redhat.

Linux Containers

A software architecture lecture

Josef Karasek
Quality Engineer, Red Hat Middleware

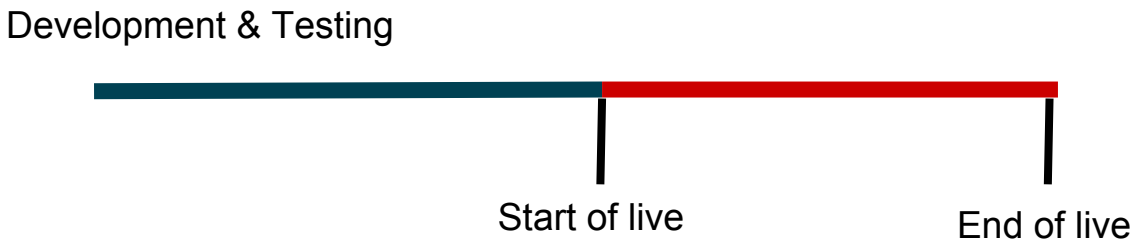
2017-04-03

Agenda

- Linux containers
  - Motivation
  - Containers as a packaging mechanism
  - Containers as process isolation

When does SW product provide value?

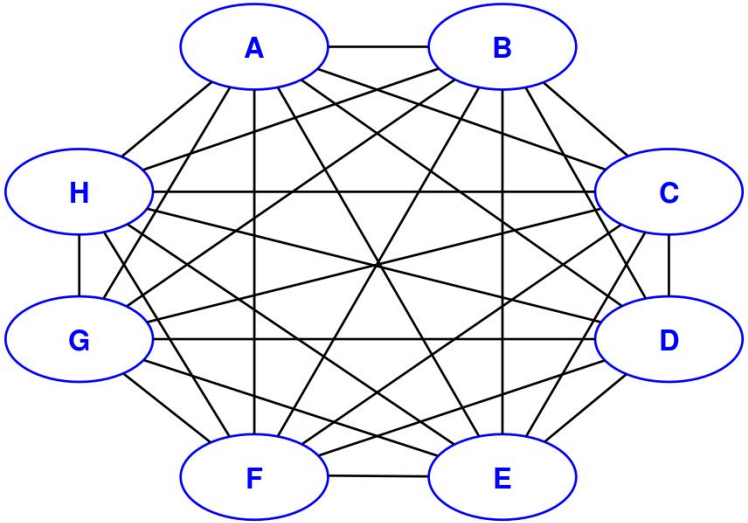
- SW development is hard and long process
- Production SW needs to be maintained and supported



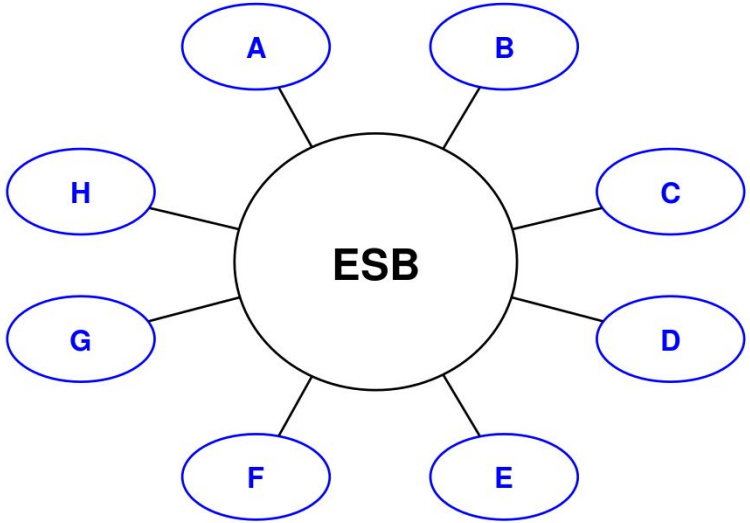
Systems have become more complex

- More subsystems involved
- More infrastructure to manage
- Development cycle getting shorter - change is very common

Topologies used today



Impact = N



Impact = 1

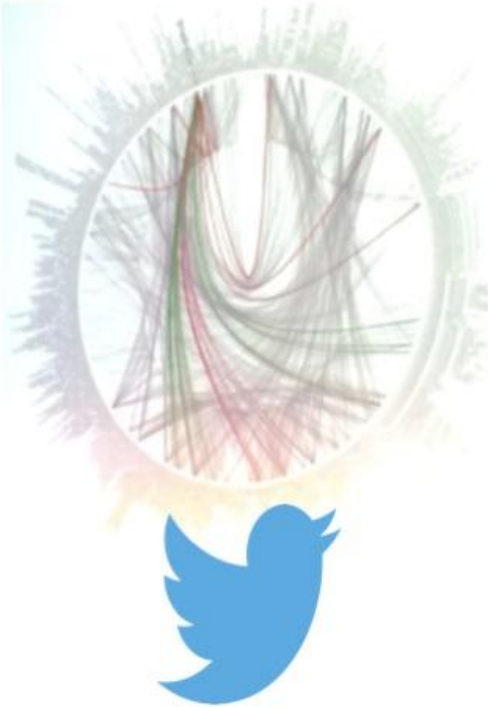
450 microservices

HAIL O

500+ microservices

NETFLIX

500+ microservices



Source:
Netflix: http://www.slideshare.net/BruceWong3/the-case-for-chaos
Twitter: https://twitter.com/adrianco/status/441883572618948608
Hail-o: https://sudo.hailoapp.com/services/2015/03/09/journey-into-a-microservice-world-part-3/

Immutable architecture

Evolving SW architecture at scale

- Systems parts are not modified but replaced by new ones
- Immutable Infrastructure brings stability & consistency to the environment (update success / fail?)
- Easy to move forward, but also backward
- Involves also process/environment change

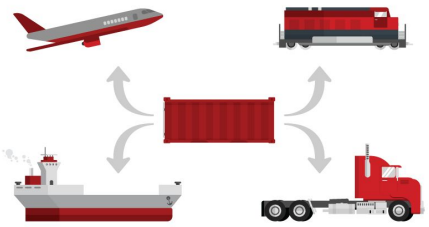
Trash Your Servers and Burn Your Code

-- Chad Fowler

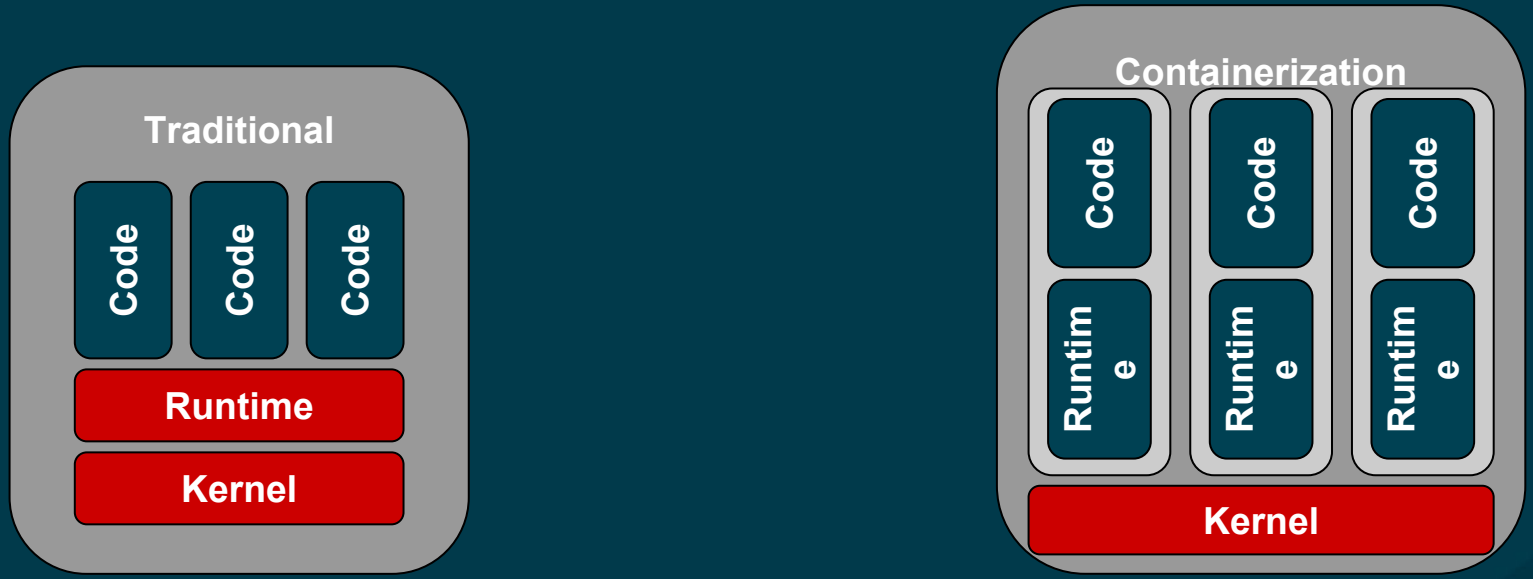
Linux containers

Containers as a packaging mechanism

- Code and its runtime dependencies bundled together
- Format of container is well understood
- Commonly a .tar archive of
  - File system
  - Static binary



Containers as a packaging mechanism II



What’s inside a container

Inside / Outside

Code

Compiled binary

Shared libraries

Configuration scripts

JRE, Python…

Configuration

Injected at runtime - easy customization

Files, environment variables…

Data

Persisted outside

Containers can be restarted with no data loss

Data has its own lifecycle, independent from code

Example container: MySQL

Code	Configuration	Data
`mysqld`	`/etc/my.cnf`	`/var/lib/mysql`

Runtime dependencies

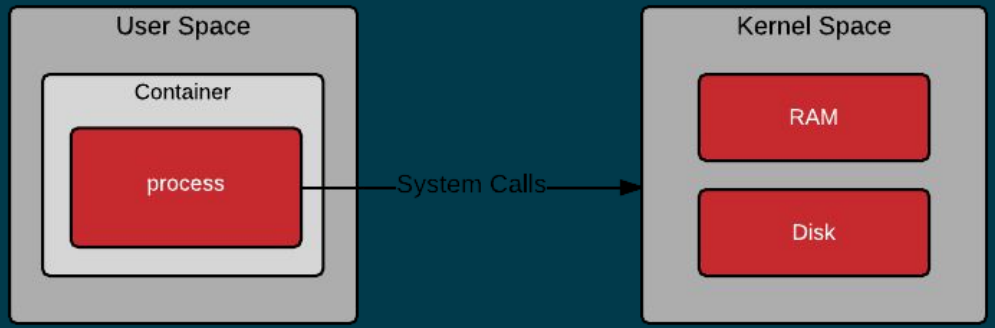
```
$ ldd /usr/libexec/mysqld
    linux-vdso.so.1
    libsystemd.so.0 => /lib64/libsystemd.so.0
    libpthread.so.0 => /lib64/libpthread.so.0
...
```

Containers as process isolation

A container is just a fancy process

Namespaces - restrict what resources the container can use

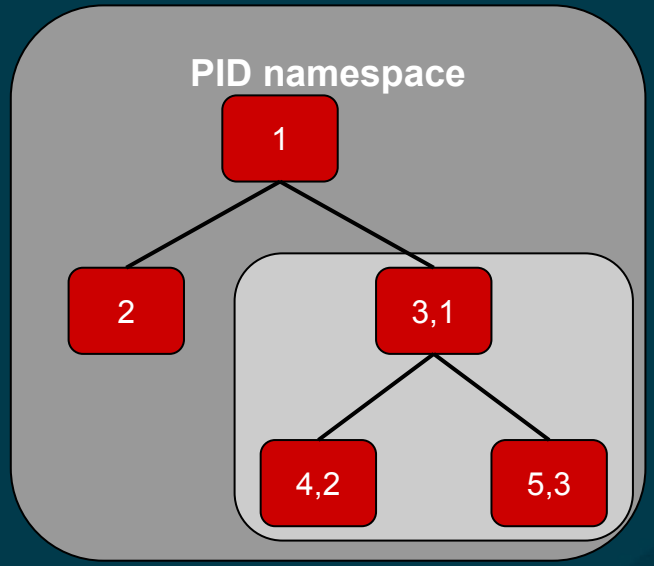
CGroups - define how much of a resource the container can use



Namespaces

Provide containers with their own view of the underlying Linux system

- pid
- mnt
- net
- ipc (inter-process comm)
- uts (hostname, domainname)
- user



Control Groups

Resource control and accounting

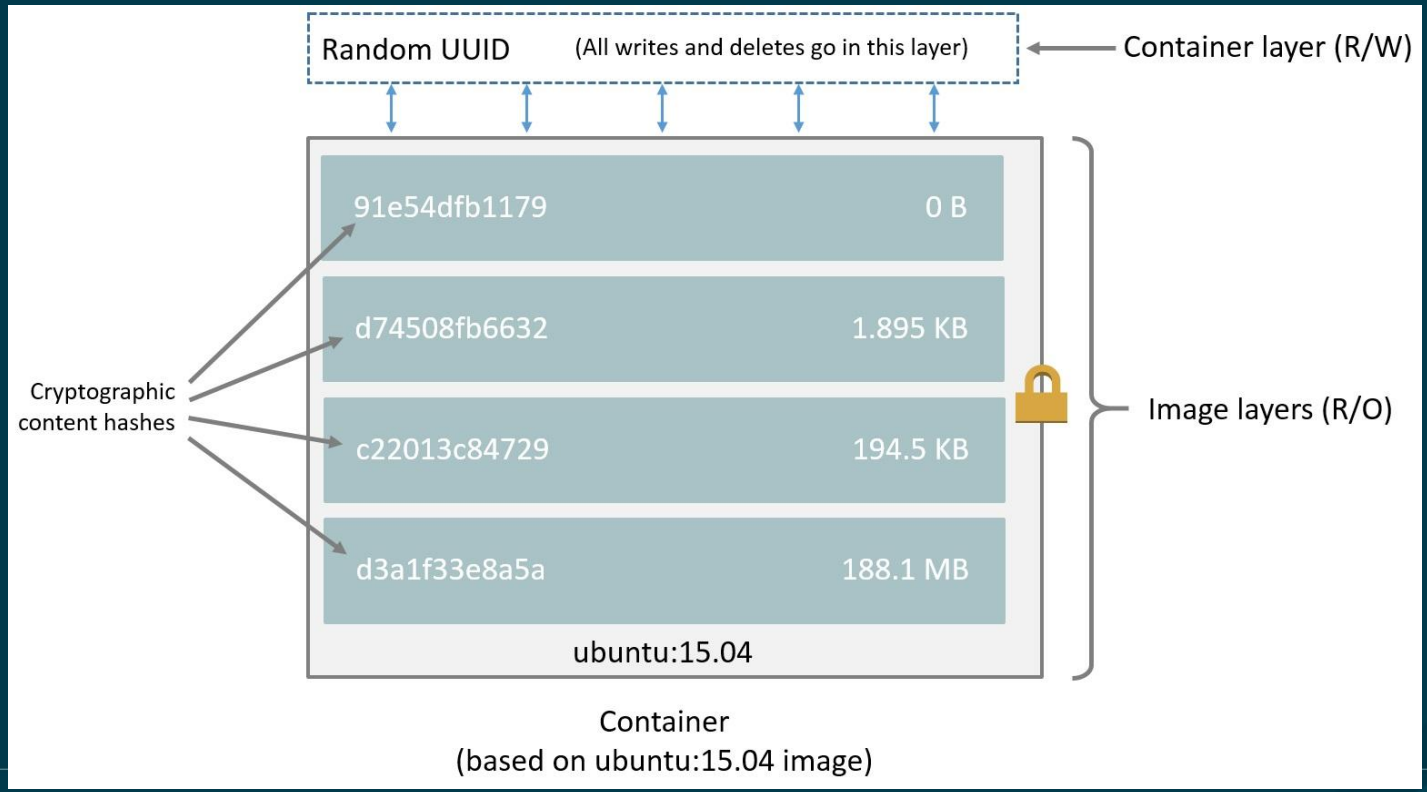
- Cpu share
- Cpuset
- Memory allocation
  - Soft vs. hard limits
- I/O
- Devices cgroup
- Freezer group
- Accounting (memory page ~ 4kB)

Union filesystem & Copy on Write

Killer feature for usable container

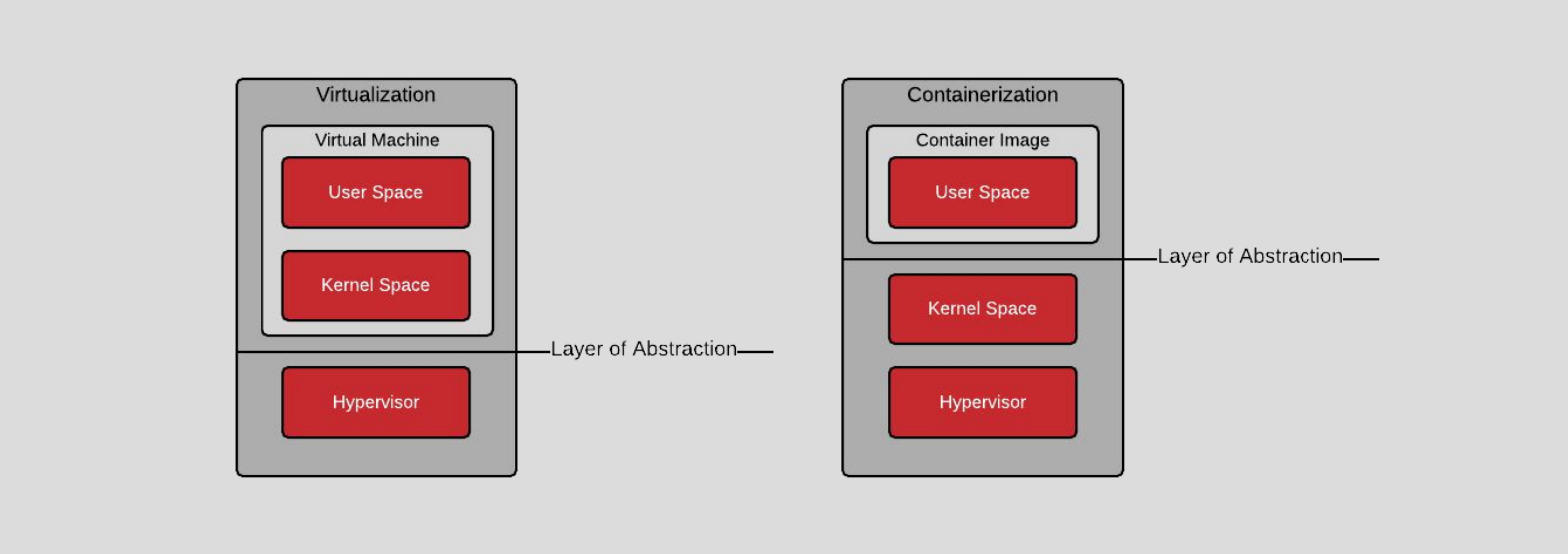
- Union filesystem
  - Image is a set of layers
  - Reusing/combining layers is efficient
- Copy on write (CoW)
  - Container is an image and a thin writable layer (container layer)
  - Fast spawning/deleting container
  - Deleting container = deleting only a layer

Union filesystem & Copy on Write II

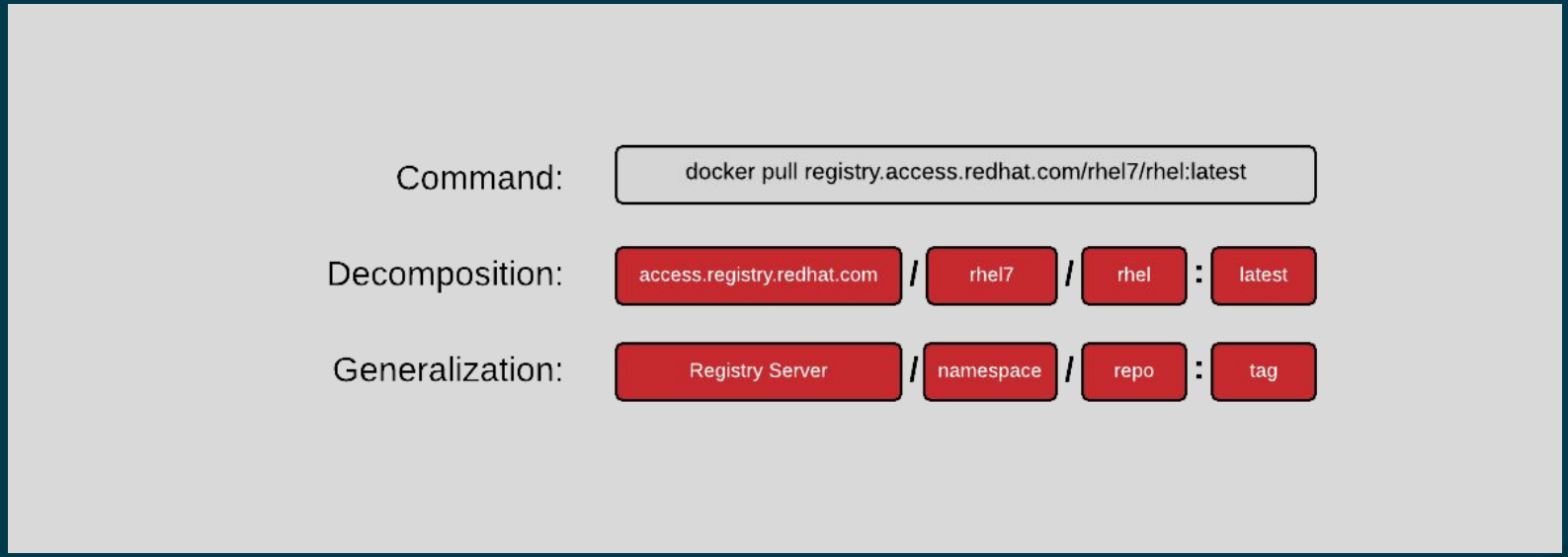


Source: https://docs.docker.com/engine/userguide/storagedriver/imagesandcontainers/

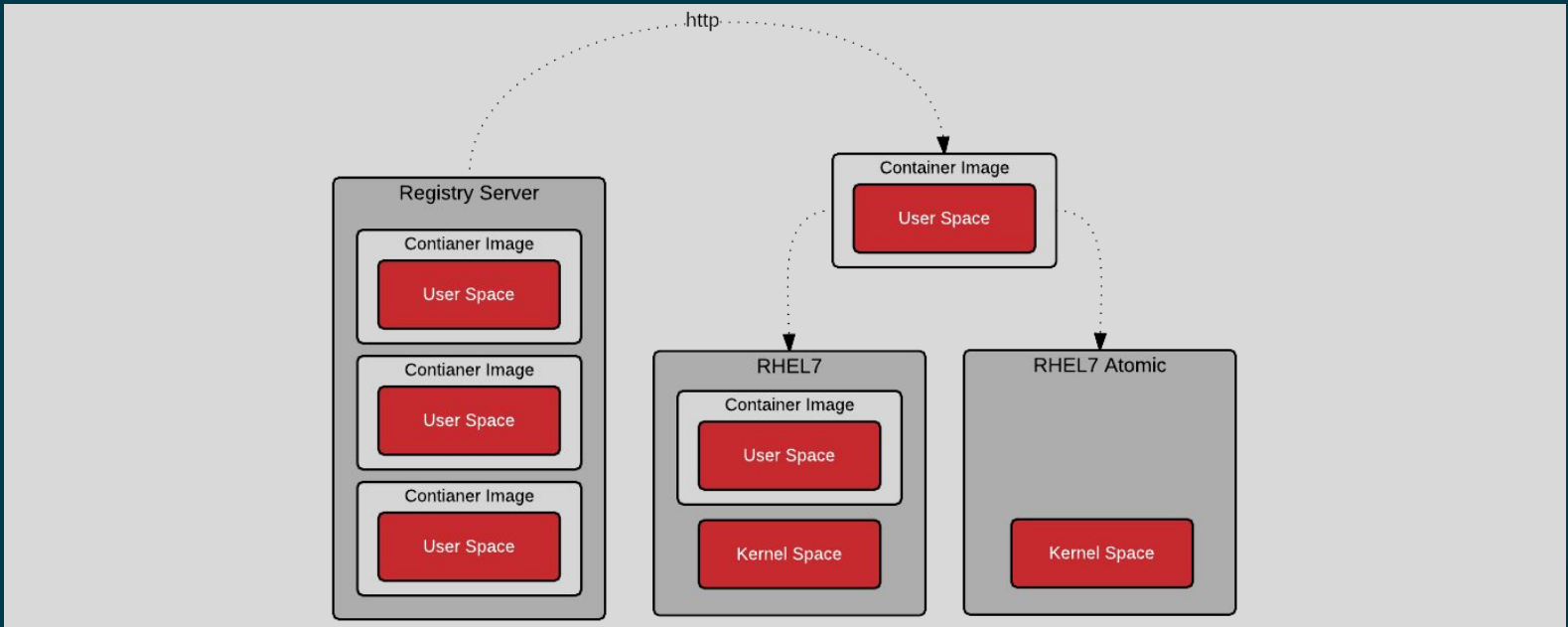
Containers are different from virtualization



Docker container image



Registry infrastructure



Demo time!

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos