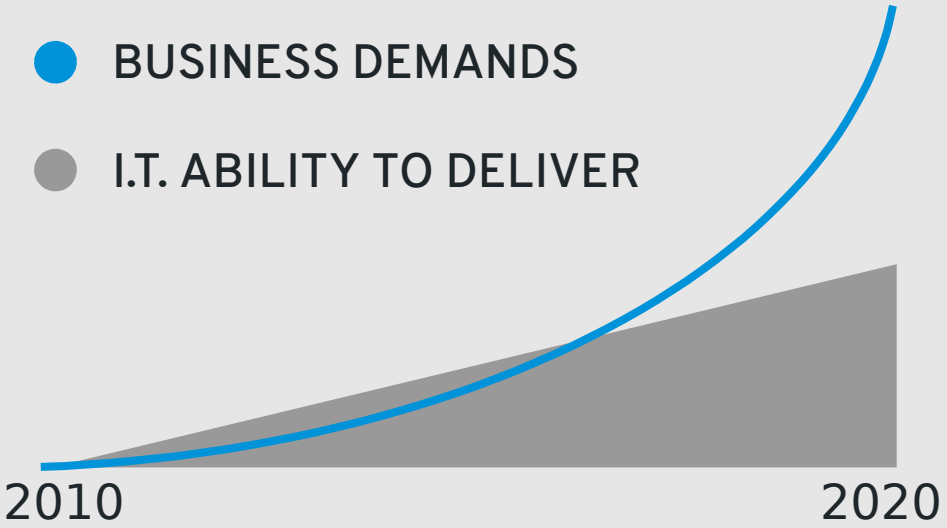
RED HAT ENTERPRISE LINUX ATOMIC HOST, CONTAINERS AND KUBERNETES

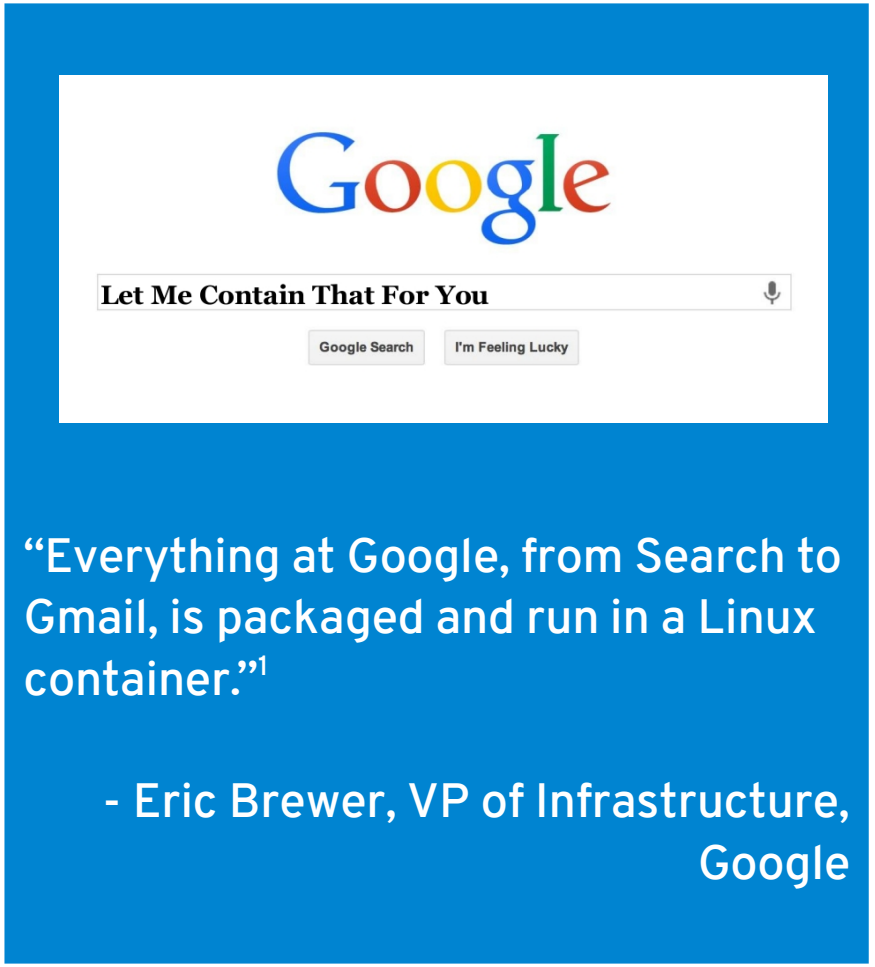
Adam Miller
Senior Software Engineer
OpenShift Online Release Engineering
January 2015 RHUG

THE PROBLEM



- Existing systems are inflexible and don't scale
- Increasing cost and complexity
- Need to invest in new platforms but do not have the time, resources or budget

CONTAINERS – A SOLUTION?



"Everything at Google, from Search to Gmail, is packaged and run in a Linux container."[1]

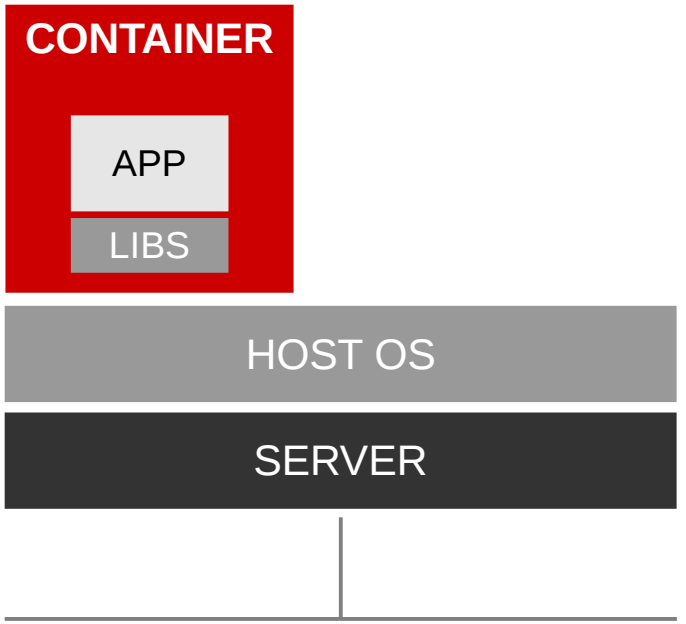
- Eric Brewer, VP of Infrastructure, Google

[1] Source: http://googlecloudplatform.blogspot.com/2014/06/an-update-on-container-support-on-google-cloud-platform.html

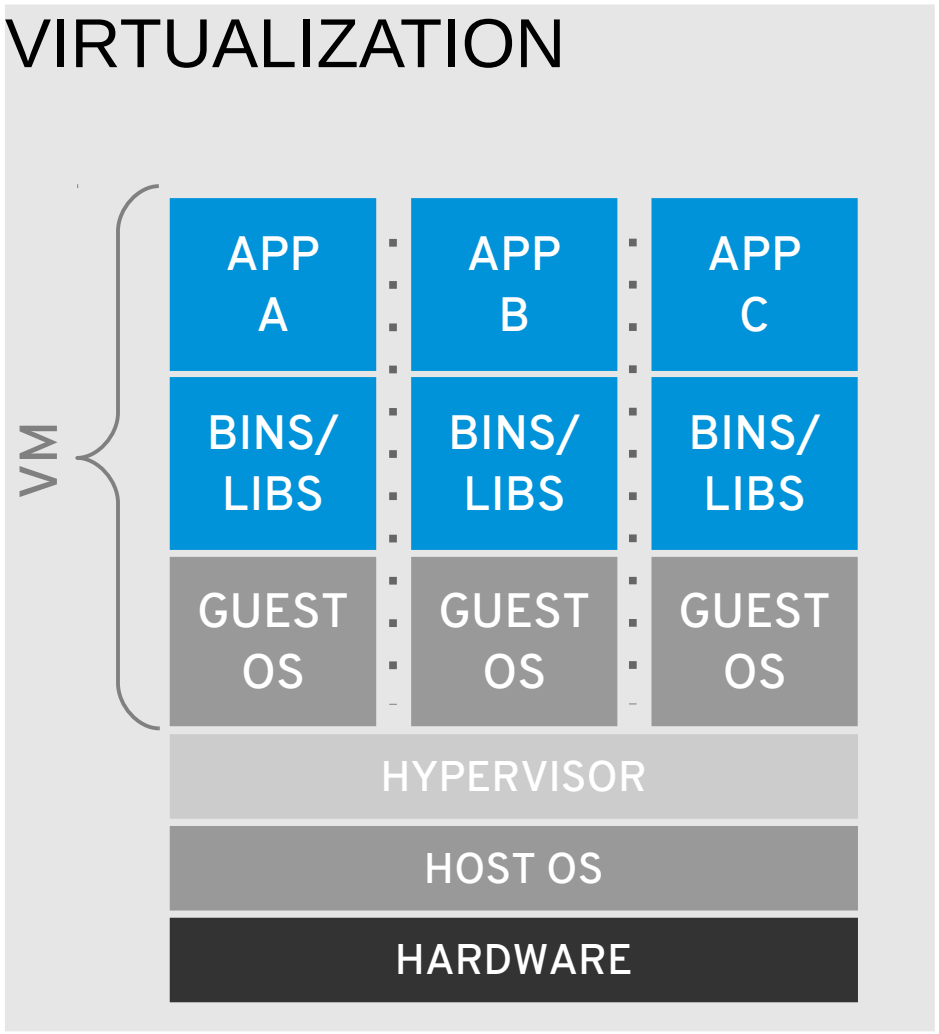
WHAT ARE CONTAINERS?

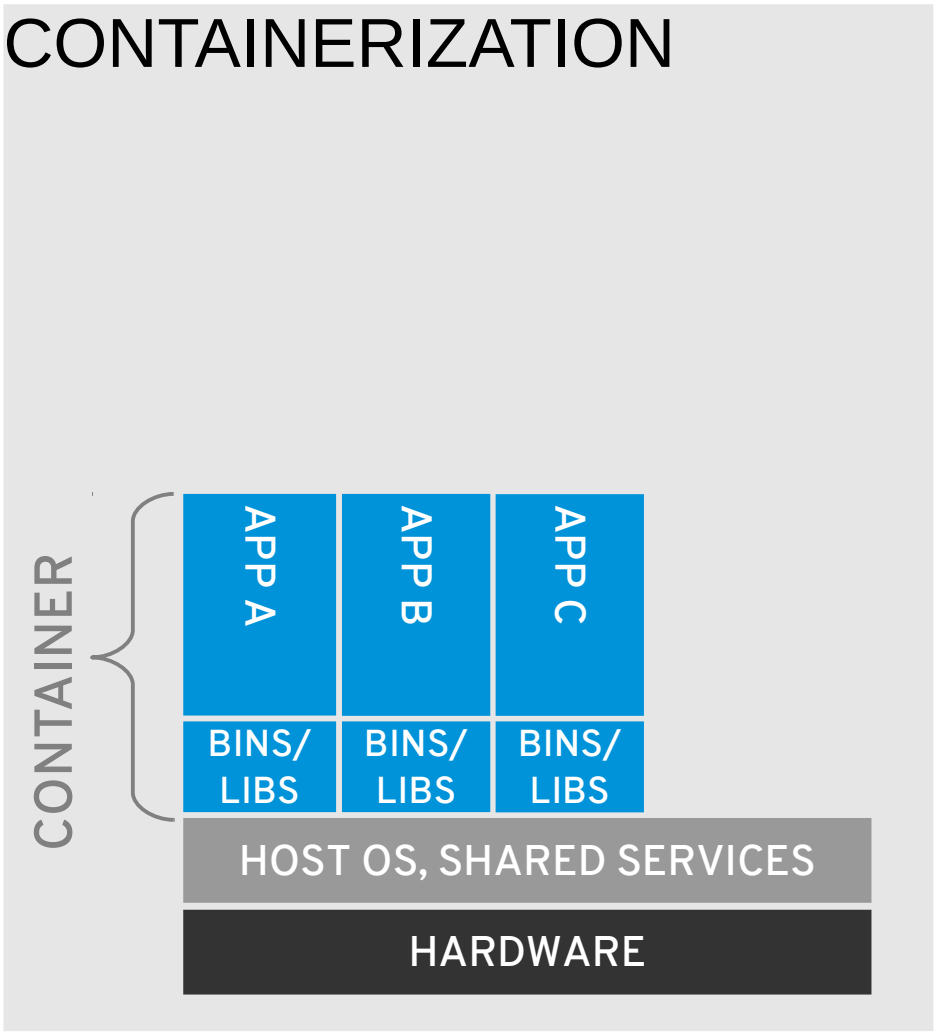
Software packaging concept that typically includes an application and all of its runtime dependencies.

- Easy to deploy and portable across host systems
- Isolates applications on a host operating system. In RHEL, this is done through:
  - Control Groups (cgroups)
  - kernel namespaces
  - SELinux, sVirt, iptables



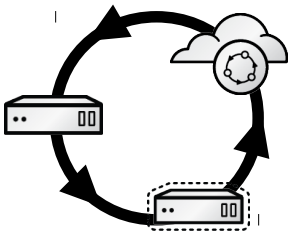
Virtualization and Containers





BENEFITS OF CONTAINERS

FASTER APP DELIVERY



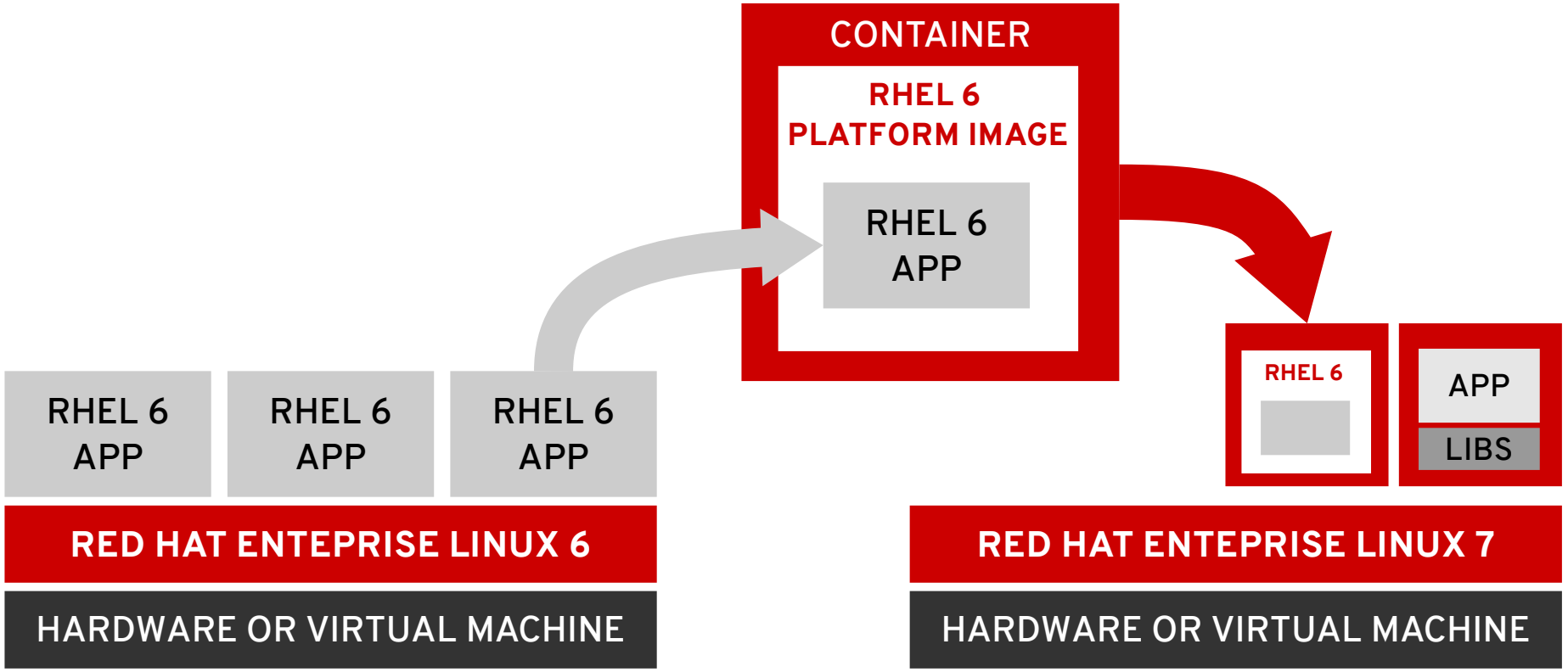
DEPLOYMENT FLEXIBILITY

OPERATIONAL EFFICIENCY

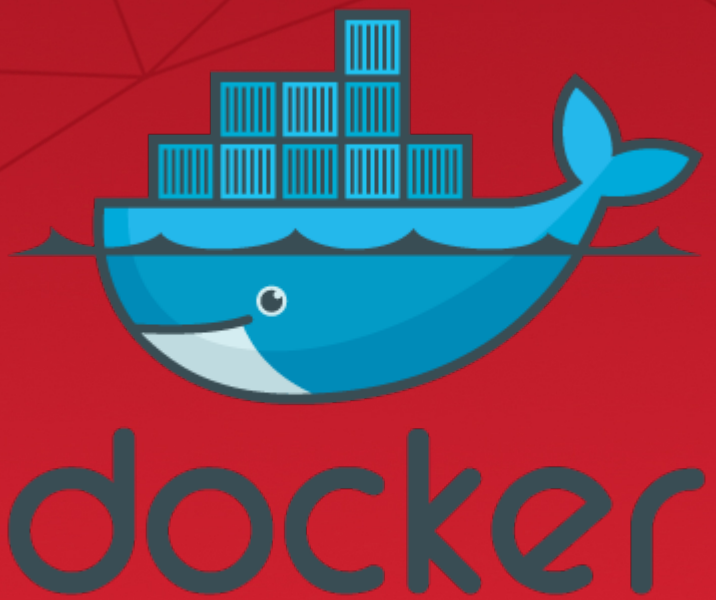
LOWERED DEPLOYMENT COSTS

CONTAINERS TRANSFORM THE WAY YOU DELIVER APPLICATIONS TO ACCELERATE INNOVATION

RUN RHEL 6 APPLICATIONS ON RHEL 7



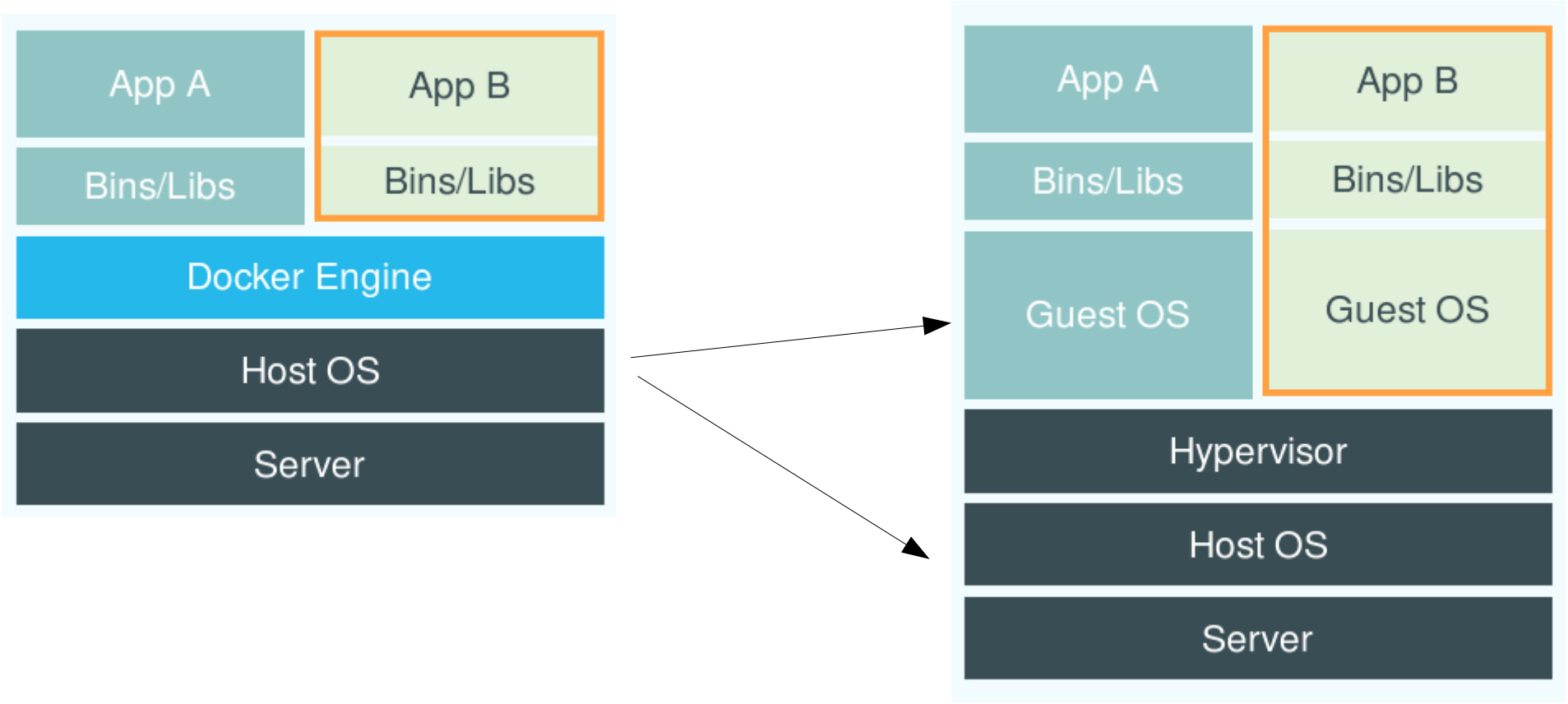
- Deploy containerized RHEL 6 applications to RHEL 7 without porting or changing source code
- Make use of innovations in Red Hat Enterprise Linux 7 without compromising the reliability and security of existing Red Hat Enterprise Linux 6 apps
- Available as part of your Red Hat Enterprise Linux subscription


docker

WHAT IS DOCKER?

Docker is an implementation of Linux Container technology, a toolset and image format.

Docker is NOT virtualization. Docker requires Linux kernel features.



Docker Containers

Virtualization

WHAT IS DOCKER?

Docker “Image” – Read Only Template. Union FS of layers to compose OS+Platform+App in one image. Layers can be used by other images.

```
# docker history 7413629aa833
IMAGE               CREATED             CREATED BY                                      SIZE
7413629aa833        57 minutes ago      /bin/sh -c #(nop) USER root                     0 B
dbcecf2c40c5        58 minutes ago      /bin/sh -c #(nop) USER jboss                    0 B
b02849afb76a        58 minutes ago      /bin/sh -c #(nop) WORKDIR /opt/jboss            0 B
ddf64d06c0ef        58 minutes ago      /bin/sh -c groupadd -r jboss -g 1000 && usera   295.3 kB
dc50405a6f74        59 minutes ago      /bin/sh -c yum -y install git saxon unzip &&    281.3 MB
eb5711564c0f        About an hour ago   /bin/sh -c yum -y update && yum clean all       255 MB
977d769f859b        About an hour ago   /bin/sh -c #(nop) MAINTAINER Greg Hoelzer gho   0 B
8dc6a04270df        5 months ago                                                        151.5 MB
```

Docker “Client” (rpm: docker) – Command line tool to interface with the Docker Engine

```
# docker ps
CONTAINER ID        IMAGE                              COMMAND                CREATED             STATUS
PORTS                    NAMES
dd775b942ff8        jbossrhel7bld:build5               "/run.sh"              24 hours ago        Up 24 hours
0.0.0.0:3322->22/tcp     jbossrhel7bld-slave1
ed7a15081f4b        jbossrhel6bld:build14              "/run.sh"              24 hours ago        Up 24 hours
0.0.0.0:2222->22/tcp     jbossrhel6bld-slave1
b90317dc3b13        eap61helloworld/7jdk7:build11      "/opt/jboss/eap-6.1/   25 hours ago        Up 25 hours 9990/tcp,
0.0.0.0:18080->8080/tcp eap61helloworld_7jdk7
```

DOCKER DAEMON

Docker daemon ('docker' rpm in RHEL, 'docker-io' in Fedora) – Runs containers from images within a kernel namespace and cgroups

```
Runs as a service:
# systemctl status docker
docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled)
   Active: active (running) since Wed 2014-11-26 14:03:50 EST; 5 days ago
     Docs: http://docs.docker.io
 Main PID: 2216 (docker)
   CGroup: /system.slice/docker.service
           ├─ 2216 /usr/bin/docker -d --selinux-enabled -H fd://
           ├─ 9039 docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 18080 -container-ip 172.17.0.82
-container-port 8080
           ├─10472 docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 2222 -container-ip 172.17.0.84
-container-port 22
           └─10566 docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 3322 -container-ip 172.17.0.85
-container-port 22
```

Uses loopback sparse file devicemapper storage in /var/lib/docker by default:

```
# du -h --apparent-size -s /var/lib/docker
106G /var/lib/docker
# du -h -s /var/lib/docker
9.9G  /var/lib/docker
```

(NOTE: RHEL Atomic Hosts uses direct LVM for more performance out of the box. This isn't realistic on RHEL standard because of varied storage configurations)

DOCKER NETWORKING

Uses docker0 bridge by default for network:

```
# ifconfig docker0
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1454
        inet 172.17.42.1  netmask 255.255.0.0  broadcast 0.0.0.0
        inet6 fe80::5484:7aff:fefe:9799  prefixlen 64  scopeid 0x20<link>
        ether 56:84:7a:fe:97:99  txqueuelen 0  (Ethernet)
        RX packets 689190  bytes 107560060 (102.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1672232  bytes 2490681128 (2.3 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# bridge link
165: vethec8d state UP : <BROADCAST,UP,LOWER_UP> mtu 1454 master docker0 state forwarding priority 32 cost 2
169: veth845f state UP : <BROADCAST,UP,LOWER_UP> mtu 1454 master docker0 state forwarding priority 32 cost 2
171: vetha958 state UP : <BROADCAST,UP,LOWER_UP> mtu 1454 master docker0 state forwarding priority 32 cost 2

# iptables -L -t nat -n
 Chain DOCKER (2 references)
target     prot opt source               destination
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:18080 to:172.17.0.82:8080
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:2222 to:172.17.0.84:22
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:3322 to:172.17.0.85:22
```

RED HAT SUPPORTS DOCKER

Red Hat provides Docker in RHEL7 in the extras channel

https://access.redhat.com/support/policy/updates/extras

Red Hat Enterprise Linux Extras Product Life Cycle

Red Hat is introducing a new channel in the Red Hat Customer Portal for Red Hat Enterprise Linux called “Extras.” The Extras channel is intended to give customers access to select, rapidly evolving technologies. These technologies may be updated more frequently than they would otherwise be in a Red Hat Enterprise Linux minor release. **The technologies delivered in the Extras channel are fully supported.**

Red Hat Container Certification:

http://www.redhat.com/en/about/press-releases/red-hat-announces-certification-for-containerized-applications-extends-customer-confidence-and-trust-to-the-cloud

REINVENTING THE WHEEL?

- Solaris Zones
- AIX WPARS
- Linux LXC
- Unix chroot

Why Docker is different:

- Docker Image format
- Docker toolchain
- Docker networking
- SELinux support
- Much much more...

DOCKER IN ACTION

- RHEL7 host with subscription management

```
# docker pull rhel7
# docker run -i -t --name some-name rhel7 bash
# yum install httpd
# exit
# docker ps -a
# docker commit -m comment -a author imageid name:tag
```

DOCKER IN ACTION: TAGS

Tags are mutable and one image can have more than one tag. Use tags symbolically.

```
# docker tag image yourname/httpd-rhel7:rhug
# docker tag image yourname/httpd:latest
```

```
REPOSITORY            TAG      IMAGE ID          CREATED             VIRTUAL SIZE
mheldebr/httpd-rhel7  rhug     dbe0a806463e      13 minutes ago       220.5 MB
mheldebr/httpd        latest   dbe0a806463e      13 minutes ago       220.5 MB
```

DOCKER IN ACTION: REGISTRY

- `docker push tag` – send it to the registry

```
server1 # docker tag httpd-rhel7 registry-host:5000/user/httpd-rhel7
server1 # docker push registry-host:5000/user/httpd-rhel7
```

- On another server, pull down that docker image via tag and run it

```
server2 # docker pull registry-host:5000/user/httpd-rhel7
```

DOCKER REGISTRY “GOTCHYAS”

- No security for the registry built in
  - Frontend it with apache/nginx:
  - https://github.com/docker/docker-registry/blob/master/ADVANCED.md
- Search not yet working from the docker command
  - Browser based search:
    - http://your.registry:5000/v1/search
    - http://your.registry:5000/v1/search?q=rhel

DOCKER COMMANDS

- `docker build` – build a docker image
- `docker run` *`image`* – run a container
  - Interactive: `-i -t (interactive, attach TTY)`
  - Background: `-d (daemonize)`
  - Ports: `-p` *`ip:`*`container:host`
  - Name: `--name name`
  - Linking: `--link name:alias`
    - sets environment variables in containers
    - <name>_PORT_<port>_<protocol>

DOCKER COMMANDS - CONTINUED

- `docker images` – what images are on this server
- `docker ps` – what containers are running on the server
- `docker ps -a` – what containers are and were running on the server

DOCKER COMMANDS - CONTINUED

- `docker logs` *`container`* – stdout of a container
- `docker inspect` *`container`* – JSON metadata
- `docker history` *`image`* – layers of an image
- `docker rm` *`container`* – delete a container
- `docker rmi` *`image`* – delete an image

DOCKERFILE – BUILDING DOCKER IMAGES

- Choose a base image
  - Red Hat registry RHEL image
  - docker load from your own server
- Install your platform
- Configuration of OS and platform
- Install your application

DOCKERFILE - EXAMPLE

```
FROM rhel7
MAINTAINER you
# Update image
RUN yum update -y && yum clean all
# Install platform
RUN yum install httpd -y && yum clean all
# Create an index.html file
ADD index.html /var/www/html/index.html'
#Run httpd in the foreground
ENTRYPOINT [“/usr/sbin/httpd”, “-DFOREGROUND”]
```

DOCKERFILE - BUILDING

```
# mkdir httpd-project
# cd httpd-project
# cp /src/index.html .
# vi Dockerfile
# docker build -t httpd-rhel7 .
Sending build context to Docker daemon 3.584 kB
Sending build context to Docker daemon
Step 0 : FROM rhel7
 ---> bef54b8f8a2f
Step 1 : MAINTAINER you
 ---> Running in ba42e5f739b5
 ---> 7a132d9b0dae
Removing intermediate container ba42e5f739b5
```

DOCKERFILE - BUILDING

```
Step 2 : RUN yum update -y && yum clean all
 ---> Running in 4cb8ce9be33a
Loaded plugins: product-id, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package ca-certificates.noarch 0:2013.1.95-71.el7 will be updated
<SNIP>
Cleaning up everything
 ---> d159981178f4
Removing intermediate container 4cb8ce9be33a
```

DOCKERFILE - BUILDING

```
Step 3 : RUN yum install httpd -y && yum clean all
 ---> Running in 0cae6a32272a
Loaded plugins: product-id, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.6-18.el7_0 will be installed
--> Processing Dependency: httpd-tools = 2.4.6-18.el7_0 for package: httpd-2.4.6-18.el7_0.x86_64
<SNIP>
Cleaning up everything
 ---> 98603b985c90
Removing intermediate container 0cae6a32272a
Step 4 : ADD index.html /var/www/html/index.html'
 ---> 781d160e0124
Removing intermediate container 7938e8e72b23
```

DOCKERFILE – RUN IT

```
Step 5 : ENTRYPOINT [“/usr/sbin/httpd”, “-DFOREGROUND”]
 ---> Running in 04b4aa491300
 ---> ea0f5eb865da
Removing intermediate container 04b4aa491300
Successfully built ea0f5eb865da
```

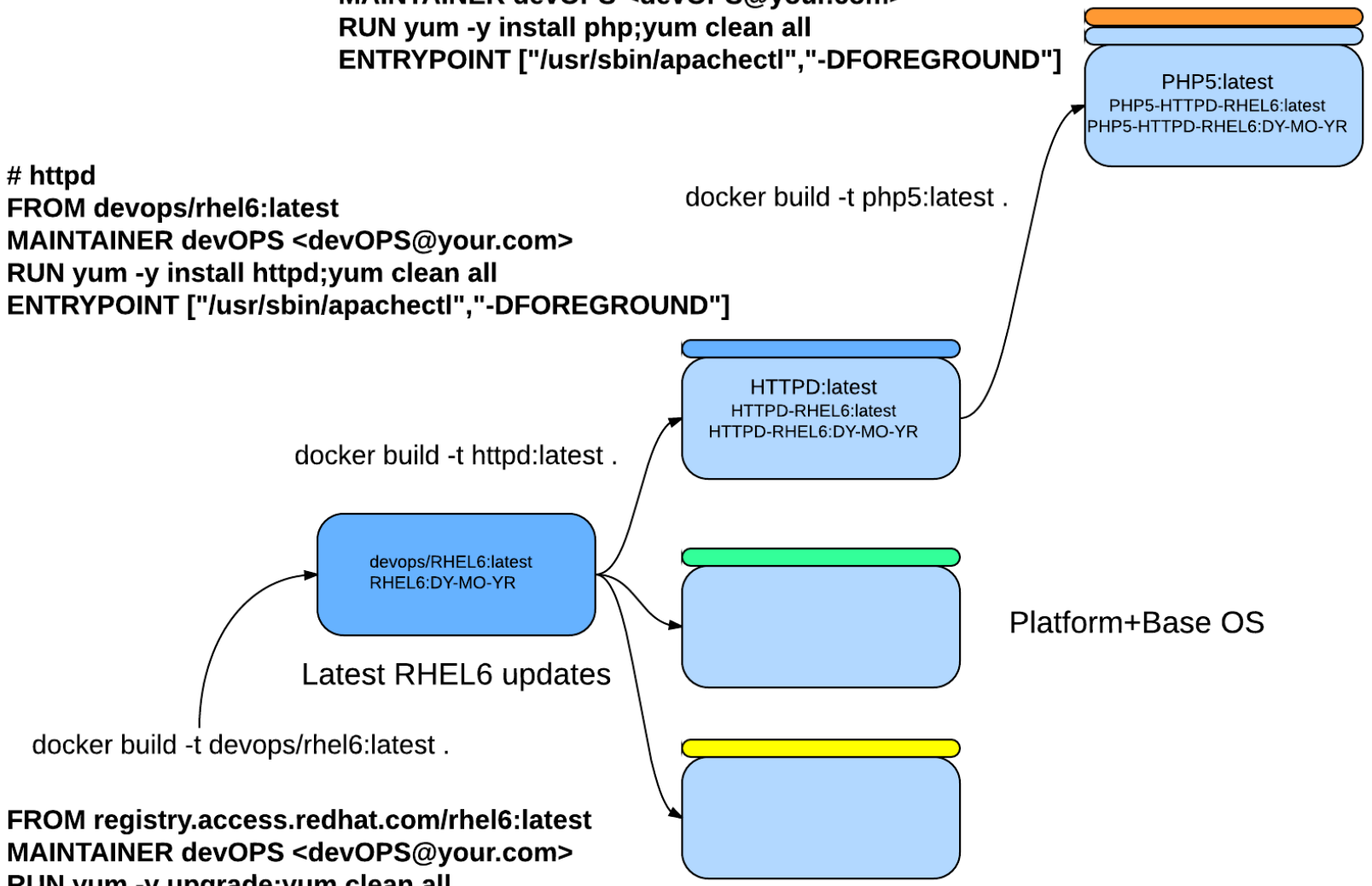
Run it:

```
# docker run -d -p 8081:80 httpd-rhel7
# curl localhost:8081
<HTML><HEAD></HEAD><BODY>Webpage!</BODY>
```

Next -> WORKFLOW

 # php5 + httpd
FROM httpd:latest
MAINTAINER devOPS <devOPS@your.com>
RUN yum -y install php;yum clean all
ENTRYPOINT ["/usr/sbin/apachectl","-DFOREGROUND"]

 # httpd
FROM devops/rhel6:latest
MAINTAINER devOPS <devOPS@your.com>
RUN yum -y install httpd;yum clean all
ENTRYPOINT ["/usr/sbin/apachectl","-DFOREGROUND"]



FROM registry.access.redhat.com/rhel6:latest
MAINTAINER devOPS <devOPS@your.com>
RUN yum -y upgrade;yum clean all

DOCKER – INSPECTING CONTAINERS

- If entrypoint is defined arguments are passed to the entrypoint command instead of a shell
  - Override the entrypoint

```
# docker run -i -t --rm \
   --name “mycontainer” --entrypoint /bin/bash httpd-rhel7
```

  - Enter the namespace of a running container

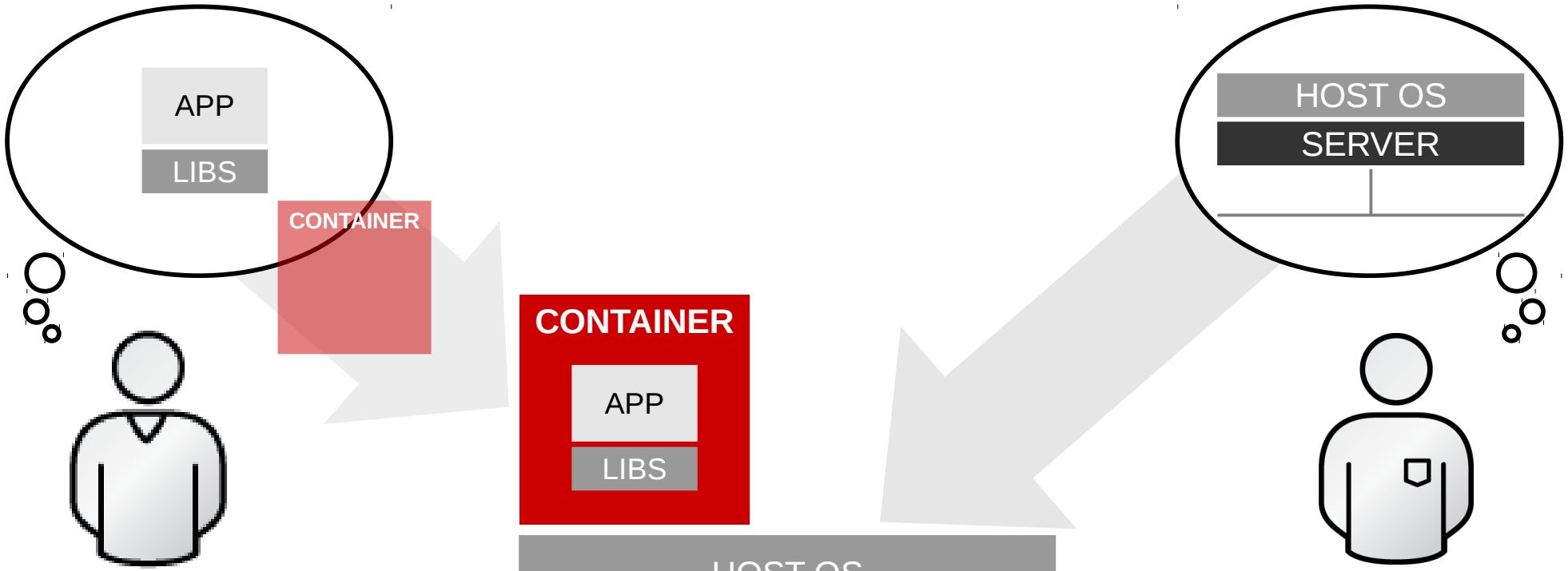
```
# docker exec -i -t mycontainer /bin/bash

bash-4.2# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 18:10 ?        00:00:00 /bin/bash
root         8     0  0 18:10 ?        00:00:00 /bin/bash
root        38     8  0 18:16 ?        00:00:00 ps -ef
```

DOCKER – RUNNING APPLICATIONS

- Run one thing per image
- Connect containers together
- Run traditional daemons in the foreground in the image
    - ssh -D
    - httpd -DFOREGROUND
- Plan your ports for your containers
    - You can share ports with a different ip address on a single server

CONTAINERS ENABLE CONTINUOUS DELIVERY



DEVELOPMENT FOCUS

- CODE
- APPLICATIONS
- LIBRARIES / DEPENDENCIES
- DATA STORE

OPERATIONS FOCUS

- HARDWARE
- OPERATING SYSTEM
- NETWORKING
- MONITORING

WHAT ABOUT DENSITY?

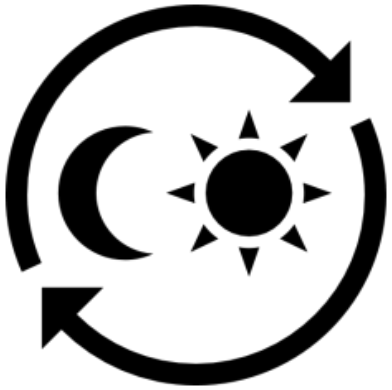
10 virtual machines

---

100 containers

OPERATIONAL EFFICIENCY
TIME TO SET UP

27 HRS



PHYSICAL
SERVER

12 MINS

VIRTUAL
MACHINE

10 SECS

CONTAINER
INSTANCE

KUBERNETES

KUBERNETES

- An orchestration engine for containers.
  - Runs on RHEL7 and RHEL7 Atomic Host
  - Provides a declarative language to specify desired system state.
- In June 2014 Google open sourced a container management project
- Named for a shipmaster in Greek
- Red Hat is collaborating in the kubernetes project

http://www.redhat.com/en/about/blog/red-hat-and-google-collaborate-kubernetes-manage-docker-containers-scale

WHAT IS KUBERNETES? - CONTINUED

- Sometimes called:
    - kube
    - k8s (that's 'k' + 8 letters + 's')
- Start, stop, update, and manage a cluster of machines running containers in a consistent and maintainable way.
- Additional functionality to make containers easier to use in a cluster (reachability and discovery).
- Kubernetes does NOT and will not expose all of the 'features' of the docker command line.

WHAT DOES IT SOLVE?

- Describe networked applications simply
  - Put app components into Docker containers
  - Containers see flat network
  - Simple relationships between containers
  - Make networked storage easy to use
- Build resiliency and redundancy into infra
  - Health checking, load balancing, failure detection
  - Run many workloads on the same machines

KUBERNETES PRIMITIVES AND KEYWORDS

- Master
- Minion/Node
- Pod
- Replication Controller
- Service
- Label

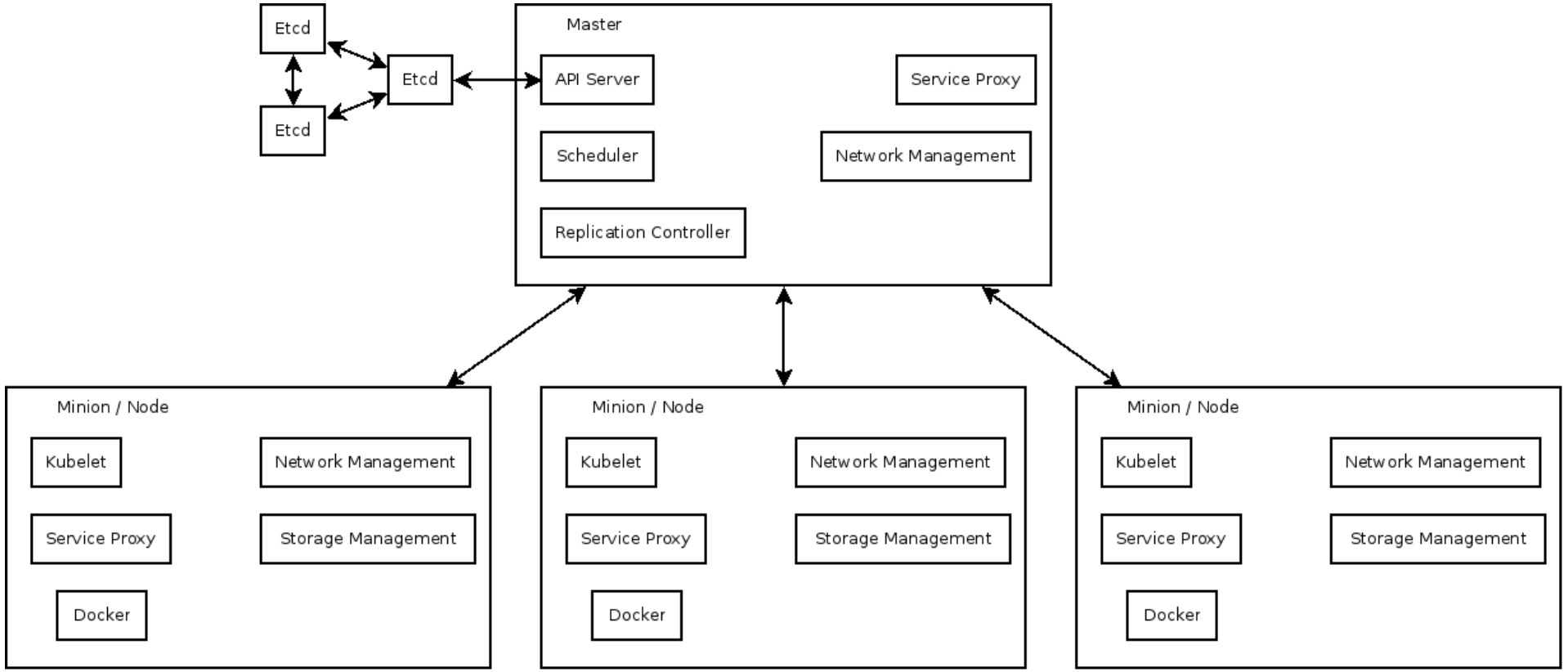
MASTER

- Typically consists of:
  - kube-apiserver
  - kube-scheduler
  - kube-controller-manager
  - etcd
- Might contain:
  - kube-proxy
  - a network management utility

MINION (NODE)

- Typically consists of:
  - kubelet
  - kube-proxy
  - cAdvisor
- Might contain:
  - a network management utility
- May be referred to by either name.

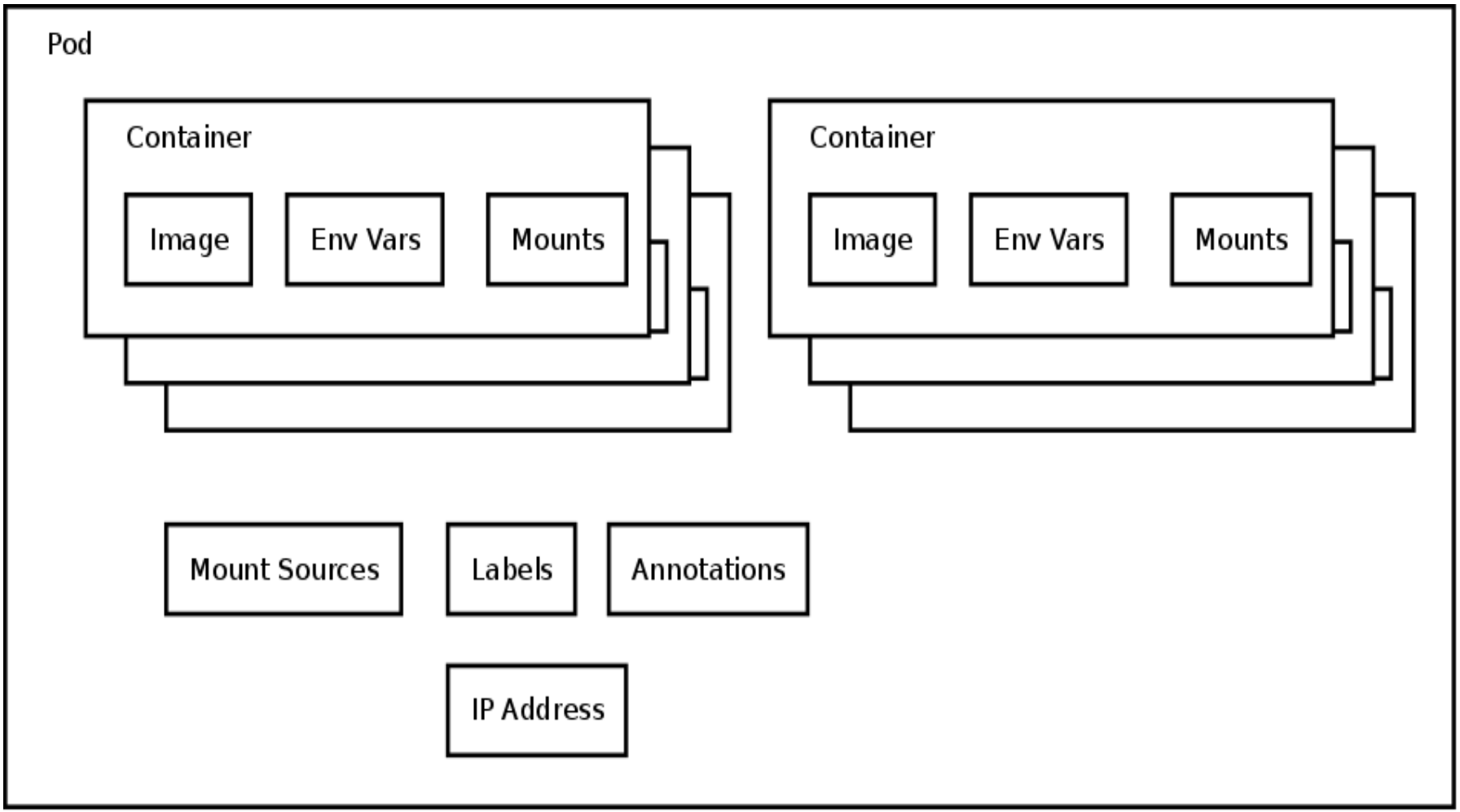
SYSTEMS AND BINARIES



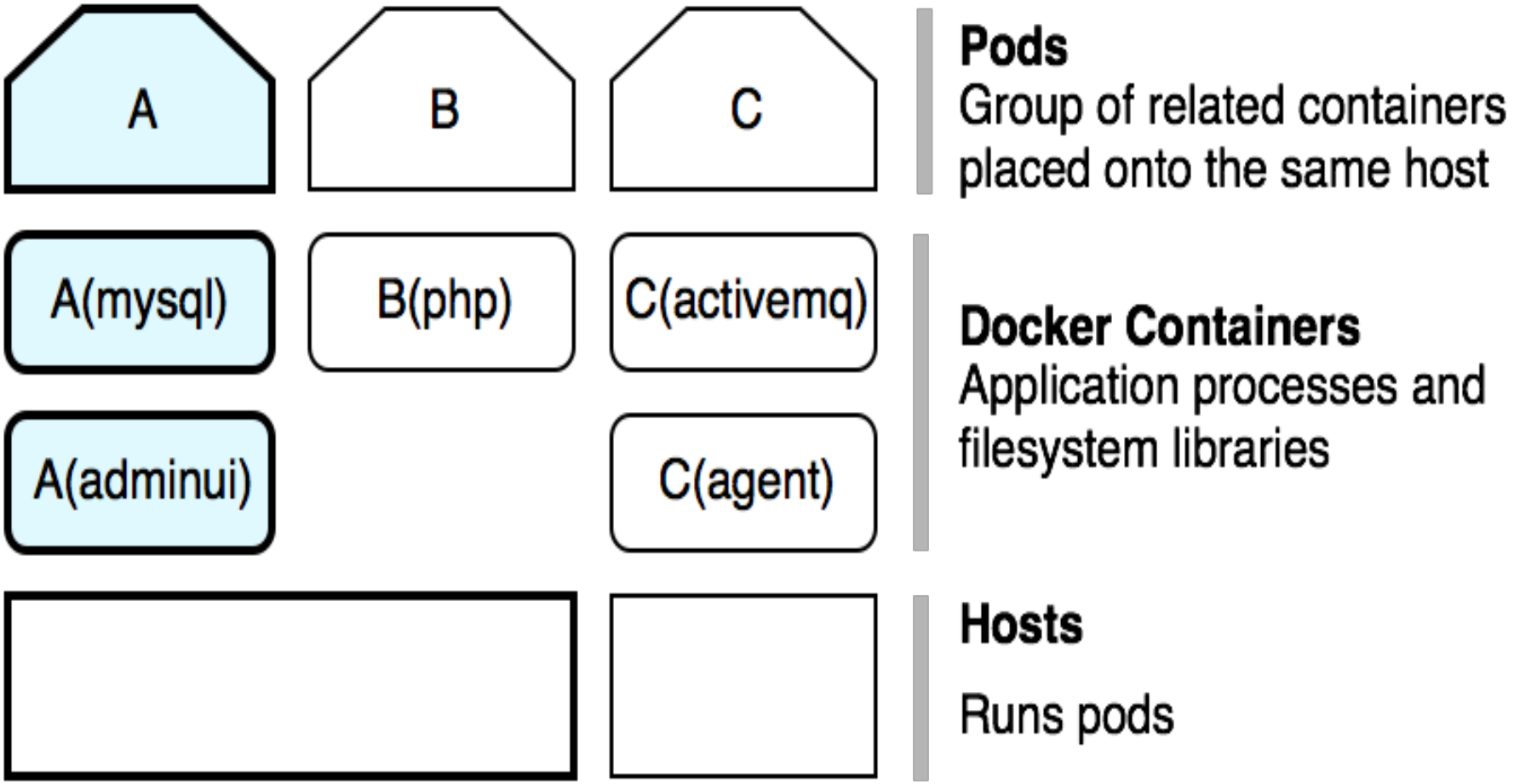
POD

- Single schedulable unit of work
  - Can not move between machines
  - Can not span machines
- One or more containers
  - Shared network namespace
- Metadata about the container(s)
- Env vars – configuration for the container
- Every pod gets an unique IP
  - Assigned by the container engine, not kube!

POD


Pod
Container
Image
Env Vars
Mounts
Container
Image
Env Vars
Mounts
Mount Sources
Labels
Annotations
IP Address

POD (CONTINUED)



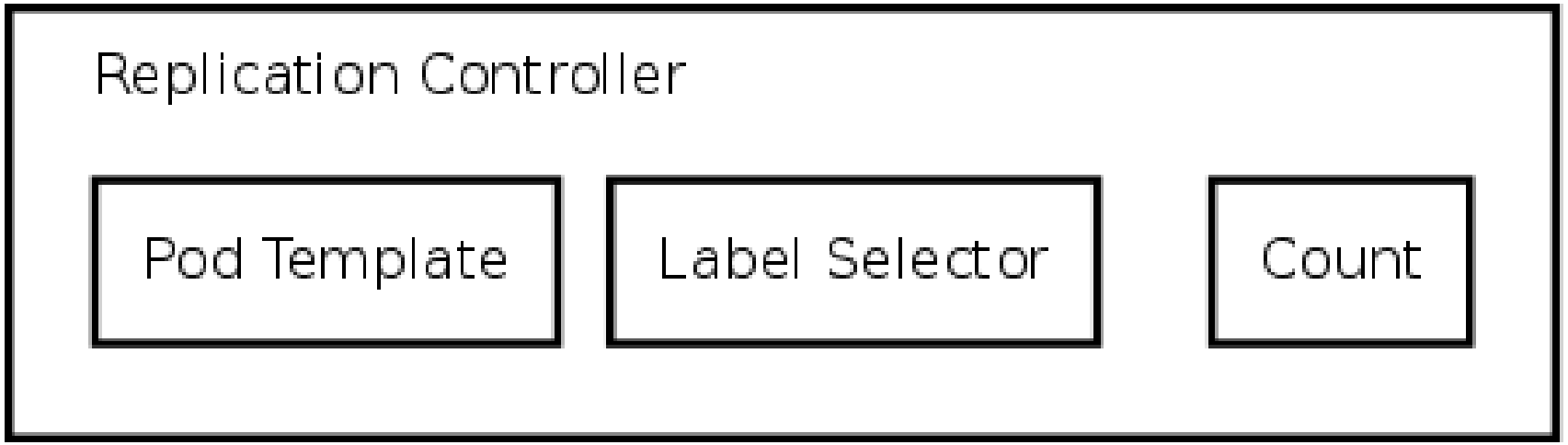
KUBERNETES POD DEFINITION EXAMPLE

```
{"apiVersion": "v1beta1",”kind": "Pod","id": "apache","namespace": "default",
    "labels": {“name": "apache"},
    "desiredState": {"manifest": {
        "version": "v1beta1","id": "apache","volumes": null,
        "containers": [ { "name": "master","image": "fedora/apache",
            "ports": [ { "containerPort": 80,"hostPort": 80,"protocol": "TCP" } ],
         } ],
         "restartPolicy": { "always": {} }
    }, },
}
```

REPLICATION CONTROLLER

- Consists of
    - Pod template
    - Count
    - Label Selector
- Kube will try to keep $count copies of pods matching the label selector running
- If too few copies are running the replication controller will start a new pod somewhere in the cluster

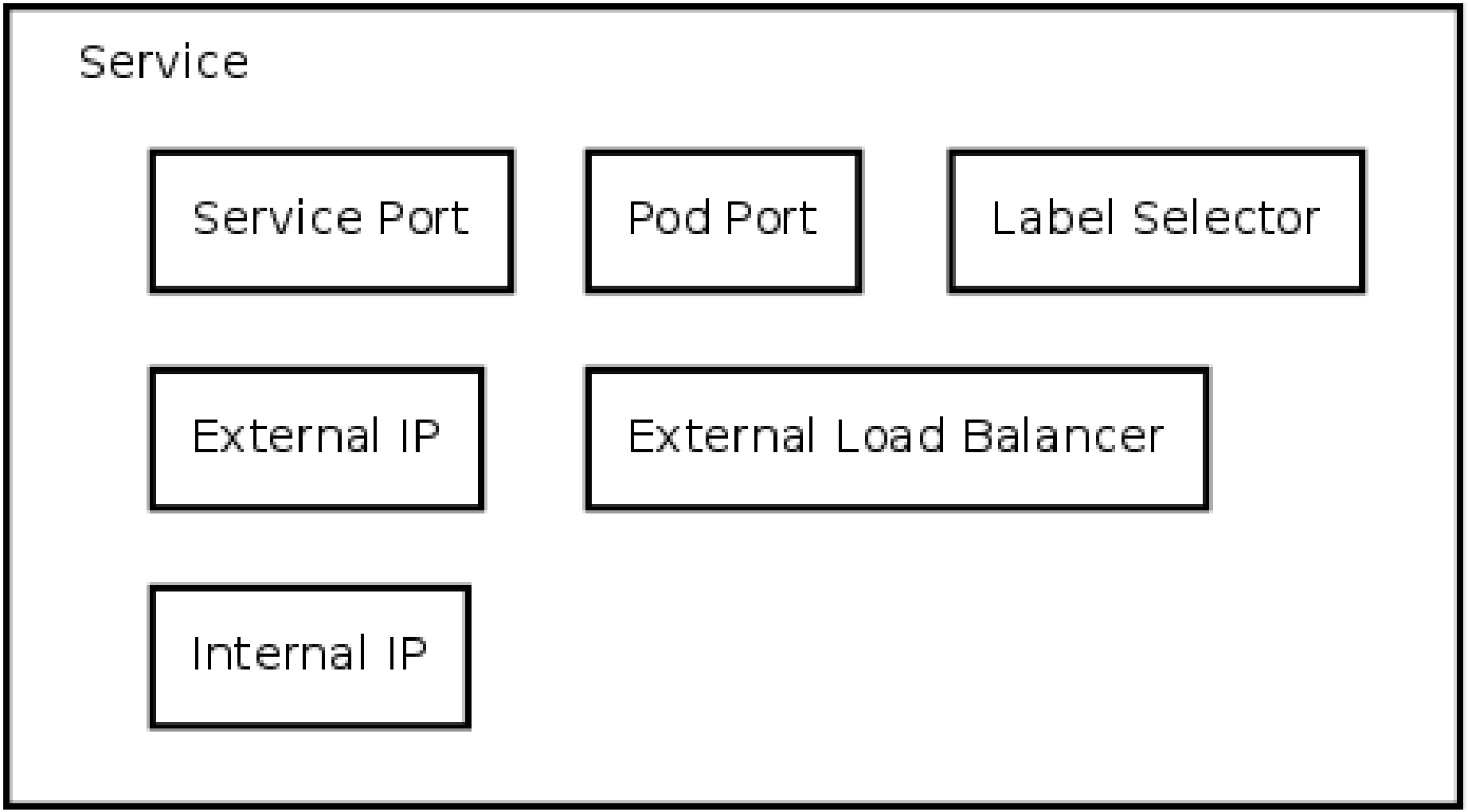
REPLICATION CONTROLLER



SERVICES

- How 'stuff' finds pods which could be anywhere
- Define:
    - What port in the container
    - Labels on pods which should respond to this type of request
- Can define:
    - What the 'internal' IP should be
    - What the 'external' IP should be
    - What port the service should listen on

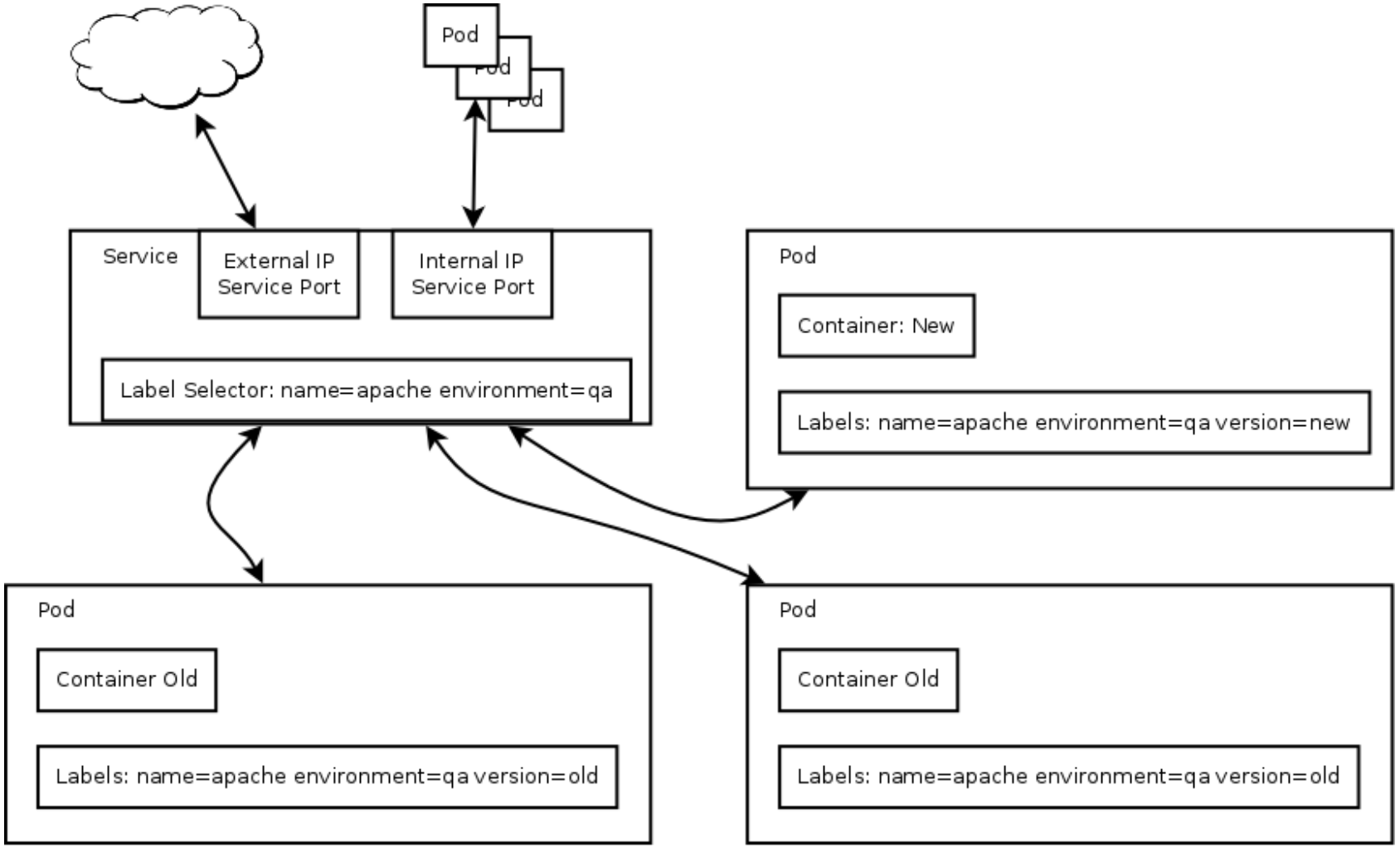
SERVICES



LABELS

- List of key=value pairs
- Attached to all objects
- Currently used in 2 main places
    - Matching pods to replication controllers
    - Matching pods to services
- Objects can be queried from the API server by label

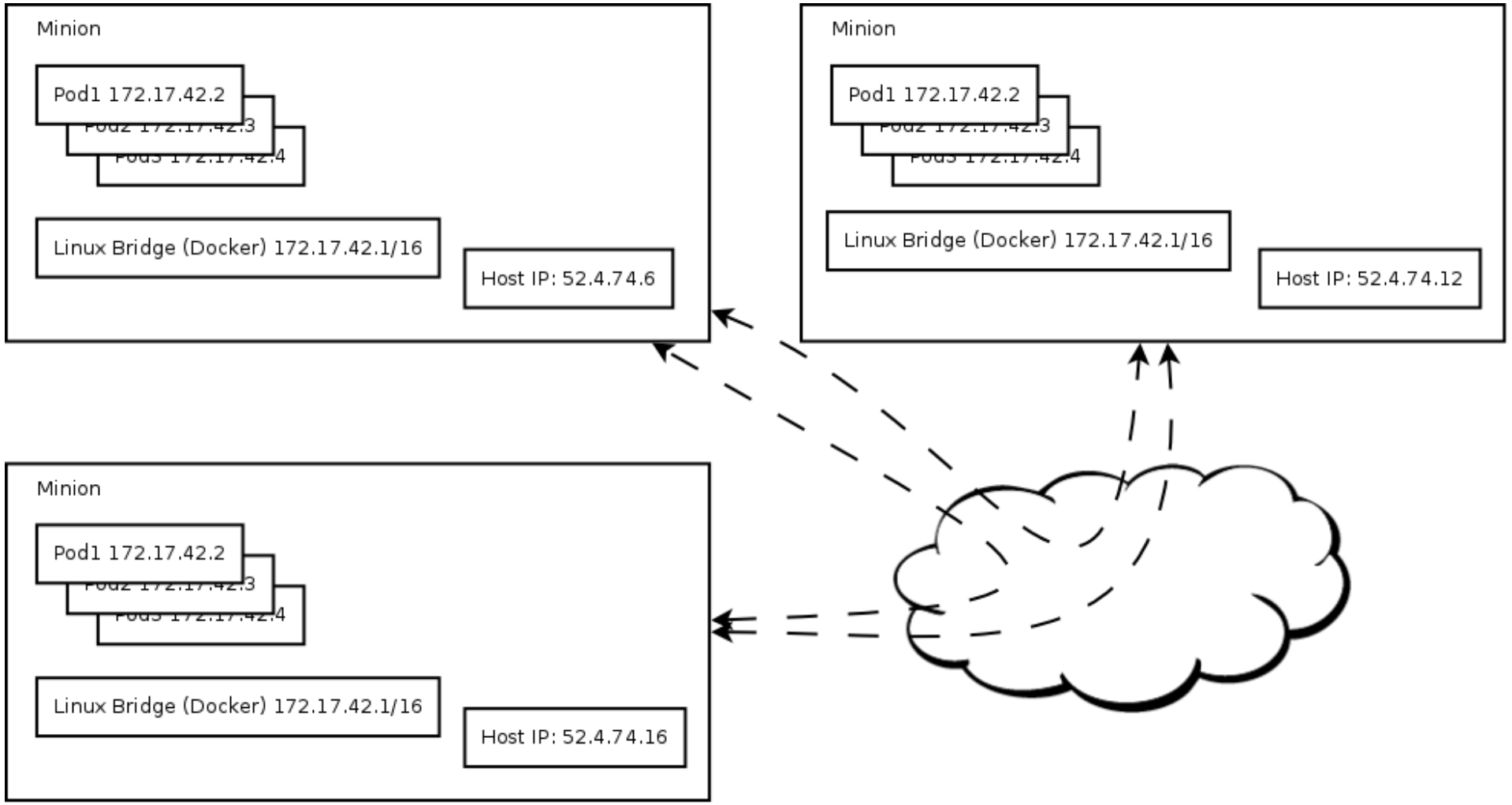
SERVICES AND LABELS



NETWORKING SETUP (WORK IN PROGRESS)

- Networking is a docker problem – not kube
  - Kube makes those problems apparent!
  - If any two docker containers on any two hosts can talk over IP, kube will just work.
    - Every container within a pod gets an unique IP within the same network space.
    - Overlay networks allow this to “just work”

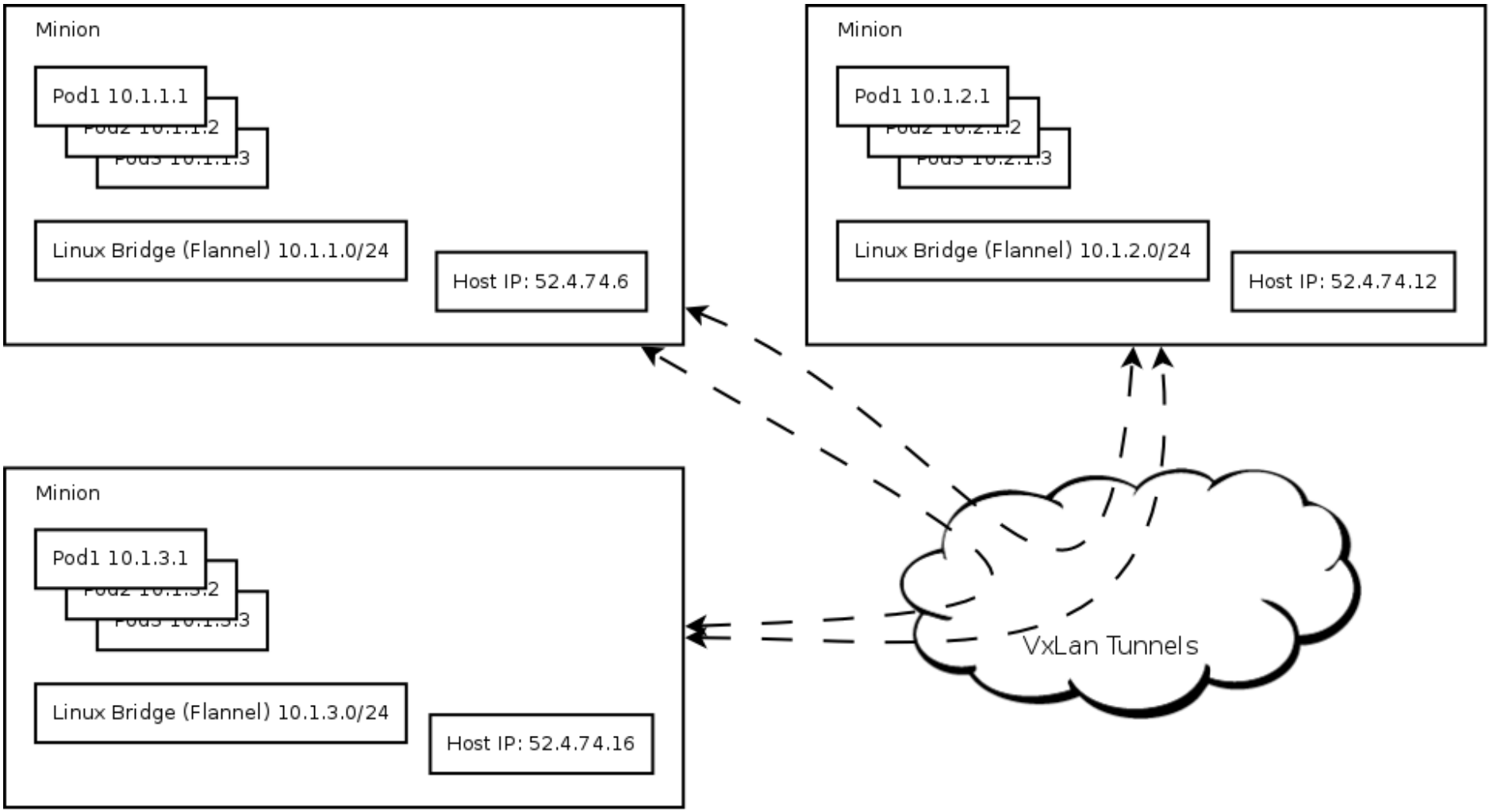
Networking Docker Out Of The Box



NETWORKING SETUP (UNDER DEVELOPMENT)

- Flannel (Not yet in RHEL)
  - Extremely easy configuration
  - Can create a vxlan overlay network
  - Can configure docker to launch pods in this overlay
  - Pods just work!
- There are many other solutions.
  - This one is easy.

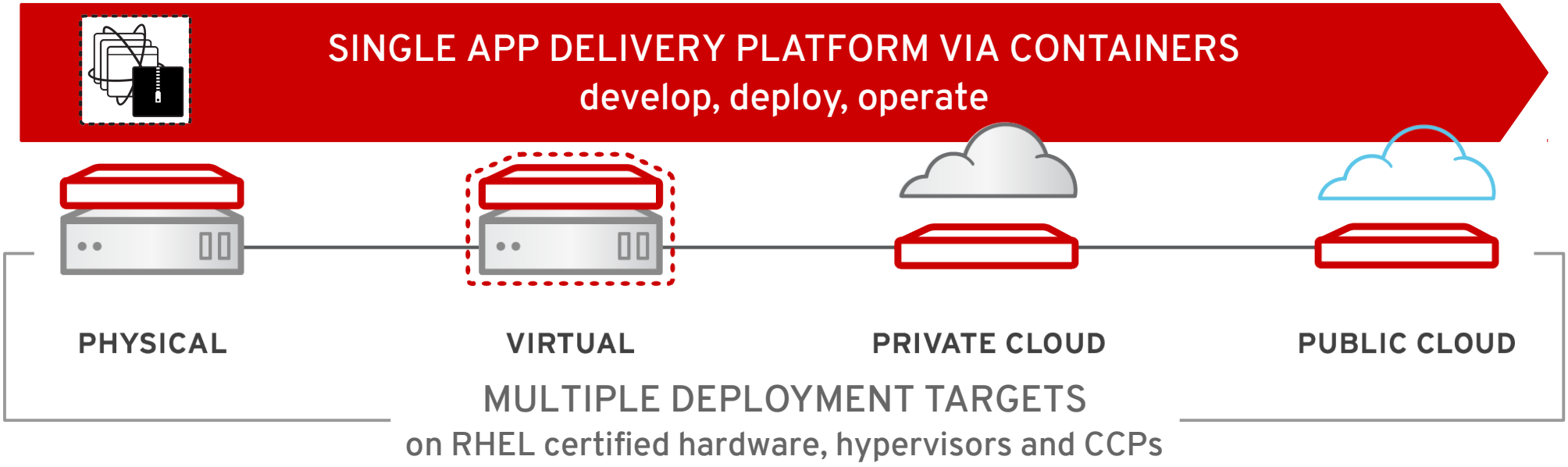
Networking with an overlay network



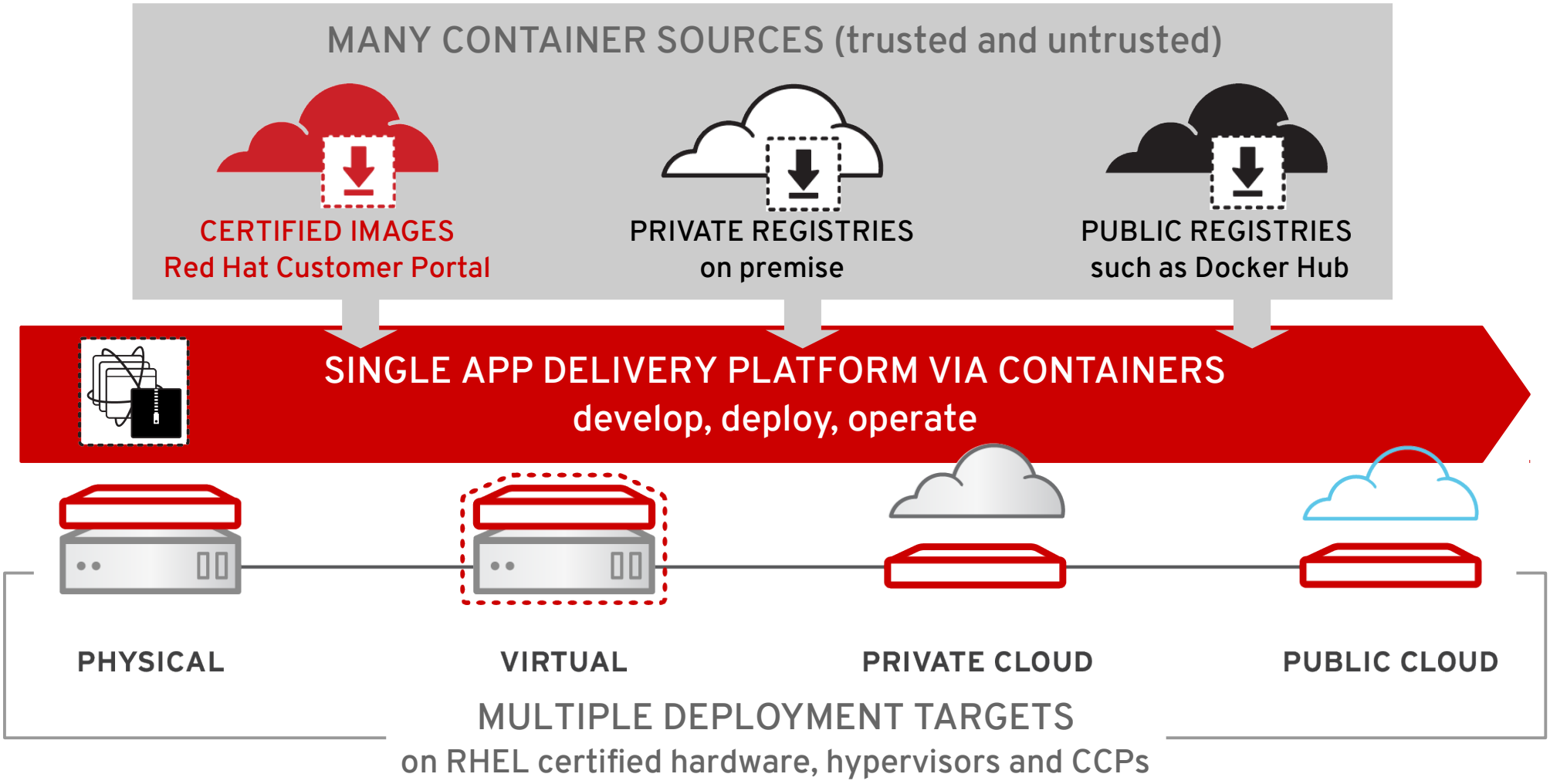
BUT ... IT'S MORE THAN JUST THE CONTAINER

RED HAT® ENTERPRISE LINUX® ATOMIC HOST

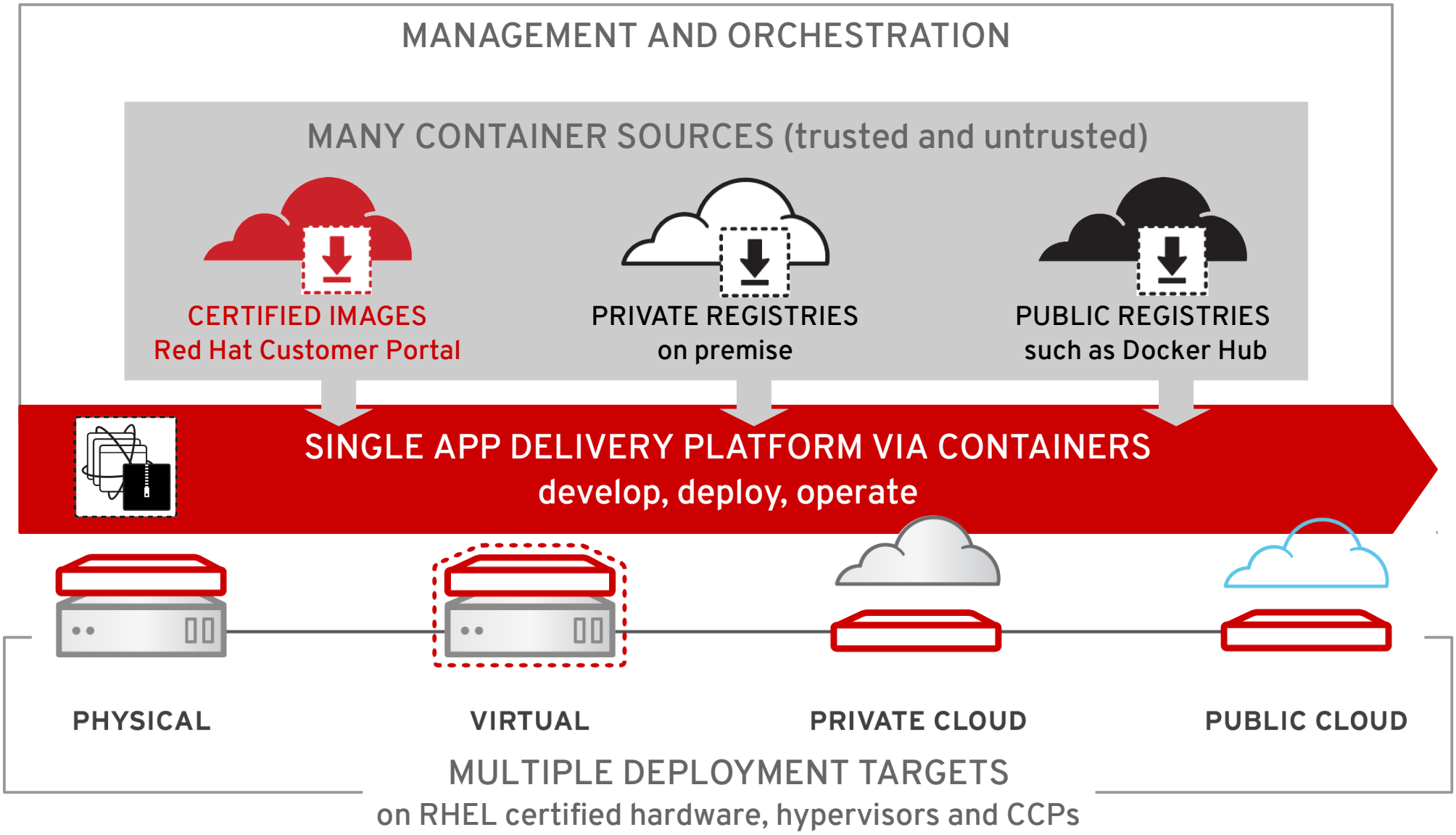
IT'S MORE THAN THE CONTAINER



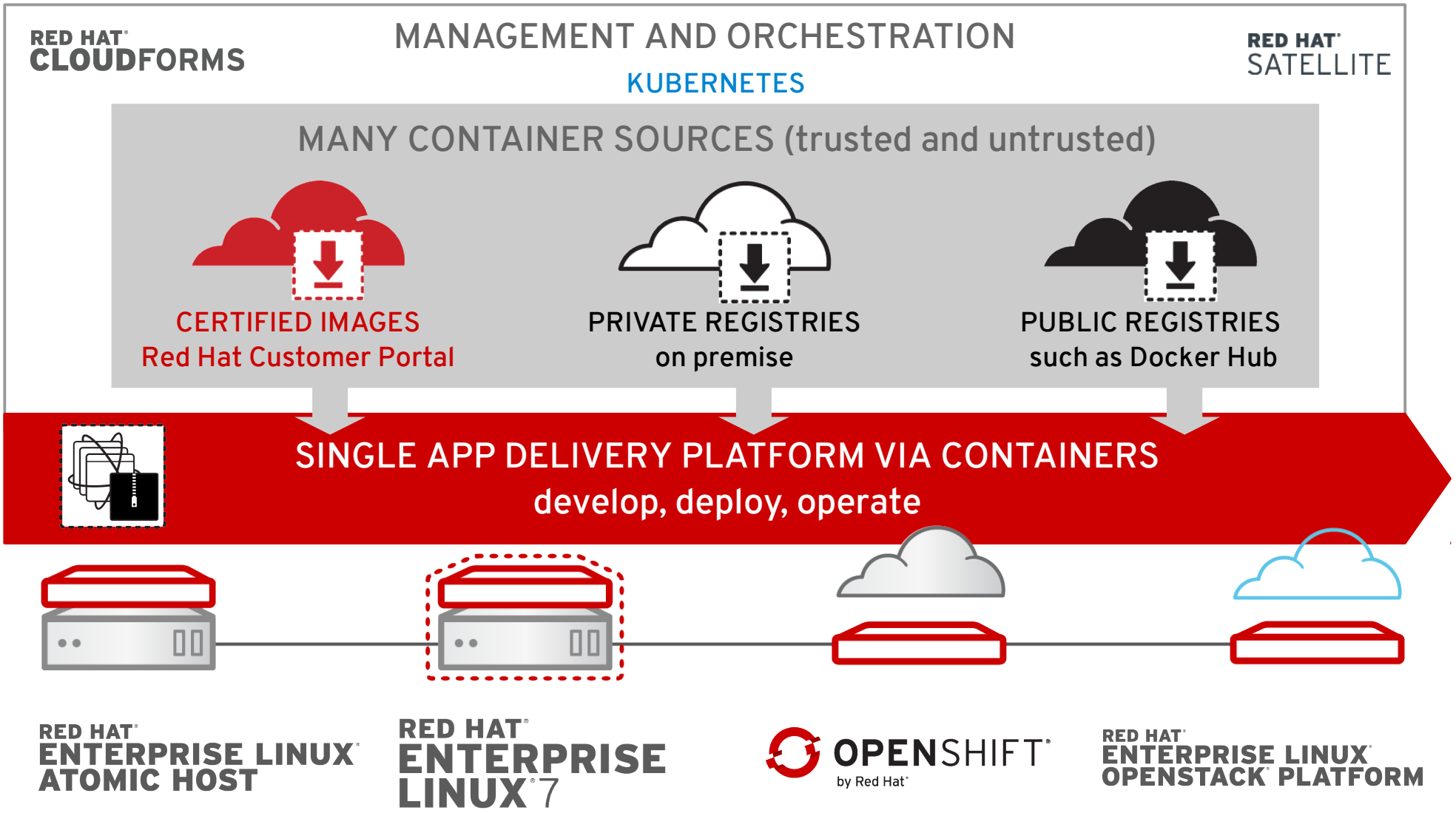
IT'S MORE THAN THE CONTAINER



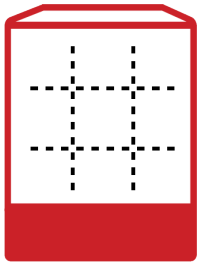
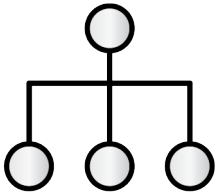
IT'S MORE THAN THE CONTAINER



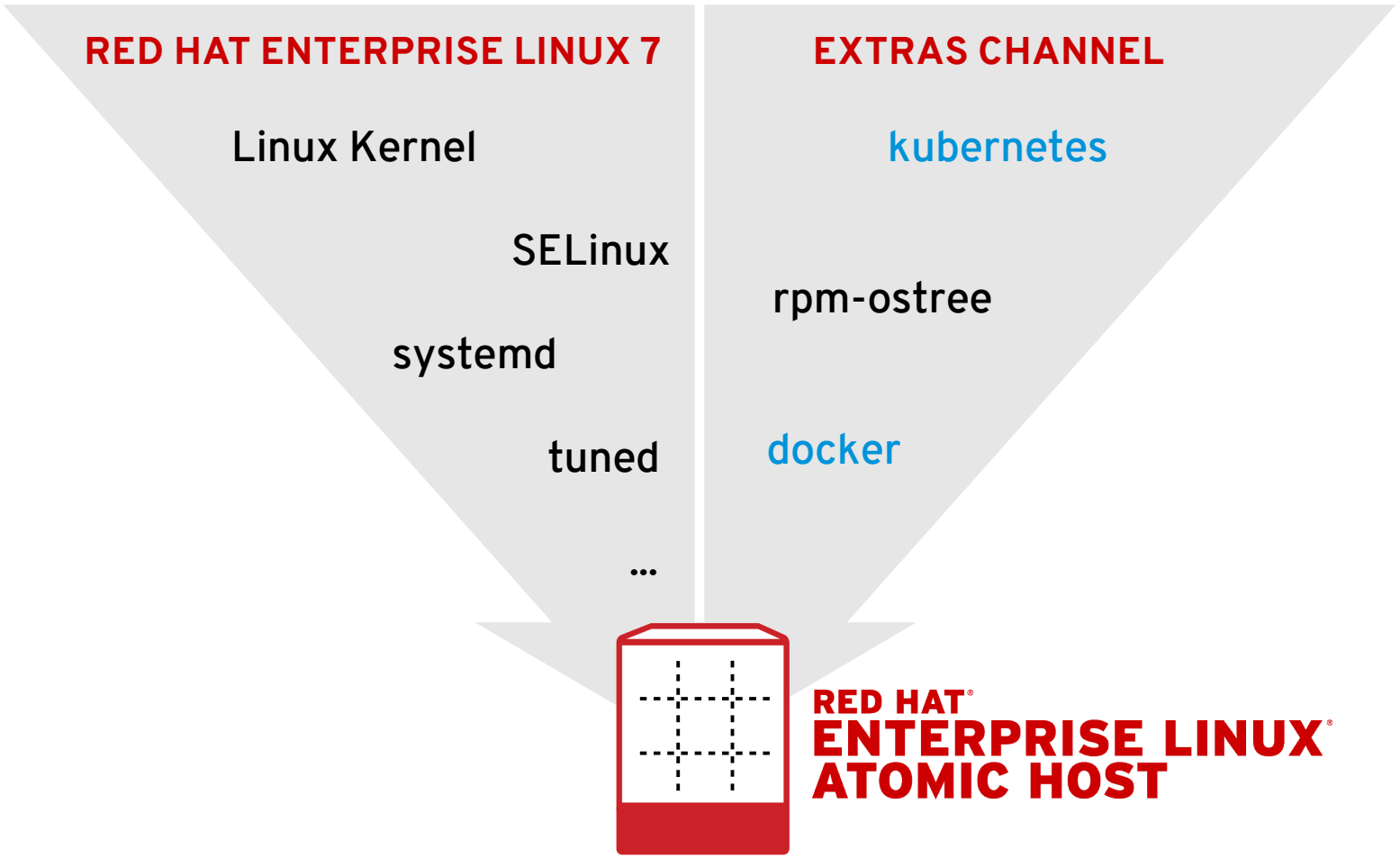
IT'S MORE THAN THE CONTAINER



RED HAT ENTERPRISE LINUX ATOMIC HOST (BETA)

IT IS RED HAT ENTERPRISE LINUX	OPTIMIZED FOR CONTAINERS		
	 **MINIMIZED FOOTPRINT**	 **SIMPLIFIED MAINTENANCE**	 **ORCHESTRATION AT SCALE**
Inherits the complete hardware ecosystem, military-grade security, stability and reliability for which Red Hat Enterprise Linux is known.	Minimized host environment tuned for running Linux containers while maintaining compatibility with Red Hat Enterprise Linux.	Atomic updating and rollback means it's easy to deploy, update, and rollback using imaged-based technology.	Build composite applications by orchestrating multiple containers as microservices on a single host instance.

RED HAT ENTERPRISE LINUX ATOMIC HOST (BETA)



RHEL ATOMIC BETA

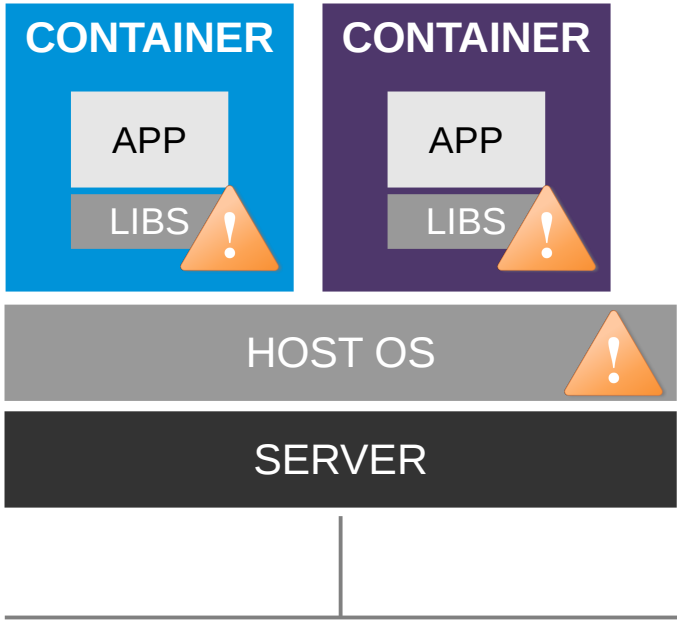
- Container Oriented OS based on RHEL7 and Project Atomic
    - Docker and Kubernetes
    - All applications are to run in a container
- Atomic updates of the OS
    - rpm-ostree

SECURING HOSTS AND CONTAINERS

RED HAT CONTAINER CERTIFICATION

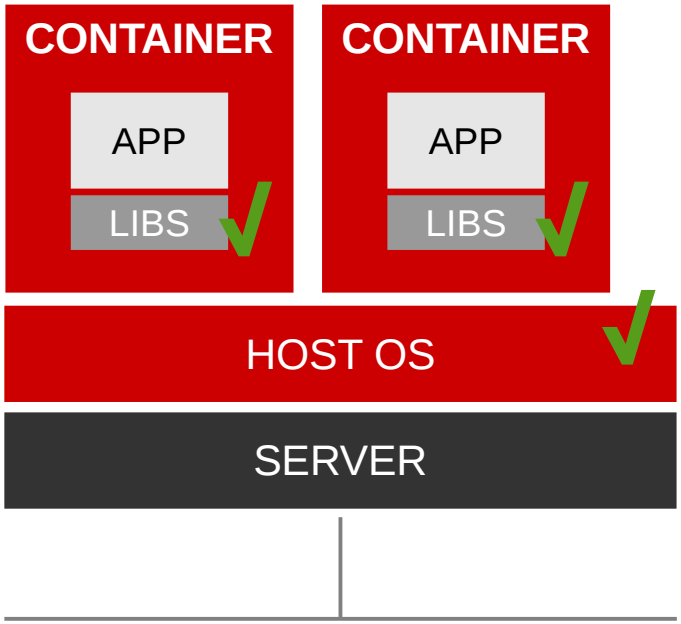
UNTRUSTED

- How can you validate what's in the host and the containers? Will it compromise your infrastructure?
- It "should" work from host to host, but can you be sure?



CERTIFIED

- Trusted source for the host and the containers
- Instant portability across multiple hosts



ONLY RED HAT...

- Provides a single application delivery platform across deployment targets
    - Certified to run on common hardware, hypervisors and cloud service providers
- Delivers a safe, cost-effective way to consume containers
- Transforms application delivery with the first credible hybrid cloud platform for container-based applications and services

DEMO!

LEARN MORE

- Access the Red Hat Enterprise Linux Atomic Host Beta
  https://access.redhat.com/products/red-hat-enterprise-linux/atomic-host-beta
- Stay informed via the Red Hat Enterprise Linux blog
  http://rhelblog.redhat.com/tag/containers/

QUESTIONS?

Thank You!

Adam Miller (@TheMaxamillion)
Senior Software Engineer
OpenShift Online Release Engineering
January 2015 RHUG