Repeatable process for building secure containers

Jorge Morales
OpenShift Field Product Manager &
Developer advocate

Agenda

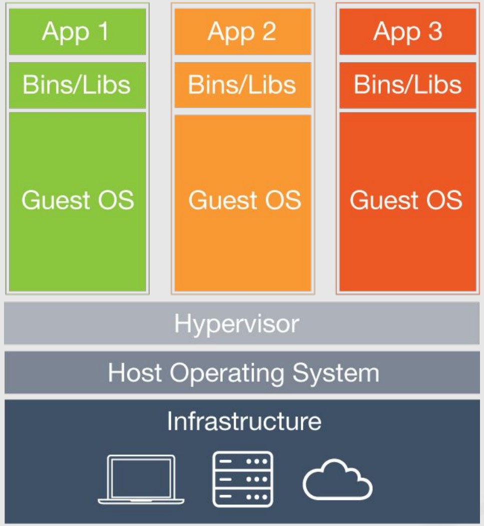
- Technology introduction
- OpenShift 3 architectural overview
- Security in Docker
- Security in OpenShift 3
- Build and deploy secure containers

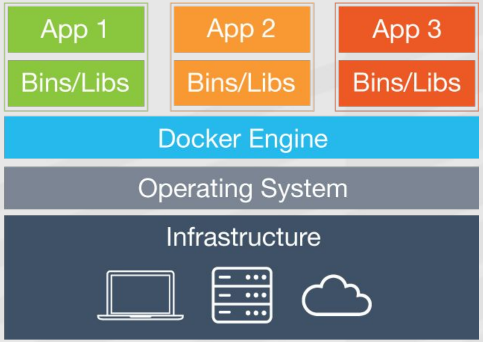
OpenShift 3

Technology introduction

What are containers?

Where **hypervisors** provide a logical abstraction of a full system (hardware, BIOS, OS), **Containers** provide an abstraction of the user space and share the same OS, services, and hardware.





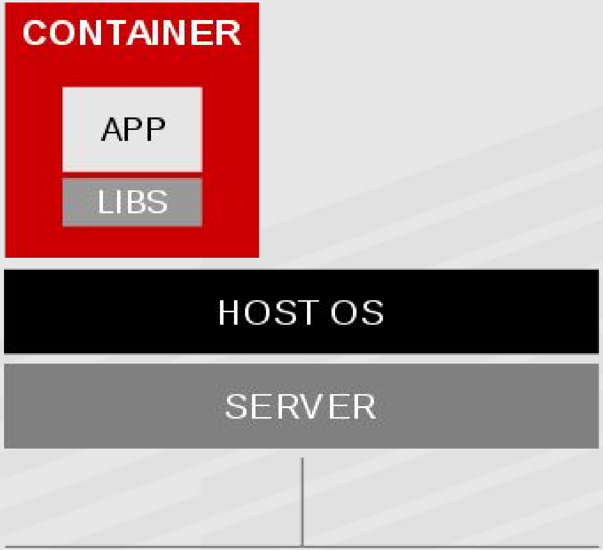
What are Linux Containers?

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**



Docker

- Container **packaging format**

“ Docker allows you to package an application with all of its dependencies into a standardized unit for software development. ”

- **Docker engine** is a set of tools to build and run containers (a **daemon runtime** and **cli**)
- **Registry** stores and distributes container images
- **Hub** is “marketplace” for containers

Docker is easy!!!

Kubernetes

- leverages Google's experience with Borg and Omega
- manages a fleet of Docker daemons
- provides coordination for components
- provides resiliency for containers
- provides high availability for containers

OpenShift 3

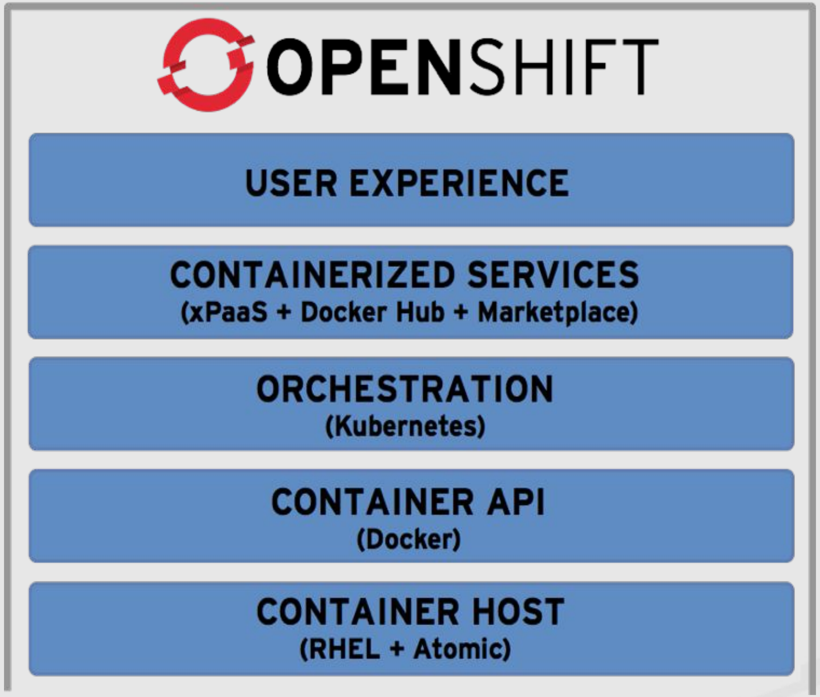
Architectural overview

PaaS

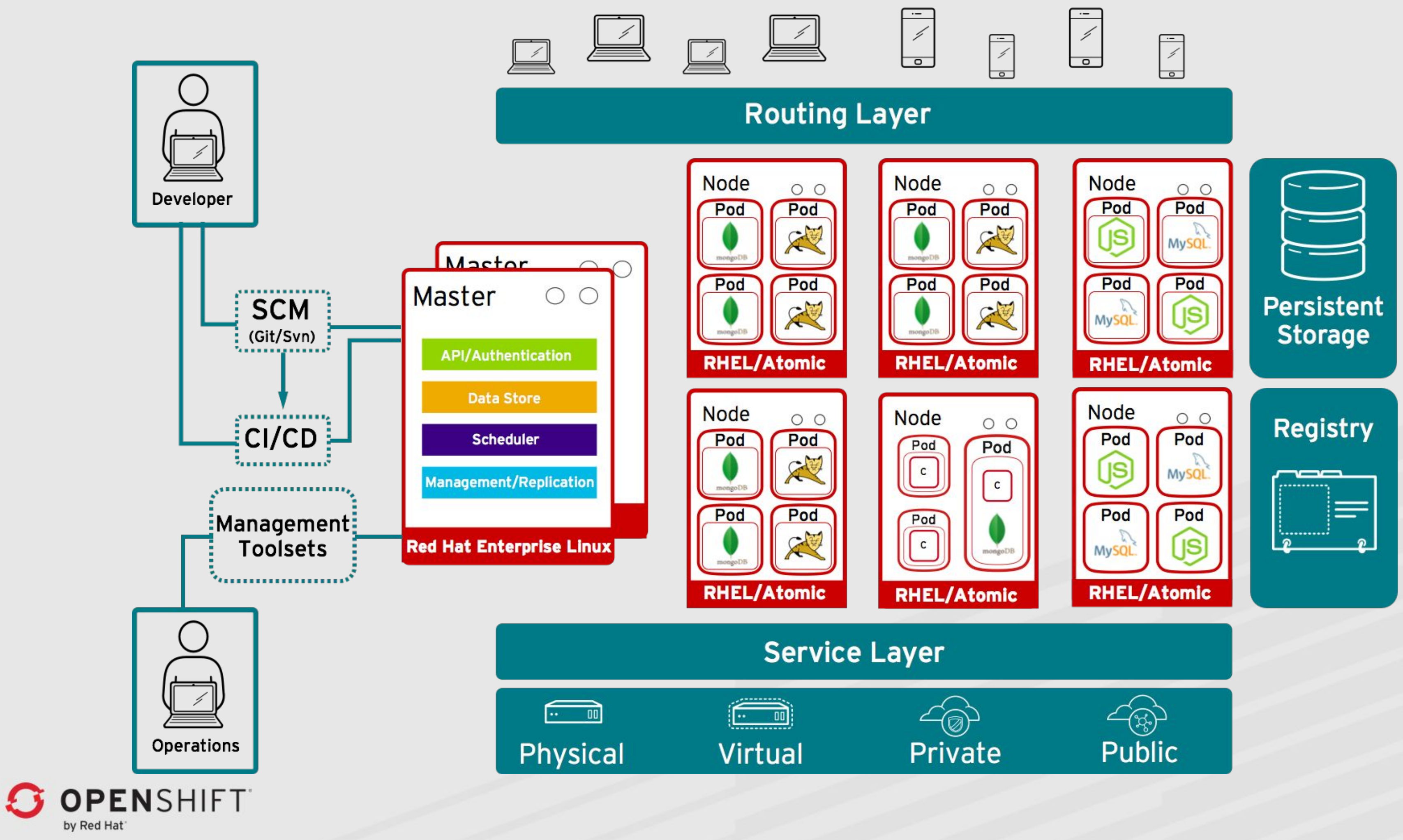
- You code the application, PaaS runs it for you
- Leverage the **ease**, **scale** and **power** of the Cloud

OPENSHIFT by Red Hat

OpenShift3

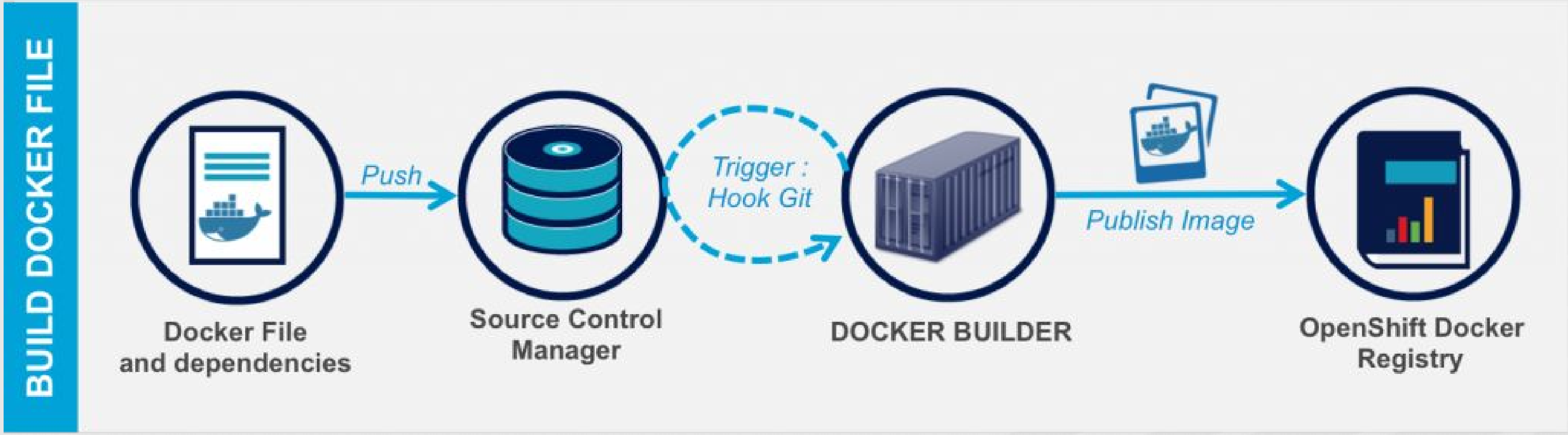


- Rich Web Console, CLI & IDE interfaces
- Multi-User Collaboration (Projects and Teams)
- Build Automation & Source-to-image
- Integration with Existing CI & Build Systems
- Deployment Automation & Regions / Zones
- OVS Container Networking
- Shared Storage Volumes
- Simplified Installation and Administration



Routing Layer
Developer
SCM
(Git/Svn)
CI/CD
Management Toolsets
Operations
Master
Master
API/Authentication
Data Store
Scheduler
Management/Replication
Red Hat Enterprise Linux
Node
Pod
RHEL/Atomic
Persistent Storage
Registry
Service Layer
Physical
Virtual
Private
Public
OPENSHIFT
by Red Hat

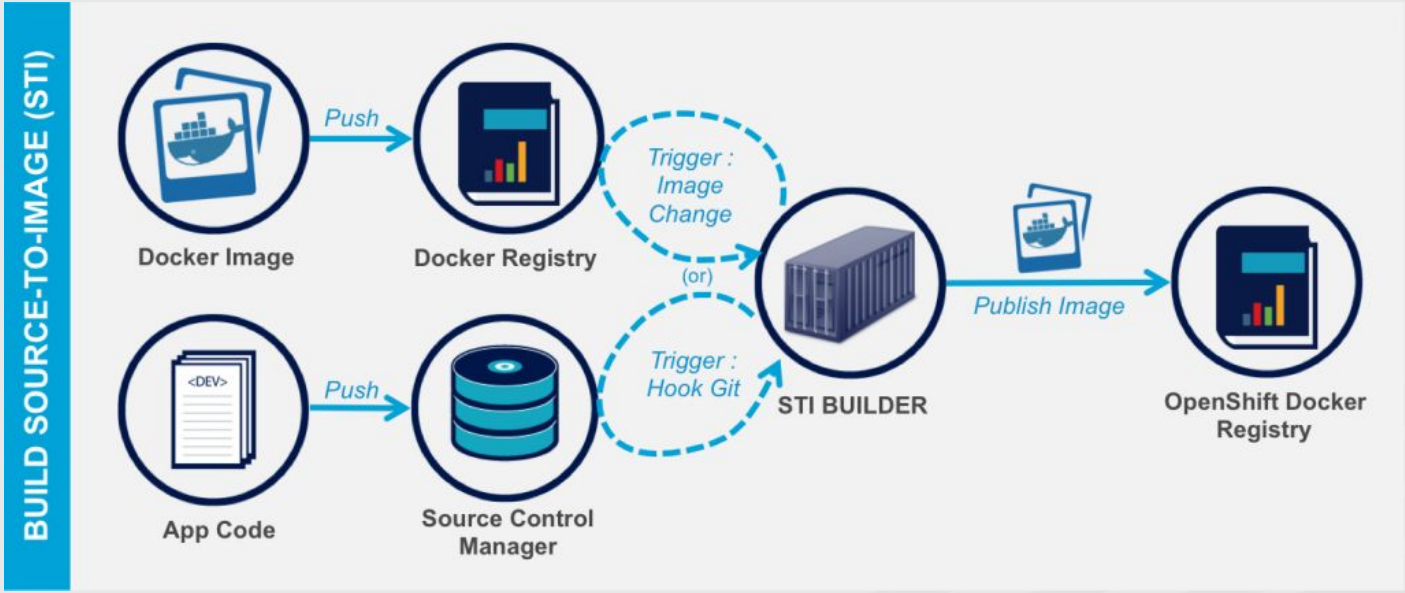
Building from Dockerfile



Traditional Docker-file method automatically builds containers by setting the SCM location into Openshift. This is a good non disruptive method for customer already using Docker Images.

OPENSHIFT by Red Hat

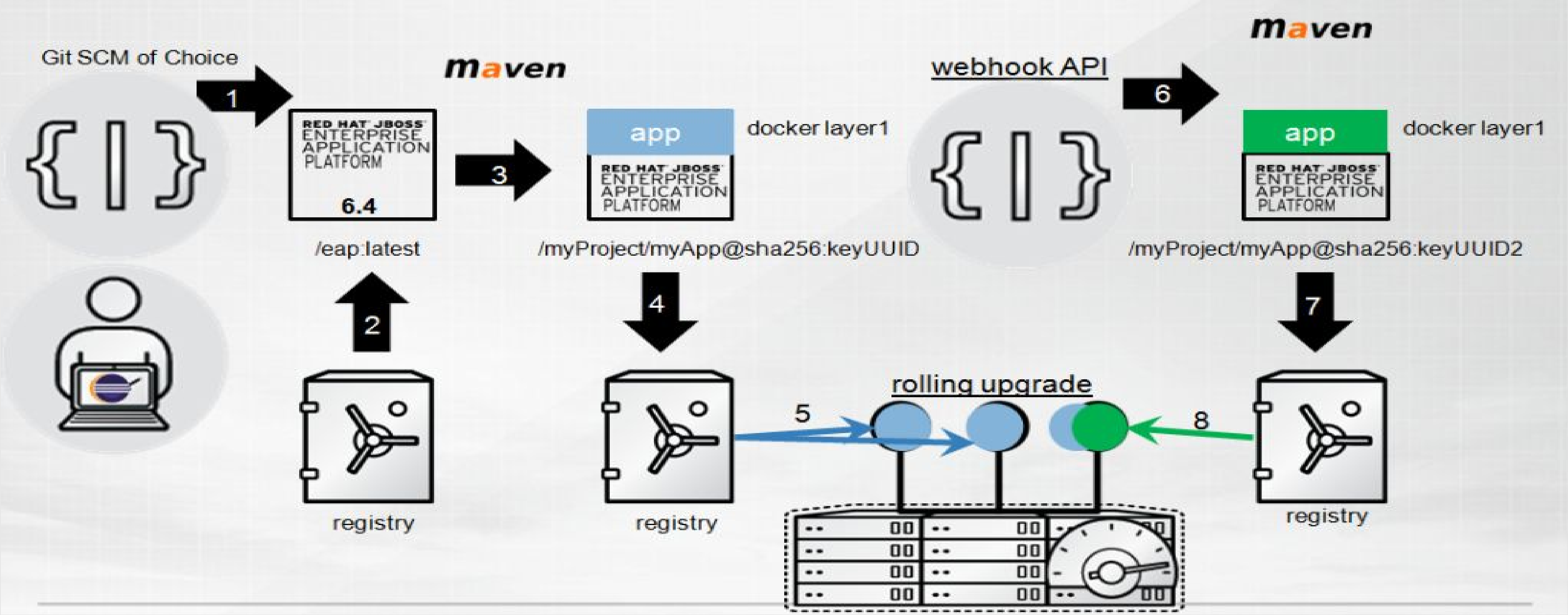
Building from application source



Source to Image (STI) is a next gen method allowing to automatically build and update containers by letting Openshift builds and links your application code to your Docker image.

This is a flexible method that can easily be plugged into any existing software delivery process.

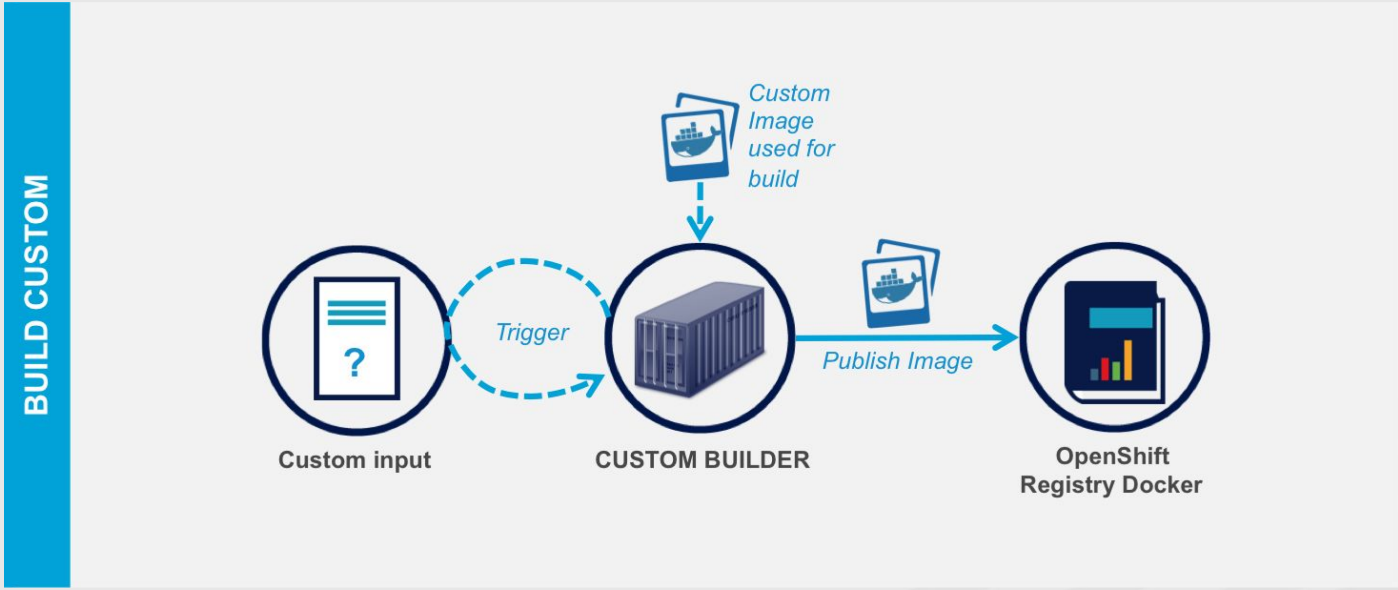
Building from application source



Source to Image (STI) is a next gen method allowing to automatically build and update containers by letting Openshift build your application code as well as your Docker image.

OPENSHIFT by Red Hat

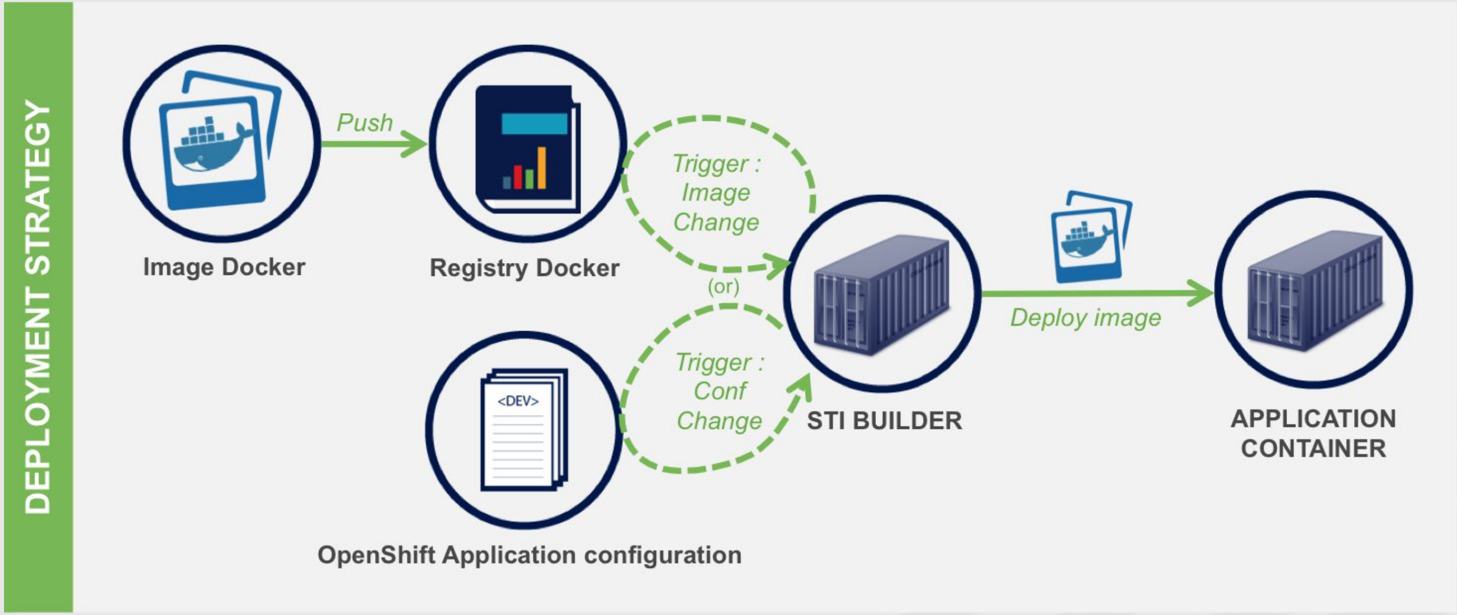
Custom build



Custom build allows to create complex process logic for non standard workflows.

OPENSHIFT by Red Hat

Deploying your application



Deployments strategies allow you to define the deployment workflows and release cycle adapted to your application.

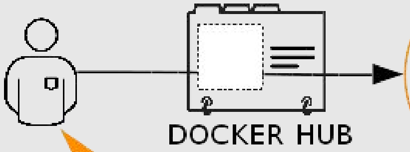
OPENSHIFT by Red Hat

OpenShift 3

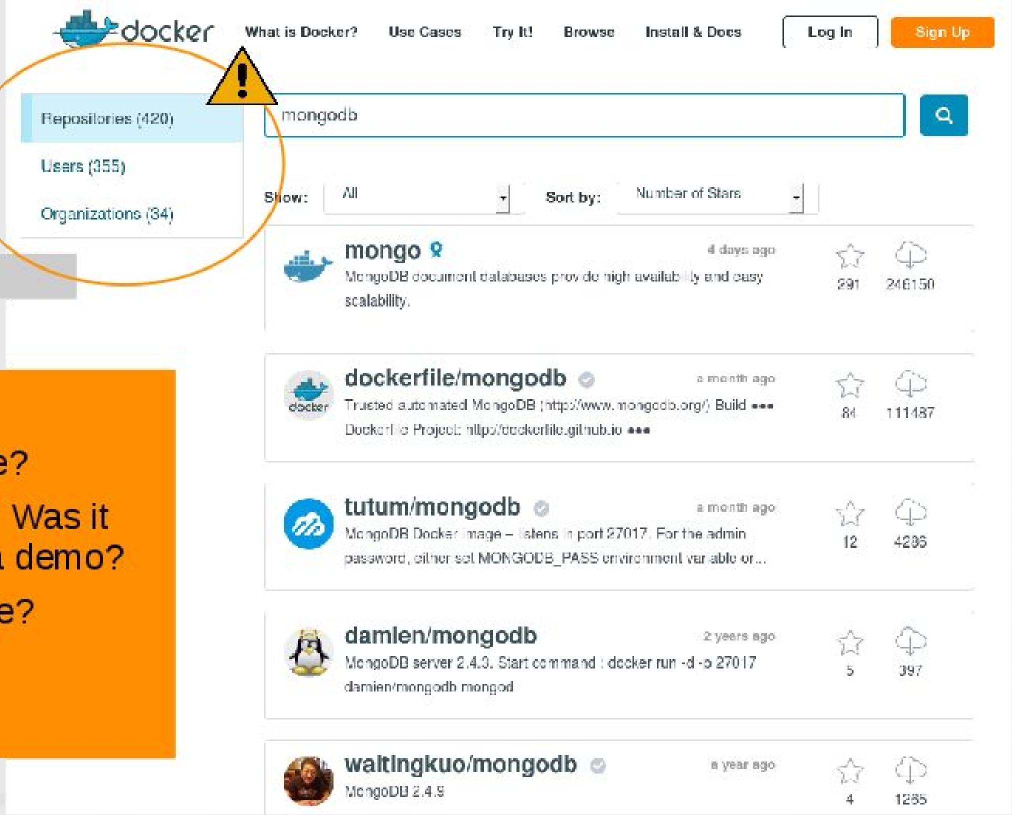
Security in Docker

OPENSHIFT by Red Hat

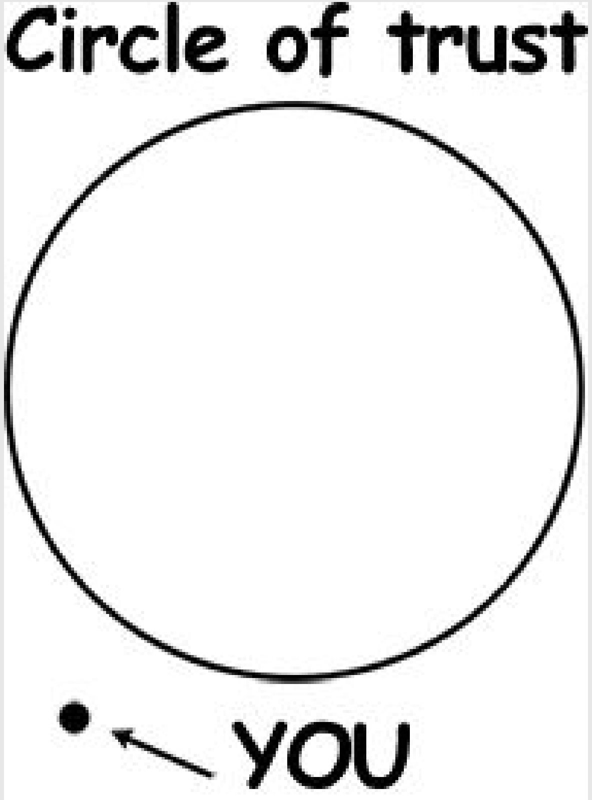
Trust what you run




```
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?


Circle of trust
YOU

Docker Content Trust: Notary

- Sign image by author (using private key) on Docker push
- Verify signature (using public key) on Docker pull

Provides:

- Protection Against Image Forgery
- Protection Against Replay Attacks
- Protection Against Key Compromise

SELinux in Docker

Volume mounts:

- -v /src:/dest:Z would give you a private label
- -v /src/dest:z will give you a shared label

```
$ docker run -it --rm -v /var/db:/var/db registry.access.redhat.com/rhel7 /bin/sh
sh-4.2# ls -Z /var/db/
-rw-r--r--. root root system_u:object_r:svirt_sandbox_file_t:s0 Makefile

$ docker run -it --rm -v /var/db:/var/db:z registry.access.redhat.com/rhel7 /bin/sh
# ls -Z /var/db
-rw-r--r--. root root system_u:object_r:svirt_sandbox_file_t:s0 Makefile

$ docker run -it --rm -v /var/db:/var/db:Z registry.access.redhat.com/rhel7 /bin/sh
# ls -Z /var/db
-rw-r--r--. root root system_u:object_r:svirt_sandbox_file_t:s0:c579,c909 Makefile
```

sVirt in Docker

Every container gets a different MCS label even if the have the same type of SELinux enforcement

```
$ docker run -itd --name fedora fedora bash
$ docker run -itd --name rhel6 registry.access.redhat.com/rhel6 bash
$ docker run -itd --name rhel7 registry.access.redhat.com/rhel7 bash
$ ps -efZ | grep -v kernel| grep svirt
system_u:system_r:svirt_lxc_net_t:s0:c158,c387 root 16396 1215  0 16:08 pts/1 00:00:00 bash
system_u:system_r:svirt_lxc_net_t:s0:c398,c448 root 16476 1215  0 16:08 pts/3 00:00:00 bash
system_u:system_r:svirt_lxc_net_t:s0:c455,c1002 root 16536 1215  0 16:08 pts/4 00:00:00 bash
```

OPENSHIFT by Red Hat

“If you have root in a container, you have root in the whole box”

- Don’t give root in a container
- If you have to give root, give “looks-like-root”
- If that’s not enough, give root but build another wall

Why don’t containers contain?

[Everything in Linux is not namespaced.]

Currently, Docker uses five namespaces to alter processes view of the system:

- Process (pid)
- Network (net)
- Mount (mount)
- Hostname (uts)
- Shared Memory (ipc)

OpenShift 3

Security in OpenShift

Authorization policies

Authorization policies determine whether a user is allowed to perform a given action within a project.

- Cluster policies
- Local policies

Security Context Contstraints

Security context constraints (SCC) that control the actions that a pod can perform and what it has the ability to access.

They allow an administrator to control the following:

- Running of privileged containers.
- Capabilities a container can request to be added.
- Use of host directories as volumes.
- The SELinux context of the container.
- The user ID.
- The use of host namespaces and networking.

OPENSHIFT by Red Hat

Secrets

Secrets provides a mechanism to hold sensitive information

- passwords
- OpenShift client config files
- dockercfg files
- private source repository credentials
- etc.

Caveats

- Drop privileges as quickly as possible
- Run your services as non-root whenever possible
- Treat root within a container as if it is root outside of the container
- Don't run random Docker images on your system.

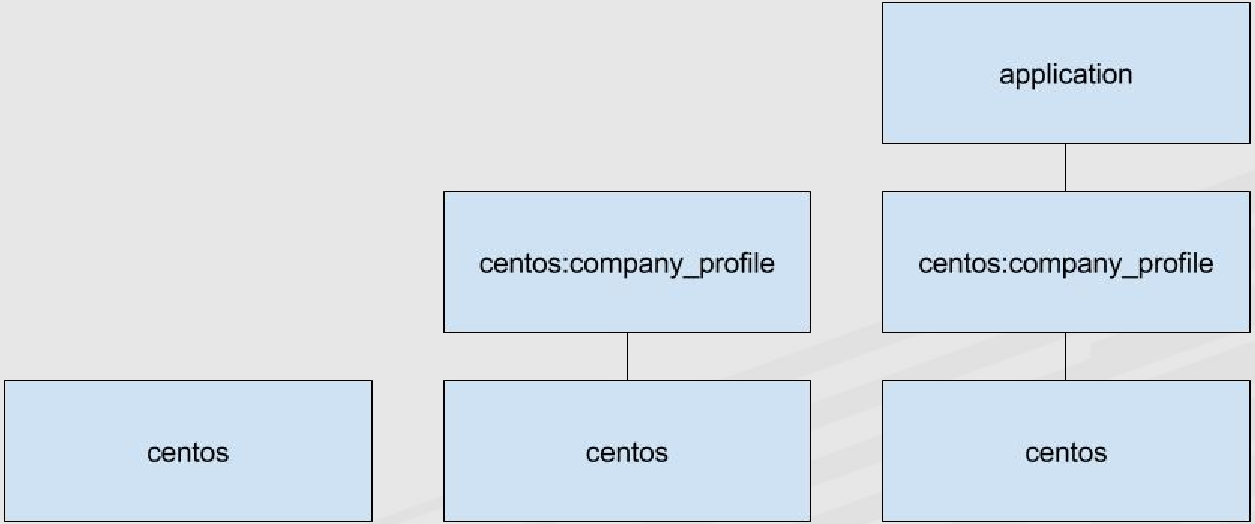
OpenShift 3

Build and Deploy secure containers

OPENSHIFT
by Red Hat

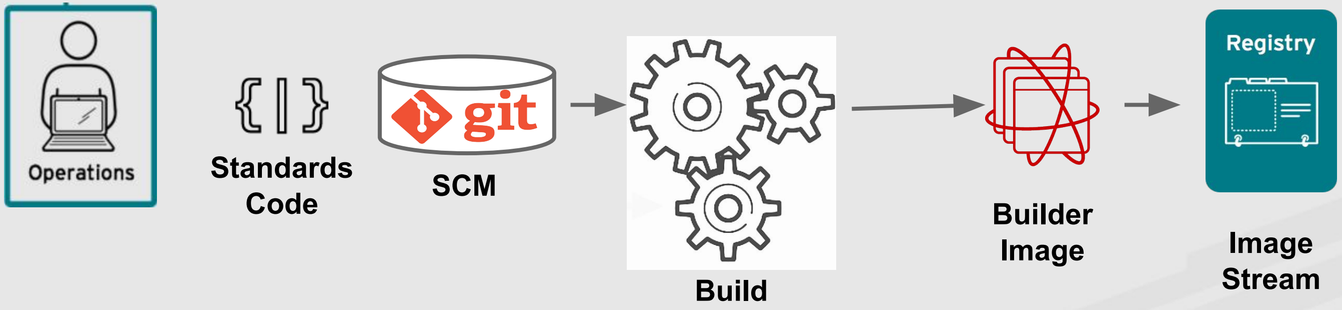
Create **standard** base images

Add all your requirements into base images using an appropriate hierarchy of layers



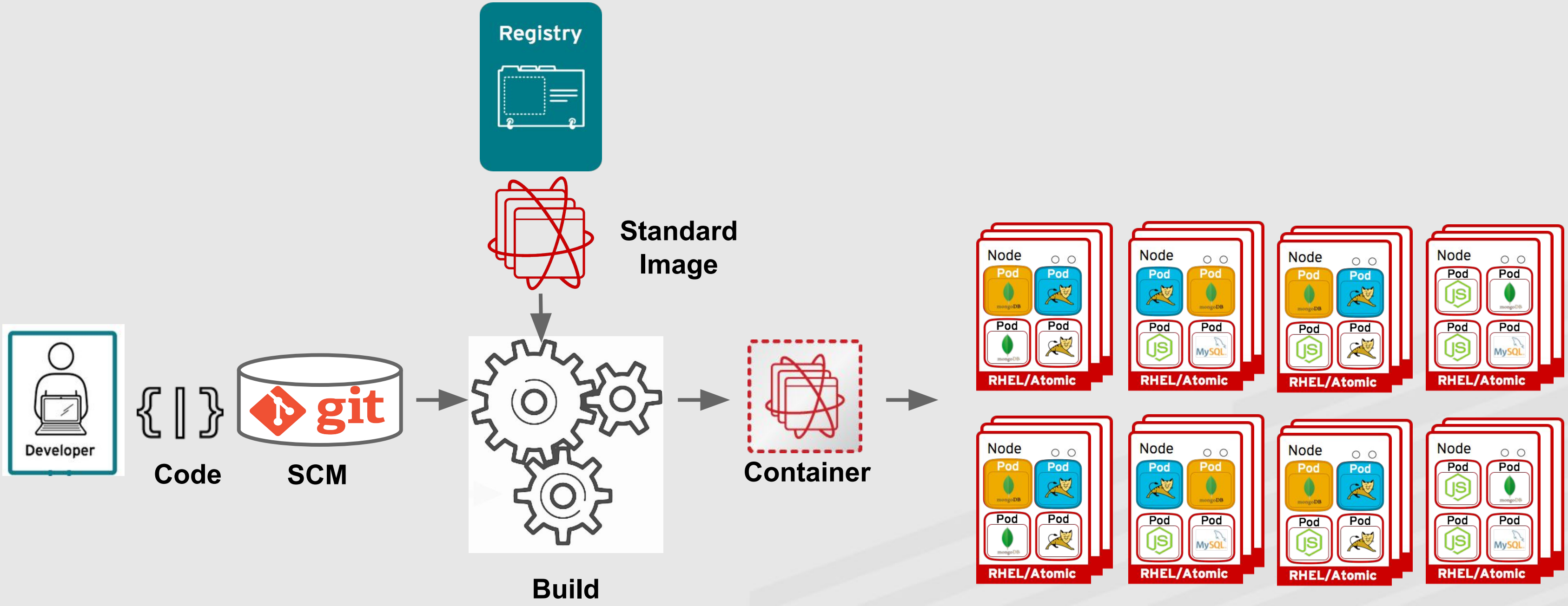
OPENSHIFT
by Red Hat

Install **standard** base images



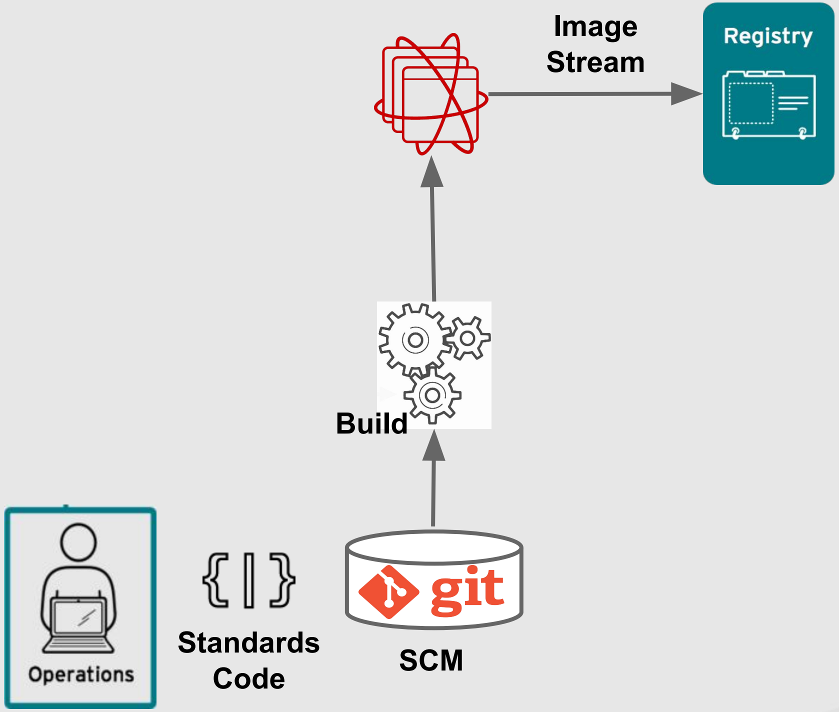
OPENSHIFT by Red Hat

Base all work on **standard** base images

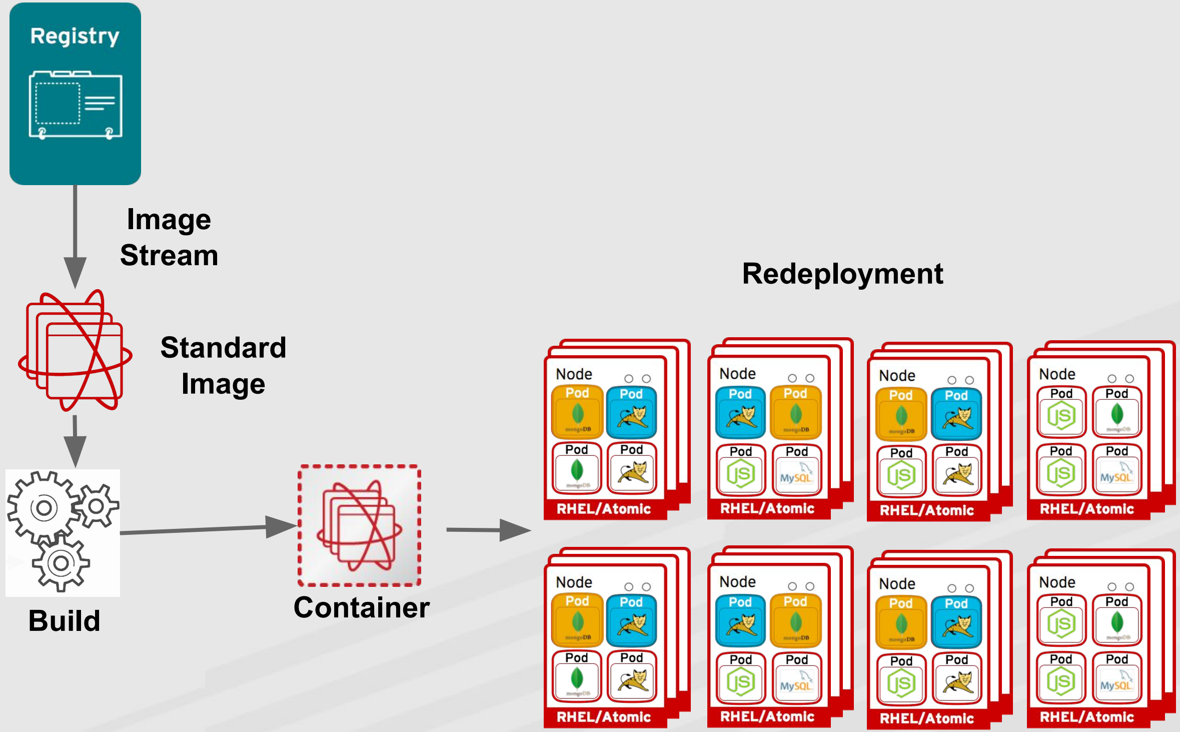


OPENSHIFT by Red Hat

Update **standard** base images



Update images based on **standard** base images



OPENSHIFT
by Red Hat

OpenShift 3

Q&A

OPENSHIFT
by Red Hat