

RED HAT
SUMMIT

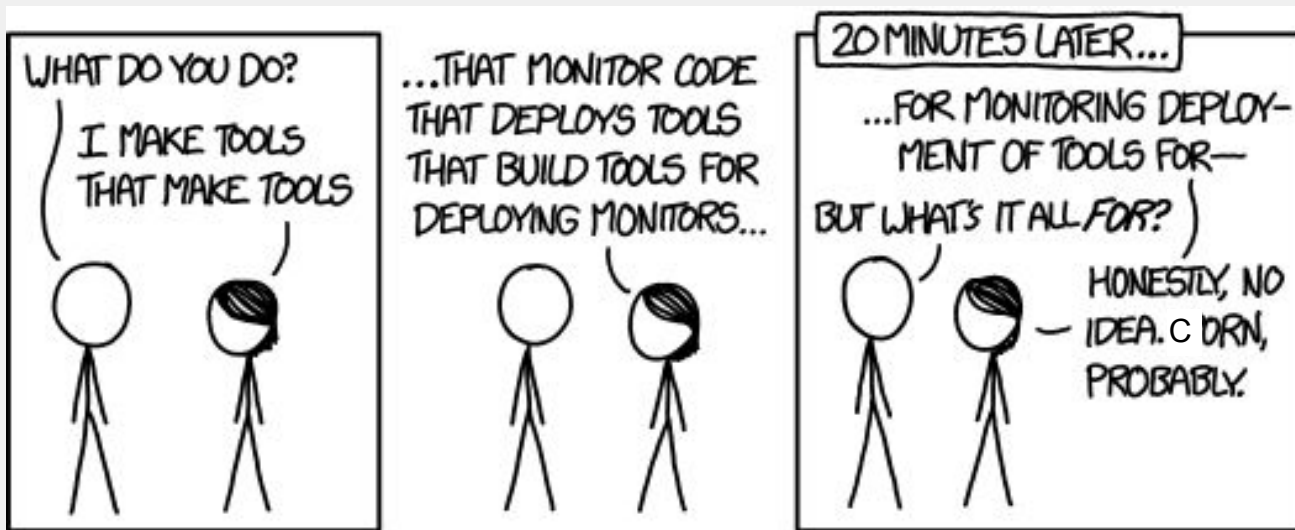
Building Production-Ready Containers

Scott McCarty, RHCA
Product Manager - Linux Containers

Ben Breard, RHCA (EOL 2018)
Product Manager - Linux Containers



Containers Make Things Easy - Right? :-P



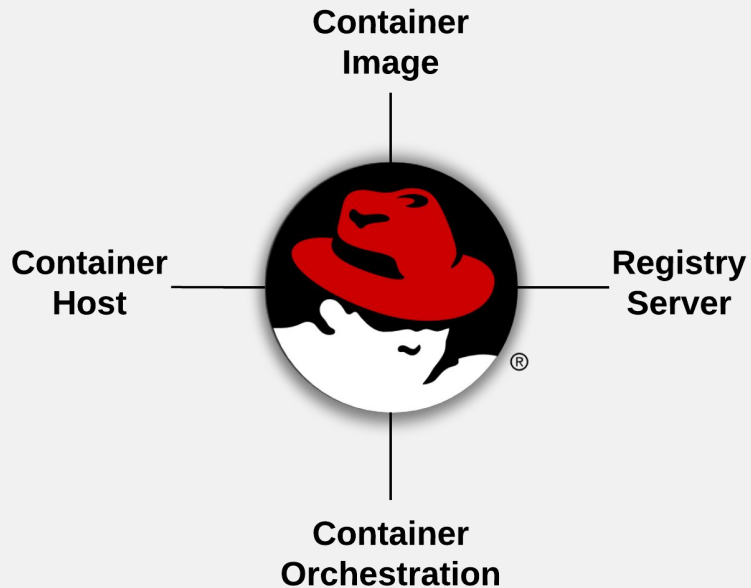
Agenda

- Capabilities, Problems, and Trade offs
- OCI Image Fundamentals
- Implications & Common Obstacles
 - (And how to overcome them!)
- Tips & Tricks
- Putting It All Together

CAPABILITIES, CHALLENGES, AND TRADE OFFS

Production-Ready Containers

What are the building blocks you need to think about?



Production-Ready Containers

What are the building blocks you need to think about?

1. Container Images
2. Orchestration Definitions
3. Delivery - Registries & Source Control

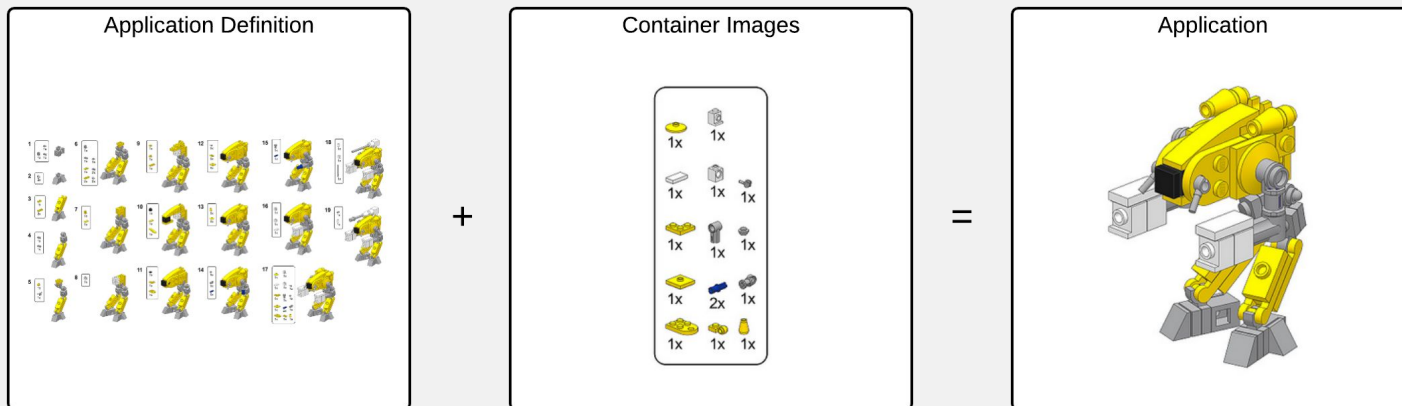
```
apiVersion: v1
kind: ReplicationController
metadata:
  name: mysql
  labels:
    name: mysql
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: mysql
    spec:
      containers:
        - name: mysql
          image: openshift3/mysql-55-rhel7
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: pizza
            - name: MYSQL_USER
              value: pizza
            - name: MYSQL_PASSWORD
              value: pizza
```

Mindset

“Using containers is as much of a business advantage as a technical one. When building and using containers, layering is crucial. You need to look at your application and think about each of the pieces and how they work together—similar to the way you can break up a program into a series of classes and functions.” - Ryan Hallisey

Application Delivery

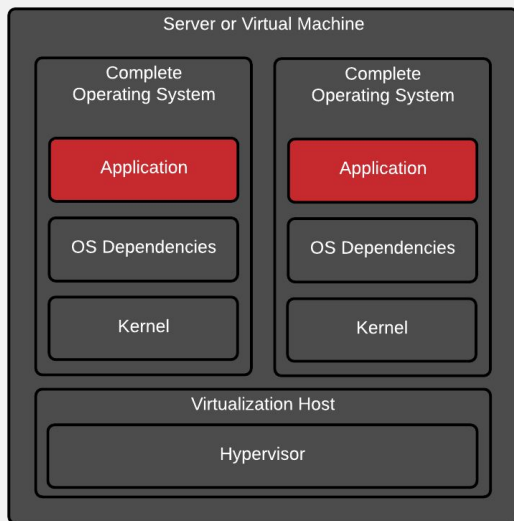
Container images, assembly instructions, and resource requirements



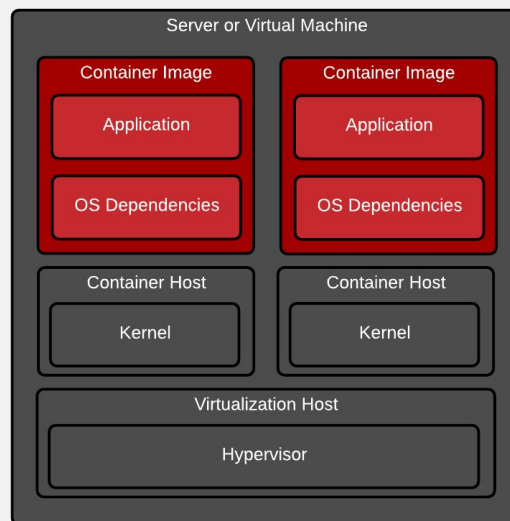
OCI Image Fundamentals

Container Images



Virtual machines and container environments



Application & Infrastructure
Updates Tightly Coupled



Application & Infrastructure
Updates Loosely Coupled

-  Optimized for agility
-  Optimized for stability

OVERVIEW OF THE DIFFERENT STANDARDS

Vendor, Community, and Standards Body driven



kubernetes



CNI

Open Containers Initiative (OCI)
Image Specification

Open Containers Initiative (OCI)
Distribution Specification

Open Containers Initiative (OCI)
Runtime Specification

Container Runtime Interface
(CRI)

Container Network Interface
(CNI)

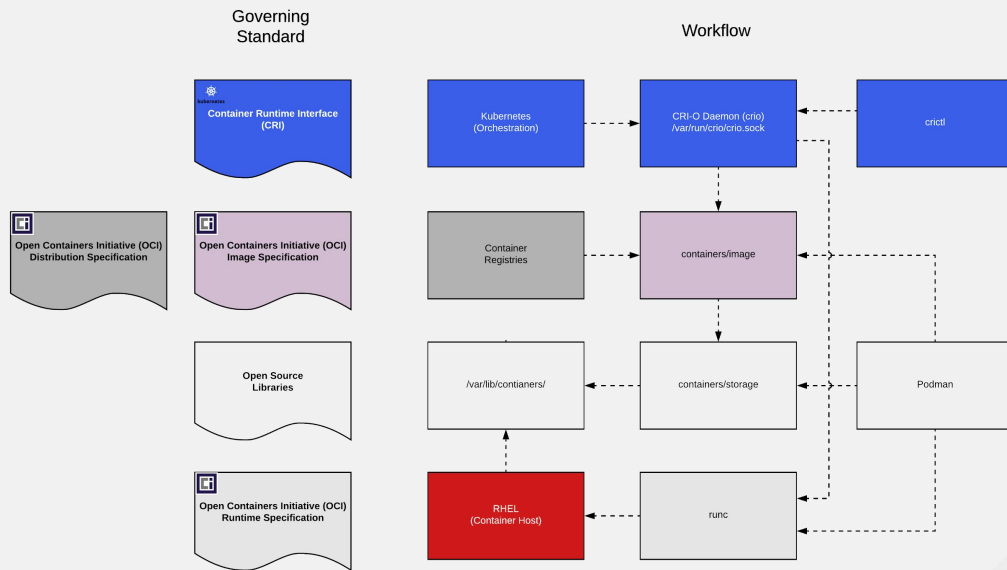
Many different standards

WORKING TOGETHER

Technical example

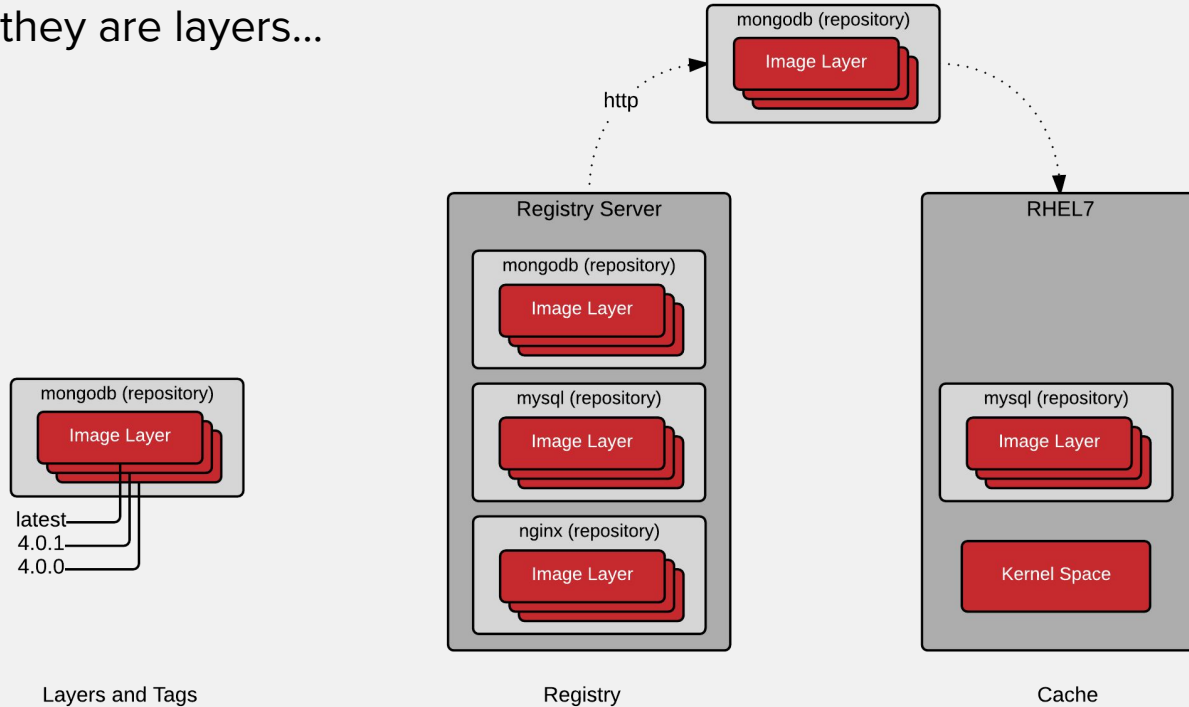
Different standards are focused on different parts of the stack.

- Tools like crictl use the CRI standard
- Tools like Podman use standard libraries
- Tools like runc are widely used



Fancy Files

Actually, they are layers...



Fancy File Servers

Actually, they are repositories

Command:

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

Decomposition:

access.registry.redhat.com

/

rhel7

/

rhel

:

latest

Generalization:

Registry Server

/

namespace

/

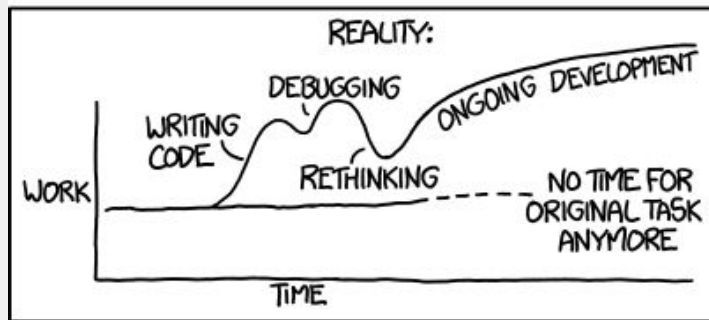
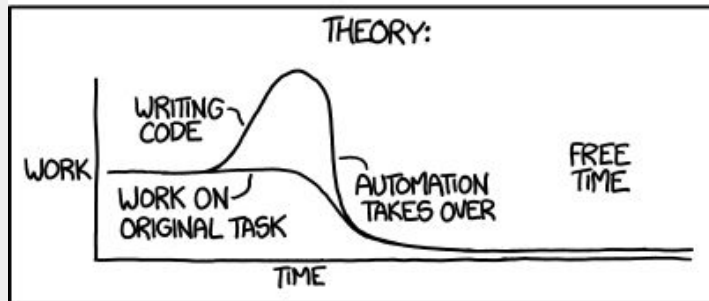
repo

:

tag

Another Hilarious XKCD Slide

"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"



The Tenets of Building

Rules

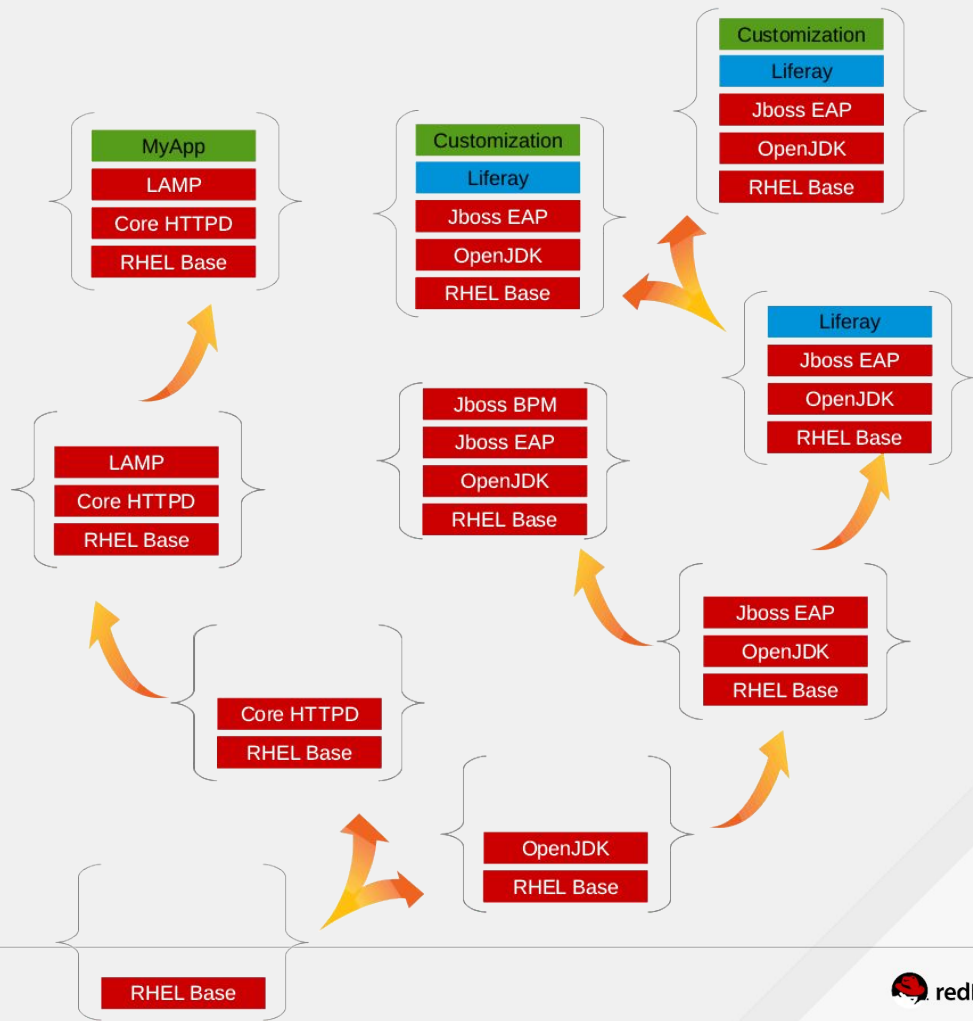
Foundational to all of these rules is **source control for everything - treat all of the artifacts as buildable from code**

- **Standardize**
- **Minimize**
- **Delegate**
- **Process**
- **Iterate**

Rule: Standardize

Goal: Publish a standard set of images with common lineage

- **Base image(s)**
 - Application Frameworks
 - Application Servers
 - Databases
 - Etc
- **Benefits:**
 - Easier scale
 - Maximize reuse of common layers
 - Limit environment anomalies





buildah

Rule: Minimize

Goal: Limit the content in the image to what serves the workload

- FROM rhel7-atomic
- buildah can populate images with tools from the host.
- Clearing package manager cache

Benefit:

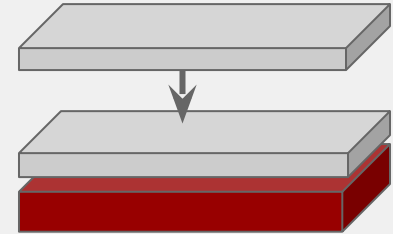
- Smaller attack / patching surface
- More efficient push/pulls

Warning: taking this to the extreme will negate layer sharing and not have the intended effect

Start from an existing image or from scratch



Generate new layers and/or run commands on existing layers



Commit storage and generate the image manifest



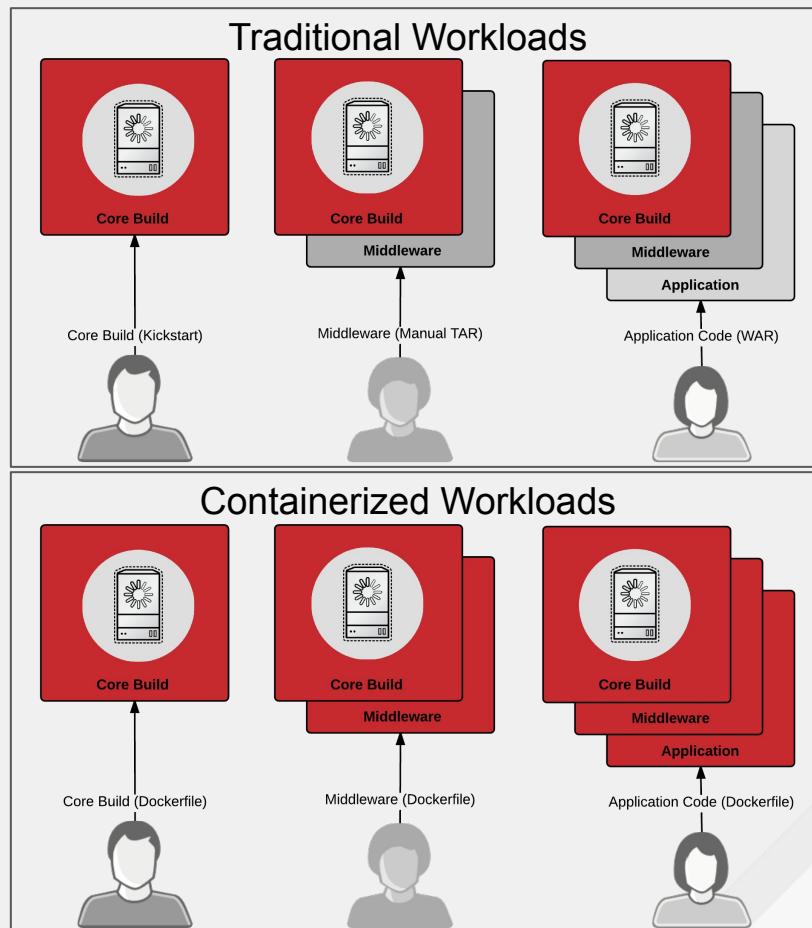
Deliver image to a local store or remote OCI / docker registry



Rule: Delegate

Goal: Ownership needs to lie with expertise

Benefit: Leverage your teams on the part of the stack they know best

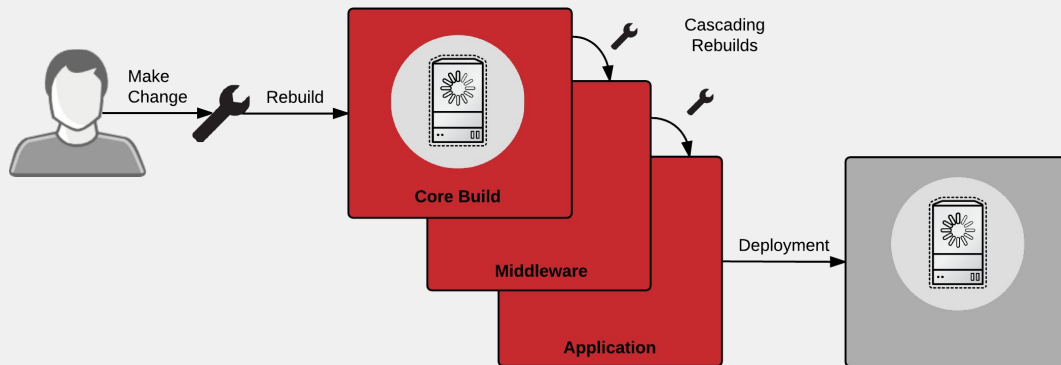


Rule: Focus on Process and Automation

Goal: Automate rebuilds of all objects

- Testing (CI, performance, etc)
- Security
- Deployments

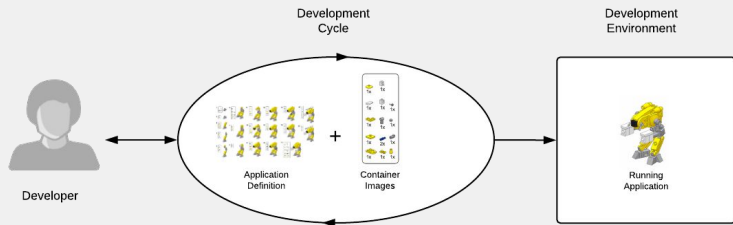
Benefits: Fast redeployment as you make changes to the environment



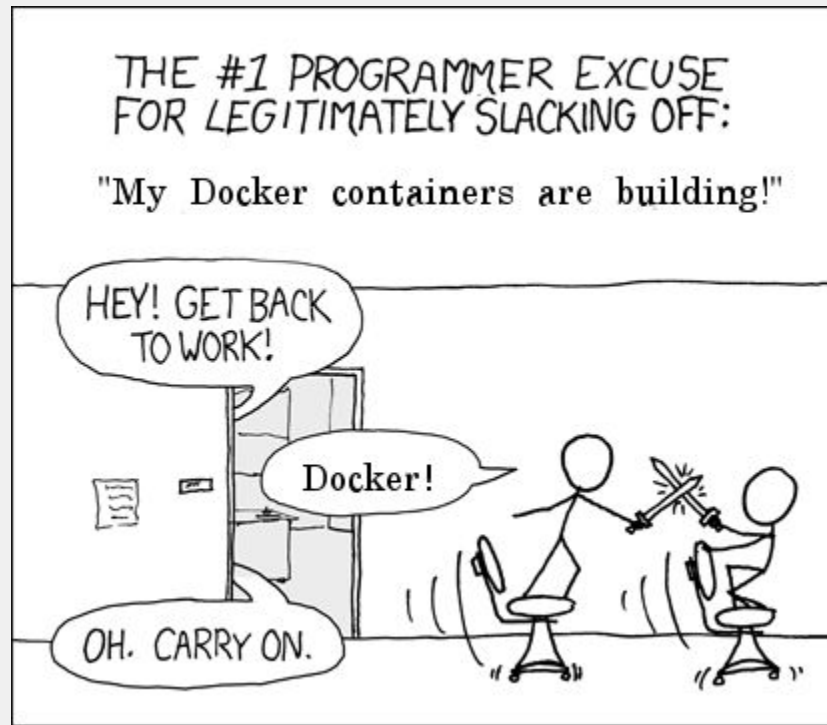
Rule: Iterate

Goal: DON'T REPEAT THE MISTAKES OF THE PAST!!!!

Benefit: Leverage the expertise of your teams on the part of the stack they know best. Capture it in code. Knowledge is temporal



3 in a row!

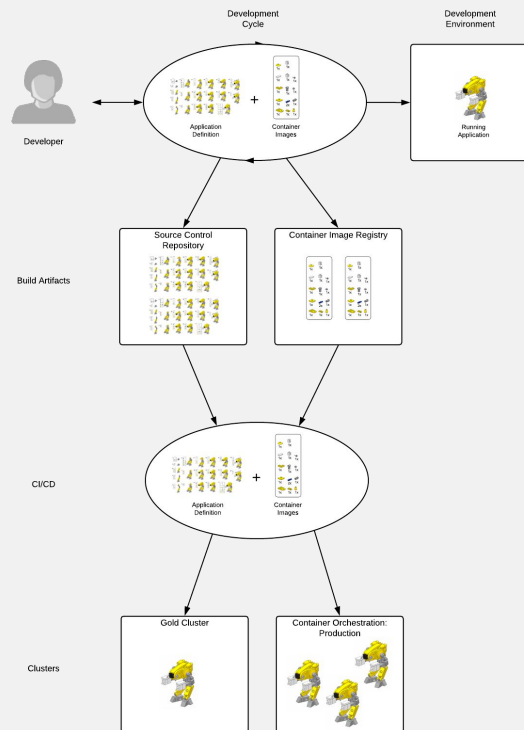


Putting it All Together

Building Production-Ready Containers

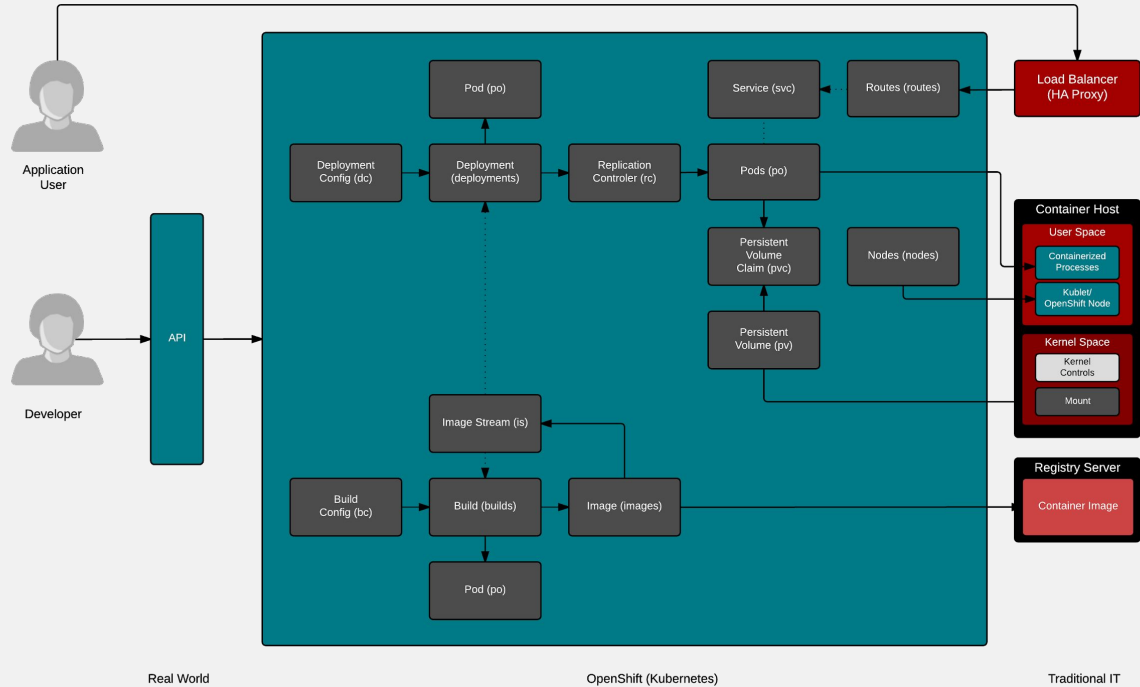
Compatibility is a requirement for portability. We must ship the container images and application definitions between environments.

Image: Developer's laptop, development data centers, and cloud data centers



Assembly Instructions

The building blocks



RED HAT
SUMMIT

THANK YOU



plus.google.com/+RedHat



facebook.com/redhatinc



linkedin.com/company/red-hat



twitter.com/RedHat



youtube.com/user/RedHatVideos

What Challenges do Containers Really Solve?

In production?

True

- Better separation of concerns between developers, operations, database administrators, middleware specialists, etc
- Compatibility and portability still need to be planned for.
- Developers and operations need a mix of new and existing skills
- Better definitions of applications & sub-components
- Truly distributed systems environment

False

- Everybody can do whatever they want. Developers will just do everything themselves. We no longer need specialists.
- Complete portability - build once, run anywhere. I...mean...anywhere
- Containers are easy. Developers just use them, don't worry...
- You must completely break your application up
- Forget everything you know, this is magic