[Docker] Containerization

ABCD-LMA Working Group

Will Kinard

October 12, 2017

WILL

Kinard

Infrastructure Architect
Software Developer

Startup
Venture
IC

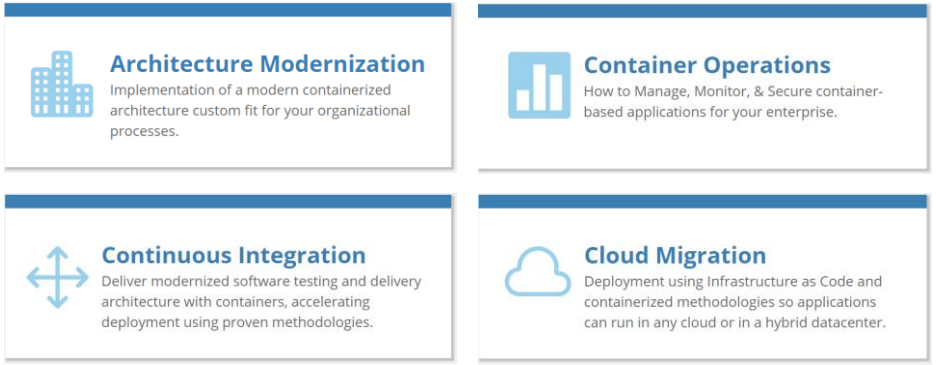
Husband
Father

That's me.

boxboat technologies

The **Container Enablement** Company

- **DevOps** Advisory and Implementation Services
- Authorized **Docker** Consulting and Integration
- Certified CloudBees **Jenkins** Consulting
- Continuous Integration, Delivery, and Deployment
- Application Modernization
- Distributed Solution Architecture



elastic cloudbees docker kubernetes OPENSHIFT Docker Datacenter Twistlock amazon web services COSCALE Microsoft Azure

boxboat technologies

The **Container Enablement** Company

PAYCHEX

BLACK KNIGHT FINANCIAL SERVICES

MassMutual

iJET

SENTARA

NIST

Waters
THE SCIENCE OF WHAT'S POSSIBLE.

xerox

DELTA AIR LINES

NSF

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

WILEY

Container Tech Isn’t New

2013

DotCloud → Docker Inc.
Introduction of Docker Open Source Project

2008

LXC Linux Containers
Userland tooling

2007

Linux Control Groups
Process containers renamed and merged into kernel 2.6.24

2006

Google Process Containers
Process aggregation for resource management

2004

Solaris Zones
Similar to jails; snapshots, cloning

2002

Linux Namespaces
Process isolation

2000

FreeBSD Jails
Early container technology

1979

Chroot
Filesystem isolation

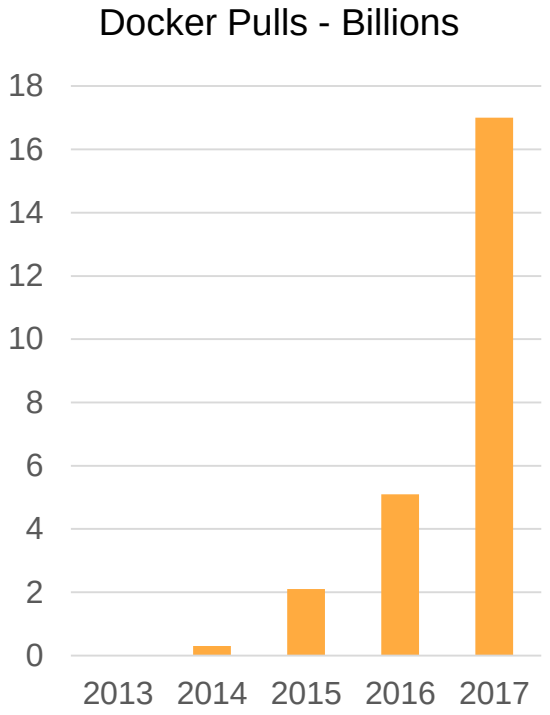
Docker Open Source Project

GitHub (github.com/docker)

- 2900+ Contributors
- 10,000+ Active Forks
- 34,000+ Stars

Docker Hub (hub.docker.com)

- 16B+ Image Downloads
- 1,000,000+ Dockerized Applications
- Exponential growth

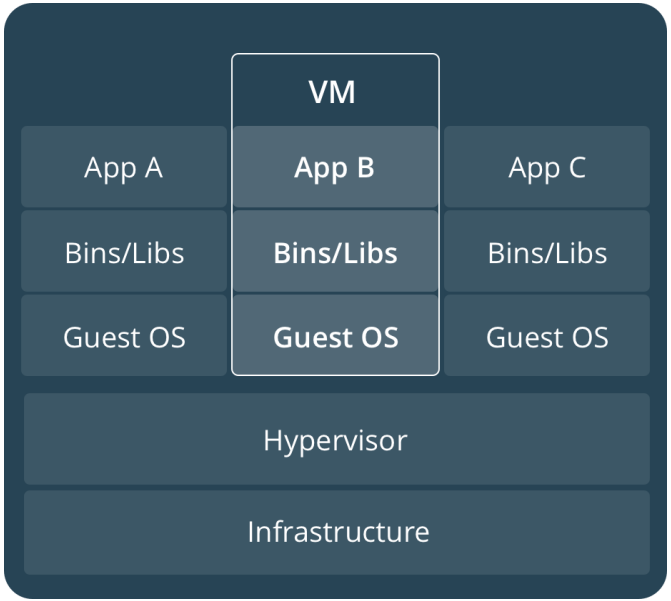


What *is* a [Docker] Container ?

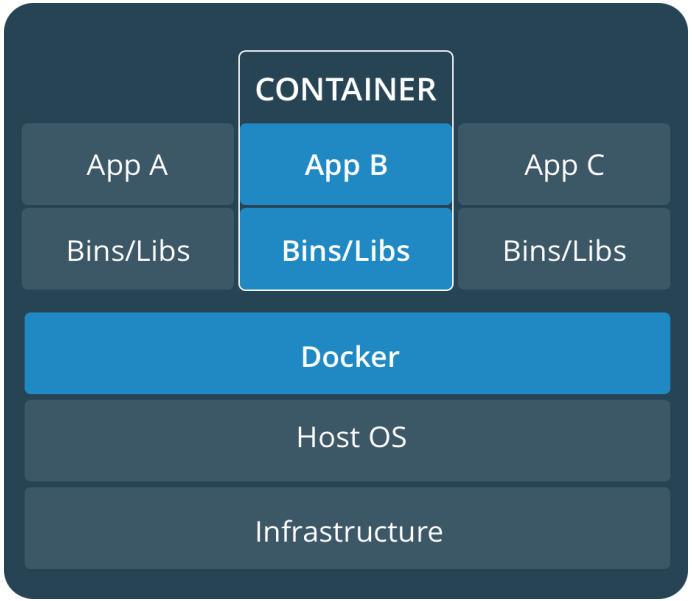
- Method to run applications in isolation
- Isolation includes namespacing pid, network, users, restricting root, cpu and memory limits, and providing separate filesystem
- Many of the technologies are old, but haven't been packaged in an easy to use toolset before Docker

“Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server.”
(https://www.docker.com/what-docker)

Containers…and VMs?



Each virtual machine includes:
- application
- binaries and libraries
- entire guest operating system

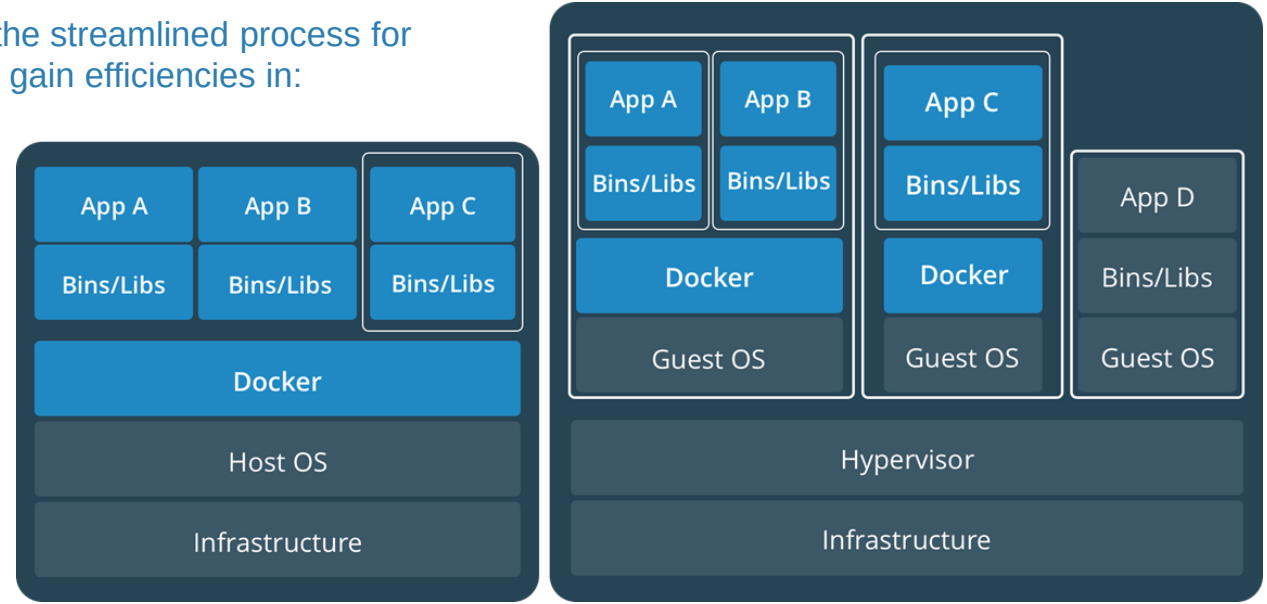


Containers:
- Include application and *all* dependencies
- Share kernel with other containers
- Run as an isolated process not tied to any specific infrastructure

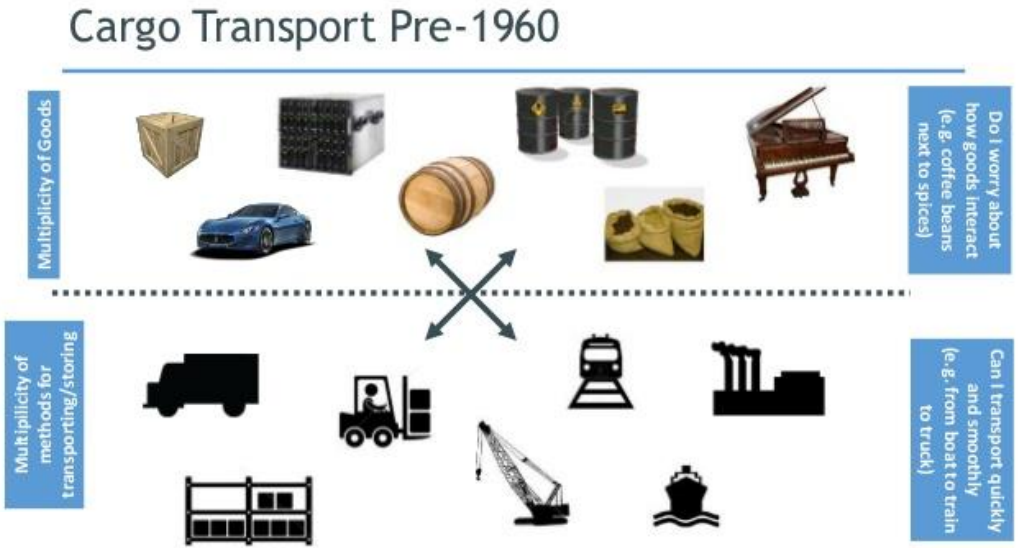
They’re different, not mutually exclusive

Take advantage of the streamlined process for VM based IaaS and gain efficiencies in:

- Higher density workloads
- Scale
- Portability
- Security

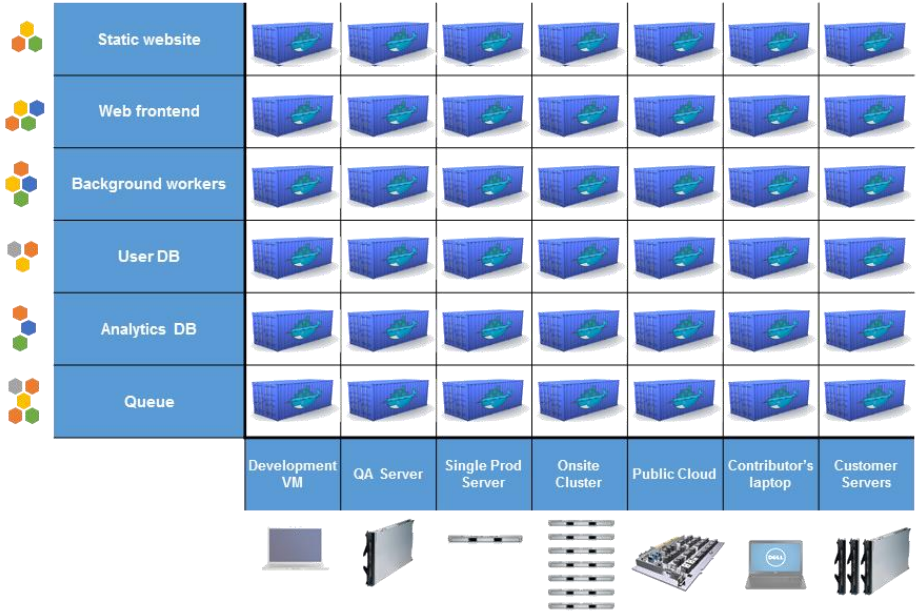


Portability is Empowering



Portability is Empowering

Portability is Empowering



Turtles all the way down

Abstractions in software development have increased over time in support of faster and easier delivery

Containers abstract infrastructure (read: “runtime”)

Docker is an accepted format for building/shipping/running containers

Application Development (Build & Ship)

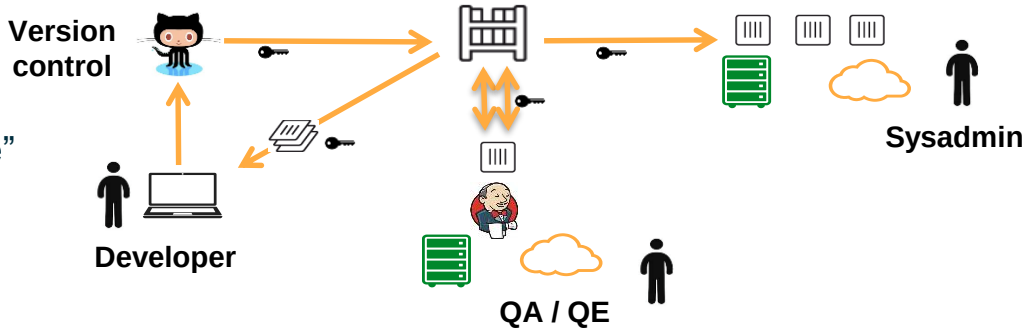
Problem: Code Migration/Deployment

- Environment replication is difficult
- Developer on-boarding is tedious
- Feedback cycle from QA is slow
- Artifact management creep

Solution: Single Artifact Deployment

- Container replicates exact environment
- Deploy and migrate in sub-seconds
- Automated and self-documenting
- Track promotion cryptographically

Begin with
“Known Trusted State”



End with
Approved Image

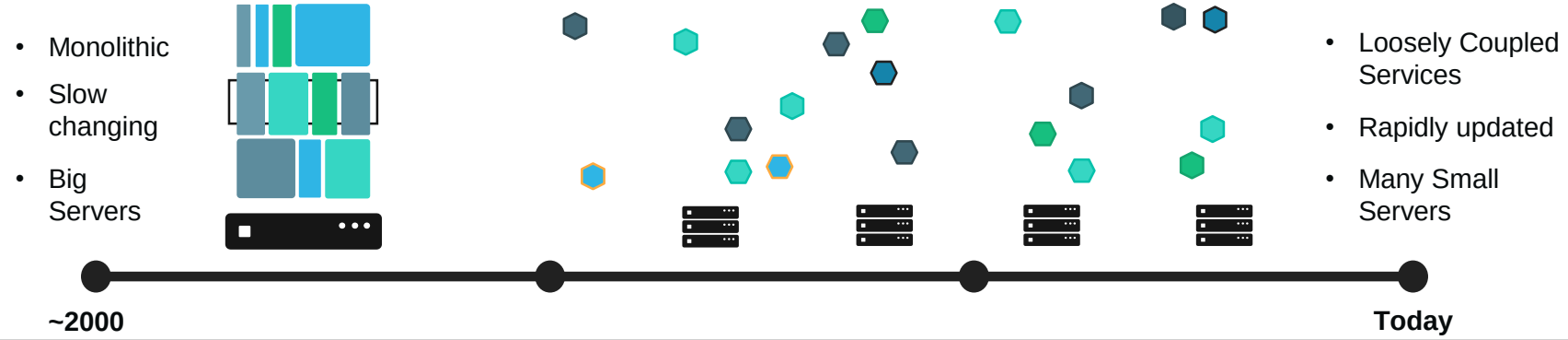
Application Modernization

Problem: Legacy Applications

- Brittle and difficult to upgrade
- Lengthy deployment cycle times
- Aging platform components and dependencies

Solution: Microservice Architecture

- Technology Diversity
- Deployment flexibility (On-Premise, Cloud, Hybrid)
- Cluster orchestration & high availability



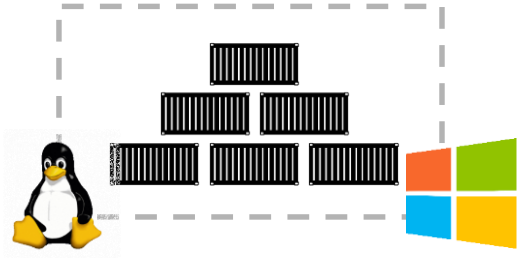
Production Operations

Problem: Inefficiency of VMs

- OS duplication for isolated work loads
- Lengthy boot and replication times
- Hardware, Storage, and Hypervisor costs $$
- Customer self-servicing leads to VM creep

Solution: Docker CaaS

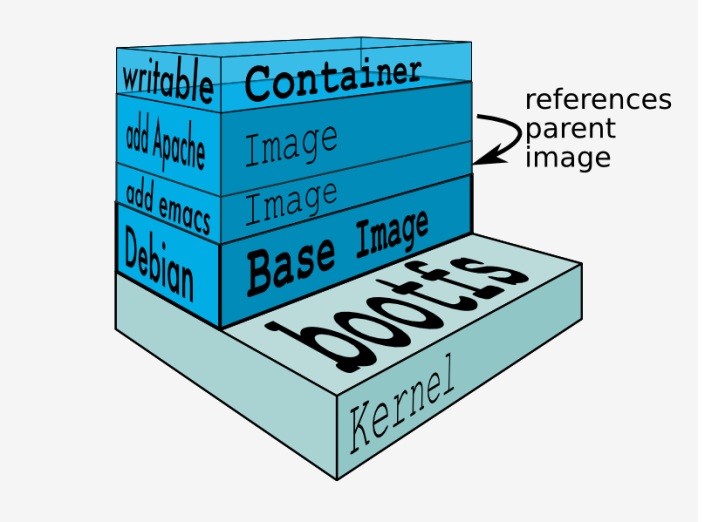
- Consolidated storage in UFS and COW
- Deployment flexibility (On-Premise, Cloud, Hybrid)
- High Availability and Scaling
- Policy driven architecture using PKI



Build

Docker Images

- Images are the definition. They include the filesystem, environment variables, and default entry points.
- Containers are an instance of an image. They isolate the application from the host, and even from other containers.



Build
Dockerfiles

- Write your image definition in a Dockerfile

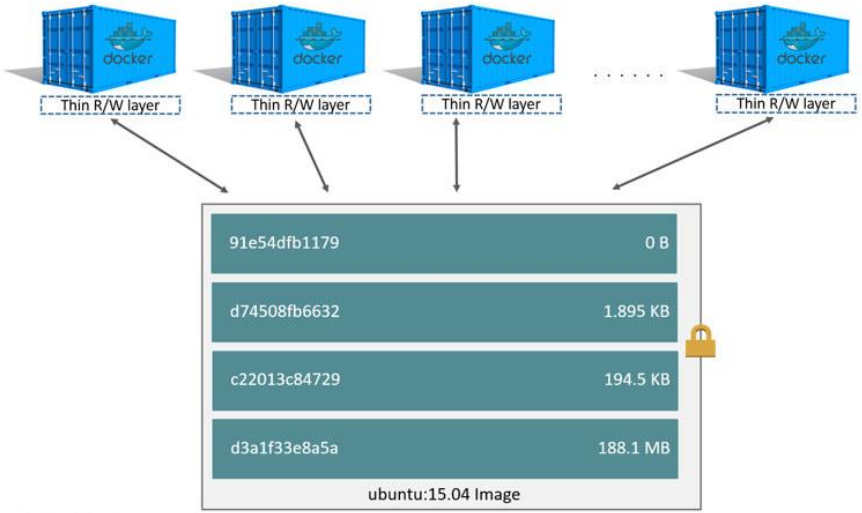
```
FROM debian:latest
RUN apt-get update && apt-get install apache2
COPY html /var/www/html
CMD httpd -DFOREGROUND
```

- Turn that Dockerfile into an image with

```
docker build
```

- Develop a new app or “lift and shift” your current codebase

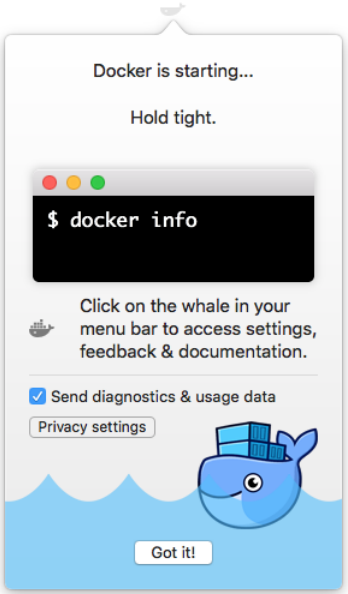
Docker Images



- Union file system
- Multiple RO layers are stacked
- Containers add a single RW layer to isolate changes
- Layers are cached for fast builds
- Layers are named with a hash inside the engine

Docker for Mac / Windows

- Docker tools for the developer
- OS native clients using internally available virtualization: xhyve and Hyper-V
- Full Docker CLI from native OS shell



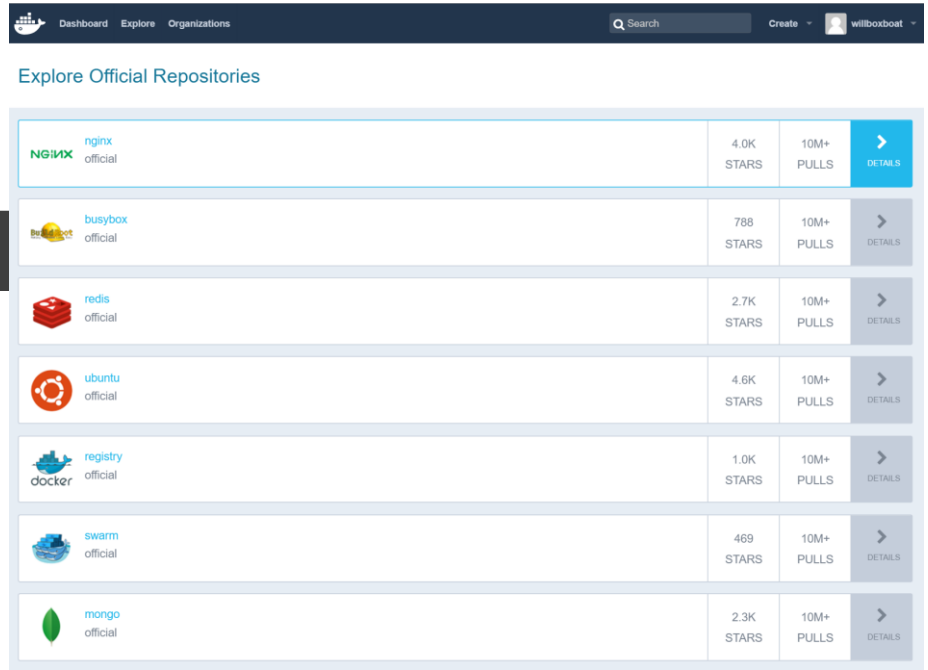
Ship

Docker Registry (and Hub)

- Push and pull to central registry

```
docker push registry/app:1.0.1
docker pull registry/app:1.0.1
```

- Organized as repositories that contain multiple tags
- Multiple options: run your own, Docker Hub, DTR, 3rd parties



Run

- Run your image

```
docker run registry/app:1.0.1
```

- Launches a container base on your image
- Options for:

❖ Volumes: link external data into the container for persistence

❖ Networking: bridged, overlay, access with exposed ports

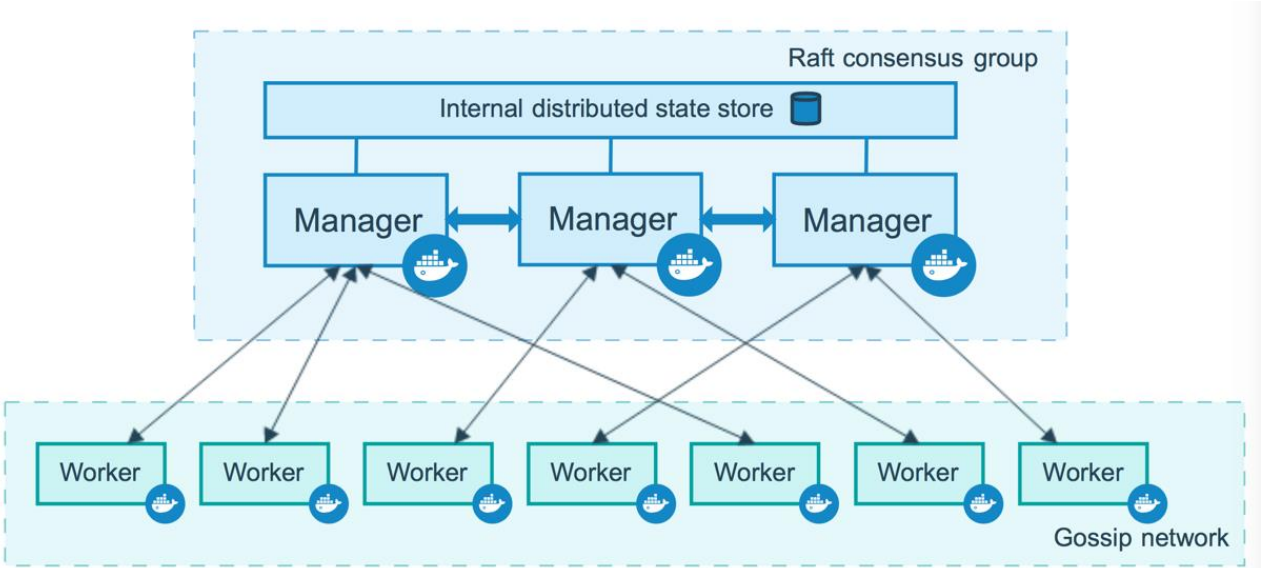
Immutable

- Containers are designed to be disposable
- New containers go back to a clean image state
- Running containers write to an isolated space

- Data is stored outside of the container
- Separates data from your application

Run
Distributed

- Fault tolerant
- Blue/Green Deployment
- Seamless rollbacks



Distributed

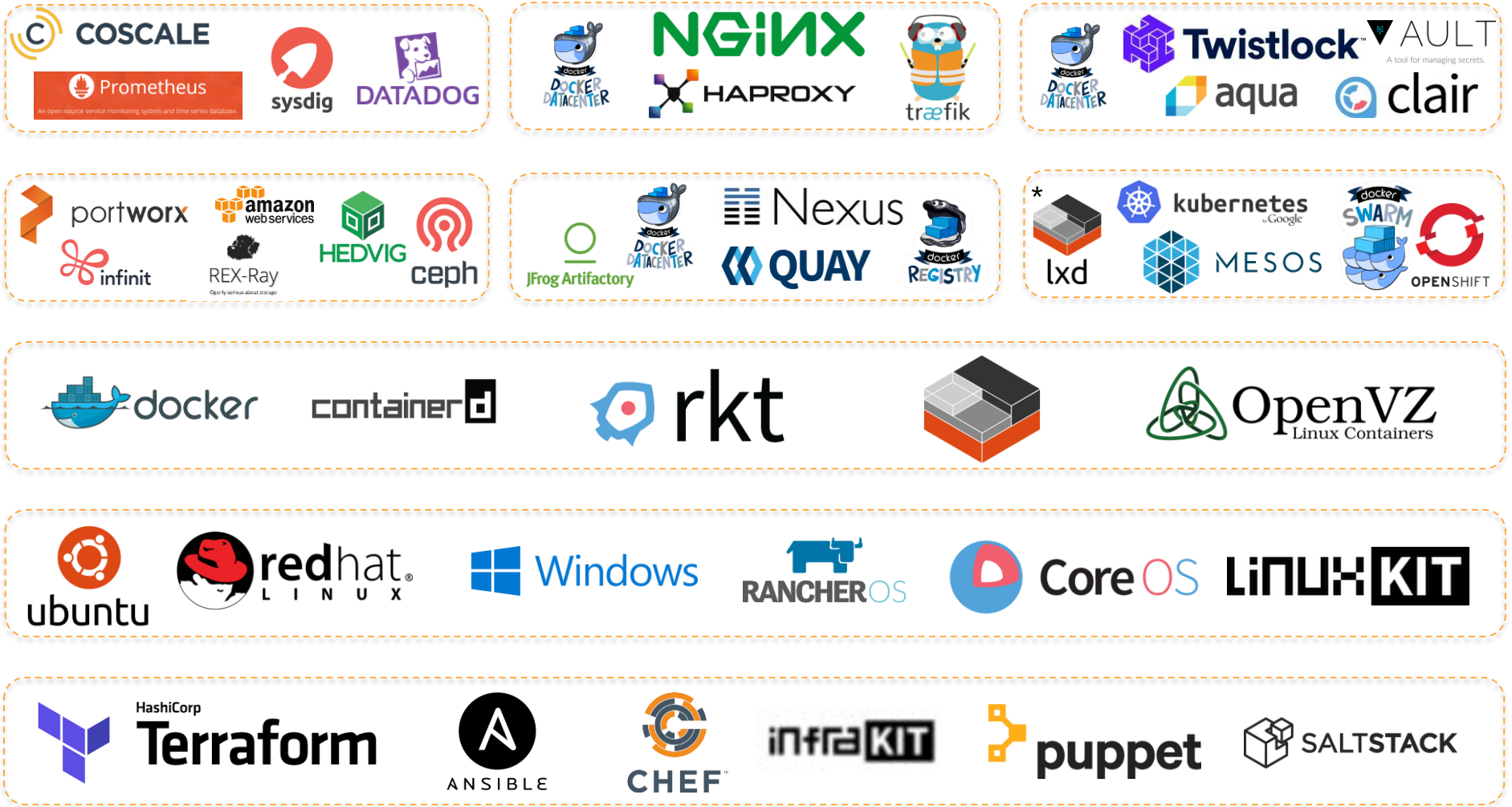
Docker Compose

- Packages multiple containers together
- Defines parameters for ‘docker run’
- Configuration is stored in ‘docker-compose.yml’
- Allows containers to be scaled, but without orchestration

```
version: '2'

# optional, only needed if you use them
volumes:
  vol-name-1:
    driver: local
networks:
  internal1:
    driver: bridge

# required
services:
  service-1:
    build: .
    image: my-service:latest
    volumes:
      - ./auth:/my-app/auth:ro
      - vol-name-1:/my-app/data-1
    networks:
      - internal1
  # ... continued
```



COSCALE
Prometheus
An open-source service monitoring system and time series database.
sysdig
DATADOG
NGINX
HAPROXY
træfik
Twistlock
AULT
A tool for managing secrets.
aqua
clair
portworx
infinit
amazon web services
REX-Ray
HEDVIG
ceph
JFrog Artifactory
Nexus
QUAY
REGISTRY
lxd
kubernetes
MESOS
SWARM
OPENSHIFT
docker
containerd
rkt
OpenVZ
Linux Containers
ubuntu
redhat LINUX
Windows
RANCHEROS
CoreOS
LINUXKIT
HashiCorp
Terraform
ANSIBLE
CHEF
infraKIT
puppet
SALTSTACK

Thank You

Will Kinard
CTO BoxBoat Technologies

will@boxboat.com
@boxboat

www.boxboat.com/blog