  - Eliminate inconsistencies between development, test, production, and customer environments.
  - Support segregation of duties.
  - Significantly improves the speed and reliability of continuous deployment and continuous integration systems.
  - Because the containers are so lightweight, address significant performance, costs, deployment, and portability issues normally associated with VMs.

“

Why They Work?

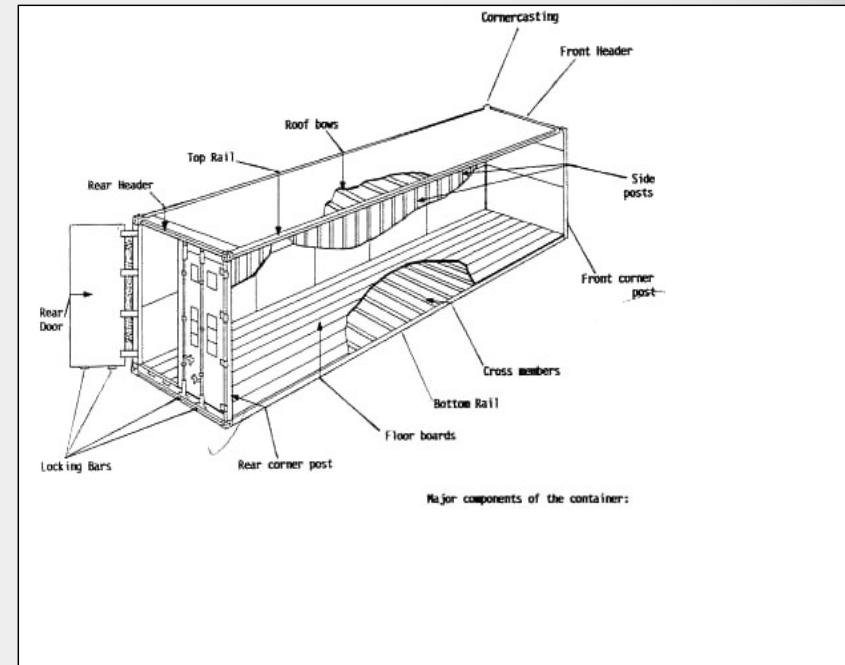
# Why it works - separation of concerns

**Operation person** Worries about what's "*outside*" the container

- Logging
- Remote access
- Monitoring
- Network config

**Developers** Worries about what's "*inside*" the container

- His code
- His Libraries
- His Package Manager
- His Apps
- His Data



# More technical explanation

## WHY

- **Run everywhere**

- Regardless of kernel version (2.6.32+)
- Regardless of host distro
- Physical or virtual, cloud or not
- Container and host architecture must match\*

- **Run anything**

- If it can run on the host, it can run in the container

## WHAT

- **High Level—It's a lightweight VM**

- Own process space
- Own network interface
- Can run stuff as root
- Can have its own /sbin/init (different from host)
- <>machine container>>

- **Low Level—It's chroot on steroids**

- Can also not have its own /sbin/init
- Container=isolated processes
- Share kernel with host
- No device emulation (neither HVM nor PV) from host
- <>application container>>

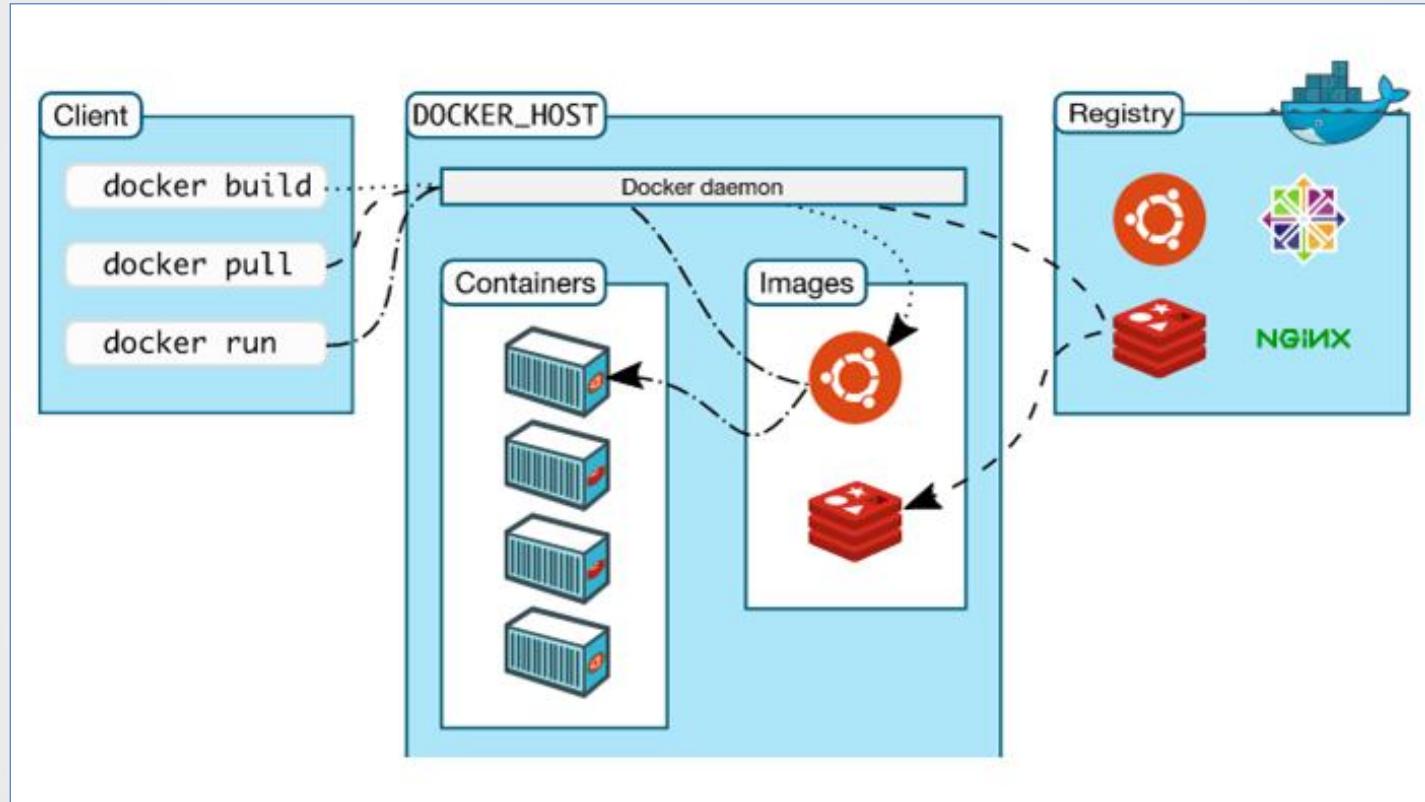
Consider a case where a company wants to test an application at scale and are using full-clone virtual machines. Full-clone VMs in the best scenario take several **minutes** to boot, and most virtual machine management platforms can only boot a handful machines simultaneously. Based on these factors standing up 1,000 full-clone virtual machines could take hours if not **days**. Meaning the test cycle itself could take days if not **weeks**.

By contrast, the same application running inside of a Docker container can be started in less than half a **second**. Standing up 1,000 containers becomes trivial. Test cycle times can be slashed from days to **hours**. This can translate into measurable savings as well as increased agility.

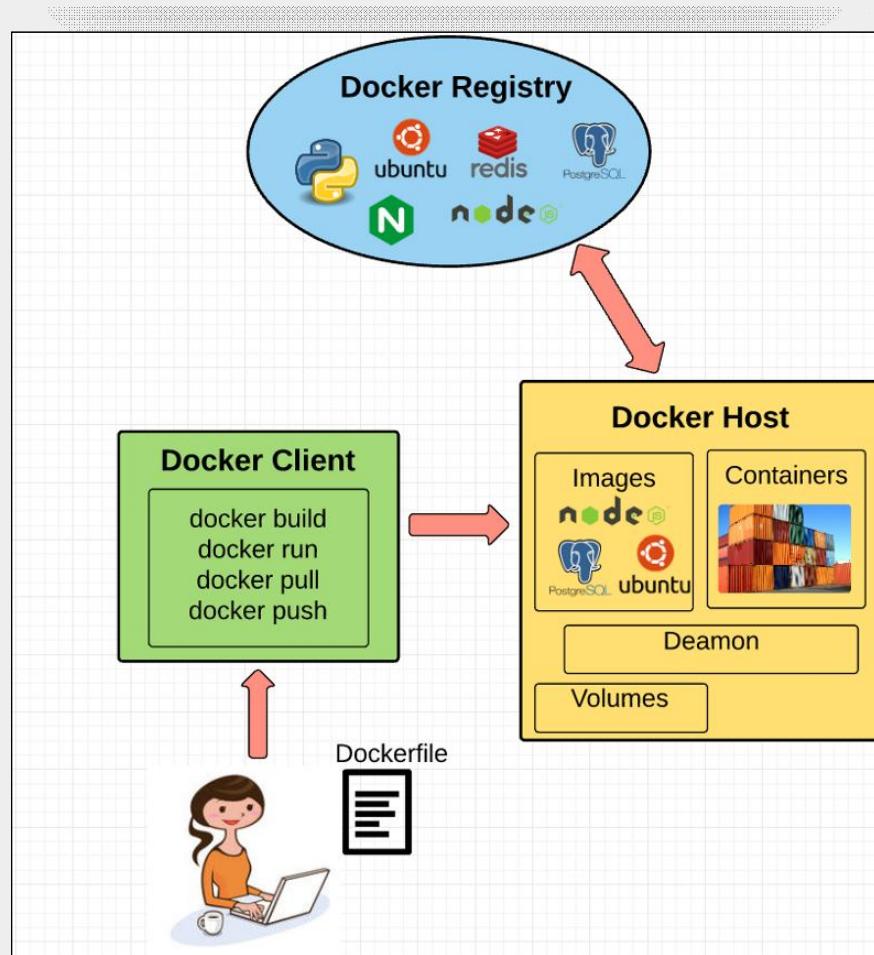


# Docker Architecture

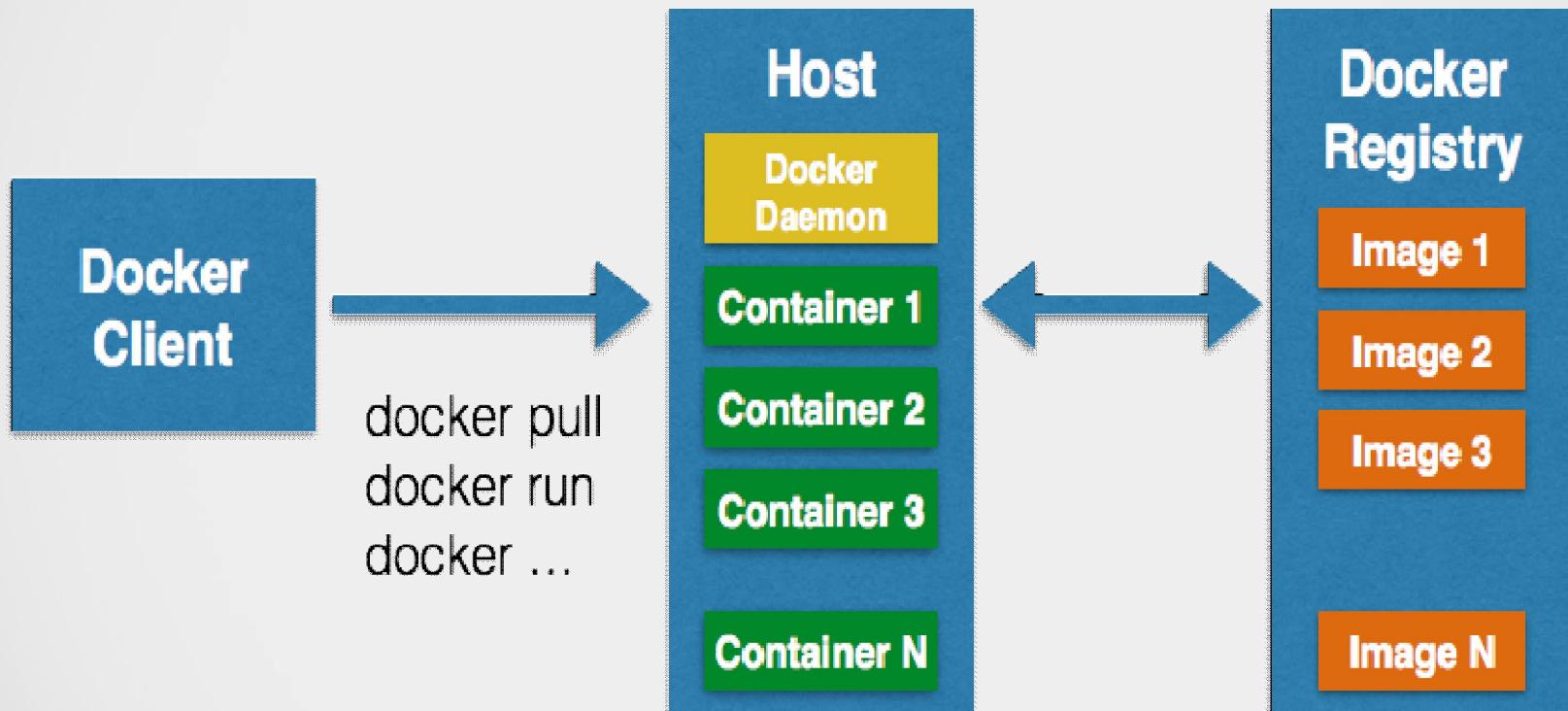
# Docker Architecture



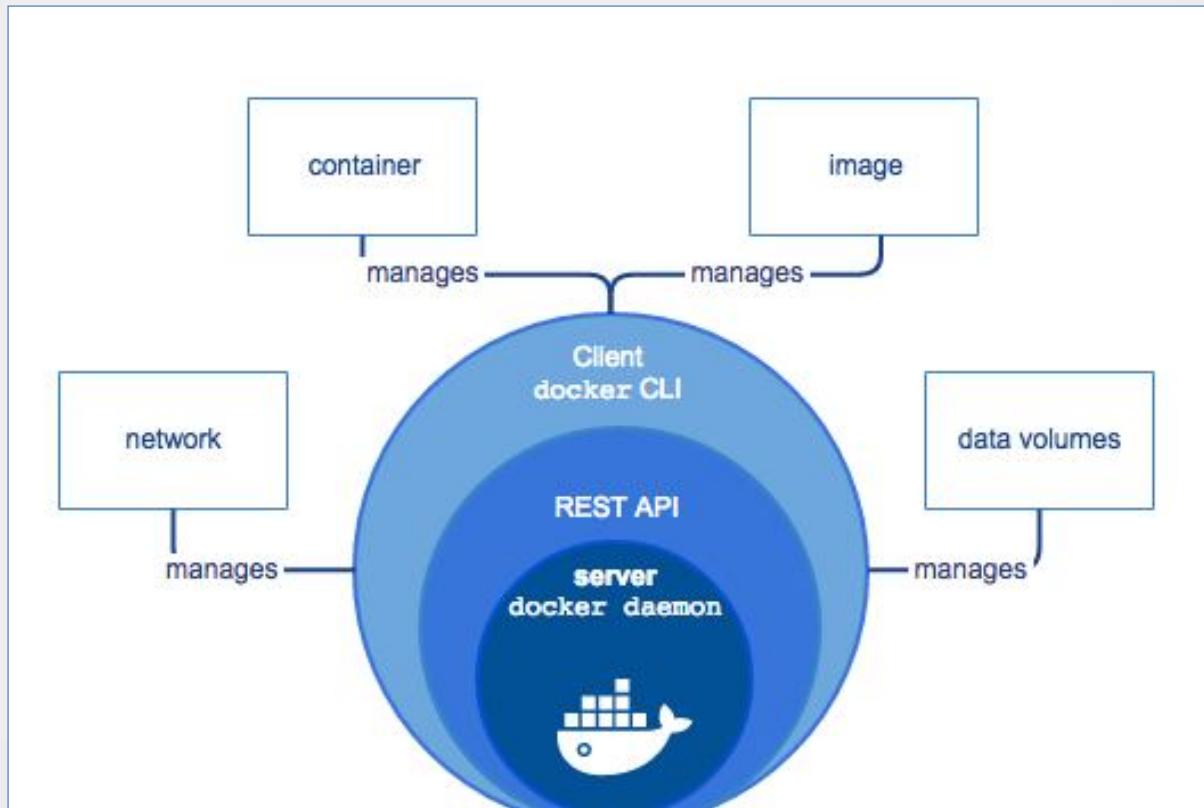
# Docker Architecture



# Docker Architecture



# Docker Engine





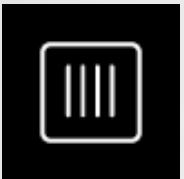
## Docker Components

# Docker components



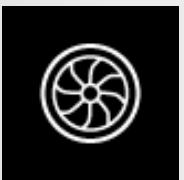
## Docker Image

The basis of a Docker container. Represents a full application



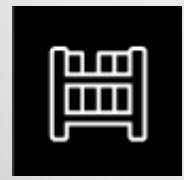
## Docker Container

The standard unit in which the application service resides and executes



## Docker Engine

Creates, ships and runs Docker containers deployable on a physical or virtual, host locally, in a datacenter or cloud service provider

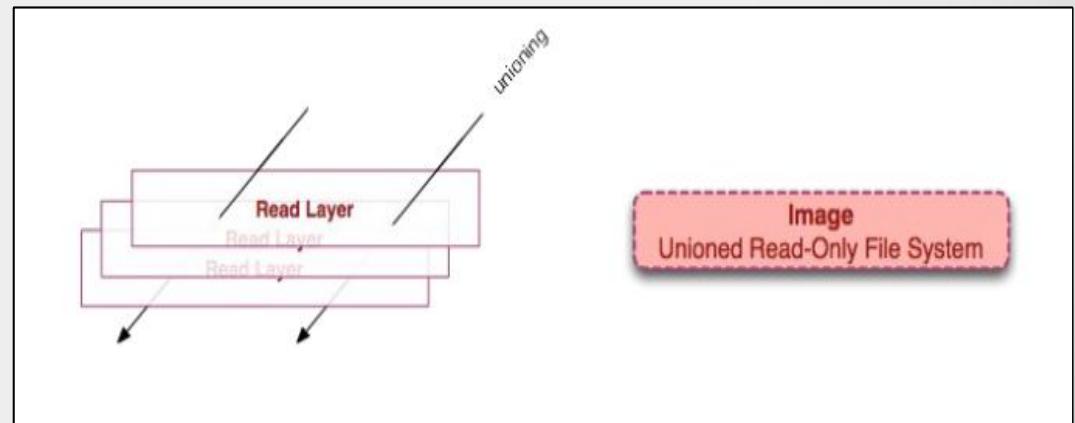


## Registry Service (Docker Hub or Docker Trusted Registry)

Cloud or server based storage and distribution service for your images

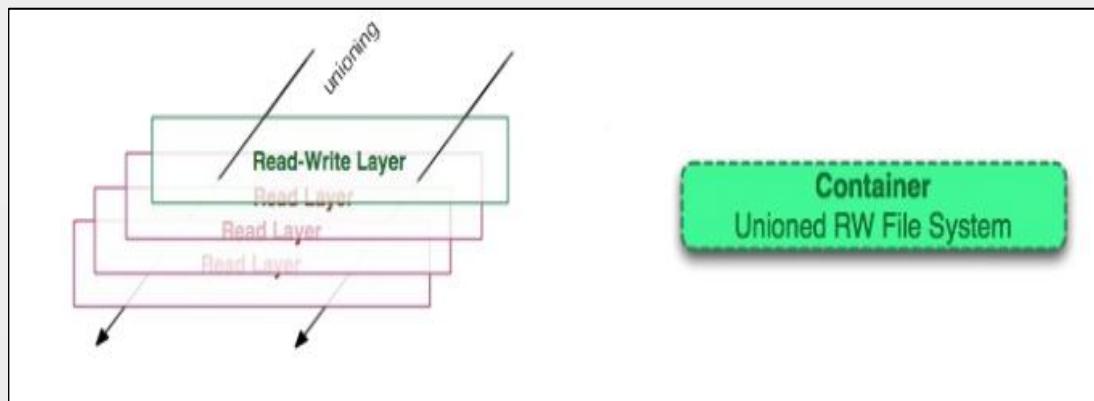
# Images

- Read only template used to create containers
- Build by you or other docker users
- Stored in the docker hub or your local registry
- Every image starts from base image
- Include:
  - ***Application***
  - ***Dependencies***
  - ***Libraries***
  - ***Binaries***
  - ***Configuration files***



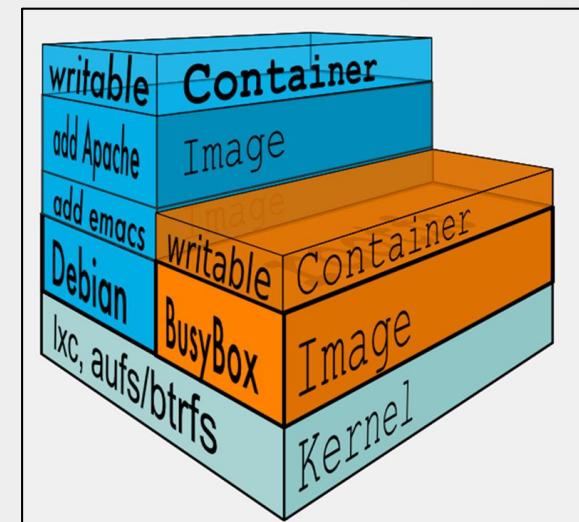
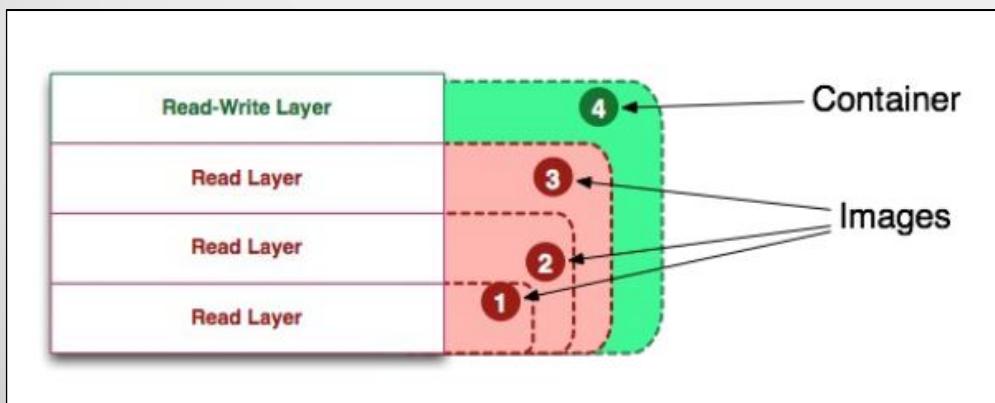
# Containers

- Isolated application platform
- Contains everything needed to run your application
- Based on one or more images
- Docker containers launched from Docker image
- When Docker container runs, it adds a read-write layer on top of the image

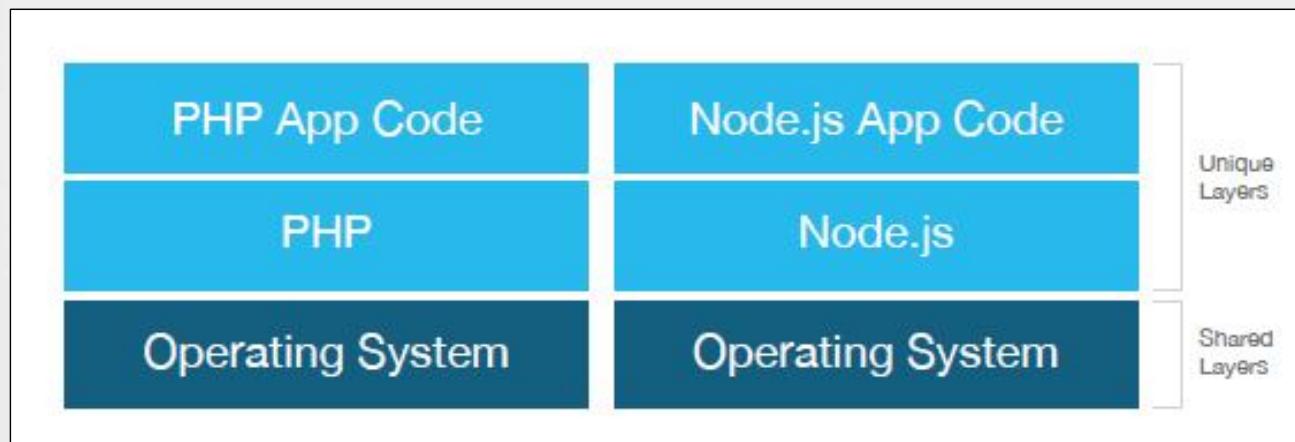


# Image vs. Container

- Docker Image is a class
- Docker Container is a instance of class



Docker will not only share the base image between containers, but it will also share the same layers between different images.



# Docker File



- Dockerfile is instructions to build Docker image
  - How to run commands
  - Add files or directories
  - Create environment variables
  - What process to run when launching container
- Result from building Dockerfile is Docker image

```

1 # Start with ubuntu 14.04
2 FROM ubuntu:14.04
3
4 MAINTAINER preethi kasireddy iam.preethi.k@gmail.com
5
6 # For SSH access and port redirection
7 ENV ROOTPASSWORD sample
8
9 # Turn off prompts during installations
10 ENV DEBIAN_FRONTEND noninteractive
11 RUN echo "debconf shared/accepted-oracle-license-v1-1 select true" | debconf-set-selections
12 RUN echo "debconf shared/accepted-oracle-license-v1-1 seen true" | debconf-set-selections
13
14 # Update packages
15 RUN apt-get -y update
16
17 # Install system tools / libraries
18 RUN apt-get -y install python3-software-properties \
19     software-properties-common \
20     bzip2 \
21     ssh \
22     net-tools \
23     vim \
24     curl \
25     expect \
26     git \
27     nano \
28     wget \
29     build-essential \
30     dialog \
31     make \
32     build-essential \
33     checkinstall \
34     bridge-utils \
35     virt-viewer \
36     python-pip \
37     python-setuptools \
38     python-dev
39
40 # Install Node, npm
41 RUN curl -SL https://deb.nodesource.com/setup_4.x | sudo -E bash -
42 RUN apt-get install -y nodejs
43
44 # Add oracle-jdk7 to repositories
45 RUN add-apt-repository ppa:webupd8team/java
46
47 # Make sure the package repository is up to date
48 RUN echo "deb http://archive.ubuntu.com/ubuntu precise main universe" > /etc/apt/sources.list
49
50 # Update apt
51 RUN apt-get -y update
52
53 # Install oracle-jdk7
54 RUN apt-get -y install oracle-java7-installer
55
56 # Export JAVA_HOME variable
57 ENV JAVA_HOME /usr/lib/jvm/java-7-oracle
58

59 # Run sshd
60 RUN apt-get install -y openssh-server
61 RUN mkdir /var/run/sshd
62 RUN echo "root:$ROOTPASSWORD" | chpasswd
63 RUN sed -i 's/PermitRootLogin without-password/PermitRootLogin yes/' /etc/ssh/sshd_config
64
65 # SSH login fix. Otherwise user is kicked off after login
66 RUN sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@' -i /etc/pam.d/su
67
68 # Expose Node.js app port
69 EXPOSE 8000
70
71 # Create tap-to-android app directory
72 RUN mkdir -p /usr/src/my-app
73 WORKDIR /usr/src/my-app
74
75 # Install app dependencies
76 COPY . /usr/src/my-app
77 RUN npm install
78
79 # Add entrypoint
80 ADD entrypoint.sh /entrypoint.sh
81 RUN chmod +x /entrypoint.sh
82 ENTRYPOINT ["/entrypoint.sh"]
83
84 CMD ["npm", "start"]

```

# Dockerfile – Linux Example

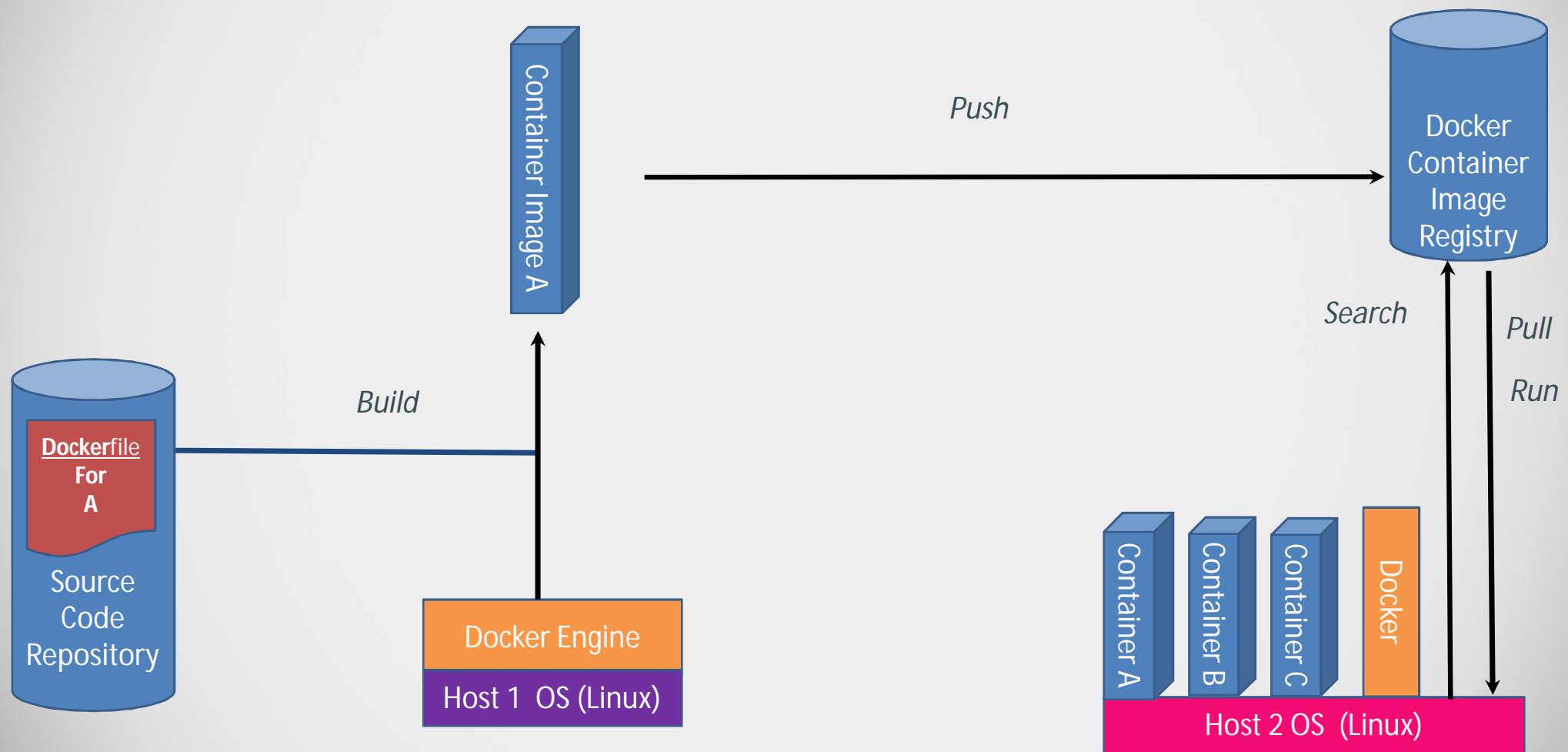
```
1 # our base image
2 FROM alpine:latest
3
4 # Install python and pip
5 RUN apk add --update py-pip
6
7 # upgrade pip
8 RUN pip install --upgrade pip
9
10 # install Python modules needed by the Python app
11 COPY requirements.txt /usr/src/app/
12 RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt
13
14 # copy files required for the app to run
15 COPY app.py /usr/src/app/
16 COPY templates/index.html /usr/src/app/templates/
17
18 # tell the port number the container should expose
19 EXPOSE 5000
20
21 # run the application
22 CMD ["python", "/usr/src/app/app.py"]
```

- Instructions on how to build a Docker image
- Looks very similar to “native” commands
- Important to optimize your Dockerfile

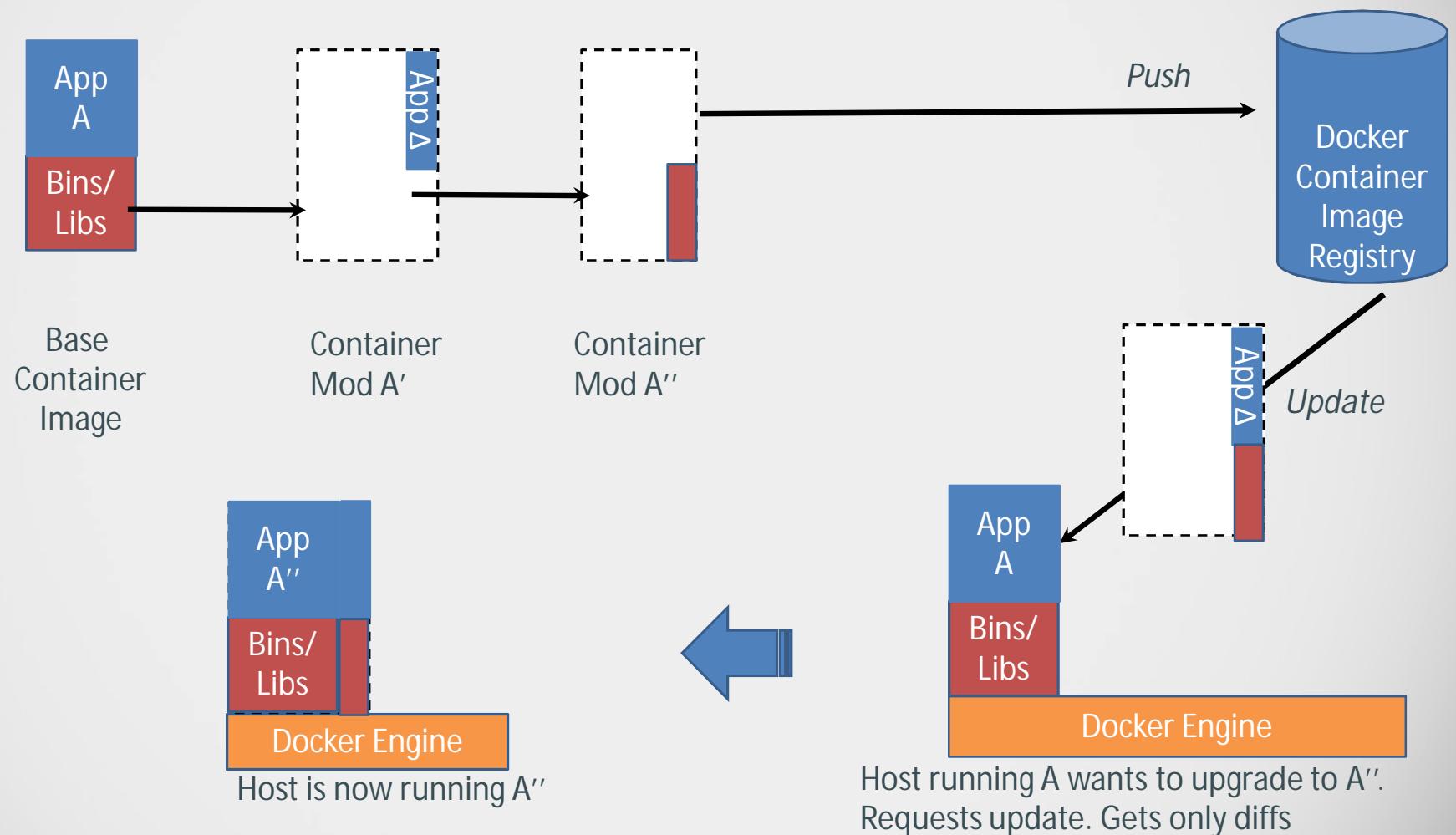
“

## Docker Workflow

# What are the basics of the Docker system?



# Changes and Updates



“

## Docker Orchestration

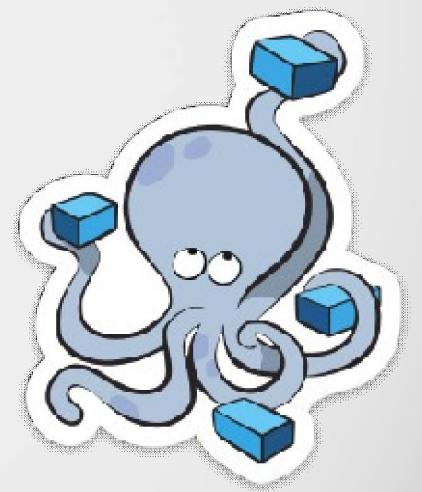
# Docker Orchestration

- Problems with standalone Docker Running a server cluster on a set of Docker containers, on a single Docker host is vulnerable to single point of failure!



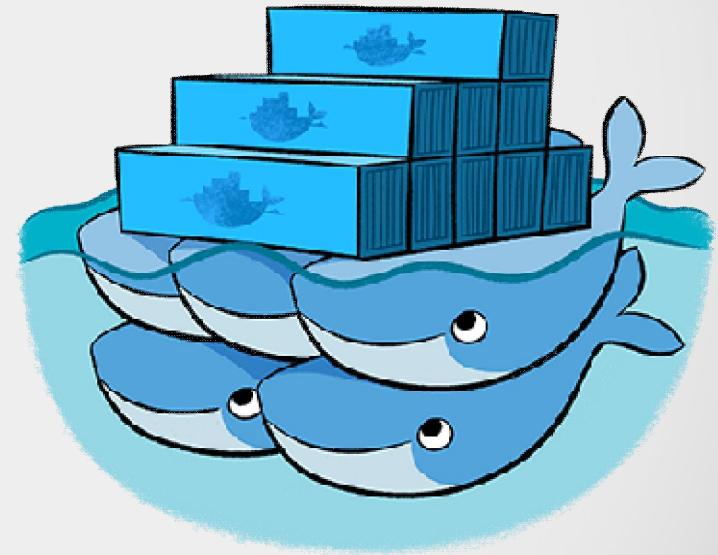
# Docker Compose

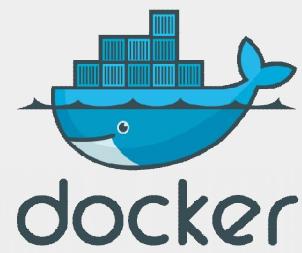
- Tool for defining and running multi-container applications with Docker in a single file
- Fast, isolated development environments using Docker.
- Quick and easy to start.



# Docker Swarm

- Native Clustering System
- Clustering (management) for Docker.
- Manage multiple Docker daemons.
- Distribute workloads.





Thank You!