docker

SAINT LOUIS JAVA USER GROUP
MAY 2014

STEVEN BORRELLI
steve@borrelli.org
@stevendborrelli

ABOUT ME

FIRST COMPUTER:

SYSTEMS ENGINEERING MANAGEMENT

FOUNDER, ASTERIS (JAN 2014) @

ORGANIZER OF STL MACHINE LEARNING AND DOCKER STL

WHY DOCKER?

Docker makes it easy to:

Package

Deploy

Share

Server Applications

Think:

```
java -jar
```

vs.

```
./configure; make install
```

DOCKER FACTS

Written by Docker, Inc. (Formerly Dotcloud)

Automates the management and control of Linux containers

Rewrite of their proprietary PAAS container engine (written in Python)

Written in Go / Apache 2 License

11,700+ Github stars

DOCKER TIMELINE

JANUARY 2013: PROJECT START

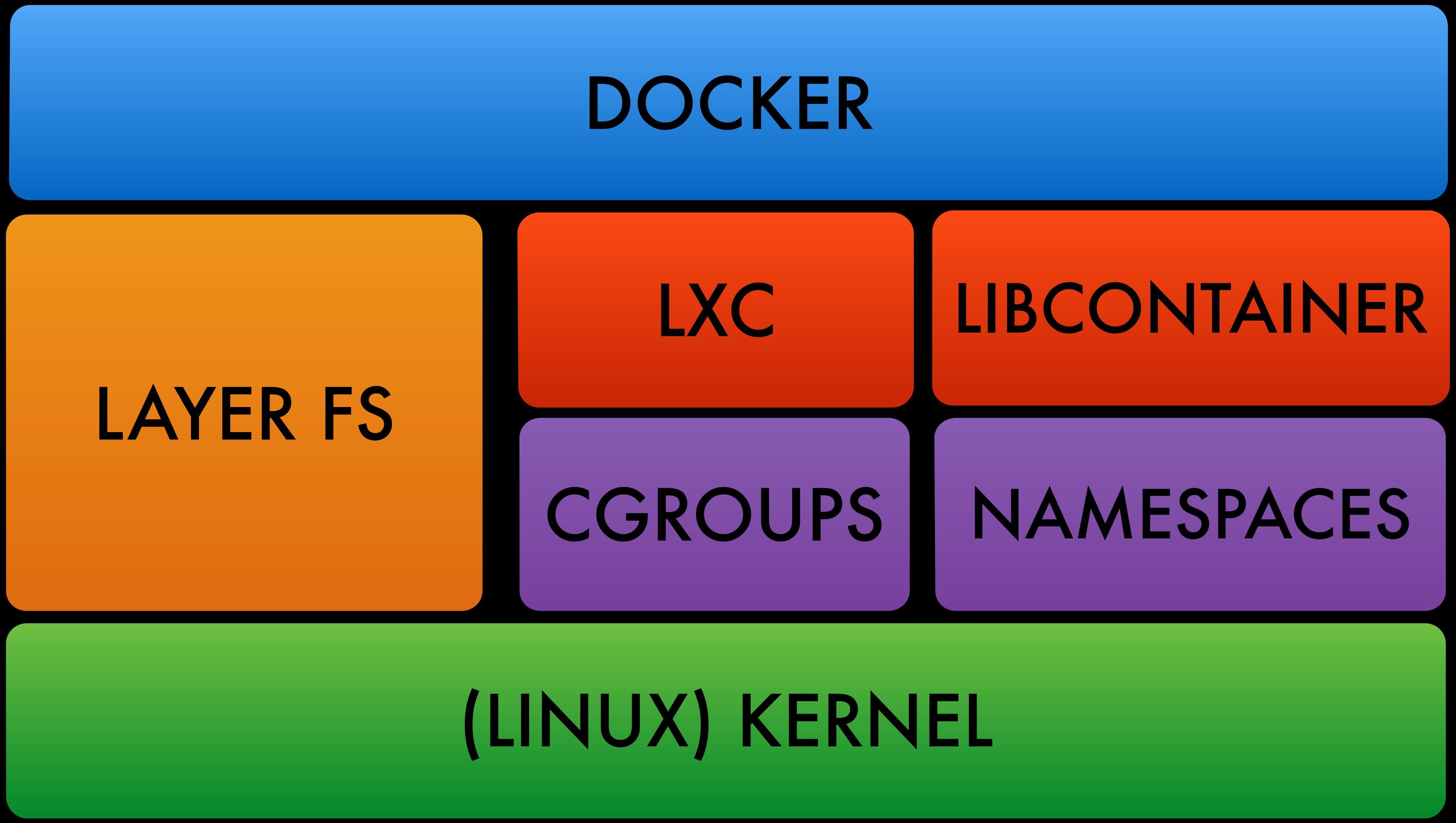
MARCH 2013: INITIAL GITHUB RELEASE

MONTHLY RELEASE CADENCE

MAY 7, 2014: 0.11 RELEASE

MAY 8, 2014: 0.11.1 RELEASE

DOCKER ARCHITECTURE



NAMESPACES VS. CGROUPS

Namespaces provide isolation:

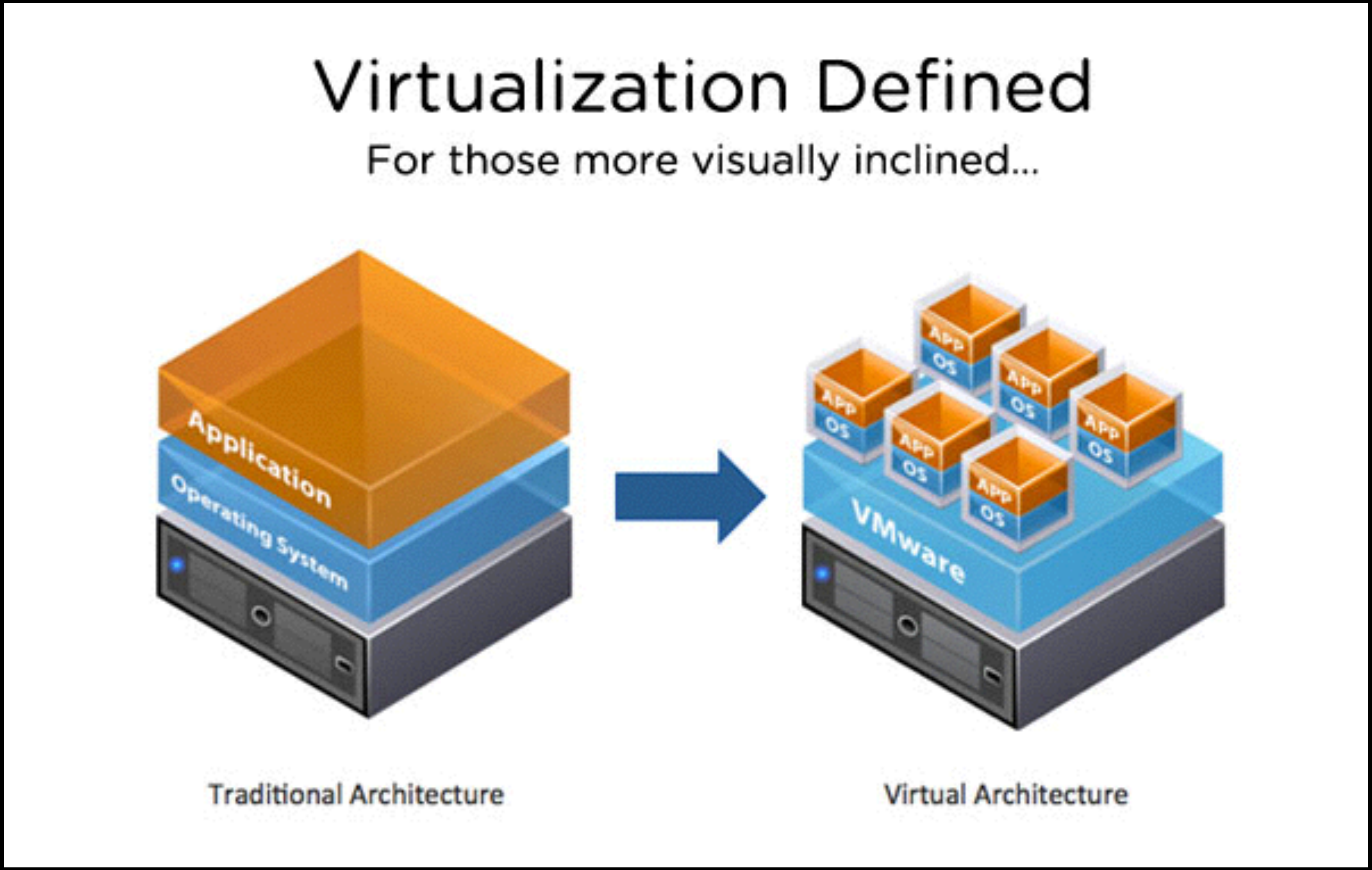
- pid (processes)
- net (network interfaces, routing…)
- ipc (System V IPC)
- mnt (mount points, filesystems)
- uts (hostname)
- user (UIDs)

Control groups control resources:

- cpu (CPU shares)
- cpusets (limit processes to a CPU)
- memory (swap, dirty pages,
- blockio (throttle reads/writes)
- devices
- net_cls, net_prio: control packet class and priority

What’s the difference between containers and virtual machines (VMs)?

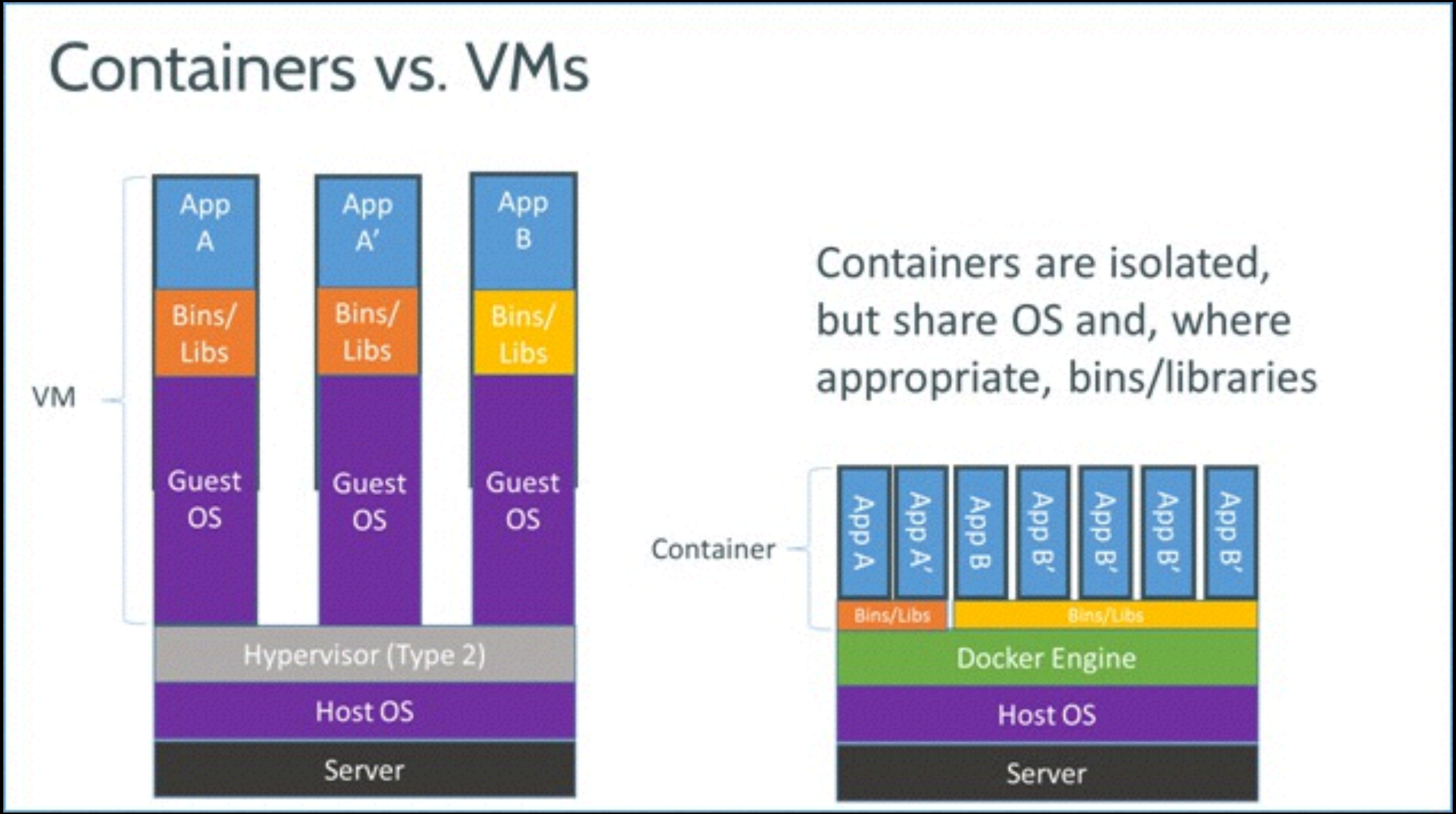
VIRTUALIZATION



HARDWARE VIRTUALIZATION

1966-1972	IBM CP/CMS
1989	Insignia SoftPC
1997	Connectix VirtualPC
1999	VMWare Workstation
2001	IBM AIX LPAR
2002	Xen
2006	*Amazon EC2*
2007	Sun Logical Domains
2007	Linux KVM
2007	InnoTek VirtualBox
2008	Microsoft Hyper-V

CONTAINERS


Containers vs. VMs
VM
App A
App A'
App B
Bins/ Libs
Bins/ Libs
Bins/ Libs
Guest OS
Guest OS
Guest OS
Hypervisor (Type 2)
Host OS
Server
Containers are isolated, but share OS and, where appropriate, bins/libraries
Container
App A
App A'
App B
App B'
App B'
App B'
App B'
Bins/Libs
Bins/Libs
Docker Engine
Host OS
Server

PROCESS VIRTUALIZATION

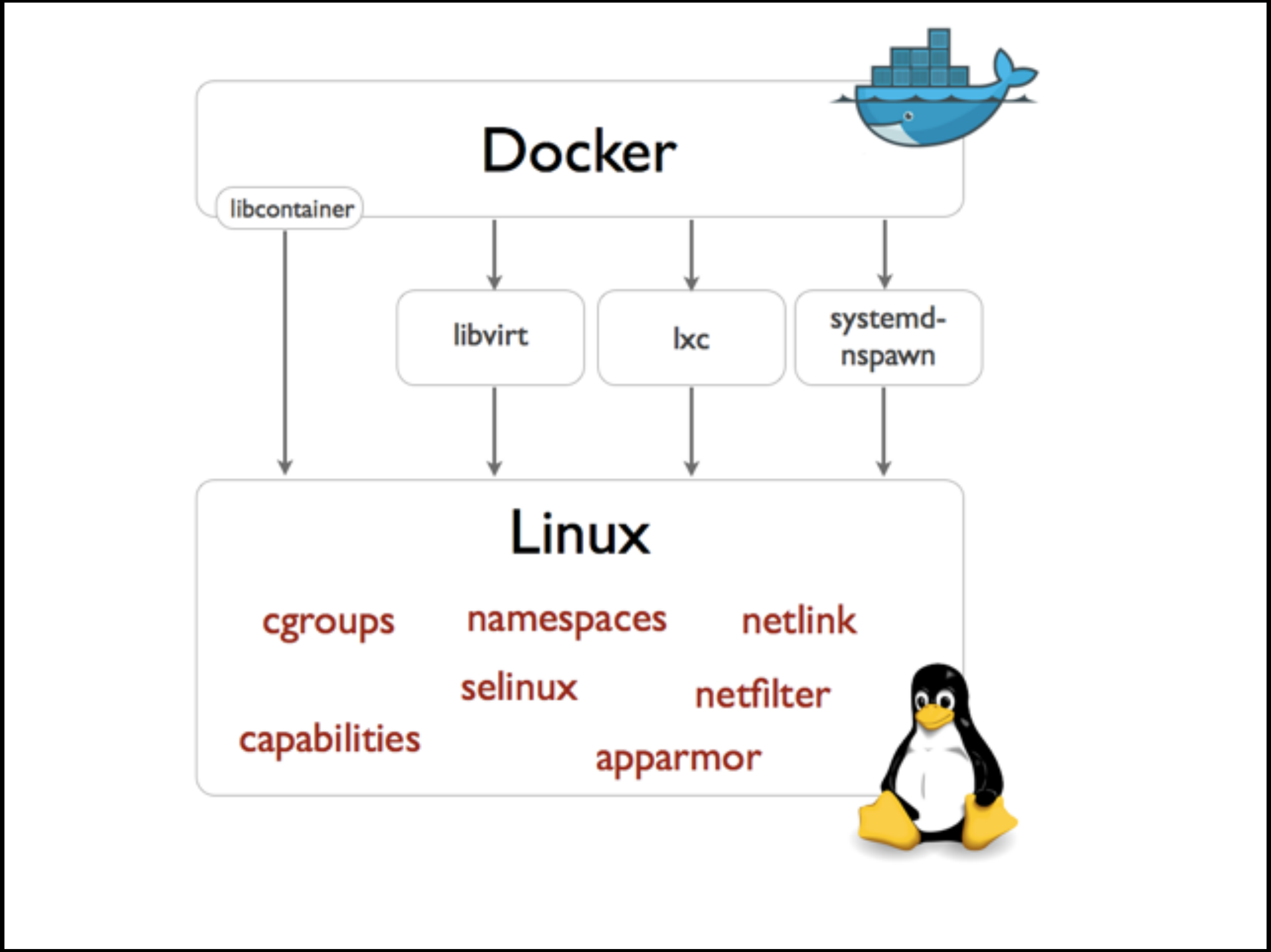
1979-1982	UNIX Chroot
1998	FreeBSD Jail
2001	Parallels Virtuozzo
2001	Linux-VServer
2005	Solaris Zones
2005	OpenVZ
2008	*Linux LXC*
2007+	*PAAS:* *Heroku, Joyent, CloudFoundry*
2013	Docker

Differences between containers and virtual machines

- Weaker isolation in containers
- Containers run near-native speed CPU/IO
- Containers launch in around 0.1 second (libcontainer)
- Less memory overhead

NOTABLE CHANGES

0.9: LIBCONTAINER



NOTABLE CHANGES

0.10:

- TLS support on docker API
- Systemd integration via API instead of /proc
- Lots of cleanups

NOTABLE CHANGES

0.11:

- Release Candidate for 1.0
- Multiple registries
- Direct host network access
- SELinux support

EXAMPLES

RUNNING A CONTAINER

Start a container:

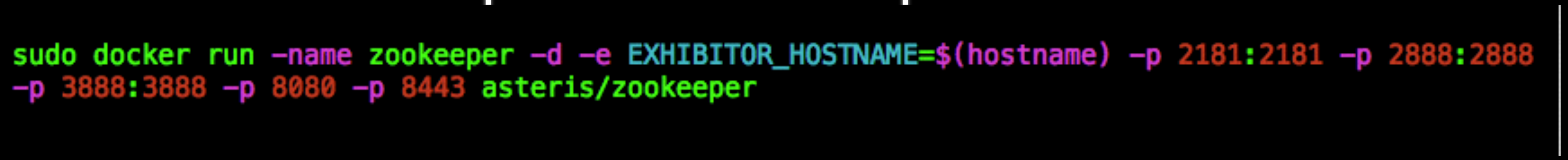
```
$ sudo docker run -i -t ubuntu:12.04 /bin/bash
root@09aa796197bc:/# 
```

Mount host filesystems:

```
$ sudo docker run  -i -t -v /var/log:/var/host/logs ubuntu /bin/bash
root@bb52ddbdb91c:/# ls /var/host/logs | head -3
apt
auth.log
boot.log
```

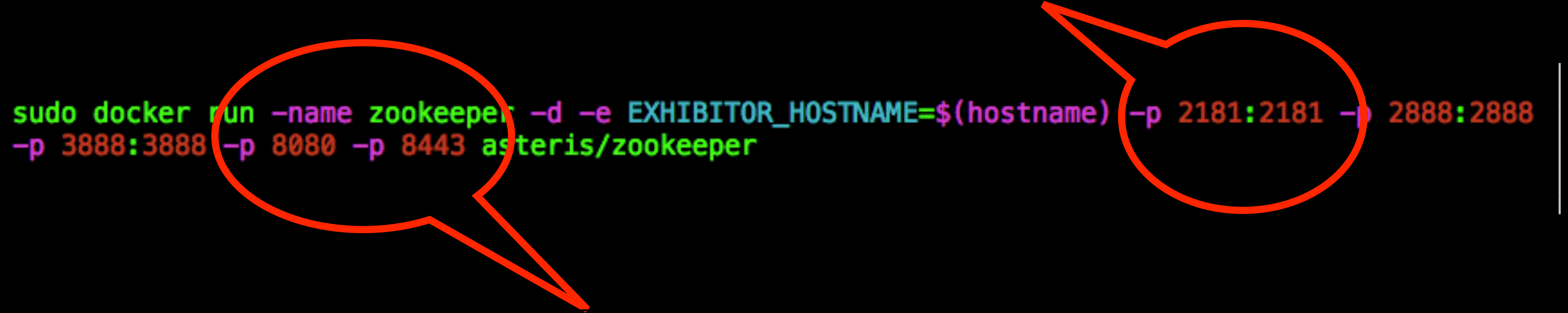
MAPPING PORTS

Example: run Zookeeper + Exhibitor



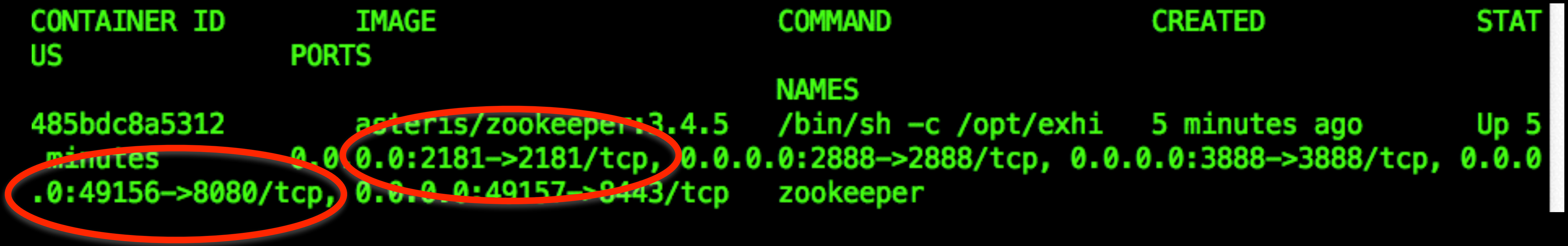
```
sudo docker run -name zookeeper -d -e EXHIBITOR_HOSTNAME=$(hostname) -p 2181:2181 -p 2888:2888
-p 3888:3888 -p 8080 -p 8443 asteris/zookeeper
```

Port 2181 on host will be mapped to 2181 on container



```
sudo docker run -name zookeeper -d -e EXHIBITOR_HOSTNAME=$(hostname) -p 2181:2181 -p 2888:2888
-p 3888:3888 -p 8080 -p 8443 asteris/zookeeper
```

Host ports will be dynamically allocated by docker



```
CONTAINER ID        IMAGE                       COMMAND                 CREATED             STAT
US                PORTS
                                                NAMES
485bdc8a5312        asteris/zookeeper:3.4.5    /bin/sh -c /opt/exhi    5 minutes ago       Up 5
 minutes          0.0.0.0:2181->2181/tcp, 0.0.0.0:2888->2888/tcp, 0.0.0.0:3888->3888/tcp, 0.0.0
.0:49156->8080/tcp, 0.0.0.0:49157->8443/tcp    zookeeper
```

DIRECT HOST NETWORK

New in 0.11, allows a container to access host adapters:

```
# docker run -d --net=host tomcat
```

Port 8080 on the container is 8080 on the host:

```
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::8080                 :::*                    LISTEN
```

IMMUTABLE SERVERS

Physical server lifetime is measured in years.

A container's lifetime can be as short as a few seconds.

Treat containers like a build artifact.

If you need to make changes, build a new container.

JAVA DOCKERFILE

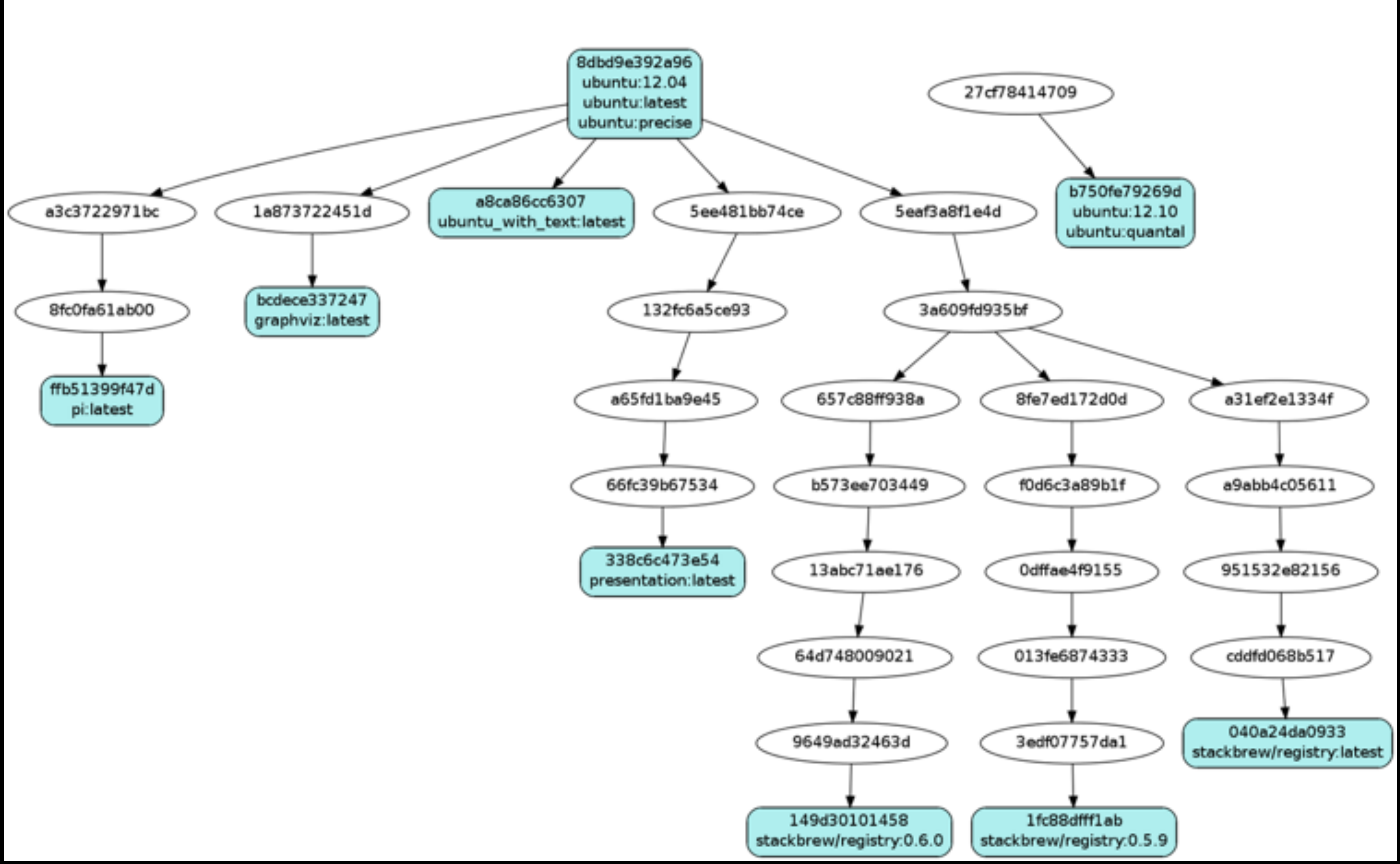
```
FROM dockerfile/ubuntu

RUN add-apt-repository -y ppa:webupd8team/java
RUN apt-get update
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
RUN apt-get install -y oracle-java7-installer
```

TOMCAT DOCKERFILE

```
FROM dockerfile/java

RUN apt-get -y update && apt-get -y install tomcat7 tomcat7-admin tomcat7-examples

RUN echo "JAVA_HOME=/usr/lib/jvm/java-7-oracle" >> /etc/default/tomcat7


EXPOSE 8080

ENV CATALINA_HOME /usr/share/tomcat7
ENV CATALINA_BASE /var/lib/tomcat7

CMD /usr/share/tomcat7/bin/catalina.sh start && tail -f /var/lib/tomcat7/logs/catalina.out
```

LAYERED FS IS A GRAPH

```
docker$ sudo docker images --viz=true | dot -Tpng -o graph.png
docker$ █
```



RUNNING JAVA IN DOCKER

Problem: keep configuration out of containers

- Pass in environment variables to Java vars (i.e. in `start.sh`):

```
java -Dkeystore.password=${KEY_PASS}
```

- When you run the container, set the vars:

```
docker run -e SSL_PASS=password tomcat
```

RUNNING JAVA IN DOCKER

Problem: keep configuration out of containers

- Link from a volume container

```
docker run -v /opt/properties -v /opt/ssl \
-name TOMCAT-CFG busybox true

docker run -t -i -rm -volumes-from TOMCAT-
CFG  -name appsrv1 tomcat
```

- Mount host filesystem:

```
docker run -v/opt/ssl:/opt/ssl tomcat
```

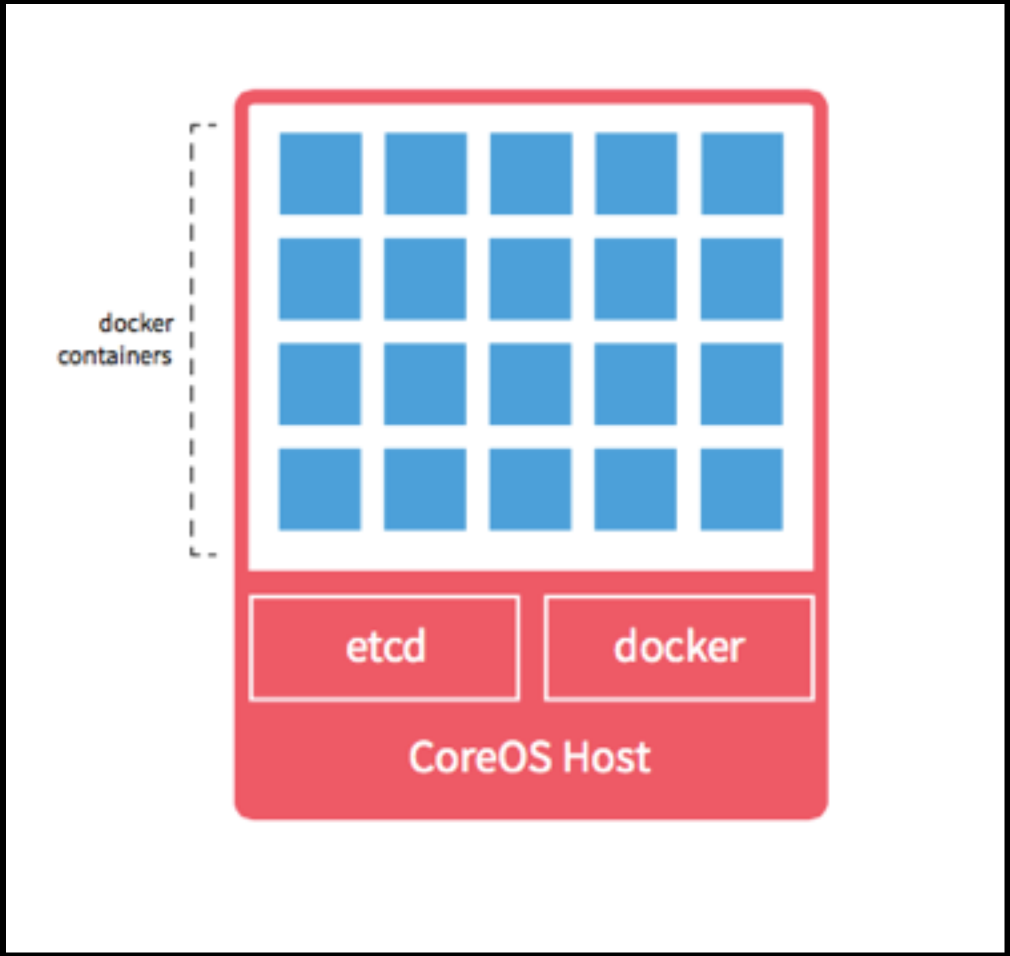
RUNNING JAVA IN DOCKER

Running stateless containers

- Use a redis/DB to store session data
- Use a shared FS (hdfs/nfs, etc.)
  or Object storage (Swift, S3) for data
- Send logs to a centralized location
- Docker future: storage plugins

WHAT'S NEXT?

Stripped-down operating systems



WHAT'S NEXT?

New operational models:

Continuous Delivery

Automated routing

Distributed consensus (Paxos, Raft)

Service Discovery (Zookeeper, etcd, serf, skydns, consul)

Distributed scheduling (Fleet, Mesos, YARN)

SUMMARY

Easy to build, run & share containers

Rapidly expanding ecosystem

Better performance vs. VMs

Layered filesystem gives us git-like control of images.

Reduces complexity of system builds

Q & A