# The Docker Way

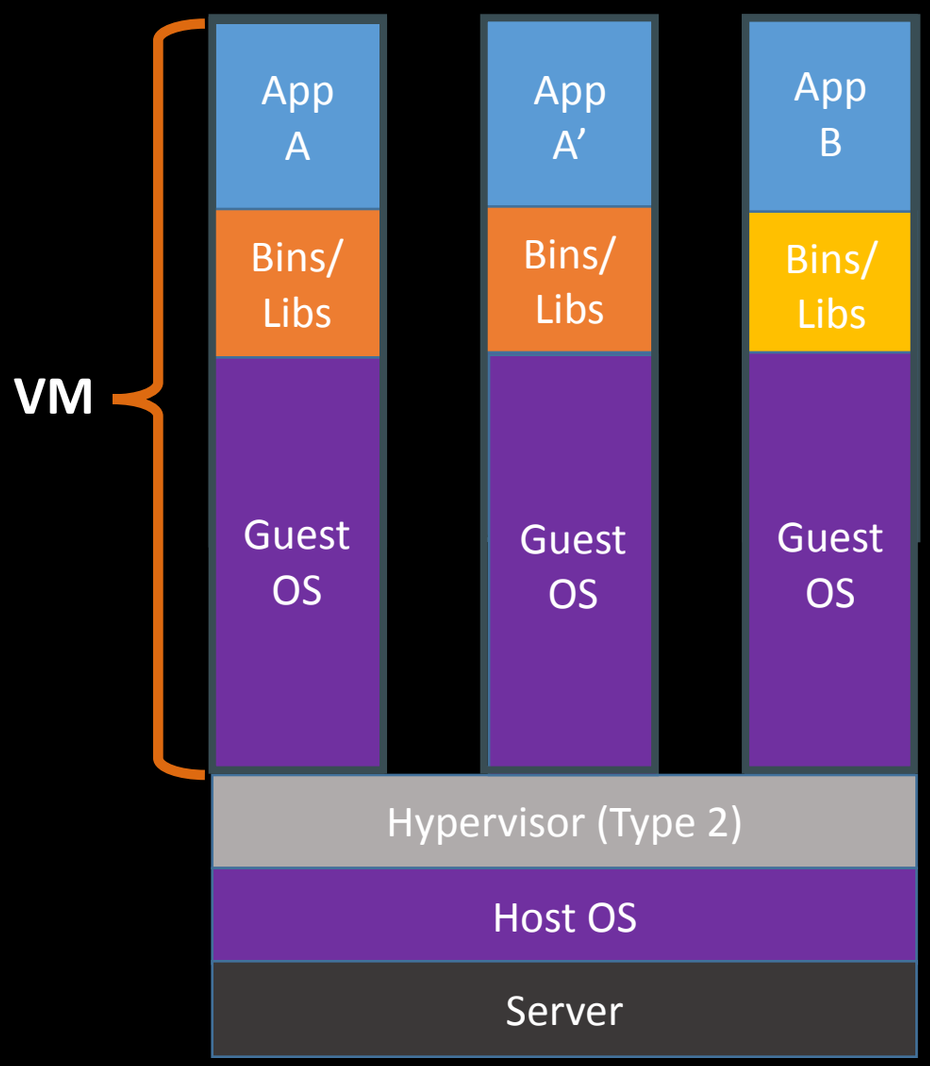
xuli@qiniu.com

# Agenda

- Docker: What, Why and How
- Story: Qiniu meet Docker
- Q & A

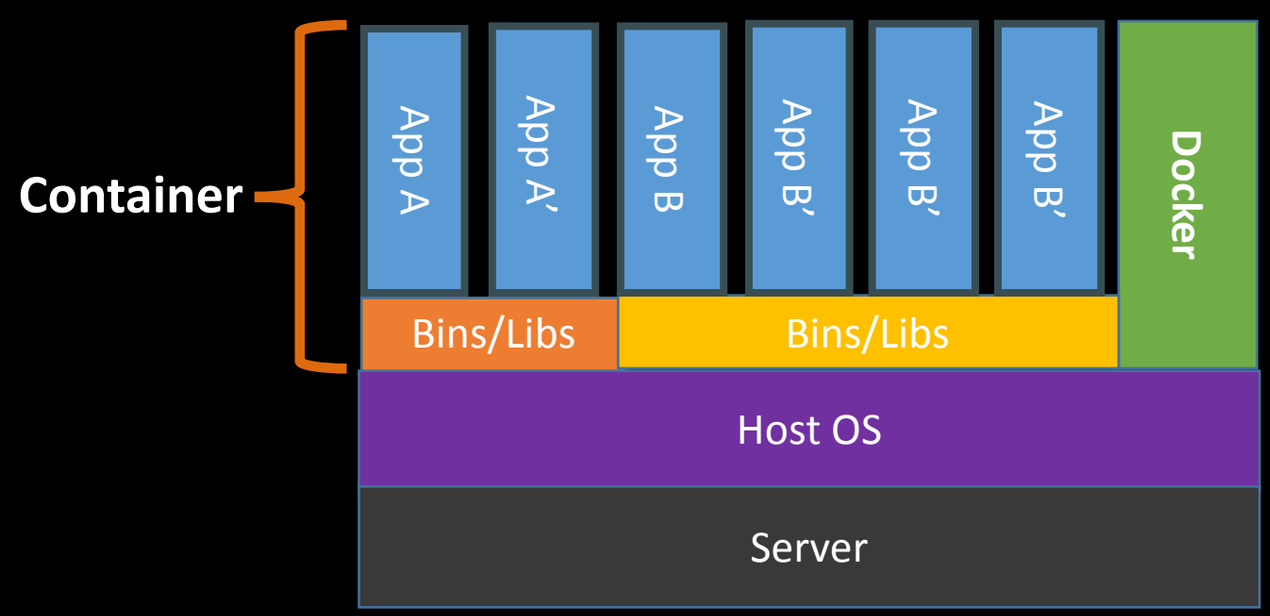
# Container do Virtualization

# Container vs. VM




Containers are isolated, but share OS and, where appropriate, bins/libraries

…result is significantly faster deployment, much less overhead, easier migration, faster restart



- The same see and shared for each Linux-process:
  - Linux Kernel
  - File System
  - Network System
  - PID, UID, IPC, etc..
  - Memory, Disk, CPU, etc..

# Virtualization

- Isolated:
  - FS, net, pid, uid, uts, mnt, ipc namespace
- Constrained:
  - Memory, CPU, Network IO, Disk IO, Disk Space

The <u>process groups</u> that meet the above restrictions are called "**lightweight virtual machine**" or **Container**

# Why Docker ?

# The Challenge

Multiplicity of Stacks

Static website

nginx 1.5 + modsecurity + openssl + bootstrap 3

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

QA server

Customer Data Center

Public Cloud

Disaster recovery

Production Servers

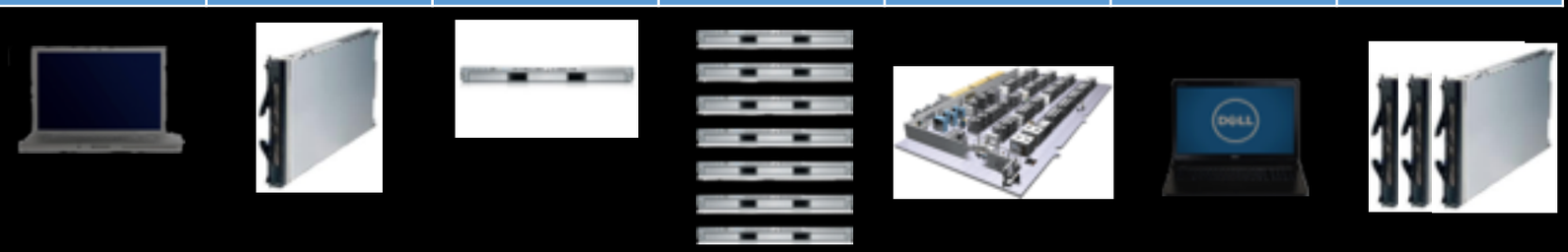
Production Cluster

Contributor's laptop

Can I migrate smoothly and quickly?

# N x N compatibility nightmare

| | 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** | ? | ? | ? | ? | ? | ? | ? |

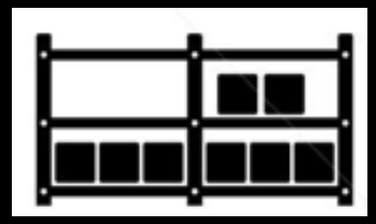


# An Inspiration.. and history: Cargo Transport Pre-1960

Multiplicity of Goods

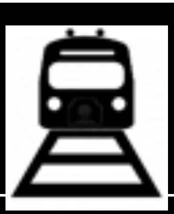
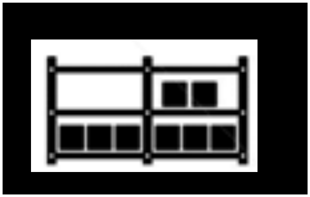
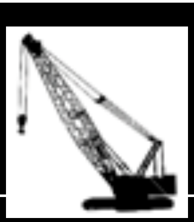
Do I worry about how goods interact (e.g. coffee beans next to spices)

Multipilicity of methods for transporting/storing



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

# Also a matrix from hell

| | | | | | | | |
|---|---|---|---|---|---|---|---|
|  | ? | ? | ? | ? | ? | ? | ? |
|  | ? | ? | ? | ? | ? | ? | ? |
|  | ? | ? | ? | ? | ? | ? | ? |
|  | ? | ? | ? | ? | ? | ? | ? |
|  | ? | ? | ? | ? | ? | ? | ? |
|  | ? | ? | ? | ? | ? | ? | ? |
| |  |  |  |  |  |  |  |

# Solution: Intermodal Shipping Container

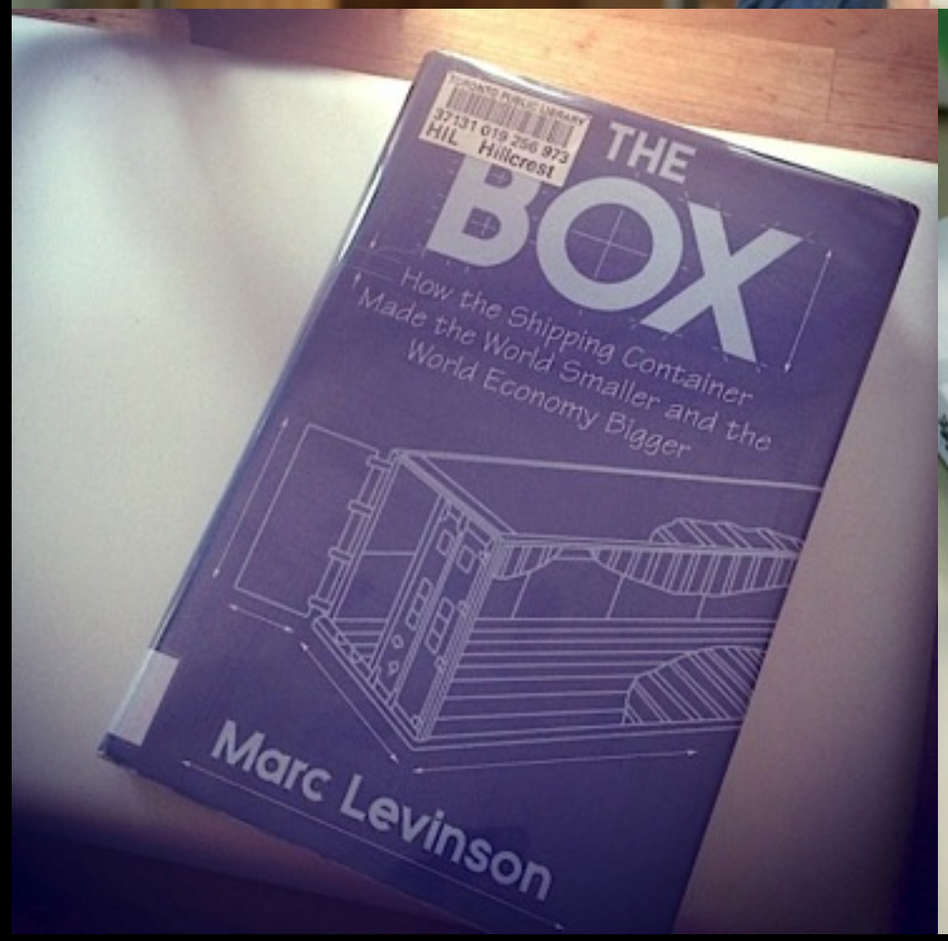
Multiplicity of Goods


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)

http://www.gatesnotes.com/About-Bill-Gates/Best-Books-2013


# A shipping container system for code

Multiplicity of Stacks

Static website

User DB

Web frontend

Queue

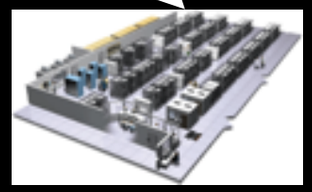
Analytics DB

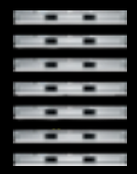
Do services and apps interact appropriately?

**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**

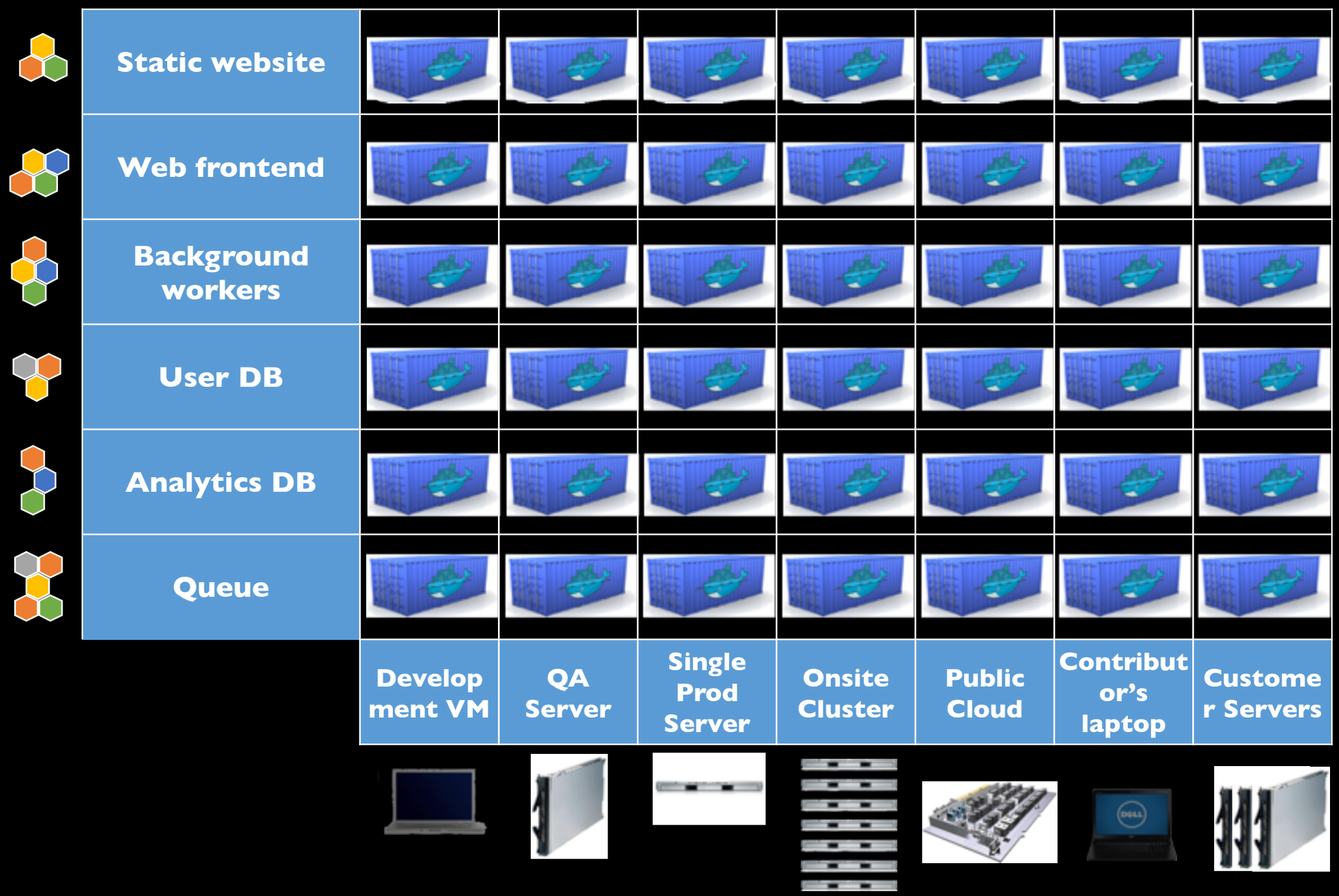
Multiplicity of hardware environments





Can I migrate smoothly and quickly

# This eliminated the N x N problem



# Build, Ship, Run

“build once, run anywhere”

*– developers*

“configure once, run anything”

*– operations*

# What can I use Docker for?

- Continuous Integration / Deployment
- Packaging and deploying applications
- Build your own PAAS
- Deploy applications at hyperscale

# Work with Puppet or Chef

- Chef and Puppet are state management  tools
- Docker images are version contronlled and layered
- Small, self-contained and lightweight

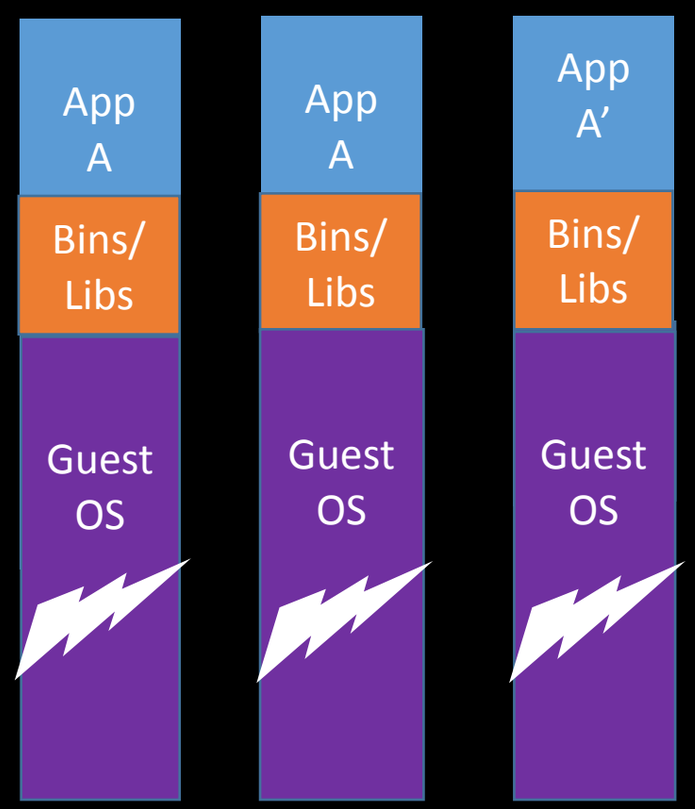
standardized,
interoperable,
automatable

# Why not virtual machines?

- Speed of deployment
- Portability
- Size aka cached layering FTW
- Density & Performance
- Cost

# Why are Docker containers lightweight?

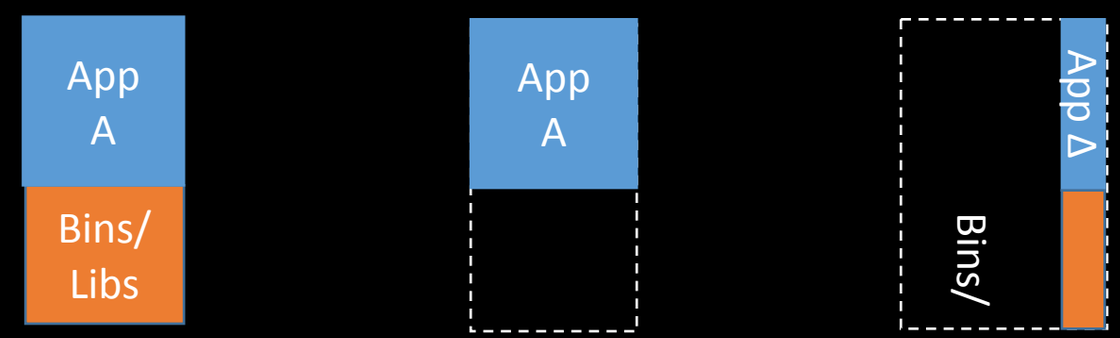
**VMs**




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

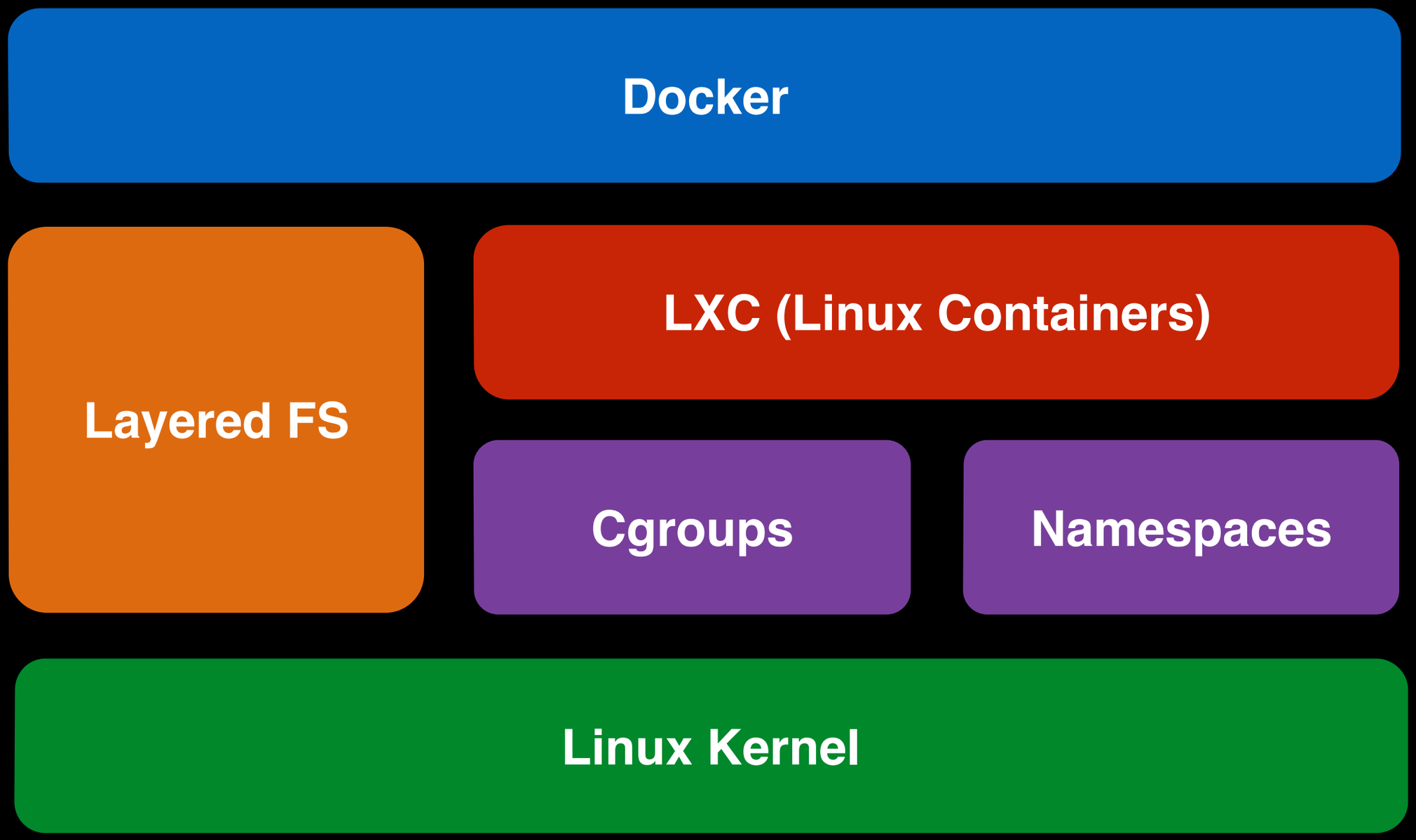
Copy on write capabilities allow us to only save the diffs Between container A and container A'

VM for IAAS,
Container for PAAS

# Technology Stack

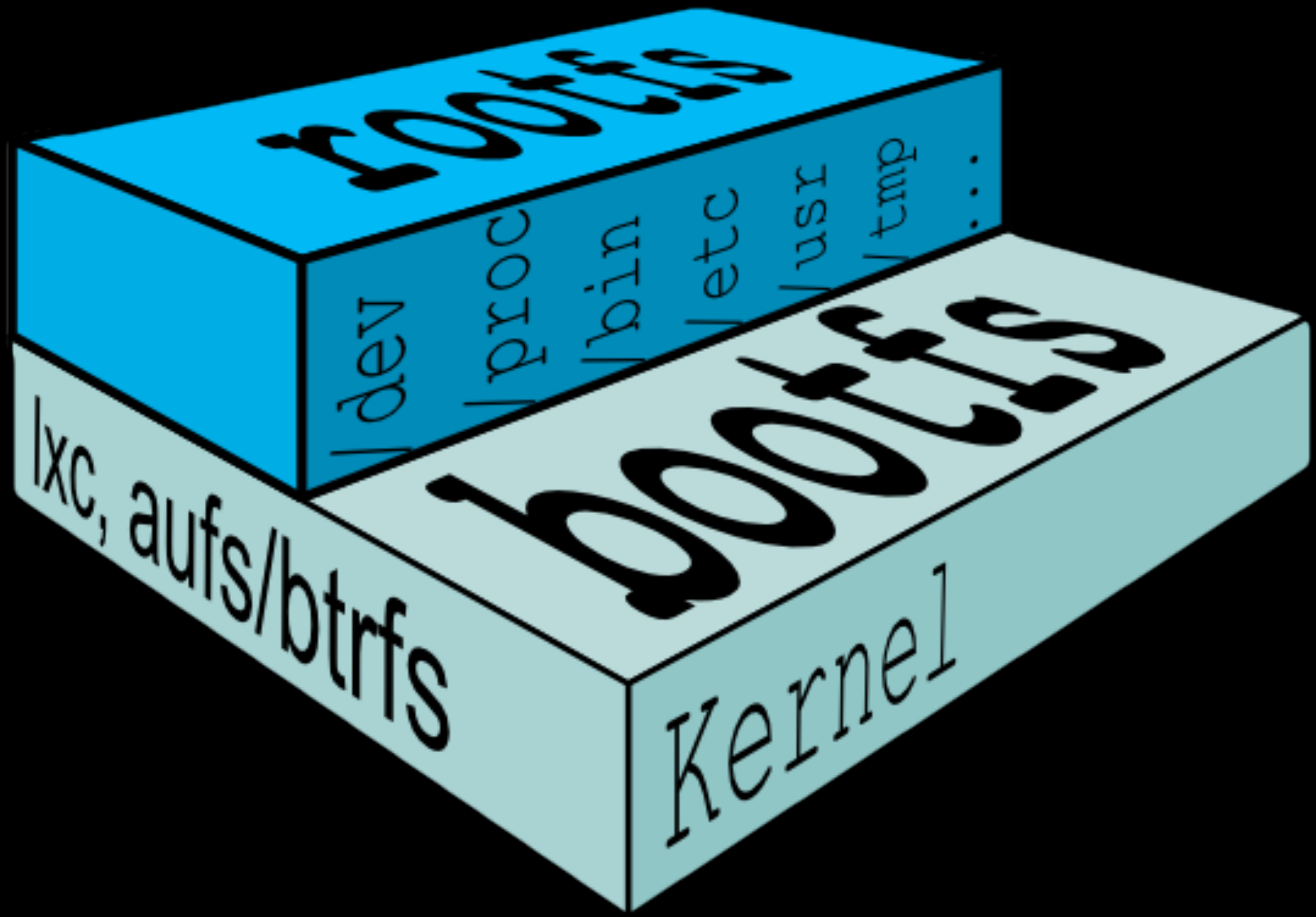
- Runs on most Linux distros
- Boot2Docker for OSX and windows
- Uses Linux Kernel features
- Storage is provided by Union File systems
- Container format

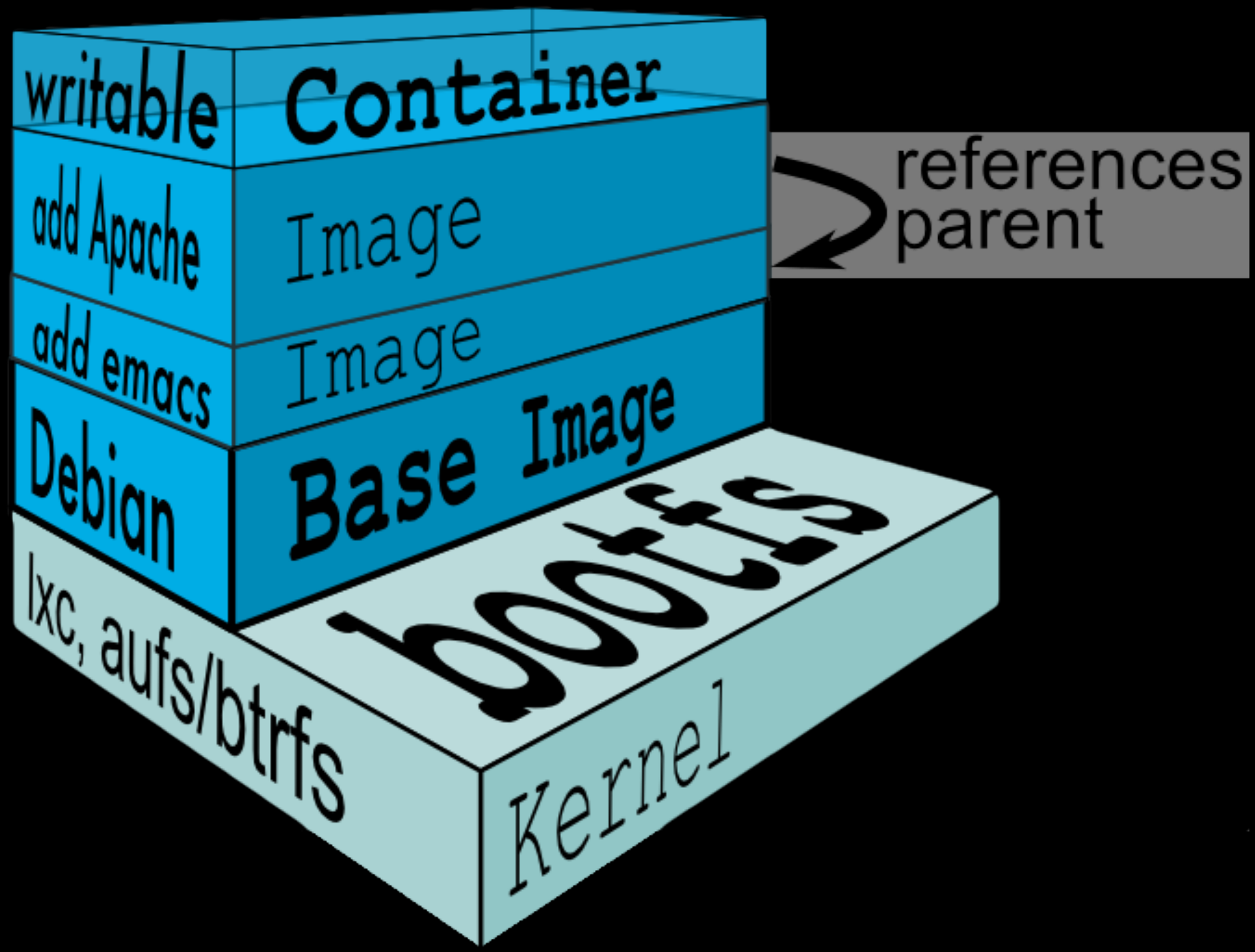
# Docker Architecture

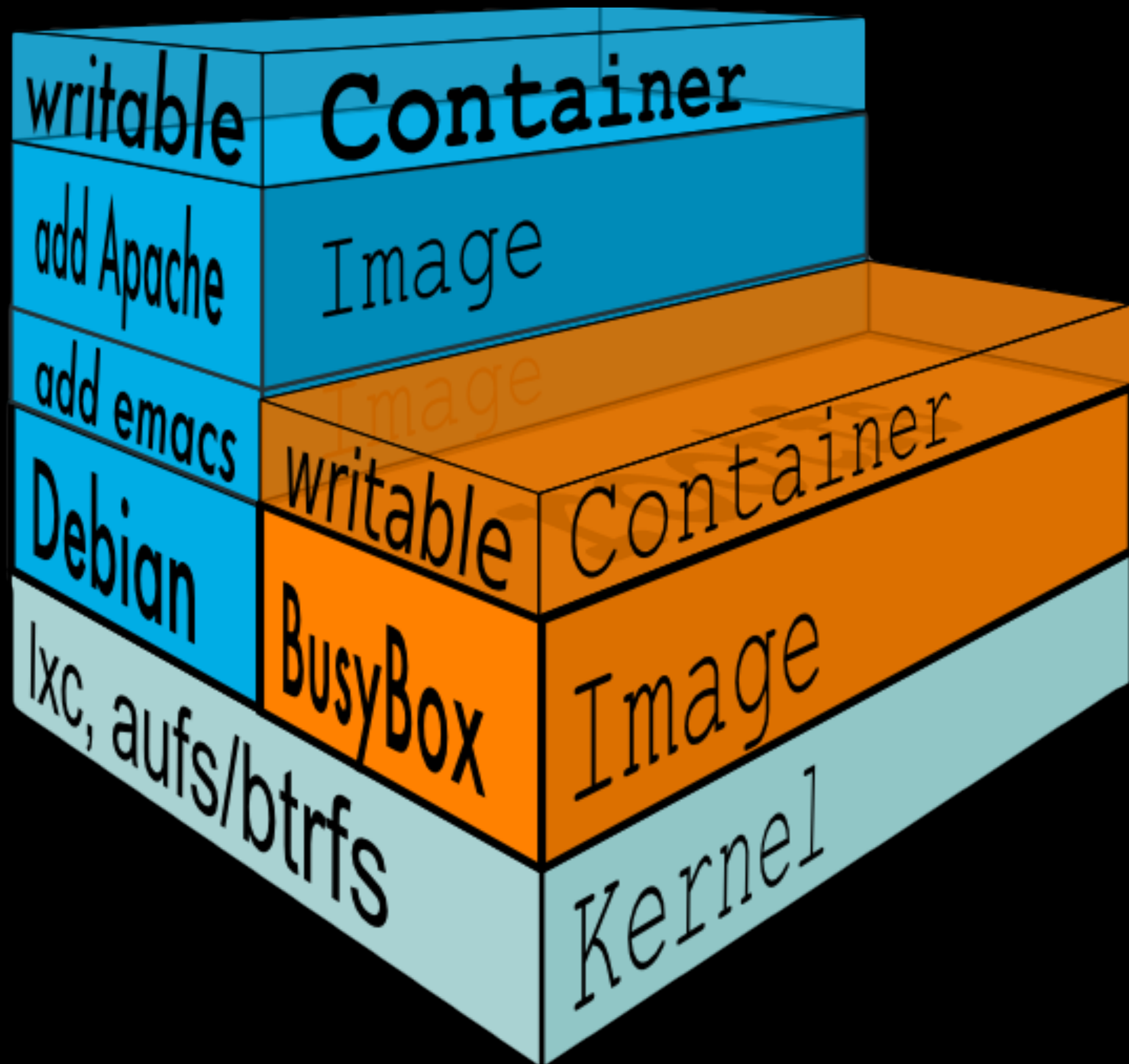


# Docker Basics

- Image
- Container
- Registry (like http://index.docker.io)


rootfs
/dev
/proc
/bin
/etc
/usr
/tmp
...
lxc, aufs/btrfs
bootfs
Kernel


writable
Container
add Apache
Image
add emacs
Image
Debian
Base Image
lxc, aufs/btrfs
bootfs
Kernel
references
parent


writable
Container
add Apache
Image
add emacs
Image
Debian
writable
Container
BusyBox
Image
lxc, aufs/btrfs
Kernel

# Images

- Images are the source of containers
- Every container is launched from an image
- Images are "built" and layered
- Images are stored in a registry

# Layers

- Each image is made up of layers
- Like Linux the bottom layer is a root file system
- Docker uses union file system mounting
- Top layer is writeable and created when a container is launched

# Containers

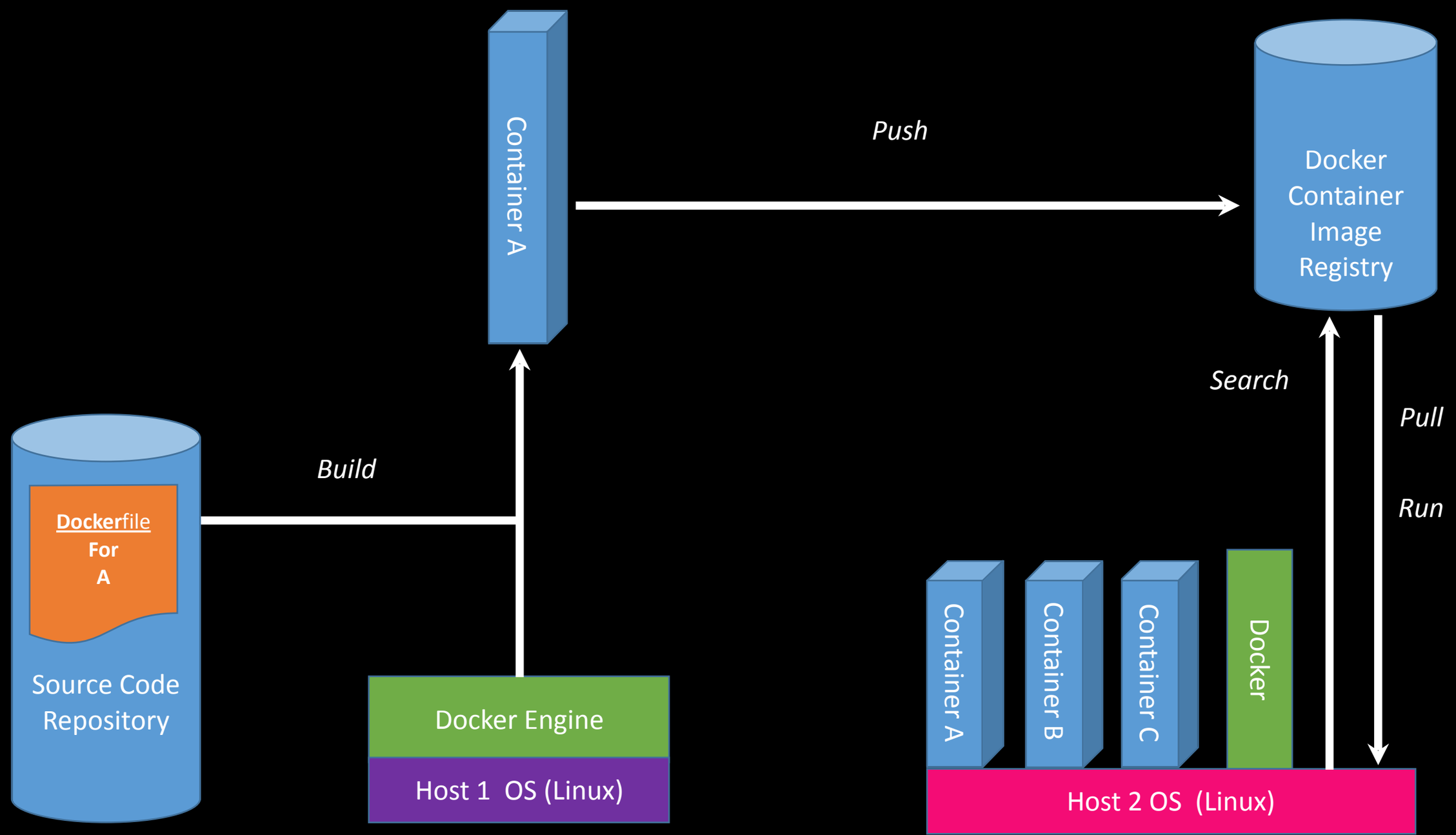
- Containers are launched from images
- They contain one or more running processes
- Can be started, stopped, restarted and killed

# Images vs. Containers

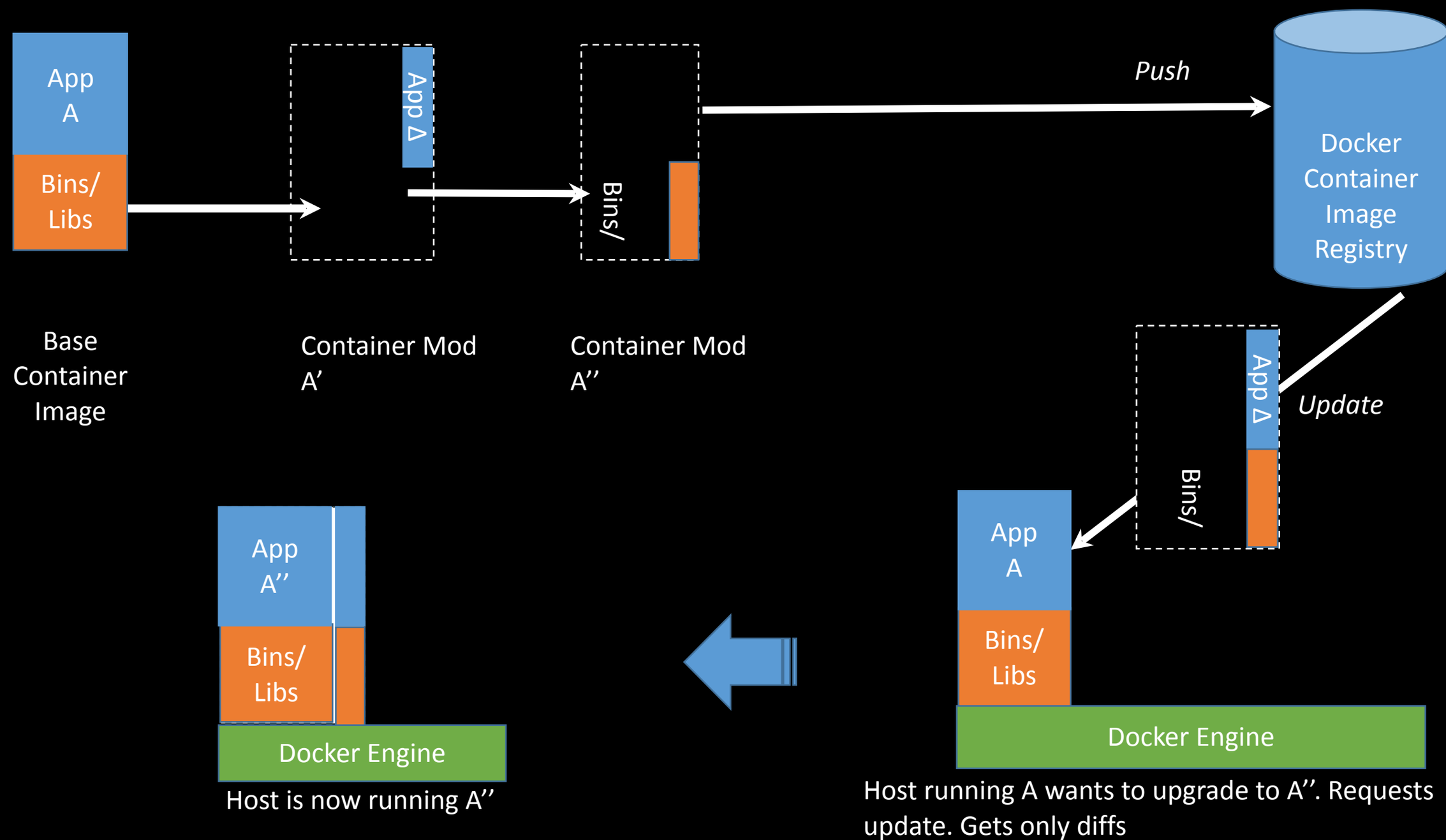
- An images is a stopped container
- You make an image by committing a container
- Images are readonly, containers are writable
- Images are building, containers are running

Build (**Image**)
Ship (by **Registry**)
Run (**Container**)

# Workflow


Container A
Push
Docker Container Image Registry
Search
Pull
Run
Build
Dockerfile For A
Source Code Repository
Docker Engine
Host 1 OS (Linux)
Container A
Container B
Container C
Docker
Host 2 OS (Linux)

# Changes and Updates



# Commands

- $ **docker search** ubuntu
- $ **docker pull** <user>/<repo>

- $ **docker run** ubuntu /bin/echo hello world

- $ id=$(docker ps -q -l)
- $ **docker stop** $id
- $ **docker start** $id

- $ **docker commit** $CONTAINER_ID <user>/<repo>
- $ **docker push** <user>/<repo>

# Dockfile
## Automating Builds

```
FROM ubuntu
MAINTAINER YOUR_NAME YOUR_EMAIL
RUN echo "deb http://archive.ubuntu.com/ubuntu precise main universe" > /etc/apt/sources.list
RUN apt-get update
RUN apt-get install -y memcached
ENTRYPOINT ["memcached"]
USER daemon
EXPOSE 11211
```

```
$ docker build -t memcached - < Dockerfile
$ docker push <user>/<repo>
```

# Resources

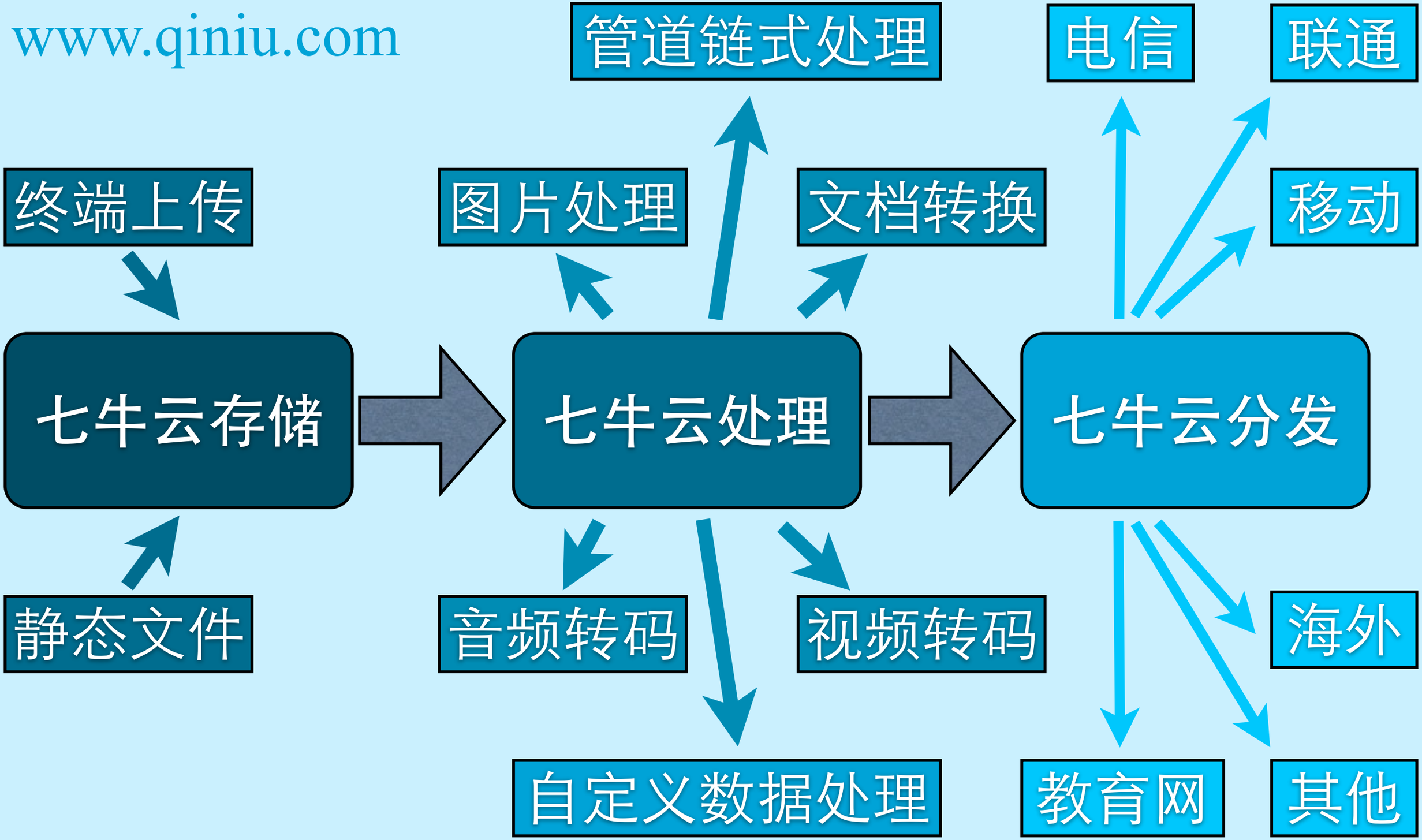
- https://www.docker.io/
- https://github.com/dotcloud/docker
- http://bit.ly/dockersources
- http://lwn.net/Articles/199643/
- http://lwn.net/Articles/236038/
- http://en.wikipedia.org/wiki/Operating_system-level_virtualization
- https://linuxcontainers.org/
- http://en.wikipedia.org/wiki/Cgroups
- http://en.wikipedia.org/wiki/Aufs

# Qiniu meet Docker

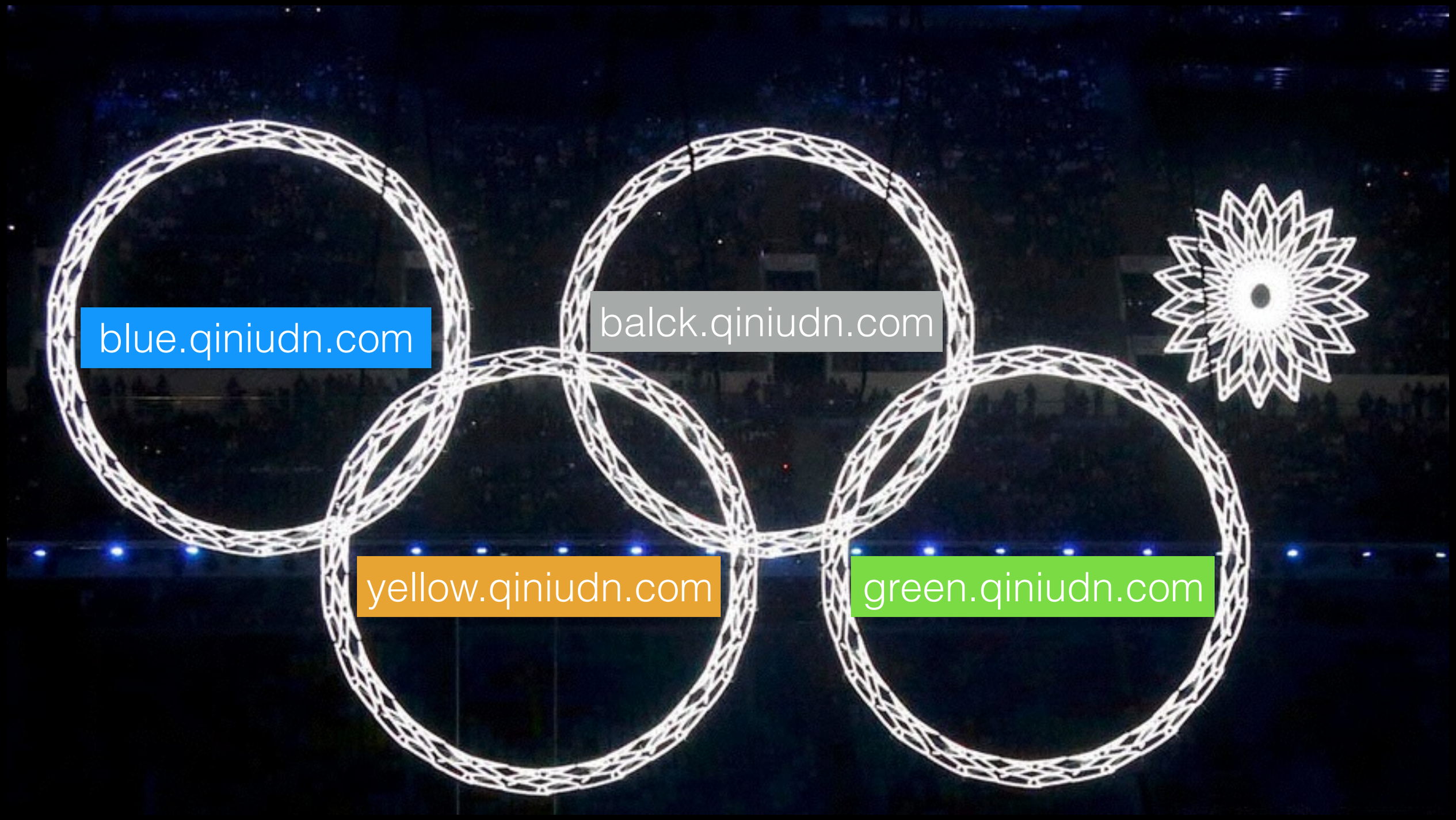
- Qiniu Cloud Storage service full-stack Go (since 2011)
- Docker is also written in Go (since 2011)
- Both Gopher

- Docker’s containers is user-level, not good for disk use
- Qiniu Cloud Storage is os-level, disk-based
- How could they can be together?

www.qiniu.com




第三方云主机 or 第三方云引擎 or 自有服务器


blue.qiniudn.com
balck.qiniudn.com
yellow.qiniudn.com
green.qiniudn.com

# Q & A

- Twitter: why404
- Wechat: why404

# THANK YOU !