RED HAT CONTAINERS LAB

Atomic + Docker + Kubernetes ...

For CA Technologies
29.04.2016

Alfred Bach
Partner Enablement Manager EMEA
abach@redhat.com

AGENDA

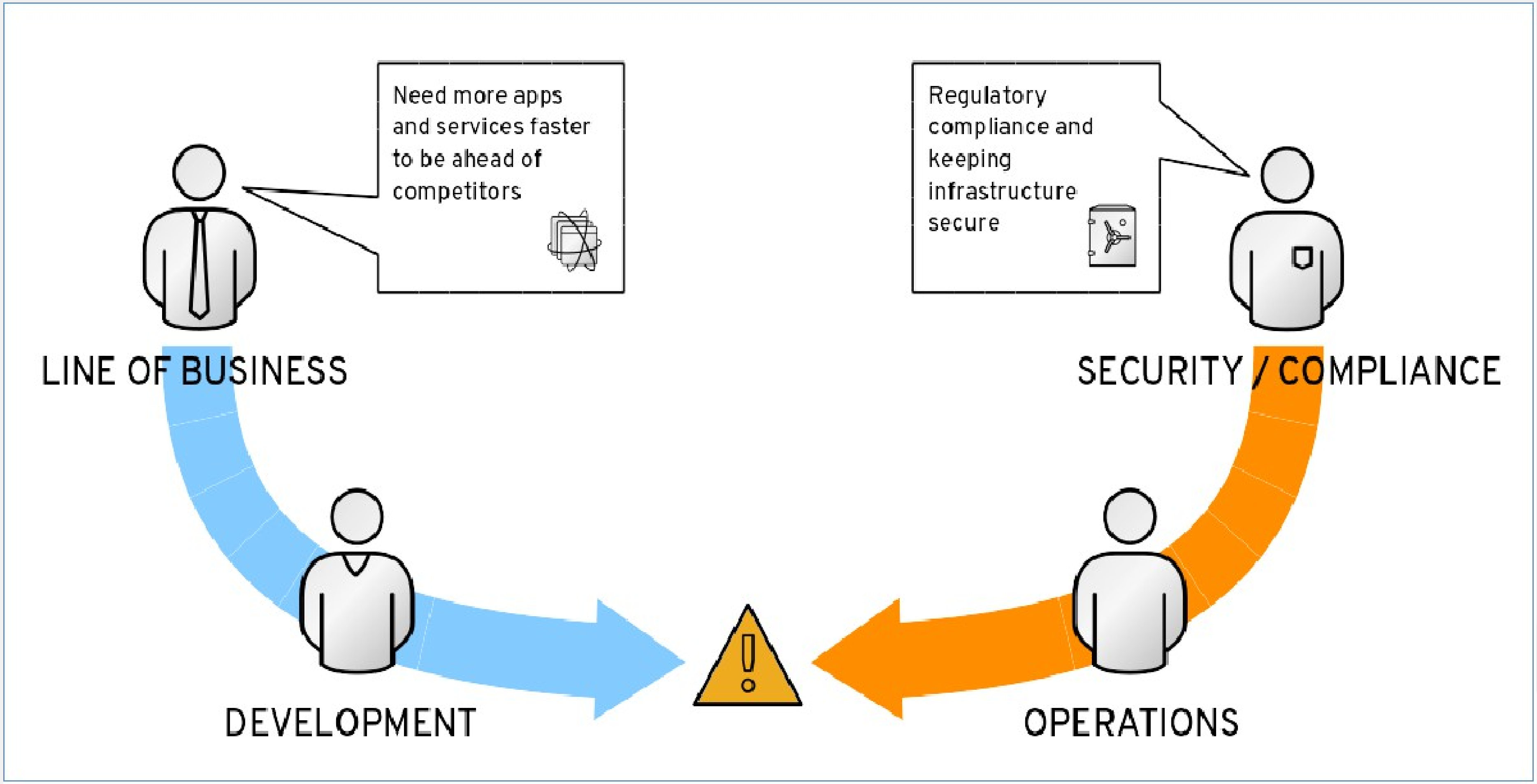
Red Hat Containers

- Red Hat strategy for containers
- Technologies
  - Containers
  - Docker
  - Kubernetes
  - Atomic
- How RHEL Atomic works
- Openshift v3

Why Does the CA SaaS Operating Platform Use OpenShift by Red Hat?

Red Hat strategy for containers

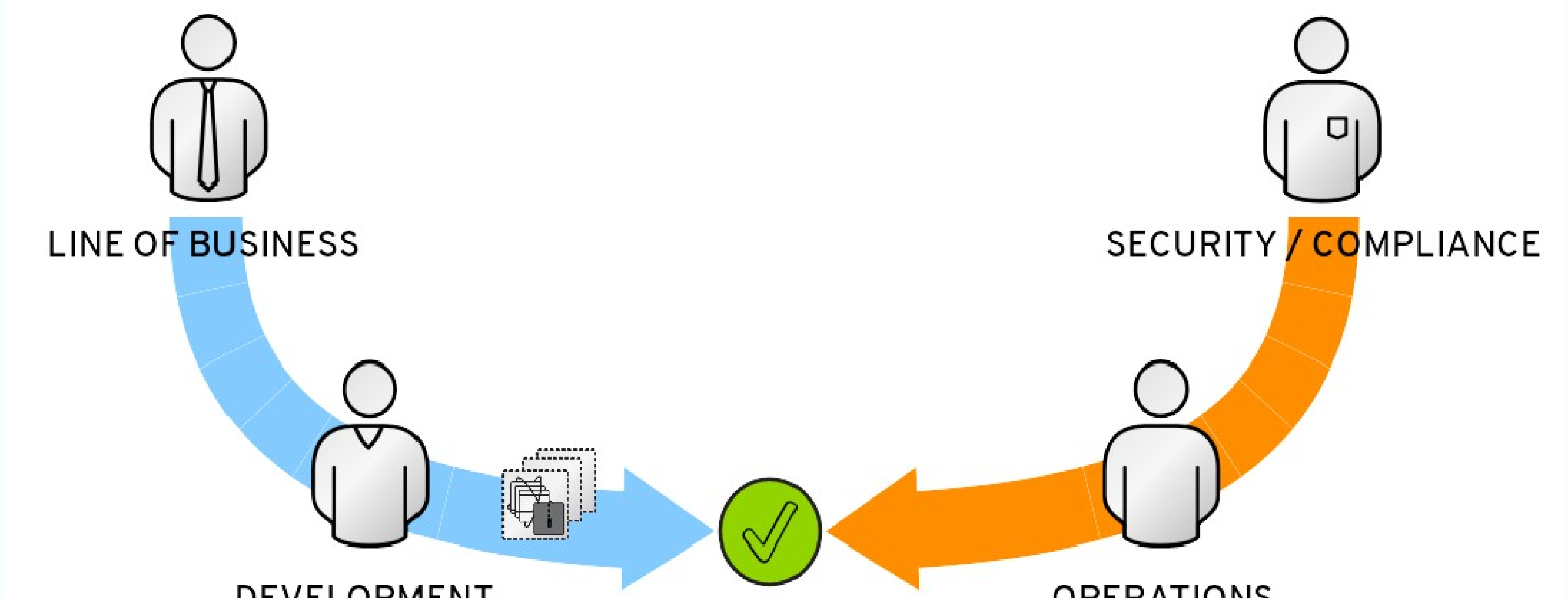
The problem



Need more apps and services faster to be ahead of competitors

Regulatory compliance and keeping infrastructure secure

Solution:
Application delivery via containers



Increases business agility

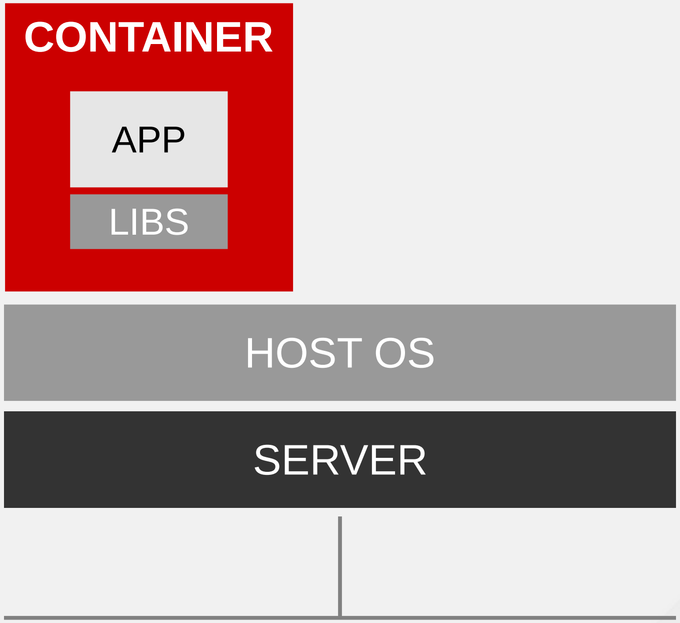
Accelerates developer productivity

Scales to public cloud faster

What are linux Containers?

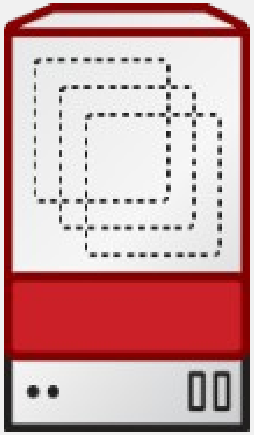
Software packaging concept that typically includes an application and all of its runtime dependencies.

- 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
  - Docker

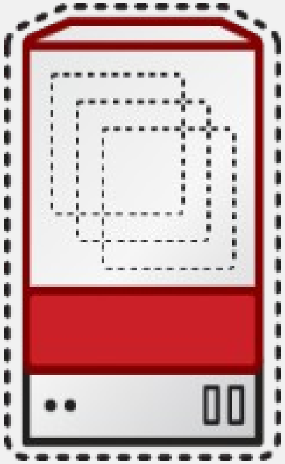


How can I use linux Containers?

It is available in two flavours Red Hat Enterprise Linux and Red Hat Enterprise Linux Atomic Host



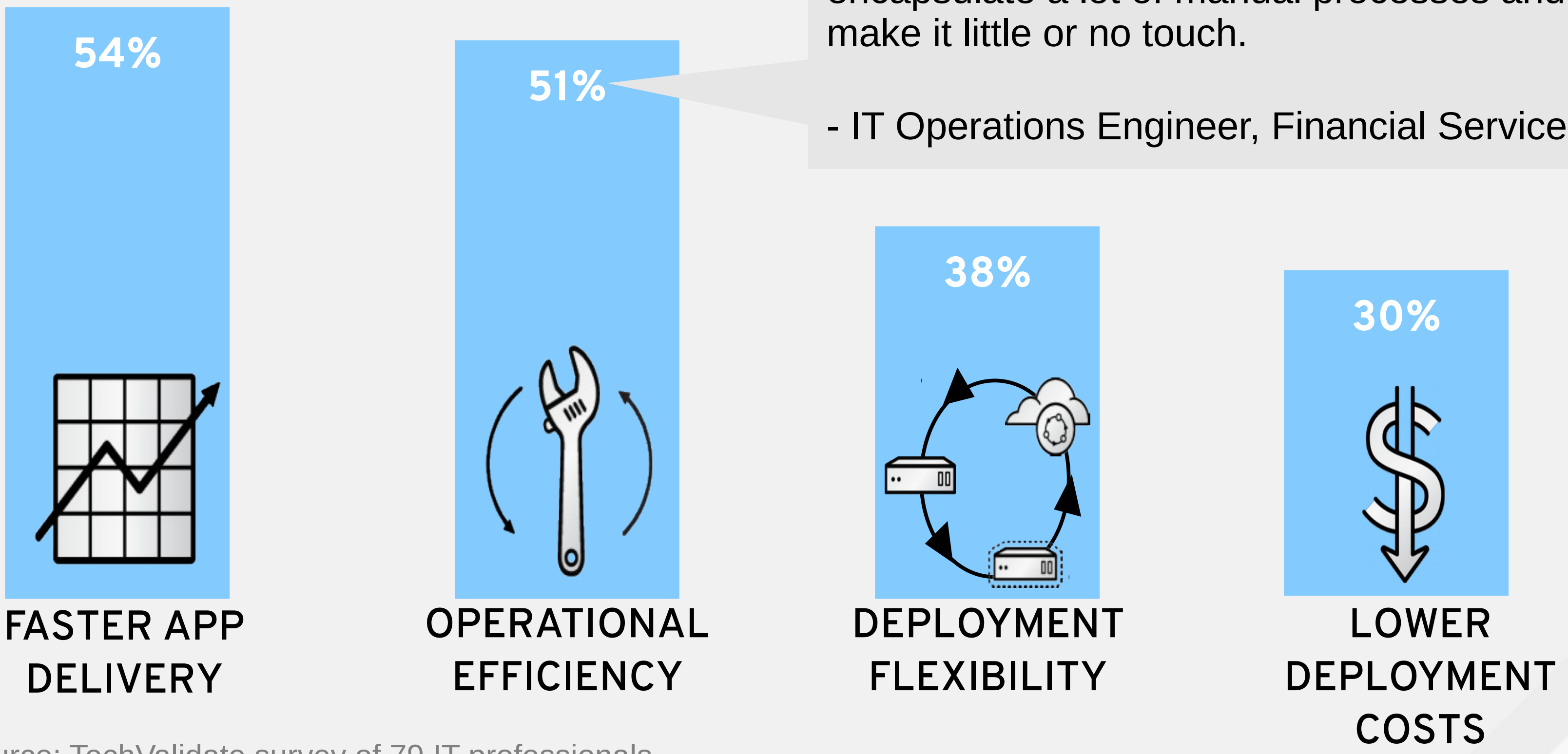
RED HAT® ENTERPRISE LINUX® 7



RED HAT® ENTERPRISE LINUX® ATOMIC HOST

Top benefits

MANY SEE CONTAINERS AS THE UTOPIA OF APPLICATION DELIVERY



Containers potentially offer the ability to encapsulate a lot of manual processes and make it little or no touch.

- IT Operations Engineer, Financial Services

Source: TechValidate survey of 79 IT professionals

It is nothing new ...

SOME OF THE MOST ADVANCED INFRASTRUCTURES RUN ON CONTAINERS

Let Me Contain That For You

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

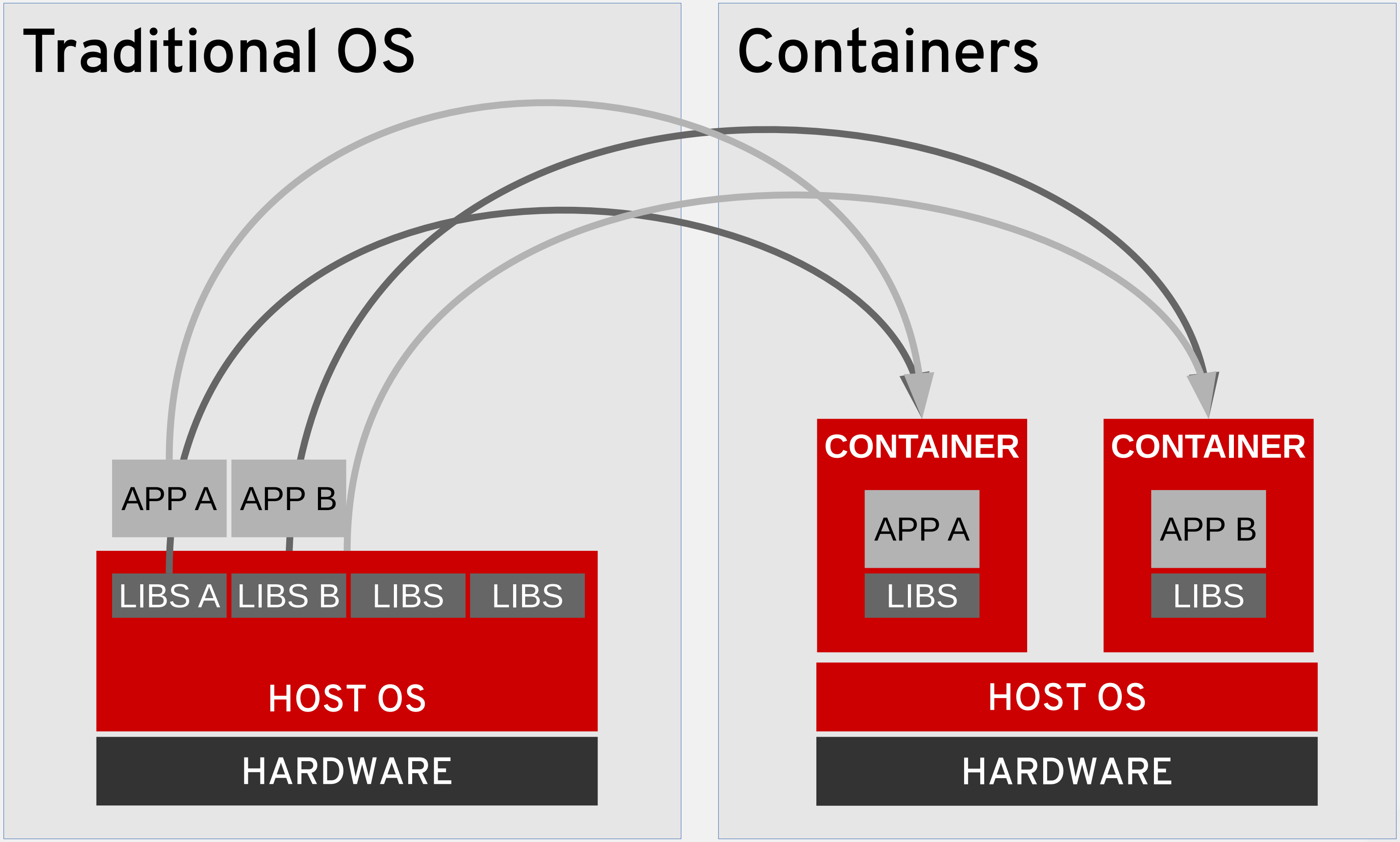
- Eric Brewer, VP of Infrastructure, Google

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

Top 5 misconceptions about containers

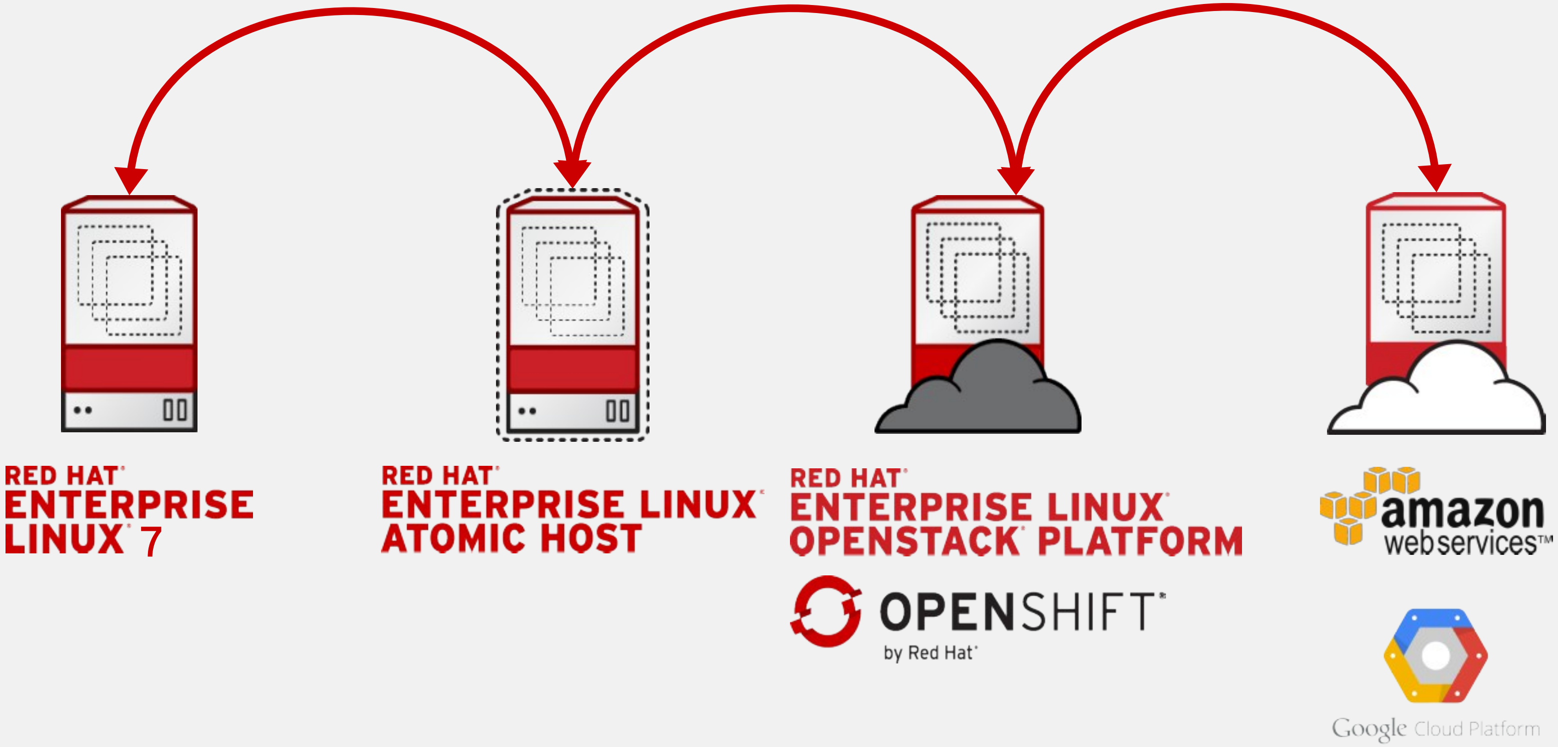
1. Containers are new
2. Containers equal virtualization
3. Containers are universally portable
4. Containers are secure by default
5. Containers are not enterprise-ready

Traditional vs Containers



Container portability

Across physical, virtual, private cloud, public cloud

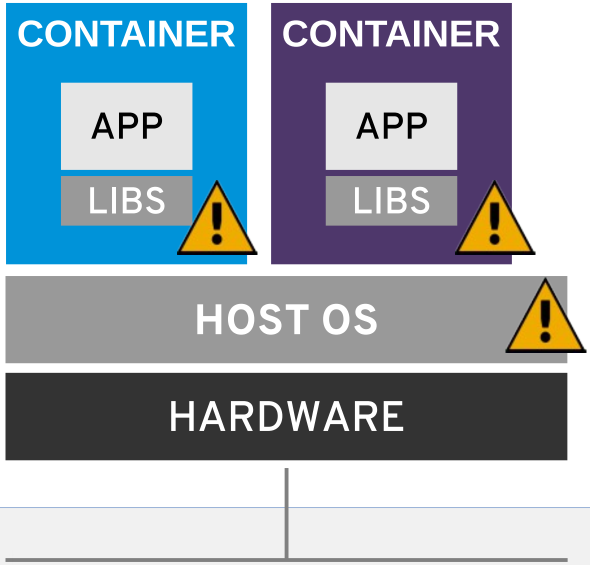

RED HAT ENTERPRISE LINUX 7
RED HAT ENTERPRISE LINUX ATOMIC HOST
RED HAT ENTERPRISE LINUX OPENSTACK PLATFORM
OPENSHIFT by Red Hat
amazon webservices™
Google Cloud Platform

Securing hosts and containers

RED HAT CONTAINER CERTIFICATION

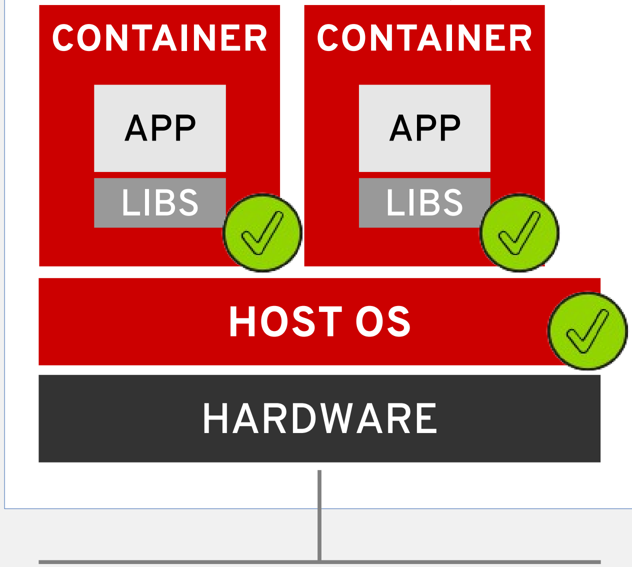
UNTRUSTED

- 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?

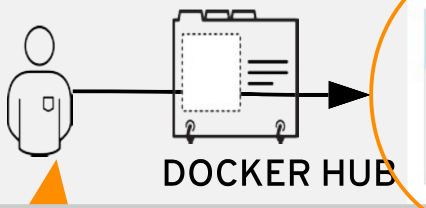


CERTIFIED

- Trusted source for the host and the containers
- Enterprise life cycle for content
- Proven portability
- Container Development Kit

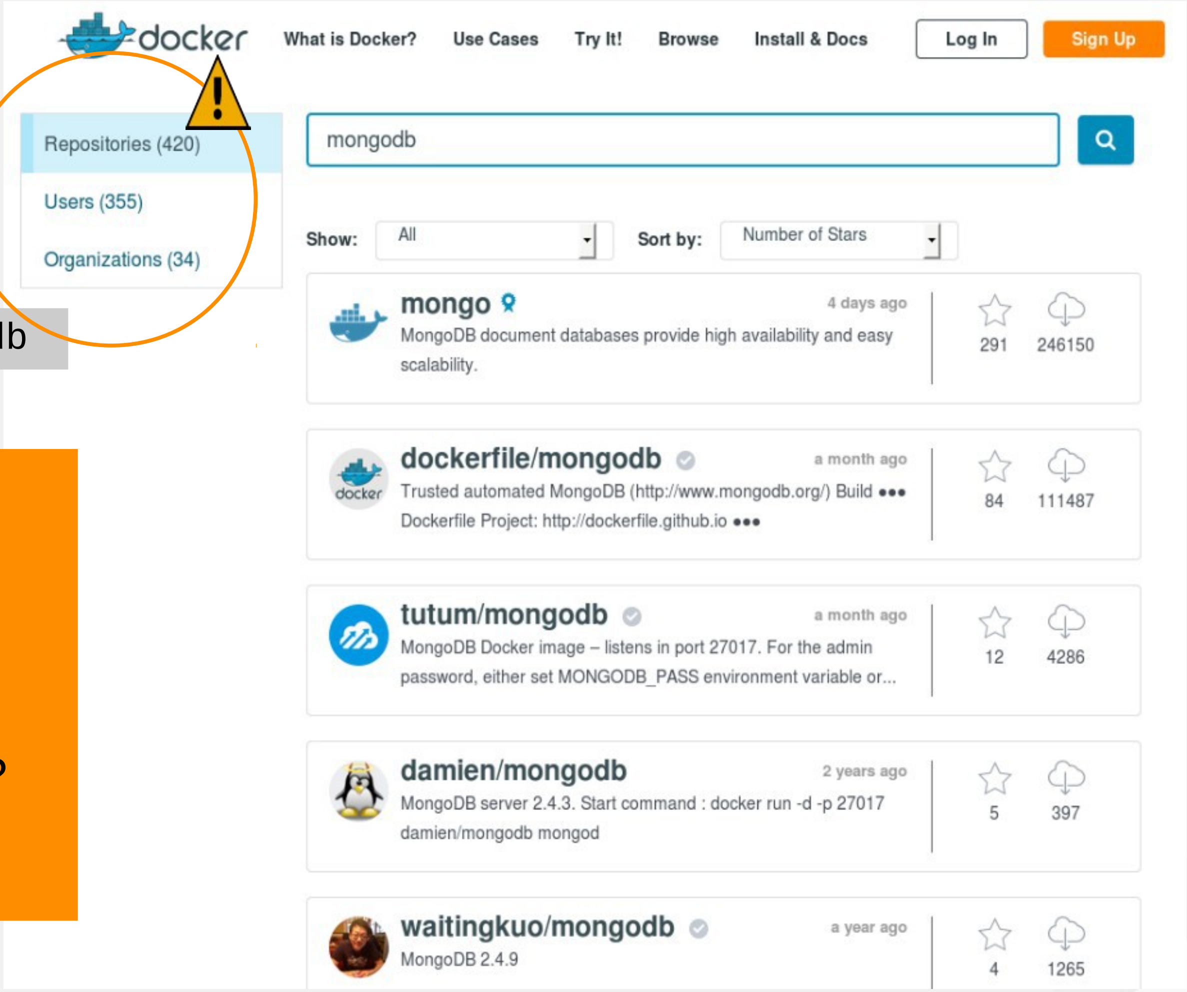


Example: Consuming MongoDB



```
docker pull mongodb
```

- Who built this image?
- What's its purpose? Was it created to support a demo?
- Is it safe to consume?
- Who maintains it?



Repositories (420)

Users (355)

Organizations (34)

mongodb

Show: All

Sort by: Number of Stars

mongo 4 days ago
MongoDB document databases provide high availability and easy scalability.
291 246150

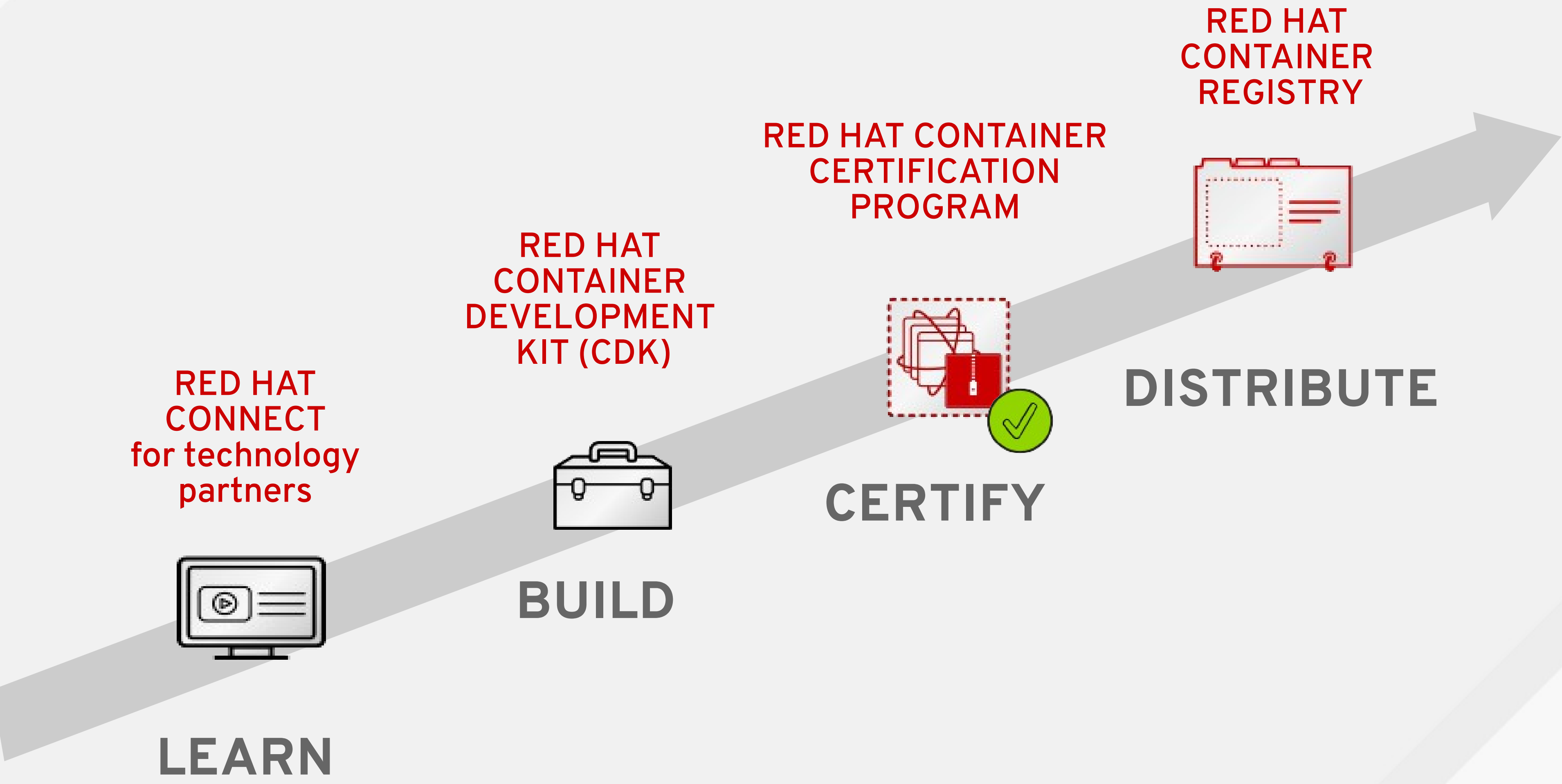
dockerfile/mongodb a month ago
Trusted automated MongoDB (http://www.mongodb.org/) Build ••• Dockerfile Project: http://dockerfile.github.io •••
84 111487

tutum/mongodb a month ago
MongoDB Docker image – listens in port 27017. For the admin password, either set MONGODB_PASS environment variable or...
12 4286

damien/mongodb 2 years ago
MongoDB server 2.4.3. Start command : docker run -d -p 27017 damien/mongodb mongod
5 397

waitingkuo/mongodb a year ago
MongoDB 2.4.9
4 1265

Simplifying container adoption for partners and customers



RED HAT CONNECT for technology partners

LEARN

RED HAT CONTAINER DEVELOPMENT KIT (CDK)

BUILD

RED HAT CONTAINER CERTIFICATION PROGRAM

CERTIFY

RED HAT CONTAINER REGISTRY

DISTRIBUTE

Establishing standards around…

Red Hat works with the open source community to drive standards for containerization.

ISOLATION LINUX CONTAINERS

- Isolating applications on host operating system
- Security with SELinux
- Portability across host systems

CONTAINER FORMAT DOCKER / RUNC

- Interface for communications, configuration, data persistence, provisioning
- Content agnostic
- Infrastructure agnostic

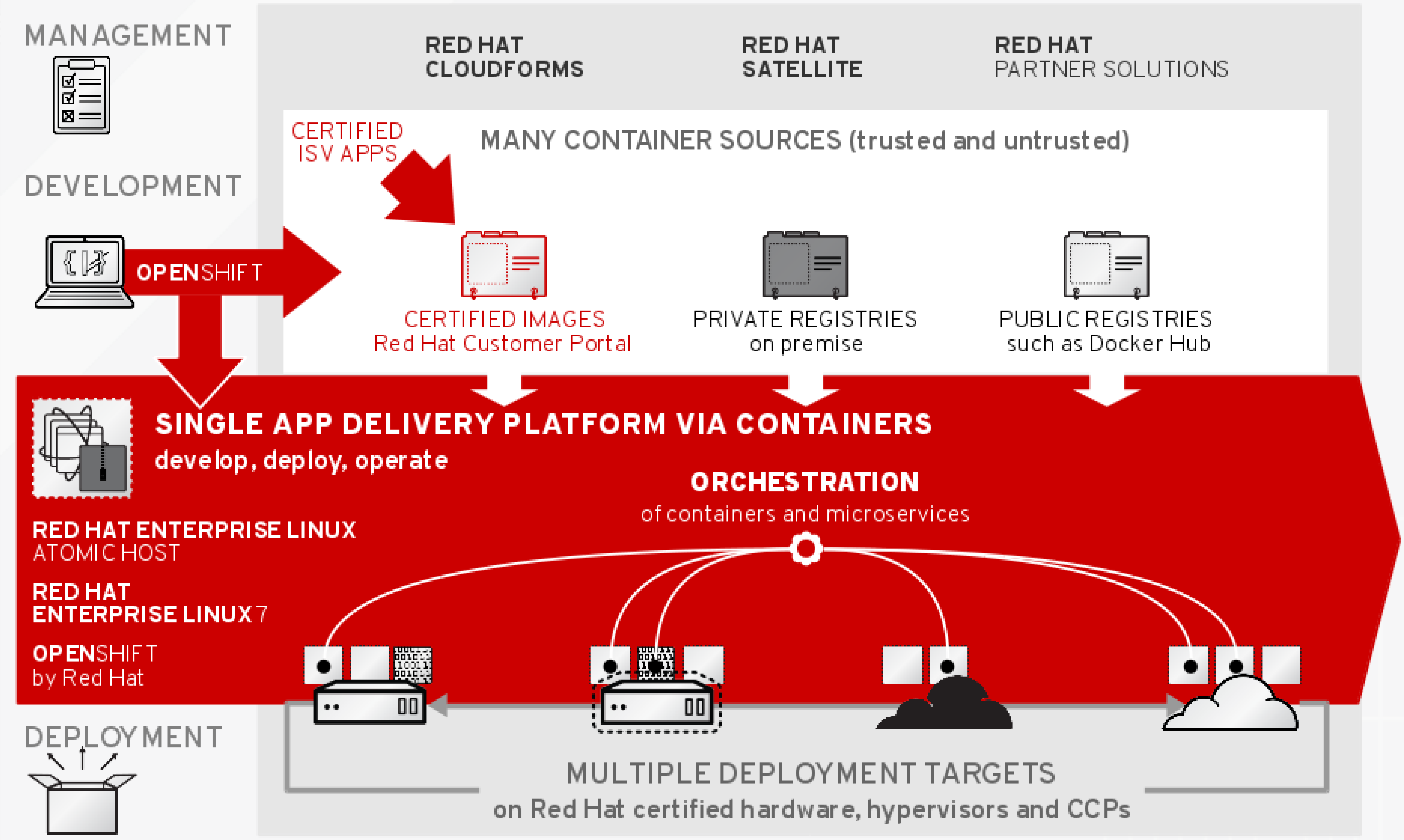
ORCHESTRATION KUBERNETES

- Orchestrate at scale
- Define application topologies
- Handle container networking
- Manage state
- Schedule across hosts

REGISTRY / CONTAINER DISCOVERY

- Easily find and consume trusted container images
- Federate consumption libraries
- Promote consistency and reuse

More than the container ...



MANAGEMENT

DEVELOPMENT

DEPLOYMENT

RED HAT CLOUDFORMS

RED HAT SATELLITE

RED HAT PARTNER SOLUTIONS

CERTIFIED ISV APPS

MANY CONTAINER SOURCES (trusted and untrusted)

OPENSHIFT

CERTIFIED IMAGES
Red Hat Customer Portal

PRIVATE REGISTRIES
on premise

PUBLIC REGISTRIES
such as Docker Hub

SINGLE APP DELIVERY PLATFORM VIA CONTAINERS
develop, deploy, operate

ORCHESTRATION
of containers and microservices

RED HAT ENTERPRISE LINUX
ATOMIC HOST

RED HAT
ENTERPRISE LINUX 7

OPENSHIFT
by Red Hat

MULTIPLE DEPLOYMENT TARGETS
on Red Hat certified hardware, hypervisors and CCPs

Technologies

Linux containers technology base

ISOLATION WITH LINUX CONTAINERS

ORCHESTRATION KUBERNETES

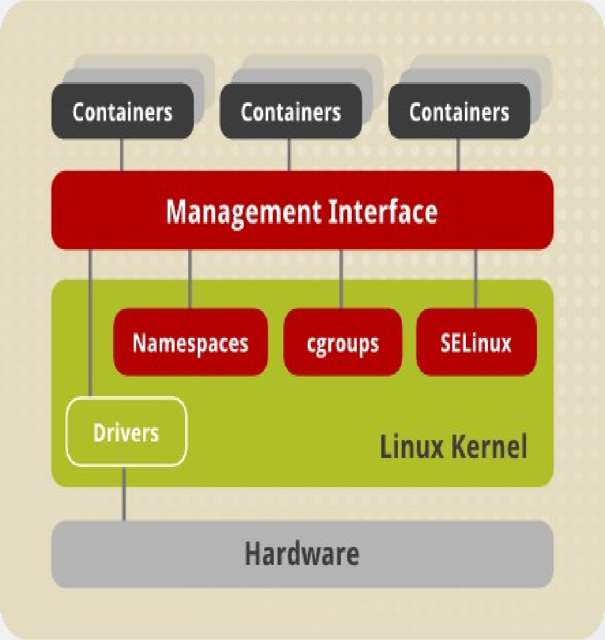
CONTAINER FORMAT DOCKER / RUNC

RPM-OSTREE FOR ATOMIC

What are linux Containers?

Operating System Level lightweight encapsulation environment for running multiple isolated Linux systems in a single kernel instance.

- SELinux, sVirt for separation
- Control Groups (cgroups) for resource management
- kernel namespaces process isolation
  - Network, pid, mounts, ipc, uts
- Docker automates containers and image management



Linux containers technology base

ISOLATION WITH LINUX CONTAINERS

ORCHESTRATION KUBERNETES

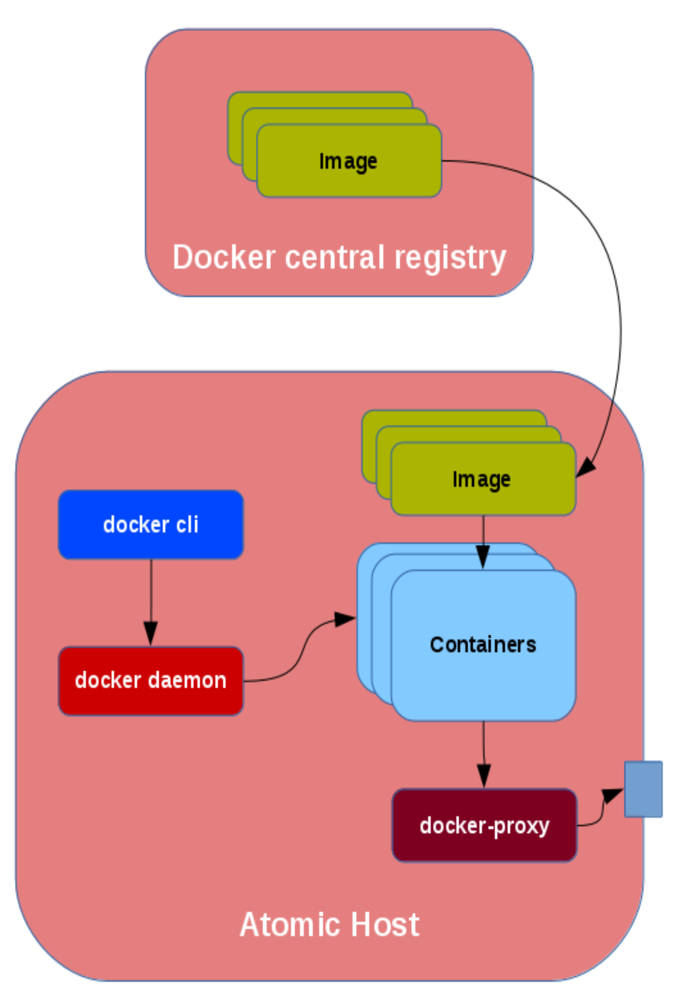
CONTAINER FORMAT DOCKER / RUNC

RPM-OSTREE FOR ATOMIC

Docker components

Service to containerize applications:

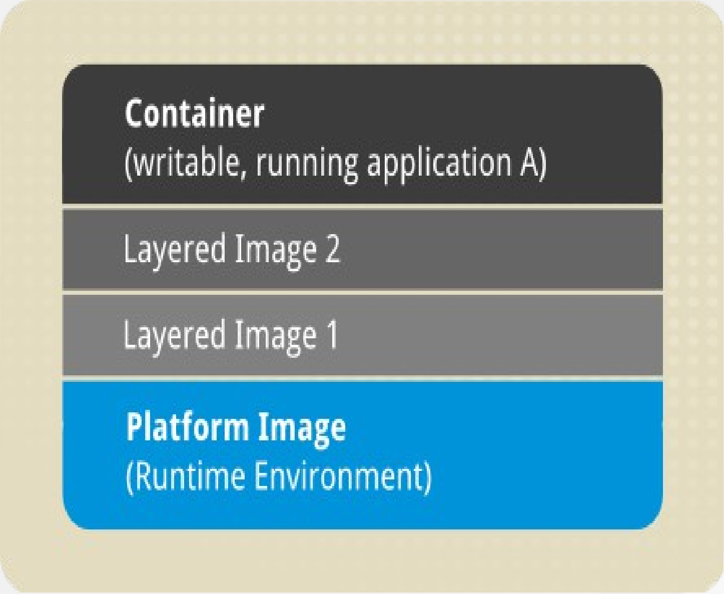
- Docker service: manages containers locally
- Docker cli: interacts with docker daemon
- Docker registry: storage and access images locally and centrally
- Docker-proxy: provides conectivity to containers via port mapping



Docker images

An application is packaged with all it's dependencies and run time and configuration in a docker image following a layered approach.

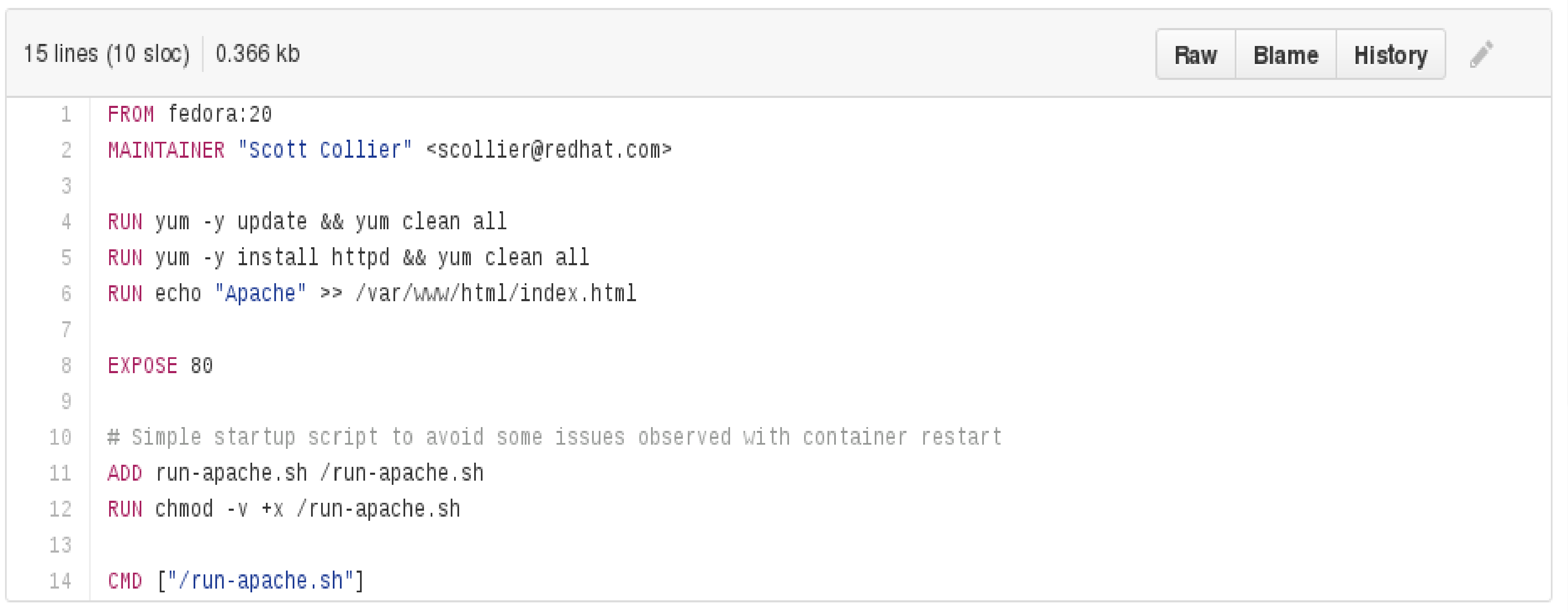
- Base or platform image
- Layered images with dependencies
- Finally a writable layer exist with each running container
- Image metadata as entrypoint, environment, author, ports, etc..



Creating images

Two different methods

- Modify on running container and commit
- Dockerfile (recommended method)

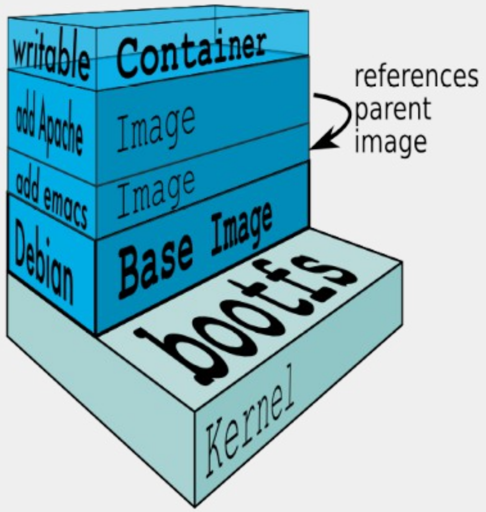


```
FROM fedora:20
MAINTAINER "Scott Collier" <scollier@redhat.com>

RUN yum -y update && yum clean all
RUN yum -y install httpd && yum clean all
RUN echo "Apache" >> /var/www/html/index.html

EXPOSE 80

# Simple startup script to avoid some issues observed with container restart
ADD run-apache.sh /run-apache.sh
RUN chmod -v +x /run-apache.sh

CMD ["/run-apache.sh"]
```

Storage in docker containers (I)

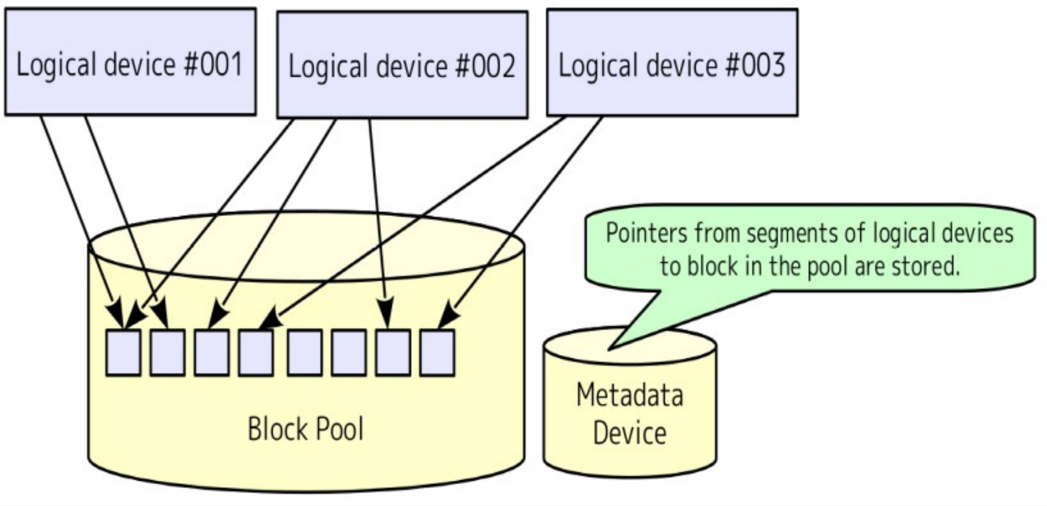
ROOTFS device in docker containers:

- Each container has a rootfs device mounted as root in containers.
- In RHEL atomic we use device mapper thin provisioning to implement CoW.
- Both base or platform and images layers are read only
- Only container layer is writable
- Each line in a dockerfile creates a new layer



Storage in docker containers (II)

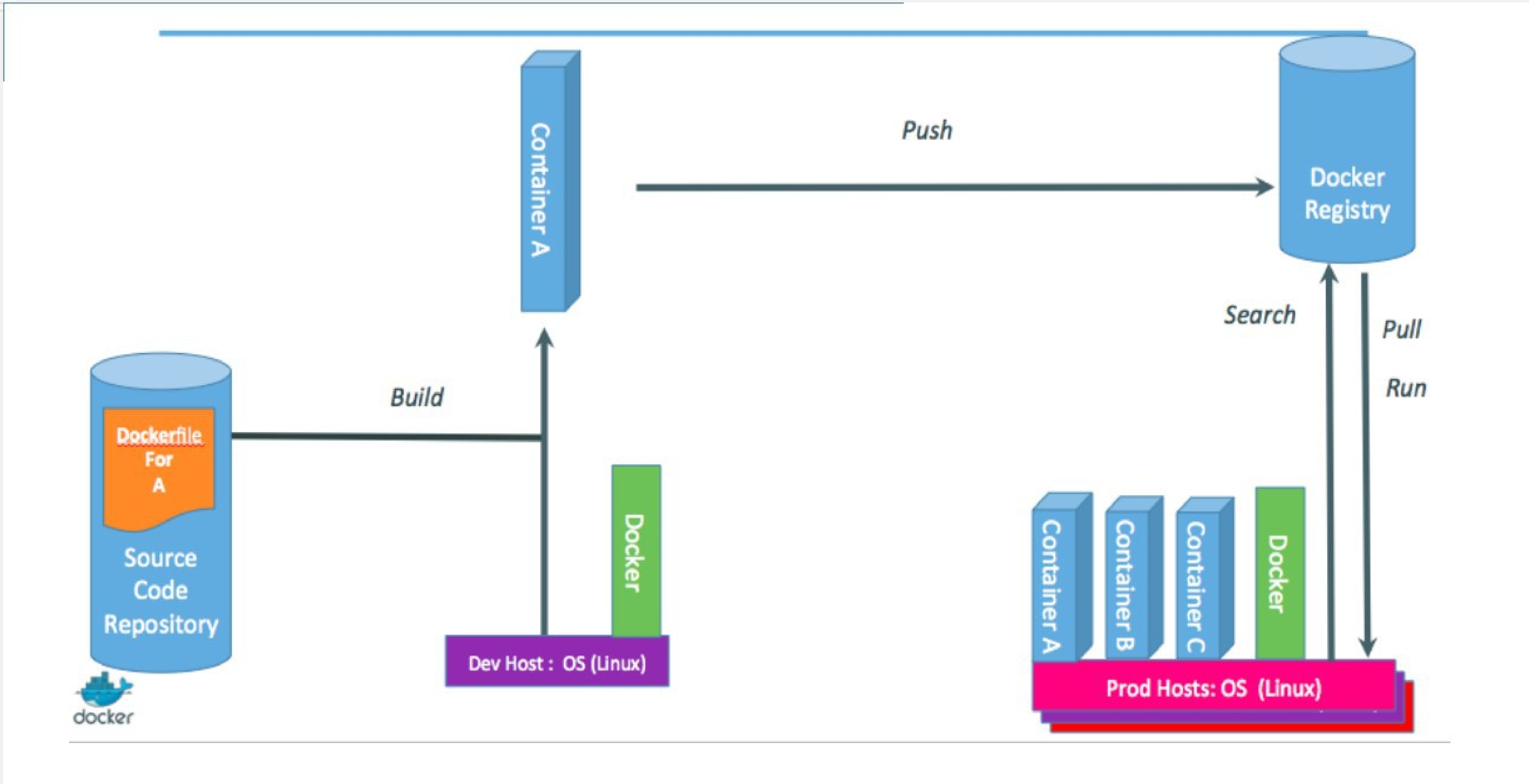
- Fixed size blocks are dynamically allocated to logical devices so that blocks are consumed only when data are actually written.
- Pointers from segments of logical devices to blocks in the block pool are stored in the metadatadevice.
- CoW (Copy on Write) snapshots are created by allowing pointing to the same block from different logical devices.



Creating images: best practices

- Containers should be ephemeral
- Reduce the packages and footprint of containers
- Run one process per container
- Balance number of layers
- Check dockerfile options and examples for inspiration
- Some people propose the use of puppet to build images (could be convenient for complex images, overkilling in most cases)

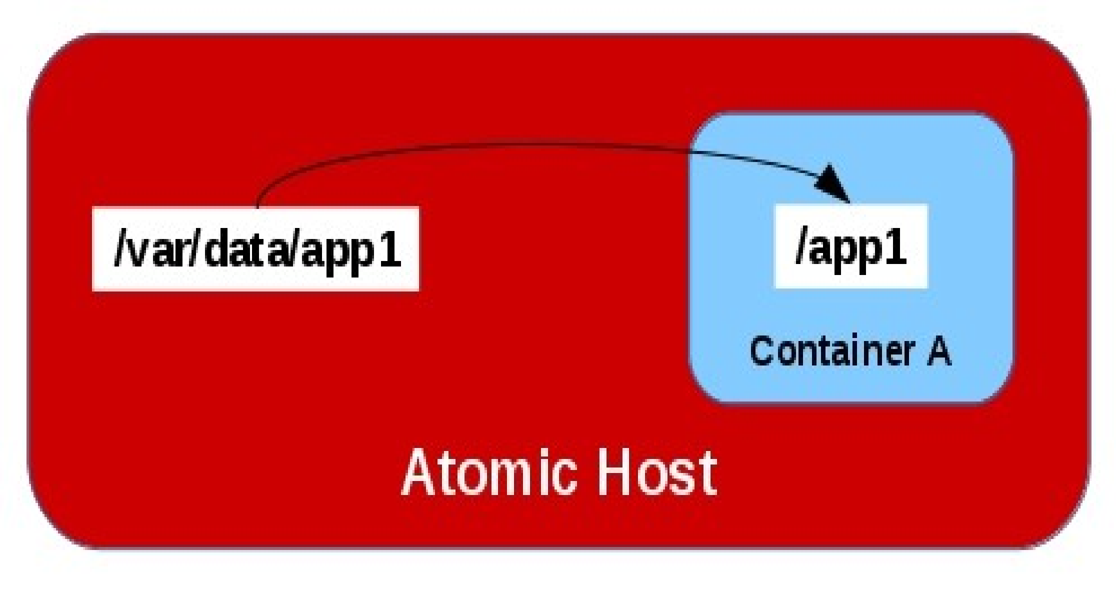
Store images: docker registry



External storage in docker containers

Containers can access external storage, volumes:

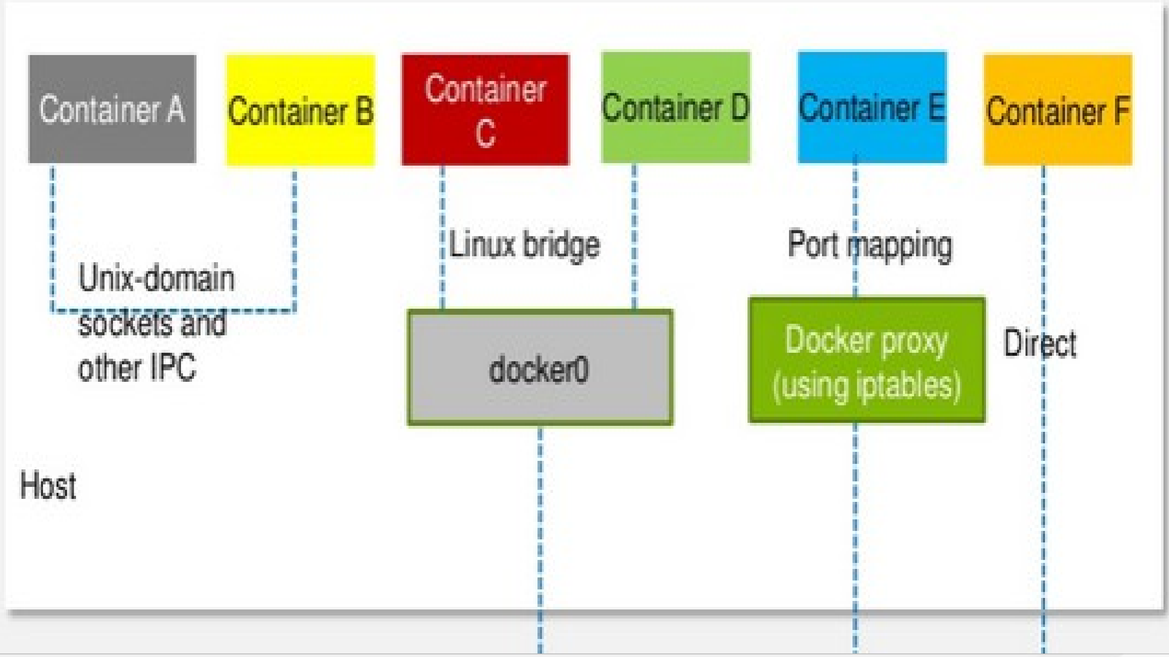
- Containers can use directories in atomic hosts as bind mounts.
- Recommended option for stateful containers.



Networking in linux containers

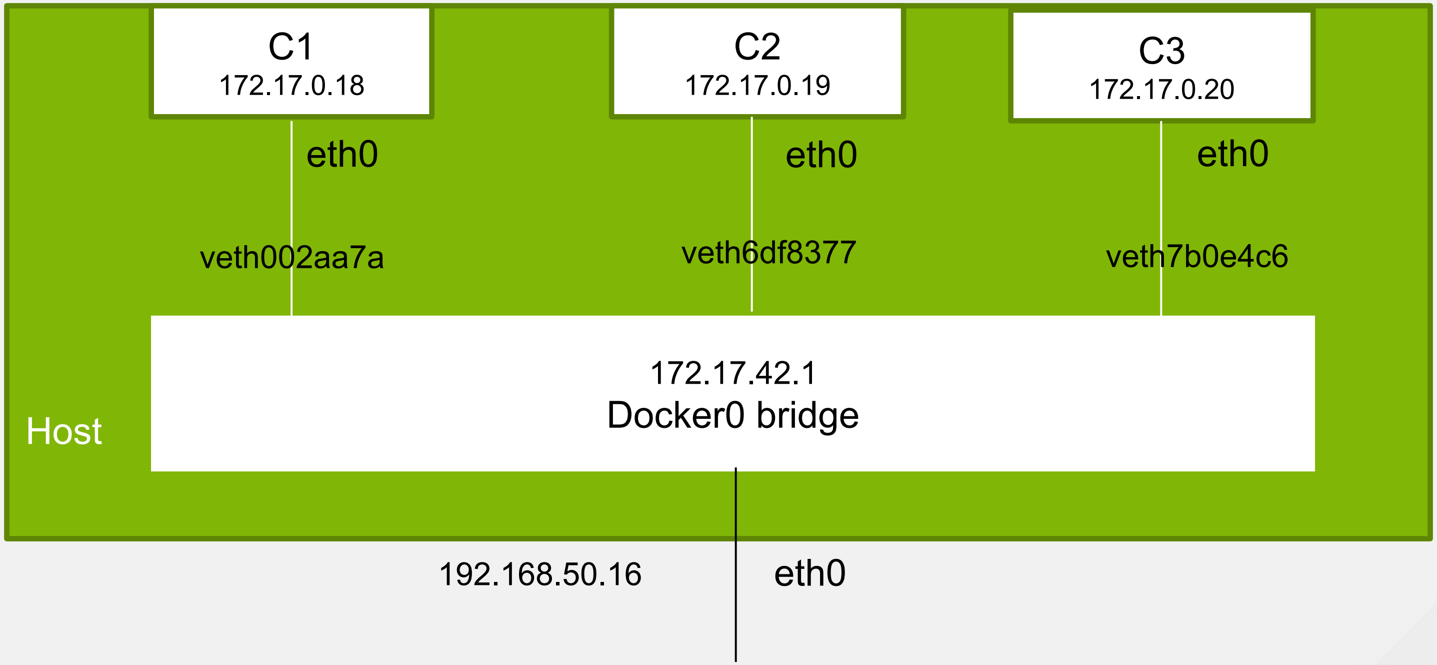
Different networking options exist depending on the use case

- Local linux bridge
- Host networking
- Port Mapping
- Via SDN



Docker proxy

- When a new container is created, it's connected to a linux bridge and assigned a static IP
- This only provides access from the host
- Docker provides access to the ports in the container using iptables DNAT rules and listener docker-proxy



Privileged Containers (I)

What if?

- Containers need access to the host.
- Containers to manage other containers.
- Containers require non-standard capabilities

Why?

- Containerize monitoring/administration applications
- Enable kernel modules
- Using docker to distribute and manage software for JEOS docker hosts

Privileged Containers (II)

Disabling isolation:

- Disable Selinux separation
- Enabling all linux capabilities
- Allow the creation of all linux devices

Turn off namespace separation:

- All but mount namespaces can be disabled

Mount hosts filesystems in the container:

- Mounting /run -> /run allow access to dbus or systemd
- Mounting / -> /host (or sysroot) to provide access to host filesystems

Privileged Containers (III)

Running Super Privileged Containers:

- Using docker command

```
# docker run -it --name rhel-tools --privileged --ipc=host --net=host --pid=host -e
HOST=/host -e NAME=rhel-tools -e IMAGE=[REGISTRY]/rhel7/rhel-tools -v
/run:/run -v /var/log:/var/log -v /etc/localtime:/etc/localtime -v /:/host
[REGISTRY]/rhel7/rhel-tools
```

- Using atomic command

```
# atomic run --spc rhel7/rhel-tools bash
```

Linux containers technology base

ISOLATION WITH LINUX CONTAINERS

ORCHESTRATION KUBERNETES

CONTAINER FORMAT DOCKER / RUNC

RPM-OSTREE FOR ATOMIC

Managing Containers

Wait, I have a great Script to manage them …

via …

@kelseyhightower

Managing Containers

… and it scales!!!

via …

@kelseyhightower

What is Kubernetes

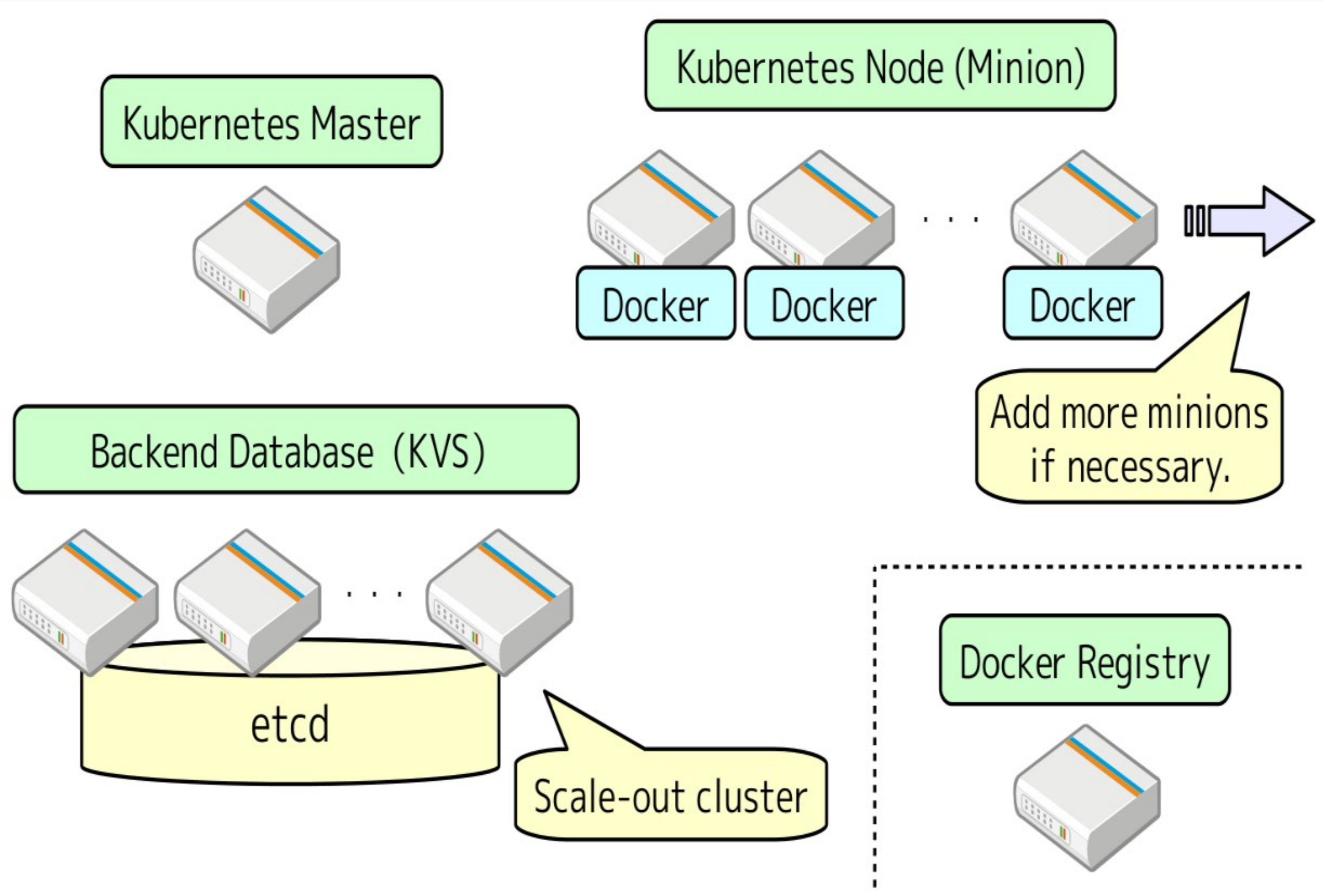
- A highly collaborative open source project originally conceived by Google
  - Google has almost 10 years experience with containerized apps
  - Red Hat has been member since day 0
  - Red Hat is second largest contributing member (after Google)
- A declarative language for managing containers
- Start, stop, update and manage a cluster of machines running containers in a consistent and maintainable way
- Also known as kube or k8s

What is Kubernetes

- Kubernetes is a container cluster manager
- Manages containerized applications in a clustered environment
- Particularly suited for horizontally scaleable, stateless, or “microservices” application architectures
  - Does not mean others will not work or are ignored
- 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

NSLU 430692 3
42G1

Cluster components



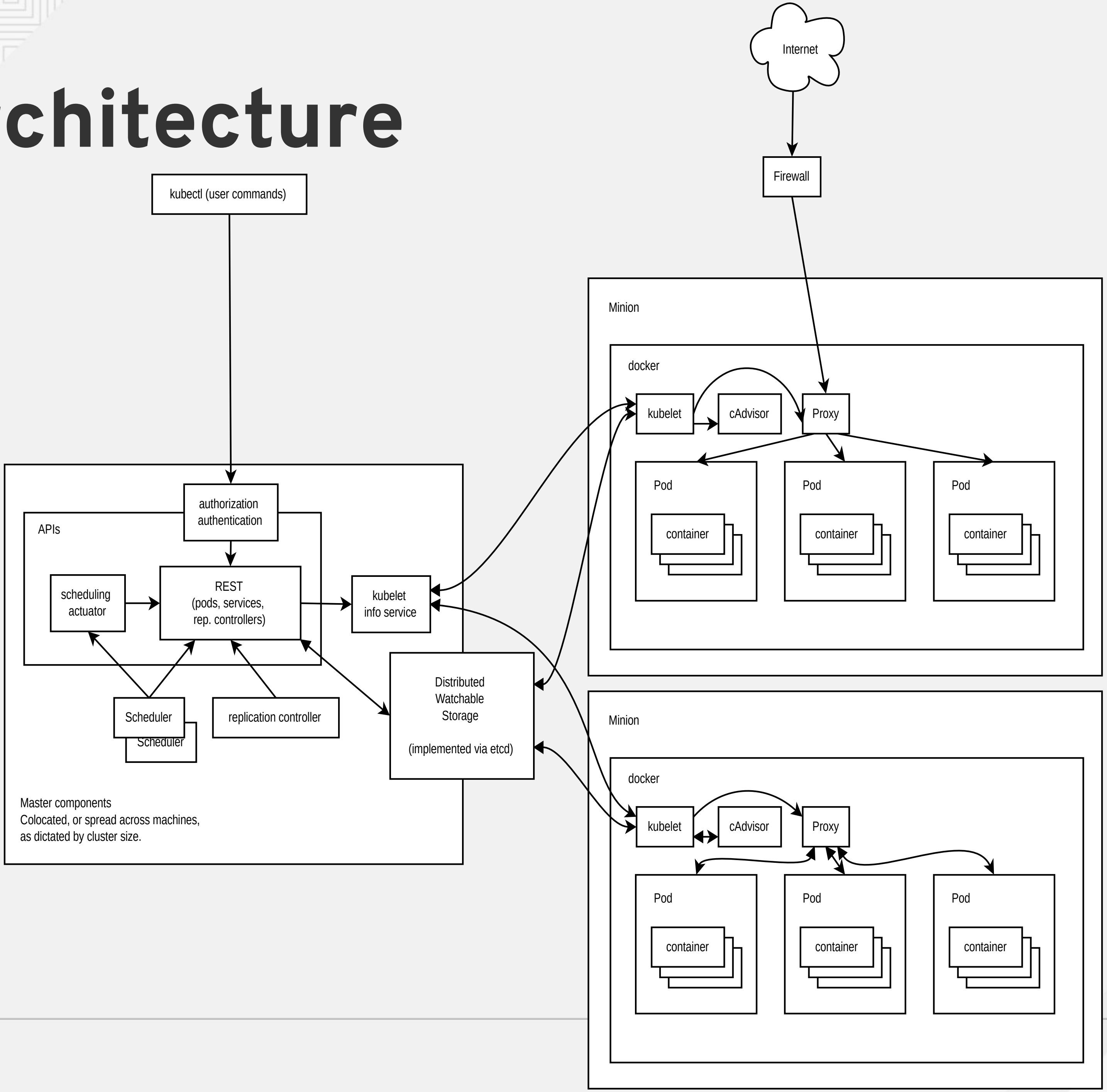
Master

- Typically consists of:
  - Kube-apiserver: manage api calls
  - Kube-scheduler: assign nodes where run are launched
  - Kube-controller-manager: manage replication controllers
  - Etcd: backend database
- Might contain:
  - Kube-proxy: manage network access to services
  - A network management utility

Node/Minion

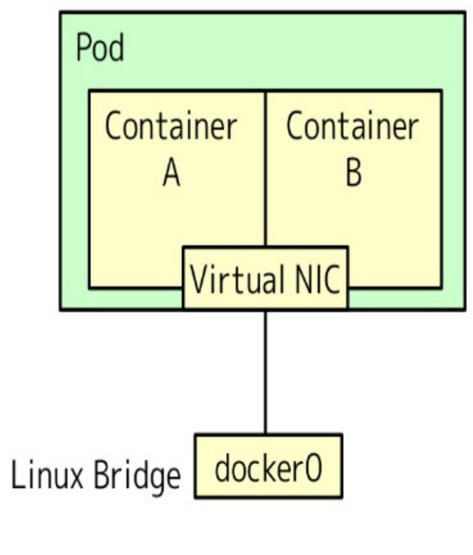
- Typically consists of:
  - Kubelet: manage pods and containers running in the node
  - Kube-proxy: manage network access to services
  - Cadvisor: monitor resources usage by containers
- Might contain:
  - A network management utility

Architecture



Pod

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

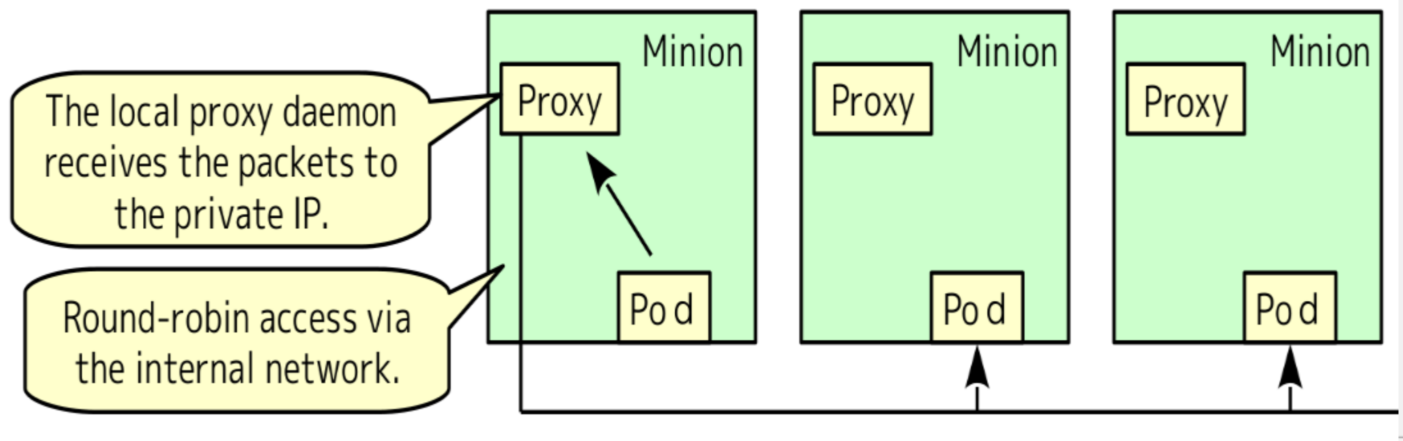


Replication Controller

- 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. If too many copies are running, the replication controller will dispose of the exceeding pods.
- The number of pods can be dynamically changed
- Able to perform rolling updates of controllers pod by pod

Services

- Every pod/replication controller will need a service. What's the point of a pod that doesn't provide som sort of service/useful work?
- How “stuff” finds pods which could be anywhere?
- Define:
  - What port in the container
  - Labels on pods which would respond to this type of request


The local proxy daemon receives the packets to the private IP.
Round-robin access via the internal network.
Minion
Proxy
Pod
Minion
Proxy
Pod
Minion
Proxy
Pod

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

Namespace

- Attached to every object
- Pods in ns1 will not get service variable from ns2
- Users with permission to CRUD (create, read, update, delete) objects in ns1 may not have permsissions to CRUD objects in ns2
- The network is not segregated
- Some people consider using a namespace per application. Some say a namespace per team.

Storage

- Containers are launched clean from the hub
  - Stateful data is hard
- Mount the same NFS mount on every node in the same place
  - Expose that location into your pods
- Stateful data is easy!

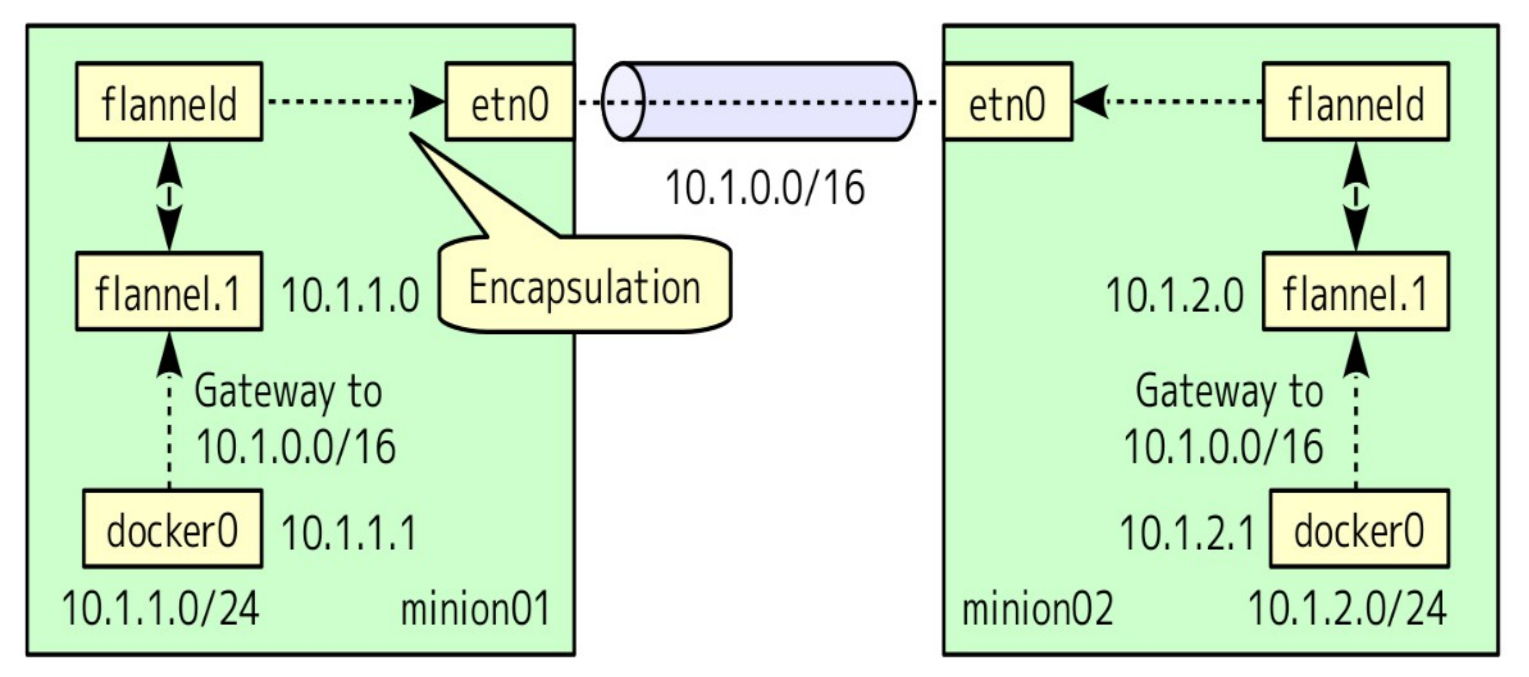
Networking setup

- 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
- Docker looks so easy
  - 2 containers on one host can easily talk
  - How to get to those 2 containers from outside?
  - How to get to from one container on one host to a container on another?
- Networking is really hard!

Networking setup

- Flannel (available in Fedora, Centos, RHEL and RHEL Atomic Host)
  - Super 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 setup



Linux containers technology base

ISOLATION WITH LINUX CONTAINERS

ORCHESTRATION KUBERNETES

CONTAINER FORMAT DOCKER / RUNC

RPM-OSTREE FOR ATOMIC

Red Hat Enterprise Linux Atomic Host

IT IS RED HAT ENTERPRISE LINUX

Inherits the complete hardware ecosystem, military-grade security, stability and reliability for which Red Hat Enterprise Linux is known.

OPTIMIZED FOR CONTAINERS

MINIMIZED FOOTPRINT

Minimized host environment tuned for running Linux containers while maintaining compatibility with Red Hat Enterprise Linux.

SIMPLIFIED MAINTENANCE

Atomic updating and rollback means it’s easy to deploy, update, and rollback using imaged-based technology.

ORCHESTRATION AT SCALE

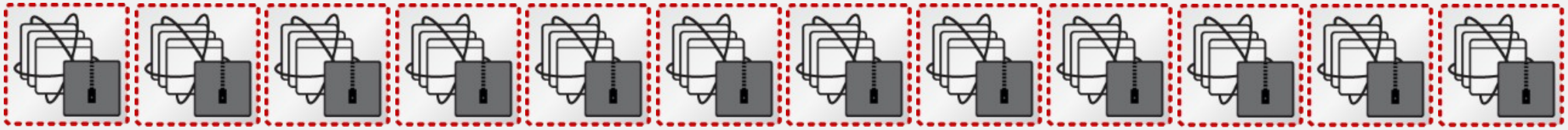
Build composite applications by orchestrating multiple containers as microservices on a single host instance.

RHEL Atomic Host Philosophy

RHEL Atomic Host minimal footprint container host

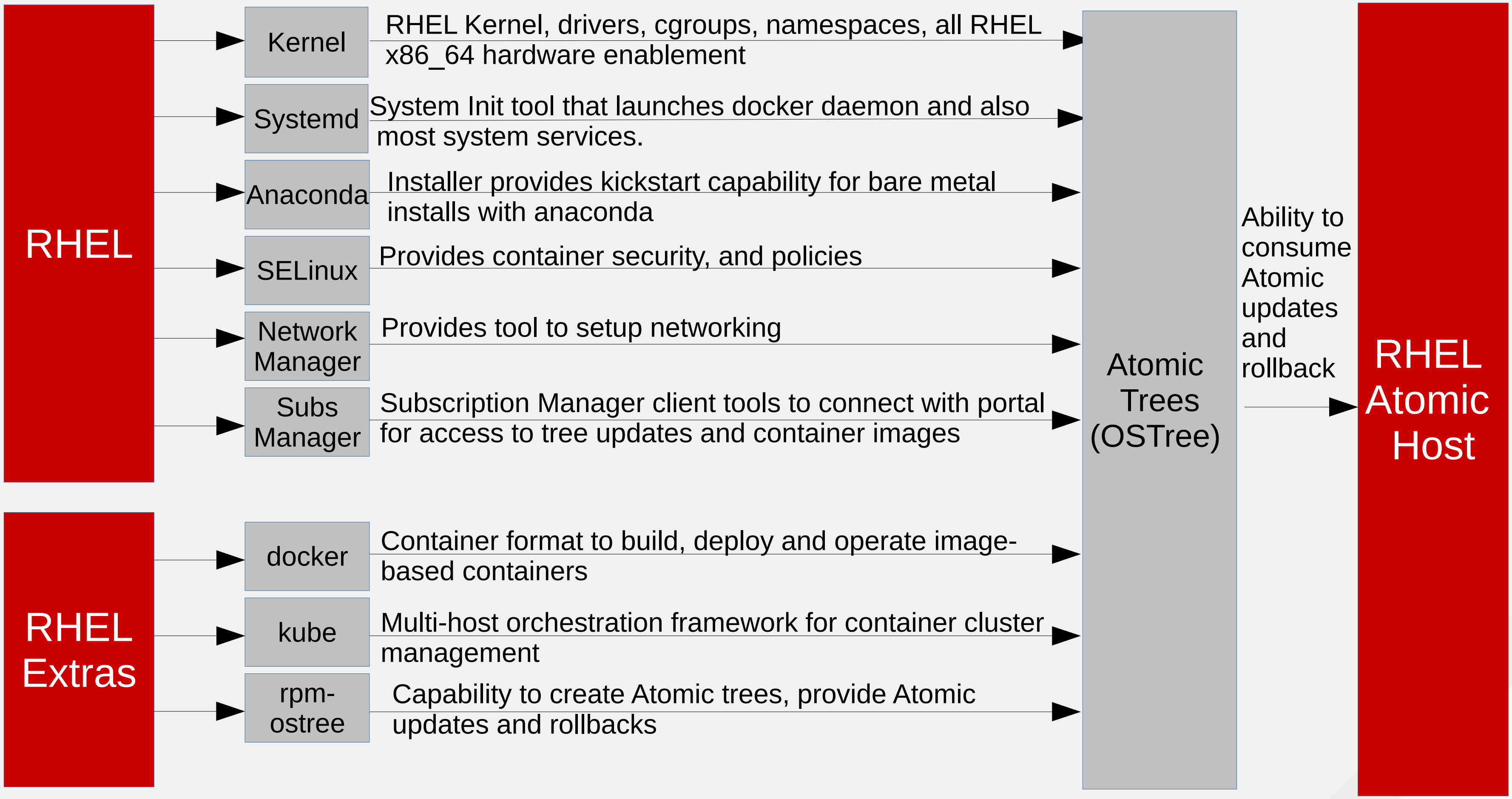
New package requests and feature enhancements added in an atomic container image (SPCs)

Packages that have been proven to not function in a container, would be considered for the host



RED HAT
ENTERPRISE LINUX
ATOMIC HOST

RHEL Atomic Host Inheritance model



RHEL

Kernel — RHEL Kernel, drivers, cgroups, namespaces, all RHEL x86_64 hardware enablement

Systemd — System Init tool that launches docker daemon and also most system services.

Anaconda — Installer provides kickstart capability for bare metal installs with anaconda

SELinux — Provides container security, and policies

Network Manager — Provides tool to setup networking

Subs Manager — Subscription Manager client tools to connect with portal for access to tree updates and container images

RHEL Extras

docker — Container format to build, deploy and operate image-based containers

kube — Multi-host orchestration framework for container cluster management

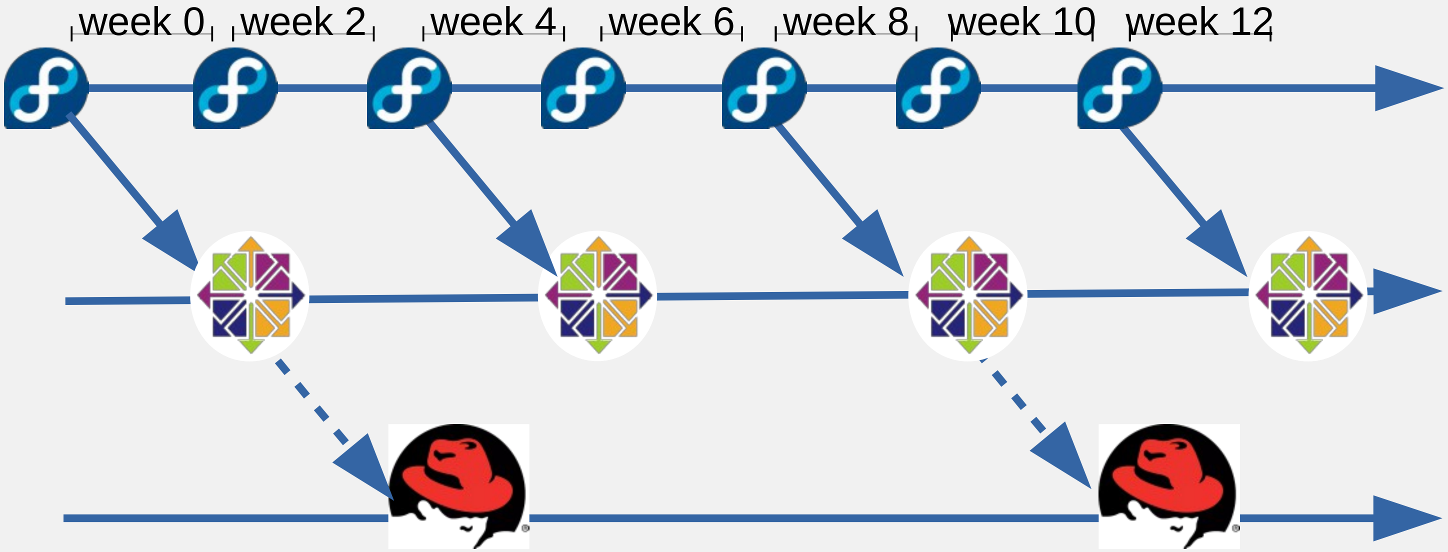
rpm-ostree — Capability to create Atomic trees, provide Atomic updates and rollbacks

Atomic Trees (OSTree)

Ability to consume Atomic updates and rollback

RHEL Atomic Host

RHEL Atomic Host Update Cadence



Utilize upstream projects to provide sneak preview into the Red Hat development for early adopters:

- Fedora: **2 weeks**
- CentOS: **4 weeks**
- RHEL Atomic Host: **6 weeks**

Note: CentOS is a community vehicle to drive innovation. Atomic project will use CentOS to land technologies to gather community feedback

RHEL Atomic Host Deployment

PHYSICAL SYSTEM ISO

- Anaconda installation
- Kickstart installation
- PXEboot installation

CERTIFIED HYPERVISORS

- RHEL (Qcow2 for KVM based HV)
- RHEV (OVA import for KVM)
- VMware (OVA)
- Microsoft (VHD for Hyper-V HV)

PRIVATE CLOUDS

- RHEL Open StackPlatform (Qcow2 for KVM based HV)

PUBLIC CLOUDS

- Images available on select public clouds via cloud access
- Amazon Web Services (AWS)
- Google Compute Engine (GCE)

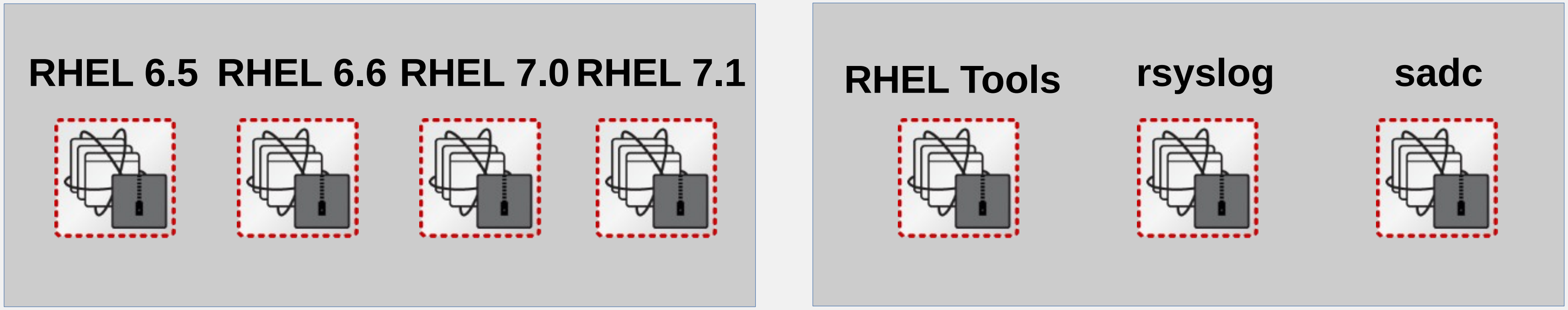
- RHEL-Atomic-Installer-7.1.0.x86_64.iso

- Rhel-atomic-cloud-7.1-0.x86_64.qcow2
- Rhel-atomic-cloud-7.1-0.x86_64.rhevm.ova
- Rhel-atomic-cloud-7.1-0.x86_64.vsphere.ova
- Rhel-atomic-cloud-7.1-0.x86_64.vhd

- AMAZON AWS AMI IMAGES

RHEL Atomic Host Container Images

Customer Portal



RHEL 6.5 RHEL 6.6 RHEL 7.0 RHEL 7.1

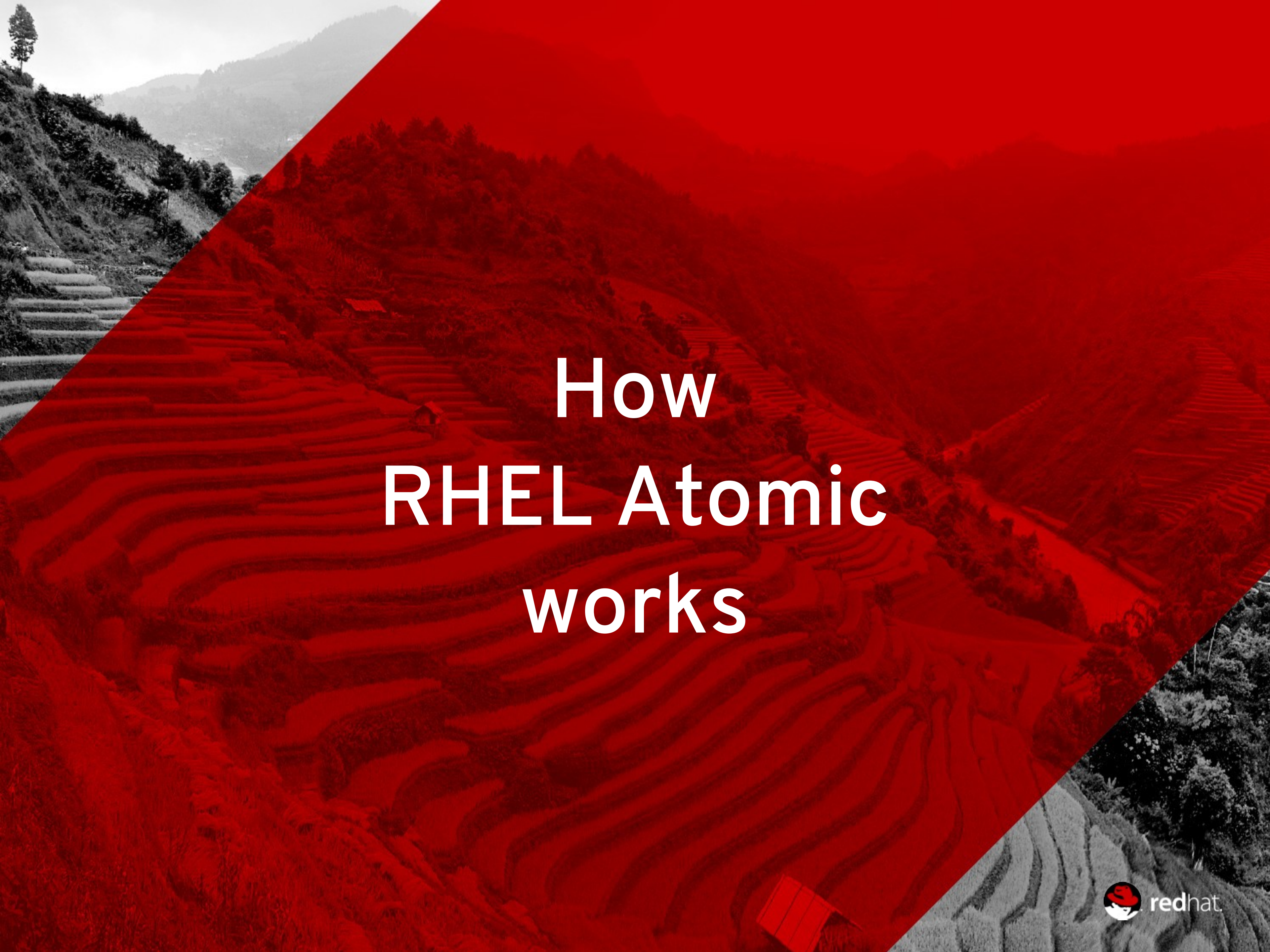
RHEL Tools rsyslog sadc

CONTAINER IMAGES

(RHEL 7 and RHEL Atomic Host 7)

ATOMIC CONTAINER IMAGES

RED HAT® ENTERPRISE LINUX® ATOMIC HOST



How RHEL Atomic works

Minimization vs functionality

- No man pages, en_US only
- Some extra files stripped (desktop backgrounds, Radeon GPU driver)
- Includes: `cloud-init` (used on bare metal too for PXE-to-Live)
- includes: `kubernetes`, `etcd`, `flannel` (no separate downloads)

RPM+OSTree: Atomic host OS upgrades+rollback

- Goal: Hybrid image/package system
- Current focus: Server side "compose" of RPM packages, replication to clients
- Atomic (Control-C at any time) updates: swap immutable trees
- current system is never modified - reboot to update
- In return, very reliable rollback: also a tree swap
- Replication model supports minimization better than RPM/yum

Admin experience: 4 cmds

- subscription-manager register …
- atomic host status
- atomic host upgrade
- atomic host rollback

OSTree filesystem model

- / has the immutable bit set
- /usr is a read-only bind mount. Always.
- /etc is "rebased" on upgrades - apply config diff to new /etc
- /var is untouched
- /home -> /var/home

/usr/bin/atomic

- /usr/bin/atomic is a new app that speaks both Docker and rpm-ostree
- atomic host upgrade ⇨ rpm-ostree upgrade, etc.
- atomic run is a convenience enabler for super-privileged containers

Commands

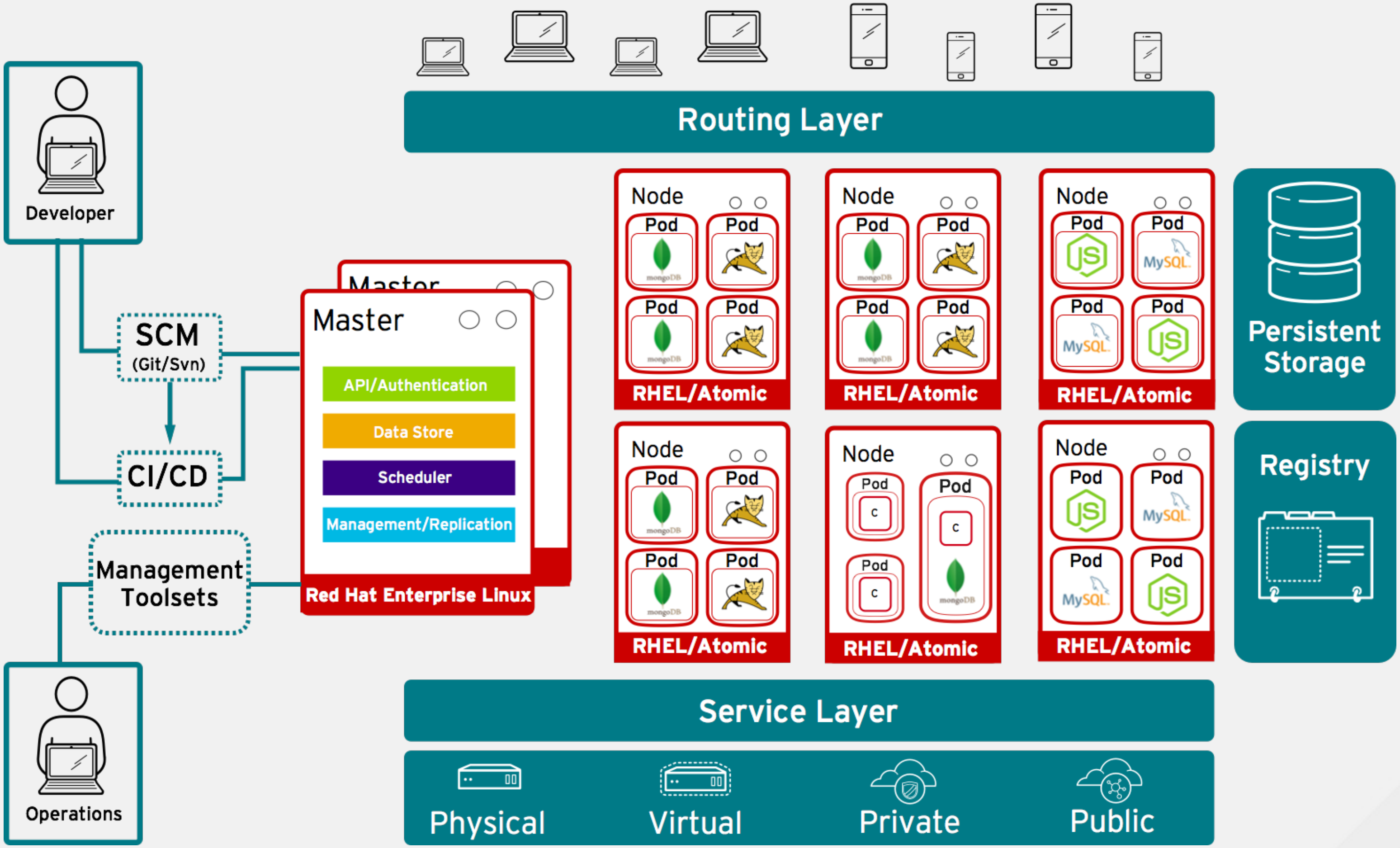
- atomic host status
- atomic host upgrade
- atomic host rollback
- atomic install <IMAGE>
- atomic run - - spc <IMAGE> <COMMAND>
- atomic uninstall <IMAGE>
- atomic update <IMAGE>
- atomic info <IMAGE>

OpenShift

What's missing?

- A nice GUI for the whole Platform
- A common API for all components
- A pre-configured SDN
- Decoupled devs and ops
- Integration with developer tools
- Software defined networks
- Users, teams, quotas, access rights, etc...
- Turning source code into deployable components
- Build, manage and deliver application descriptions at scale
- Etc, etc, etc ...

OpenShift v3 Architecture



THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos