Introduction to Docker

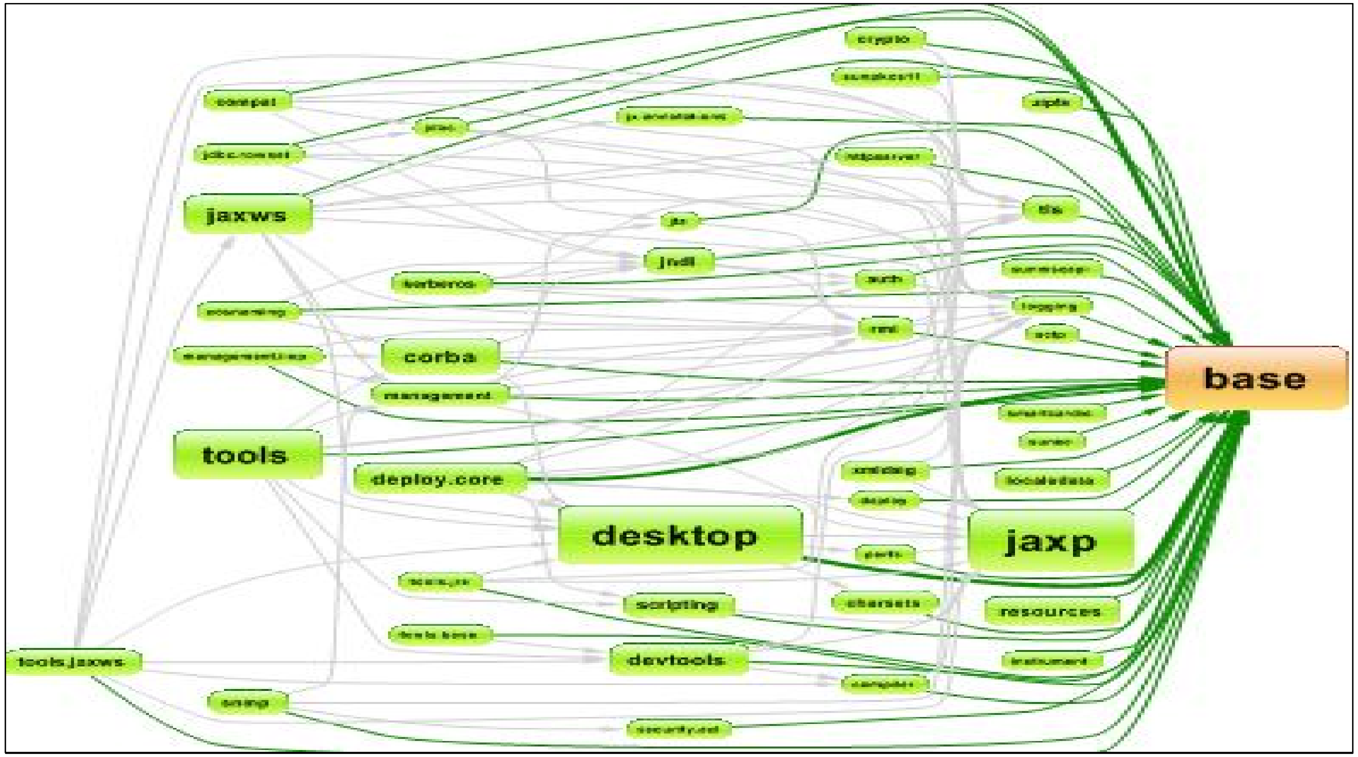
Contents

- 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

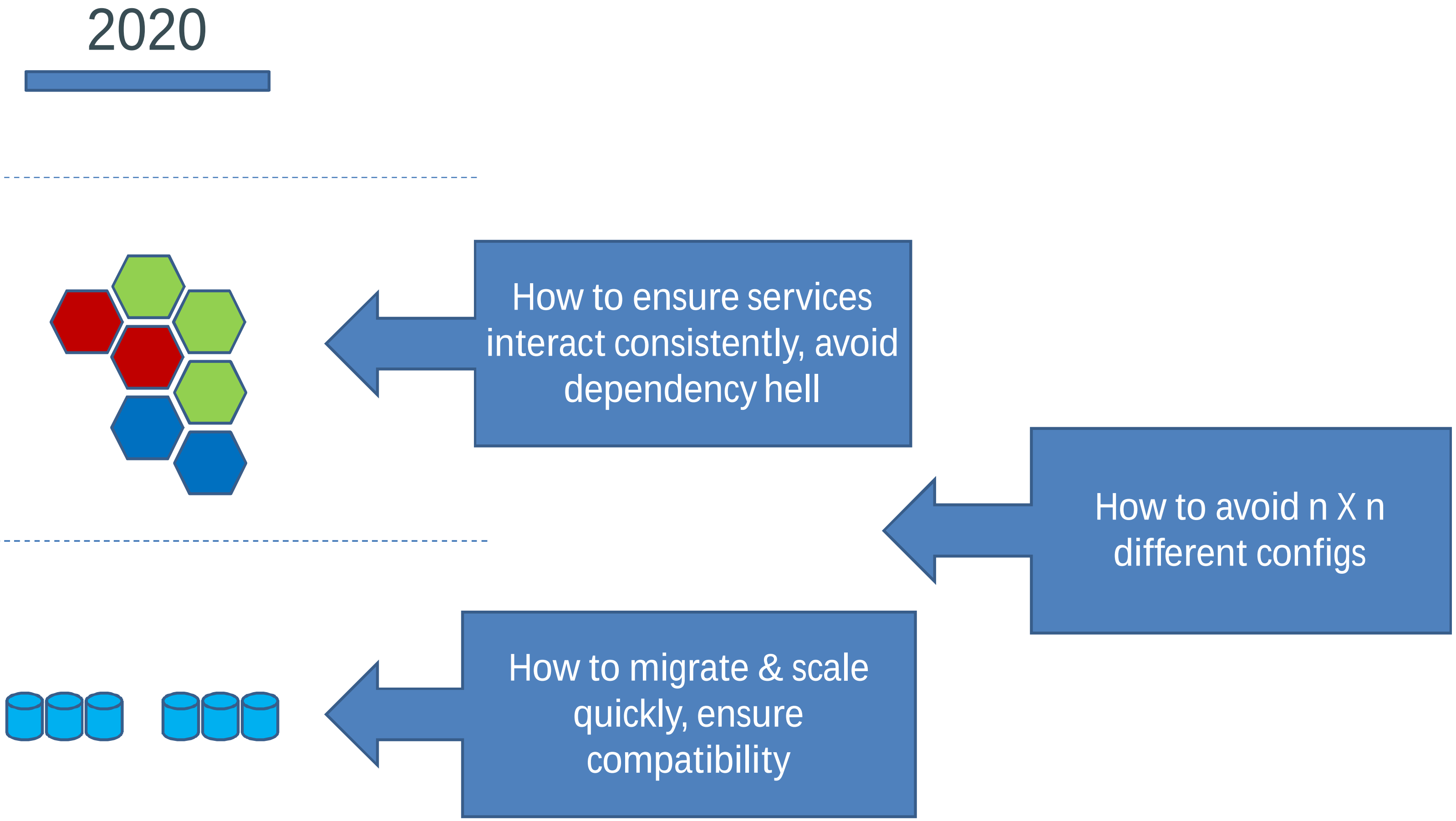
What Is the Problem?

The dependency hell

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



Challenges



2020

Thin app on mobile, tablet

Assembled by developers using best available services

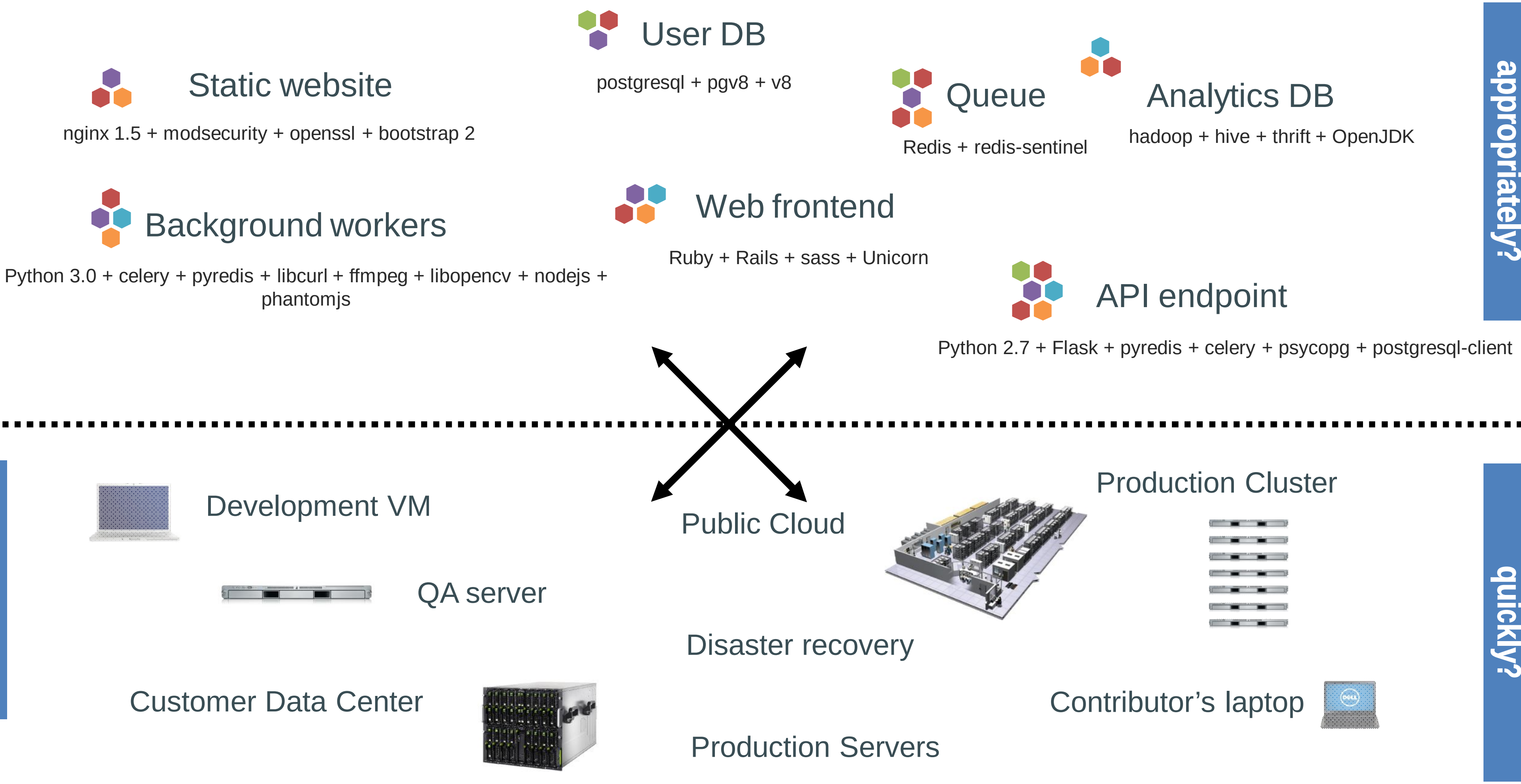
Running on any available set of physical resources (public/private/ virtualized)

How to ensure services interact consistently, avoid dependency hell

How to avoid n X n different configs

How to migrate & scale quickly, ensure compatibility

The Challenge



Multiplicity of Stacks

Static website

nginx 1.5 + modsecurity + openssl + bootstrap 2

User DB

postgresql + pgv8 + v8

Queue

Redis + redis-sentinel

Analytics DB

hadoop + hive + thrift + OpenJDK

Background workers

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

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

QA server

Customer Data Center

Public Cloud

Disaster recovery

Production Servers

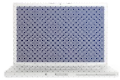
Production Cluster

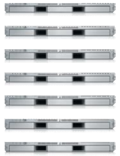
Contributor's laptop

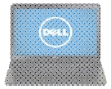
Can I migrate smoothly and quickly?

The Matrix From Hell

	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	?	?	?	?	?	?	?



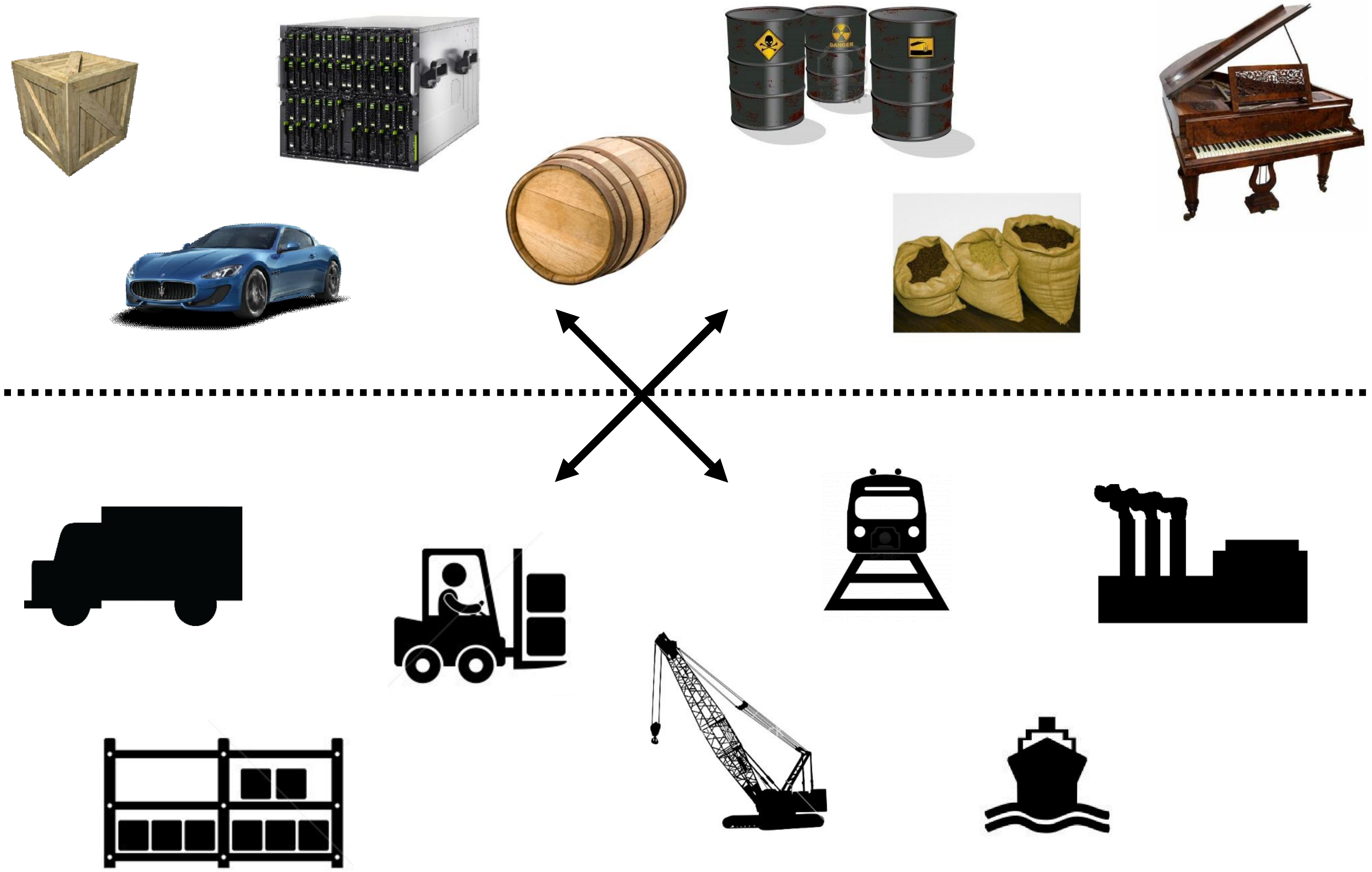




Cargo Transport Pre-1960

Multiplicity of Goods

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



Multipilicity of methods for transporting/storing

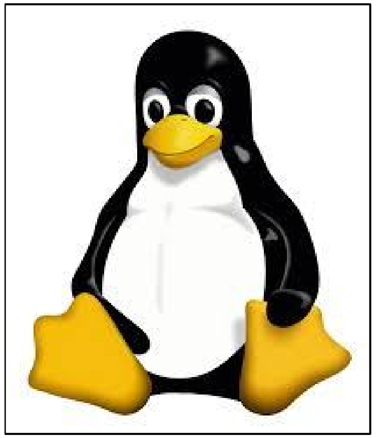
Can I transport quickly and smoothly (e.g. from boat to train to truck)

The Solution

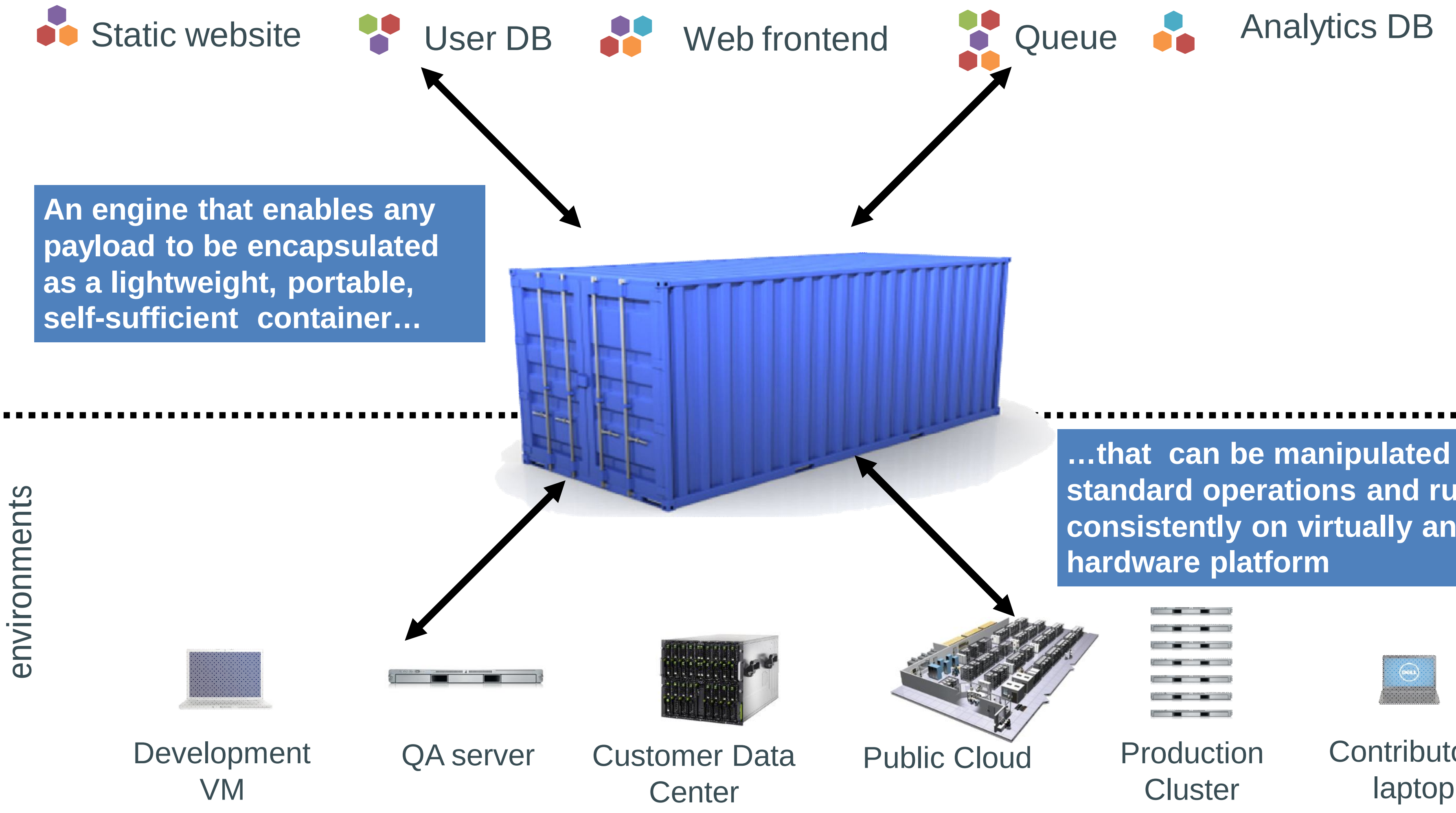
The Solution

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



Multiplicity of Stacks

Static website

User DB

Web frontend

Queue

Analytics DB

Do services and apps interact appropriately?

An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…

…that can be manipulated using standard operations and run consistently on virtually any hardware platform

Multiplicity of hardware environments

Development VM

QA server

Customer Data Center

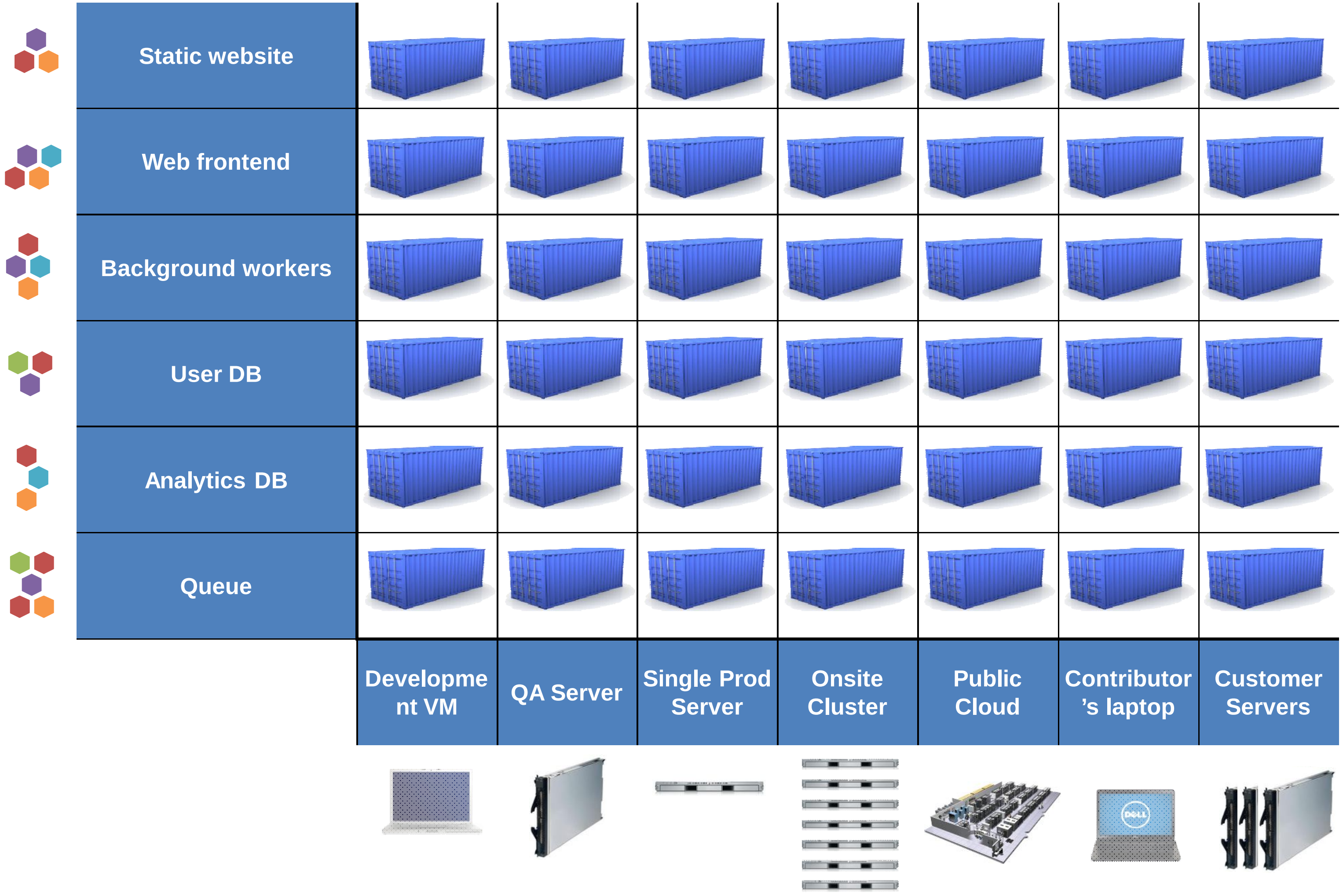
Public Cloud

Production Cluster

Contributor’s laptop

Can I migrate smoothly and quickly

Docker eliminates the matrix from Hell



Static website

Web frontend

Background workers

User DB

Analytics DB

Queue

Development VM

QA Server

Single Prod Server

Onsite Cluster

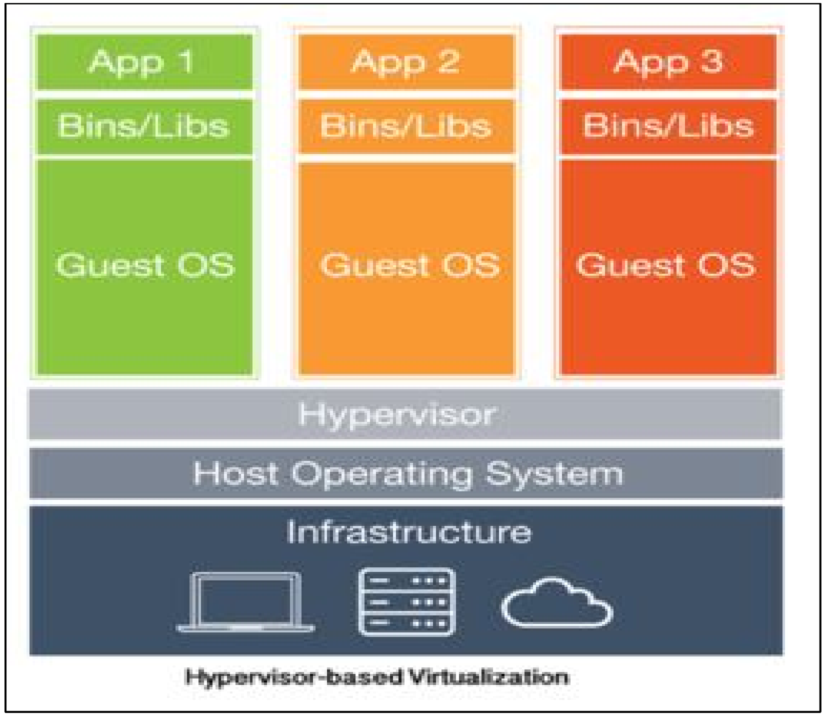
Public Cloud

Contributor's laptop

Customer Servers

VMs vs. Containers

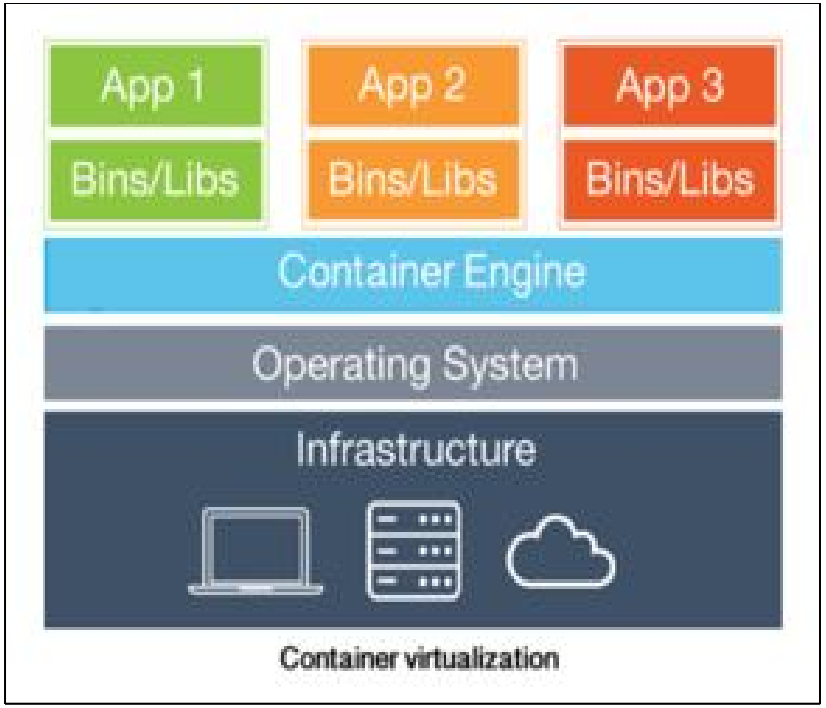
VMs



Hypervisor-based Virtualization

docker

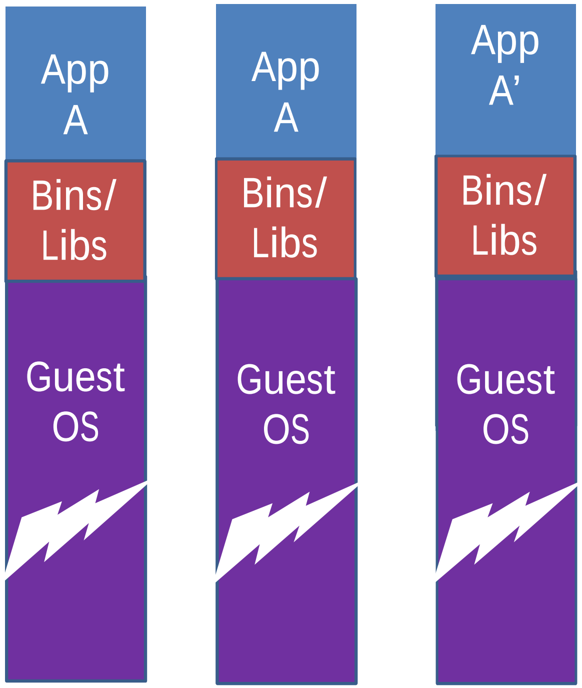
Containers



Container virtualization

Why are containers lightweight?

VMs



App A

Bins/ Libs

Guest OS

App A

Bins/ Libs

Guest OS

App A’

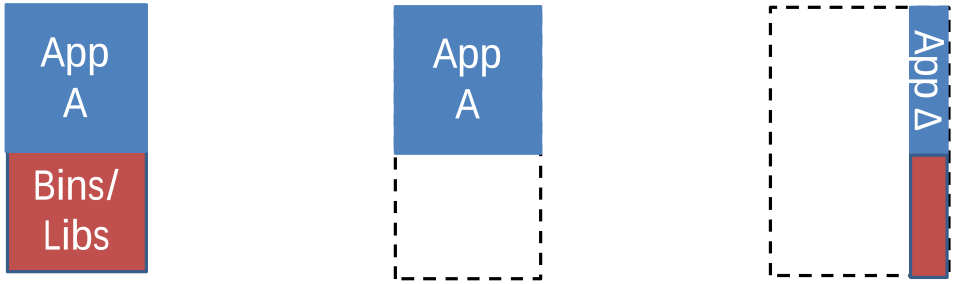
Bins/ Libs

Guest OS

VMs

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

Containers



App A

Bins/ Libs

App A

App Δ

Original App

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

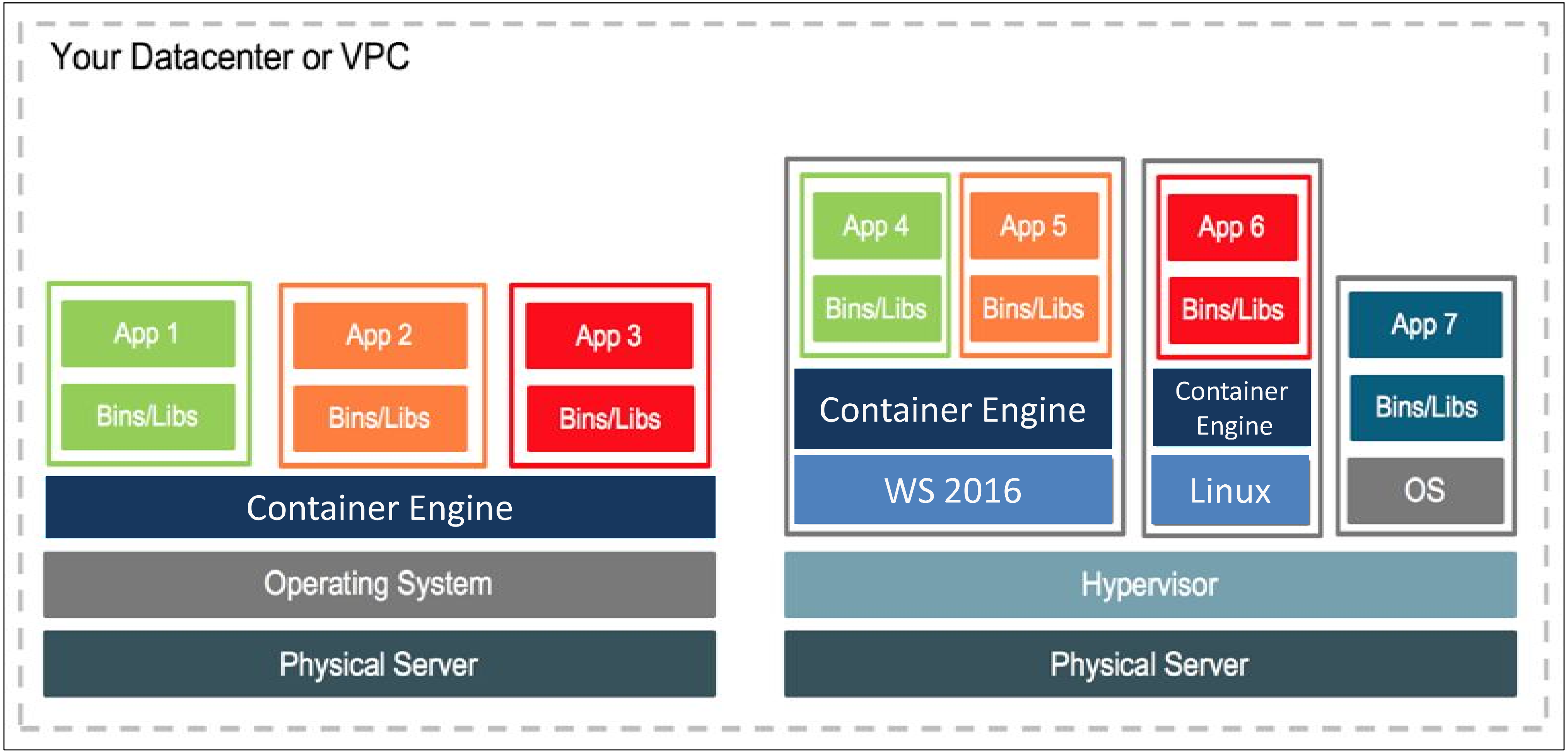
Copy of App

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



docker

VMs vs. Containers

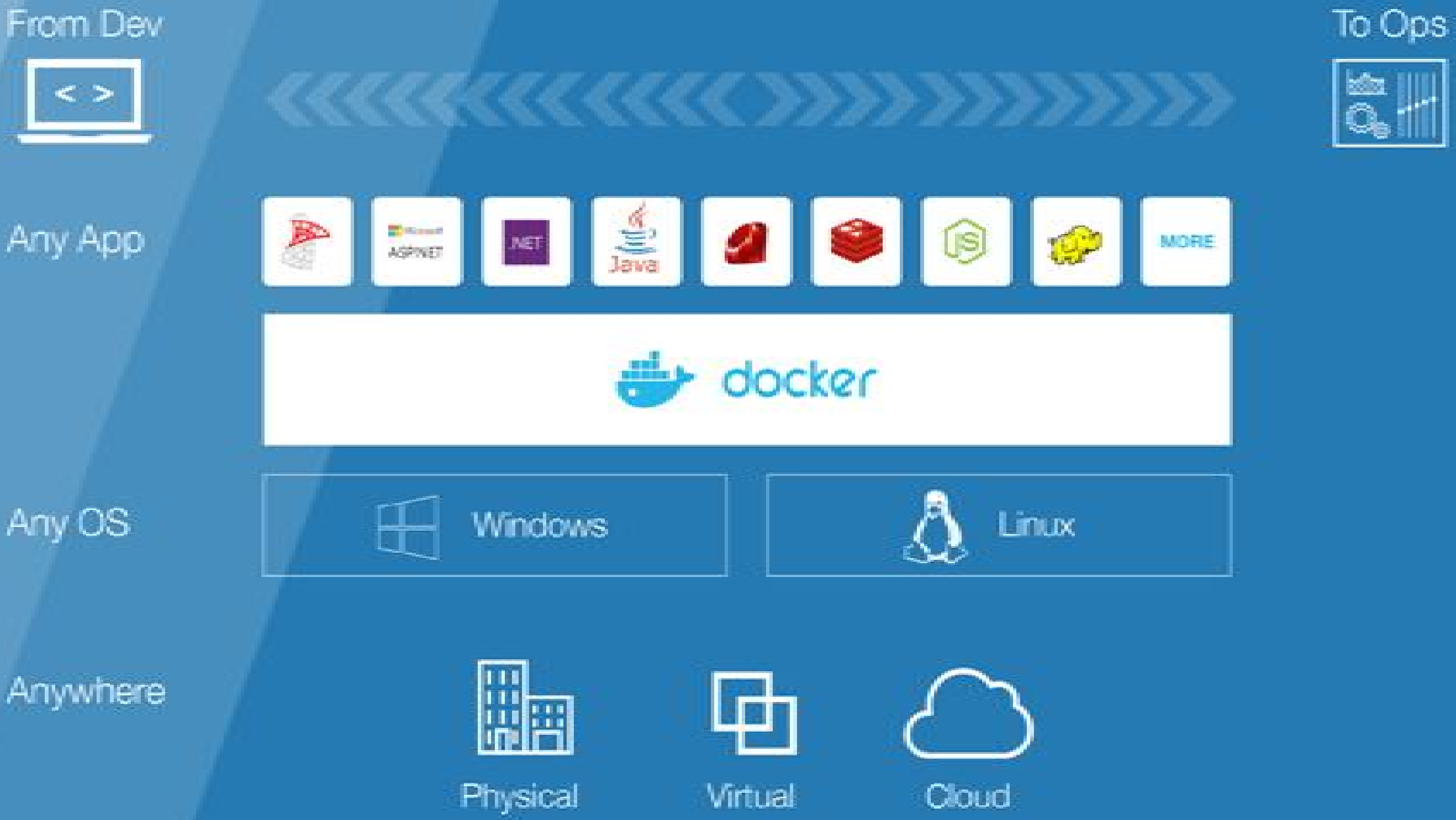
	VMs	Containers
Security	More isolated	Less isolated
Size	GBs	MBs
Provision	Mins	Secs
OS	More flexible	Less flexible

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

MORE

docker

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?

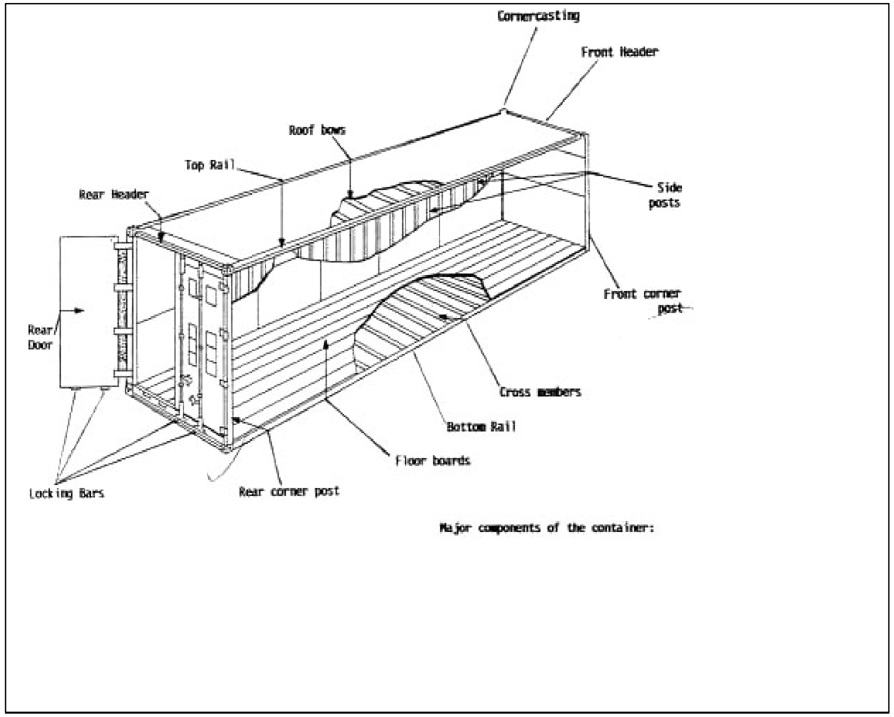
Why it works - separation of concerns

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



Major components of the container:

More technical explanation

WHY

- **Run everywhere**
  - o Regardless of kernel version
  - o Regardless of host distro
  - o Physical or virtual, cloud or not
  - o Container and host architecture must match*

- **Run anything**
  - o If it can run on the host, it can run in the container

WHAT

- **High Level—It's a lightweight VM**
  - o Own process space
  - o Own network interface
  - o Can run stuff as root
  - o Can have its own /sbin/init (different from host)
  - o <<machine container>>

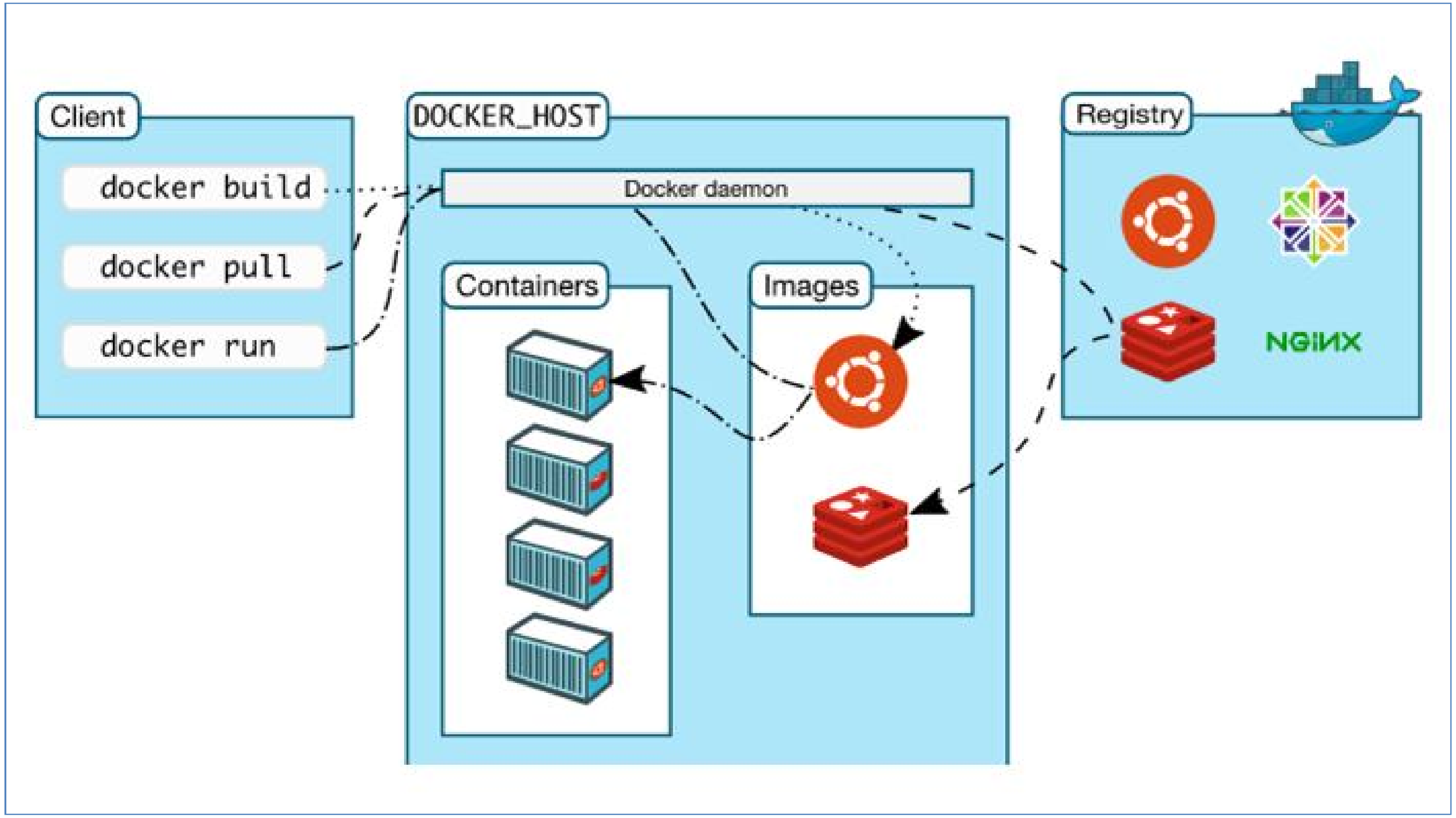
- **Low Level—It's chroot on steroids**
  - o Can also not have its own /sbin/init
  - o Container=isolated processes
  - o Share kernel with host
  - o No device emulation (neither HVM nor PV) from host
  - o <<application container>>

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 hours if not **days**. Meaning the test cycle itself could take days if not **weeks**.

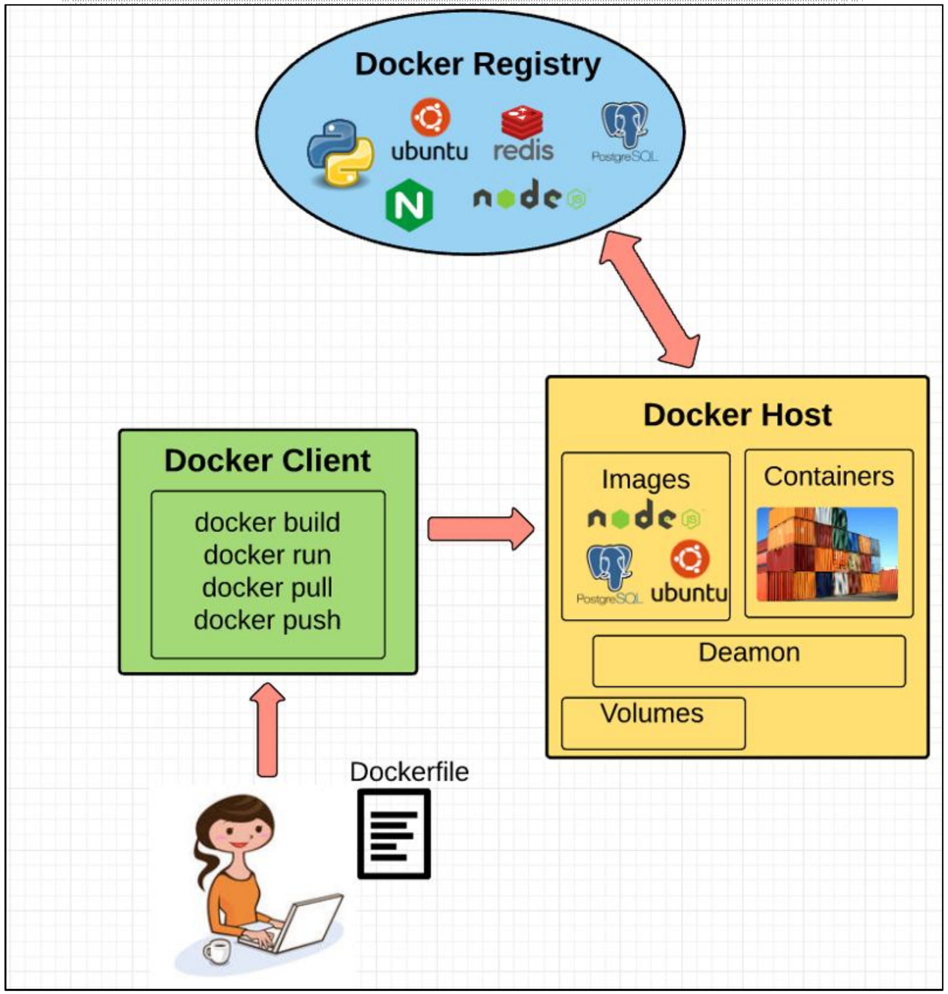
By contrast, the same application running inside of a Docker container can be started in less than half a **second**. Standing up 1,000 containers becomes trivial. Test cycle times can be slashed from days to **hours**. This can translate into measurable savings as well as increased agility.

Docker Architecture

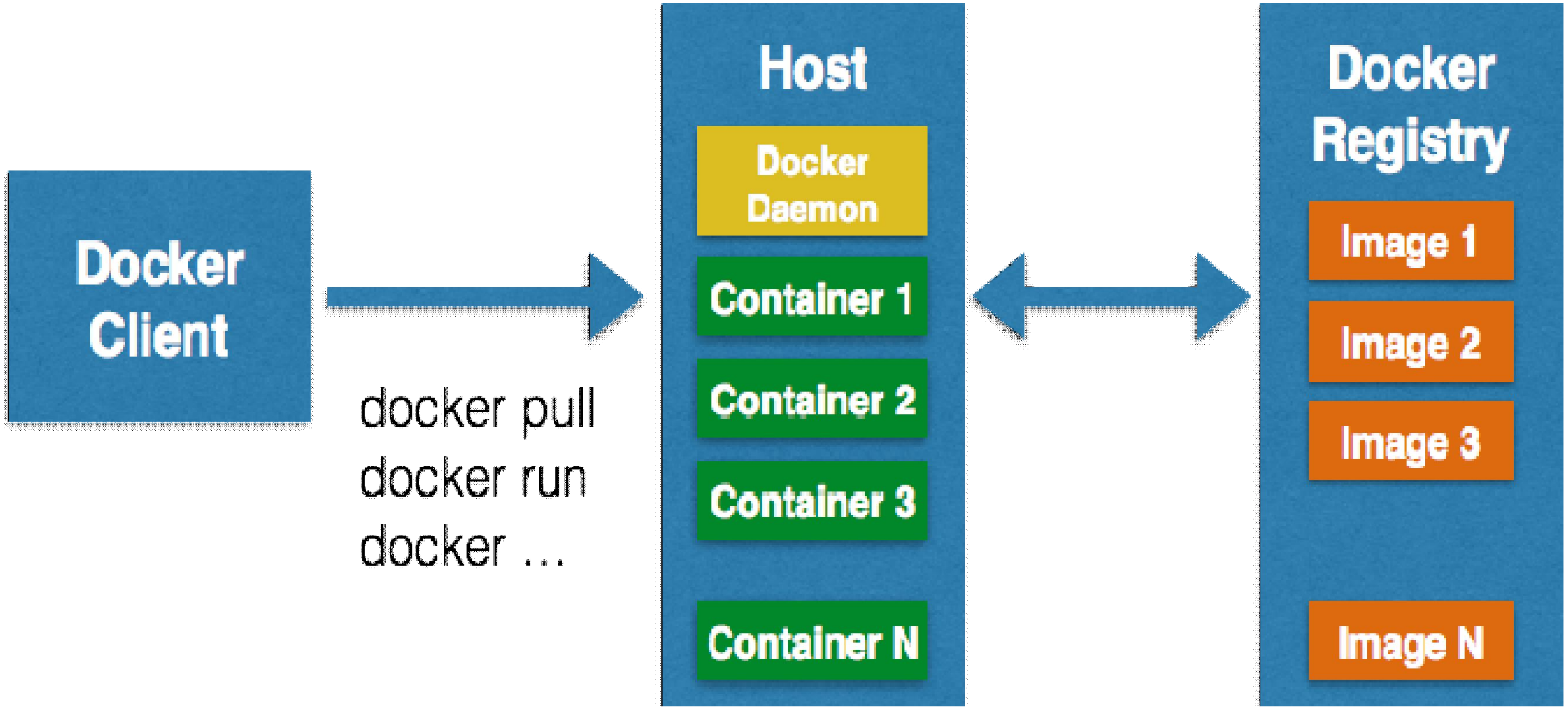
Docker Architecture



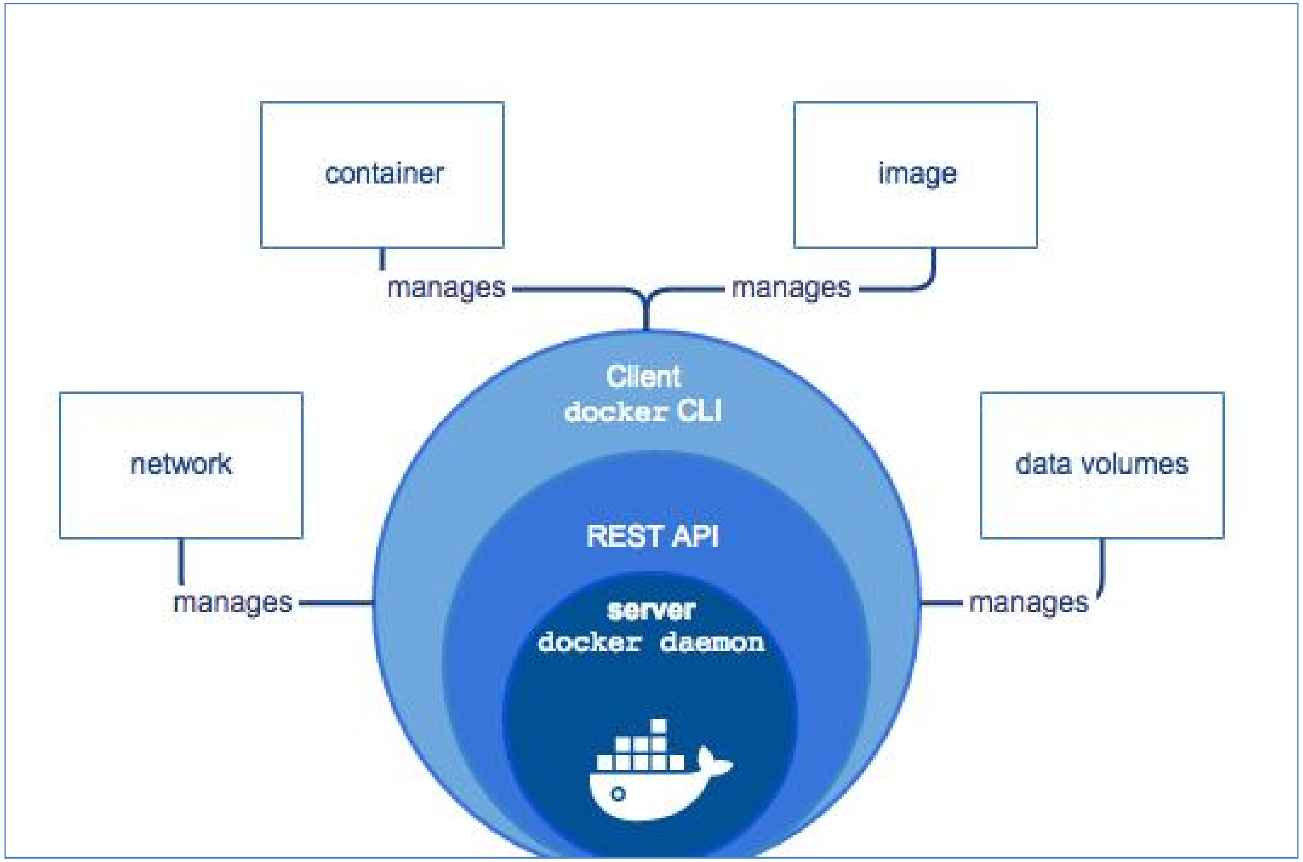
Docker Architecture



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

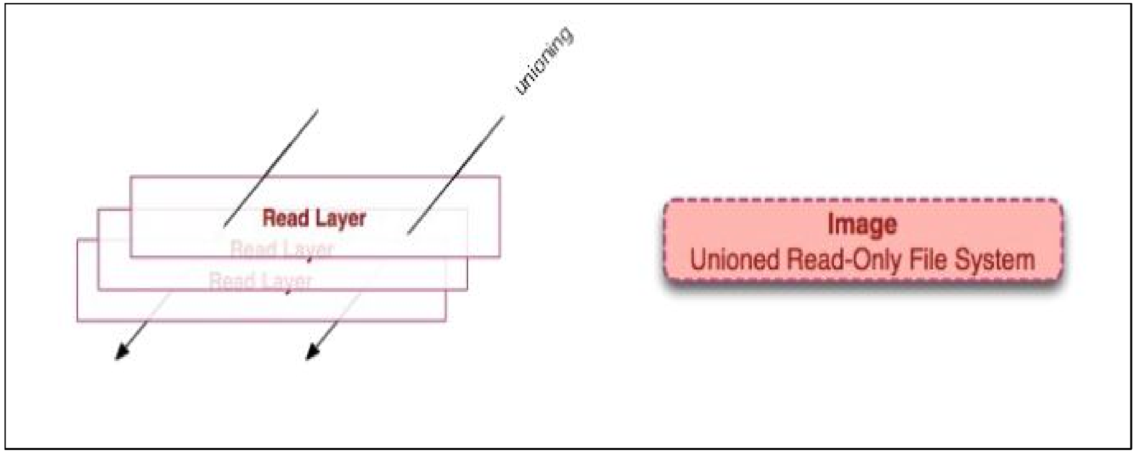
Creates, ships and runs Docker containers deployable on a physical or virtual, host locally, in a datacenter or cloud service provider

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 you local registry
- Every image starts from base image
- Include:
  - *Application*
  - *Dependencies*
  - *Libraries*
  - *Binaries*
  - *Configuration files*



Containers

- Isolated application platform
- Containers everything needed to run you application
- Based on one or more images
- Docker containers launched from Docker image
- When D d-write layer or

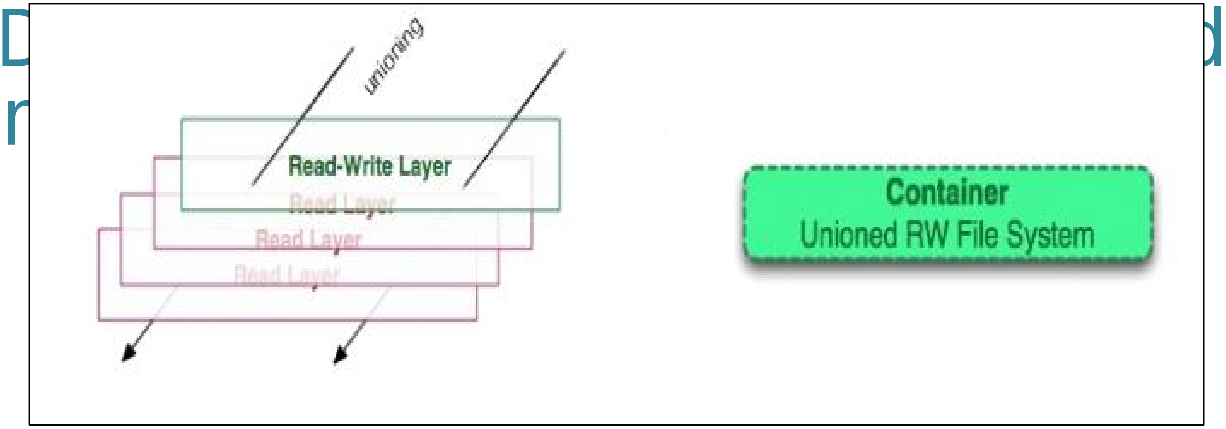
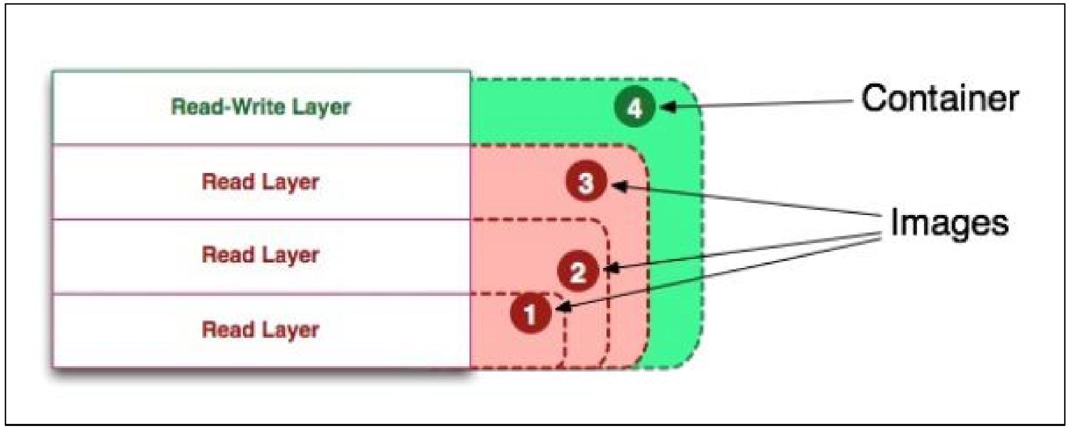
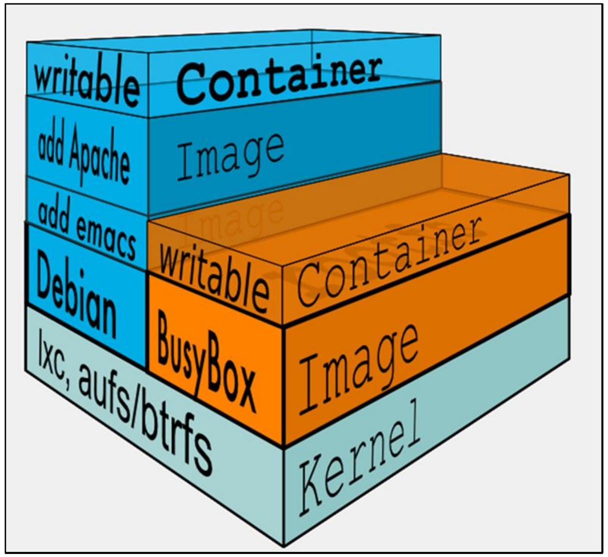


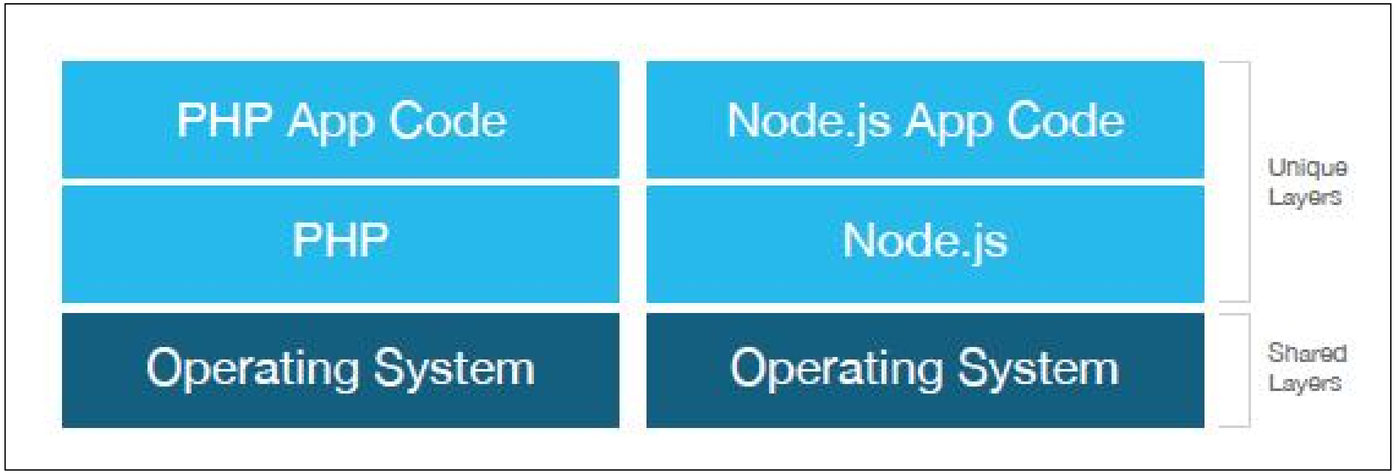
Image vs. Container

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



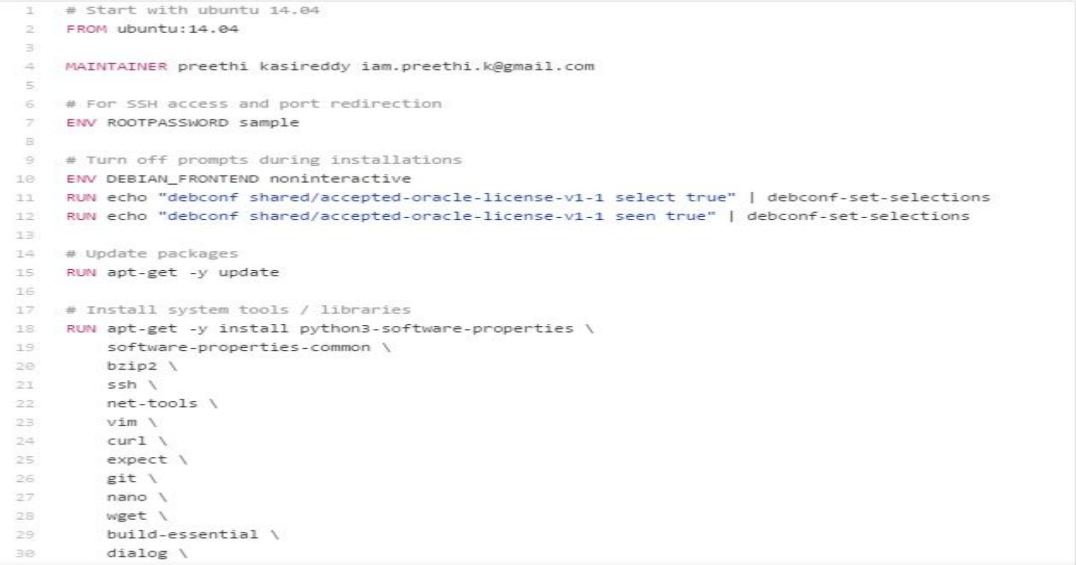


Docker will not only share the base image between containers, but it will also share the same layers between different images.

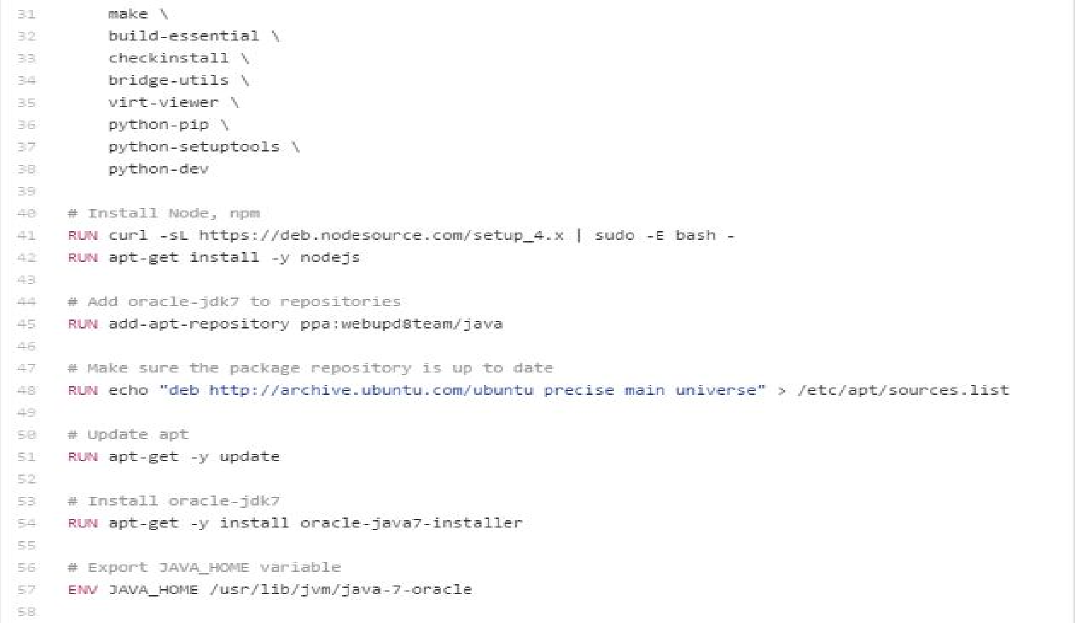


Docker File

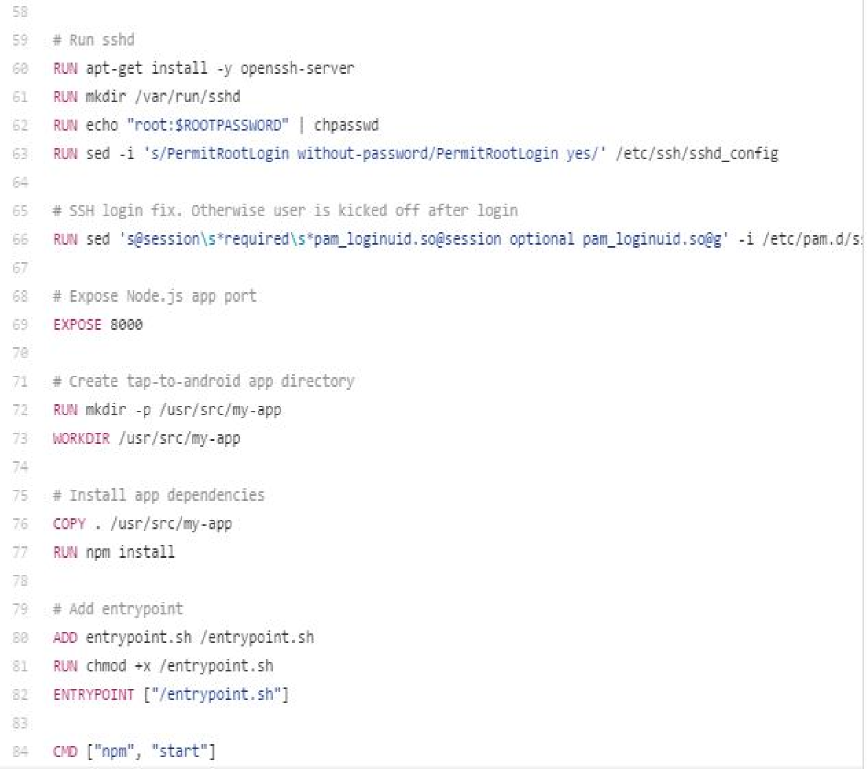
- Dockerfile is instructions to build Docker image
  - How to run commands
  - Add files or directories
  - Create environment variables
  - What process to run when launching container
- Result from building Dockerfile is Docker image



```
# Start with ubuntu 14.04
FROM ubuntu:14.04

MAINTAINER preethi kasireddy iam.preethi.k@gmail.com

# For SSH access and port redirection
ENV ROOTPASSWORD sample

# Turn off prompts during installations
ENV DEBIAN_FRONTEND noninteractive
RUN echo "debconf shared/accepted-oracle-license-v1-1 select true" | debconf-set-selections
RUN echo "debconf shared/accepted-oracle-license-v1-1 seen true" | debconf-set-selections

# Update packages
RUN apt-get -y update

# Install system tools / libraries
RUN apt-get -y install python3-software-properties \
    software-properties-common \
    bzip2 \
    ssh \
    net-tools \
    vim \
    curl \
    expect \
    git \
    nano \
    wget \
    build-essential \
    dialog \
```

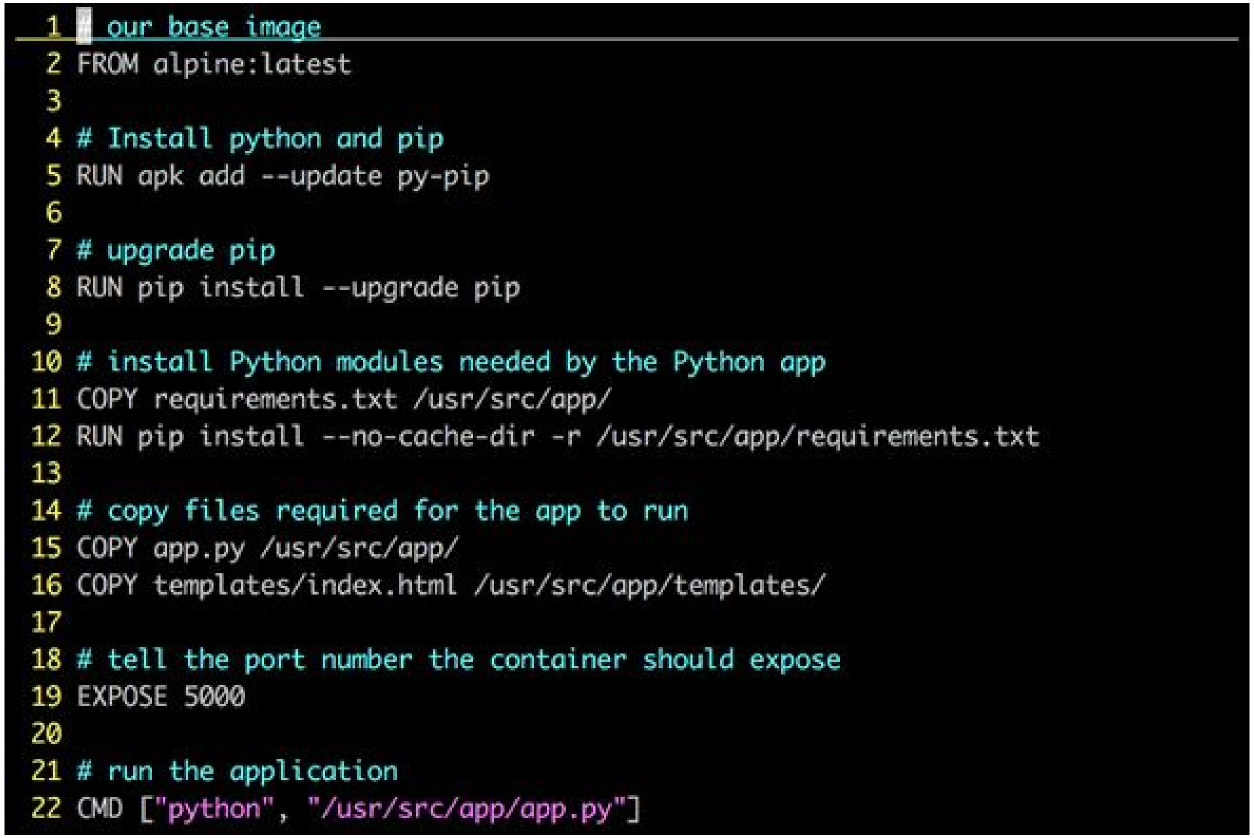


```
    make \
    build-essential \
    checkinstall \
    bridge-utils \
    virt-viewer \
    python-pip \
    python-setuptools \
    python-dev

# Install Node, npm
RUN curl -sL https://deb.nodesource.com/setup_4.x | sudo -E bash -
RUN apt-get install -y nodejs

# Add oracle-jdk7 to repositories
RUN add-apt-repository ppa:webupd8team/java

# Make sure the package repository is up to date
RUN echo "deb http://archive.ubuntu.com/ubuntu precise main universe" > /etc/apt/sources.list

# Update apt
RUN apt-get -y update

# Install oracle-jdk7
RUN apt-get -y install oracle-java7-installer

# Export JAVA_HOME variable
ENV JAVA_HOME /usr/lib/jvm/java-7-oracle

```



```

# Run sshd
RUN apt-get install -y openssh-server
RUN mkdir /var/run/sshd
RUN echo "root:$ROOTPASSWORD" | chpasswd
RUN sed -i 's/PermitRootLogin without-password/PermitRootLogin yes/' /etc/ssh/sshd_config

# SSH login fix. Otherwise user is kicked off after login
RUN sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -i /etc/pam.d/s

# Expose Node.js app port
EXPOSE 8000

# Create tap-to-android app directory
RUN mkdir -p /usr/src/my-app
WORKDIR /usr/src/my-app

# Install app dependencies
COPY . /usr/src/my-app
RUN npm install

# Add entrypoint
ADD entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]

CMD ["npm", "start"]
```

Dockerfile – Linux Example

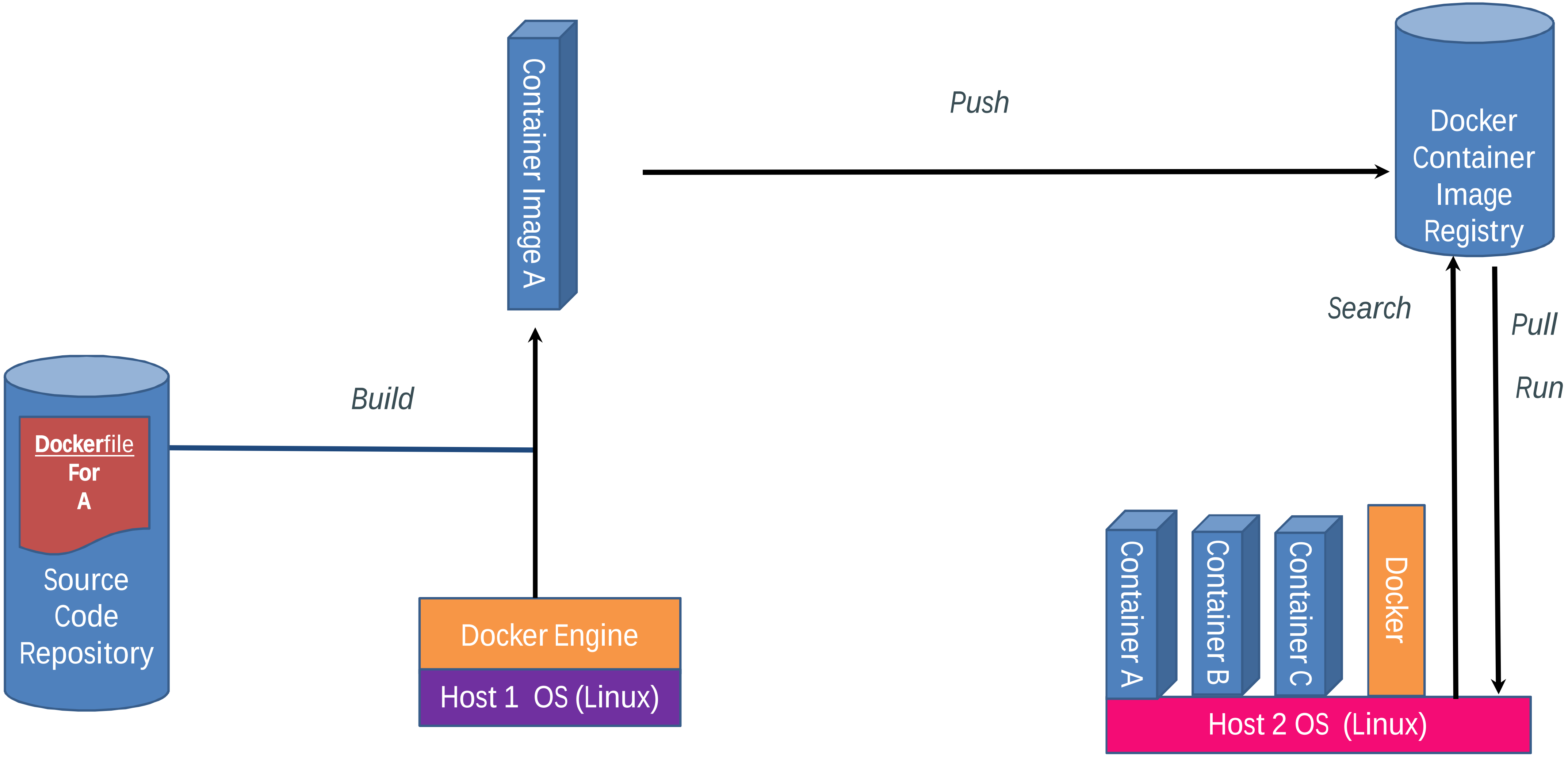


```
# our base image
FROM alpine:latest

# Install python and pip
RUN apk add --update py-pip

# upgrade pip
RUN pip install --upgrade pip

# install Python modules needed by the Python app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

# copy files required for the app to run
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/

# tell the port number the container should expose
EXPOSE 5000

# run the application
CMD ["python", "/usr/src/app/app.py"]
```

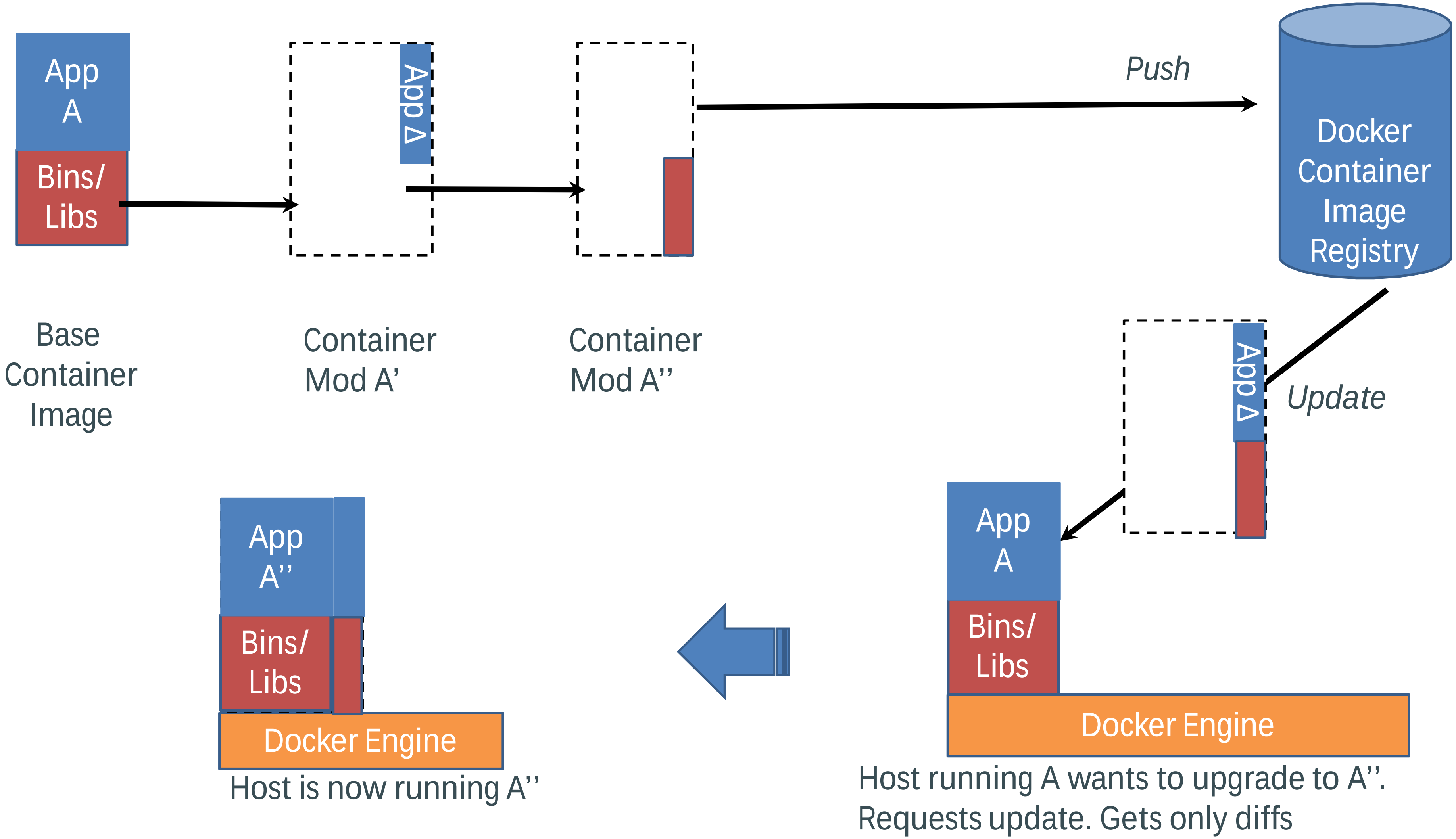
- Instructions on how to build a Docker image
- Looks very similar to “native” commands
- Important to optimize your Dockerfile

Docker Workflow

What are the basics of the Docker system?



Changes and Updates



Host is now running A''

Host running A wants to upgrade to A''.
Requests update. Gets only diffs

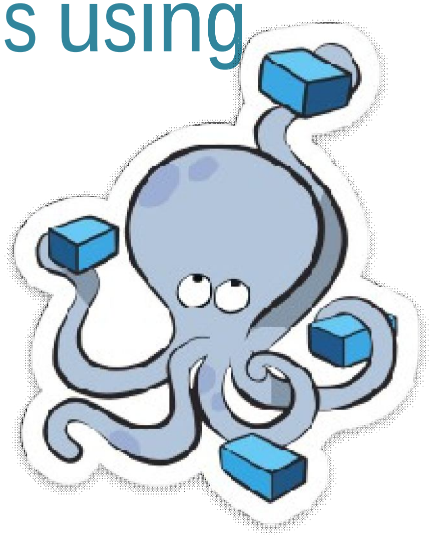
Docker Orchestration

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!

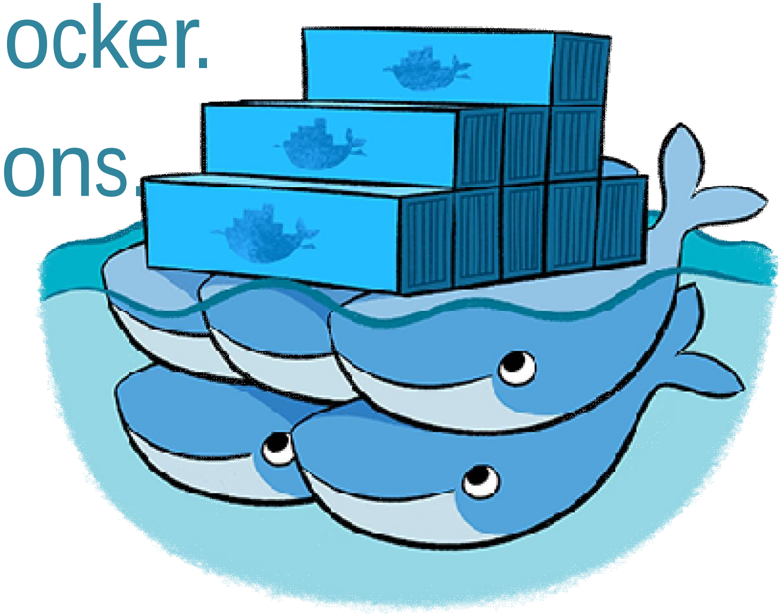
Docker Compose

- 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.



docker

Thank You!