Container Security

Marc Skinner
mskinner@redhat.com
Principal Solutions Architect

A bit about me ...

Marc Skinner

- 10 years at Red Hat
- Live in Minneapolis, MN
- Married, 2 kids, 1 cat
- 1st time in Calgary
- Run the MSP RHUG
- http://people.redhat.com/mskinner

Security in pre-container era

Security Best Practices

- Reduce attack surface area
- Standard Operating Environment
- Errata updates for vulnerabilities
- Run processes at minimum privilege level
- Grant users minimum privilege level
- Log everything within reason
- Encrypt sensitive data at rest and in transit
- Application tiering (web/app/db)

RHEL Security Features

- Security Certifications (EAL4+)
- SELinux/sVirt
- CGroups and Namespaces
- Packet filtering
- Kernel capabilities
- Satellite and Errata
- OpenScap Scanning

Security presos

- “SeLinux for mere mortals” by Thomas Cameron
  - http://people.redhat.com/tcameron/Summit2015/selinux/cameron-selinux-summit_2015.pdf

- “RHEL Security in the real world” by Marc Skinner
  - http://people.redhat.com/mskinner/rhug/q3.2012/rhel_security-in_the_real_world.pdf

What about virtualization?

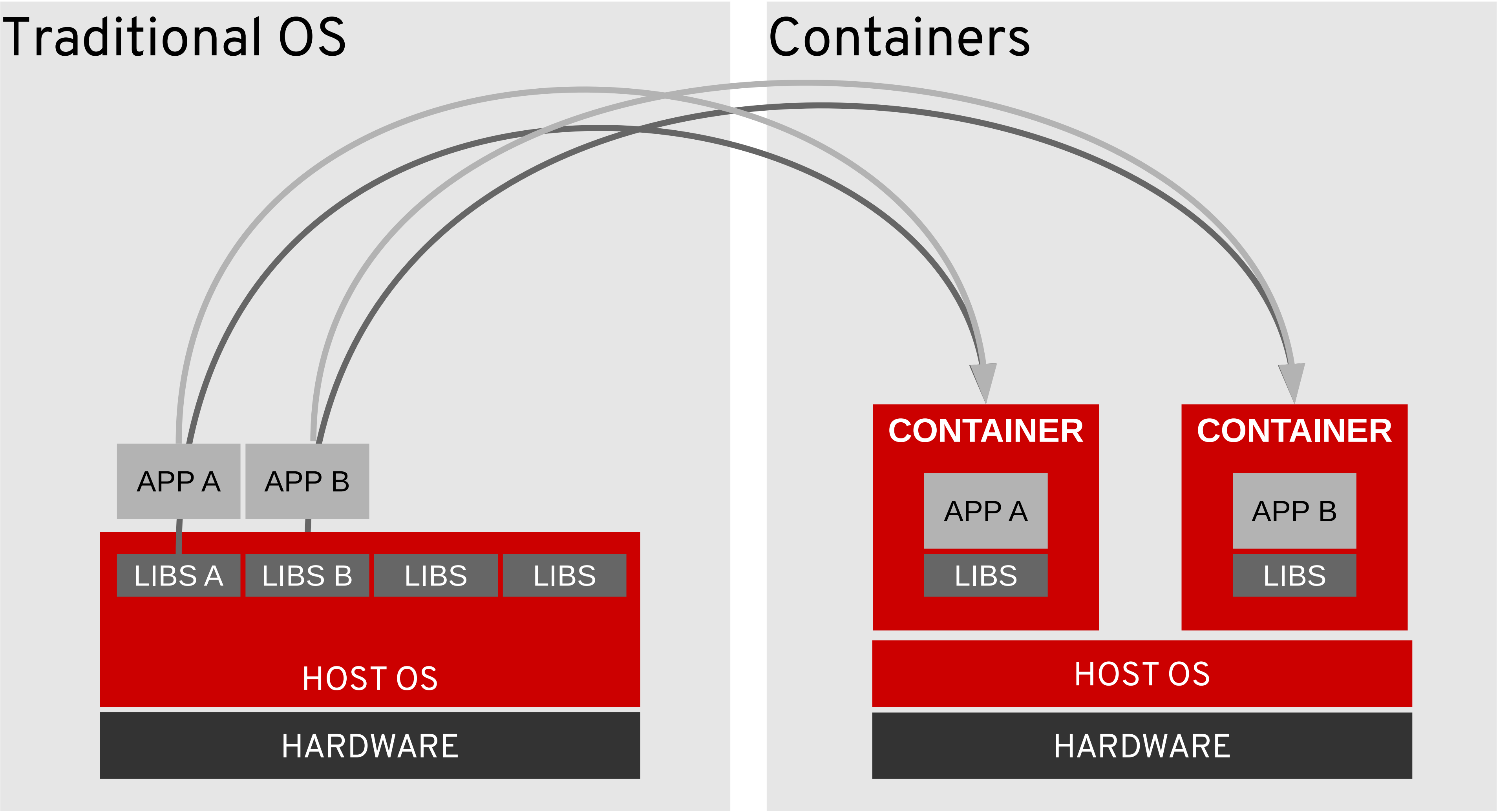
- Same best practices for security apply
- Hypervisor host security matters
  - RHEL and KVM use SELinux/sVirt
- Breaking out of VM requires
  - Gaining root on VM
  - Break out of SELinux/sVirt

What about containers?

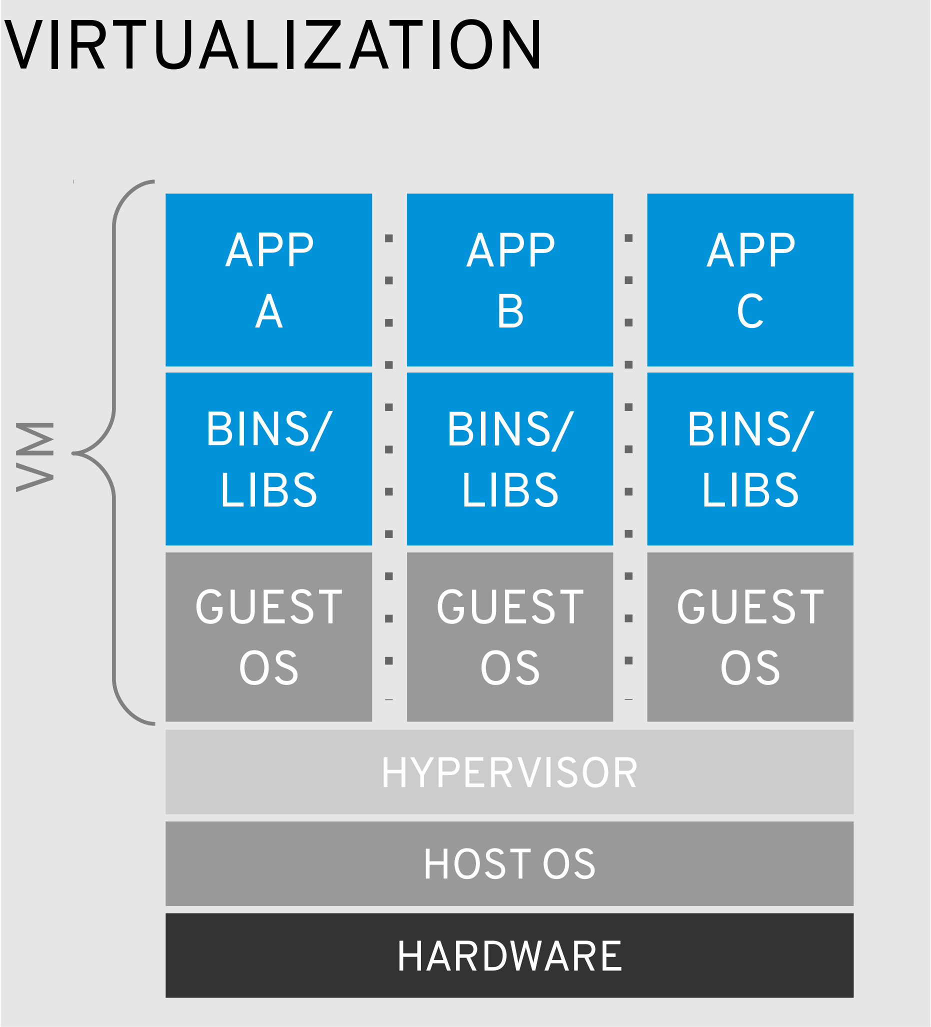
- Same best practices for security apply
- Container and virtualization technologies differ
  - Containers isolate processes on same system
  - Virtualization isolates entire hosts on same system
- Containers and VMs enforce different security layers
- Breaking out of container requires
  - Gaining (root)
  - Breaking out of Namespaces
  - Break out of SELinux/sVirt

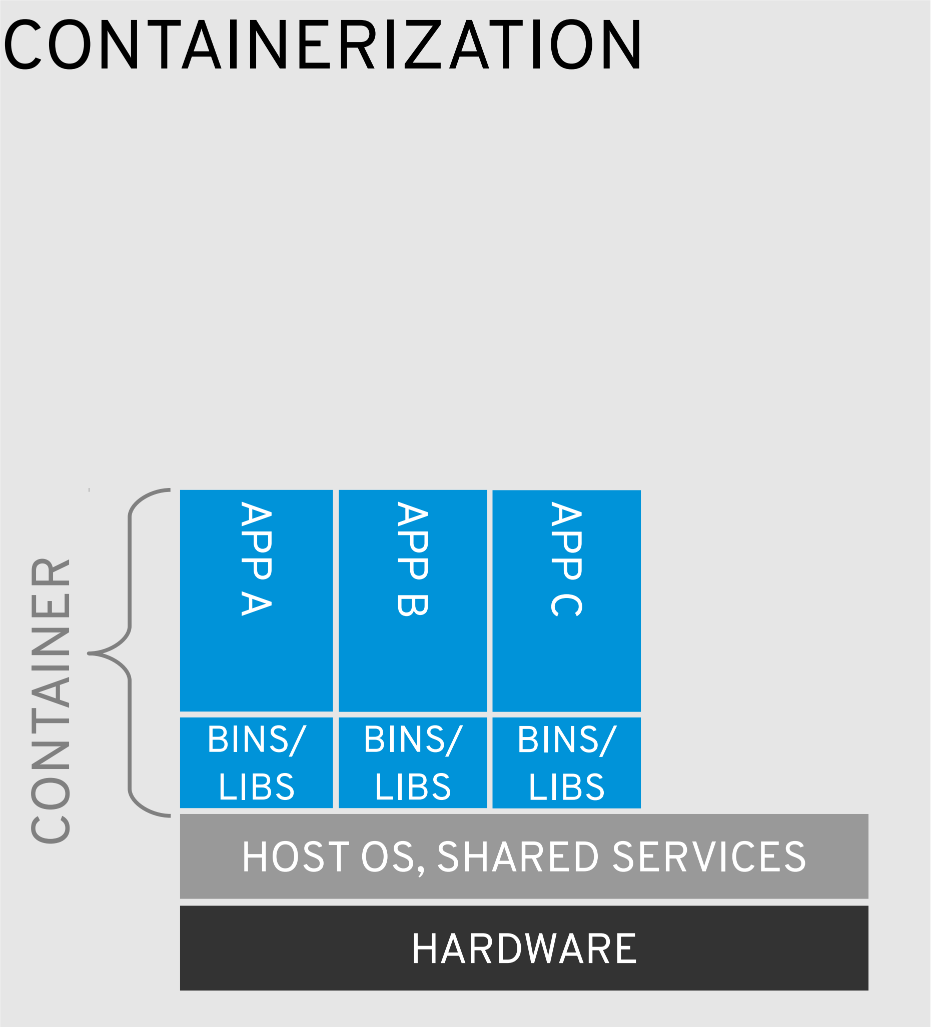
What are Linux Containers?

TRADITIONAL OS VS. CONTAINERS



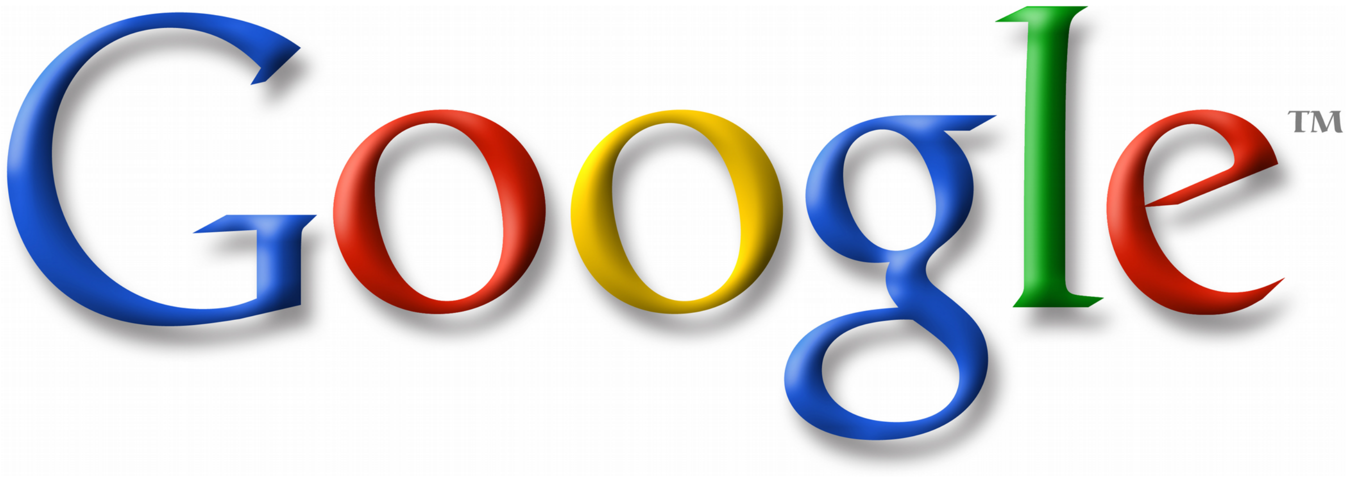
VIRTUALIZATION AND CONTAINERS


VIRTUALIZATION


CONTAINERIZATION

TOP 4 FACTS ABOUT CONTAINERS

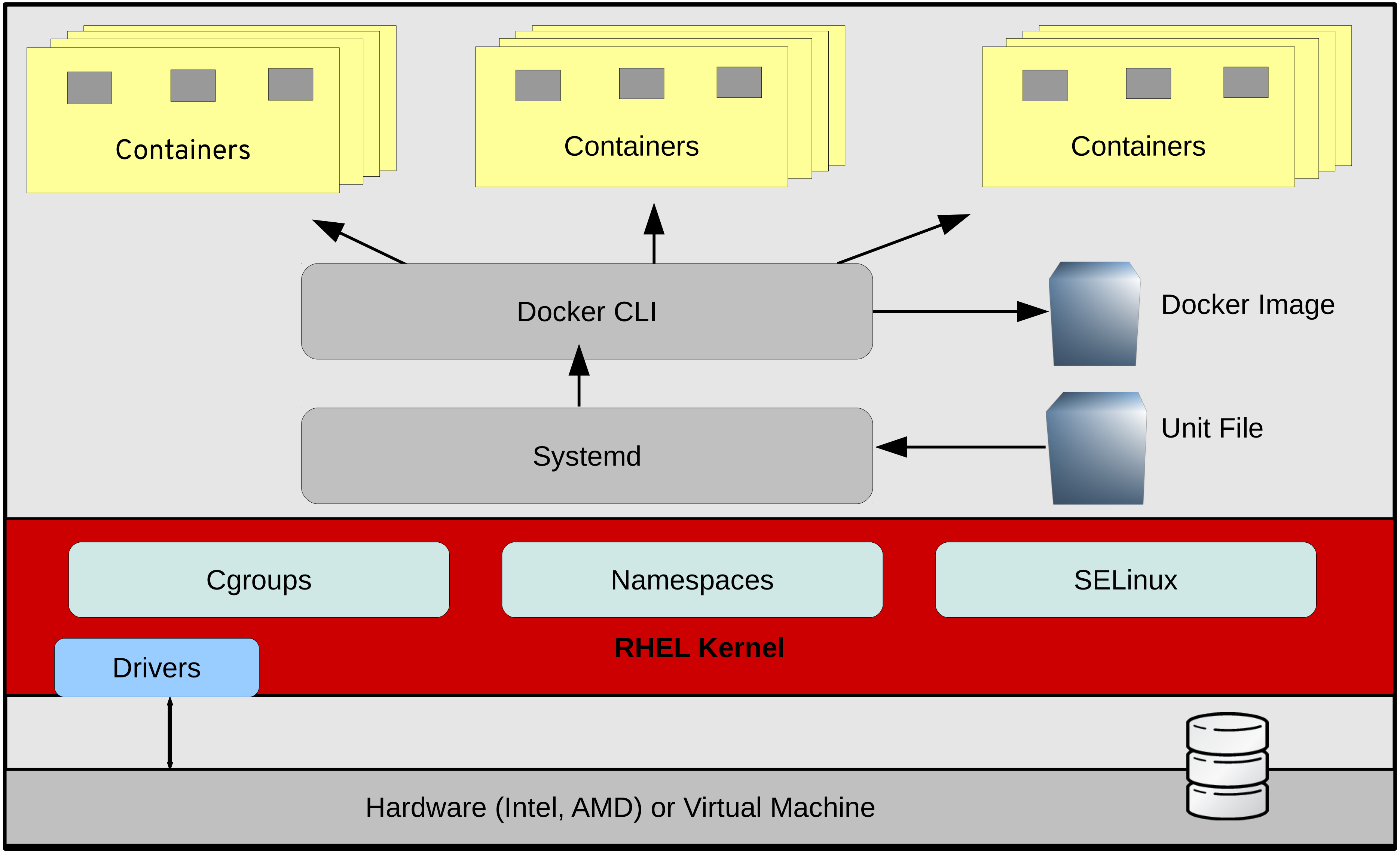
1. Containers are not new
2. Containers do not equal virtualization
3. Containers are not universally portable
4. Containers are enterprise-ready



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

- Eric Brewer, VP of Infrastructure, Google

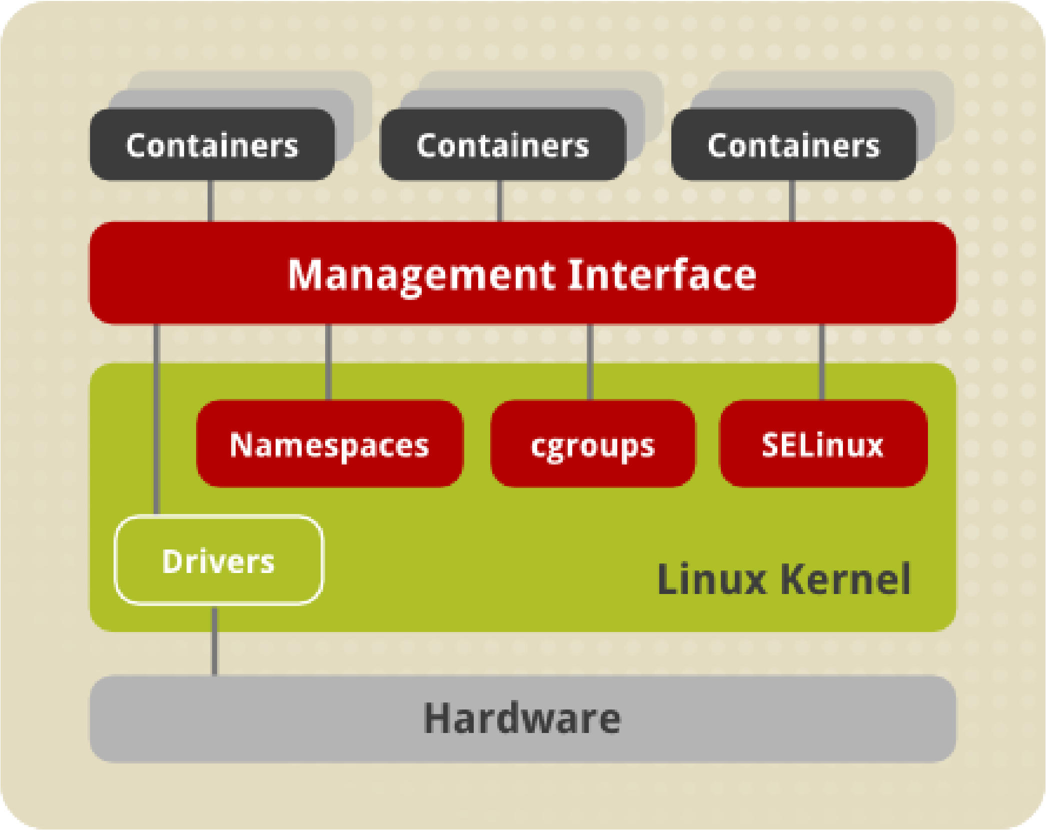
RHEL 7 Containers Architecture



Linux Containers Architecture

Namespaces

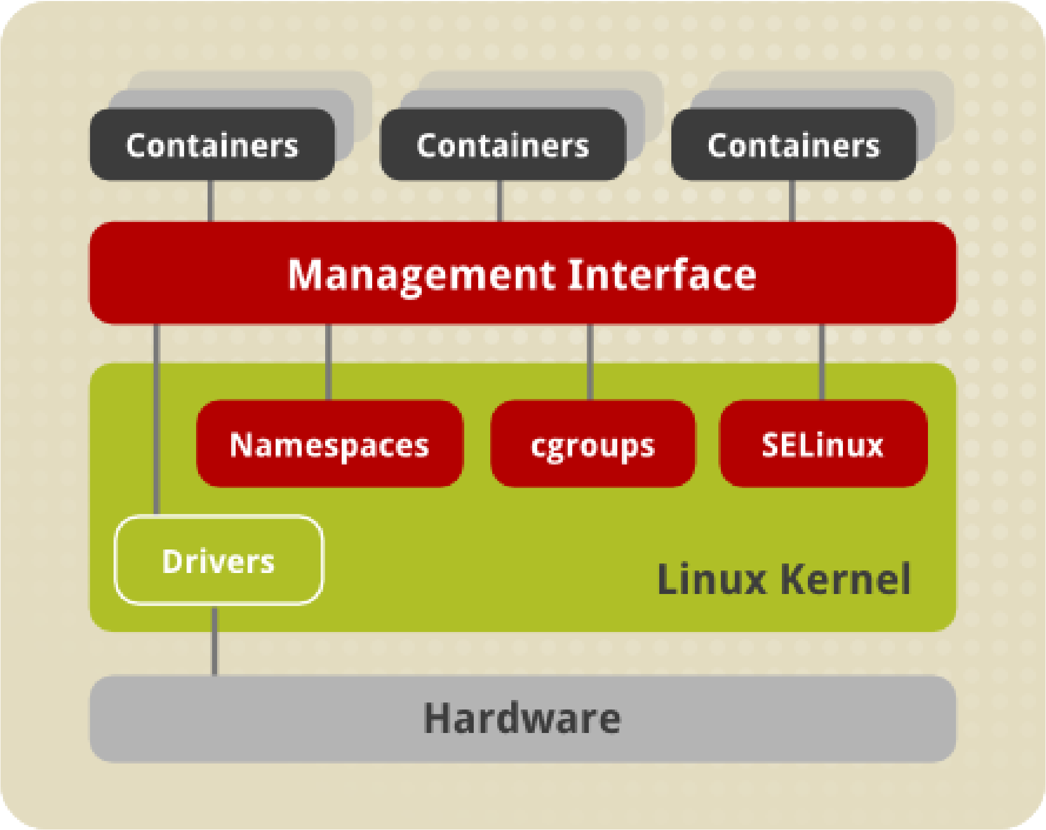
- Allow abstraction of a system resource and make it appear as a separated instance
- Several containers can use the same resource simultaneously without creating a conflict
- Introduced into upstream kernel July 2008 time frame



Linux Containers Architecture

Control Groups (cgroups)

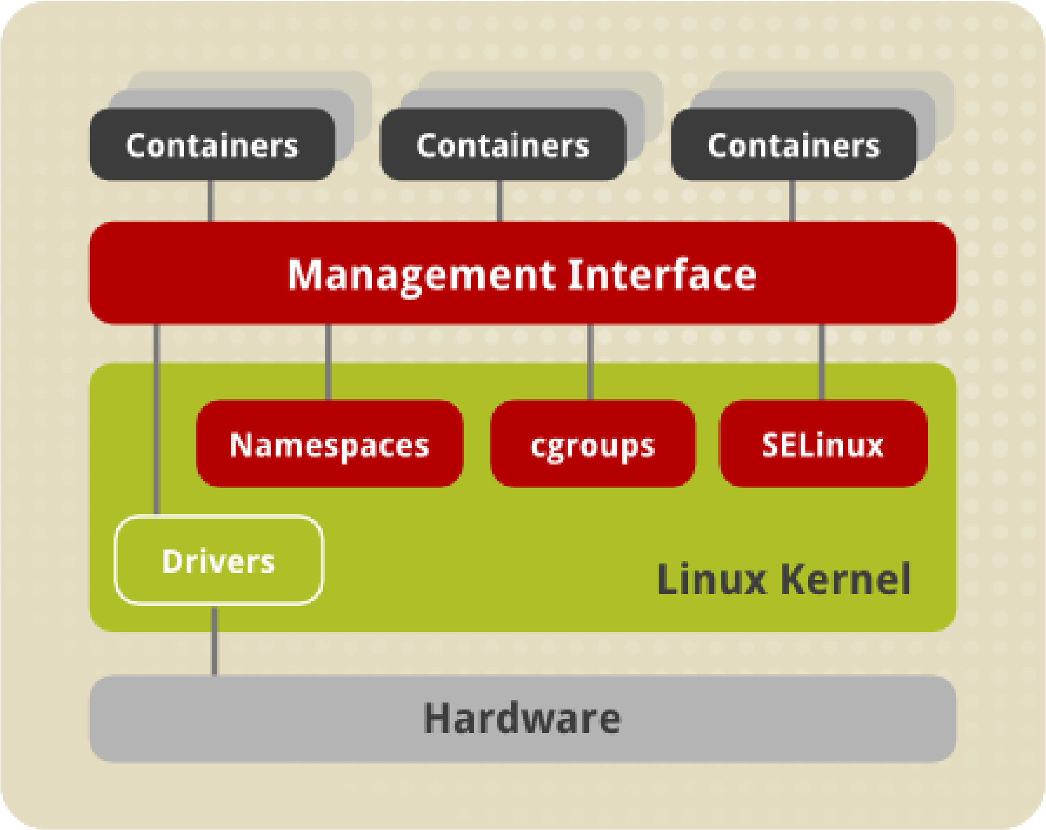
- Allow processes to be grouped for system resource management
- Allocates CPU time, system memory, network bandwidth, or combinations of these among users defined groups of tasks
- Managed with systemd slice, scope, and service units
- Introduced into upstream kernel in early 2006



Linux Containers Architecture

SELinux

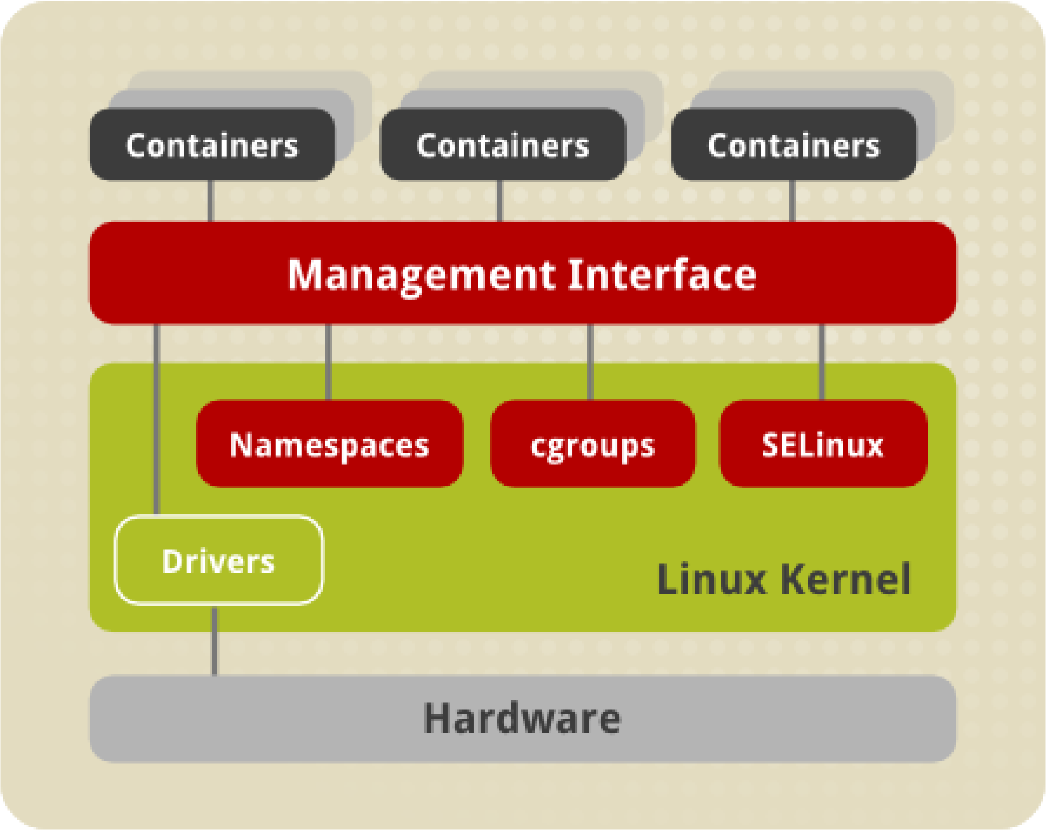
- Provides secure separation of containers by applying policies and labels
- Released upstream December 2000
- Integrates with containers through sVirt
- sVirt released upstream early 2009



Linux Containers Architecture

Management Interface

- In RHEL 7, the Docker application is the main management tool for Linux Containers
- Docker adds several enhancements, such as portability, version control and application packaging



Container Host Security

Security outside container

- Host System
- Kernel Capabilities
- SELinux
- Control Groups
- Namespaces
- Logging

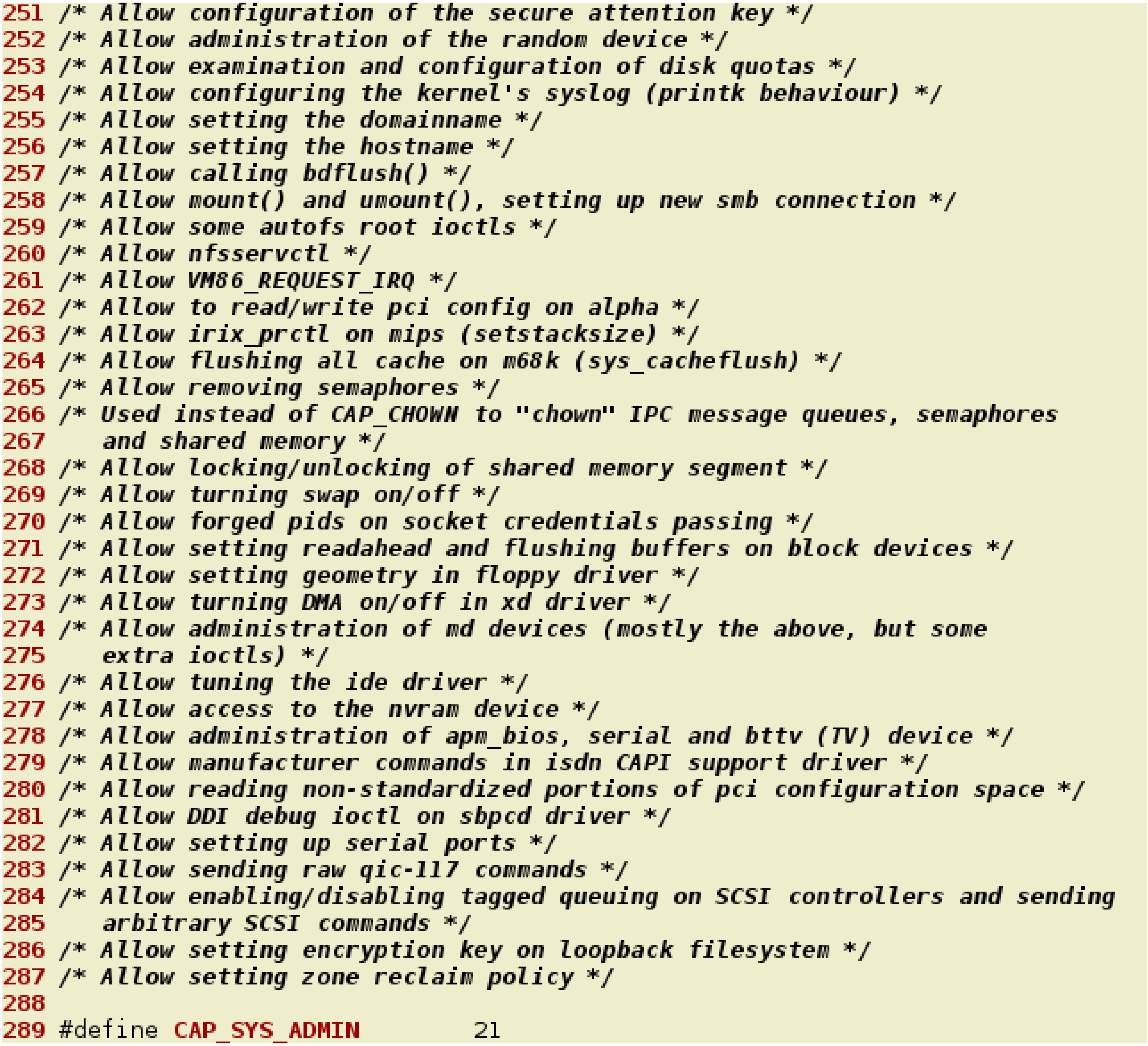
Host System

- Frequent security updates
    - Kernel, docker, kubernetes, systemd and journald
- Firewall rules
- By design Docker has to run as root
    - Only trusted users should have access
    - API access locked down, use TLS
    - Disable Docker Hub
        - /etc/hosts : 127.0.0.1 index.docker.io

Linux Kernel Capabilities

- /usr/src/linux/include/linux/capability.h
- Fine grained access control via capabilities
- 38 distinct sets
- Enable / Disable Kernel system calls

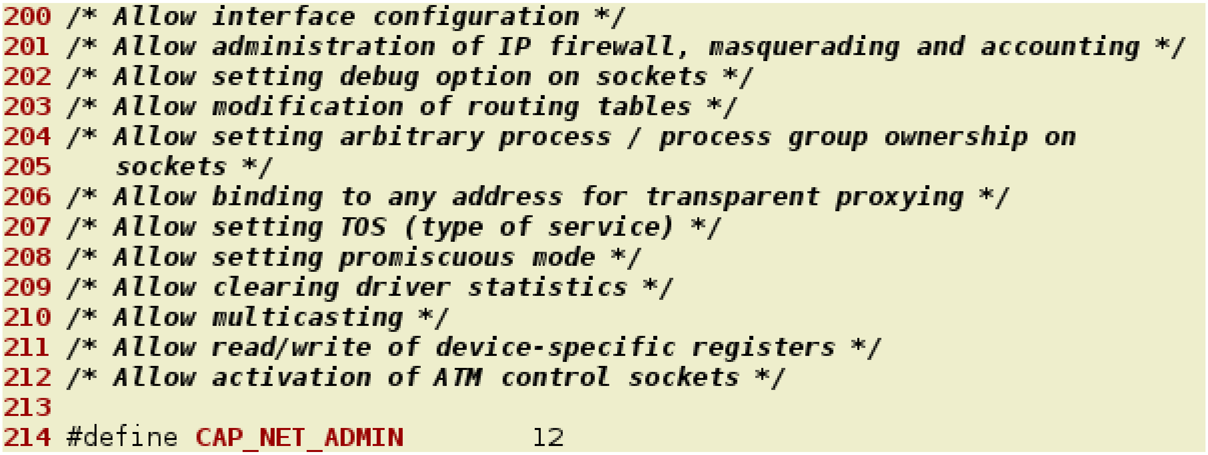
CAP SYS_ADMIN “Catch all” - removed



```
/* Allow configuration of the secure attention key */
/* Allow administration of the random device */
/* Allow examination and configuration of disk quotas */
/* Allow configuring the kernel's syslog (printk behaviour) */
/* Allow setting the domainname */
/* Allow setting the hostname */
/* Allow calling bdflush() */
/* Allow mount() and umount(), setting up new smb connection */
/* Allow some autofs root ioctls */
/* Allow nfsservctl */
/* Allow VM86_REQUEST_IRQ */
/* Allow to read/write pci config on alpha */
/* Allow irix_prctl on mips (setstacksize) */
/* Allow flushing all cache on m68k (sys_cacheflush) */
/* Allow removing semaphores */
/* Used instead of CAP_CHOWN to "chown" IPC message queues, semaphores
   and shared memory */
/* Allow locking/unlocking of shared memory segment */
/* Allow turning swap on/off */
/* Allow forged pids on socket credentials passing */
/* Allow setting readahead and flushing buffers on block devices */
/* Allow setting geometry in floppy driver */
/* Allow turning DMA on/off in xd driver */
/* Allow administration of md devices (mostly the above, but some
   extra ioctls) */
/* Allow tuning the ide driver */
/* Allow access to the nvram device */
/* Allow administration of apm_bios, serial and bttv (TV) device */
/* Allow manufacturer commands in isdn CAPI support driver */
/* Allow reading non-standardized portions of pci configuration space */
/* Allow DDI debug ioctl on sbpcd driver */
/* Allow setting up serial ports */
/* Allow sending raw qic-117 commands */
/* Allow enabling/disabling tagged queuing on SCSI controllers and sending
   arbitrary SCSI commands */
/* Allow setting encryption key on loopback filesystem */
/* Allow setting zone reclaim policy */

#define CAP_SYS_ADMIN        21
```

CAP NET_ADMIN “Configure network” - removed



```
/* Allow interface configuration */
/* Allow administration of IP firewall, masquerading and accounting */
/* Allow setting debug option on sockets */
/* Allow modification of routing tables */
/* Allow setting arbitrary process / process group ownership on
   sockets */
/* Allow binding to any address for transparent proxying */
/* Allow setting TOS (type of service) */
/* Allow setting promiscuous mode */
/* Allow clearing driver statistics */
/* Allow multicasting */
/* Allow read/write of device-specific registers */
/* Allow activation of ATM control sockets */

#define CAP_NET_ADMIN        12
```

32bit system calls - removed

- *** need to add ALL capabilities *** be aware!

```
#docker run --cap-add=ALL rhel7 /bin/my32bitapp.bin
```

Allowed Capabilities

- CHOWN
- DAC_OVERRIDE
- FSETID
- FOWNER
- MKNOD
- NET_RAW
- SETGID
- SETUID
- SETFCAP
- SETPCAP
- NET_BIND_SERVICE
- SYS_CHROOT
- KILL
- AUDIT_WRITE

```
#docker run --cap-drop SETUID --cap-drop SETGID --cap-drop
FOWNER rhel7 /bin/sh
```

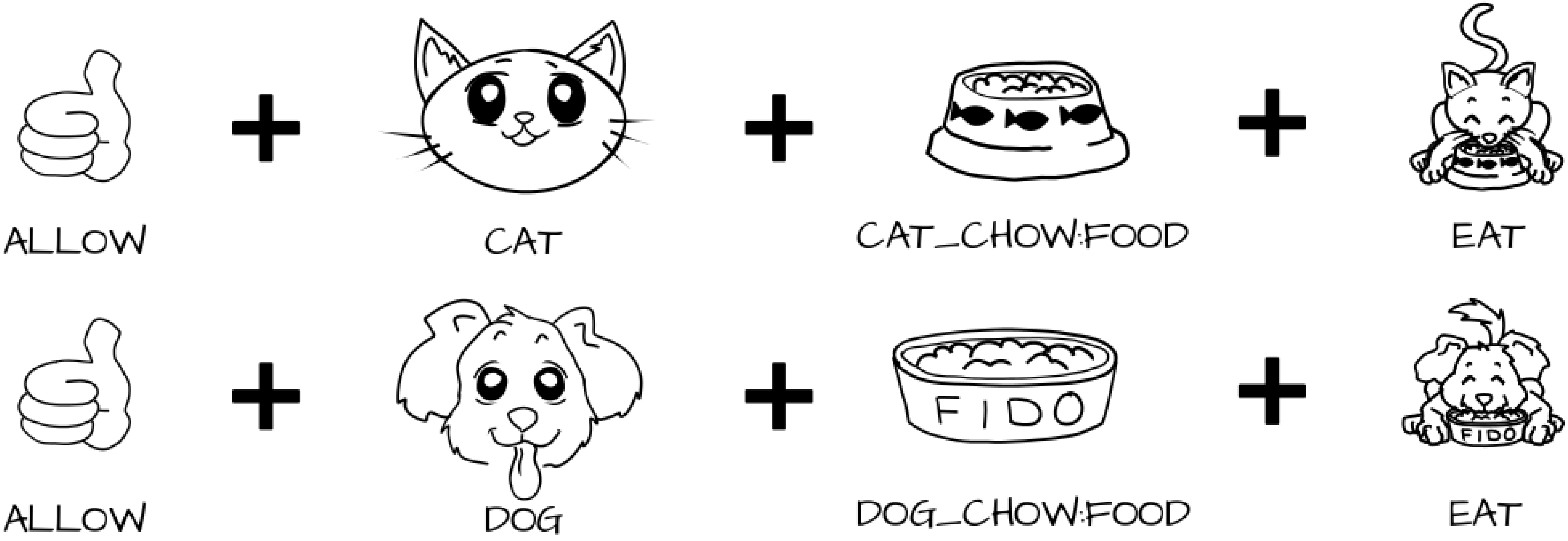
Add Capabilities, don't jump to –privileged mode

- Running ntpd or crony in your container?

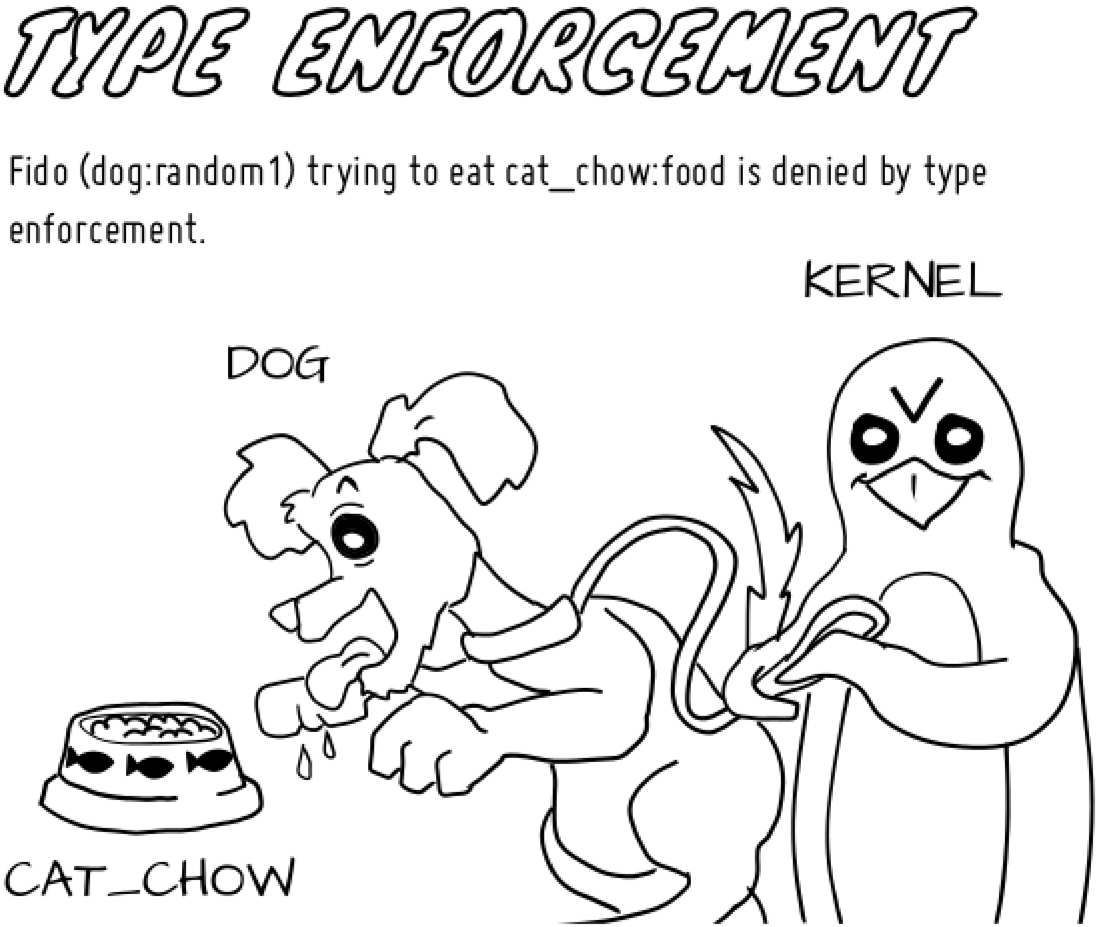
```
#docker run -d -n ntpd --cap_add SYS_TIME ntpd
```

SELinux Type Enforcement 1/2

- Container Process type svirt_lxc_net_t
- Container File type svirt_sandbox_file_t
- Container can only write to svirt_sandbox_file_t



SELinux Type Enforcement 2/2

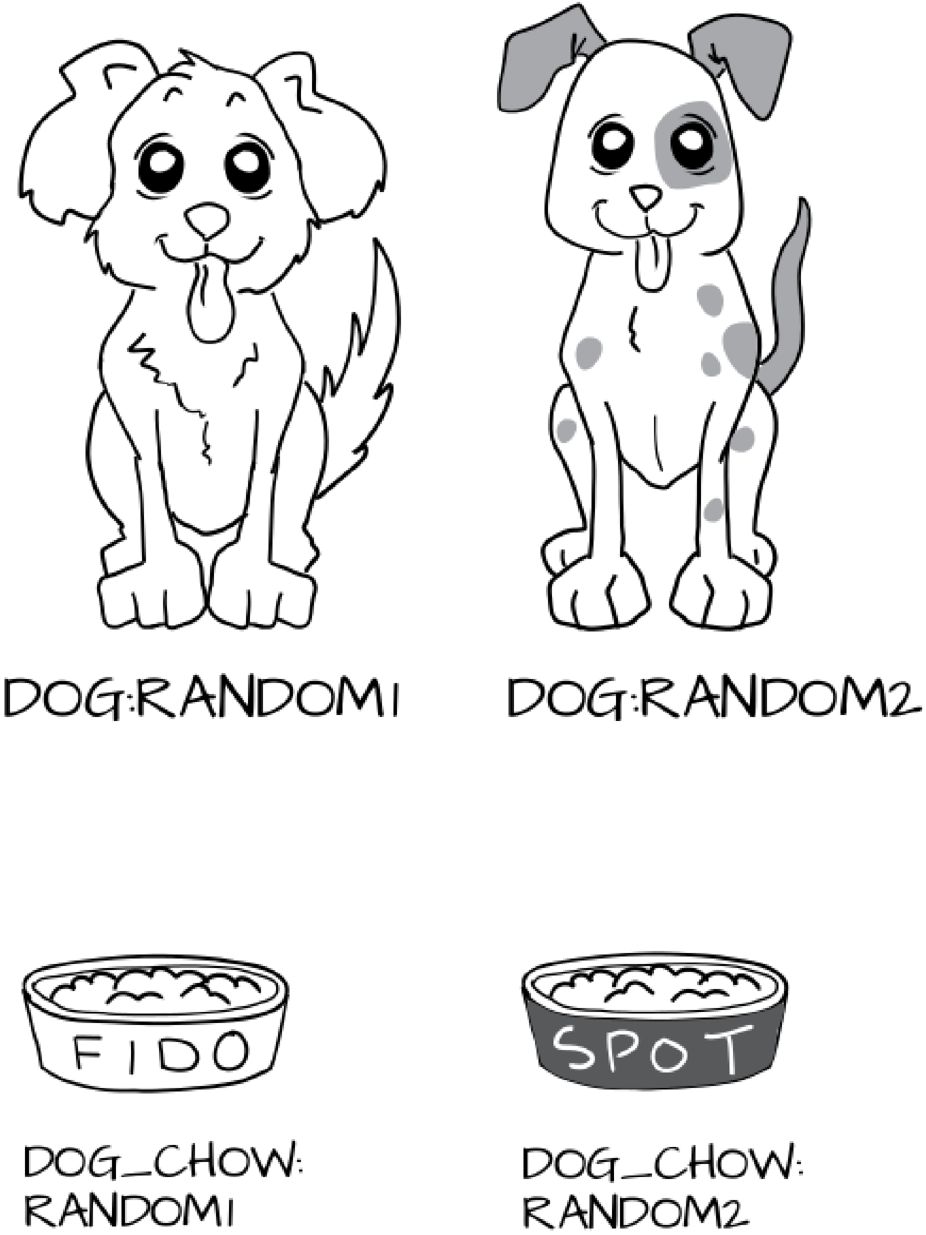


TYPE ENFORCEMENT

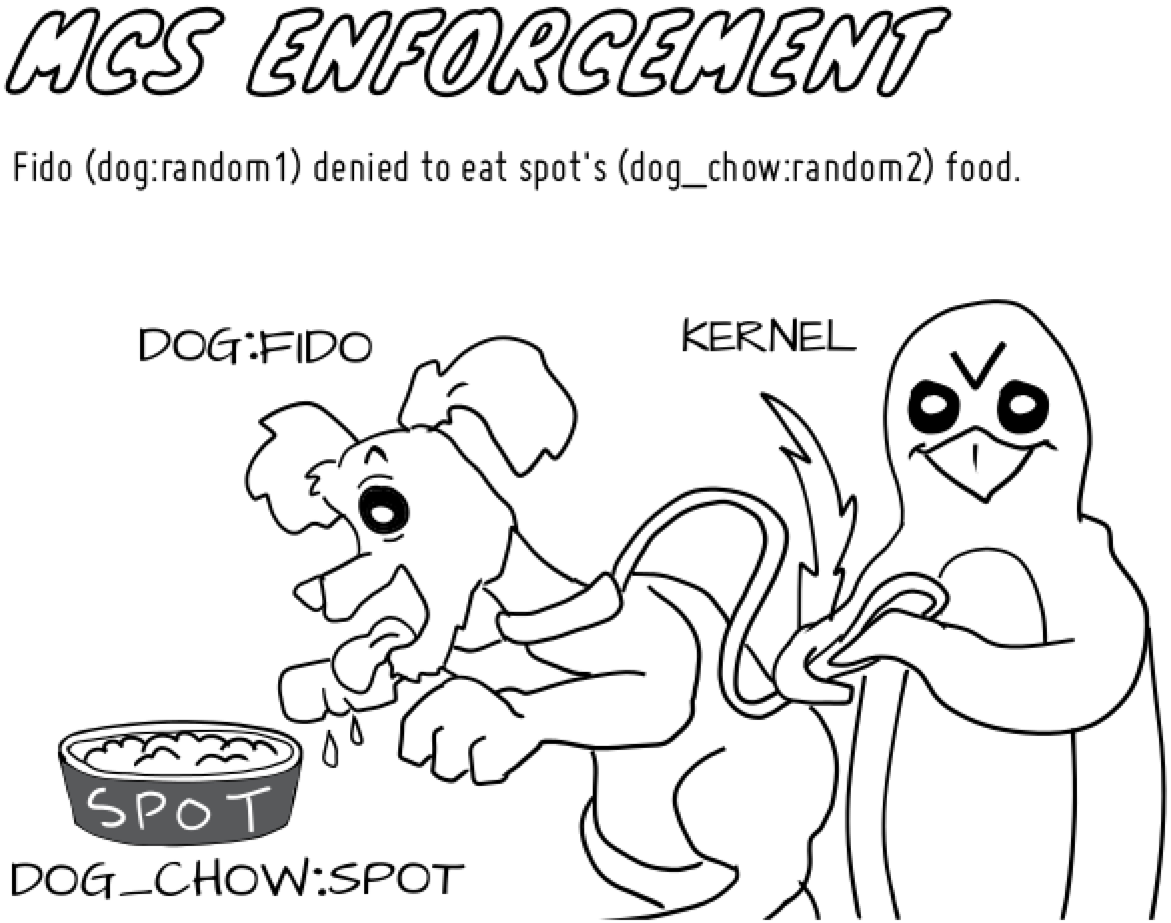
Fido (dog:random1) trying to eat cat_chow:food is denied by type enforcement.

SELinux MCS (Multi Category Security) 1/2

- Containers use same SELinux types
- Docker daemon picks random label when starting container
- All container content and processes are labeled
- Locks container objects and processes down



SELinux MCS 2/2


MCS ENFORCEMENT
Fido (dog:random1) denied to eat spot's (dog_chow:random2) food.
DOG:FIDO
KERNEL
SPOT
DOG_CHOW:SPOT

CGroups

- Noisy neighbor
  - Resource accounting and limiting
  - Limit container resource impact
- Prevent denial-of-service attacks

Namespaces

- Not everything in Linux is namespaced
  - SELinux, CGroups, /sys, /proc/sys and kernel mods
- Docker uses
  - User, Process, Network, Mount, Hostname and Shared Memory

Logging

- Volume mount /dev/log

```
# docker run -v /dev/log:/dev/log fedora logger "this is a test"
# journalctl -b |grep "this is a test"
Jul 16 15:05:41 myhost.domain logger[29422]: this is a test
```

- Docker > 1.7 supports journald as log driver

```
# docker -d –selinux-enabled –log-driver=journald
```

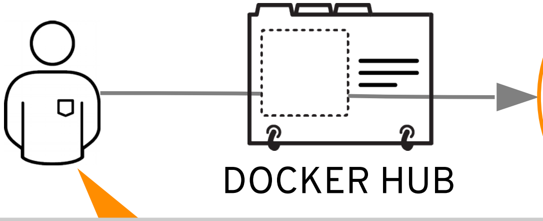
- Docker logs using json-file driver by default

Container Security

Security inside container

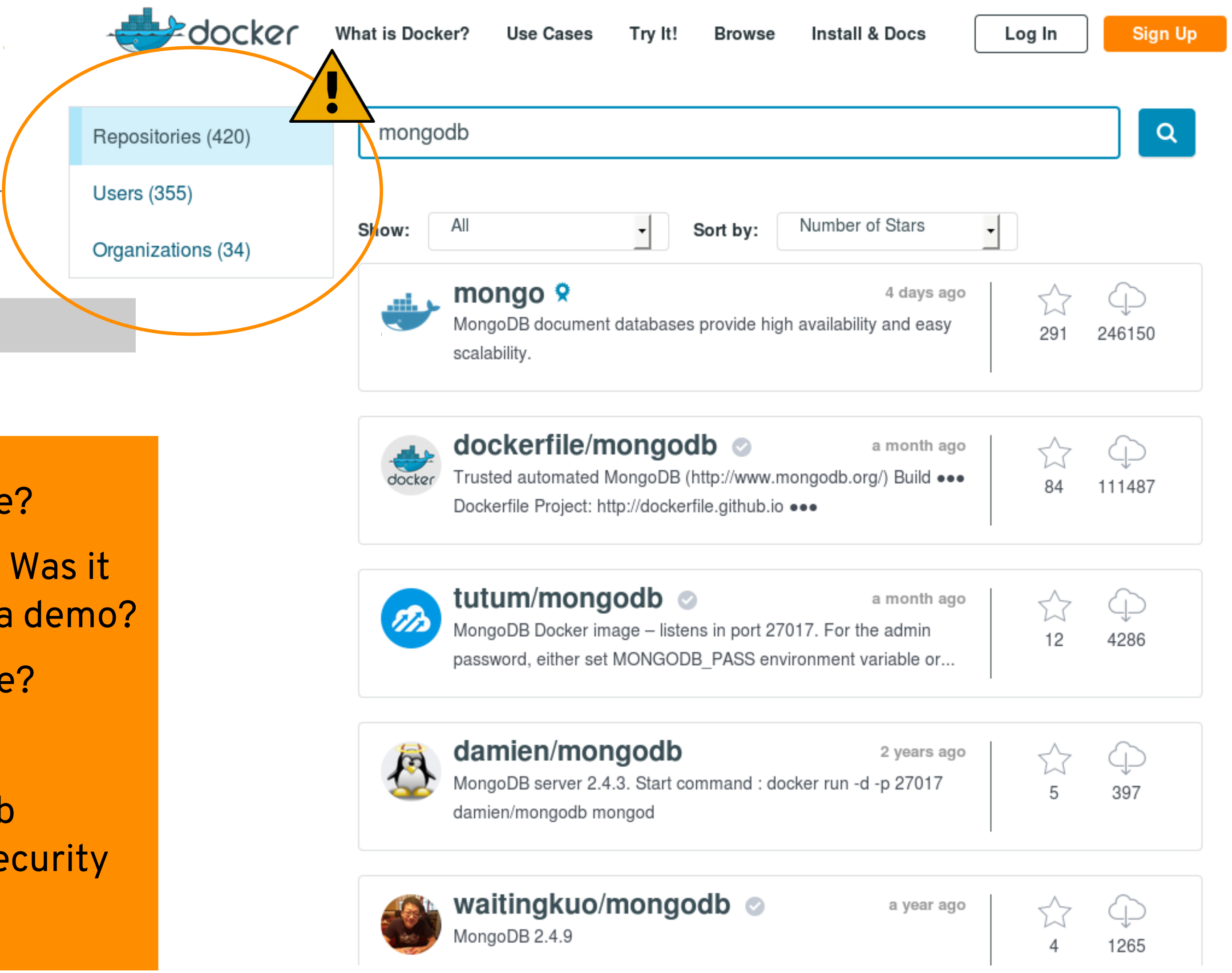
- Kernel file systems read-only
- Container image mounted with nodev option

Chain of Trust



```
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?
- +30% of docker hub images have high security vulnerabilities



Repositories (420)

Users (355)

Organizations (34)

mongodb

Show: All

Sort by: Number of Stars

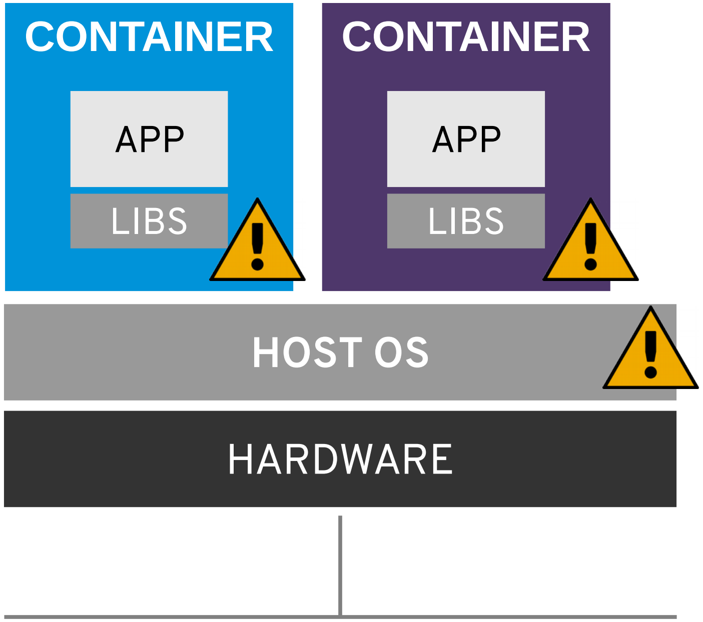
Repository	Updated	Description	Stars	Downloads
mongo	4 days ago	MongoDB document databases provide high availability and easy scalability.	291	246150
dockerfile/mongodb	a month ago	Trusted automated MongoDB (http://www.mongodb.org/) Build ••• Dockerfile Project: http://dockerfile.github.io •••	84	111487
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

Secure Hosts and Containers

RED HAT CONTAINER CERTIFICATION

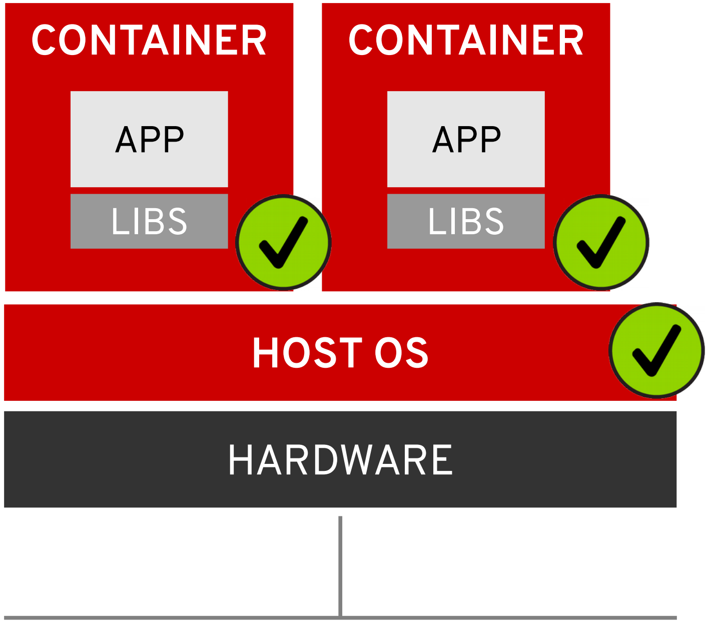
UNTRUSTED

- 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 container content
- Proven portability
- Container Development Kit



Container Development

- **CDK 2.3 available to Red Hat Partners/Customers**
- Components
  - RHEL 7 Vagrant for Libvirt and Virtualbox
    - Currently targeting Atomic Host and Stand Alone
  - Includes 30+ RHSCL “technology stacks” - Dockerfiles for easy container set-up
    - Python, Ruby, PHP, Perl, Node.js, MariaDB, MySQL, PostgreSQL, MongoDB, Apache, nginix, etc.
  - **Docker-lint** : a tool for assessing the quality of Dockerfiles
  - Container certification tools and documentation

Container Development

- Docker Development Best Practices
  - https://access.redhat.com/articles/1483053#image_scanner
- Linter for verifying Dockerfiles
  - https://access.redhat.com/labs/linterfordockerfile

RHEL Atomic

Red Hat Enterprise Linux Atomic Host

Foundational offering in Red Hat's container solution portfolio

IT IS RED HAT ENTERPRISE LINUX

- The hardware ecosystem
- Military-grade security
- Stability and Reliability

OPTIMIZED FOR CONTAINERS

MINIMIZED FOOTPRINT

- Tuned for running Linux containers
- Compatibility with Red Hat Enterprise Linux

SIMPLIFIED MAINTENANCE

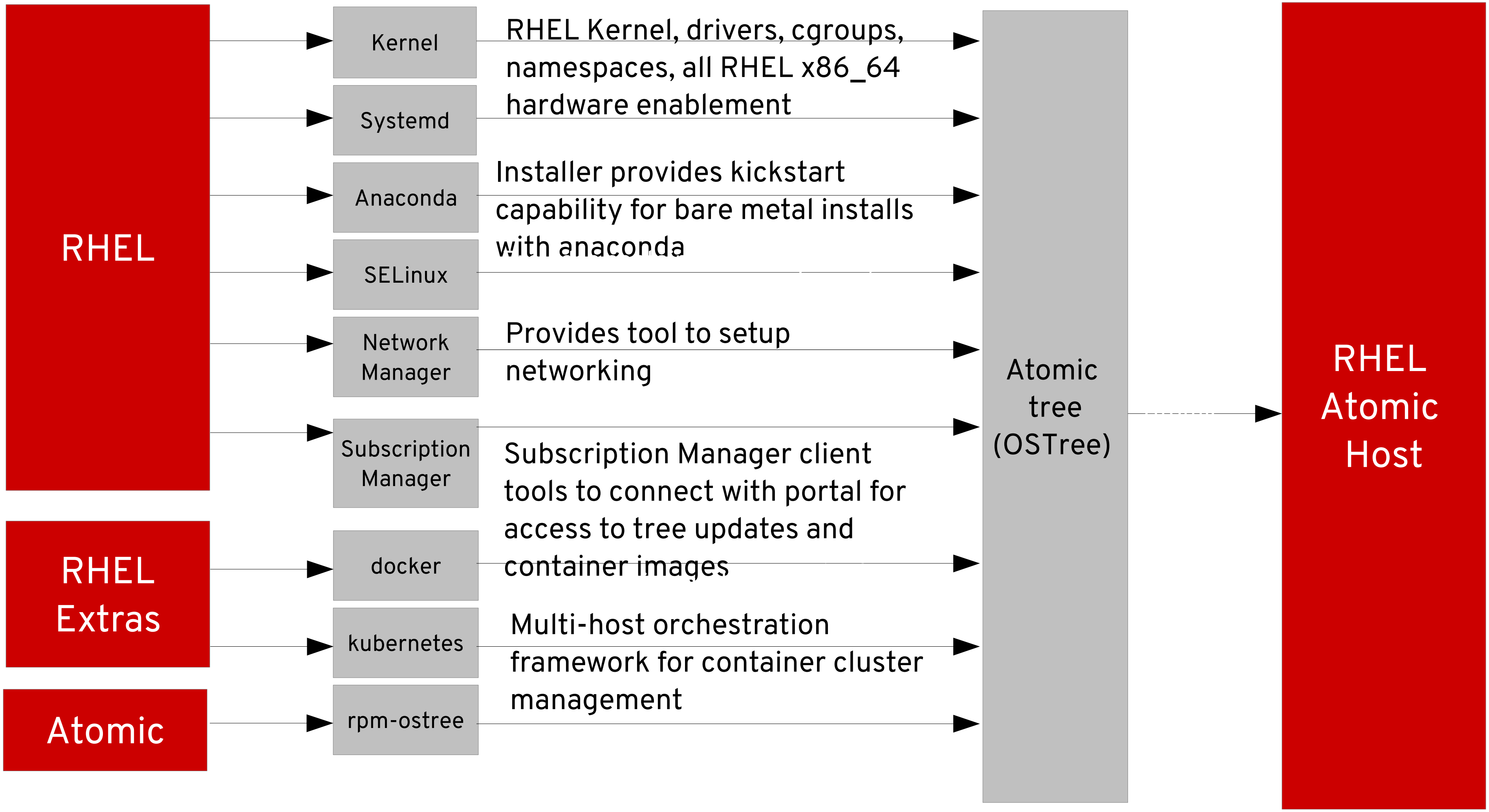
Easy to use images:

- Deploy
- Update
- Rollback

ORCHESTRATION AT SCALE

- Container orchestration
- Multi-host
- Simple building block

RHEL Atomic Host Component Diagram
RHEL & RHEL Extras Inheritance Model



RHEL

RHEL Extras

Atomic

Kernel

Systemd

Anaconda

SELinux

Network Manager

Subscription Manager

docker

kubernetes

rpm-ostree

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

Installer provides kickstart capability for bare metal installs with anaconda

Provides tool to setup networking

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

Multi-host orchestration framework for container cluster management

Atomic tree (OSTree)

RHEL Atomic Host

Additional Security

- Immutable
- OSTree – bit level updates (atomic), with roll back ability – full OS versioning!
- Seccomp (Security Profiles)
  - Profile in JSON format
  - Block specific system calls, some are overlapped by CAP_SYS

Atomic Management

- Satellite 6
  - Content Views support OSTree
  - Golden Image

- Cockpit
  - Web front end to manage Atomic hosts in real time
  - Performance metrics in real time

Container Scanning

Scanning

- Atomic Scan
    - Allows for the inspection of Linux containers to identify known vulnerabilities and out-of-compliance issues.
    - Plug-able framework
        - OpenScap
        - 3rd party - Black Duck

OpenShift Container Platform Security

OpenShift Security

- API Authentication
    - X509 Cert, Oauth Access Token, SAML
- Identity Integrations
    - Roles, LDAP, AD
- Service Accounts
- Security Contexts
- Secrets
    - Encrypted variable values
- Image Build control – secure registry

OpenShift Network Security

- OVS Multi-tenant Plugin
  - Provides unique VNID for each project

In Closing

Container Security is like an Onion

- Host Security
- Container Security
- Platform Security

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos