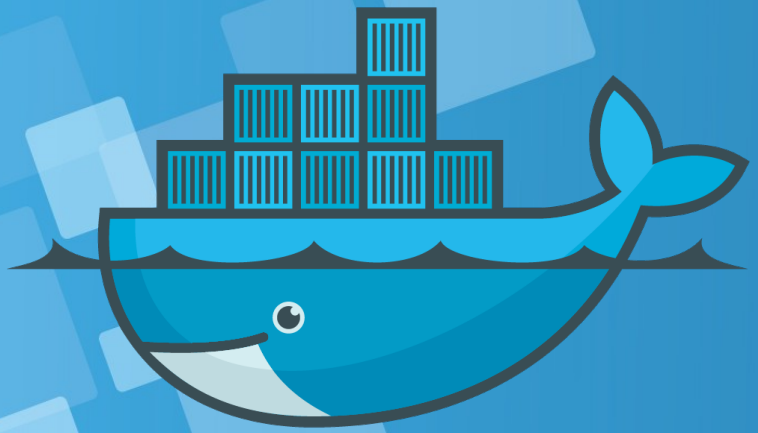
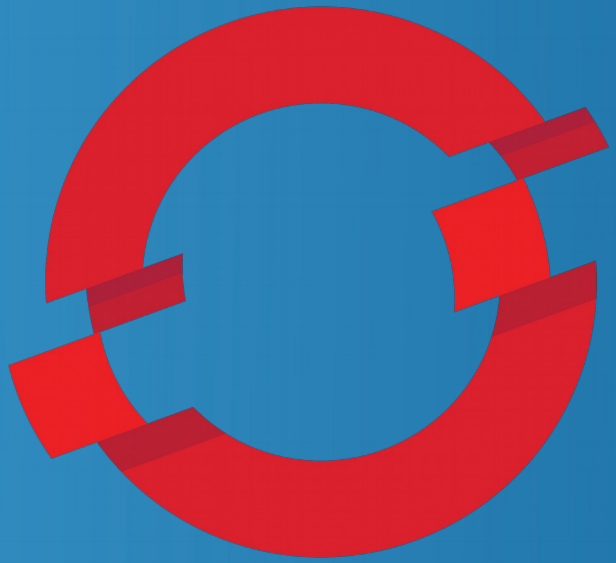


docker



OPENSHIFT

Docker Meetup 20 Apil 2017
Openshift on ~~*Production*~~ Development

Yusuf Hadiwinata Sutandar
LinuxGeek,OpenSourceEnthusiast,SecurityHobbies

Docker
User Group
Indonesia

Docker Meetup

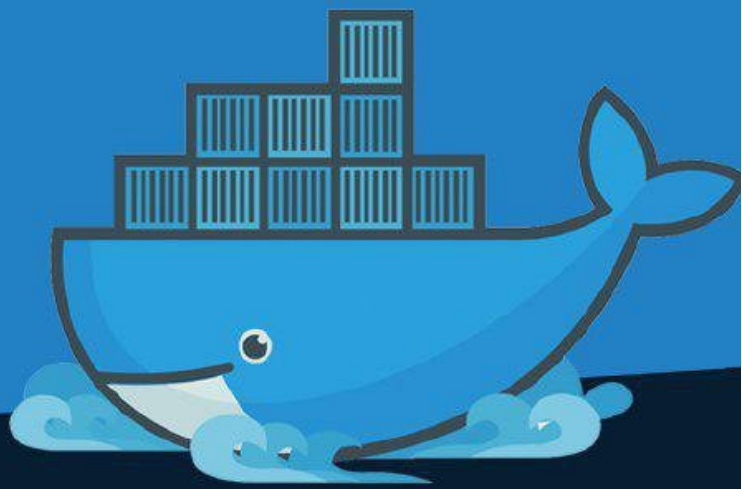
Docker is hype and phenonemon, lets find out the new technologies of docker management, build your agile docker system will be revealed on this event.

Registration:
http://bit.do/dockerid

Guest Speaker:

OPENSHIFT on Production

Yusuf Hadiwinata Sutandar
PT. Inovasi Informatika Indonesia

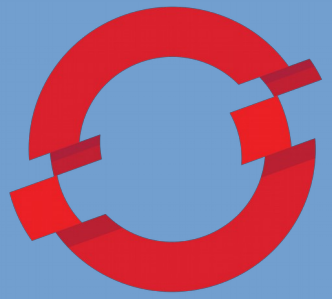


Thursday, April 20, 2017
06:00 PM - Drop
MidPlaza 2, 15 Floor
Jln Jendral Sudirman Kav 10-11
Jakarta Pusat

Sponsored by : Biznet GioCloud

Agenda

- Container, Docker, Kubernetes & Openshift Introduction
- Openshift Installation
- Docker Orchestration using Openshift
- Auto-Scaling using Openshift
- Source to Image deployment
- Pipeline for CI/CD



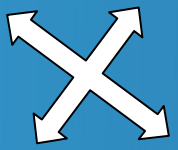
OPENSHIFT

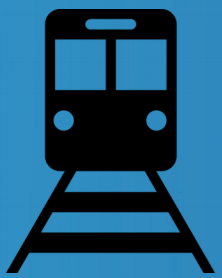
Brief Intro to Container & Docker

History of Container
Docker Introduction

The Problem

Cargo Transport 1960s





Solution?

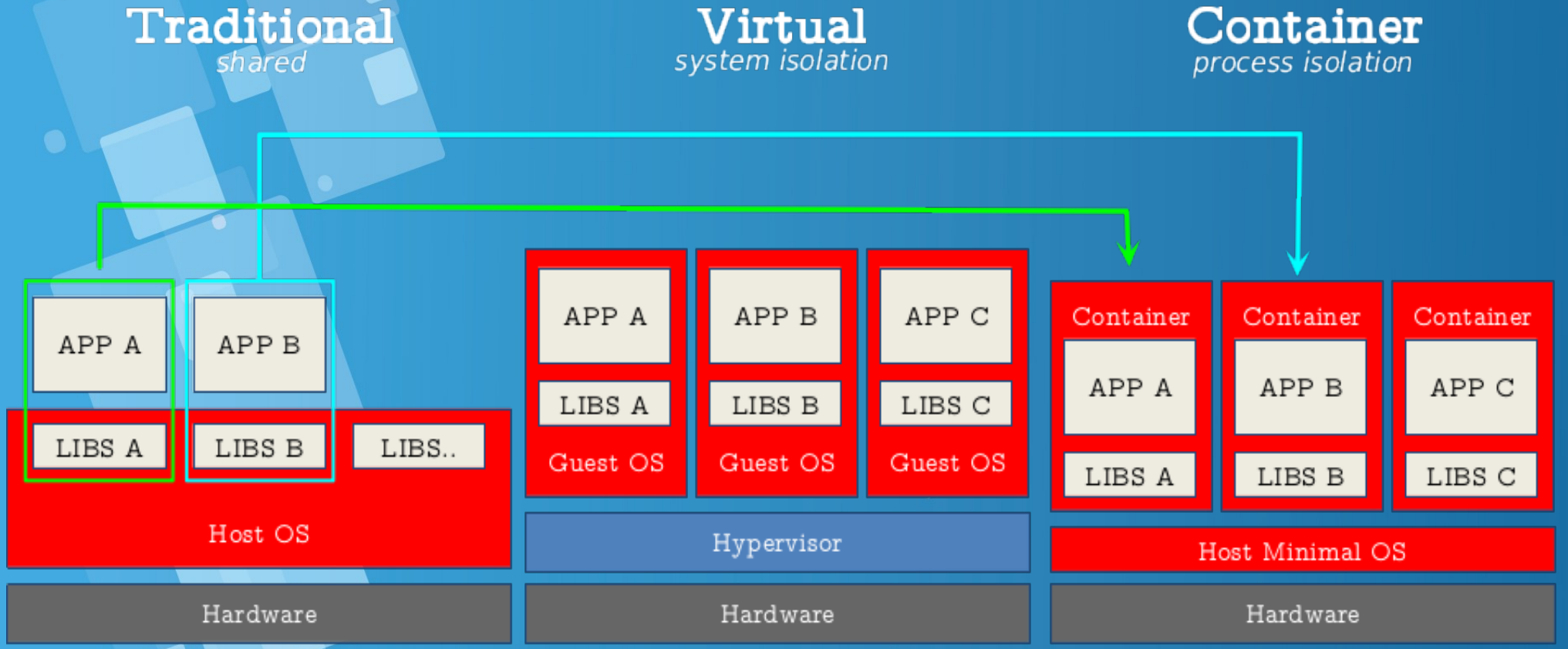
Intermodal Shipping Container

The Solution

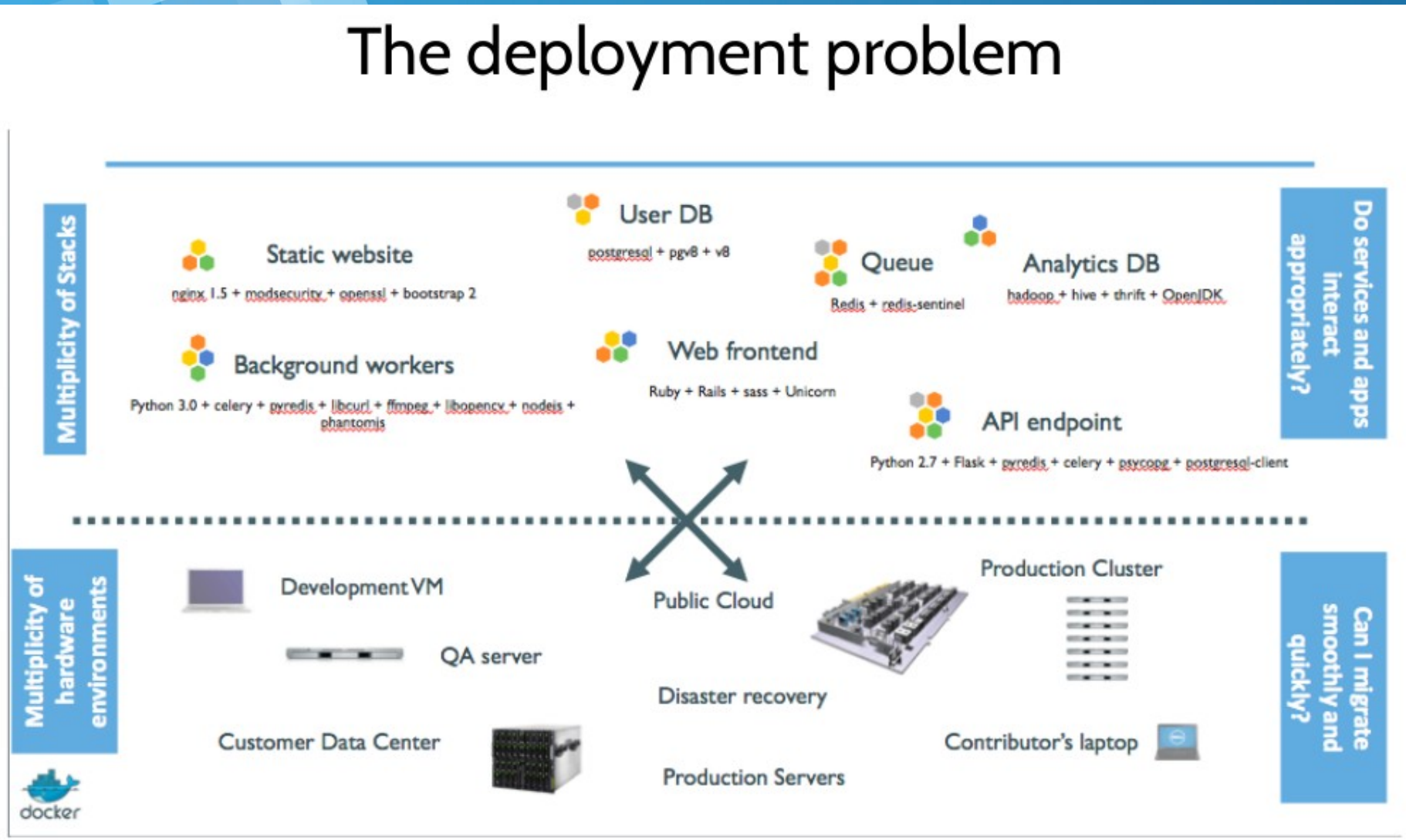
90% of all cargo now shipped in a standard container

Order of magnitude reduction in cost and time to load and unload ships, trains, trucks

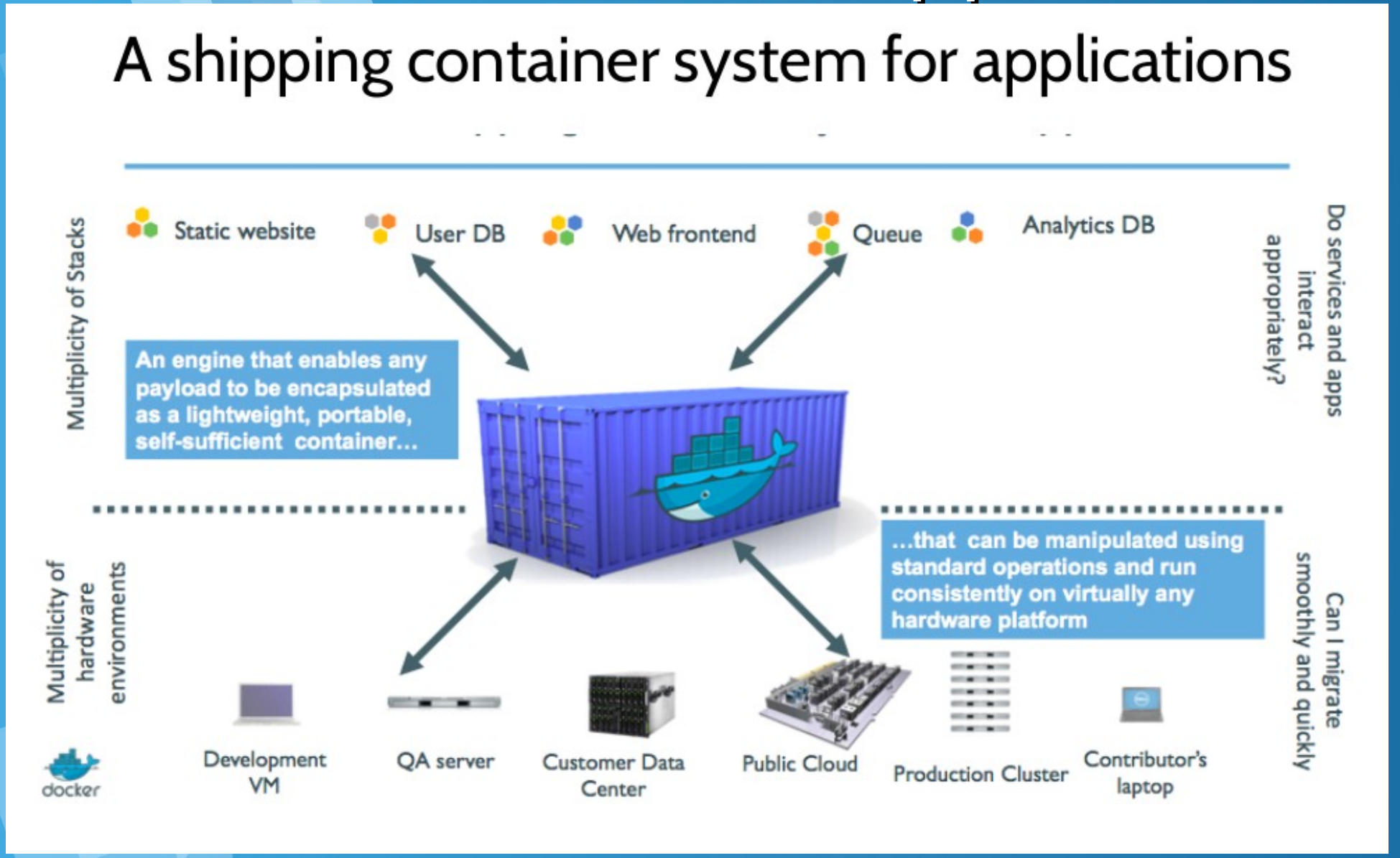
The Evolution



The App Problem



The App Solution



Container Technology

One way of looking at containers is as improved chroot jails. Containers allow an operating system (OS) process (or a process tree) to run isolated from other processes hosted by the same OS. Through the use of Linux kernel namespaces, it is possible to restrict a process view of:

- Other processes (including the pid number space)
- File systems
- User and group IDs
- IPC channels
- Devices
- Networking

Container Technology

Other Linux kernel features complement the process isolation provided by kernel namespaces:

- Cgroups limit the use of CPU, RAM, virtual memory, and I/O bandwidth, among other hardware and kernel resources.
- Capabilities assign partial administrative faculties; for example, enabling a process to open a low

  network port (<1024) without allowing it to alter routing tables or change file ownership.
- SELinux enforces mandatory access policies even if the code inside the container finds a way to

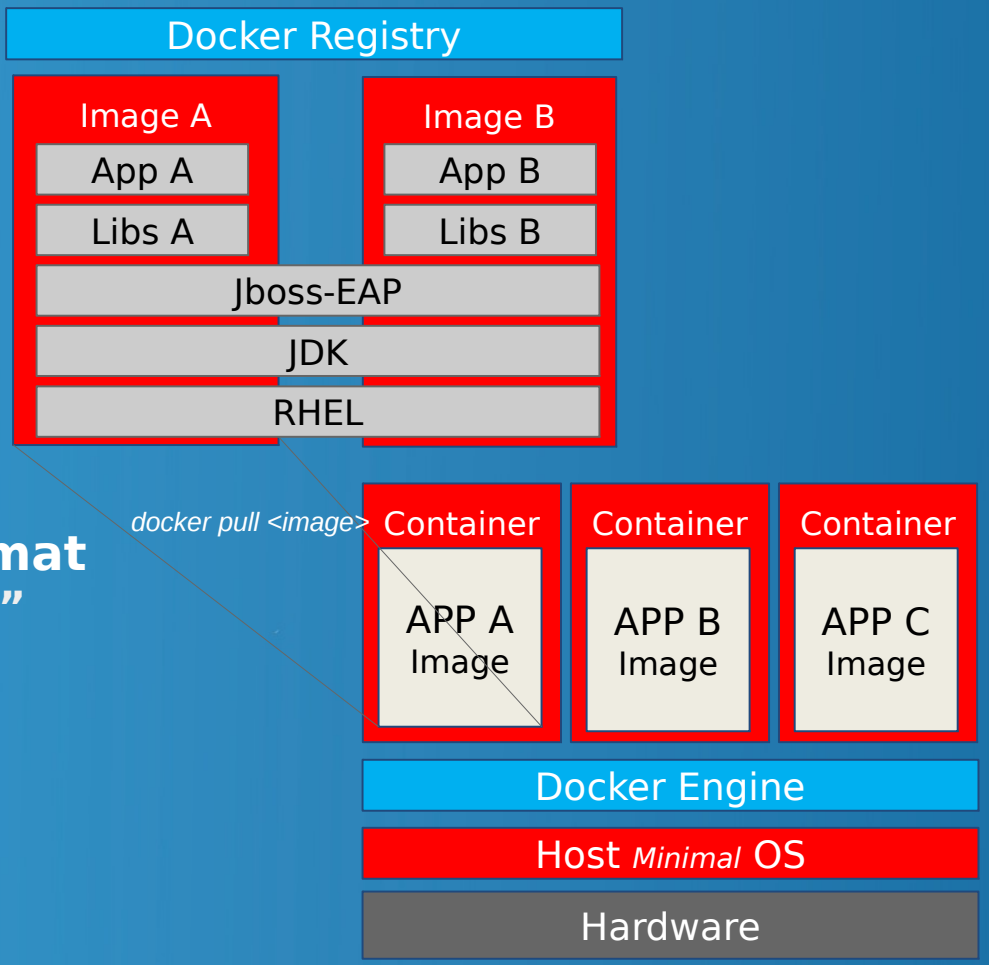
  break its isolation

Container Technology

Images & Containers

- **Docker “Image”**
  - **Unified Packaging format**
    - **Like “war” or “tar.gz”**
    - **For any type of Application**
    - **Portable**

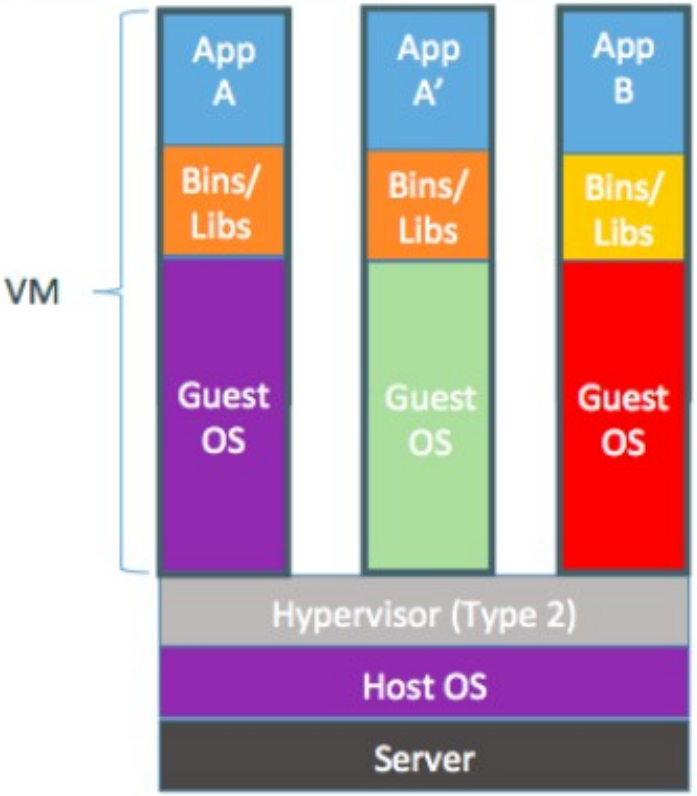
- **Docker “Container”**
  - **Runtime**
  - **Isolation**



Container Solution

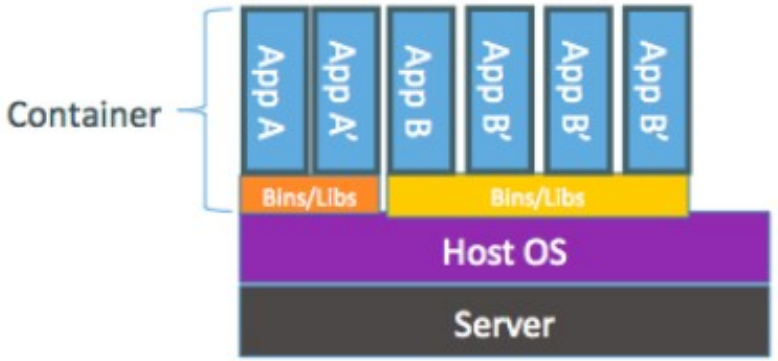
containers as lightweight VMs

Less overhead!



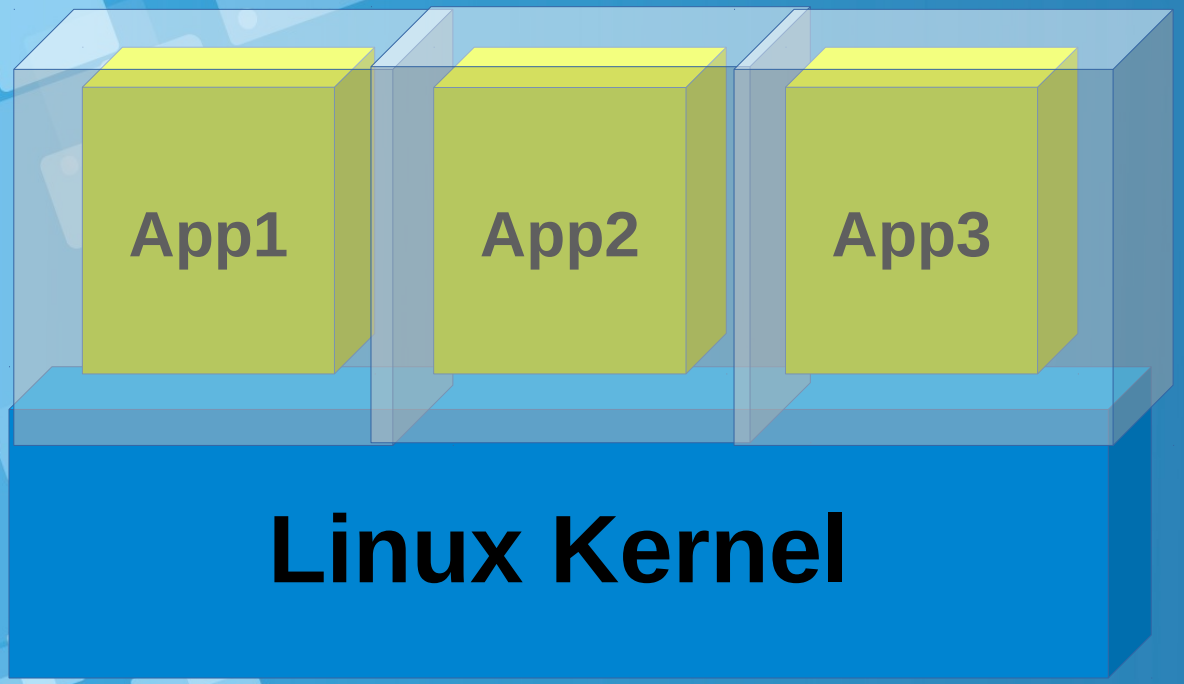
Containers are isolated, but share OS kernel and, where appropriate, bins/libraries

...result is significantly faster deployment, much less overhead, easier migration, faster restart



Is not Virtualizaiton :)

Isolation, not Virtualization



- Kernel Namespaces
  - Process
  - Network
  - IPC
  - Mount
  - User
- Resource Limits
  - Cgroups
- Security
  - SELinux

Container Solution

Virtual Machine and Container Complement each other

Virtual Machine

- Virtual machines include the application, the necessary binaries and libraries, and an entire guest operating system
- Each Guest OS has its own Kernel and user space

Containers

- Containers run as isolated processes in user space of host OS
- They share the kernel with other container (container-processes)
- Containers include the application and all of its dependencies
- Not tied to specific infrastructure

Container Problem

Containers before Docker

- No standardized exchange format.

(No, a rootfs tarball is not a format!)

- Containers are hard to use for developers.

(Where's the equivalent of docker run debian?)

- No re-usable components, APIs, tools.

(At best: VM abstractions, e.g. libvirt.)

Analogy:

- Shipping containers are not just steel boxes.
- They are steel boxes that are a standard size, with the same hooks and holes

Docker Solution

Containers after Docker

- Standardize the container format, because containers were not portable.
- Make containers easy to use for developers.
- Emphasis on re-usable components, APIs, ecosystem of standard tools.
- Improvement over ad-hoc, in-house, specific tools.

What IT`s Said about Docker:

Developer Say:

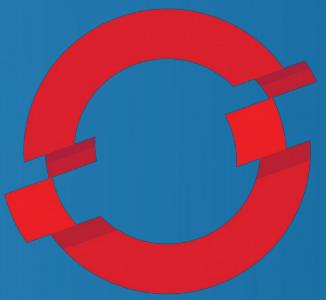
Build Once, Run Anywhere

Operator: Configure Once, Run Anything

Docker - Container Problems

We need more than just packing and isolation

- **Scheduling** : Where should my containers run?
- **Lifecycle and health** : Keep my containers running despite failures
- **Discovery** : Where are my containers now?
- **Monitoring** : What's happening with my containers?
- **Auth{n,z}** : Control who can do things to my containers
- **Aggregates** : Compose sets of containers into jobs
- **Scaling** : Making jobs bigger or smaller



OPENSHIFT

Kubernetes is a Solution?

Kubernetes – Container Orchestration at Scale

Greek for “Helmsman”; also the root of the word “Governor” and “cybernetic”

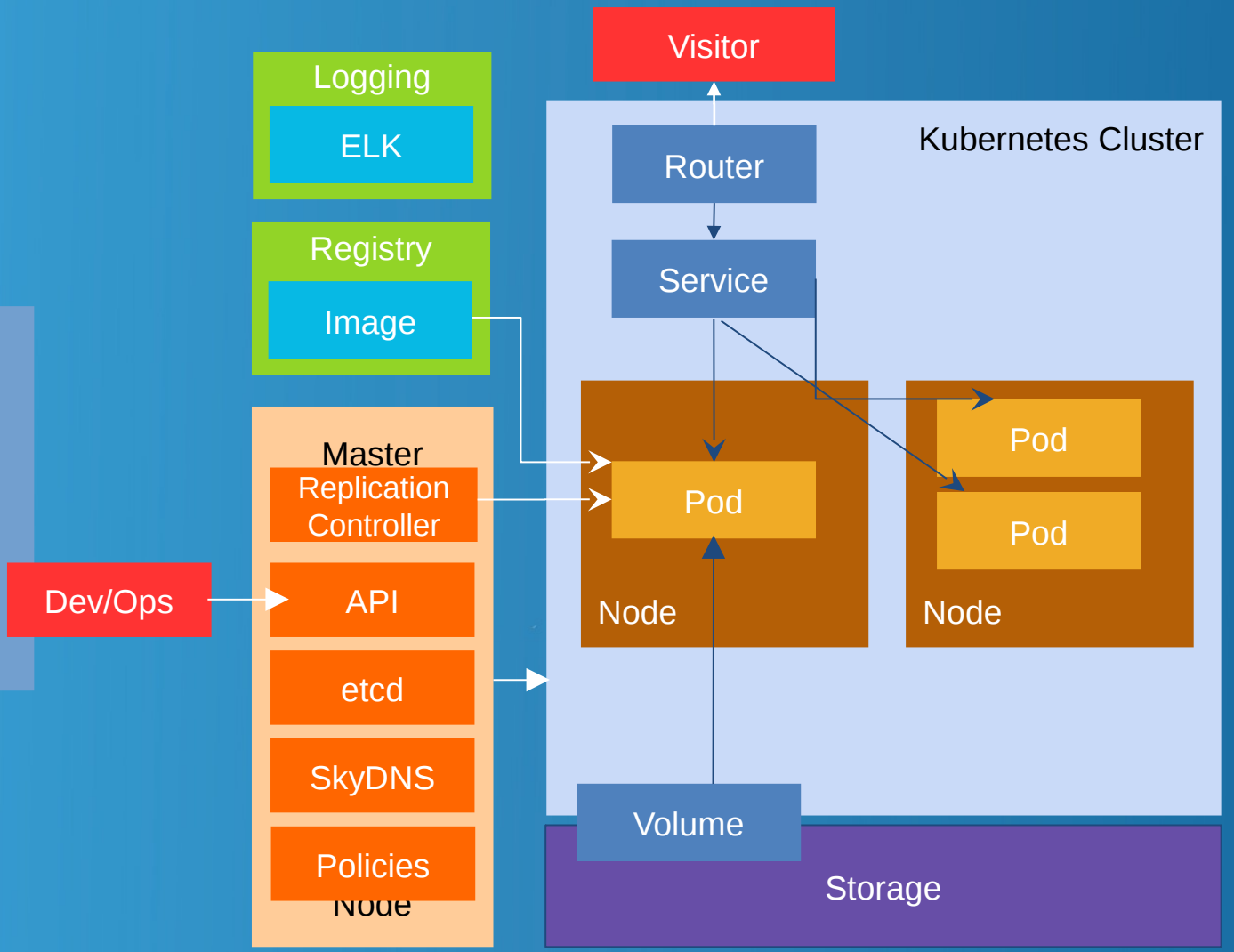
- **Container Cluster Manager**
  - **Inspired by the technology that runs Google**
- **Runs anywhere**
  - **Public cloud**
  - **Private cloud**
  - **Bare metal**
- **Strong ecosystem**
  - **Partners: Red Hat, VMware, CoreOS..**
  - **Community: clients, integration**

Kubernetes Solution Detail

Core Concepts

Pod

- **Labels & Selectors**
- **ReplicationController**
- **Service**
- **Persistent Volumes**



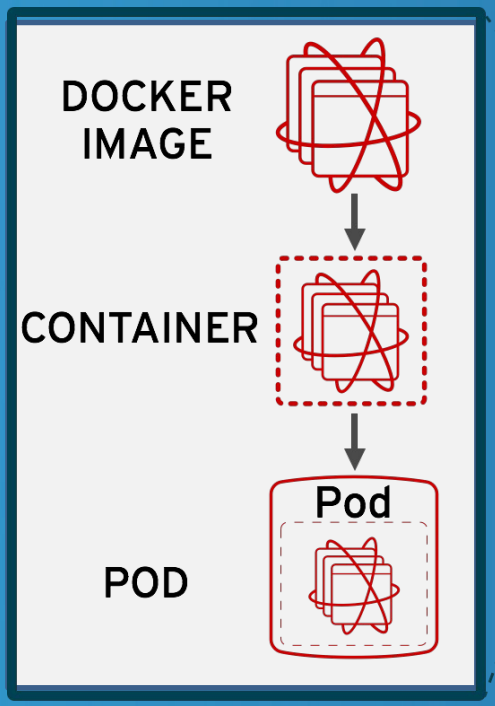
Kubernetes: The Pods

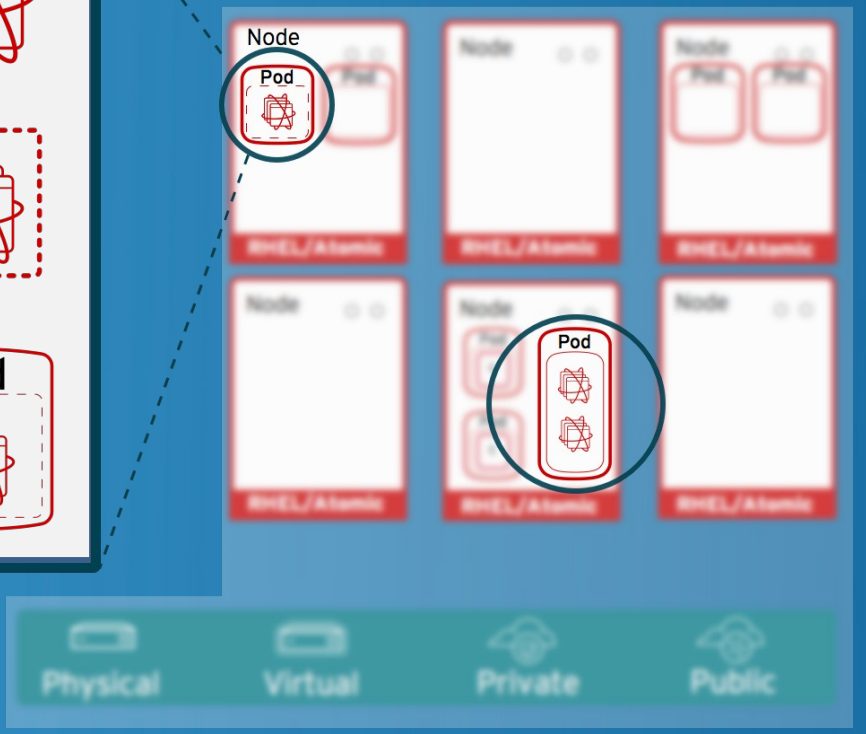
POD Definition:

- **Group of Containers**
- **Related to each other**
- **Same namespace**
- **Emphemeral**

Examples:

- **Wordpress**
- **MySQL**
- ~~**Wordpress + MySQL**~~
- **ELK**
- **Nginx+Logstash**
- **Auth-Proxy+PHP**
- **App + data-load**





Kubernetes: Building Pod

```
{
  "apiVersion": "v1",
  "kind": "Pod",
  "metadata": {
    "name": "hello-openshift"
  },
  "spec": {
    "containers": [
      {
        "name": "hello-openshift",
        "image": "openshift/hello-openshift",
        "ports": [
          {
            "containerPort": 8080
          }
        ]
      }
    ]
  }
}
```

- OpenShift/Kubernetes runs containers inside Kubernetes pods, and to create a pod from a container image, Kubernetes needs a pod resource definition. This can be provided either as a **JSON** or **YAML** text file, or can be generated from defaults by oc new-app or the web console.
- This JSON object is a pod resource definition because it has attribute "**kind**" with value "**Pod**". It contains a single "**container**" whose name is "**hello-openshift**" and that references the "**image**" named "**openshift/hello-openshift**". The container also contains a single "**ports**", which listens to TCP port **8080**.

```
# kubectl create –f hello-openshift.yaml
```

```
# oc create –f hello-openshift.yaml
```

Kubernetes: List Pod

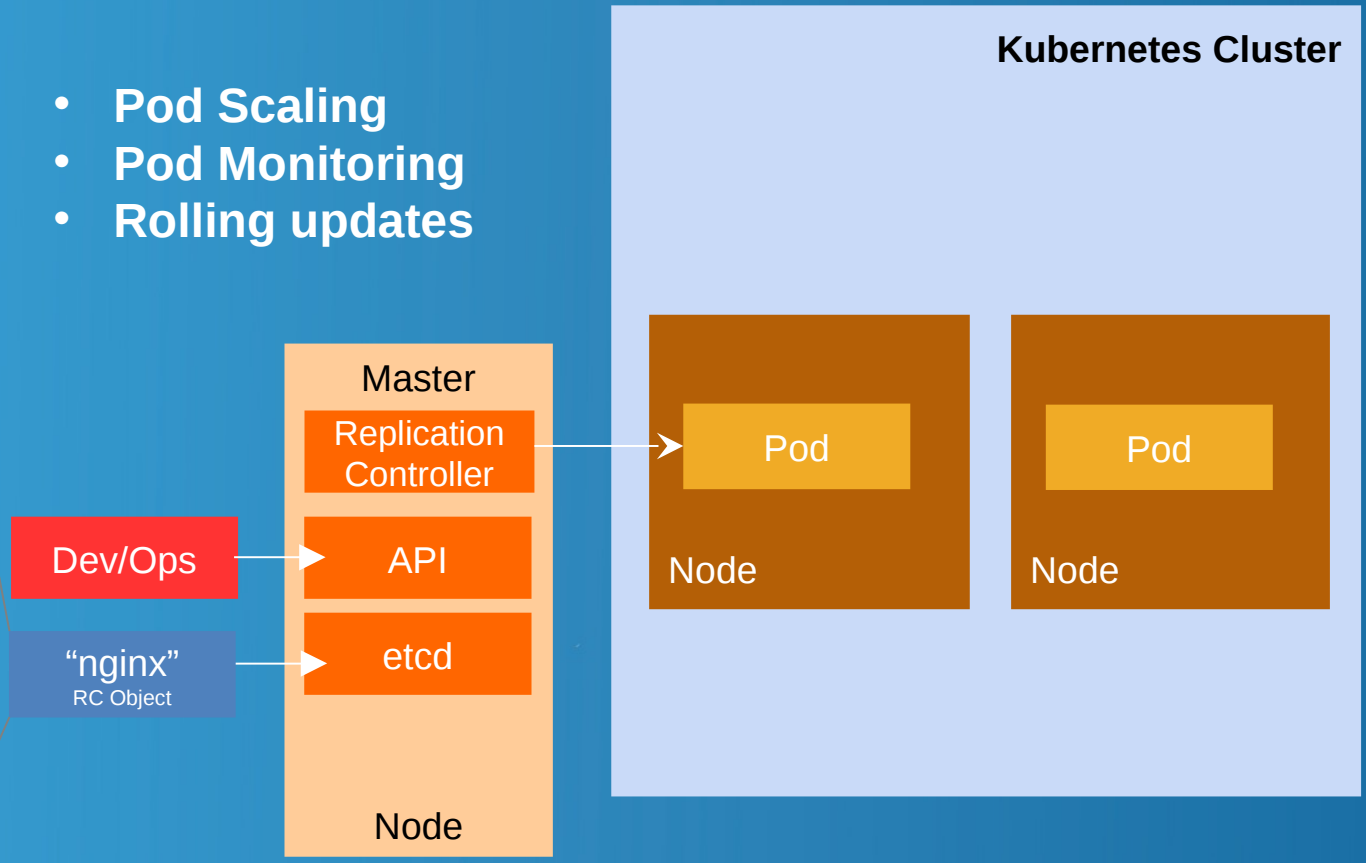
```
[root@centos-16gb-sgp1-01 ~]# oc get pod
NAME                          READY     STATUS      RESTARTS   AGE
bgdemo-1-build                0/1       Completed   0          16d
bgdemo-1-x0wlq                1/1       Running     0          16d
dc-gitlab-runner-service-3-wgn8q   1/1       Running     0          8d
dc-minio-service-1-n0614      1/1       Running     5          23d
frontend-1-build              0/1       Completed   0          24d
frontend-prod-1-gmcrw         1/1       Running     2          23d
gitlab-ce-7-kq0jp             1/1       Running     2          24d
hello-openshift               1/1       Running     2          24d
jenkins-3-8grrq               1/1       Running     12         21d
os-example-aspnet-2-build     0/1       Completed   0          22d
os-example-aspnet-3-6qncw     1/1       Running     0          21d
os-sample-java-web-1-build    0/1       Completed   0          22d
os-sample-java-web-2-build    0/1       Completed   0          22d
os-sample-java-web-3-build    0/1       Completed   0          22d
os-sample-java-web-3-sqf41    1/1       Running     0          22d
os-sample-python-1-build      0/1       Completed   0          22d
os-sample-python-1-p5b73      1/1       Running     0          22d
```

Kubernetes: Replication Controller

- **Pod Scaling**
- **Pod Monitoring**
- **Rolling updates**

```
kind: ReplicationController
metadata:
   name: nginx
spec:
   replicas: 2
   selector:
      app: nginx
template:
   metadata:
      name: nginx
      labels:
         app: nginx
   spec:
      containers:
       - name: nginx
         image: nginx:v2.2
         ports:
          - containerPort: 80
```

```
# kubectl create –f nginx-rc.yaml
```



Kubernetes: Service

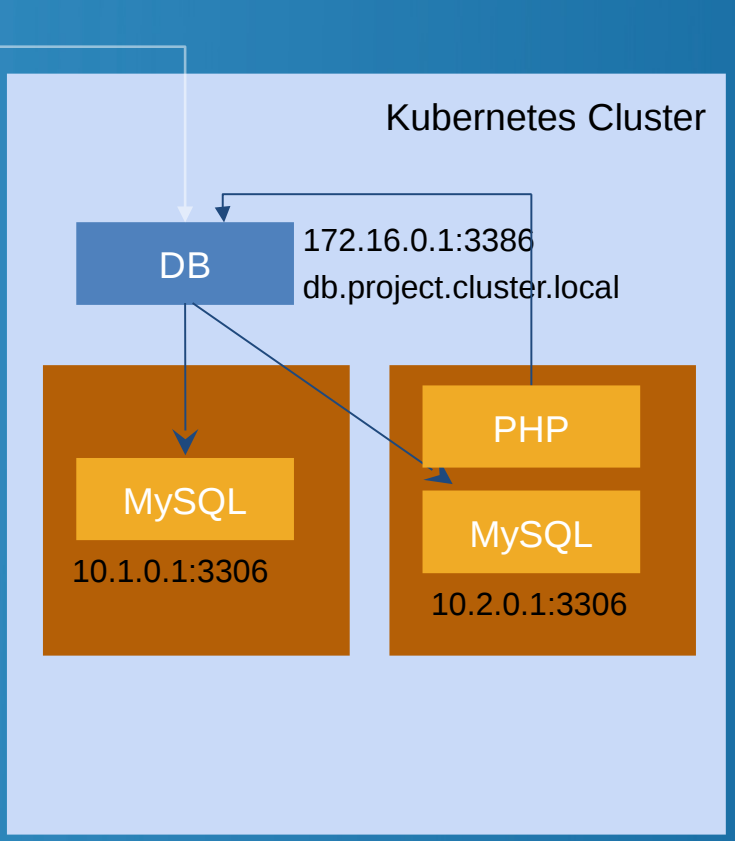
Service Definition:

- **Load-Balanced Virtual-IP *(layer 4)***
- **Abstraction layer for your App**
- **Enables Service Discovery**
  - **DNS**
  - **ENV**

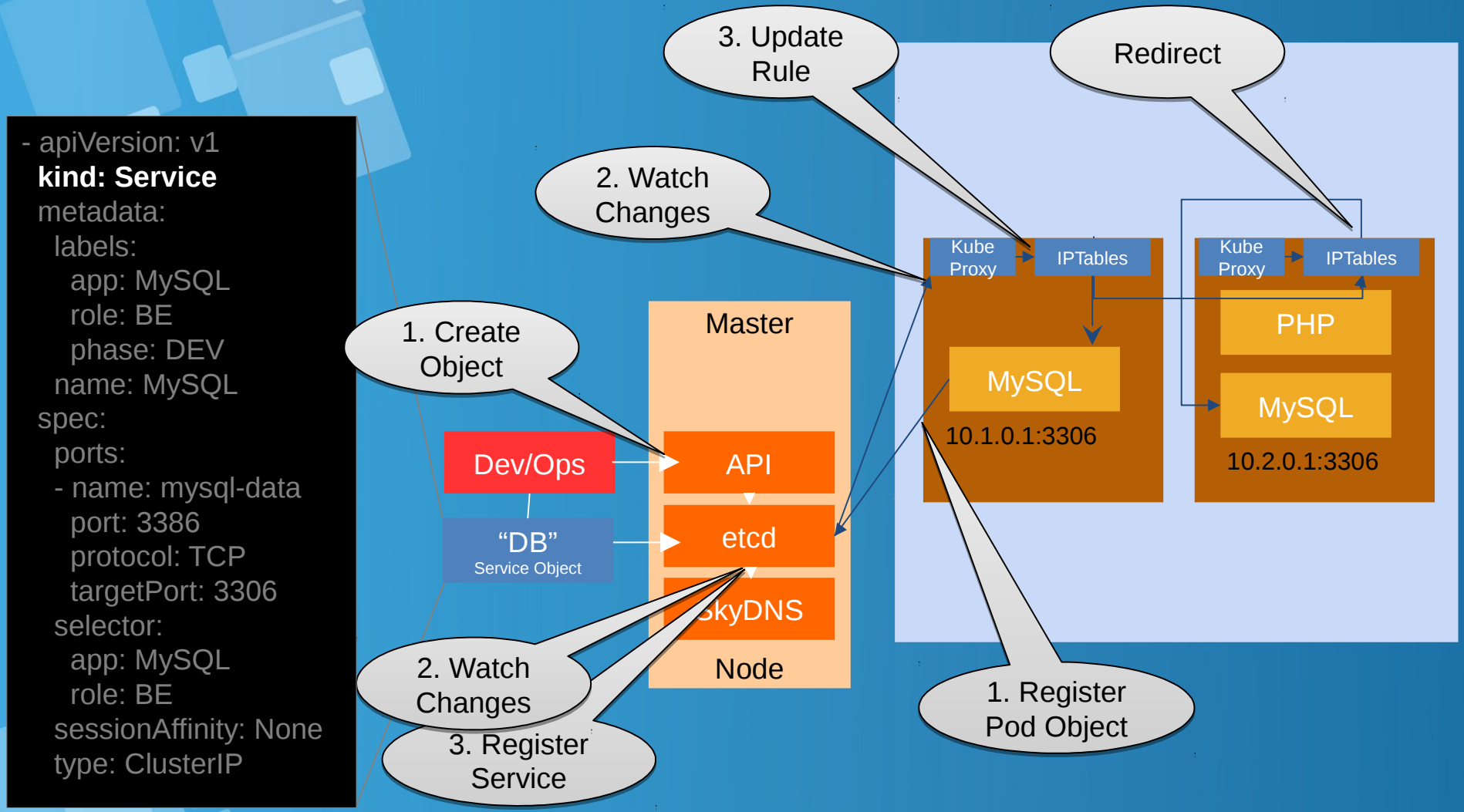
Examples:

- **frontend**
- **database**
- **api**

```
<?php
   mysql_connect(getenv("db_host"))
   mysql_connect("db:3306")
?>
```



Kubernetes: Service Continue..

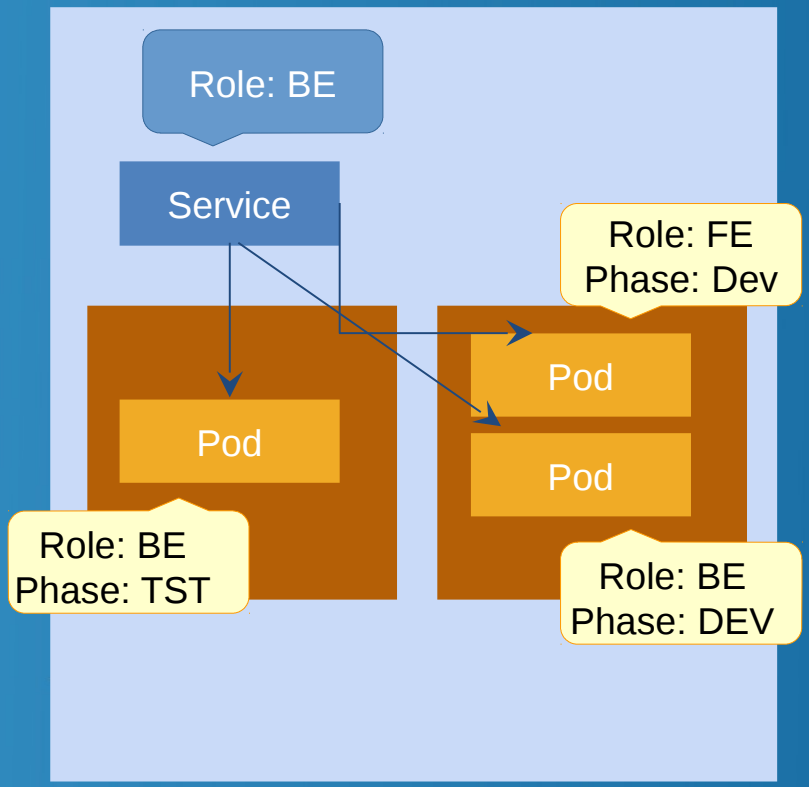


Kubernetes: Labels & Selectors

think SQL 'select ... where ...'

```
- apiVersion: v1
  kind: Service
  metadata:
    labels:
      app: MyApp
      role: BE
      phase: DEV
    name: MyApp
  spec:
    ports:
    - name: 80-tcp
      port: 80
      protocol: TCP
      targetPort: 8080
    selector:
      app: MyApp
      role: BE
    sessionAffinity: None
    type: ClusterIP
```

```
- apiVersion: v1
  kind: Pod
  metadata:
    labels:
      app: MyApp
      role: BE
      phase: DEV
    name: MyApp
```



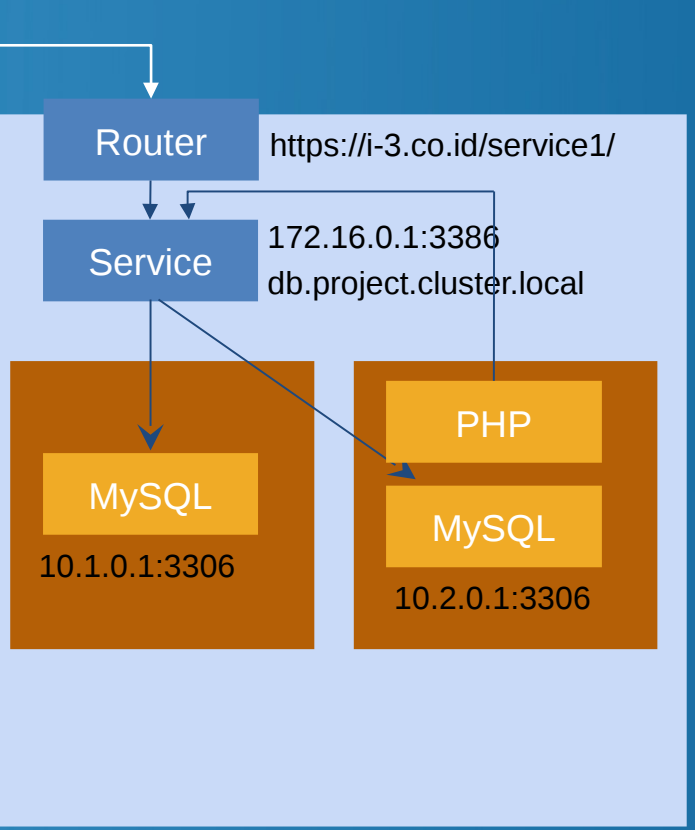
Kubernetes: Ingress / Router

- **Router Definition:**
  - **Layer 7 Load-Balancer / Reverse Proxy**
  - **SSL/TLS Termination**
  - **Name based Virtual Hosting**
  - **Context Path based Routing**
  - **Customizable (image)**
    - **HA-Proxy**
    - **F5 Big-IP**

Examples:

- **https://www.i-3.co.id/myapp1/**
- **http://www.i-3.co.id/myapp2/**

```
apiVersion:
extensions/v1beta1
kind: Ingress
metadata:
   name: mysite
spec:
   rules:
   - host: www.i-3.co.id
     http:
        paths:
        - path: /foo
          backend:
             serviceName: s1
             servicePort: 80
        - path: /bar
          backend:
             serviceName: s2
             servicePort: 80
```



Kubernetes: Router Detail

```
[root@centos-16gb-sgp1-01 ~]# oc env pod router-1-b97bv --list
# pods router-1-b97bv, container router
DEFAULT_CERTIFICATE_DIR=/etc/pki/tls/private
ROUTER_EXTERNAL_HOST_HOSTNAME=
ROUTER_EXTERNAL_HOST_HTTPS_VSERVER=
ROUTER_EXTERNAL_HOST_HTTP_VSERVER=
ROUTER_EXTERNAL_HOST_INSECURE=false
ROUTER_EXTERNAL_HOST_INTERNAL_ADDRESS=
ROUTER_EXTERNAL_HOST_PARTITION_PATH=
ROUTER_EXTERNAL_HOST_PASSWORD=
ROUTER_EXTERNAL_HOST_PRIVKEY=/etc/secret-volume/router.pem
ROUTER_EXTERNAL_HOST_USERNAME=
ROUTER_EXTERNAL_HOST_VXLAN_GW_CIDR=
ROUTER_SERVICE_HTTPS_PORT=443
ROUTER_SERVICE_HTTP_PORT=80
ROUTER_SERVICE_NAME=router
ROUTER_SERVICE_NAMESPACE=default
ROUTER_SUBDOMAIN=
STATS_PASSWORD=XXXXXX
STATS_PORT=1936
STATS_USERNAME=admin
```

- Check the router environment variables to find connection parameters for the HAProxy process running inside the pod

Kubernetes: Router-HAProxy

HAProxy

Statistics Report for pid 3301

> General process information

pid = 3301 (process #1, nbproc = 1)
uptime = 1d 14h17m37s
system limits: memmax = unlimited; ulimit-n = 40057
maxsock = 40057; maxconn = 20000; maxpipes = 0
current conns = 2; current pipes = 0/0; conn rate = 0/sec
Running tasks: 1/53; idle = 100 %

- active UP
- active UP, going down
- active DOWN, going up
- active or backup DOWN
- active or backup DOWN for maintenance (MAINT)
- active or backup SOFT STOPPED for maintenance
- backup UP
- backup UP, going down
- backup DOWN, going up
- not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:

- Primary site
- Updates (v1.5)
- Online manual

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	2	-	1	2	20 000	27 577			1 668 964	4 591 520	0	0	13 788					OPEN								
Backend	0	0		0	0		0	0	2 000	0	0	0s	1 668 964	4 591 520	0	0		0	0	0	0	1d14h UP		0	0	0		0		

public

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	3	-	1	2	20 000	33			37 357 332	27 189 019	0	0	6					OPEN								

public_ssl

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	18	-	0	7	20 000	1 175			2 245 759	19 381 416	0	0	0					OPEN								

be_sni

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
fe_sni	0	0	-	0	18		0	6	-	1 015	1 015	2m35s	2 059 121	6 125 257		0		0	0	0	0			1	Y	-				-
Backend	0	0		0	18		0	6	2 000	1 015	1 015	2m35s	2 059 121	6 125 257	0	0		0	0	0	0	1d14h UP		1	1	0		0	0s	

fe_sni

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	18	-	0	6	20 000	1 015			1 591 702	3 659 588	0			60				OPEN								

be_no_sni

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
fe_no_sni	0	0	-	0	1		0	1	-	6	6	14m46s	1 956	14 511		0									Y					-

HAPROXY
Powering Your Uptime

Kubernetes: Persistent Storage

For Ops:

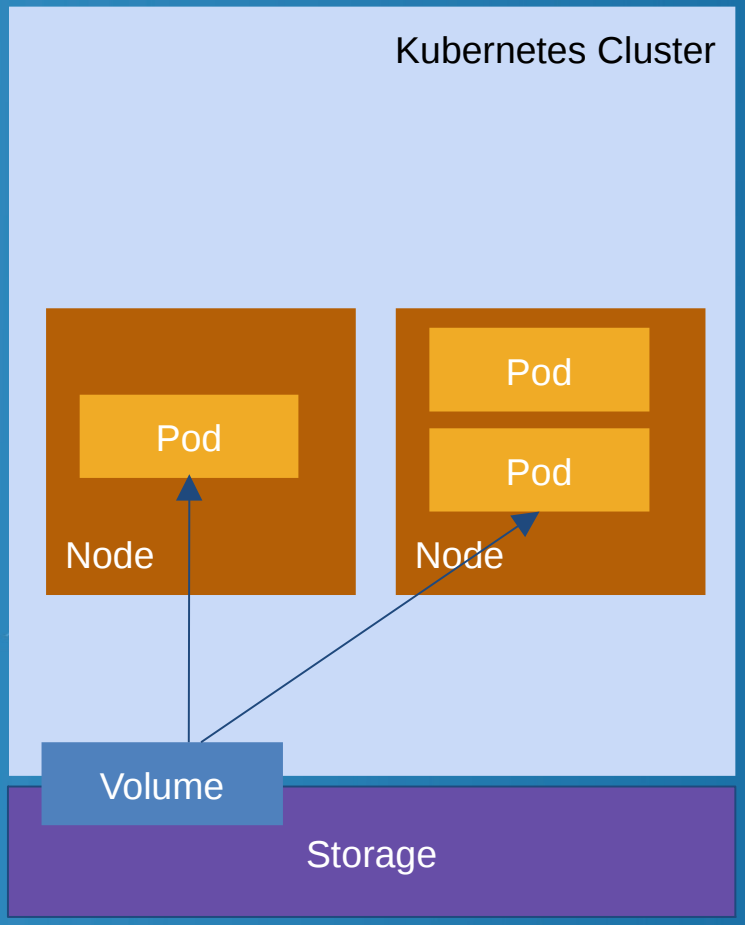
- **Google**
- **AWS EBS**
- **OpenStack's Cinder**
- **Ceph**
- **GlusterFS**
- **NFS**
- **iSCSI**
- **FibreChannel**
- **EmptyDir**

for Dev:

- **“Claim”**

```
kind: PersistentVolume
metadata:
    name: pv0003
spec:
    capacity:
        storage: 8Gi
    accessModes:
     - ReadWriteOnce
    nfs:
        path: /tmp
        server: 172.17.0.2
```

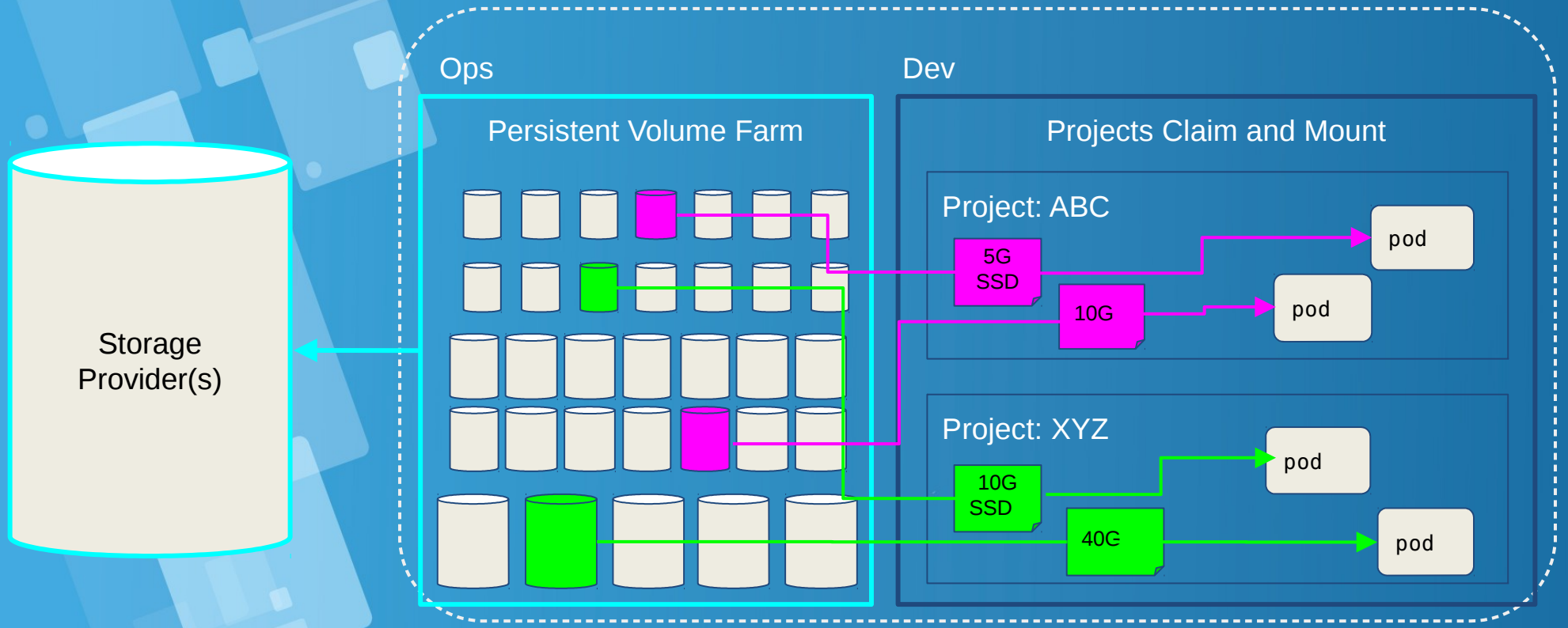
```
kind: PersistentVolumeClaim
metadata:
    name: myclaim
spec:
    accessModes:
     - ReadWriteOnce
    resources:
        requests:
            storage: 8Gi
```



Kubernetes: Persistent Storage

- Kubernetes provides a framework for managing external persistent storage for containers. Kubernetes recognizes a PersistentVolume resource, which defines local or network storage. A pod resource can reference a PersistentVolumeClaim resource in order to access a certain storage size from a PersistentVolume.
- Kubernetes also specifies if a PersistentVolume resource can be shared between pods or if each pod needs its own PersistentVolume with exclusive access. When a pod moves to another node, it keeps connected to the same PersistentVolumeClaim and PersistentVolume instances. So a pod's persistent storage data follows it, regardless of the node where it is scheduled to run.

Kubernetes: Persisten Volume Claim



Kubernetes: Networking

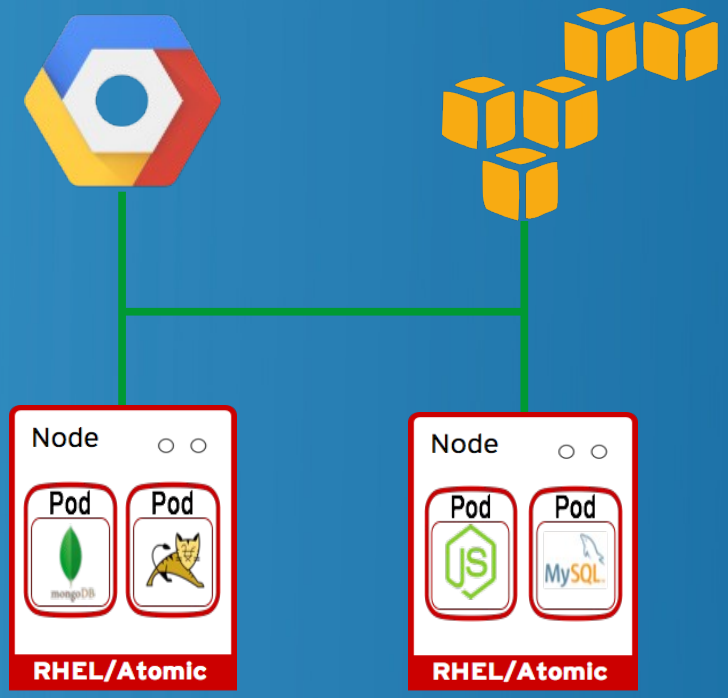
- Each Host = 256 IPs
- Each POD = 1 IP

Programmable Infra:

- GCE / GKE
- AWS
- OpenStack
- Nuage

Overlay Networks:

- Flannel
- Weave
- OpenShift-SDN
- Open vSwitch

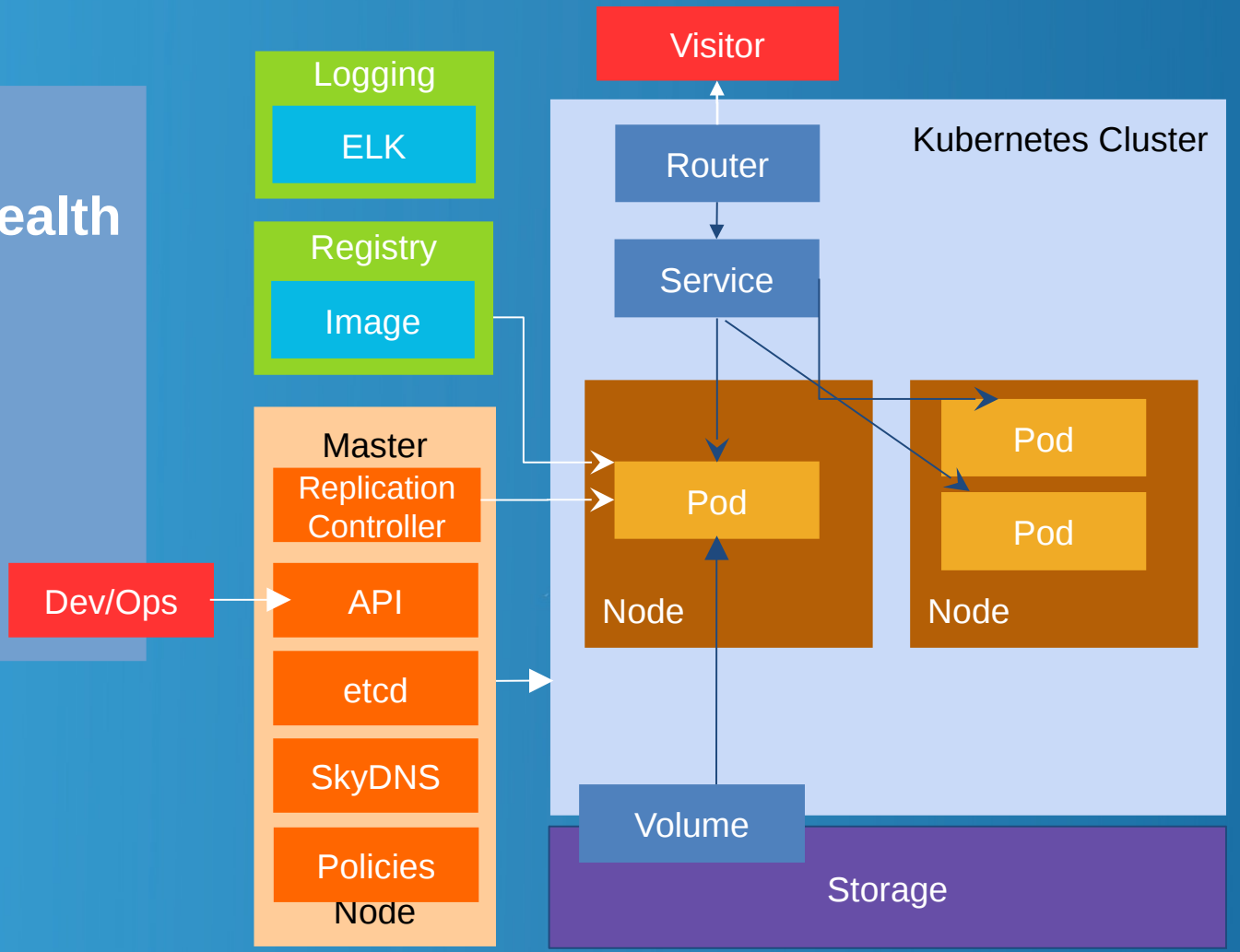


Kubernetes: Networking

- Docker networking is very simple. Docker creates a virtual kernel bridge and connects each container network interface to it. Docker itself does not provide a way to allow a pod from one host to connect to a pod from another host. Docker also does not provide a way to assign a public fixed IP address to an application so external users can access it.
- Kubernetes provides service and route resources to manage network visibility between pods and from the external world to them. A service load-balances received network requests among its pods, while providing a single internal IP address for all clients of the service (which usually are other pods). Containers and pods do not need to know where other pods are, they just connect to the service. A route provides an external IP to a service, making it externally visible.

Kubernetes: Hosting Platform

- **Scheduling**
- **Lifecycle and health**
- **Discovery**
- **Monitoring**
- **Auth{n,z}**
- **Scaling**



Kubernetes: High Avaibility

- High Availability (HA) on an Kubernetes/OpenShift Container Platform cluster has two distinct aspects: HA for the OCP infrastructure itself, that is, the masters, and HA for the applications running inside the OCP cluster.
- For applications, or "pods", OCP handles this by default. If a pod is lost, for any reason, Kubernetes schedules another copy, connects it to the service layer and to the persistent storage. If an entire Node is lost, Kubernetes schedules replacements for all its pods, and eventually all applications will be available again. The applications inside the pods are responsible for their own state, so they need to be HA by themselves, if they are stateful, employing proven techniques such as HTTP session replication or database replication.

Authentication Methods

- Authentication is based on OAuth , which provides a standard HTTP-based API for authenticating both interactive and non-interactive clients.
  - **HTTP Basic, to delegate to external Single Sign-On (SSO) systems**
  - **GitHub and GitLab, to use GitHub and GitLab accounts**
  - **OpenID Connect, to use OpenID-compatible SSO and Google Accounts**
  - **OpenStack Keystone v3 server**
  - **LDAP v3 server**

Kubernetes: Authorization policies

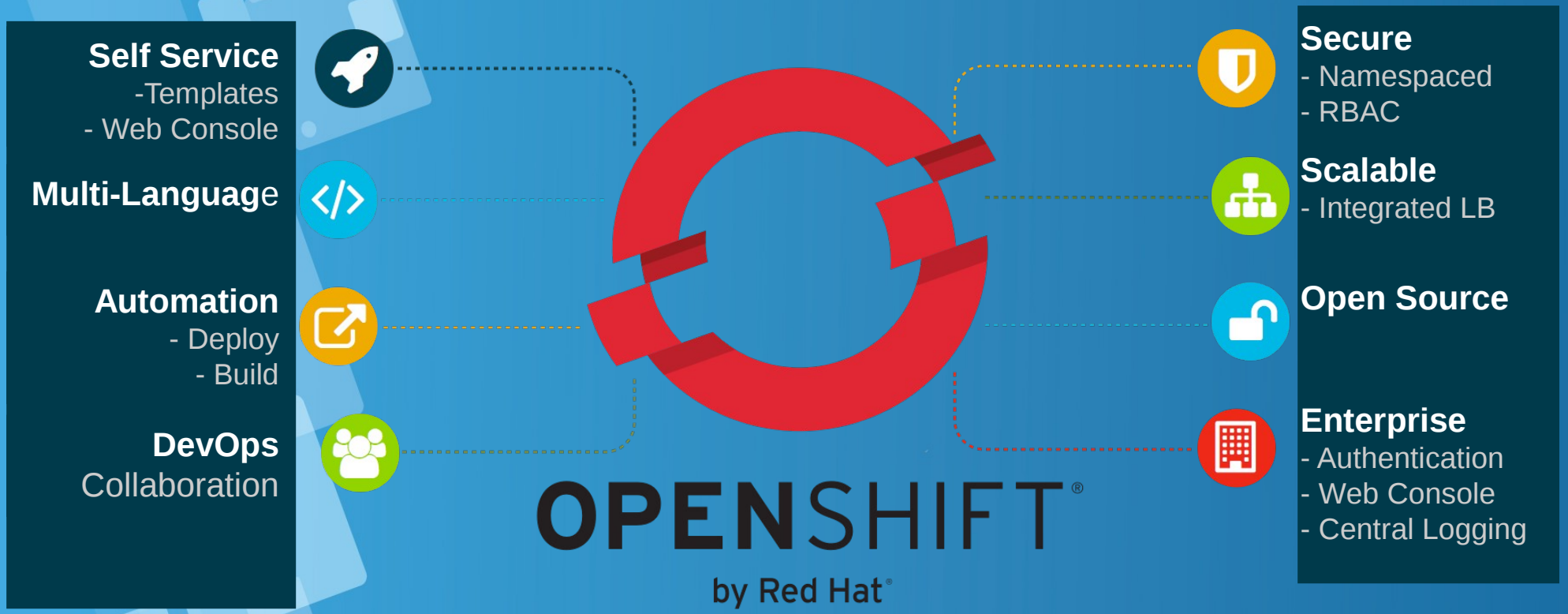
- **There are two levels of authorization policies:**
  - **Cluster policy**: Controls who has various access levels to Kubernetes / OpenShift Container Platform and all projects. Roles that exist in the cluster policy are considered cluster roles.
  - **Local policy**: Controls which users have access to their projects. Roles that exist in a local policy are considered local roles.

- **Authorization is managed using the following:**
  - **Rules**: Sets of permitted verbs on a set of resources; for example, whether someone can delete projects.
  - **Roles**: Collections of rules. Users and groups can be bound to multiple roles at the same time.
  - **Binding**: Associations between users and/or groups with a role.

OpenShift as a Development Platform

Project spaces
Build tools
Integration with your IDE

We Need more than just Orchestration



This past week at KubeCon 2016, Red Hat CTO Chris Wright (@kernelcdub) gave a keynote entitled OpenShift is Enterprise-Ready Kubernetes. There it was for the 1200 people in attendance: OpenShift is 100% Kubernetes, plus all the things that you'll need to run it in production environments. - https://blog.openshift.com/enterprise-ready-kubernetes/

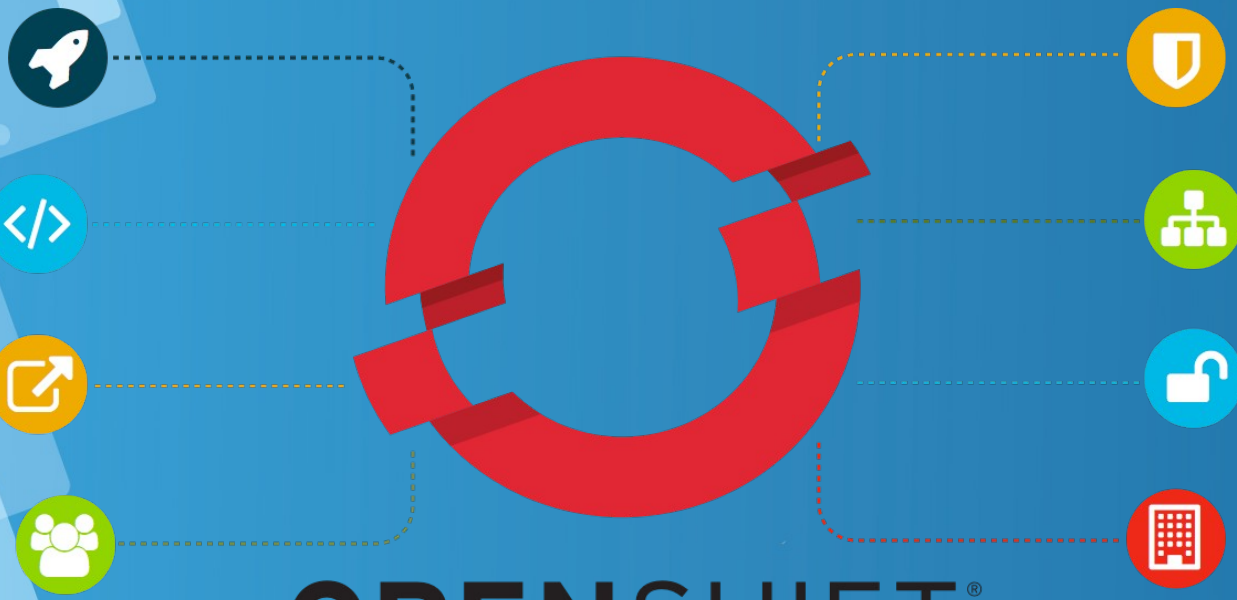
OpenShift is Red Hat Container Application Platform (PaaS)

Self Service
-Templates
- Web Console

Multi-Language

Automation
- Deploy
- Build

DevOps
Collaboration



Secure
- Namespaced
- RBAC

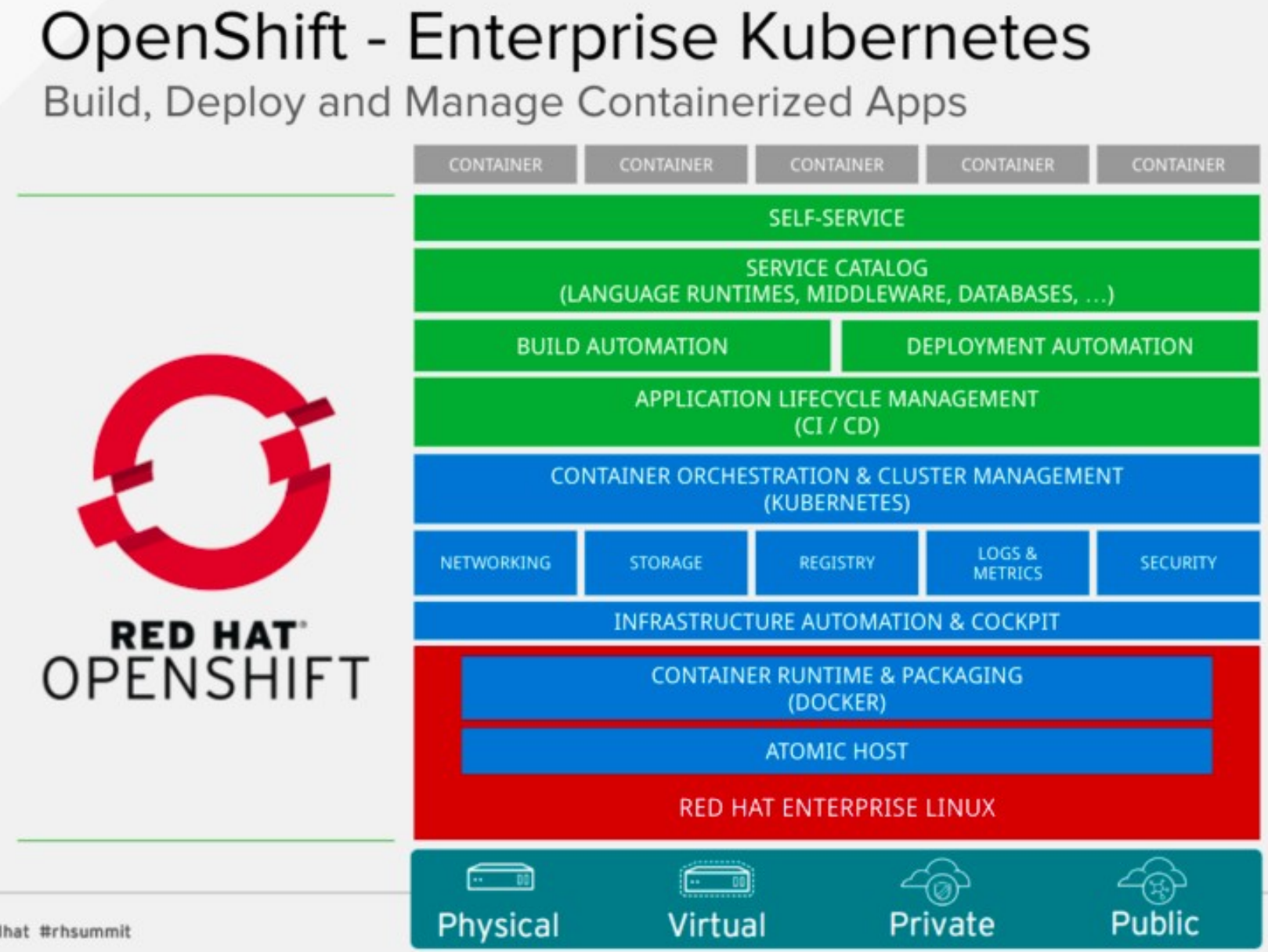
Scalable
- Integrated LB

Open Source

Enterprise
- Authentication
- Web Console
- Central Logging

https://blog.openshift.com/red-hat-chose-kubernetes-openshift/
https://blog.openshift.com/chose-not-join-cloud-foundry-foundation-recommendations-2015/

OpenShift=Enterprise K8s



OpenShift Software Stack

DevOps Tools and User Experience
Web Console, CLI, REST API, SCM integration

Containerized Services
Auth, Networking, Image Registry

Runtimes and xPaaS
Java, Ruby, Node.js and more

Kubernetes
Container orchestration and management

Etcd
Cluster state and configs

OCP-kubernetes Extensions

Docker
Container API and packaging format

RHEL
Container optimized OS

OpenShift Technology

Basic container infrastructure is shown, integrated and enhanced by Red Hat

- The base OS is **RHEL/CentOS/Fedora**.
- **Docker** provides the basic container management API and the container image file format.
- **Kubernetes** is an open source project aimed at managing a cluster of hosts (physical or virtual) running containers. It works with templates that describe multicontainer applications composed of multiple resources, and how they interconnect. If Docker is the "core" of OCP, Kubernetes is the "heart" that keeps it moving.
- **Etcd** is a distributed key-value store, used by Kubernetes to store configuration and state information about the containers and other resources inside the OCP cluster.

Kubernetes Embedded

https://master:8443/api = Kubernetes API
/oapi = OpenShift API
/console = OpenShift WebConsole

OpenShift:

- **1 Binary for Master**
- **1 Binary for Node**
- **1 Binary for Client**

- **Docker-image**
- **Vagrant-image**

Kubernetes:

- **ApiServer, Controller, Scheduler, Etcd**
- **KubeProxy, Kubelet**
- **Kubectl**

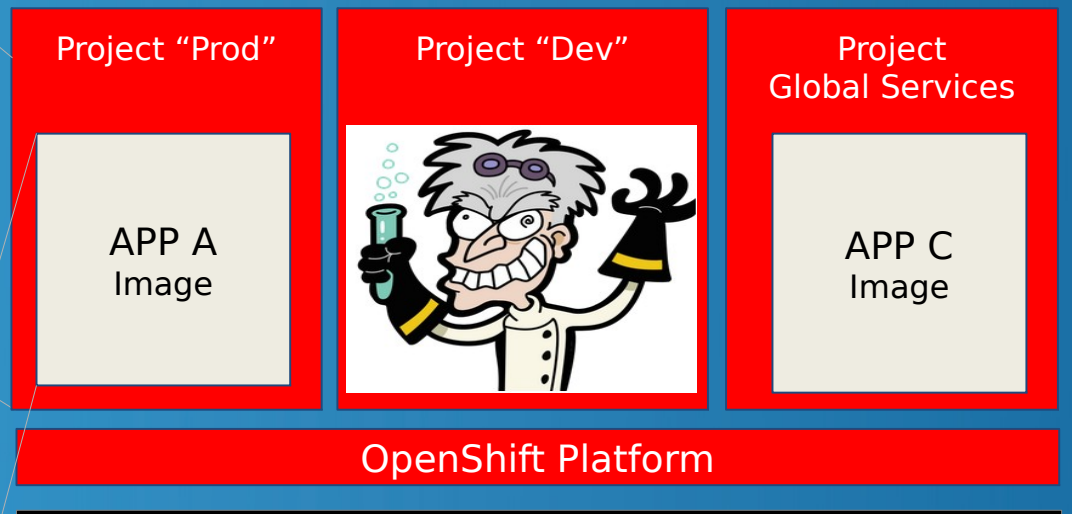
Project Namespace

Project

- **Sandboxed Environment**
- **Network VXLan**
- **Authorization Policies**
- **Resource Quotas**
- ***Ops in Control, Dev Freedom***

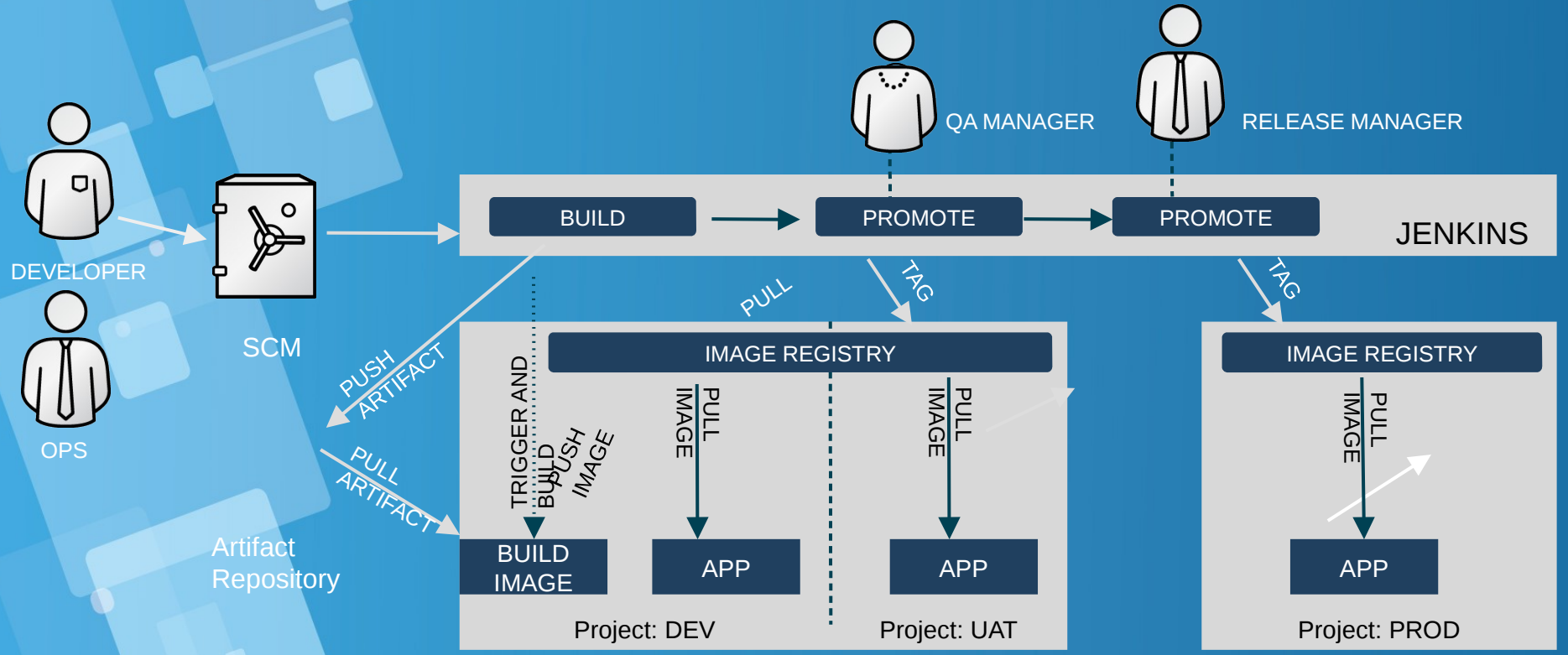
App

- **Images run in Containers**
- **Grouped together as a Service**
- **Defined as Template**

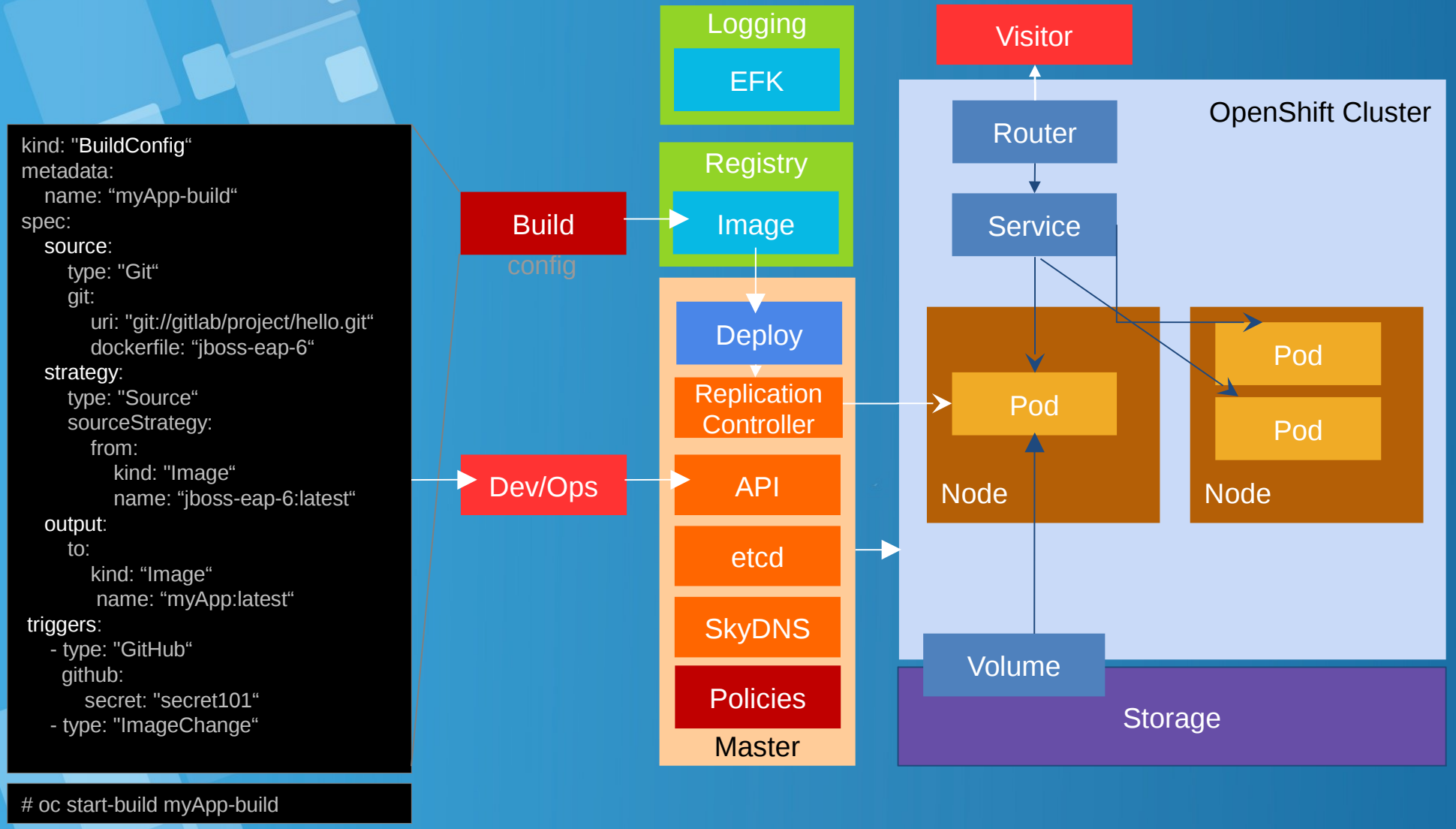



```
oc new-project  Project-Dev
oc policy add-role-to-user admin scientist1
oc new-app
  --source=https://gitlab/MyJavaApp
  --docker-image=jboss-eap
```

CI/CD Flow


DEVELOPER
OPS
SCM
QA MANAGER
RELEASE MANAGER
BUILD
PROMOTE
PROMOTE
JENKINS
PUSH ARTIFACT
PULL ARTIFACT
Artifact Repository
PULL
TAG
TAG
IMAGE REGISTRY
TRIGGER AND BUILD
PUSH IMAGE
PULL IMAGE
PULL IMAGE
BUILD IMAGE
APP
APP
Project: DEV
Project: UAT
IMAGE REGISTRY
PULL IMAGE
APP
Project: PROD

OpenShift Build & Deploy Architecture



Building Images

- **OpenShift/Kubernetes can** build a pod from three different sources
  - A **container image**: The first source leverages the Docker container ecosystem. Many vendors package their applications as container images, and a pod can be created to run those application images inside OpenShift
  - A **Dockerfile**: The second source also leverages the Docker container ecosystem. A Dockerfile is the Docker community standard way of specifying a script to build a container image from Linux OS distribution tools.
  - **Application source code (Source-to-Image or S2I)**: The third source, S2I, empowers a developer to build container images for an application without dealing with or knowing about Docker internals, image registries, and Dockerfiles

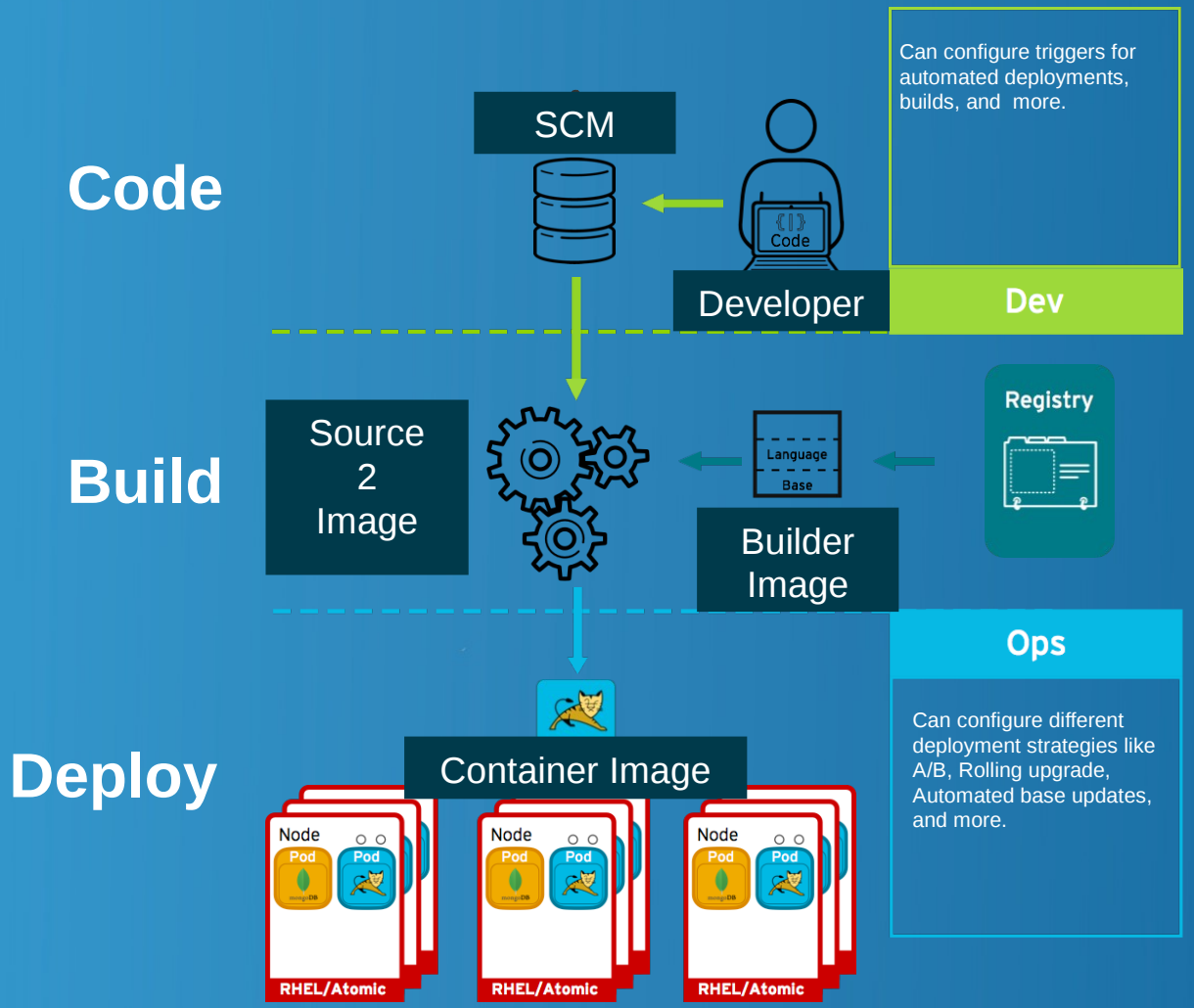
Build & Deploy an Image

Builder Images

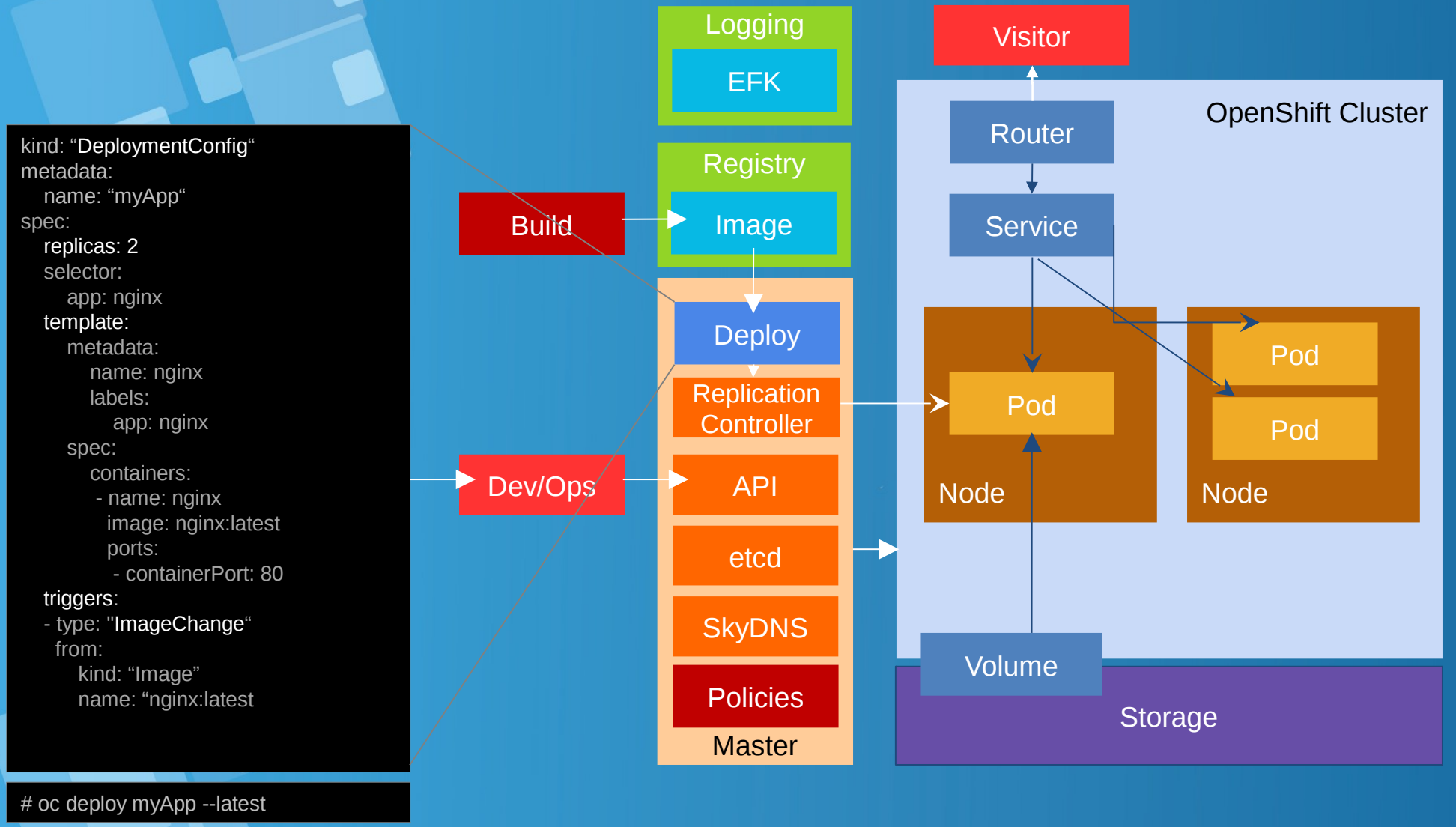
- Jboss-EAP
- PHP
- Python
- Ruby
- Jenkins
- Customer
  - C++ / Go
  - S2I (bash) scripts

Triggers

- Image Change (tagging)
- Code Change (webhook)
- Config Change



OpenShit Build & Deploy Architecture



Pop Quiz

- What is a valid source for building a pod in OpenShift or Kubernetes (Choose three)?

  A) Java, Node.js, PHP, and Ruby source code

  B) RPM packages

  C) Container images in Docker format

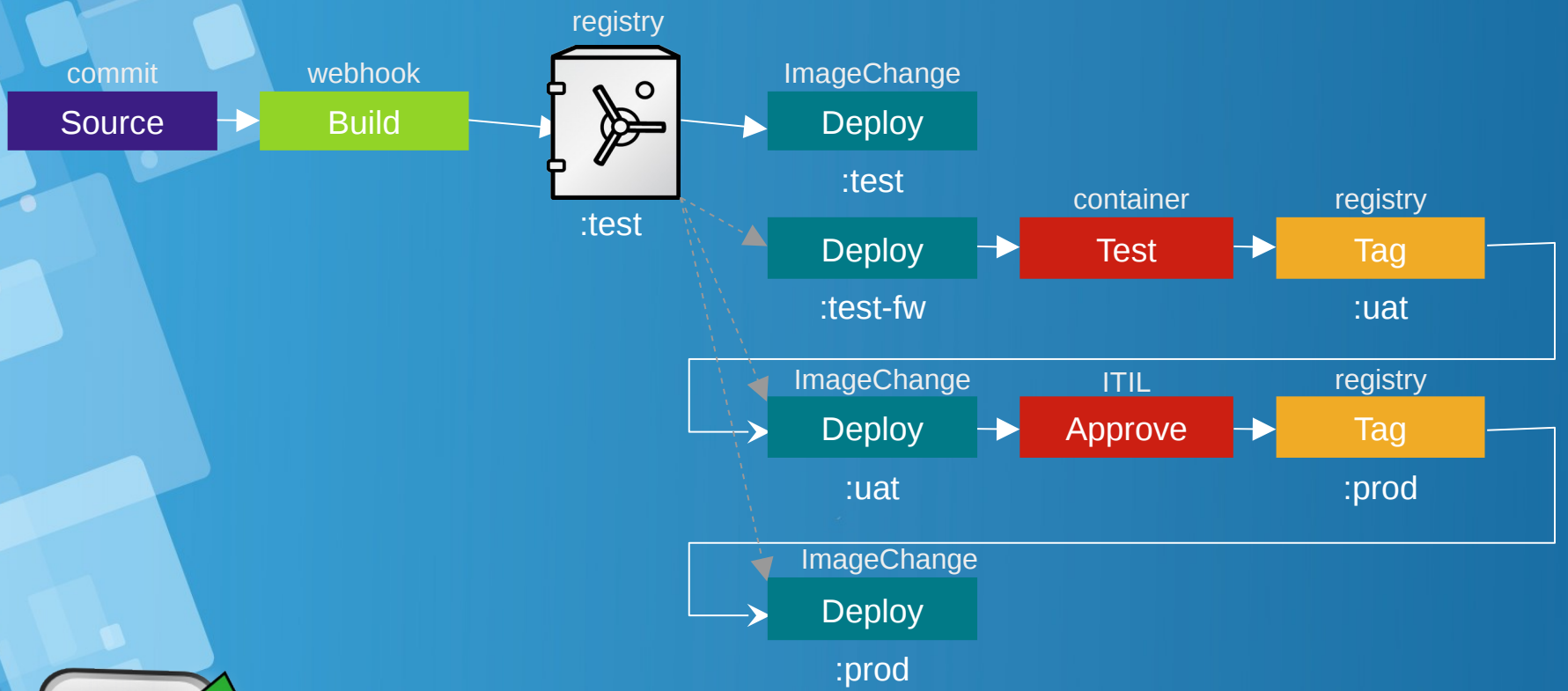
  D) XML files describing the pod metadata

  E) Makefiles describing how to build an application

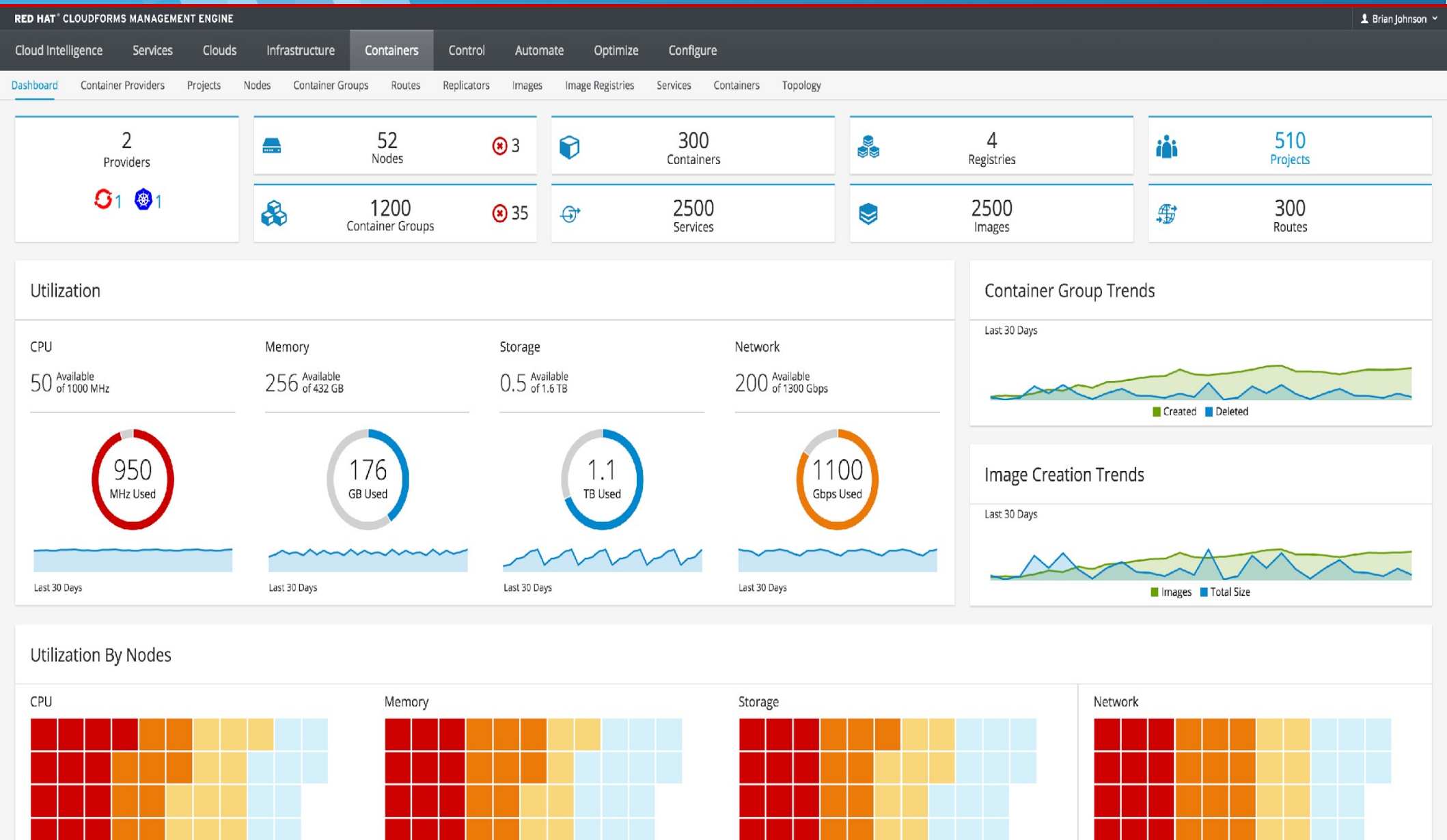
  F) Dockerfiles

Answers the question and Win Merchandize

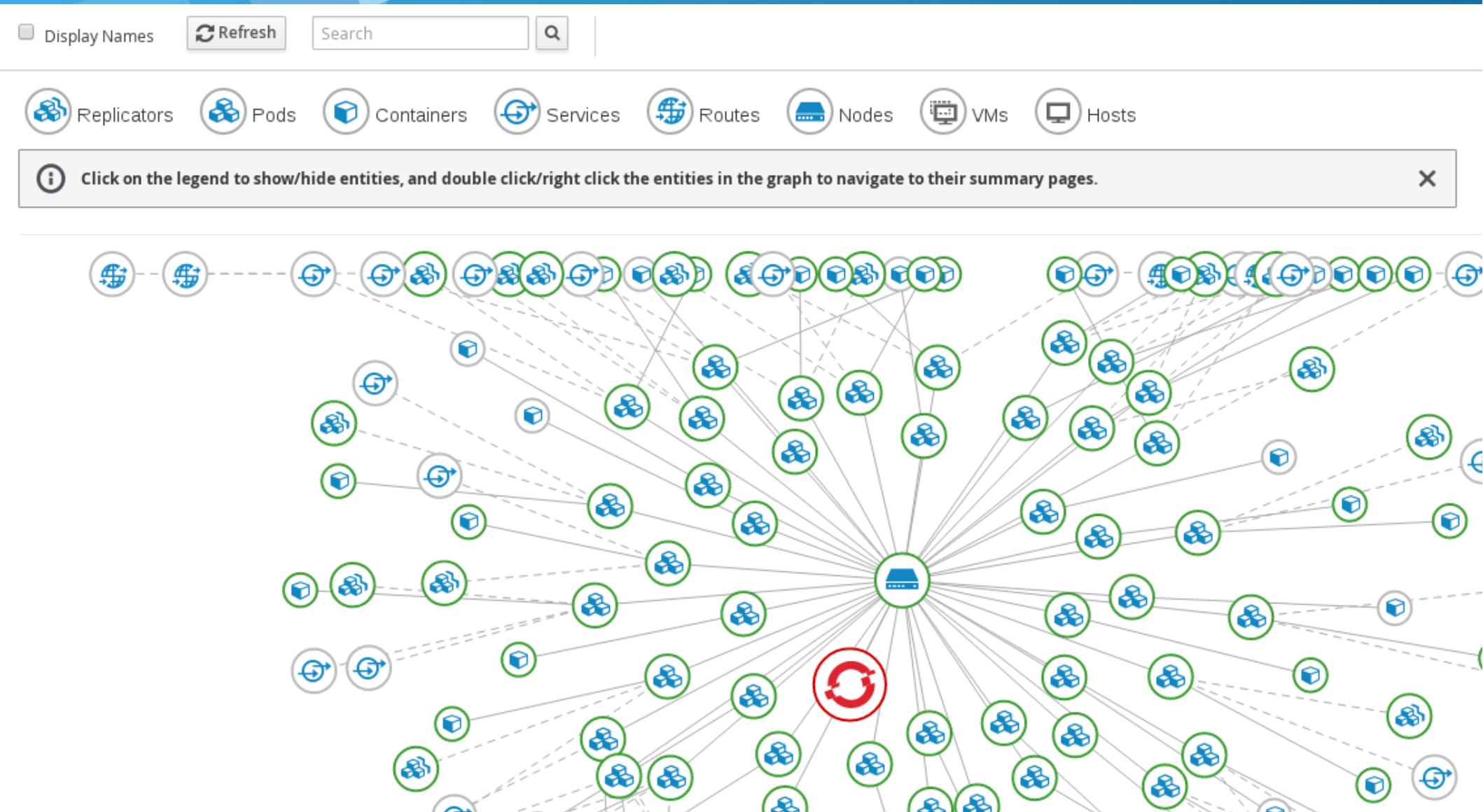
Continous Integration Pipeline Example



Monitoring & Inventory: CloudForm



CloudForm Management



Search

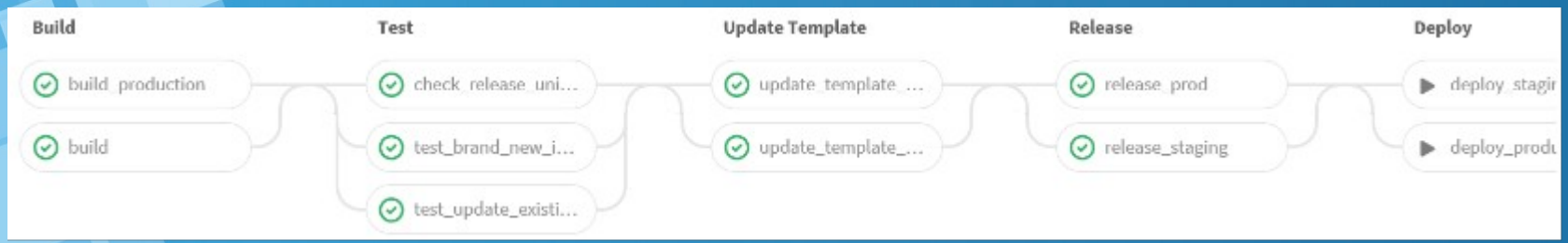
Pods

		Name	Provider	Project Name	Ready	Containers	Phase	Restart Policy	DNS Policy
☐		cakephp-mysql-persistent-2-qwgh7	Openshift-devops-yhs	default	False	1/1	Running	Always	ClusterFirst
☐		cloudforms-1-3b65h	Openshift-devops-yhs	openshift-infra	True	1/1	Running	Always	ClusterFirst
☐		database-1-phqrz	Openshift-devops-yhs	test	True	1/1	Running	Always	ClusterFirst
☐		dc-gitlab-runner-service-1-2xqrm	Openshift-devops-yhs	devops	True	1/1	Running	Always	ClusterFirst
☐		dc-minio-service-1-n0614	Openshift-devops-yhs	devops	True	1/1	Running	Always	ClusterFirst
☐		docker-registry-1-8gz8k	Openshift-devops-yhs	default	True	1/1	Running	Always	ClusterFirst
☐		frontend-1-build	Openshift-devops-yhs	devops	False	0/1	Succeeded	Never	ClusterFirst
☐		frontend-1-vvxx1	Openshift-devops-yhs	test	True	1/1	Running	Always	ClusterFirst
☐		frontend-1-w48hp	Openshift-devops-yhs	test	True	1/1	Running	Always	ClusterFirst

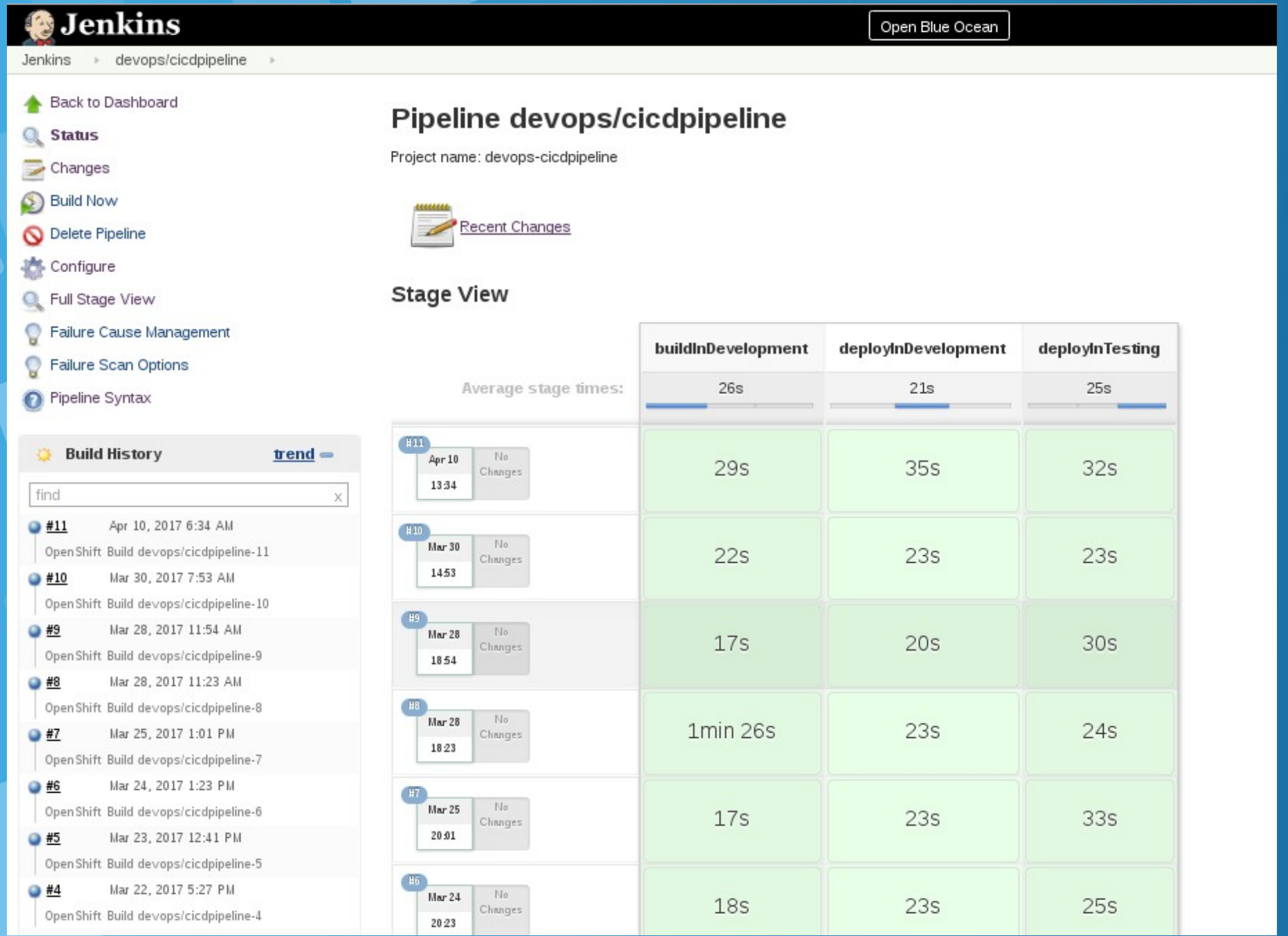
(Check All) Name 20 items 1-20 of 44

Openshift as a tool for developers

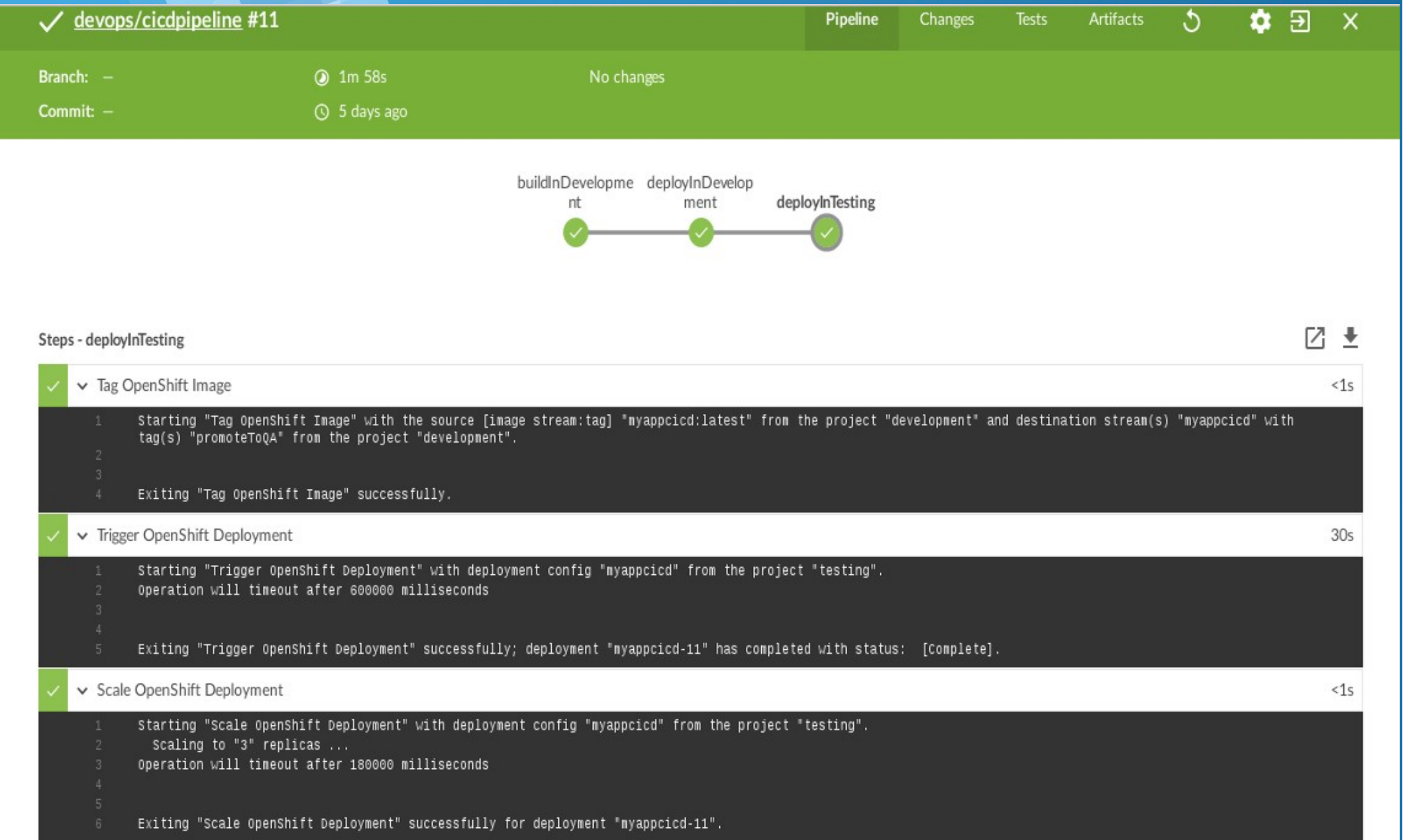
- Facilitate deployment and operation of web applications:
- Getting started with a web application/prototype
- Automate application deployment, rollback changes
- No need to maintain a VM and its OS
- Switch hosting platform (container portability)
- Good integration with code hosting (GitLab)
- CI/CD pipelines (GitLab/Jenkins)
- GitLab Review apps



Openshift: Jenkins CI example



BlueOcean...


devops/cicdpipeline #11
Pipeline
Changes
Tests
Artifacts
Branch: –
Commit: –
1m 58s
5 days ago
No changes
buildInDevelopment
deployInDevelopment
deployInTesting
Steps - deployInTesting
Tag OpenShift Image
<1s
Starting "Tag OpenShift Image" with the source [image stream:tag] "myappcicd:latest" from the project "development" and destination stream(s) "myappcicd" with tag(s) "promoteToQA" from the project "development".
Exiting "Tag OpenShift Image" successfully.
Trigger OpenShift Deployment
30s
Starting "Trigger OpenShift Deployment" with deployment config "myappcicd" from the project "testing".
Operation will timeout after 600000 milliseconds
Exiting "Trigger OpenShift Deployment" successfully; deployment "myappcicd-11" has completed with status: [Complete].
Scale OpenShift Deployment
<1s
Starting "Scale OpenShift Deployment" with deployment config "myappcicd" from the project "testing".
Scaling to "3" replicas ...
Operation will timeout after 180000 milliseconds
Exiting "Scale OpenShift Deployment" successfully for deployment "myappcicd-11".

Q & A

Any Question?

Lets go to Demo..

Installing OpenShift

Preparing OS
All-In-One OpenShift
Post-Installation

Installing OpenShift

OpenShift: Installing Operating System

- 1 VM with:
  - 2 GB Ram + 2-4 Core CPU
  - 20 Gb disk space
  - Additional disk for docker persistent storage lvm
  - Install Centos 7.3 Minimal Install
  - Setting /etc/hosts file point to your domain fqdn "contoh: 192.168.1.1 openshift.example.com"
  - You can bring your own laptop and provide the VM or you can use Cloud services like amazon/Digital-Ocean/etc
  - Internet Connection on VM

https://github.com/isnuryusuf/openshift-install/blob/master/openshift-origin-quickstart.md

Installing OpenShift

OpenShift: Pre-Setup

- Setting hostname at /etc/hosts file, for example:

  ip-address domain-name.tld

- Setting hostname at server:

  # hostnamectl set-hostname domain-name.tld

  # hostname

- Install needed packages

  # yum install wget git net-tools bind-utils iptables-services bridge-utils bash-completion origin-clients

  # yum install centos-release-openshift-origin

https://github.com/isnuryusuf/openshift-install/blob/master/openshift-origin-quickstart.md

Installing OpenShift

OpenShift: Installing Docker

- Install and setup docker

  # yum install docker

- Edit /etc/sysconfig/docker file and add "--insecure-registry 172.30.0.0/16" to the OPTIONS parameter.

  # sed -i '/OPTIONS=.*/c\OPTIONS="--selinux-enabled --insecure-registry 172.30.0.0/16"' /etc/sysconfig/docker

  # systemctl is-active docker

  # systemctl enable docker

  # systemctl start docker

https://github.com/isnuryusuf/openshift-install/blob/master/openshift-origin-quickstart.md

Installing OpenShift

OpenShift: Setting Up

- Pick One, don't do all four
  - OC CLUSTER
  - Running in a Docker Container
  - Running from a rpm
  - Installer Installation Steps
- Refer to github.com/isnuryusuf/openshift-install/
  - File: openshift-origin-quickstart.md

https://github.com/isnuryusuf/openshift-install/blob/master/openshift-origin-quickstart.md

Installing OpenShift

OpenShift: Testing deploy app

- Quick Testi 1:
  - # oc login

    Username: test

    Password: test
  - # oc new-project test
  - # oc new-app **openshift/deployment-example**

https://github.com/isnuryusuf/openshift-install/blob/master/openshift-origin-quickstart.md

Installing OpenShift

OpenShift: Testing Continue..

- Cek Deployment status:
  - # oc status

    In project test on server https://139.59.243.79:8443

    svc/deployment-example - **172.30.235.55:8080**

      dc/deployment-example deploys istag/deployment-example:latest

        deployment #1 deployed about a minute ago - 1 pod

    2 warnings identified, use 'oc status -v' to see details.

https://github.com/isnuryusuf/openshift-install/blob/master/openshift-origin-quickstart.md

Installing OpenShift

OpenShift: Testing Continue..

- Test app:
  - # curl http://**172.30.235.55:8080**

    (example v1) (Use URL that it gives you for svc/deployment-example)
  - # oc tag deployment-example:v2 deployment-example:latest

https://github.com/isnuryusuf/openshift-install/blob/master/openshift-origin-quickstart.md

Installing OpenShift

OpenShift: Basic Configuration

- Login as system:admin
  - # oc login -u system:admin -n default

    Logged into "https://139.59.243.79:8443" as "system:admin" using existing credentials.
    You have access to the following projects and can switch between them with 'oc project <projectname>':

    * **default**
      kube-system
      myproject
      openshift
      openshift-infra
      test
      test-project1
      test-project2
      test2

    Using project "default".

Installing OpenShift

OpenShift: Basic Configuration

- Login as system:admin
  - # oc status
    In project default on server https://139.59.243.79:8443

    svc/docker-registry - 172.30.248.225:5000
      dc/docker-registry deploys docker.io/openshift/origin-docker-registry:v1.4.1
        deployment #1 deployed 35 minutes ago - 1 pod

    svc/kubernetes - 172.30.0.1 ports 443, 53->8053, 53->8053

    svc/router - 172.30.4.117 ports 80, 443, 1936
      dc/router deploys docker.io/openshift/origin-haproxy-router:v1.4.1
        deployment #1 deployed 35 minutes ago - 1 pod

    View details with 'oc describe <resource>/<name>' or list everything with 'oc get all'

Installing OpenShift

OpenShift: Lets Continue on Github

https://github.com/isnuryusuf/openshift-install/blob/master/openshift-origin-quickstart.md

OpenShift Another Demo

- Docker Orchestration
- Source to Image deployment
- Pipeline for CI/CD
- Auto-Scaling using Openshift

Thnk you for Coming

More about me

- https://www.linkedin.com/in/yusuf-hadiwinata-sutandar-3017aa41/
- https://www.facebook.com/yusuf.hadiwinata
- https://github.com/isnuryusuf/

Join me on:

- “Linux administrators” & “CentOS Indonesia Community” Facebook Group
- Docker UG Indonesia: https://t.me/dockerid

Reference

- openshiftenterprise3-160414081118.pptx
- 2017-01-18_-_RedHat_at_CERN_-_Web_application_hosting_with_Openshift_and_Docker.pptx
- DO280 OpenShift Container Platform Administration I
- https://github.com/openshift/origin/

Other Usefull Link

- https://ryaneschinger.com/blog/rolling-updates-kubernetes-replication-controllers-vs-deployments/
- https://kubernetes.io/docs/concepts/storage/persistent-volumes/
- http://blog.midokura.com/2016/08/kubernetes-ready-networking-done-midonet-way/
- https://blog.openshift.com/red-hat-chose-kubernetes-openshift/
- https://blog.openshift.com/chose-not-join-cloud-foundry-foundation-recommendations-2015/
- https://kubernetes.io/docs/concepts/workloads/pods/pod/
- https://blog.openshift.com/enterprise-ready-kubernetes/