Getting Started with OpenShift v3

Vineet Reynolds · Red Hat

Who am I ?

What does this talk cover ?

What does this talk cover ?

- Docker and Kubernetes

What does this talk cover ?

- Docker and Kubernetes
- **OpenShift v3**

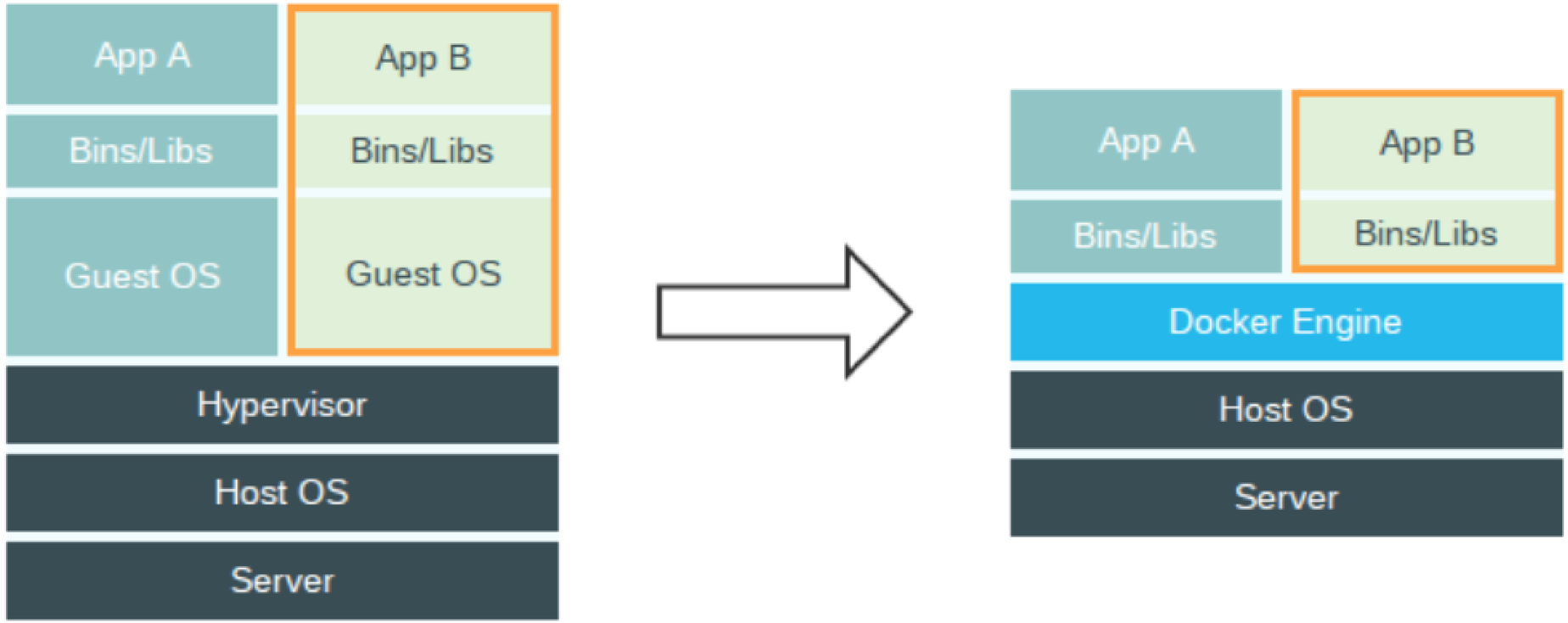
Docker and Kubernetes

for the uninitiated

Containers

- Provide a light-weight virtualization solution
- Through the use of Linux kernel features -
  - `cgroups` to share/limit hardware resources
  - `namespaces` to provide isolation
- without requiring a guest OS

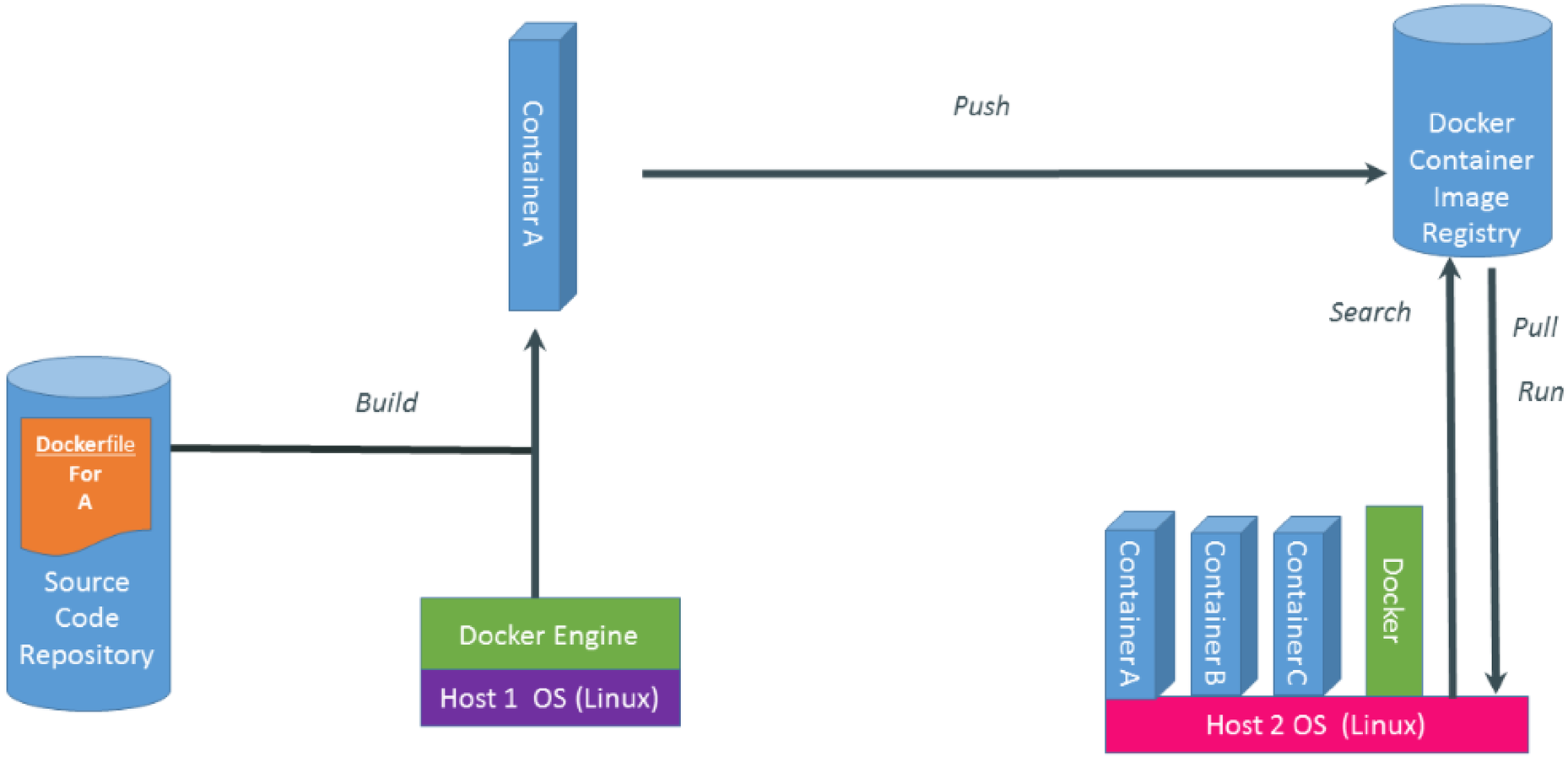
Containers


App A
App B
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Hypervisor
Host OS
Server
App A
App B
Bins/Libs
Bins/Libs
Docker Engine
Host OS
Server

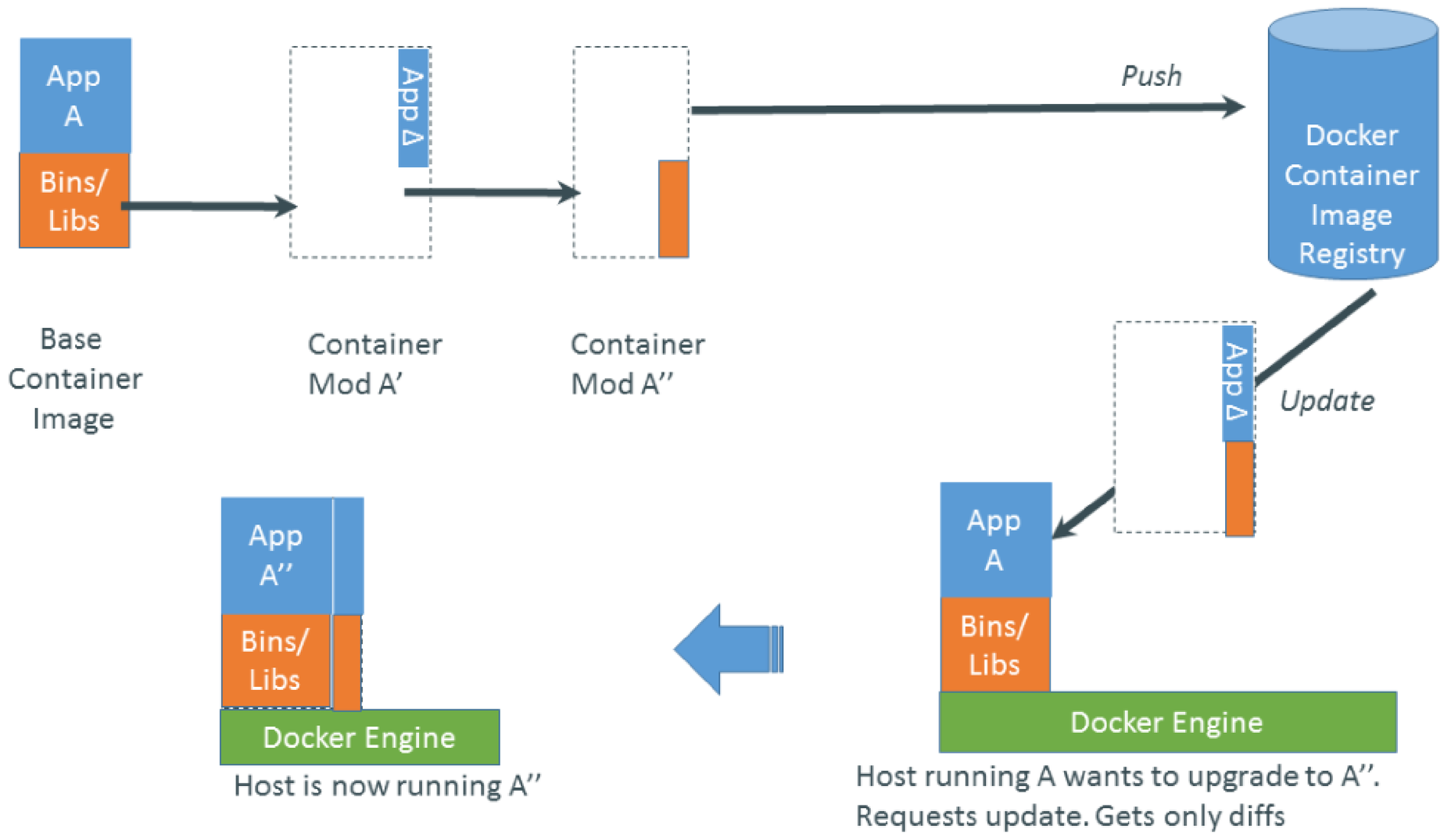
Docker is a shipping container system for code. Consists of -

- A portable, lightweight runtime and packaging tool (Docker Engine)
- A cloud service for sharing applications and automating workflows (Docker Hub)

Basics of the Docker system



Changing and Updating images



Kubernetes

- Supports runtime and operational management of containers
- Describes the intended state of the cluster
  - Record links between containers - 'frontend' depends on 'backend'
  - Replicate containers onto the desired number of nodes; 'frontend' should always run on X nodes, 'backend' should run only on 1 node
- Provides self-healing capabilities to repair the cluster to intended state

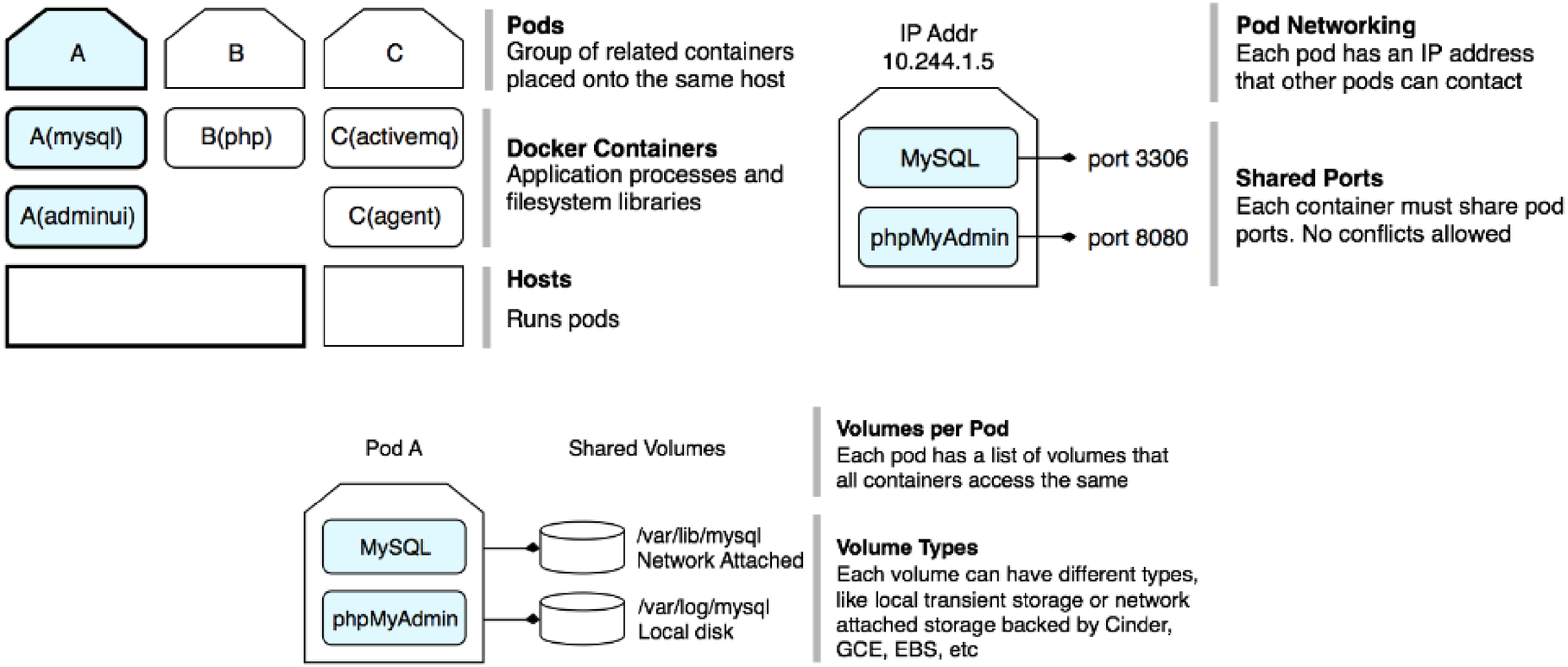
- Solves the Cluster Container Management problem
  - the substrate for running containers at scale
  - contains just the runtime and operational tools for containers
  - Composable system - only enough to enable other use cases

Concepts - Pods and Containers

- **Fundamental unit in the system**
  - Pod is a group of related containers on the same system
  - Each container can be its own image with its own env
  - Pods share an IP address and data volumes
- **Pods are "transient"**
  - Pods should be able to be deleted at any time
  - Storage can be detached and reattached elsewhere
  - Different pods talk to each other through abstractions

- Single container - JBoss, MySQL etc.
- Web server and log parser
  - Web server container logs HTTP requests to disk
  - Log parser reads from disk and sends summary info elsewhere

Concepts - Pods (contd.)



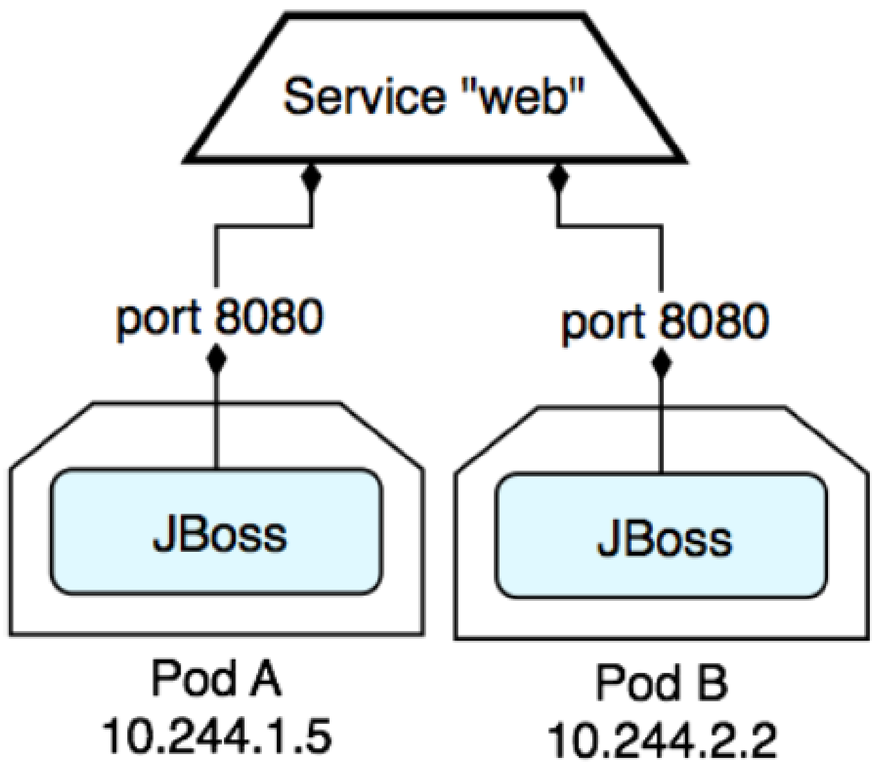
Connecting Pods

- 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 A, 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
  - No central load balancer (no SPOF)
- Creates environment variables in other pods
  - Like "docker link", but across hosts
  - Service named "mysql" gets `MYSQL_HOST` and `MYSQL_PORT`

Services(contd.)



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.

Why Services?

- Stable endpoint for pods to reference
  - Allows list of pods to change dynamically
- Need a way to know where other pods are
  - By name (mysql is recognizable)
  - Simple service discovery
- Lots of other options for service discovery
  - The simplest thing that works at scale

OpenShift adds

- Ability to build, manage, and deliver app descriptions at scale
- Turning source code into new deployable components
- Lifecyle to applications - Build, deploy, manage, promote

OpenShift extends Kubernetes:

- To provide application templates
  - Single JSON file to configure Kube resources
  - Parameterizable
- To build images from source code
  - Host source code in git repos
  - Performs builds and host private Docker images
  - Kicks off new builds on updates to git repos

OpenShift - What is an app?

- One or more set of pods
  - JBoss, MySQL, Redis, ActiveMQ, Backend Workers, Frontends
- Linked together by services
  - Edge router for myapp.com → JBoss frontend pods
  - JBoss frontend pods → MySQL service
  - MySQL service → MySQL pod with attached storage

OpenShift

- Turning source code into new deployable components
- Lifecyle to applications - Build, deploy, manage, promote

Getting started with

OpenShift

- Setup Docker
  - On Fedora

```
sudo yum -y install docker
```

  - On Ubuntu Trusty 14.04

```
wget -qO- https://get.docker.com/ | sh
```

  - Use Boot2Docker on Mac/Windows
  - But, using a Linux VM is preferable

Download OpenShift Binaries

- **Download the binaries from GitHub**
  - 0.4 - Beta2 **is recommended**

Starting OpenShift

Start OpenShift.

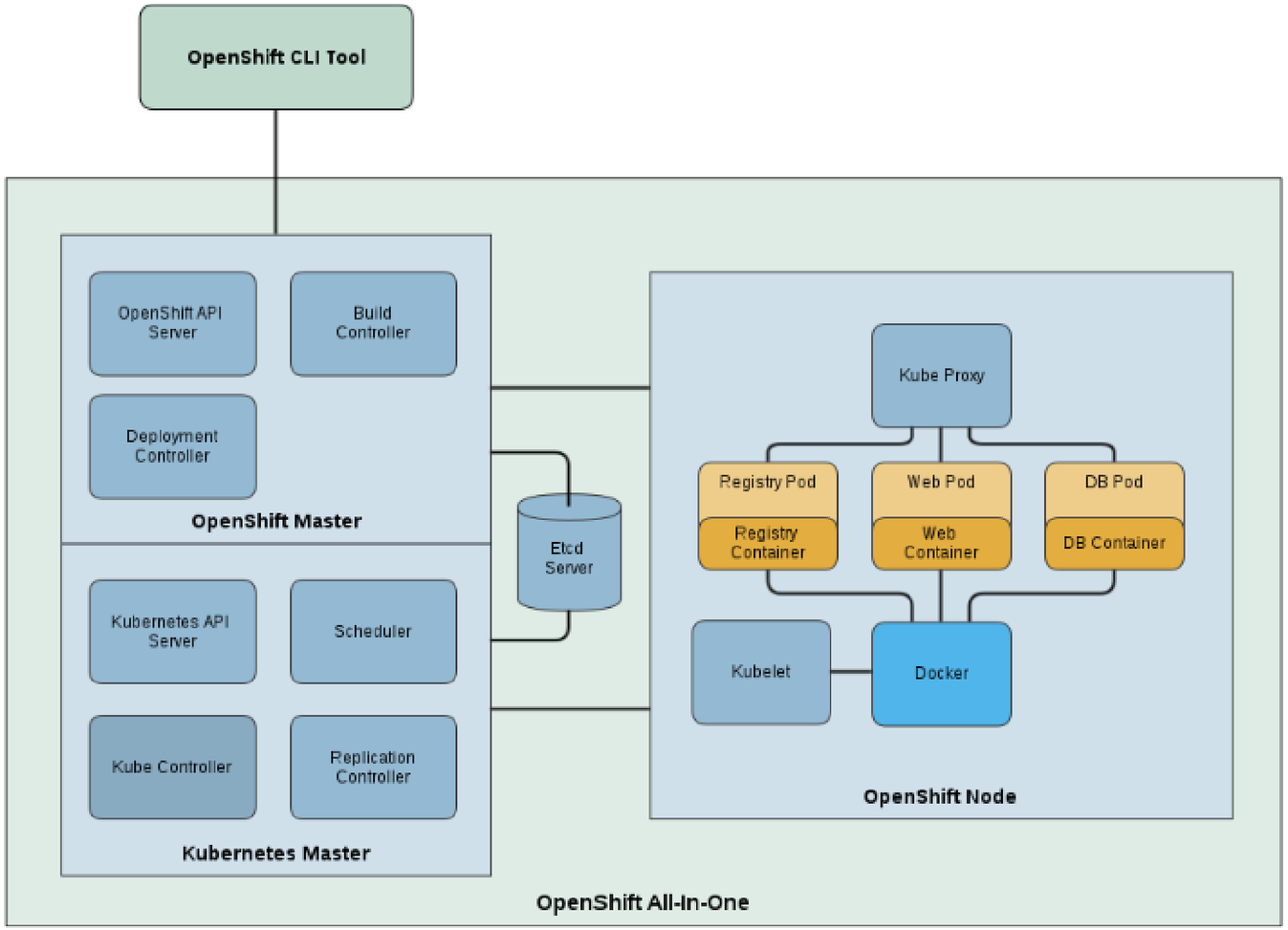
```
sudo openshift start &> logs/openshift.log &
```

The `openshift` executable

Starts various components

- The Kubernetes master
- A kubelet
- The OpenShift master
- An `etcd` server
- Controllers

The state of the cluster



Configure the client for TLS

```
export KUBECONFIG=`pwd`/openshift.local.certificates/admin/.kubeconfig
export CURL_CA_BUNDLE=`pwd`/openshift.local.certificates/ca/cert.crt
sudo chmod +r "$KUBECONFIG"
```

- The `.kubeconfig` file is used by `osc`.
- The CA cert is used by `curl`.

Bind a user `test-admin` to the `view` role

```
openshift ex policy add-user view anypassword:test-admin
```

- Launch the OpenShift UI @ `http://localhost:8444/console`
- Log in with the `test-admin` user. Specify any password.

Deploying the registry

- Images to be deployed in production can be hosted in a registry
- This is available as a Docker image, can be deployed through a command:

```
openshift ex registry --create --credentials="$KUBECONFIG"
```

Verify the registry is up and running.

```
osc describe service docker-registry
```

```
curl `osc get service docker-registry --template="{{ .portalIP }}:{{ .port
}}"`
```

Creating the project on OpenShift

- **Create a new project**

```
openshift ex new-project test --display-name="OpenShift 3 Sample" --
description="This is an example project to demonstrate OpenShift v3" --
admin=anypassword:test-admin
```

- **Submit the application template for processing**

```
osc process -n test -f application-template-jeebuild.json | osc create -n test
-f -
```

By now, the following OpenShift entities have been created:

- An OpenShift build config
- ImageStreams for images used (the base) and created (the production images)
- DeploymentConfigs to define the deployments after build
- Services for the deployments performed as output of build

- **Trigger a build**

```
osc start-build -n test jee-sample-build
```

- **Observe the build**

```
osc get -n test builds
osc get -n test builds --watch
osc build-logs -n test <BUILD_ID>
```

Query for pods and services, once the build is complete

```
osc get -n test pods
```

```
osc get -n test services
```

- The `IP:port` can be found in the frontend and database services.
- Use the browser to access the frontend, and MySQL workbench to access the database.

OpenShift

- offers a lightweight virtualization solution
- links various components of your project together
- stores the intended state of your application for the cluster
- provides tight-knit integration for devs and ops alike

Questions?