RED HAT
FORUM
Europe, Middle East & Africa

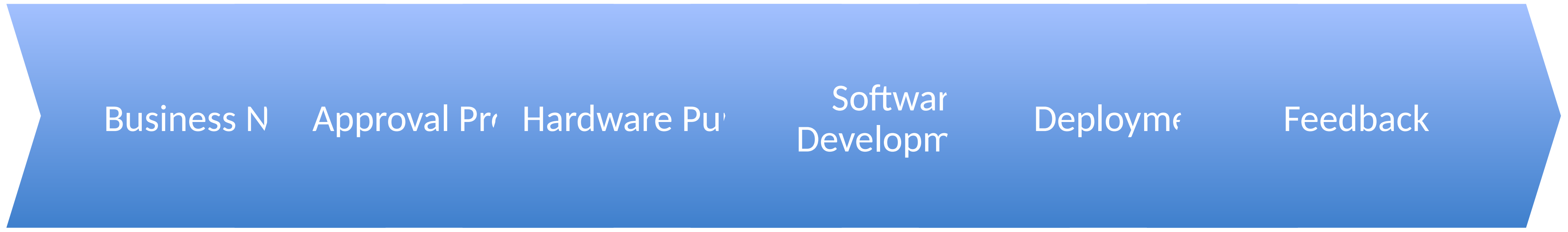
redhat.

OpenShift V3 – The next generation PAAS

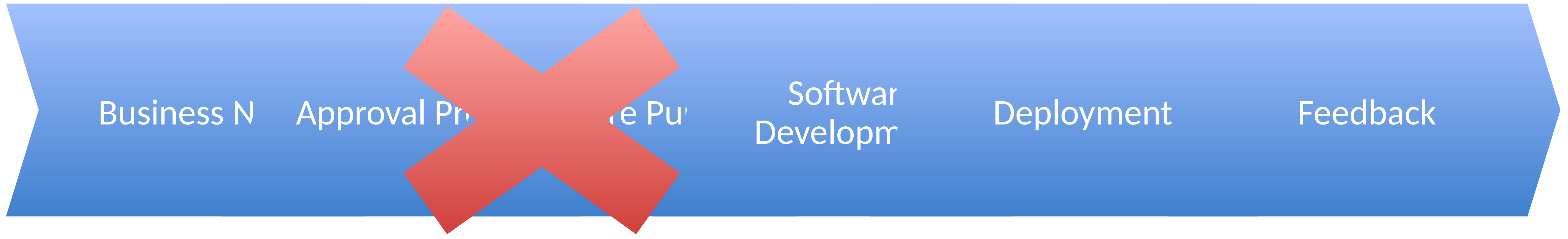
Chmouel Boudjnah – Twitter: chmouel
Principal Cloud Architect / Red Hat

PAAS

Traditional application development



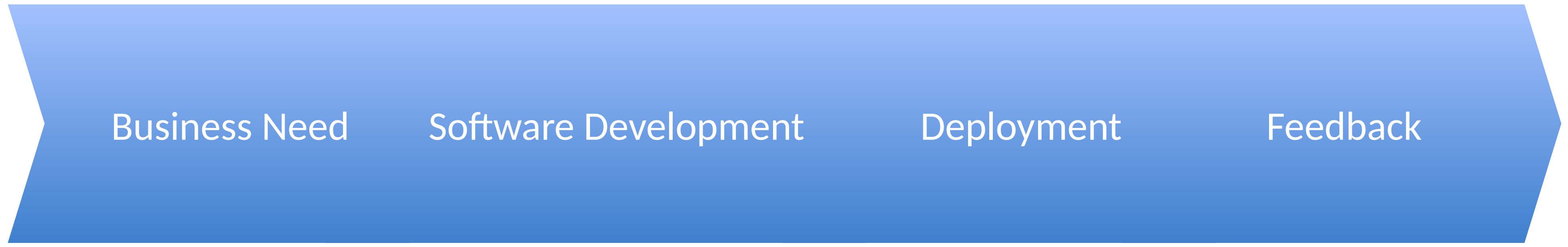
What IaaS cuts out



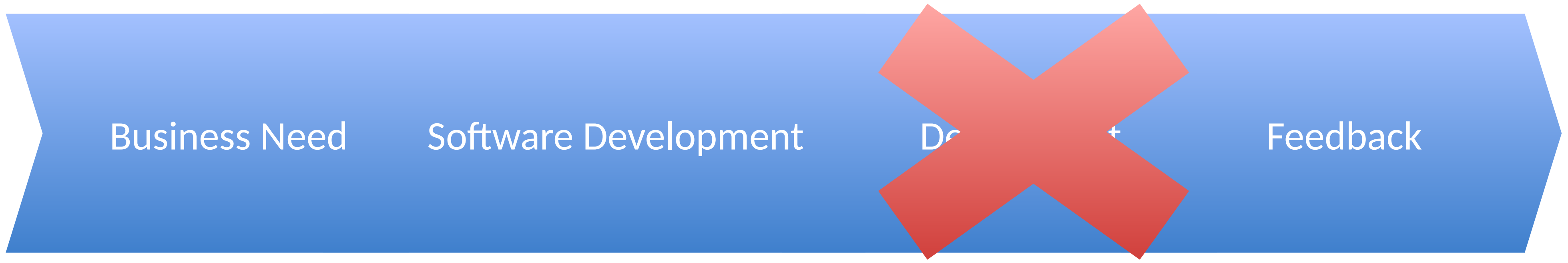
Hardware Provisioning is undifferentiated heavy lifting – use IaaS

Infrastructure services simplify your architecture with standardised, virtualised systems available anywhere on demand. Free up time to simplify the hard things.

Still more gains to be had



Still more gains to be had



Software provisioning is undifferentiated heavy lifting – use PaaS

Platform services drive complexity from applications to standardised infrastructure.

Paas as a platform for Devops

- **Developers build applications**
    - **Operations deploy applications**

  It can take weeks to get a VM after a developer files a ticket!
  And they still have to provision it (somehow)!

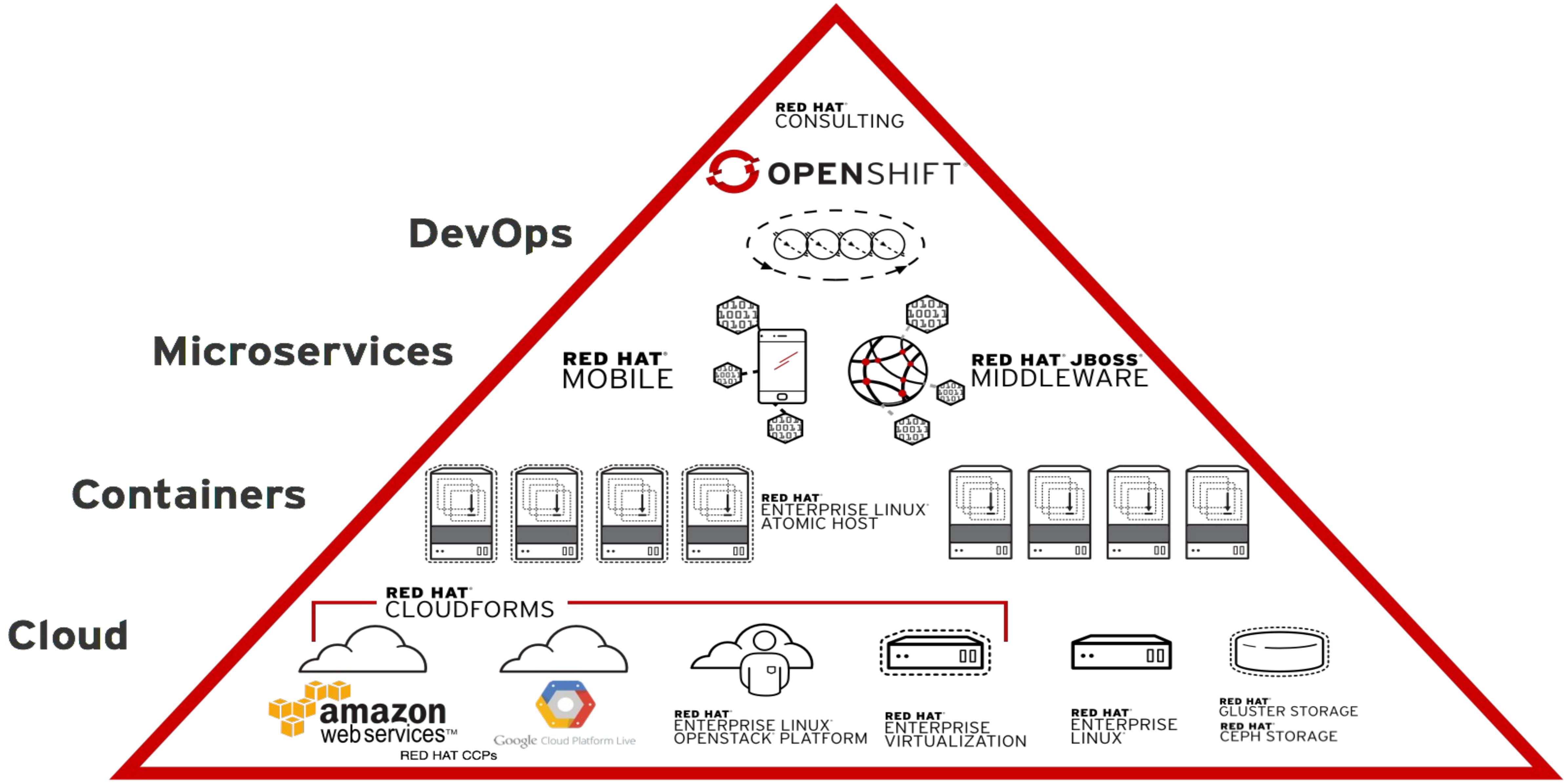
- **But if operations is a self service interface (API/Web Console)**

  Developers run their own applications
  Developers own their environments
  Developers are free and enabled
  Developers have incentives to be responsible

- **Less down time**

- **Less meetings**

Red Hat brings it all together



Containers

Containers the facts

- Containers are not new.
- Containers is not virtualisation
- Containers is not universally portable.
- Containers do not contain.

VELOCITY AND DENSITY

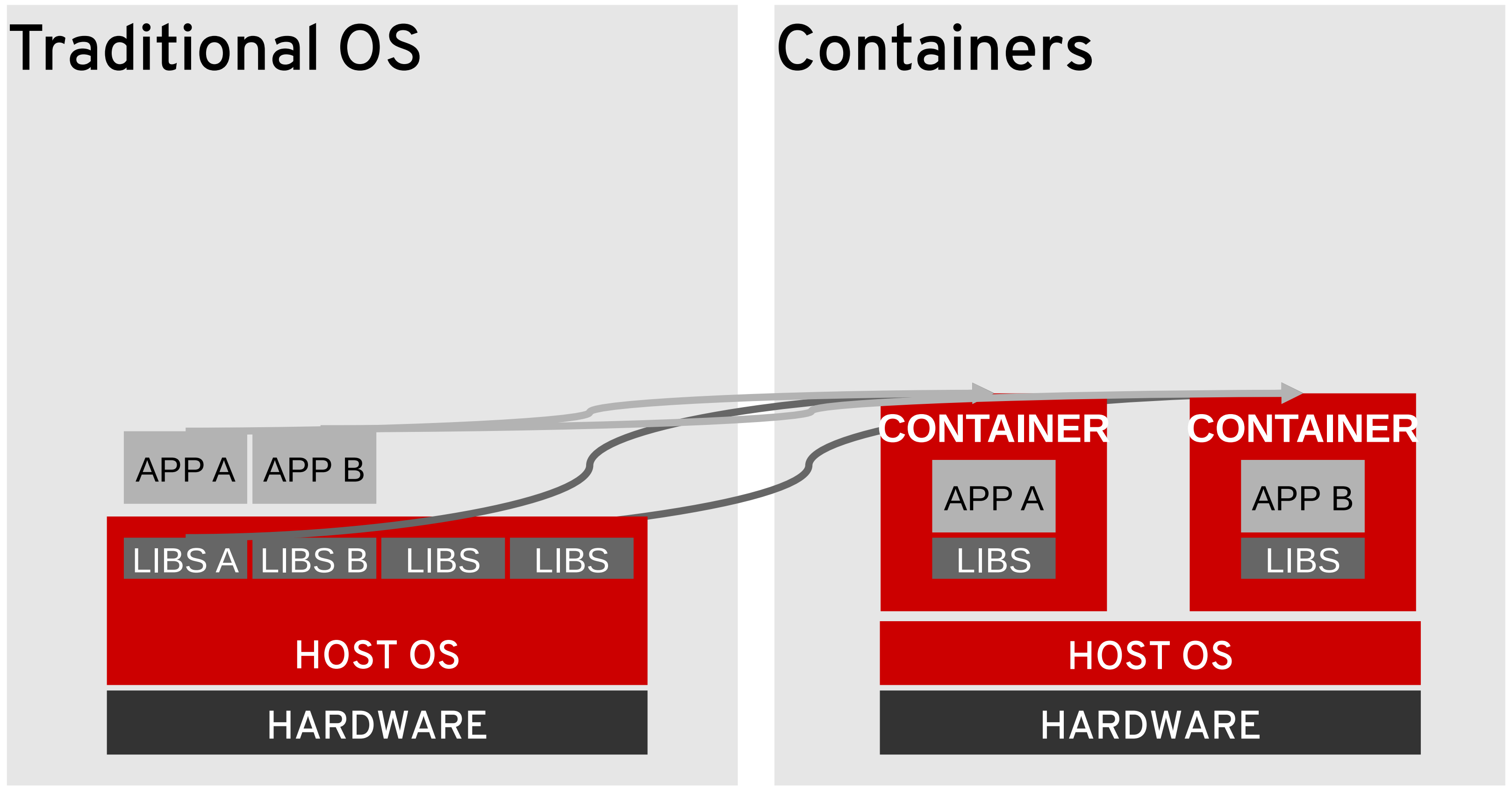
Time to set up

27 HRS	12 MINS	10 SECS
PHYSICAL SERVER	VIRTUAL MACHINE	CONTAINER INSTANCE

Density

VIRTUAL MACHINE INSTANCES 10 → 100 CONTAINER INSTANCES

TRADITIONAL OS VS. CONTAINERS



Containers 101

- 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
- Encourage microservices.

Containers Orchestration

Kubernetes

While Docker defines the container format and builds and manages individual containers, an orchestration tool is needed to deploy and manage sets of containers.

Kubernetes (the Helshman of the ship) drive those containers taking care of them on multiple nodes.

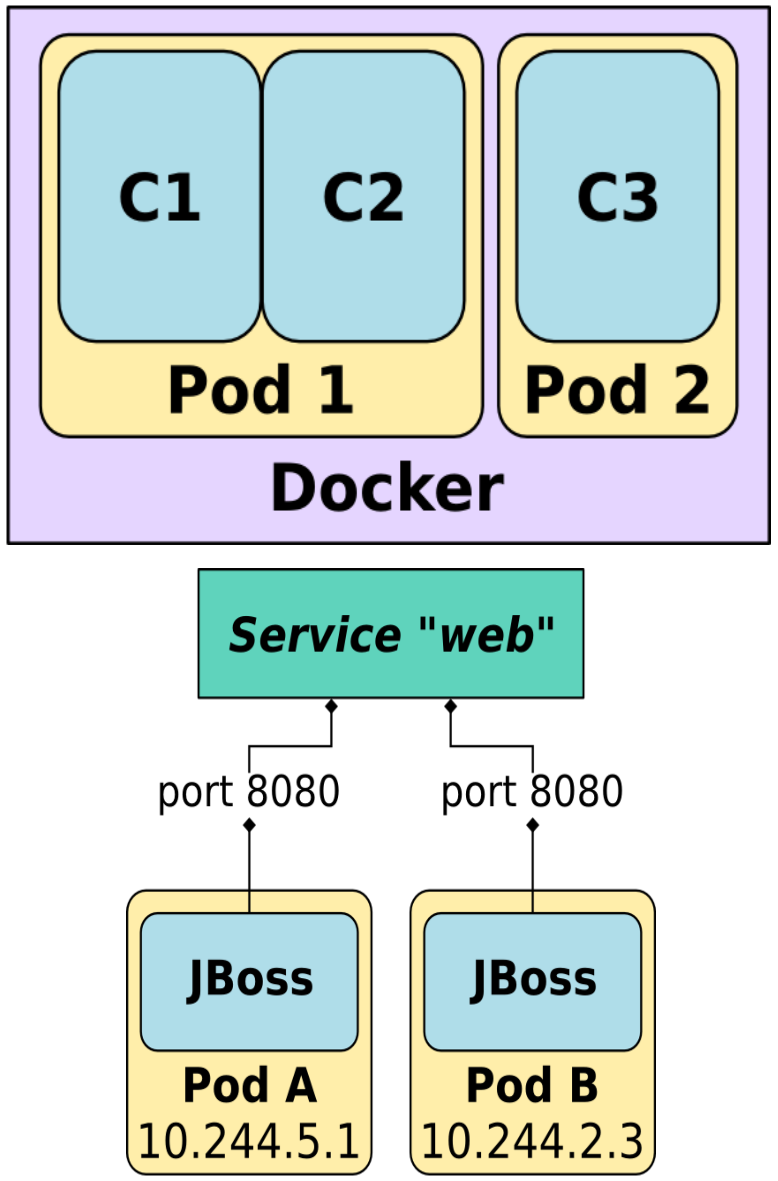
What is Kubernetes

- Google has been using containers for over a decade – they start over 2 billion containers a week!
- Kubernetes is the fourth iteration of a cluster manager that Google has developed
- Red Hat is collaborating in the kubernetes project

How Kubernetes orchestrate

- Declarative API how to launch containers
- Monitor state and maintain, increase or reduce copies of containers
- Container oriented networking for non kubernetes native applications
- Shared storage between hosts and failover
- Service ubiquity

Kubernetes architecture



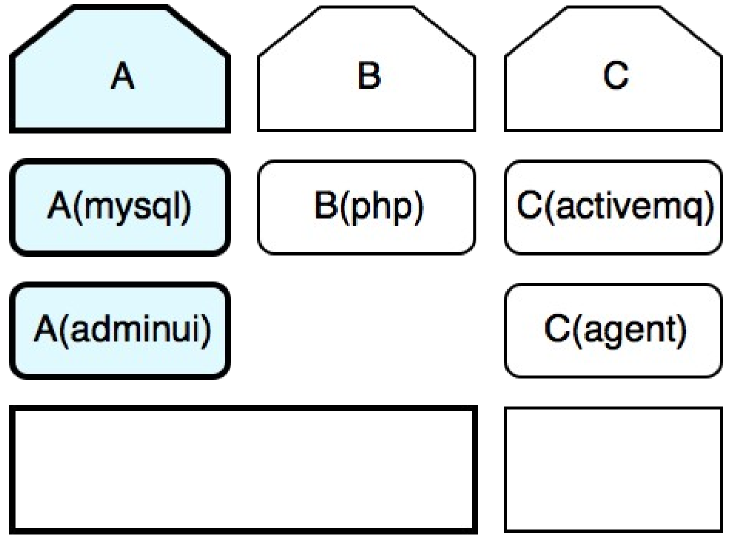
Pod: colocated group of Docker containers that share and IP and and storage volumes

Service: provides a single, stable name for set of pods and acts as basic load balancer

Replication controller: manages the lifecycle of pods and ensures specified number are running

Label: used to organise and select groups of pods

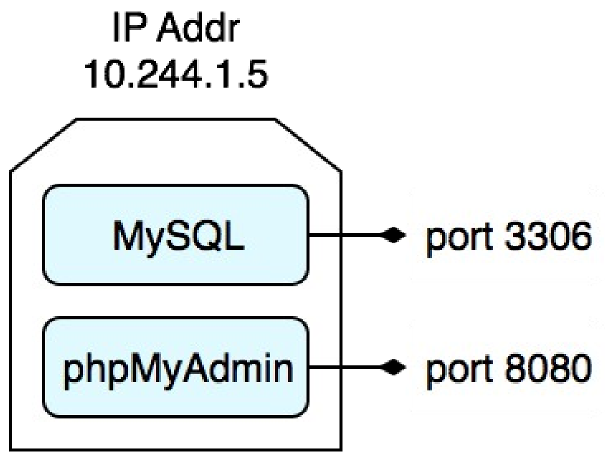
Pods



Pods
Group of related containers placed onto the same host

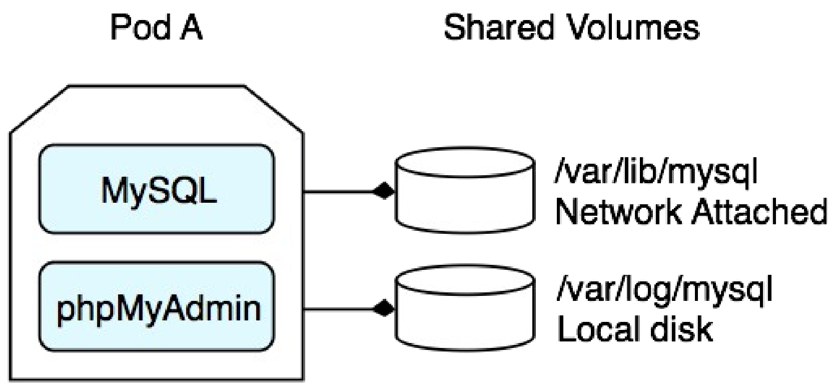
Docker Containers
Application processes and filesystem libraries

Hosts
Runs pods



Pod Networking
Each pod has an IP address that other pods can contact

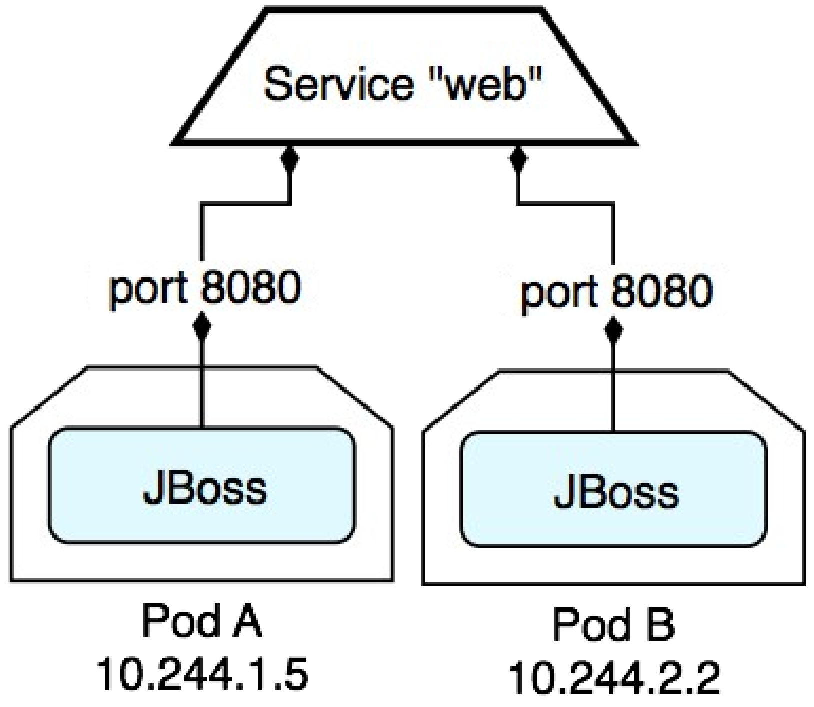
Shared Ports
Each container must share pod ports. No conflicts allowed



Volumes per Pod
Each pod has a list of volumes that all containers access the same

Volume Types
Each volume can have different types, like local transient storage or network attached storage backed by Cinder, GCE, EBS, etc

Services

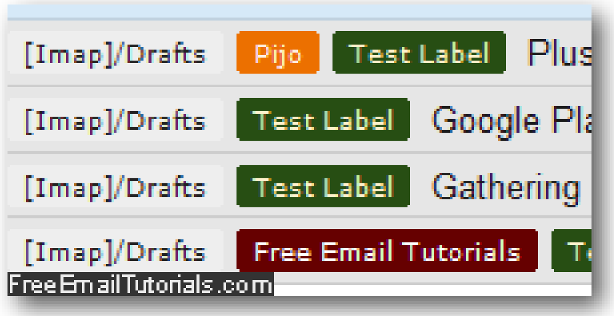


Services abstract other pods
A service is a TCP port that may transparently load balance other ports

Replication controllers copy pods
A controller ensures there are a certain number of copies of a pod, so if a host is lost another pod gets created.

Labels

- It's basically like Gmail label but for infrastructure.
- Labels are key/value pairs that are attached to objects, such as pods.
- Labels are intended to be used to specify identifying attributes of objects
- Labels can be used to organize and to select subsets of objects.
- Labels can be attached to objects at creation time and subsequently added and modified at any time. (eg: CI/CD)



[Imap]/Drafts Pijo Test Label Plus
[Imap]/Drafts Test Label Google Pla
[Imap]/Drafts Test Label Gathering
[Imap]/Drafts Free Email Tutorials Te
FreeEmailTutorials.com

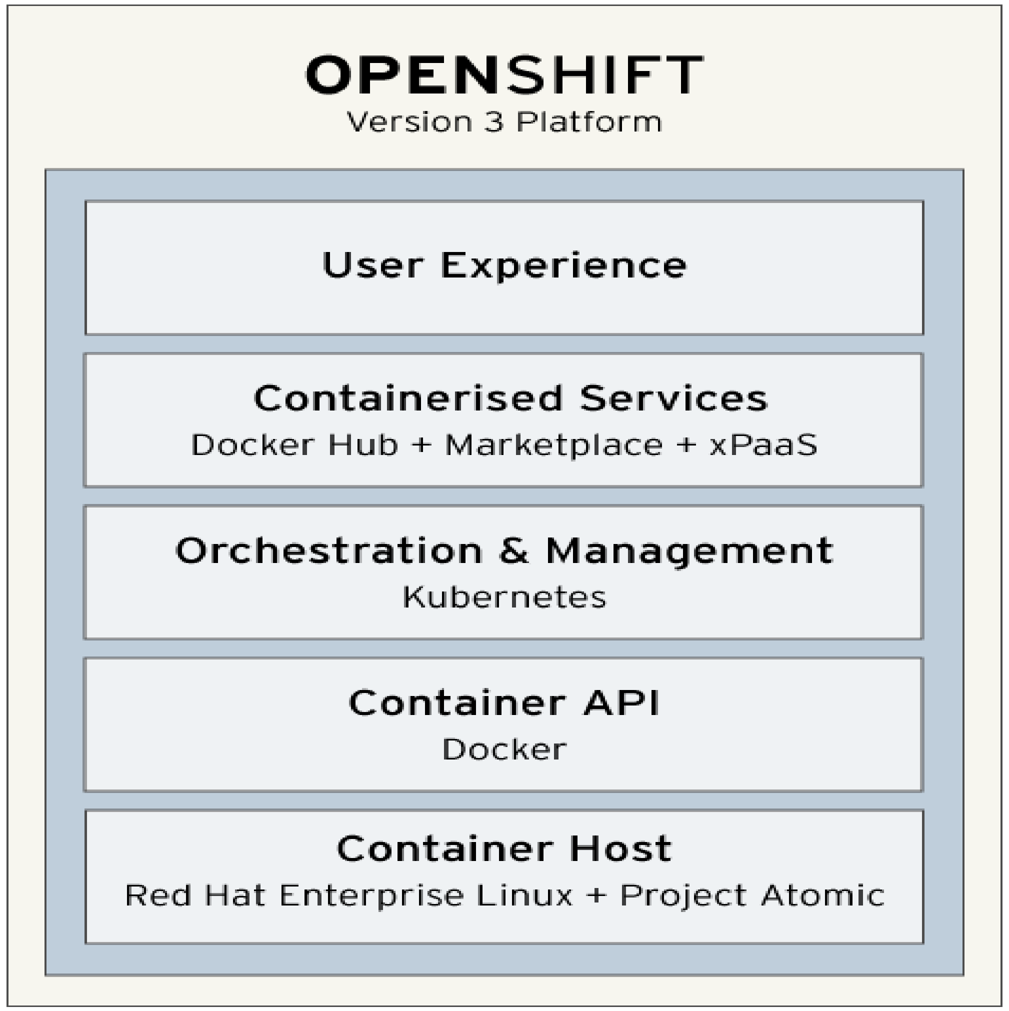
What’s missing ?

- Turning source code into deployable components
- Decoupled devs and ops
- Integration with developer tools
- Software defined networks
- Users, teams, quotas, access rights, etc...
- Build, manage and deliver application descriptions at scale

OpenShift

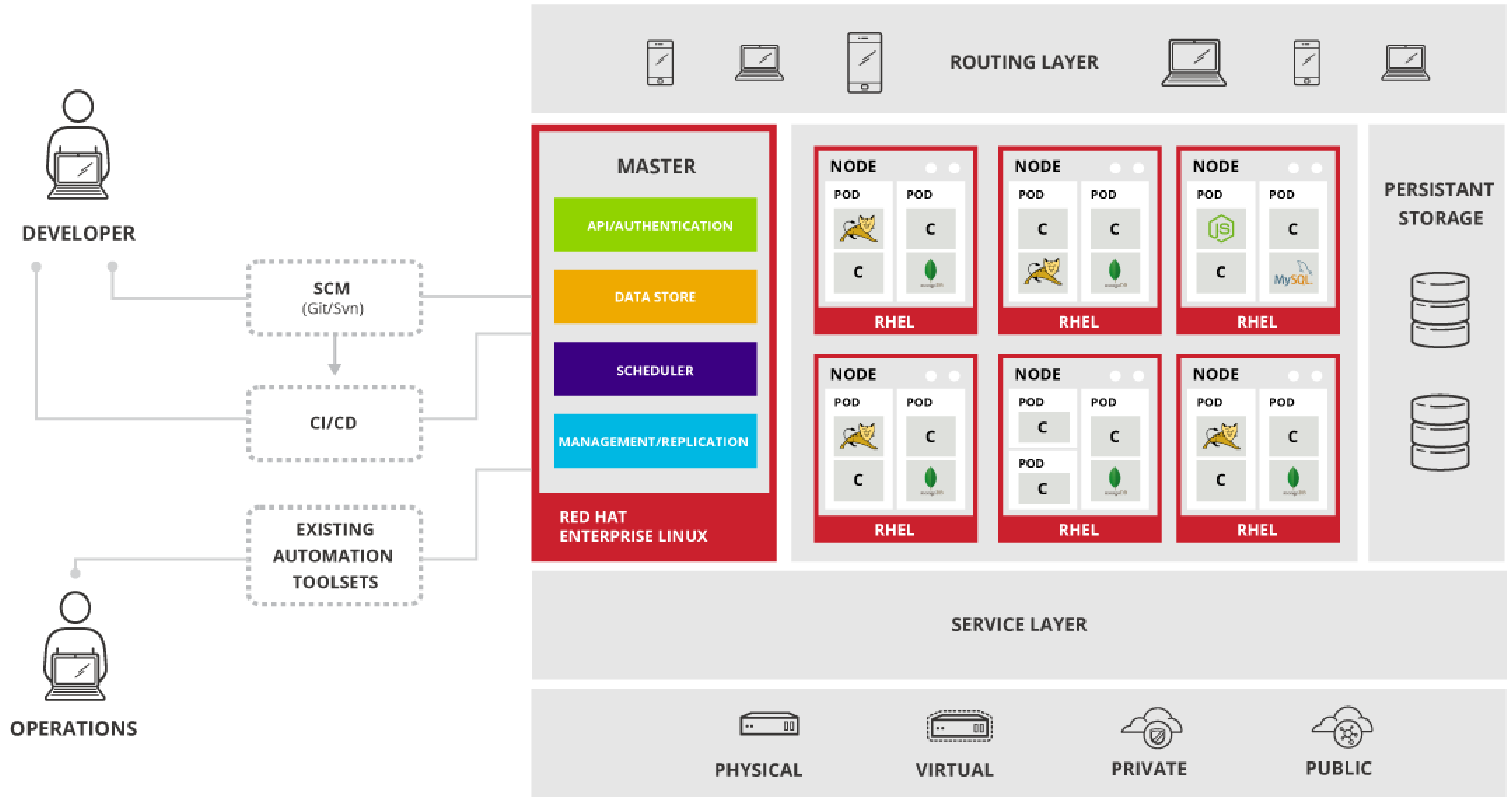
OpenShift v3 Stack

- Industry standard, web scale distributed application platform
- Container-optimized OS
- Standard containers API
- Web-scale container orchestration & management
- Largest selection of supported application runtimes & services
- Robust tools and UX for Development & Operations



OpenShift v3 Design Goals

- Rich user experience for Developers and Operators
- Multi-tenant collaboration - users, teams, projects
- Application build and deployment automation
- Integration with CI and ALM
- Container networking / routing
- Scheduler (regions / zones)
- Simplified installation and operational management


ROUTING LAYER
DEVELOPER
SCM
(Git/Svn)
CI/CD
EXISTING AUTOMATION TOOLSETS
OPERATIONS
MASTER
API/AUTHENTICATION
DATA STORE
SCHEDULER
MANAGEMENT/REPLICATION
RED HAT ENTERPRISE LINUX
NODE
POD
C
RHEL
PERSISTANT STORAGE
SERVICE LAYER
PHYSICAL
VIRTUAL
PRIVATE
PUBLIC

OpenShift tenets

Networking

App deployers should see flat networks

Define private vs public, internal vs external, fast vs slow

Storage

Most components need *simple* persistent storage

Ensure storage is not coupled to the host

Health

Every component should expose health information

OpenShift components on top of Kubernetes

- Internal registry
  Buildimages from source code securely (Source to Image)

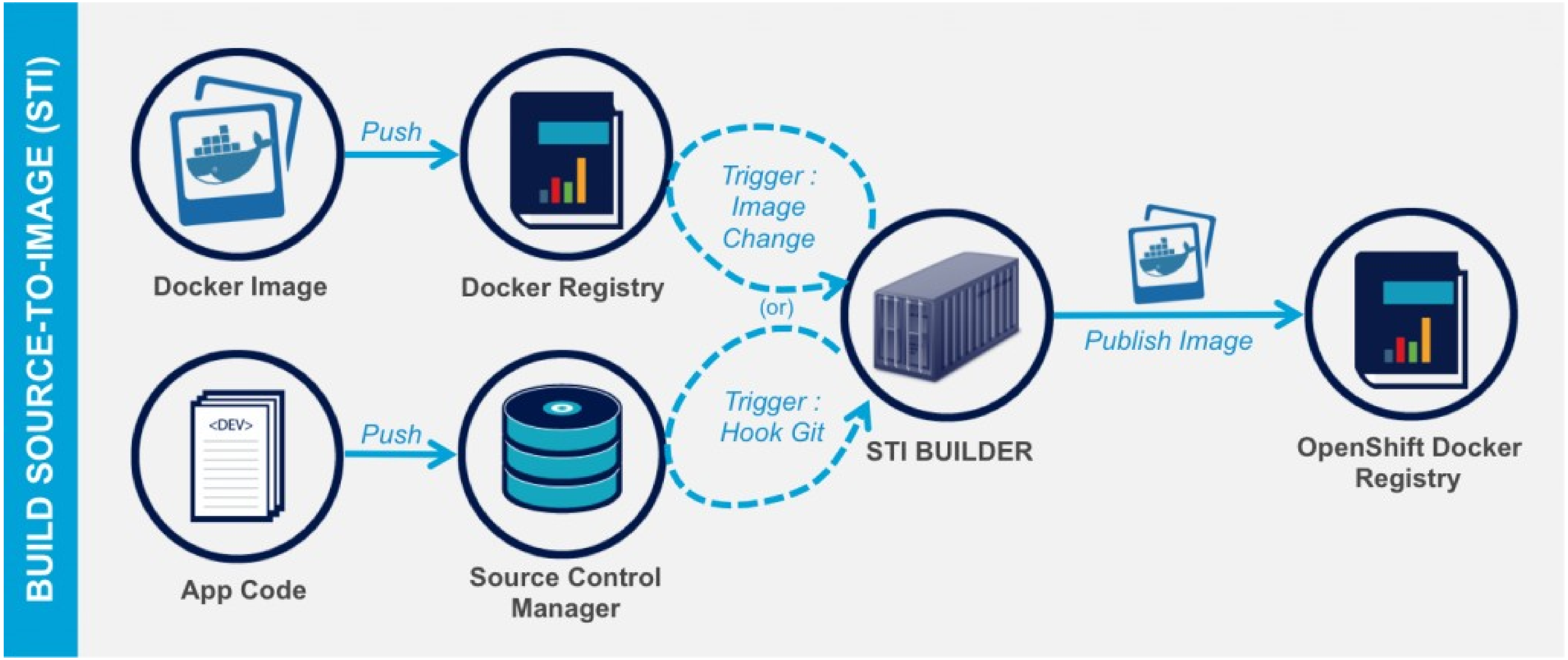
- GitHub/SCM integration
  Including notification with webhooks support

- Web ‘router’ layer
  exposing services directly to the internet

- Security context constraints
  Fine grained security management of OpenShift resources.

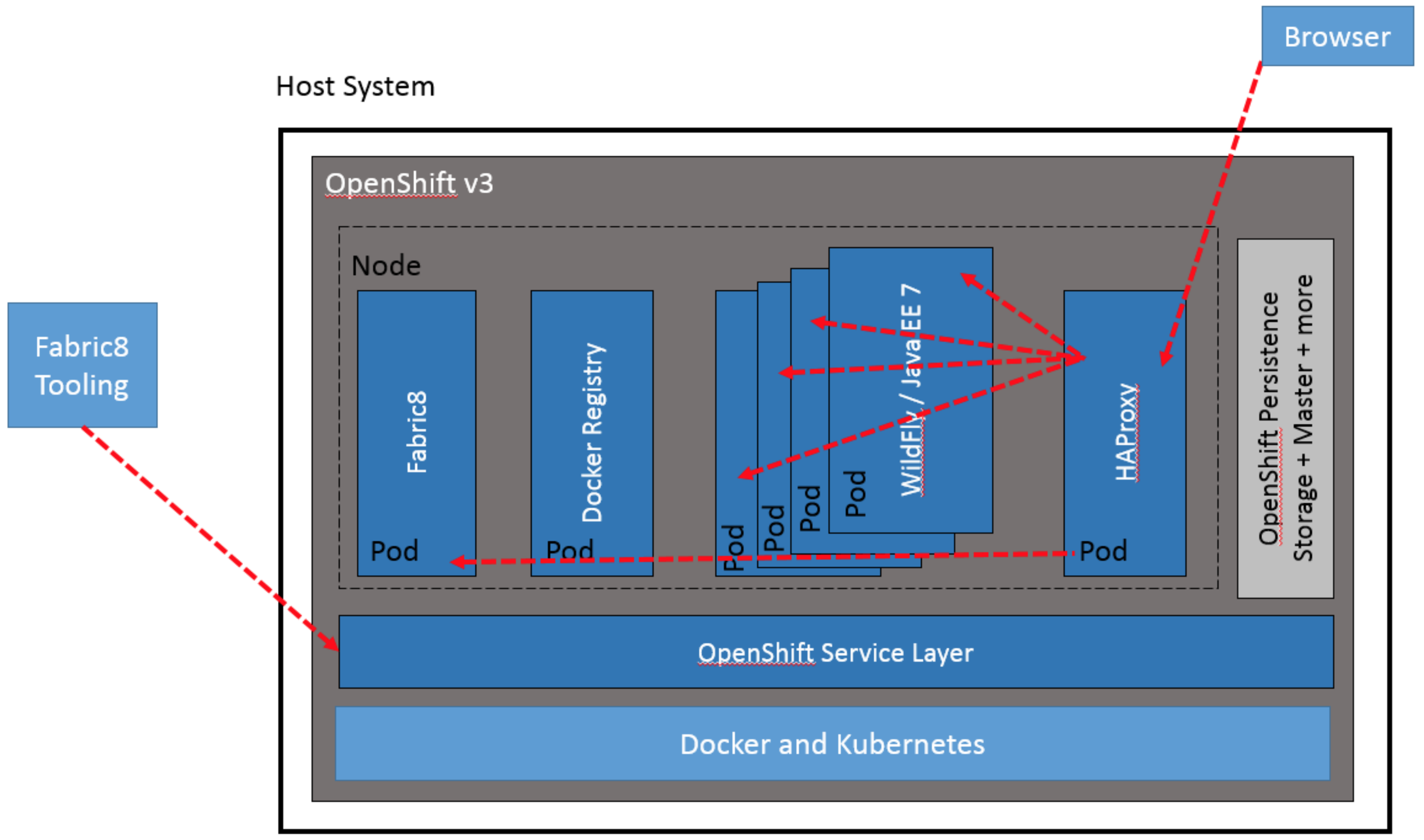
- Enteprise components
  HA, Packaging, Log aggregation, Monitoring, Deployment etc..

OpenShift Source to Image process



OpenShift Router

By default Kubernetes expose Services provide by the PODs, those services are not exposed to the outside easily. OpenShift router allow to expose those services to the public dynamically via API specified by the developer.



OpenShift Security

- **Ensure containers “contain”**
  - SELinux, user namespaces, audit
  - Random UUID when running containers
  - Decompose the Docker daemon over time
  - Fine grained security controls on SSH access
- **Allow easy integration with existing security tools**
  - Kerberos, system wide security, improved scoping of access
  - More customization possible
- **Allow application network isolation**

Twitter: @Chmouel