DevOpsCon

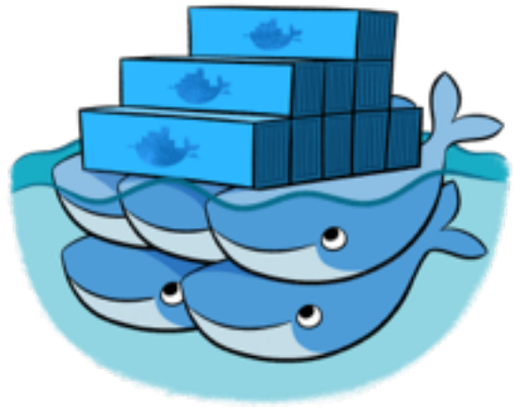
Dr. Roland Huß | Red Hat

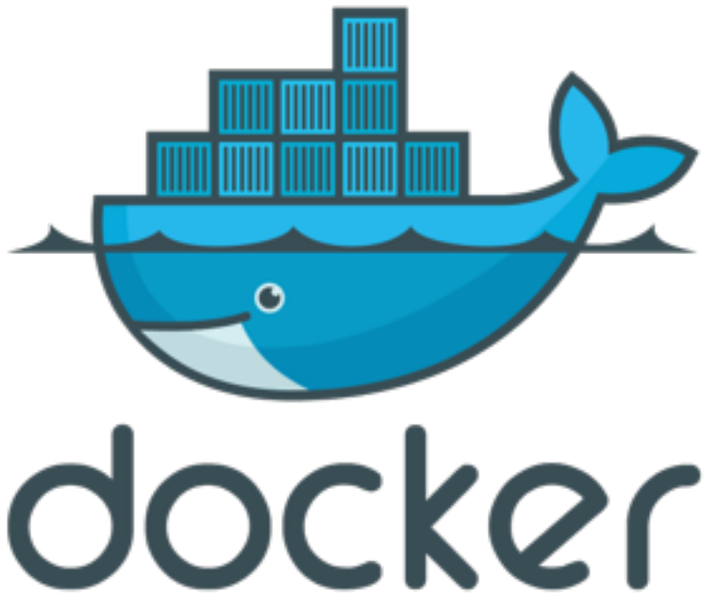
fabric8

… and Docker, Kubernetes, OpenShift

MESOS

DEIS



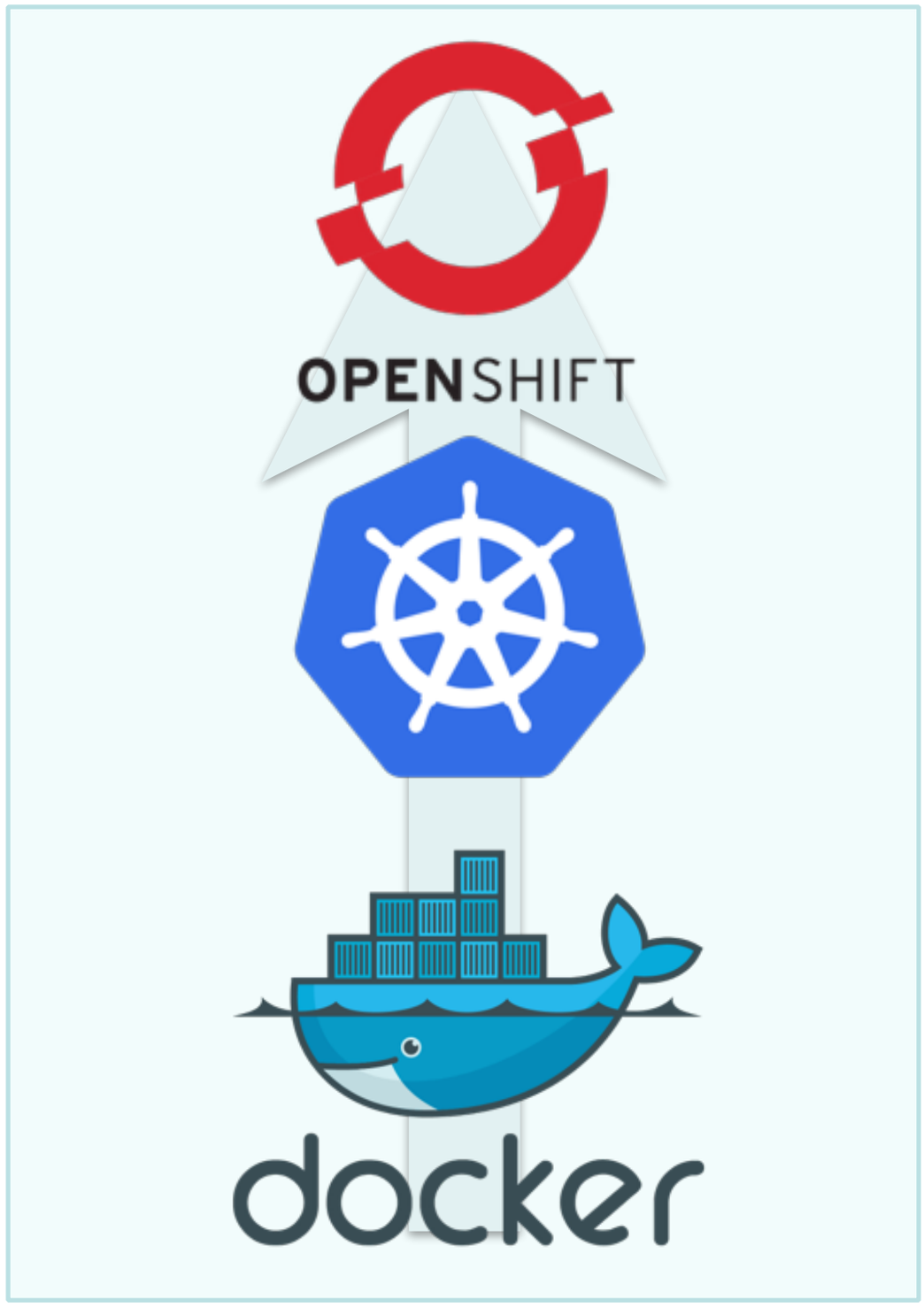


docker

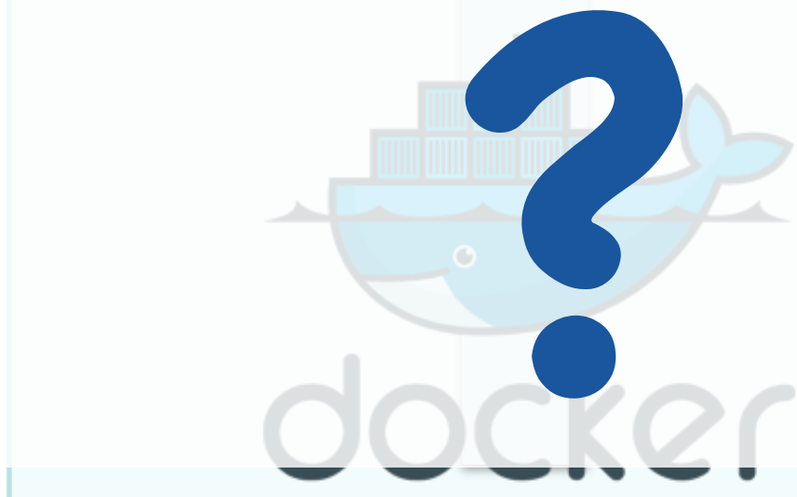
tutum

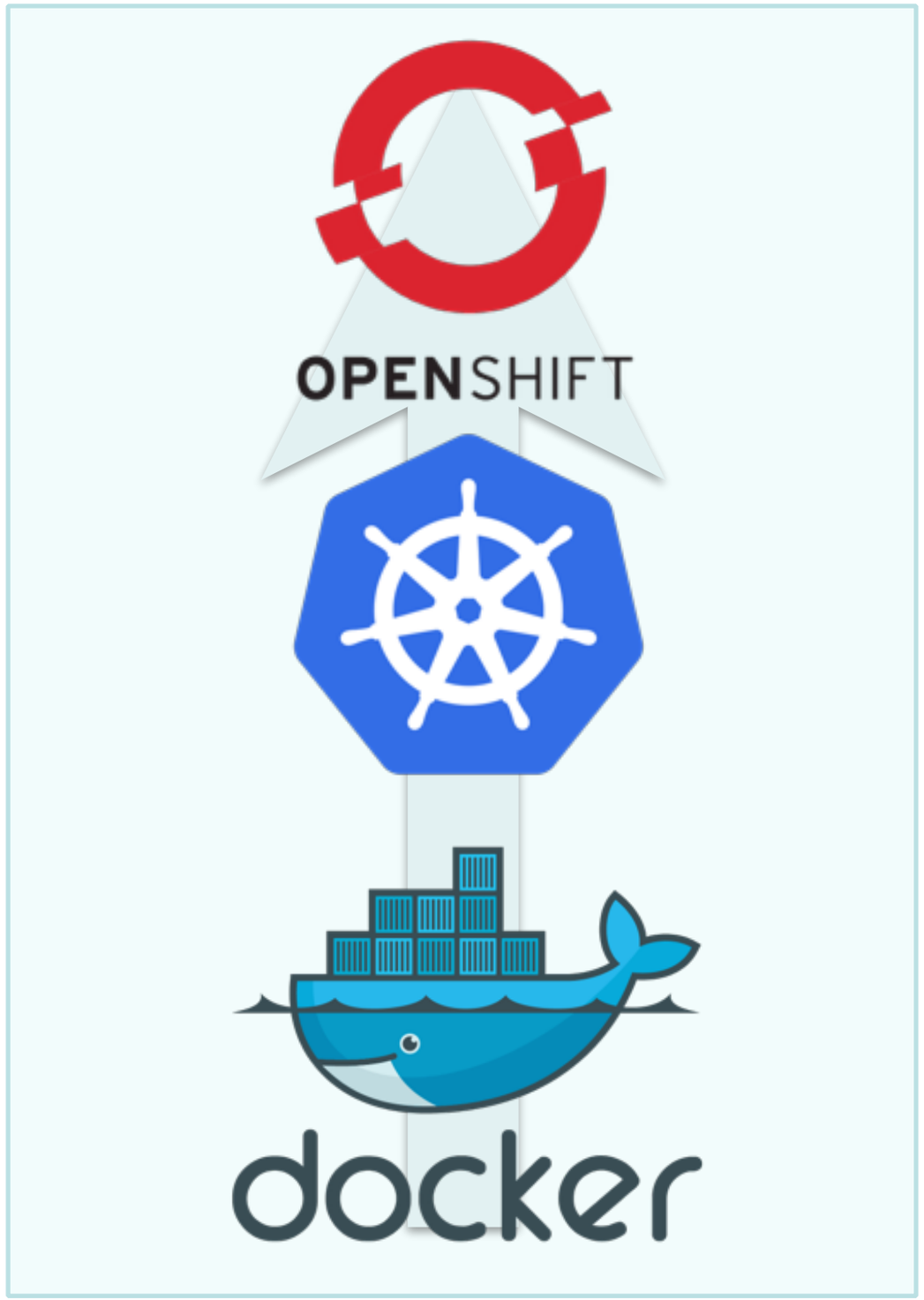
OPENSHIFT

CLOUD FOUNDRY™

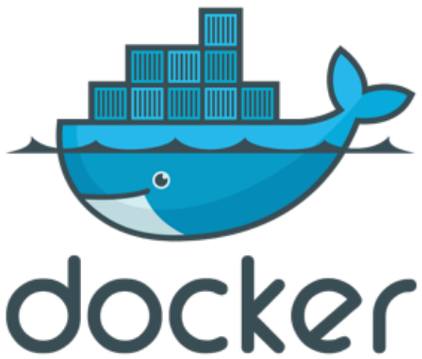

OPENSHIFT
docker

fabric8





fabric8



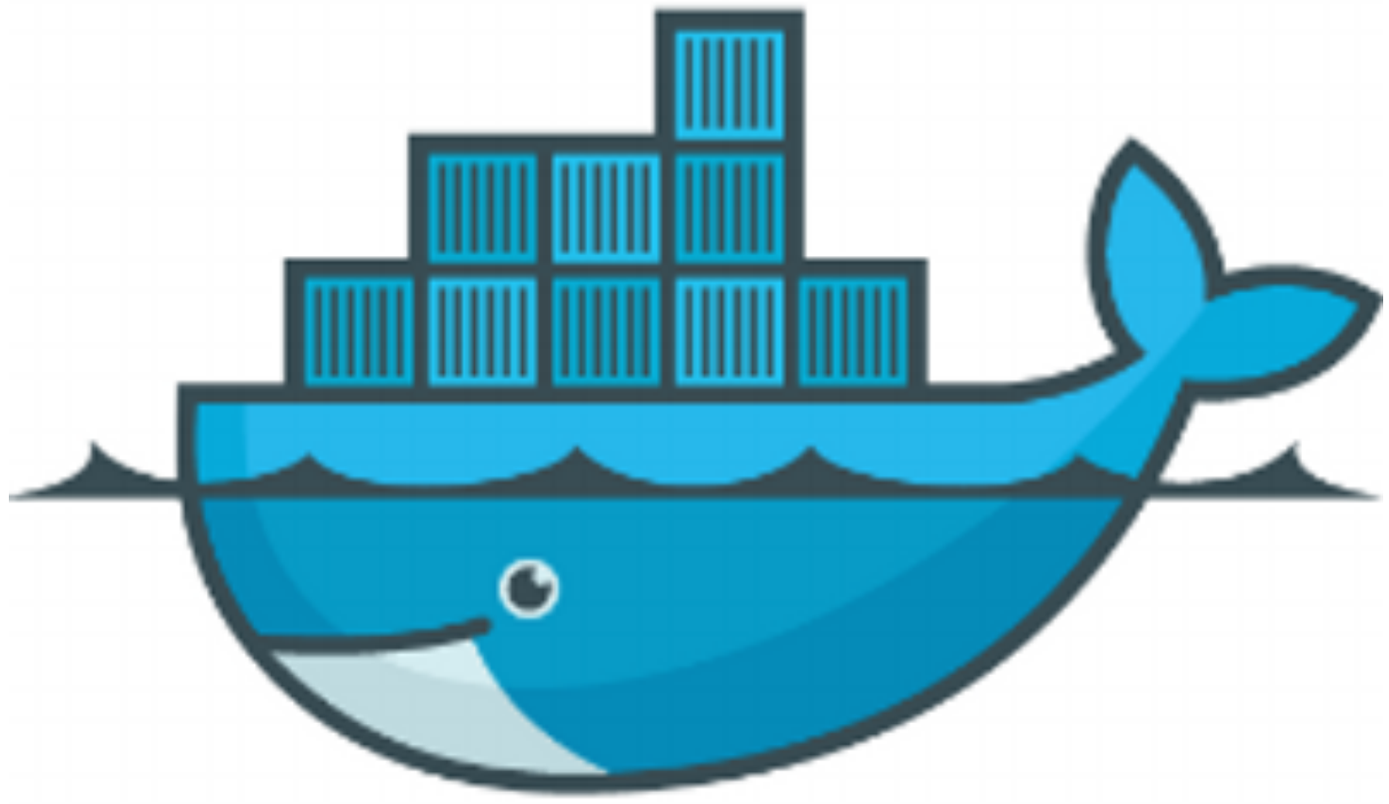
OS Level Virtualisation

Docker Orchestration

PaaS Platform on top of Kubernetes

Services and Tools for Kubernetes and OpenShift

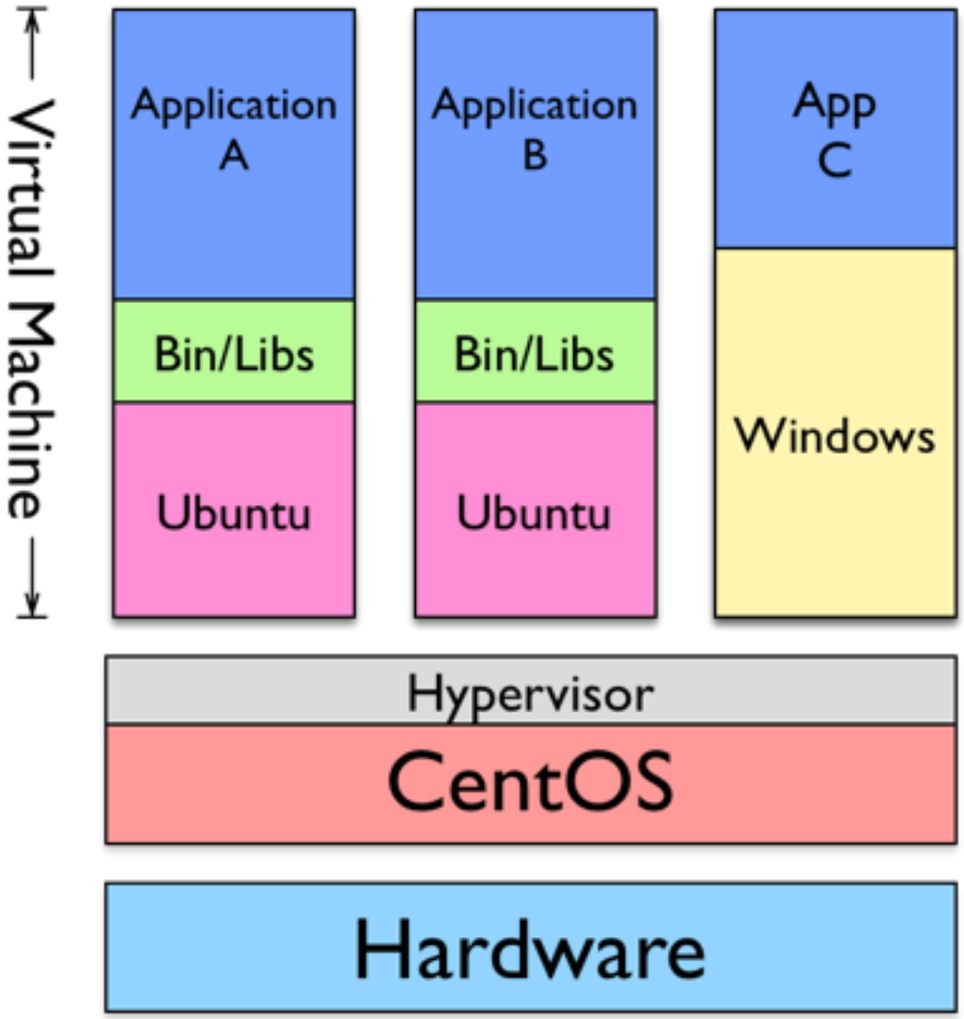
Docker



Facts

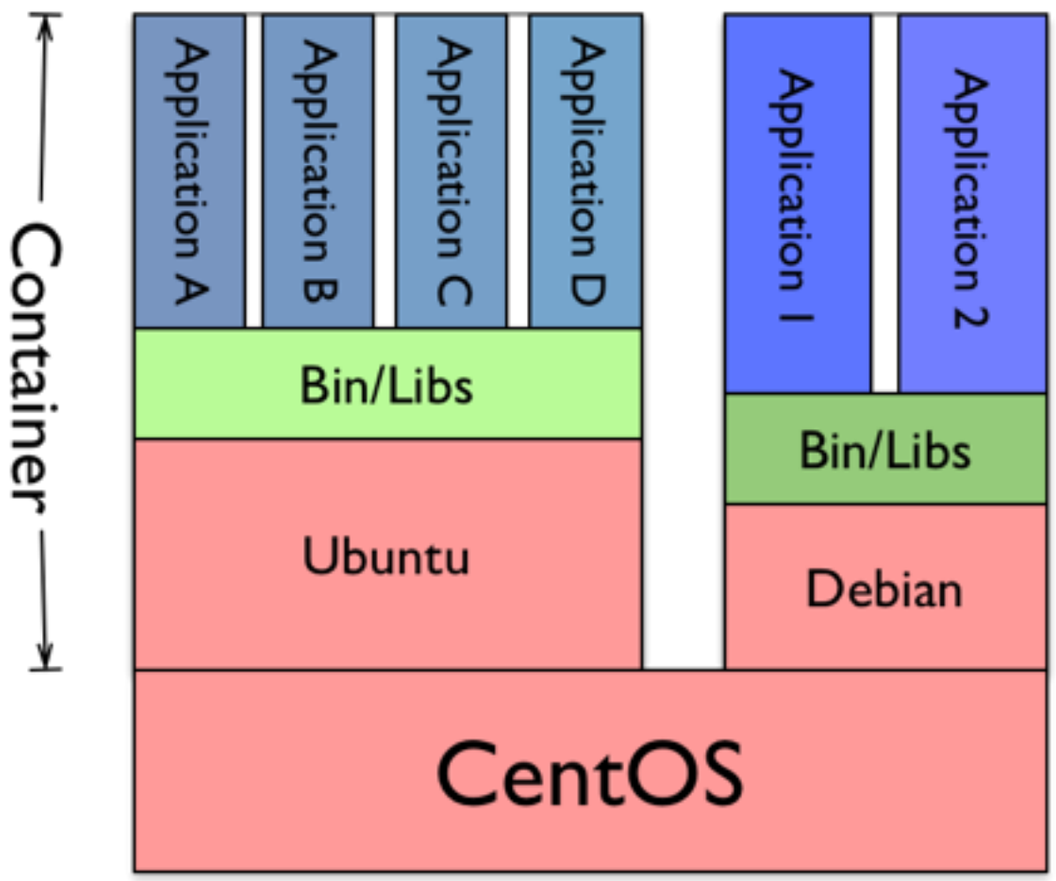
- » OS level virtualisation tool suite
- » Client-Server architecture
  - – Server communicates via Unix- or INET-Sockets with a REST API
- » Docker commands via CLI
- » Written in Go
- » Current version: 1.6

Lightweight Container vs. VM



Containers are isolated, but sharing the kernel and (some) files

➜ faster & lighter



Concepts

- **Image**
  - Read-only filesystem layer
  - Deploy & Share
  - Blueprint for a container
- **Container**
  - Read-write filesystem layer (copy-on-write)
  - Instance of an image
  - Has a lifecycle (start & stop)

Concepts

- **Repository**
  - Collection of layered images
  - often synonym for “Image”
  - Has a name: `registry/user/repository:tag`
- **Registry**
  - Storage for repositories
  - Default: **docker.io** (public docker hub)

docker

» CLI for managing Docker

- `docker <sub-command> …`

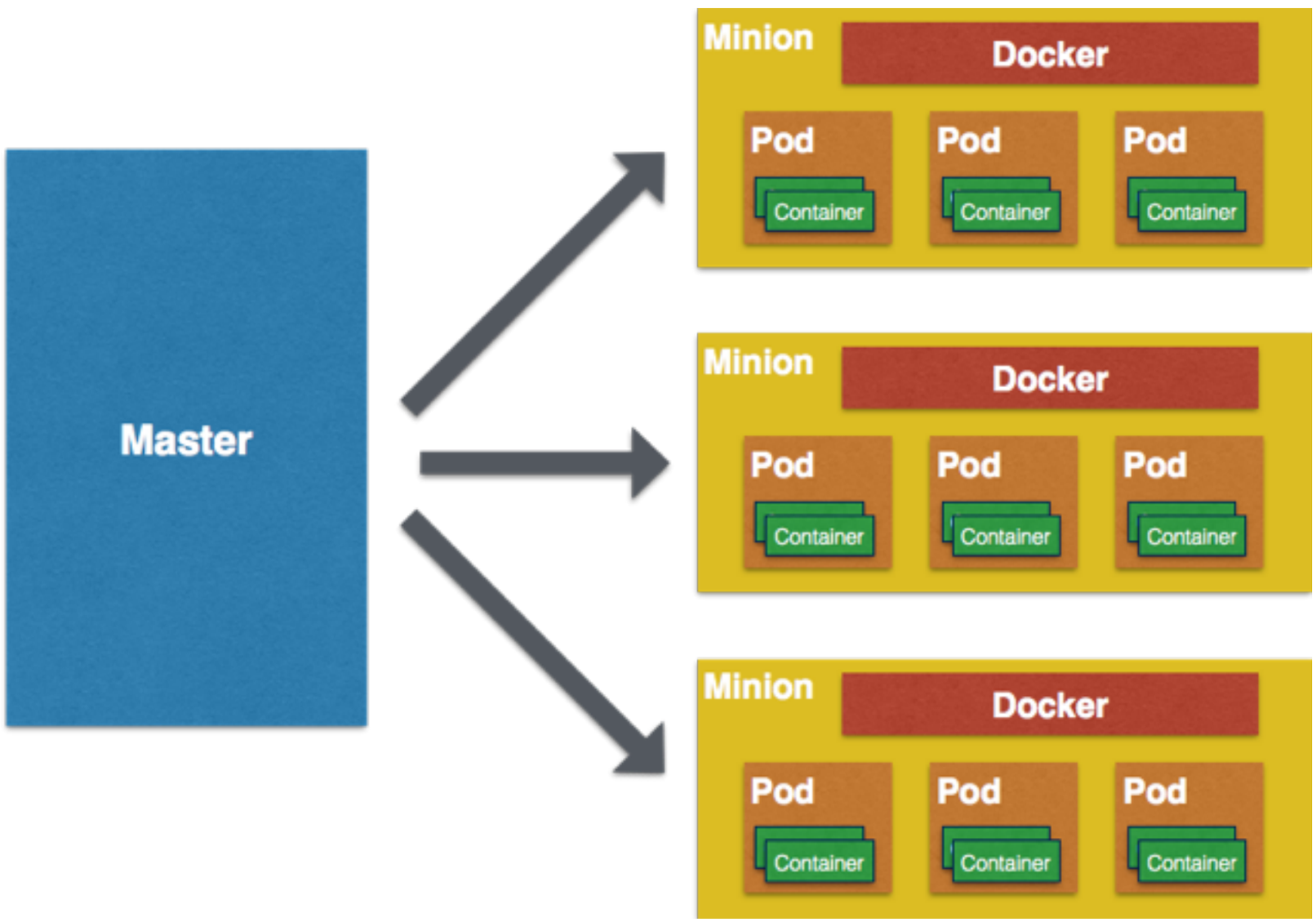
`ps`	Show all containers
`images`	Show all images
`run`	Create and run a container
`search`	Seaarch for images on a registry
`pull`	Dowmload of images
`rm`	Remove container
`rmi`	Remove image

Kubernetes

Facts

» Open Source orchestration platform for Docker containers
  – Rewrite of Google's internal framework "Borg"
» Declarative specification of a desired state
» Self-healing
» Service discovery
» Scheduling across hosts
» Simple replication

Architecture



Concepts

- **Pods**
  - Collection of one or more Docker containers
- **Replication Controller**
  - Creates and takes care of Pods
- **Services**
  - Proxy for a collection of Pods
- **Labels**
  - Grouping and organisation of Objects

Pod

» Collection of Docker containers running on the same host.

» Pods have a unique IP

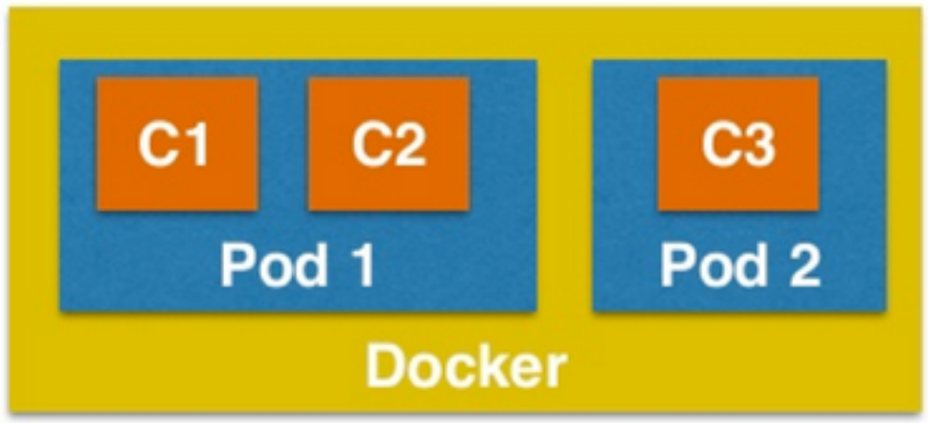
» Containers in a Pod ….

– …. share the same IP

– …. can reach each other via local ports

– …. can share data via volumes

» Pods can have one or more ***Labels***



Replication Controller

» Controls Pods selected by **Labels**

» Ensures that a specified number of Pod replicas is running

» Holds Pod Templates for creating new Pods

» Autoscaling

» Rolling Updates

```
{
  "kind":"ReplicationController",
  "apiVersion":"v1beta3",
  "metadata":{
     "name":"redis-master",
     "labels":{
        "name":"redis-master"
     }
  },
  “spec":{
     "replicas":1,
     "selector":{
        "name":"redis-master"
     },
     "template": …
  }
}
```

```
"template":{
   "metadata":{
      "labels":{
         "name":"redis-master"
      }
   },
   "spec":{
      "containers":[{
          "name":"master",
          "image":"redis",
          "ports":[{
             "containerPort":6379,
             "protocol":"TCP"
          }]
      }]
   }
}
```

Service

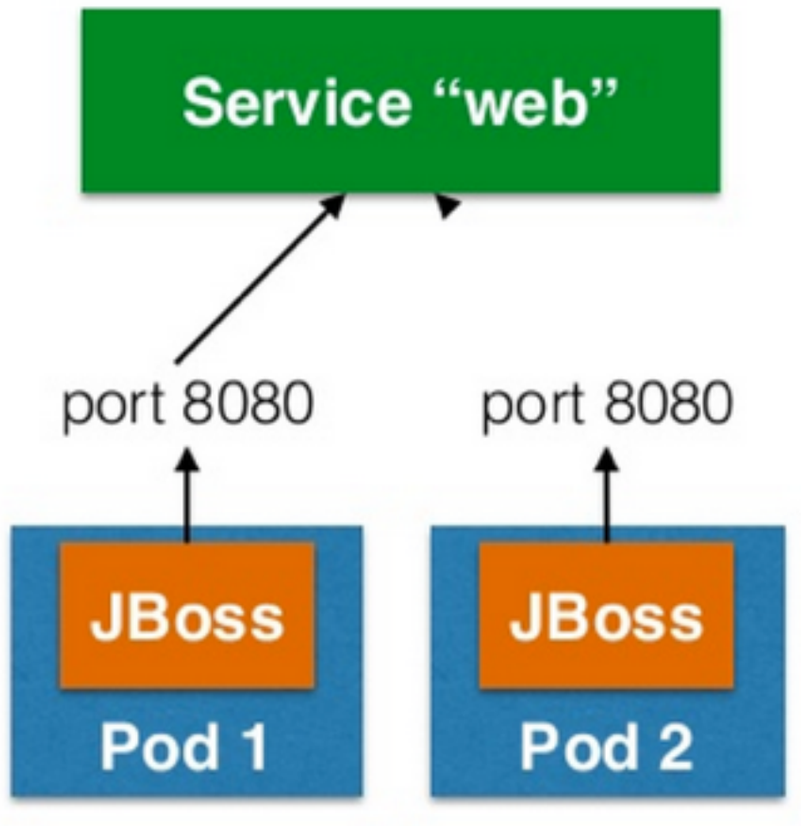
» View on a set of Pods with single IP address and port

» Pods are selected by **Label**

» Services are referenced by environment variables

» Service addresses stay stable

– Pods come and go (with different IPs)



```
{
    "kind":"Service",
    "apiVersion":"v1beta3",
    "metadata":{
        "name":"redis-master",
        "labels":{
            "name":"redis-master"
        }
    },
    "spec":{
        "ports": [{
            "port":6379,
            "targetPort":6379,
            "protocol":"TCP"
          }],
        "selector":{
            "name":"redis-master"
        }
    }
}
```

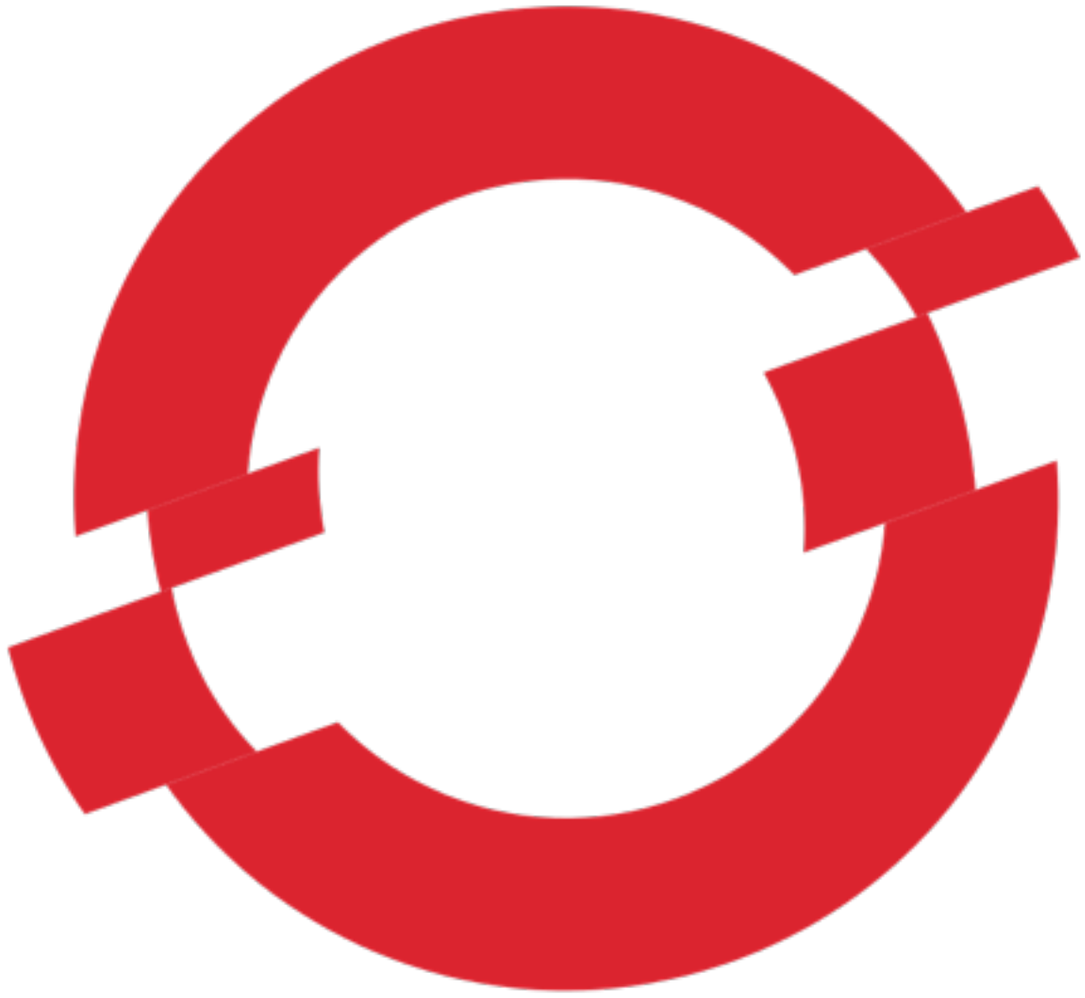
kubectl

» CLI for managing Kubernetes

- `kubectl <sub-command> …`

`get pods`
`get services`
`get rc`
	Show pods/service/replication controllers

`create`	Create objects
`update`	Update objects
`delete`	Delete objects
`resize`	New size for a RC



OPENSHIFT

History

- 2011: Platform-as-a-Service (PaaS) from Red Hat
- Three variants:
  - Online - Public PaaS
  - Enterprise - Private PaaS
  - Origin - Community PaaS
- OpenShift V3: Complete rewrite on basis of Kubernetes

Features

- Adds the “Build” aspect to Kubernetes
- Developer and Operation Tools
- Application Component Libraries
- Infrastructure Services
  - Registry, Router, OAuth2 Security
- Team and user isolation (multi-tenancy)
- Management UI

Builds

» Extension for **building** images

» Docker Builds

– Build images get access to enclosing Docker daemon.

» Source-To-Image

– Assembly of new image from a *builder image* and *source code*

– Often combined with a Webhook for automatic builds

Templates

» Templates allow the specifications of replication controller, services, …

» Parameter slots can be filled in …

– from the CLI wit `osc process`

– from the User Interface

» might become a Kubernetes feature in the future

Templates

```
{
  "apiVersion": "v1beta1",
  "kind": "Template",
  "metadata": {
    "name": "Template_Name",
    "annotations": {
      "description": "Description"
    }
  },
  "parameters": [{
      "name": "username"
      "value": "admin"
      "description": "administrative user"
    }],
  "labels": {
    "custom_label": "Label_Name"
  },
  "items": [{
      ...
    }]
}
```

Deployments

- Update of a replication controller's pod template
  - based on triggers
    - image change
    - configuration change
  - custom deployment strategies
  - rollback support
  - replication scaling

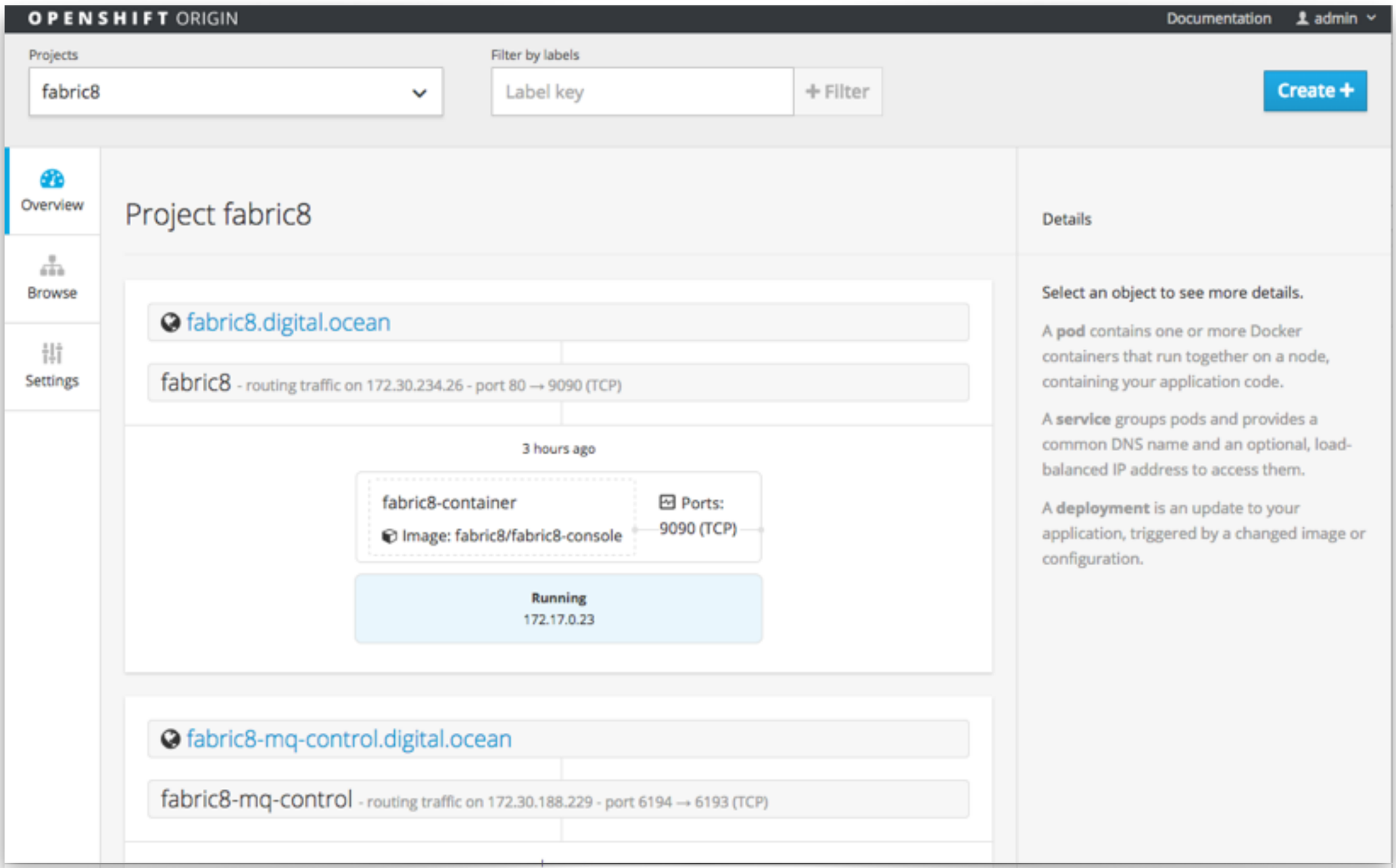
Registry

- OpenShift provides an own Docker registry as service
- OpenShift projects are mapped to registry user
  - e.g. for an image “fabric8/console” to be pushed there must exist a OpenShift project “fabric8”

Router

- External DNS mapping to services
  - based on HAProxy
- Different types of TLS termination
  - **edge** : TLS terminates at the router
  - **passthrough**: TLS stream is handle through to the service
  - **re-encryption**: TLS terminates at the router and is re-encrypted for the service

Web-Console


OPENSHIFT ORIGIN
Documentation
admin
Projects
fabric8
Filter by labels
Label key
+ Filter
Create +
Overview
Browse
Settings
Project fabric8
Details
fabric8.digital.ocean
fabric8 - routing traffic on 172.30.234.26 - port 80 → 9090 (TCP)
3 hours ago
fabric8-container
Image: fabric8/fabric8-console
Ports:
9090 (TCP)
Running
172.17.0.23
fabric8-mq-control.digital.ocean
fabric8-mq-control - routing traffic on 172.30.188.229 - port 6194 → 6193 (TCP)
Select an object to see more details.
A pod contains one or more Docker containers that run together on a node, containing your application code.
A service groups pods and provides a common DNS name and an optional, load-balanced IP address to access them.
A deployment is an update to your application, triggered by a changed image or configuration.

`osc`

» OpenShift CLI

» Extension to `kubectl`

`process`	Process Templates
`project`	Change namespace/project
`get routes`	Show created routes
`port-forward`	Port forwarding into pod
`exec`	Execute process in running pod

fabric8

fabric8

» Tools and Services for value add to Kubernetes and OpenShift

– **Management**: console, logging, metrics, …
– **Continous Delivery** Workflow
– **iPaaS**: Camel route visualisation, API registry, Messaging as a Service, …
– **Tools**: Kubernetes/OpenShift build integration, Kubernetes component test support, CDI extensions

History

- **Fuse ESB**: Open Source integration platform by FuseSource
- **Fabric**: Extension for managing many ESBs
- Red Hat acquired FuseSource in 2012
  - Fuse ESB ⇒ JBoss Fuse
  - Fabric (closed) ⇒ fabric8 (open source)

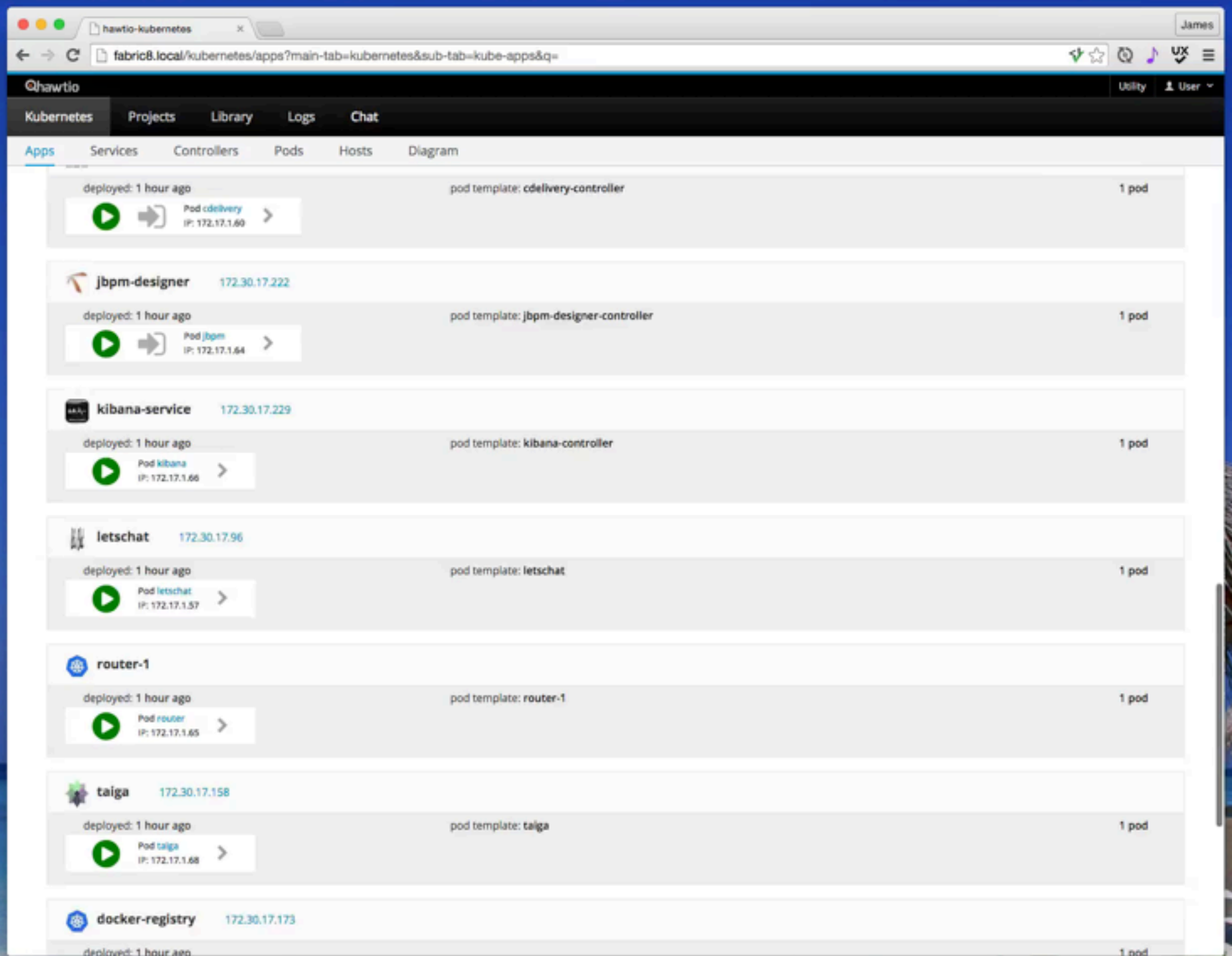
- **fabric8 1.x** is based on **Zookeeper** as central view of the system
  - JBoss Fuse 6.1: fabric8 1.0
  - JBoss Fuse 6.2: fabric8 1.2.x
- **fabric8 2.x** sits on top of **Kubernetes**
  - fabric8 1.x functionality became *Jube,* a pure Java implementation of the Kubernetes API

Management

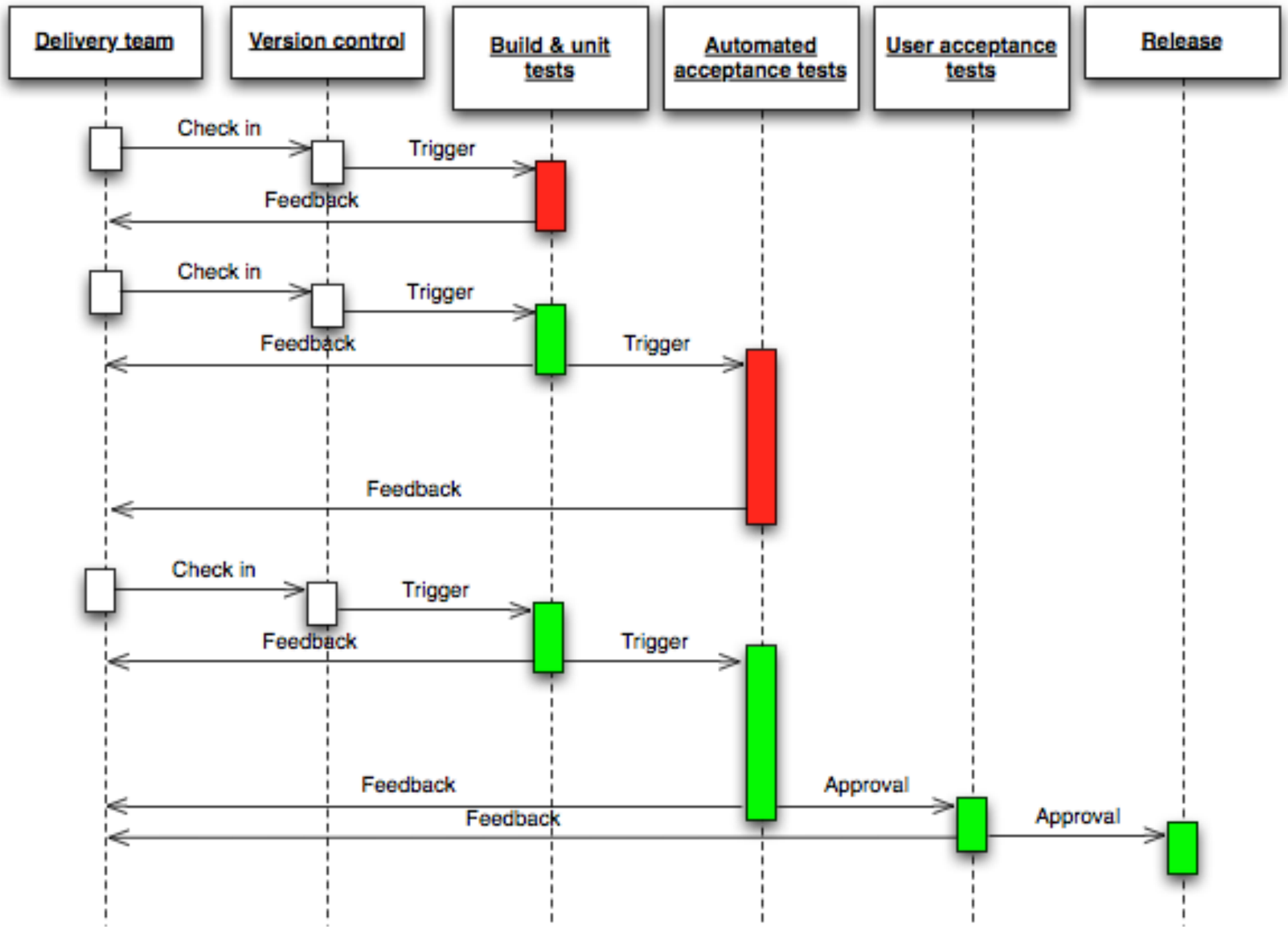
- Web console for Kubernetes
  - Starting/Stopping of pods
  - Changing Replicas
  - Individual management of pods
  - based on hawt.io

redhat.

DevOpsCon



Continous Delivery



iPaas

- **Console** for visualising and working with integration services
  - e.g. showing the Camel routes
- **API registry** for a global view of all RESTful and WebServices
- **MQ** provides *Messaging as a Service*
  - based on ActiveMQ
  - allows autoscaling

Tools

- **fabric8-maven-plugin**
  - Creates and apply Kubernetes descriptors out of build informations
  - Creates OpenShift routes
  - Deploys `kubernetes.json` as Maven artefacts

Tools

» **Arquillian extension** for testing
  – Provision containers to Kubernetes
  – Separate namespace per test (isolation)
  – Annotations for injecting Kubernetes objects
  – Assertions on Kubernetes objects
» **Java Libraries**
  – Access to Kubernetes API
  – CDI injections of Kubernetes Services
  – ....

Summary

- **Docker** is the perfect foundation for a container based infrastructure
- **Kubernetes** is a powerful Docker orchestration platform backed with great momentum
- **OpenShift** as a PaaS adds the “Build” dimension to Kubernetes
- **Fabric8** adds services and Java tooling to Docker, Kubernetes and OpenShift