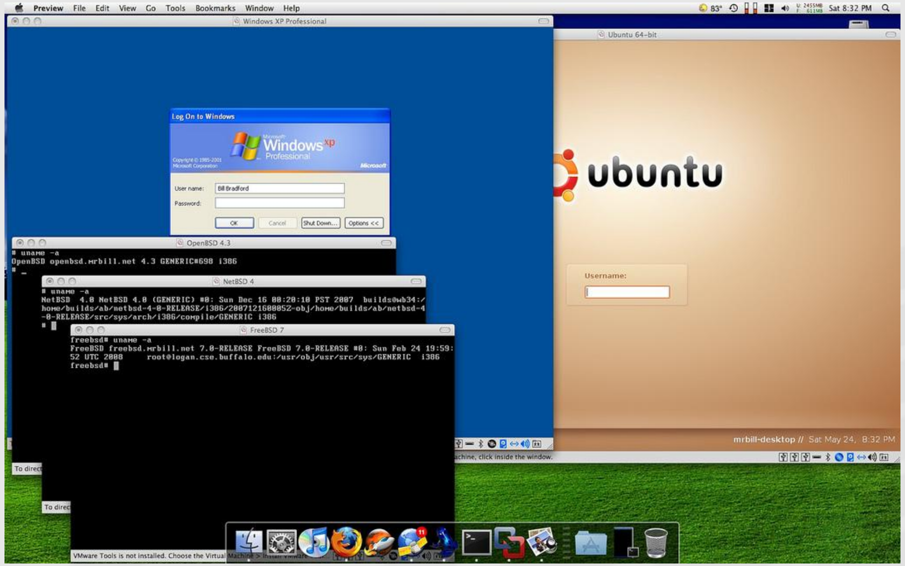
Container History

How did we get here?

- Virtual Machines
  - Have existed in some form since the 1960s
  - x86 was virtualized by VMWare in 1998
  - Slow
    - Especially startup time
  - Resource Intensive
    - The operating system is duplicated for each VM
  - Layering too heavyweight to be effective for most use cases



```
# uname -a
OpenBSD openbsd.mrbill.net 4.3 GENERIC#698 i386
# _
```

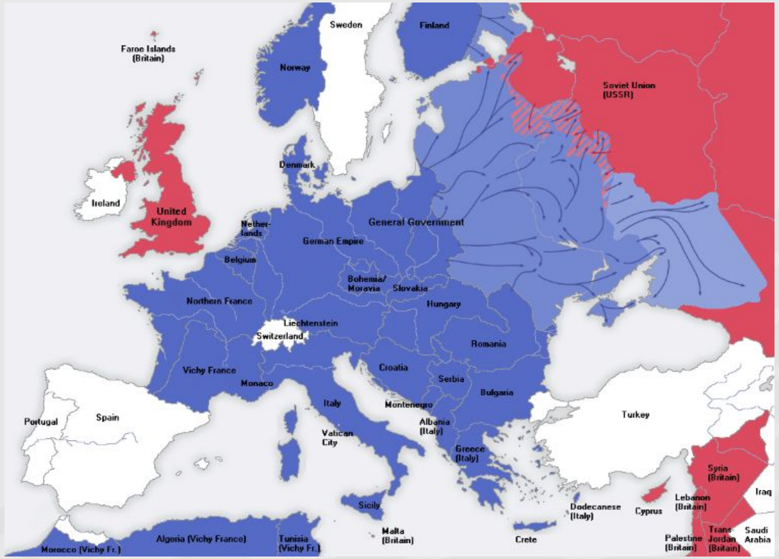
```
# uname -a
NetBSD  4.0 NetBSD 4.0 (GENERIC) #0: Sun Dec 16 00:20:10 PST 2007  builds@wb34:/
home/builds/ab/netbsd-4-0-RELEASE/i386/200712160052-obj/home/builds/ab/netbsd-4
-0-RELEASE/src/sys/arch/i386/compile/GENERIC i386
#
```

```
freebsd# uname -a
FreeBSD freebsd.mrbill.net 7.0-RELEASE FreeBSD 7.0-RELEASE #0: Sun Feb 24 19:59:
52 UTC 2008     root@logan.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC  i386
freebsd#
```

VMware Tools is not installed. Choose the Virtual Machine > Install VMware

While VMs were taking over the world

- Solaris Zones - 2005
- Linux Containers
    - selinux (Developed by the NSA) - 2000
    - cgroups (Developed by Google) - 2006
    - Linux Namespaces (mnt, pid, net, ipc, uts, user) - 2008
    - LXC - 2008
    - Docker - 2013 (March)



OPENSHIFT by Red Hat

Docker

- The first 1 min demo:
    - dnf install docker
    - docker run -it --rm fedora /bin/bash
- Everyone gets it and it’s amazing
- Common patterns make it easy have 1 container per application component
    - Single process per container
    - Keeping state out of containers
        - Use persistent volumes and external datastores
        - Logs and metrics can be standardized and aggregated externally
    - Building dependencies into ancestor image layers
- However, you wouldn’t want to maintain a complex application with lots of components using a series of `docker run` commands

System Architecture

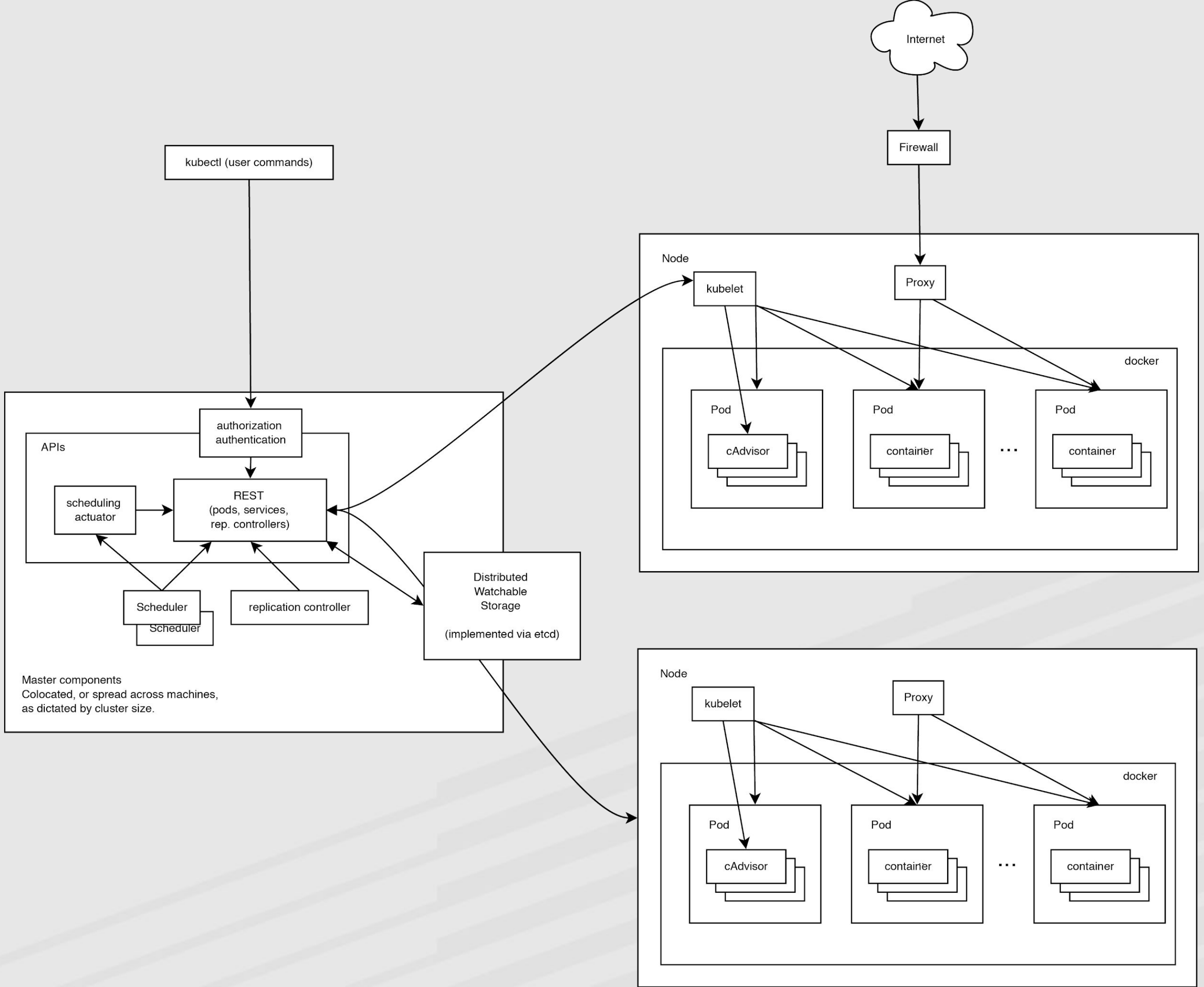
The moving pieces

OPENSHIFT
by Red Hat

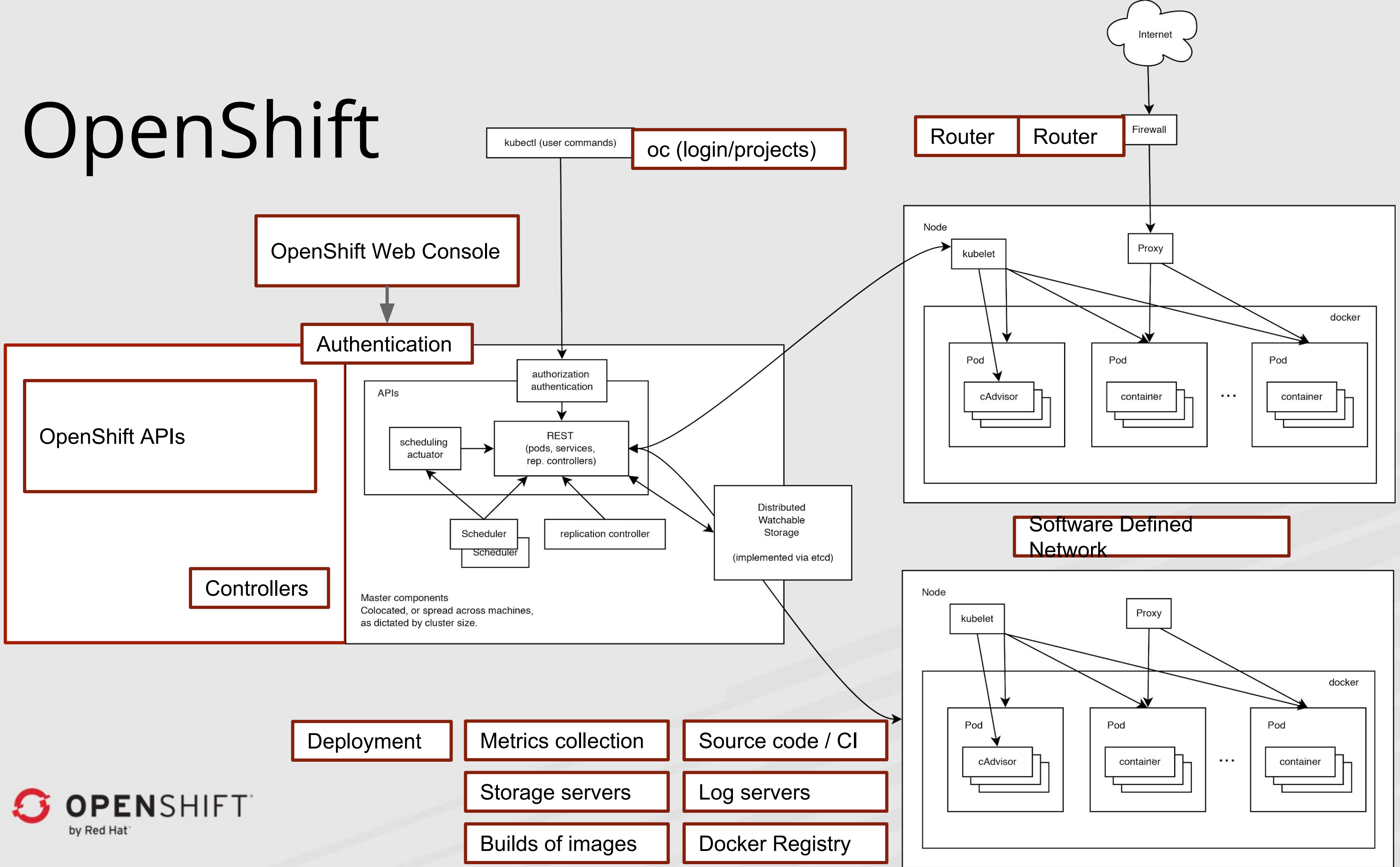
What are the pieces?

- Docker
  - Container runtime and image distribution
- Kubernetes
  - Runtime and operational management of containers
- OpenShift
  - Lifecycle of applications - build, deploy, manage, promote
  - Manage tens to thousands of applications with teams

Kube


Internet
Firewall
kubectl (user commands)
Node
kubelet
Proxy
docker
Pod
cAdvisor
Pod
container
...
Pod
container
authorization
authentication
APIs
scheduling
actuator
REST
(pods, services,
rep. controllers)
Scheduler
Scheduler
replication controller
Distributed
Watchable
Storage
(implemented via etcd)
Master components
Colocated, or spread across machines,
as dictated by cluster size.
Node
kubelet
Proxy
docker
Pod
cAdvisor
Pod
container
...
Pod
container

OpenShift



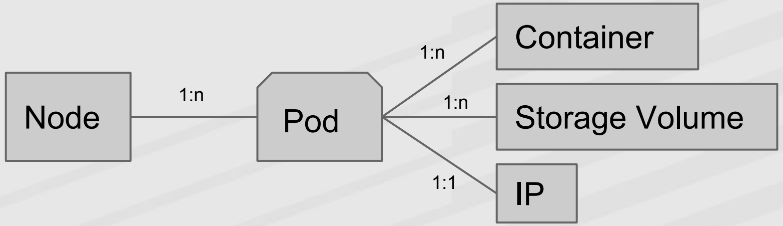
OPENSHIFT by Red Hat

Concepts

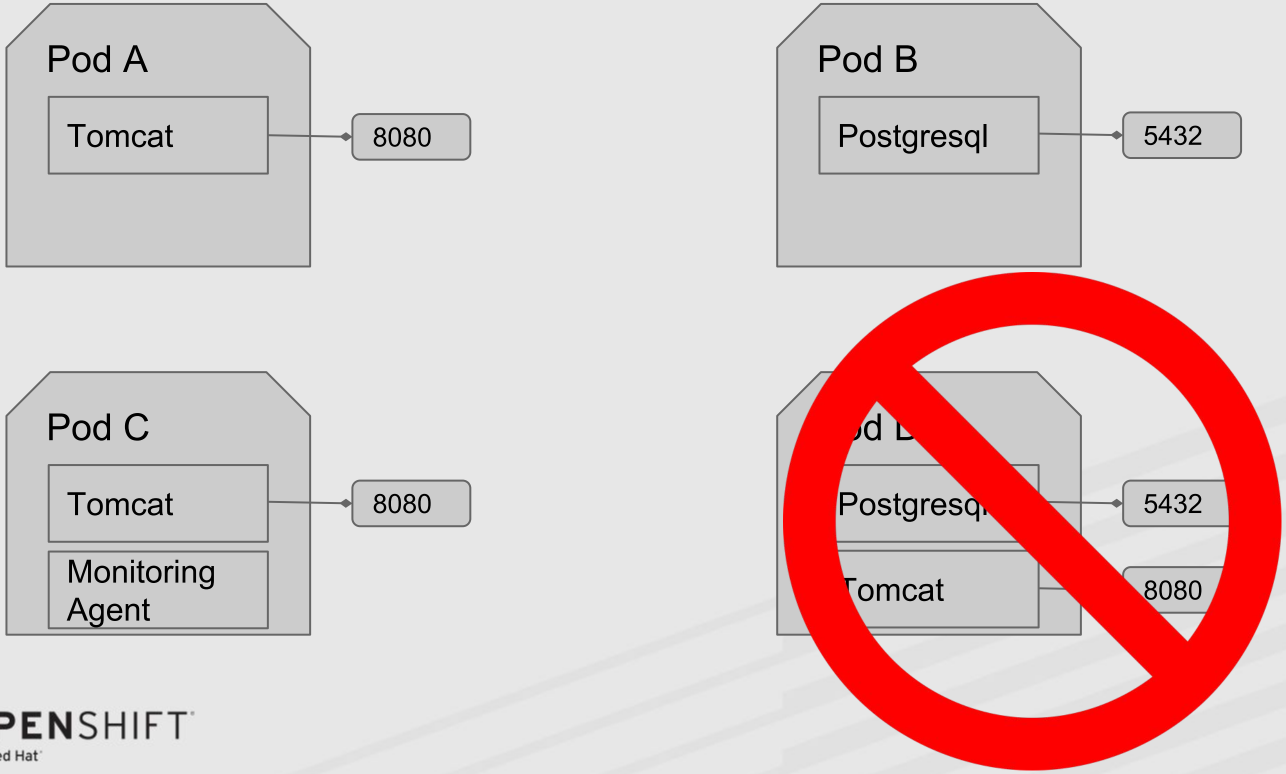
OpenShift Nouns

Pods and Containers

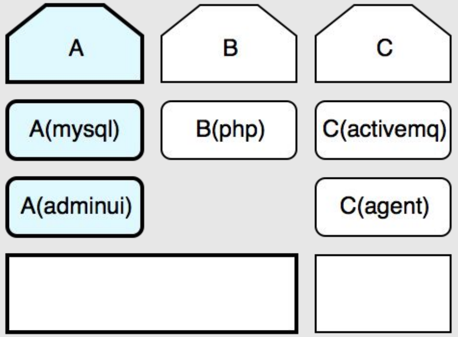
- Fundamental unit in the system
  - Pod is a group of related containers on the same node
  - Each container can be its own image with its own env
  - Pods share an IP address and volumes
- Pods are transient and not “special”
  - Pods should be able to be deleted at any time
  - Storage can be detached and reattached elsewhere



Pod Examples



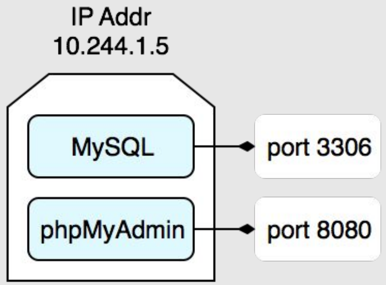
Pods (cont.)



Pods
Group of related containers placed onto the same host

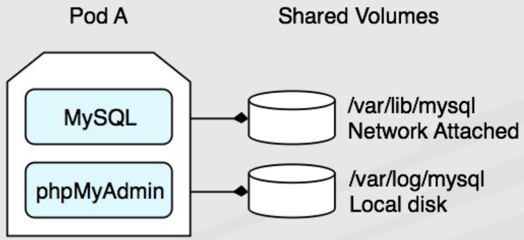
Docker Containers
Application processes and filesystem libraries

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

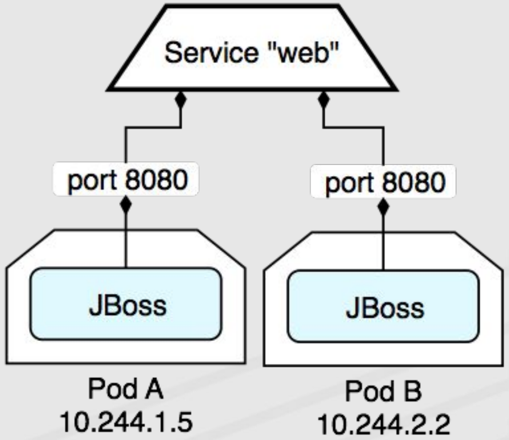
Volume Types
Each volume can have different types, like local transient storage or network attached storage backed by Cinder, GCE, EBS, etc

Connecting Pods

- Need a way for pod A to talk to pod B
  - Option 1: Hardcode IP address
  - Option 2: Query the server
- If there are 10 copies of pod B, which do you use?
  - Pick one randomly?
  - Load balance!
- What if it fails?
  - Want to have all copies of pod A talk to all copies of pod B

Services

- Abstract a set of pods as a single IP and port
  - Each host has a proxy that knows where other pods are
  - Simple TCP/UDP load balancing

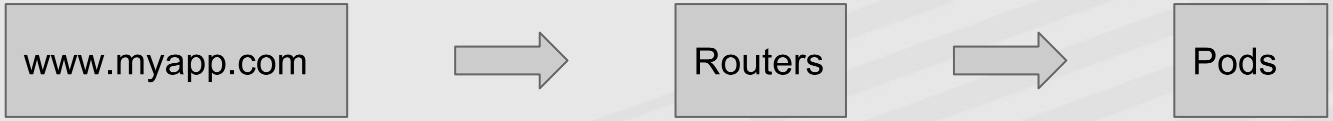


Services abstract other pods
A service is a TCP port that may transparently load balance other ports

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.

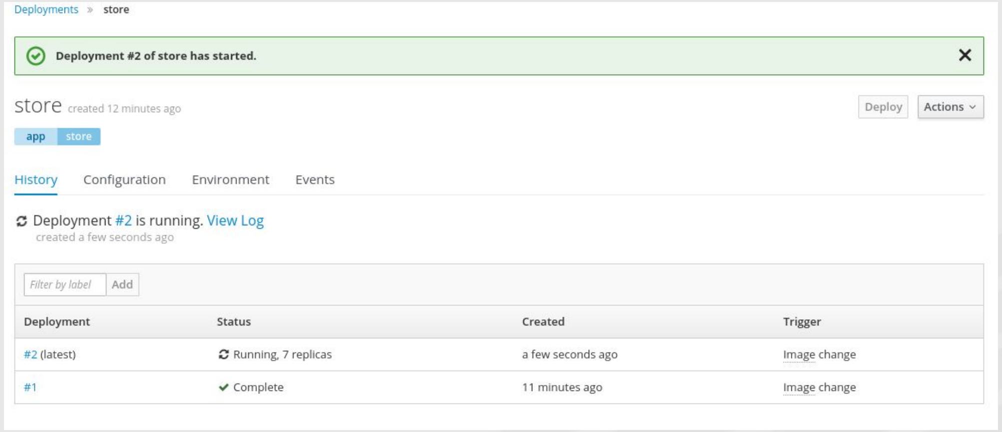
Routes

- Getting external traffic into the cluster
- Reference DNS (www.google.com)
    - Wildcard DNS to provide a cluster default
- Uses service end points but bypasses the service proxy and routes directly to the pods



Deployments

- Define the lifecycle for a single image
  - Each deployment records a particular image and settings for that image at a point in time



Deployments » store

Deployment #2 of store has started.

store created 12 minutes ago

Deploy | Actions

app store

History | Configuration | Environment | Events

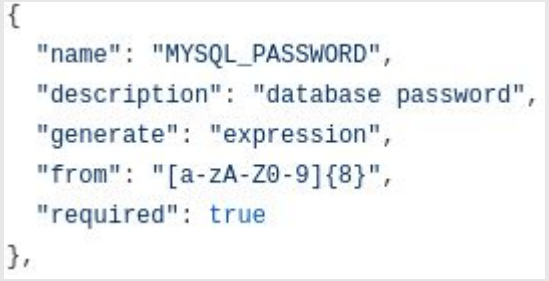
Deployment #2 is running. View Log
created a few seconds ago

Deployment	Status	Created	Trigger
#2 (latest)	Running, 7 replicas	a few seconds ago	Image change
#1	✓ Complete	11 minutes ago	Image change

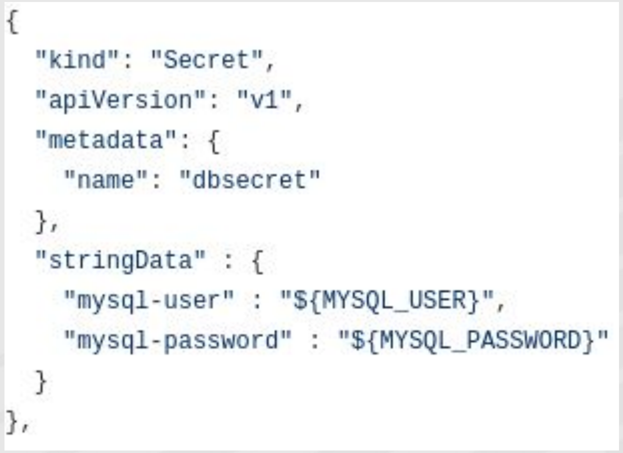
OPENSHIFT by Red Hat

Templates and Config

- Config is declarative description of topology
- Templates let you parameterize config
  - Simple key/value substitution with basic value generation



```
{
  "name": "MYSQL_PASSWORD",
  "description": "database password",
  "generate": "expression",
  "from": "[a-zA-Z0-9]{8}",
  "required": true
},
```



```
{
  "kind": "Secret",
  "apiVersion": "v1",
  "metadata": {
    "name": "dbsecret"
  },
  "stringData" : {
    "mysql-user" : "${MYSQL_USER}",
    "mysql-password" : "${MYSQL_PASSWORD}"
  }
},
```

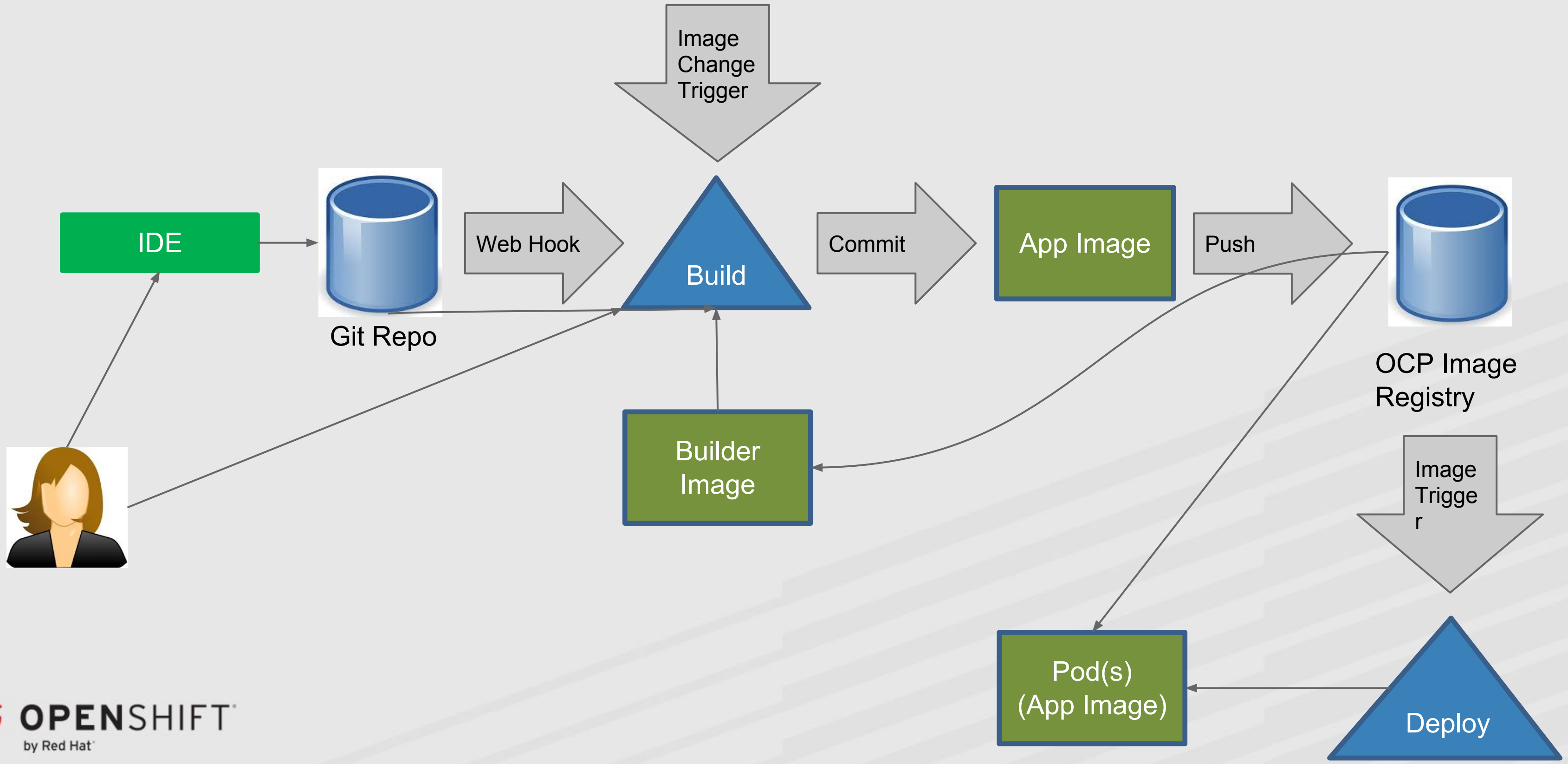
OPENSHIFT
by Red Hat

Building Images

- Allow infrastructure to build images
  - Source to Image (S2I) and Dockerfile builds
  - Integration with Jenkins and other build systems
  - Builds are run in containers under user resource limits
  - Integrated Registry
- Easy integration for existing build infrastructure
  - Push images into an image repository
- Easy integration with external source repos
  - GitHub and generic webhooks

OPENSHIFT by Red Hat

Build Flow



OPENSHIFT by Red Hat

Resources

- Try out OpenShift for yourself
  - https://learn.openshift.com/

Demo

OPENSHIFT
by Red Hat