# Linux Containers & Kubernetes

A systems integration lecture

Josef Karasek
Quality Engineer, Red Hat Middleware

2017-03-03

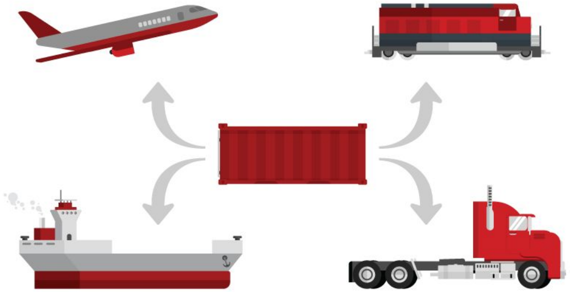
# Agenda

- Linux containers
    - Containers as a packaging mechanism
    - Containers as process isolation
- Kubernetes
    - Basic concepts
    - Pods
    - Services

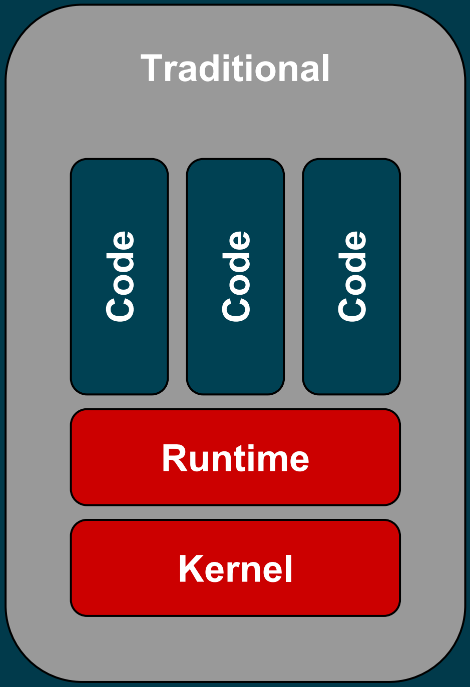
# Linux containers

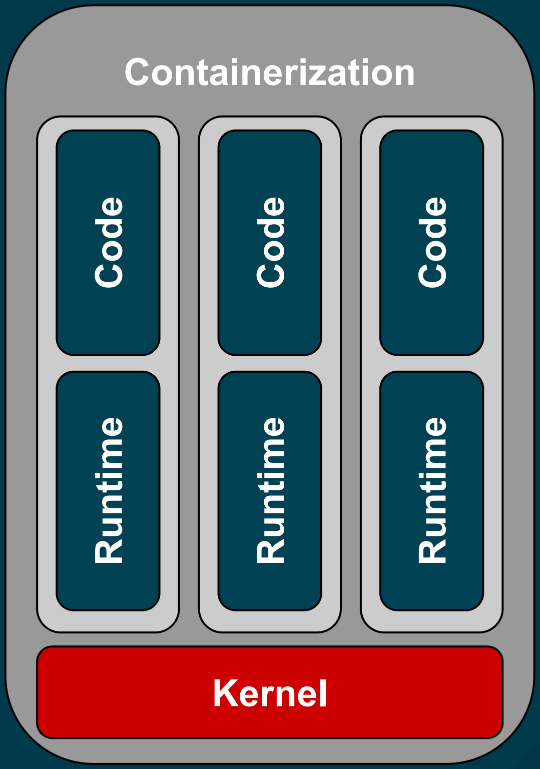
# Containers as a packaging mechanism

- 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 | | |

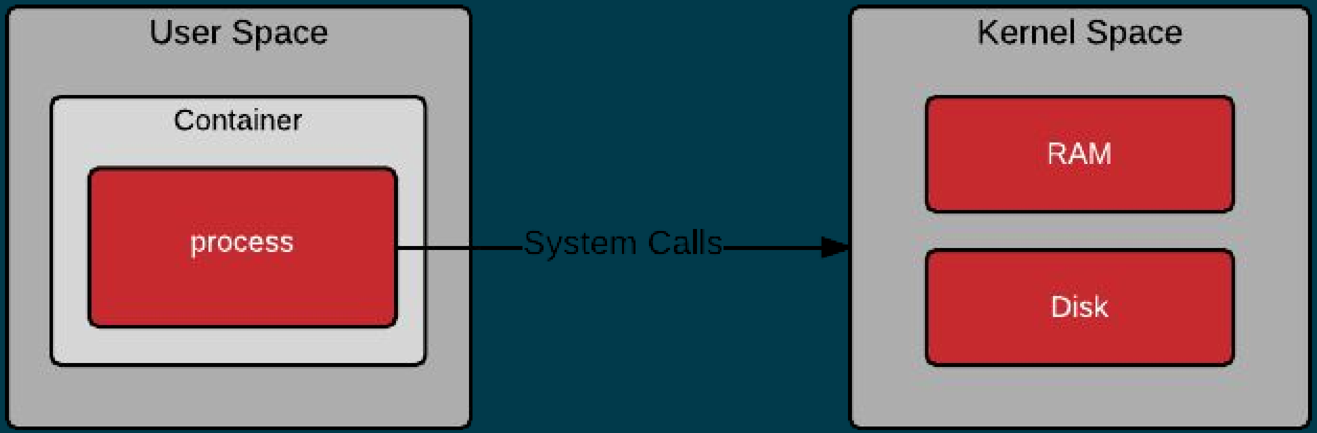
```
$ ldd /usr/libexec/mysqld
    linux-vdso.so.1
    libsystemd.so.0 => /lib64/libsystemd.so.0
    libpthread.so.0 => /lib64/libpthread.so.0
...
```

# Containers as process isolation

A container is just a fancy process

Namespaces - restrict what resources the container can use

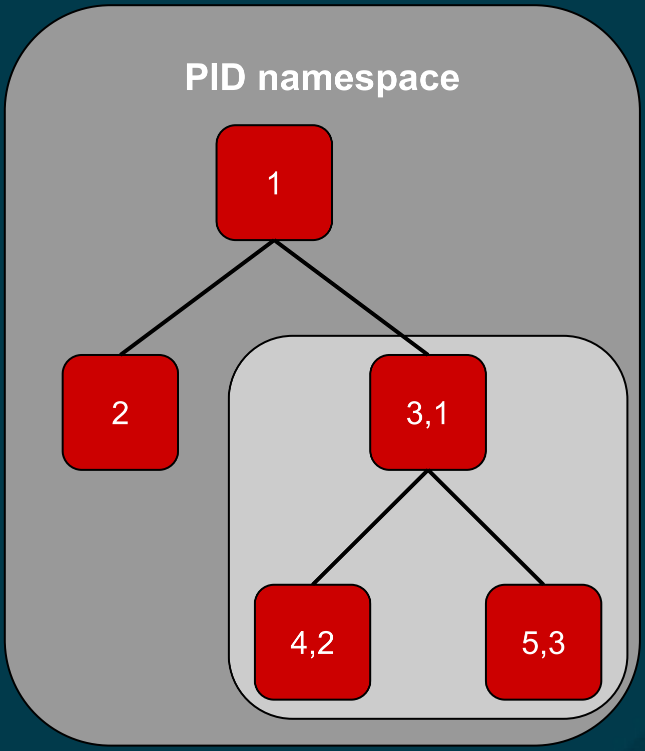
CGroups - define how much of a resource the container can use



# 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

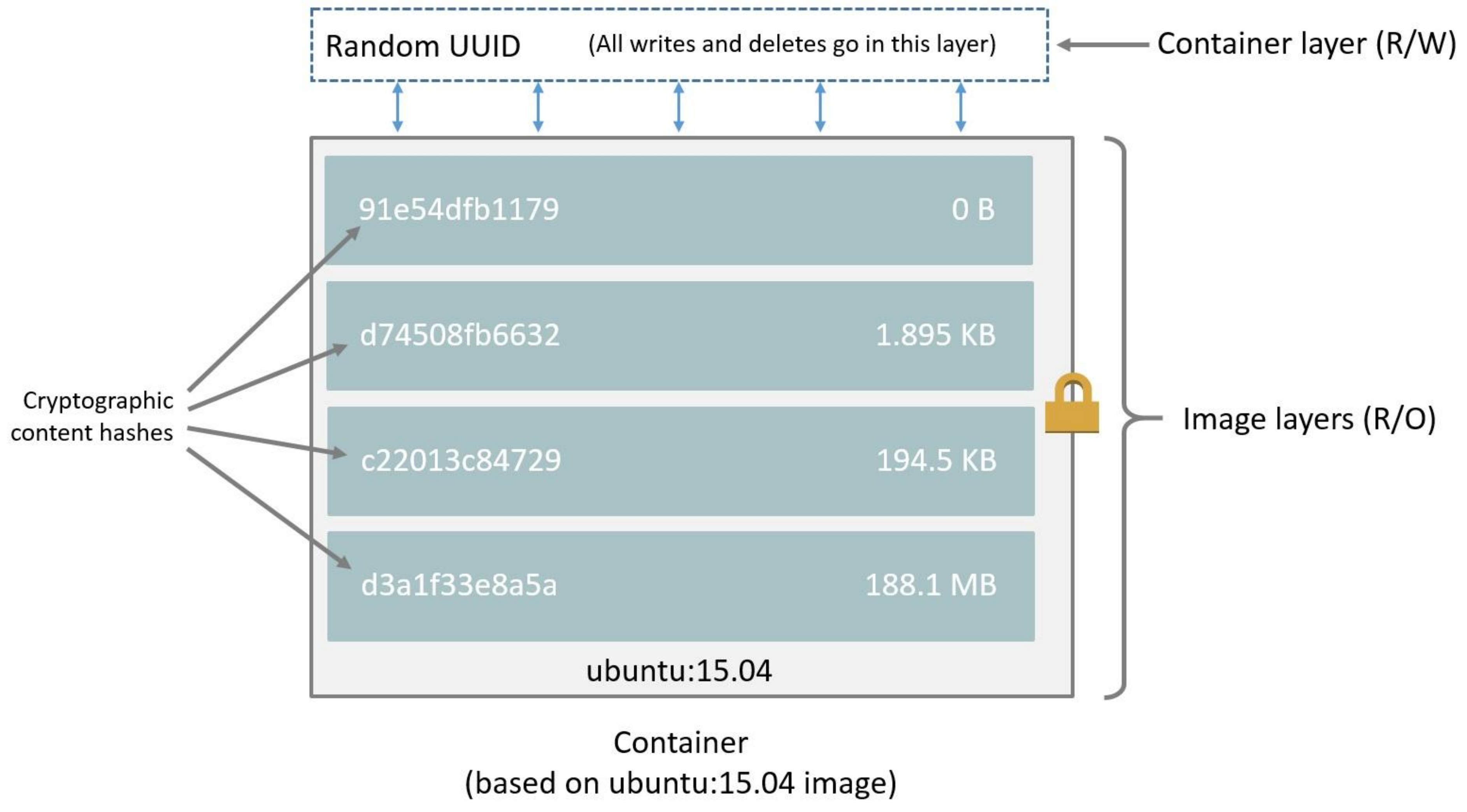
- Cpu share
- Cpuset
- Memory allocation
    - Soft vs. hard limits
- I/O
- Devices cgroup
- Freezer group
- Accounting (memory page ~ 4kB)

# Union filesystem & Copy on Write

Killer feature for usable container

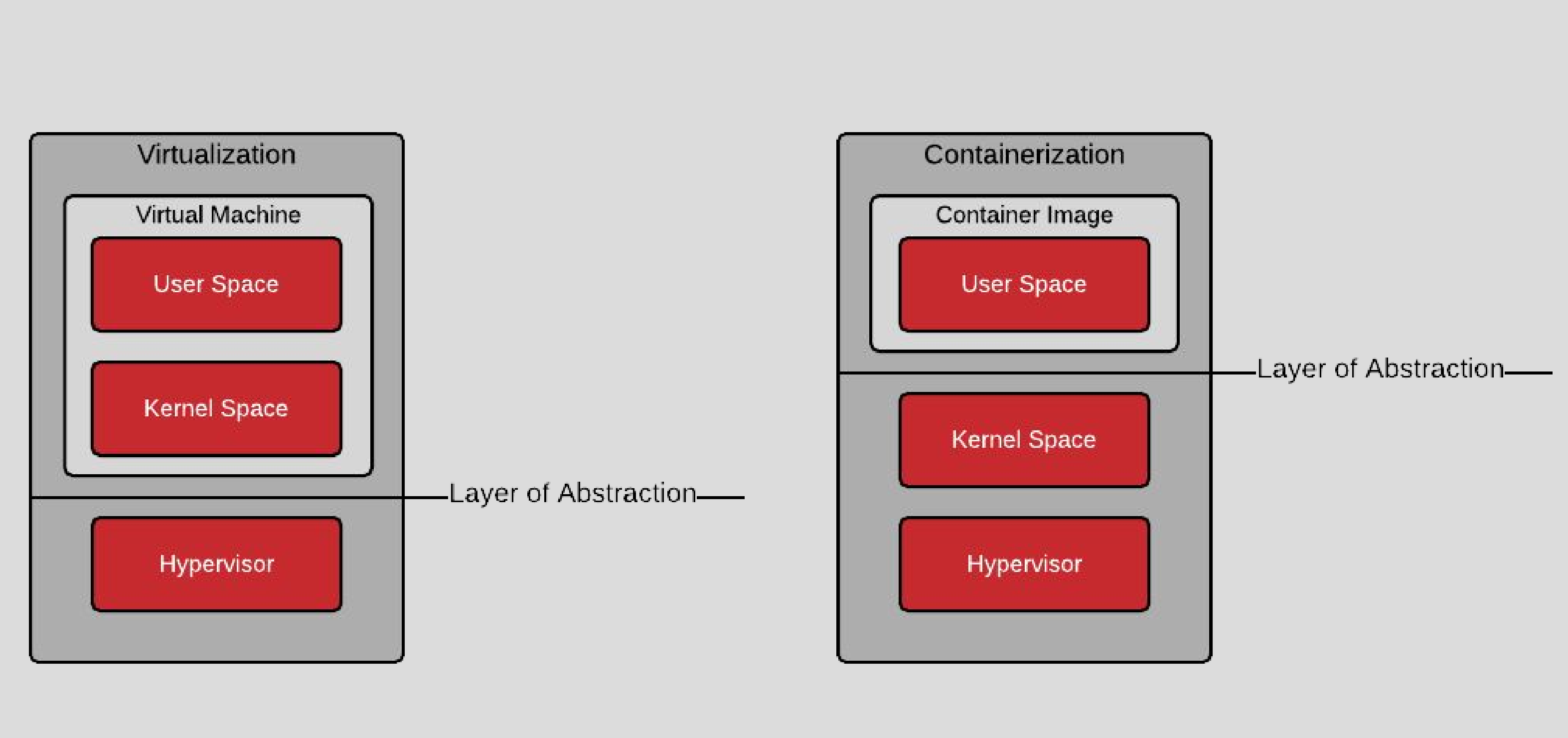
- Union filesystem
  - Image is a set of layers
  - Reusing/combining layers is efficient
- Copy on write (CoW)
  - Container is an image and a thin writable layer (container layer)
  - Fast spawning/deleting container
  - Deleting container = deleting only a layer

# Union filesystem & Copy on Write II

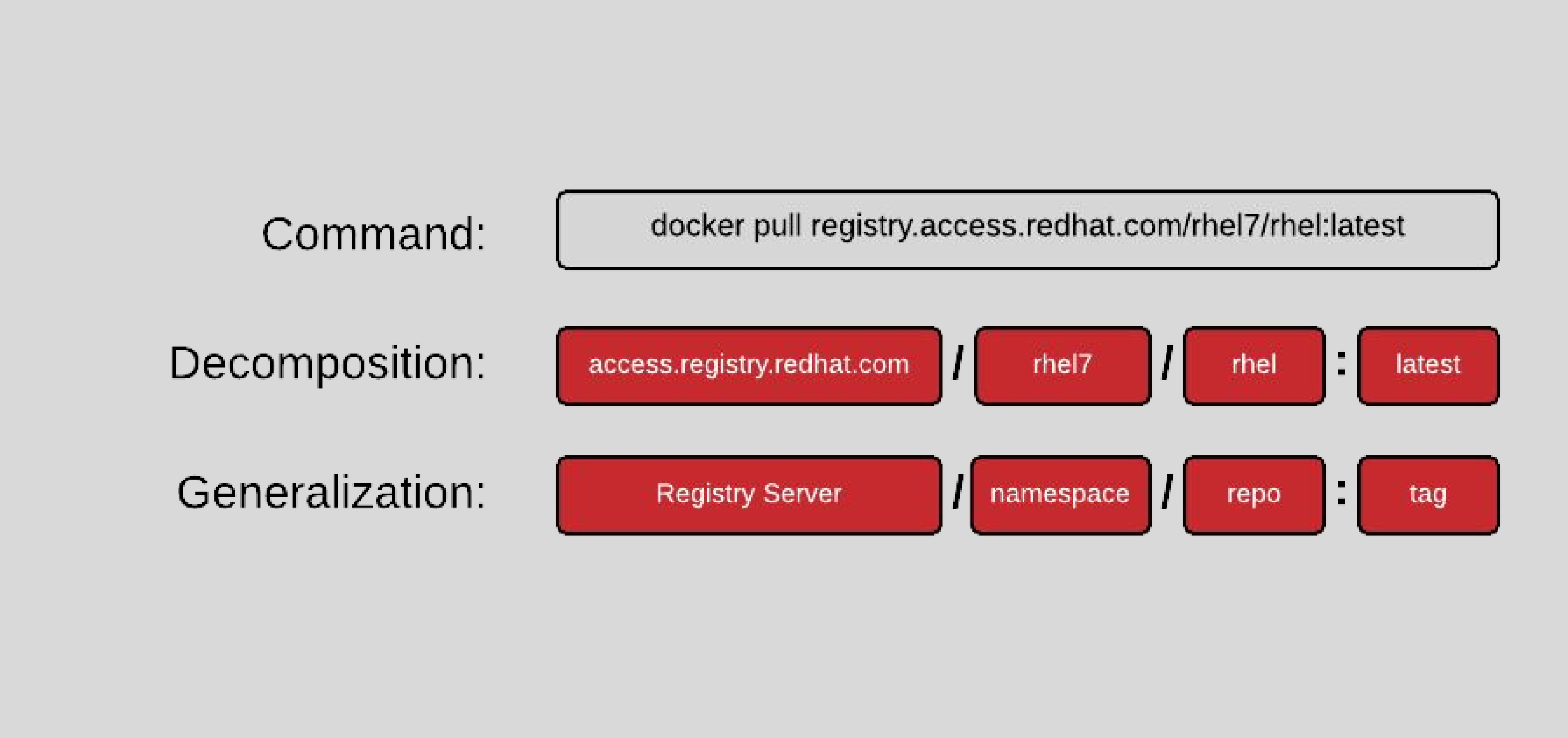


Source: https://docs.docker.com/engine/userguide/storagedriver/imagesandcontainers/

# Containers are different from virtualization



# Docker container image

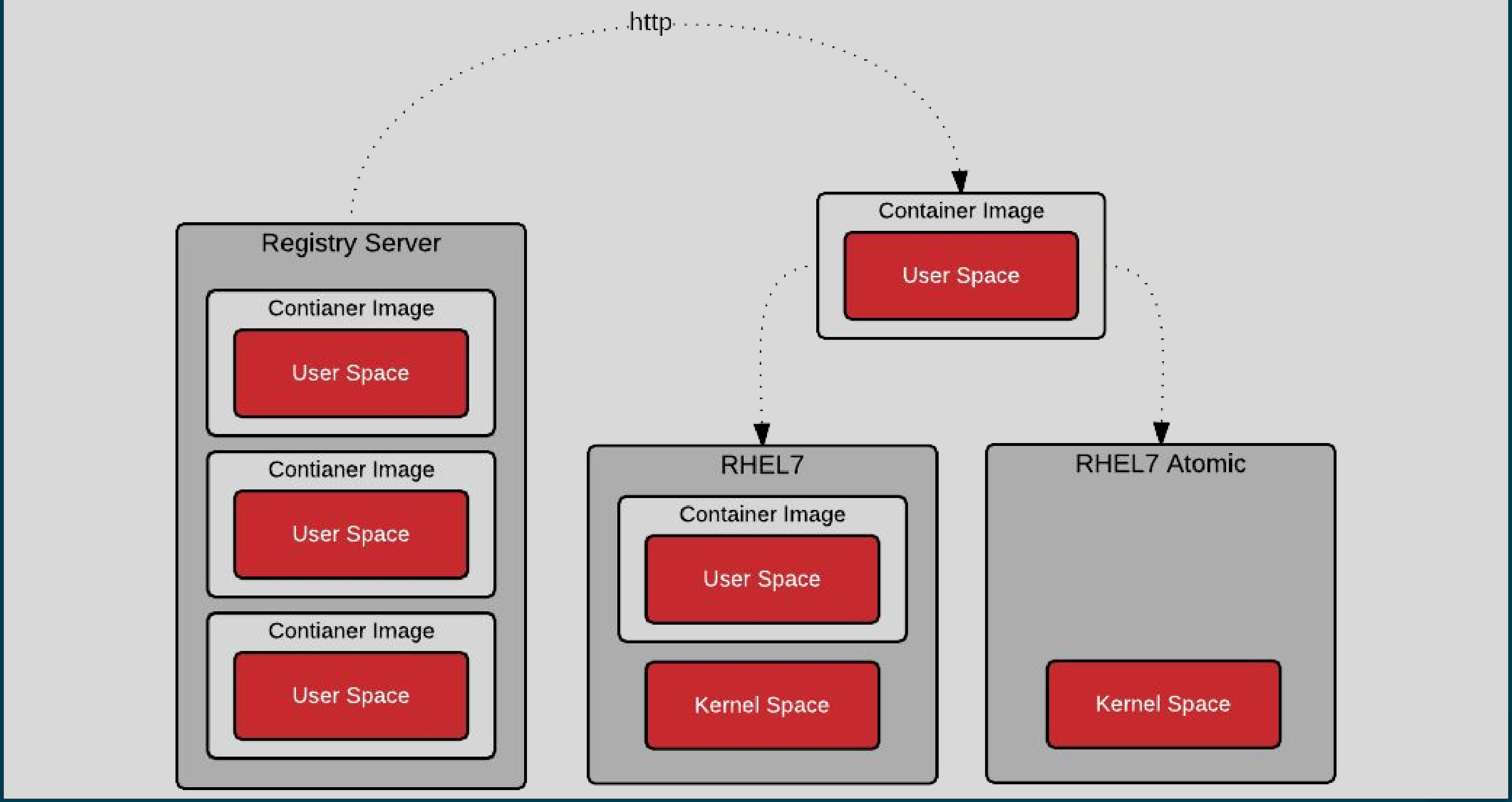


Command:

```
docker pull registry.access.redhat.com/rhel7/rhel:latest
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Decomposition: | access.registry.redhat.com | / | rhel7 | / | rhel | : | latest |
| Generalization: | Registry Server | / | namespace | / | repo | : | tag |

# Registry infrastructure



# Demo time!

# Kubernetes

# Kubernetes

The Cluster Manager for containers | Greek word for ‘helmsman’

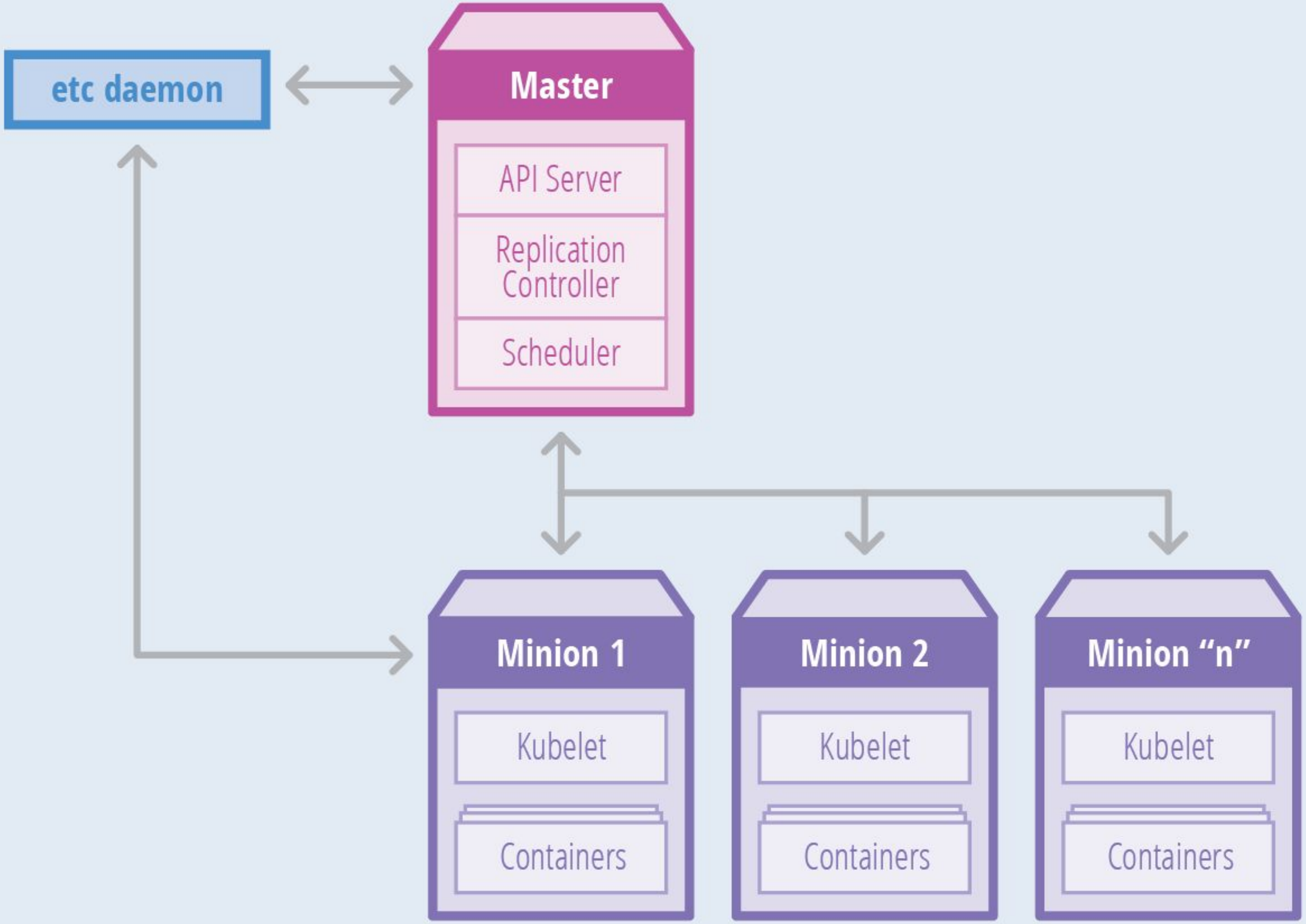
- Deploy
- Discover
- Scale
- ...and manage services
- Self-healing platform - designed for failure
    - Services fail, often.
    - Monitor service health
    - Automatic re-schedule
- Based on ideas provet at Google over 10 years
- Written in Go

# Architecture

Master servers & nodes

- API server
- Scheduler
  - Selects hosts & deploys containers
- Node agents - node controller (kubelet)
  - On every node, spawns containers
- Cluster state backed by a distributed storage system - etcd
- Container runtime
  - cri-o, docker, rkt
- kube-proxy
  - Service discovery

# Kubernetes: Building on Architectural Roots



Source: The New Stack.

THENEWSTACK

Source https://thenewstack.io/containers-container-orchestration/

# Kubernetes key concepts

- Pods
- Services
- Replication controller

# Pods

A group of containers

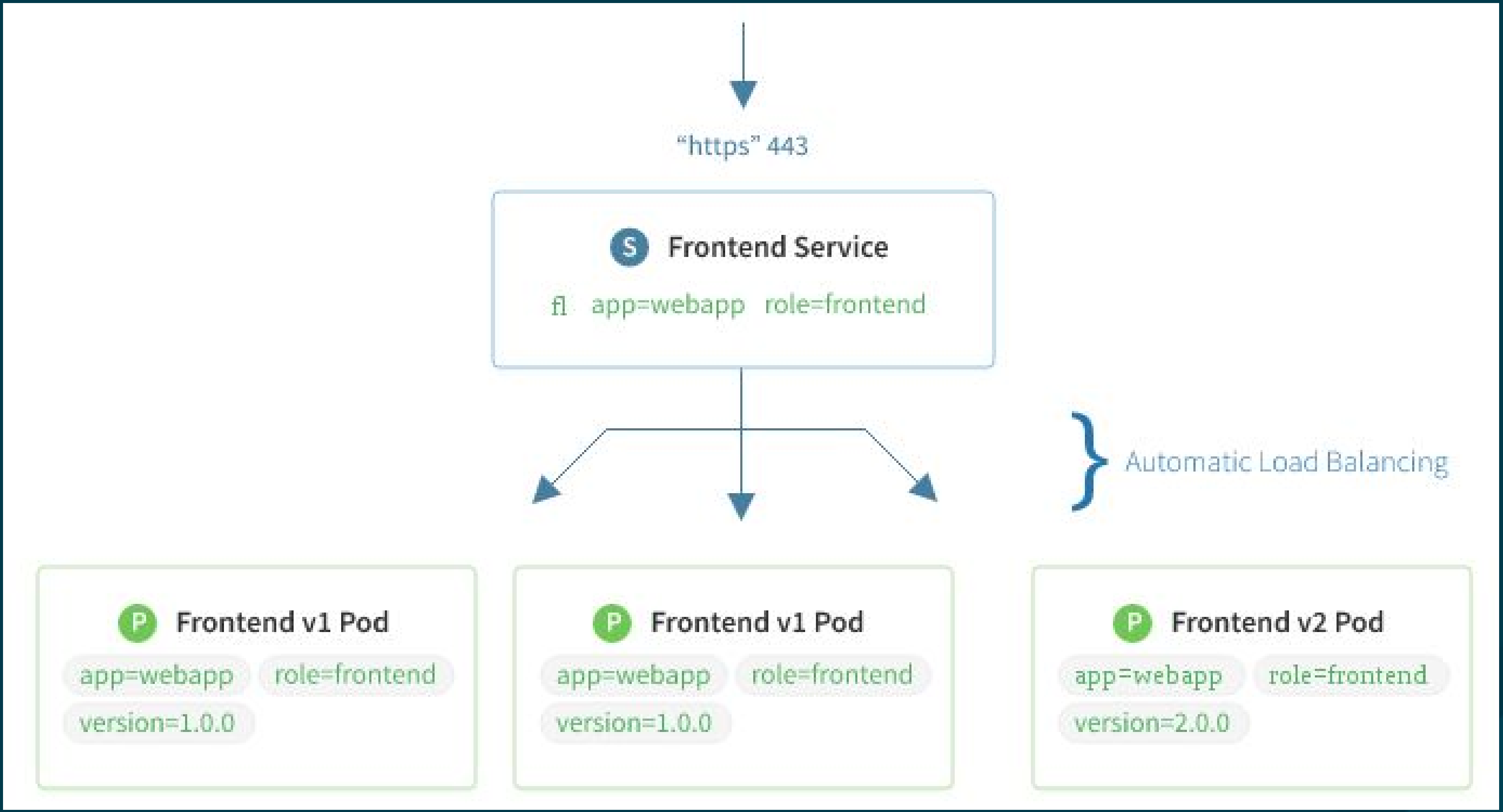
- Always scheduled together
- Share the same network namespace (localhost)
- …and all other ns’s, cgroup properties

# Services

## Cluster service discovery

- Logical bindings between containers
- Labels and Label selector
- Actual IP addresses of containers can change

# Services II


"https" 443
S Frontend Service
fl app=webapp role=frontend
Automatic Load Balancing
P Frontend v1 Pod
app=webapp role=frontend
version=1.0.0
P Frontend v1 Pod
app=webapp role=frontend
version=1.0.0
P Frontend v2 Pod
app=webapp role=frontend
version=2.0.0

# Replication controllers

Replicas = multiple copies of a pod

- Ensures that a specified number of pod replicas are running
- Monitor pod health
- Re-schedule pods upon error

# More k8s concepts

- Namespaces
- Readiness & Liveness probes

# Namespaces

Logically called groups

- Cluster used by multiple users/groups of users
- Unique per user:
    - Resources (pods, services, replication controllers, etc)
    - Policies (who can and cannot)
    - Constraints (quotas)

# Health checks

Periodically check pod status

- Readiness probe
  - Mark starting pod as ‘schedulable’ only when it’s ready
- Liveness probe
  - Application code fail
  - Hardware fails
  - Electricity…
  - Pods are very ephemeral - reschedules are very common

redhat.

# THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos