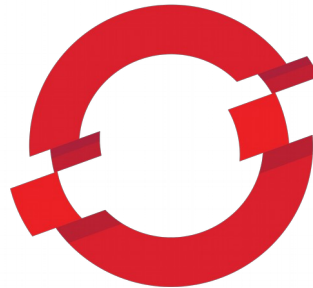


Yusuf Hadiwinata Sutandar

Running AutoScaling JBOSS Cluster with OpenShift



docker

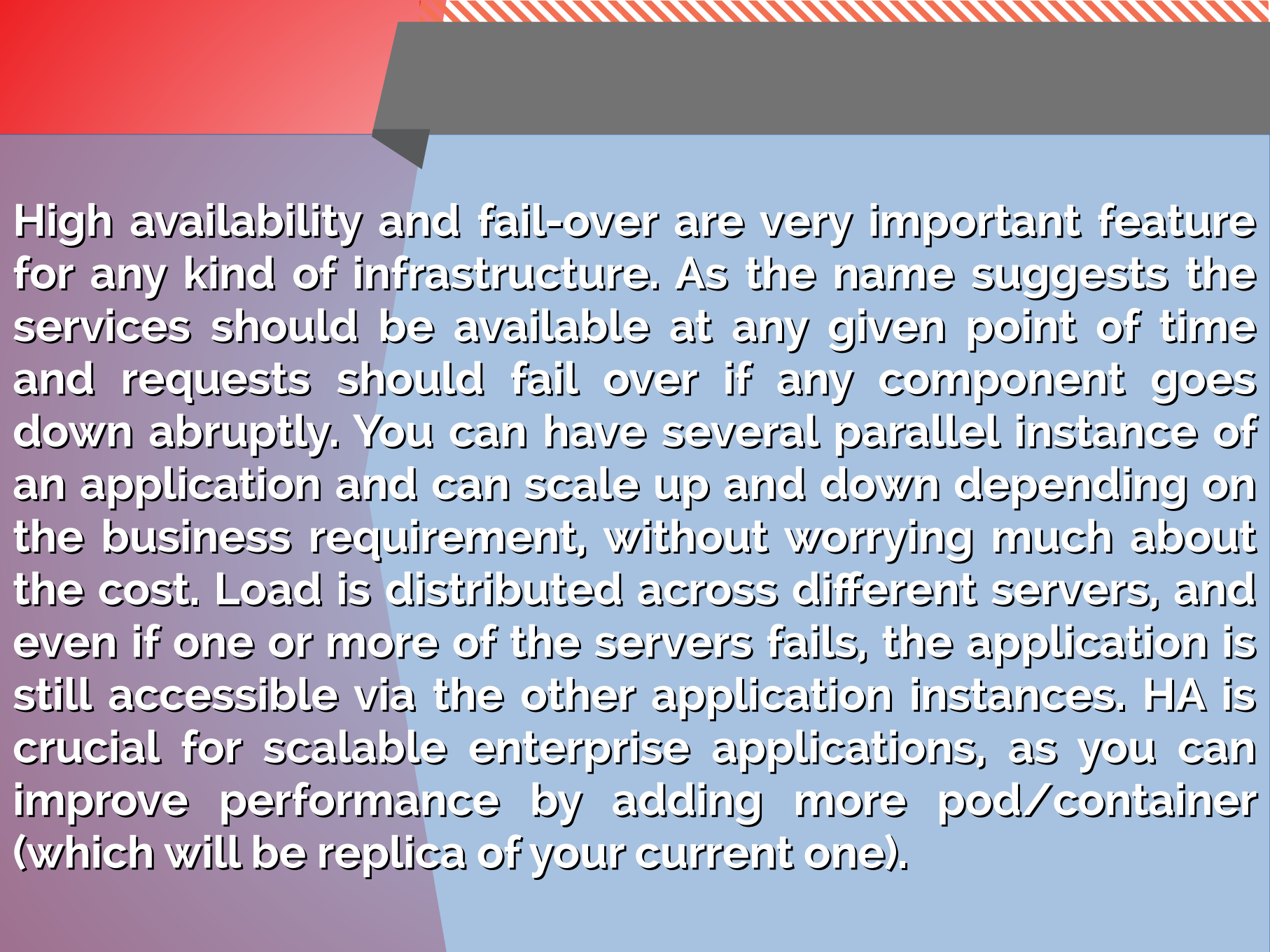


OPENS SHIFT



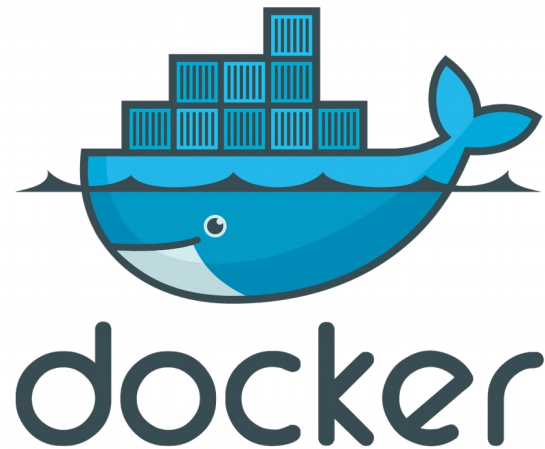
Agenda

- **What Problem Need to Solve**
- **Container & Docker Solution**
- **Kubernetes & OpenShift Solution**
- **Building Simple Application with Openshift**
- **Building JBOSS Cluster with OpenShift**
- **Auto scaling JBOSS Node Cluster with Openshift**
- **Bonus: Continues Integration and Delivery with OpenShift, Jenkins, SonarQube, Nexus and Gogs**



High availability and fail-over are very important feature for any kind of infrastructure. As the name suggests the services should be available at any given point of time and requests should fail over if any component goes down abruptly. You can have several parallel instance of an application and can scale up and down depending on the business requirement, without worrying much about the cost. Load is distributed across different servers, and even if one or more of the servers fails, the application is still accessible via the other application instances. HA is crucial for scalable enterprise applications, as you can improve performance by adding more pod/container (which will be replica of your current one).

Introduction to Container & Docker



The Problem

Cargo Transport Pre-1960



Solution?

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



Another Question!!

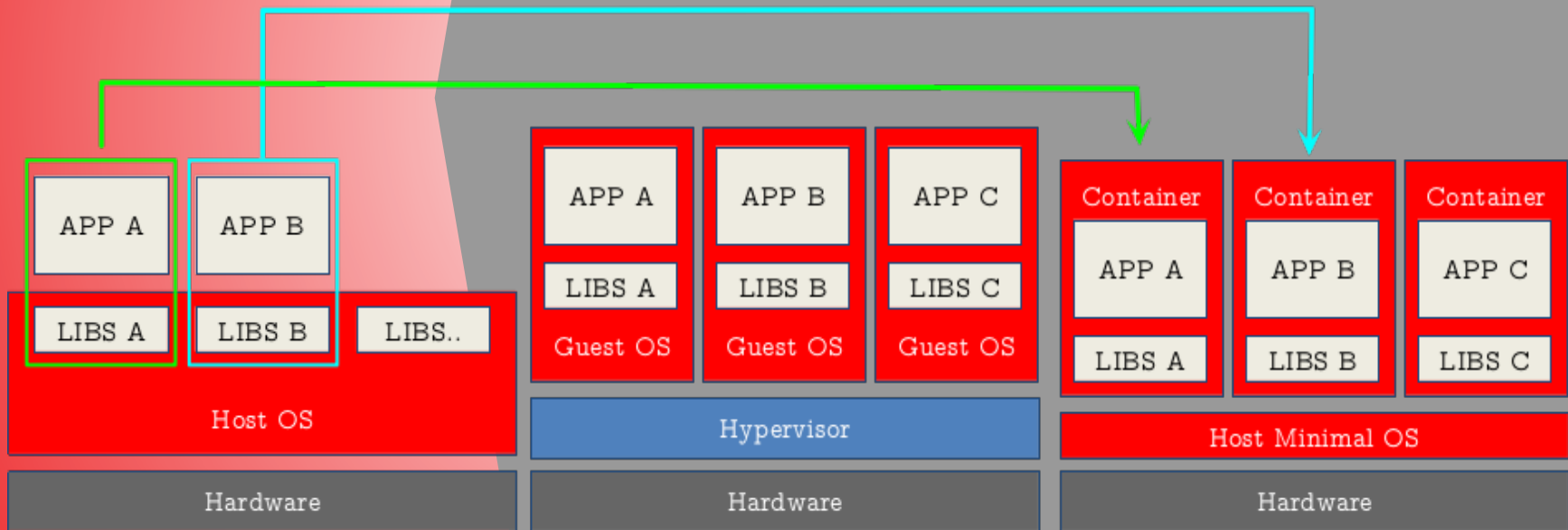
How about Application Development & Deployment???

The Evolution

Traditional
shared

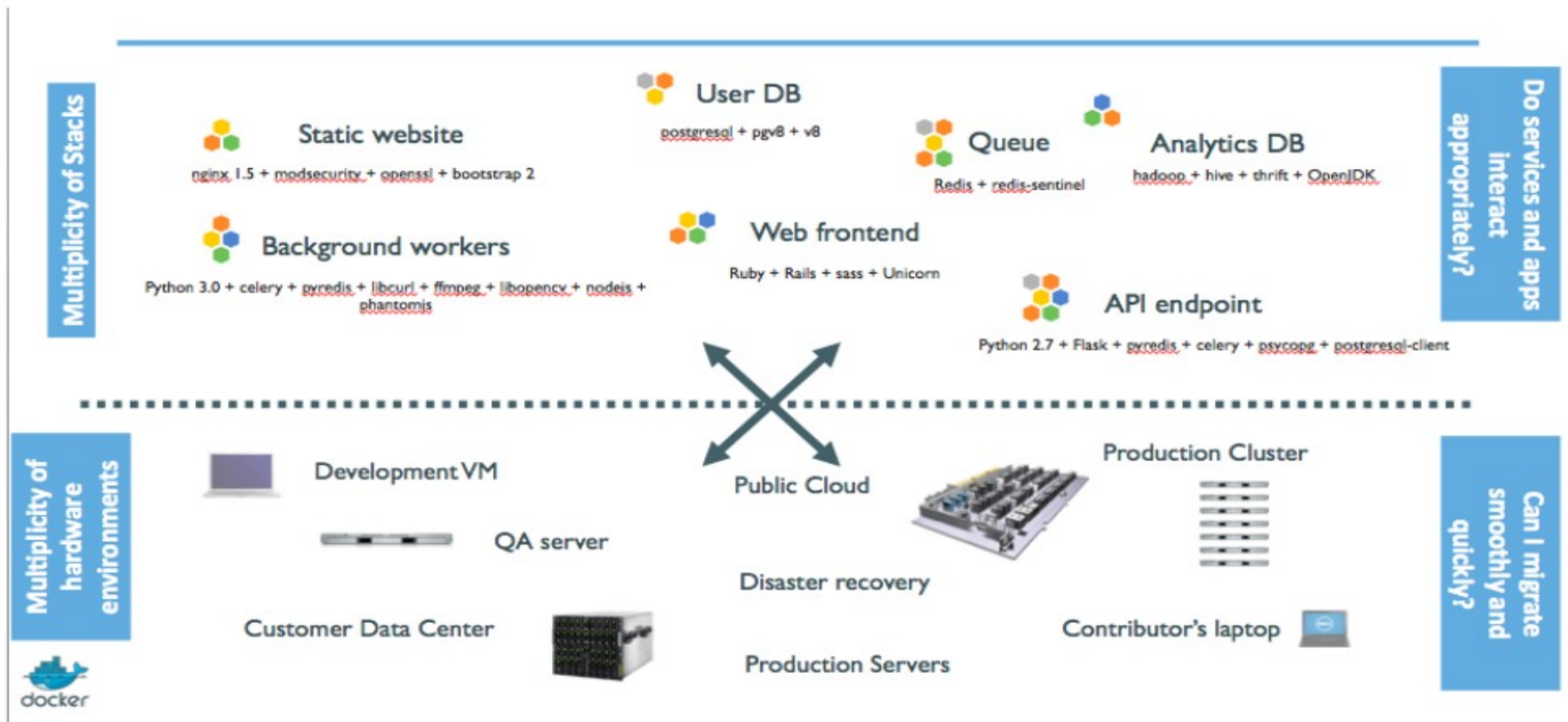
Virtual
system isolation

Container
process isolation



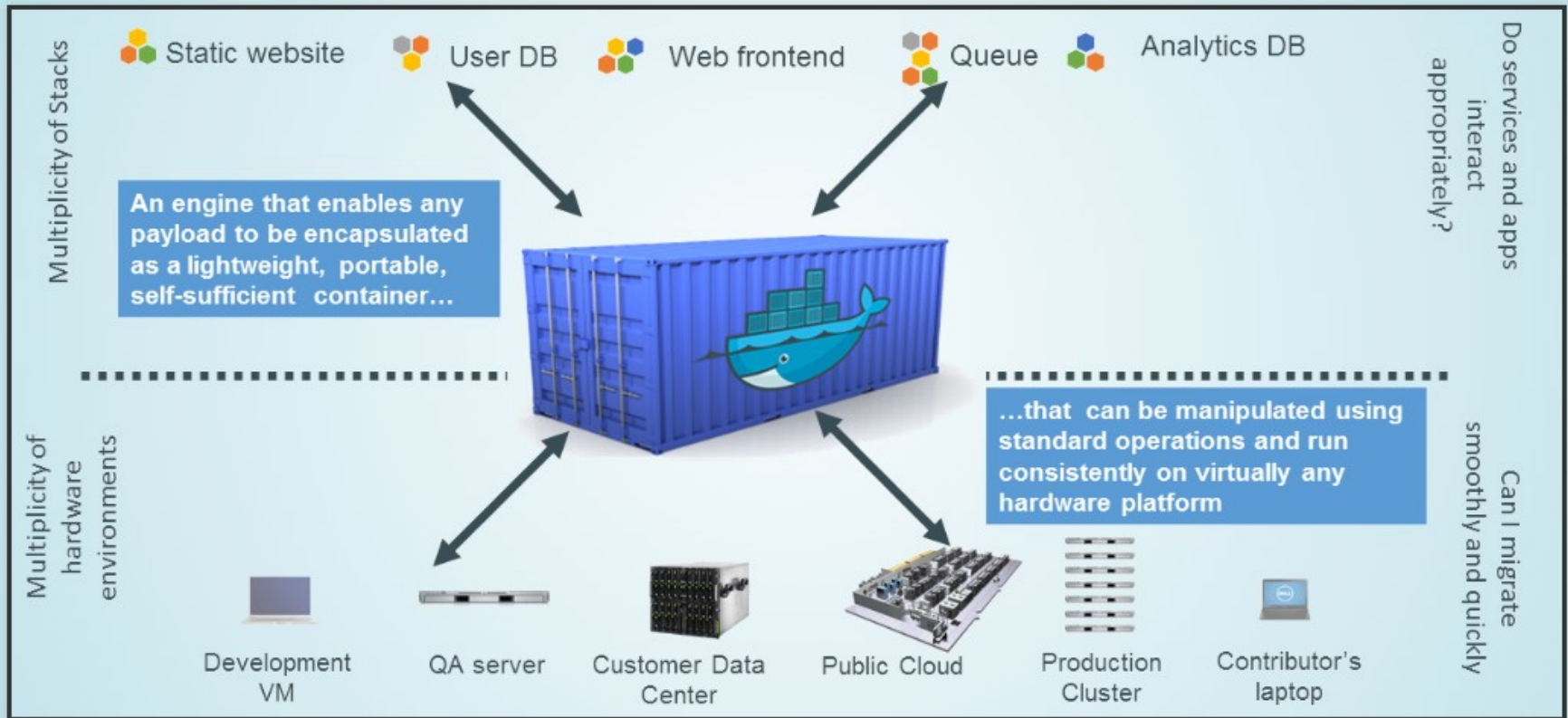
The App Problem

The deployment problem



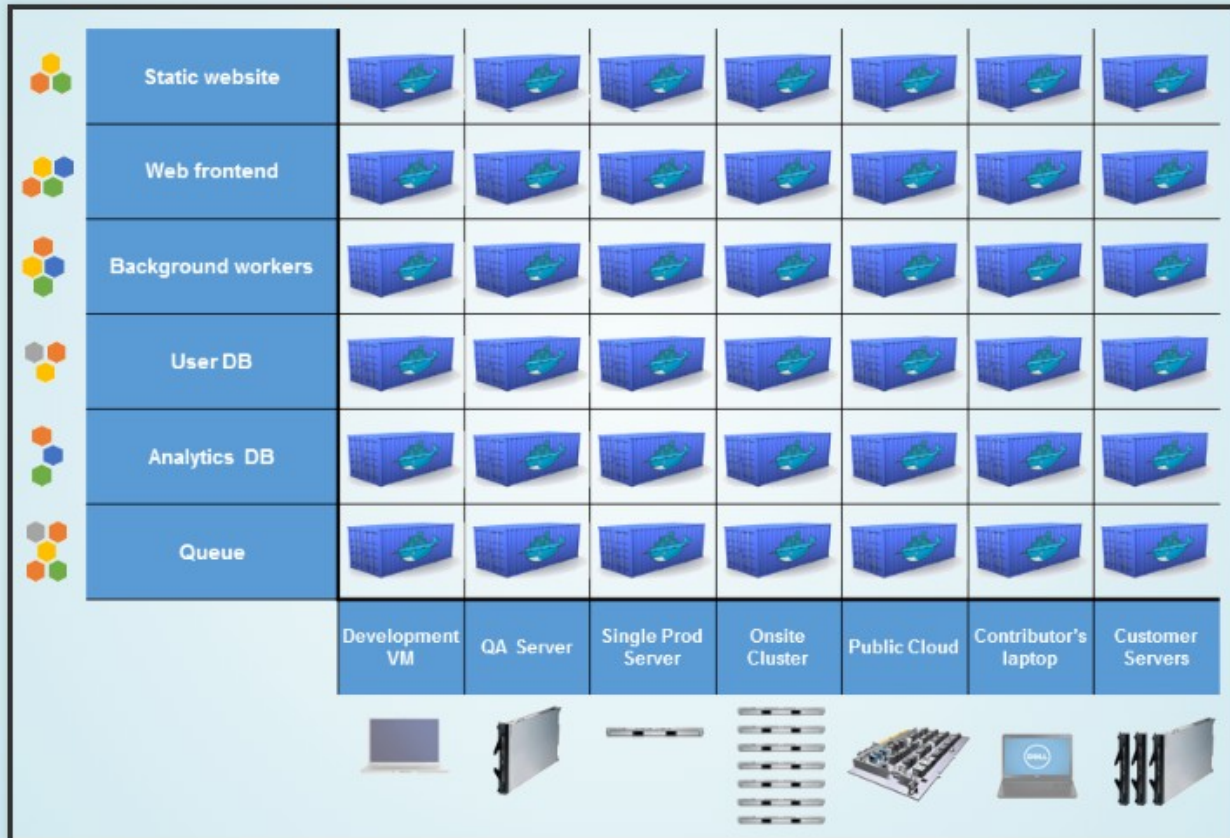
The App Solution

Docker is a Container System for Code



The App Solution

Docker Eliminates the Matrix from Hell



Advantages of Containers

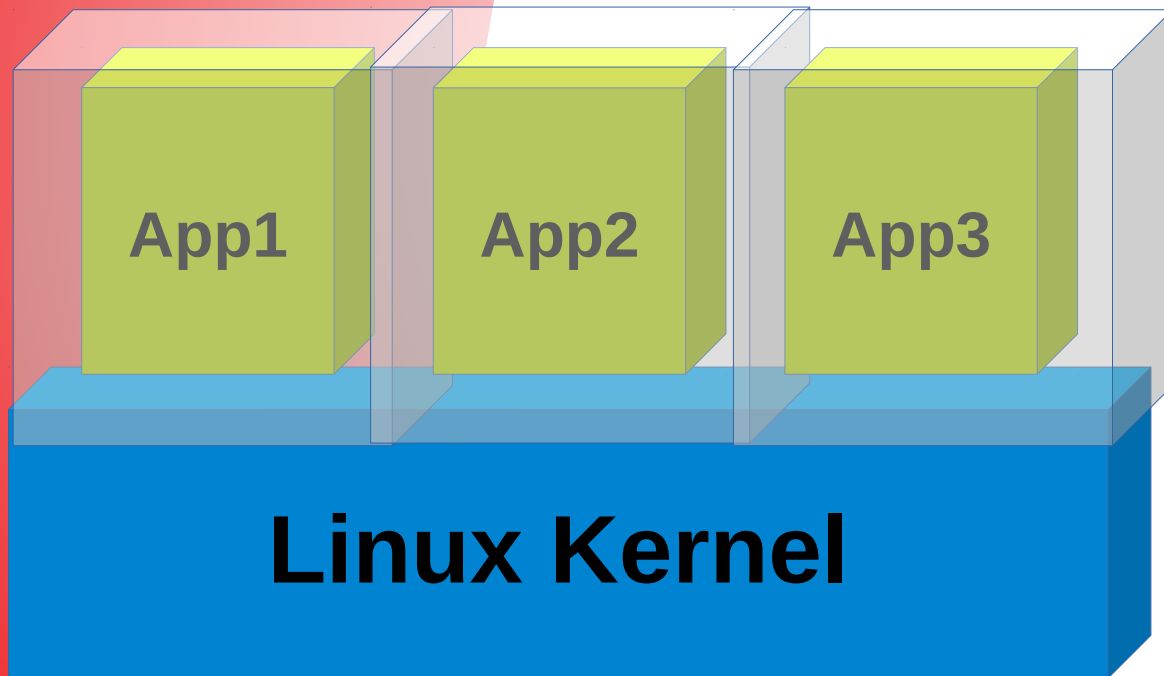
- **Low hardware footprint**
 - Uses OS internal features to create an isolated environment where resources are managed using OS facilities such as namespaces and cgroups. This approach minimizes the amount of CPU and memory overhead compared to a virtual machine hypervisor. Running an application in a VM is a way to create isolation from the running environment, but it requires a heavy layer of services to support the same low hardware footprint isolation provided by containers.
- **Environment isolation**
 - Works in a closed environment where changes made to the host OS or other applications do not affect the container. Because the libraries needed by a container are self-contained, the application can run without disruption. For example, each application can exist in its own container with its own set of libraries. An update made to one container does not affect other containers, which might not work with the update.

Advantages of Containers cont..

- **Quick deployment**
 - Deploys any container quickly because there is no need for a full OS install or restart. Normally, to support the isolation, a new OS installation is required on a physical host or VM, and any simple update might require a full OS restart. A container only requires a restart without stopping any services on the host OS.
- **Multiple environment deployment**
 - In a traditional deployment scenario using a single host, any environment differences might potentially break the application. Using containers, however, the differences and incompatibilities are mitigated because the same container image is used.
- **Reusability**
 - The same container can be reused by multiple applications without the need to set up a full OS. A database container can be used to create a set of tables for a software application, and it can be quickly destroyed and recreated without the need to run a set of housekeeping tasks. Additionally, the same database container can be used by the production environment to deploy an application.

Is not Virtualization :)

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**

Advantages of Containers

Developer Say:

Build Once, Run Anywhere

A clean, safe, hygienic, portable runtime environment for your app.

No worries about missing dependencies, packages and other pain points during subsequent deployments.

Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying.

Automate testing, integration, packaging...anything you can script.

Reduce/eliminate concerns about compatibility on different platforms, either your own or your customers.

Cheap, zero-penalty containers to deploy services. A VM without the overhead of a VM. Instant replay and reset of image snapshots.

Advantages of Containers

Operator: **Configure Once, Run Anything**

Make the entire lifecycle more efficient, consistent, and repeatable

Increase the quality of code produced by developers.

Eliminate inconsistencies between development, test, production, and customer environments.

Support segregation of duties.

Significantly improves the speed and reliability of continuous deployment and continuous integration systems.

Because the containers are so lightweight, address significant performance, costs, deployment, and portability issues normally associated with VMs.

Introduction to JBOSS EAP



JBoss Enterprise Application Platform

Build once, deploy Java EE apps everywhere

Red Hat® JBoss® Enterprise Application Platform (JBoss EAP) delivers enterprise-grade security, performance, and scalability in any environment. Whether on-premise; virtual; or in private, public, or hybrid clouds, JBoss EAP can help you deliver apps faster, everywhere.

JBoss Enterprise Application Platform

Optimized for cloud and containers

JBoss EAP 7 is built to provide simplified deployment and full Java™ EE performance for applications in any environment.

Whether on-premise or in virtual, private, public, and hybrid clouds, JBoss EAP features a modular architecture that starts services only as they are required. The extreme low memory footprint and fast startup times mean that JBoss EAP is ideal for environments where efficient resource utilization is a priority, such as Red Hat OpenShift.

Jboss Enterprise Application Platform

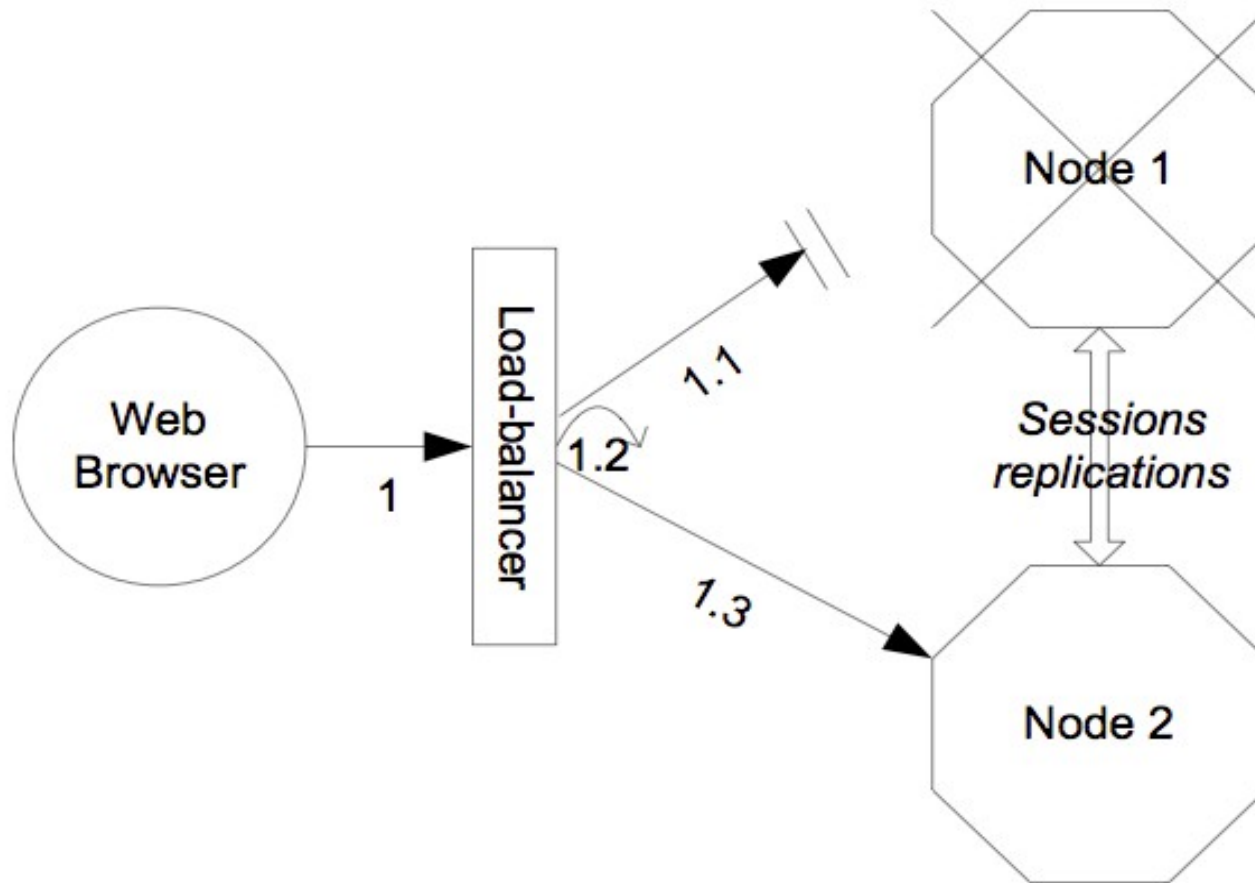
More detail about JBOSS EAP please visit:

<https://www.redhat.com/en/technologies/jboss-middleware/application-platform>

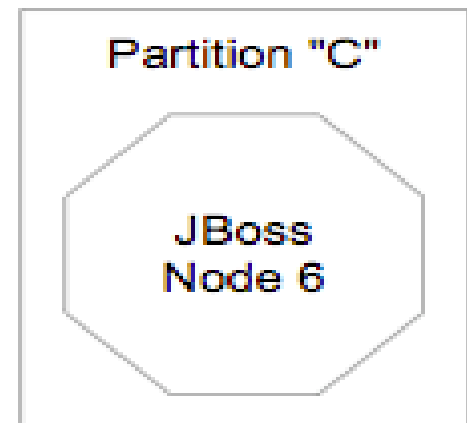
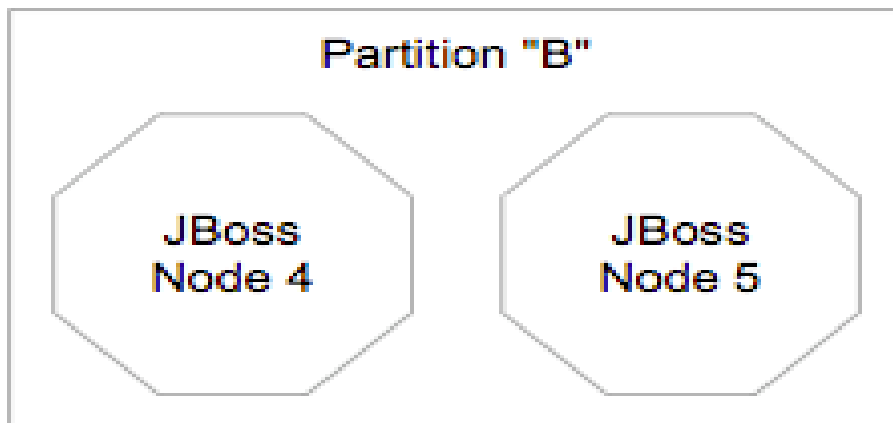
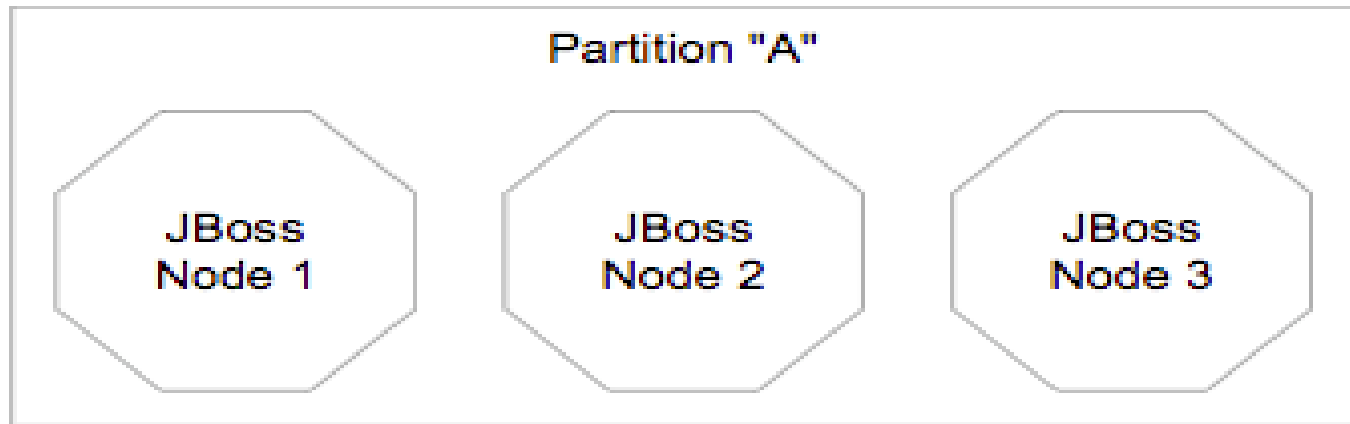
Jboss EAP Cluster Term

Clustering allows us to run an applications on several parallel servers (a.k.a cluster nodes). The load is distributed across different servers, and even if any of the servers fails, the application is still accessible via other cluster nodes. Clustering is crucial for scalable enterprise applications, as you can improve performance by simply adding more nodes to the cluster.

JBOSS Cluster Topology



JBOSS Cluster Topology



Deploying JBOSS Cluster

- Install Linux on 2 Server / VM – Duration 60 Minutes
- Install JBOSS on 2 Server / VM – Duration 35 Minutes
- Configure JBOSS Cluster & Testing Cluster – 35 Minutes
- Deploy Application and Configure Datastore – 20 Minutes
- Testing Application and Session Replication – 25 Minutes
- Configure Load Balancer for 2 JBOSS Server – 60 Minutes

Total Time: 235 Minute!! / 4 Hours for Deploying 2 Node JBOSS Cluster

Adding Another JBOSS Node to Cluster: +2 Hours

Another Question!!

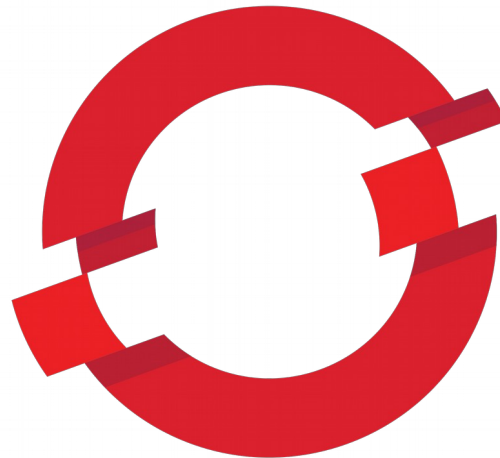
So... How to Reduce Deployment time???

Short Answer: Dockerize your JBOSS!!

How to make JBOSS Cluster more easy to Manage & Scale???

We not only need Docker, we also need orchestration

Its time to Openshift!!!



OPENSIFT

Yess!!..its OpenShift

OpenShift is an auto-scalable Platform as a Service. Auto-scalable means OpenShift can horizontally scale your application up or down depending on the number of concurrent connections. OpenShift supports the JBoss application server, which is a certified platform for Java EE 6/7 development. As an OpenShift user, you have access to both the community version of JBoss and JBoss EAP 6/7(JBoss Enterprise Application Platform) for free.

We Need more than just Orchestration

Self Service

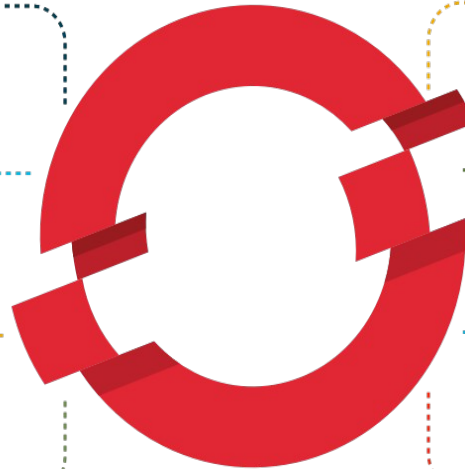
- Templates
- Web Console

Multi-Language

Automation

- Deploy
- Build

DevOps Collaboration



OPENSHIFT®

by Red Hat®



Secure

- Namespaced
- RBAC

Scalable

- Integrated LB

Open Source

Enterprise

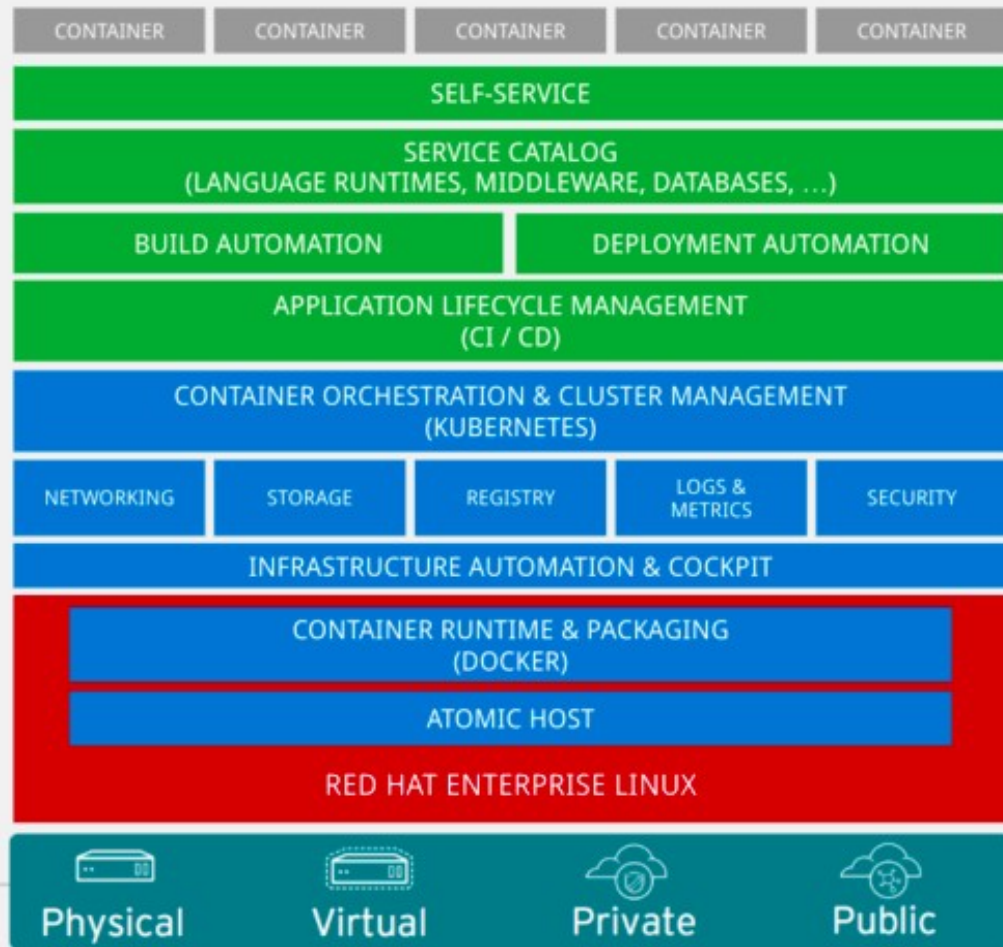
- Authentication
- Web Console
- Central Logging

OpenShift is Red Hat Container Application
Platform (PaaS)

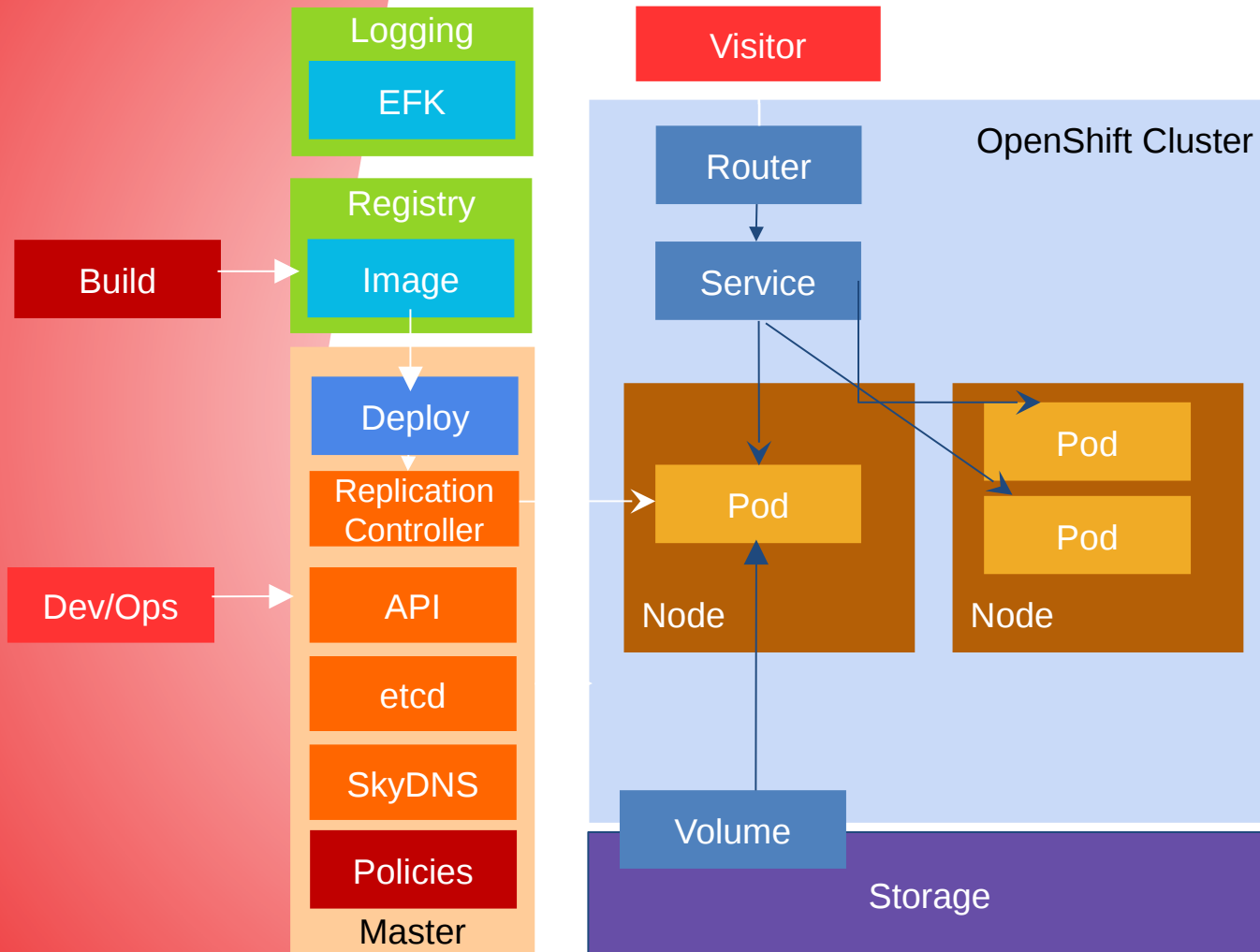
OpenShift=Enterprise K8s

OpenShift - Enterprise Kubernetes

Build, Deploy and Manage Containerized Apps



OpenShift Build & Deploy Architecture



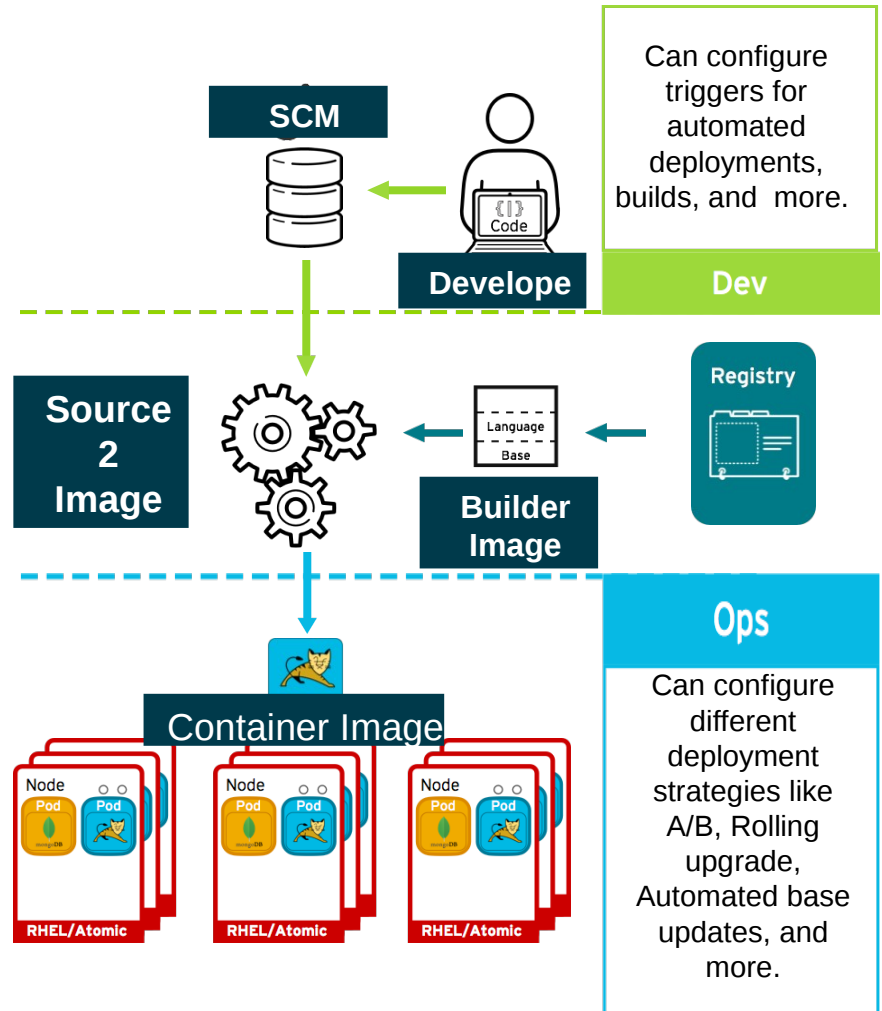
Build & Deploy an Image

Builder Images

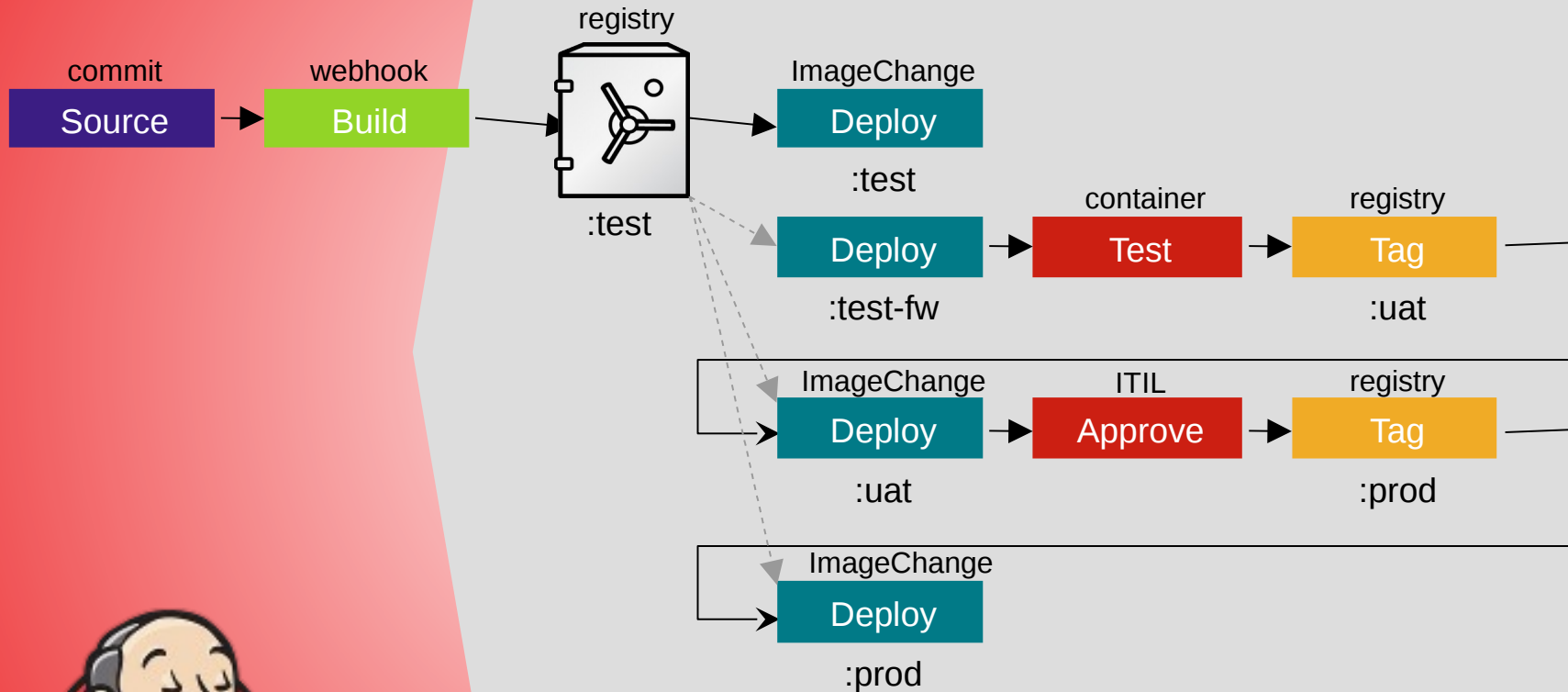
- Jboss-EAP
- PHP
- Python
- Ruby
- Jenkins
- Customer
 - C++ / Go
 - S2I (bash) scripts

Triggers

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



Continuous Integration Pipeline Example



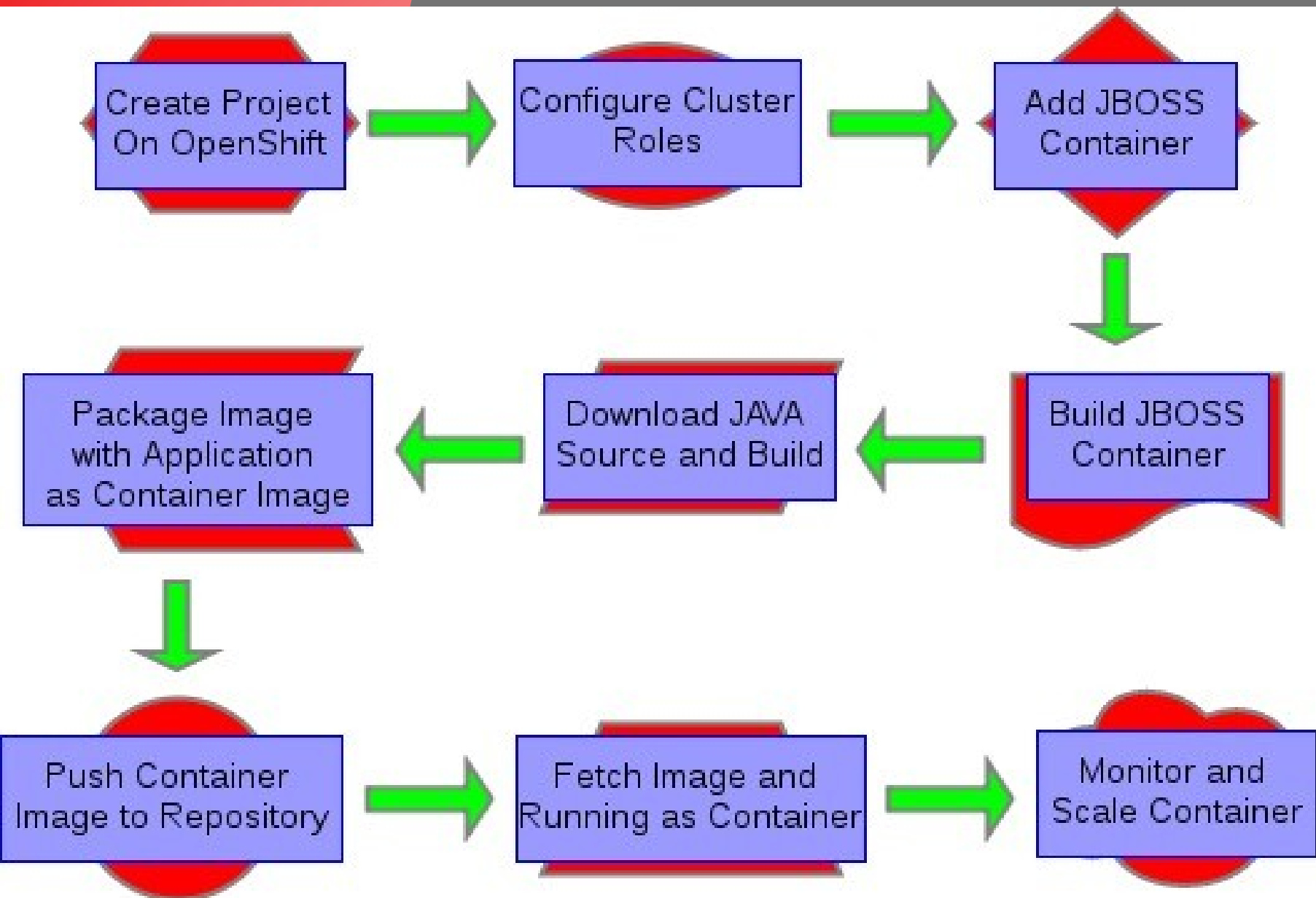
Demo time!!... Yeay...

Any Question?

**Lets continue to
Openshift Demo..**

Its time to Openshift!!!

DEMO



Deploying JBOSS EAP

Create project, and assign cluster roles

```
oc delete project haexample
oc new-project haexample
oc policy add-role-to-user view system:serviceaccount:haexample:default -n
haexample
```

Create App and Define Cluster Partition

```
oc new-app --image-stream="openshift/jboss-eap70-openshift:latest"
https://github.com/isnuryusuf/haexample#master --name='haexample' -l
name='haexample' -e SELECTOR=haexample -e
OPENSIFT_KUBE_PING_NAMESPACE=haexample
OPENSIFT_KUBE_PING_LABELS=app=haexample

oc env dc/haexample -e OPENSIFT_KUBE_PING_NAMESPACE=haexample
OPENSIFT_KUBE_PING_LABELS=app=haexample
```

For Installing OpenSift, Please Refer to: <https://github.com/isnuryusuf/openshift-install/blob/master/openshift-origin-quickstart.md>

Setting up Cluster

Configuring Cluster on Openshift

```
ports:
  - containerPort: 8080
    protocol: TCP
  - containerPort: 8443
    protocol: TCP
  - containerPort: 8778
    protocol: TCP
  - name: ping
    containerPort: 8888
    protocol: TCP
```

Its Running!!!

```
[org.jgroups.protocols.openshift.KUBE_PING] (MSC service thread 1-5)
namespace [haexample] set; clustering enabled
```

```
[org.jgroups.protocols.openshift.KUBE_PING] (MSC service thread 1-4)
namespace [haexample] set; clustering enabled
```

```
[org.infinispan.remoting.transport.jgroups.JGroupsTransport] (MSC service
thread 1-5) ISPN000094: Received new cluster view for channel server:
[haexample-2-pxsd9|0] (1) [haexample-2-pxsd9]
```

Configure Auto Scale

Configuring Auto Scale

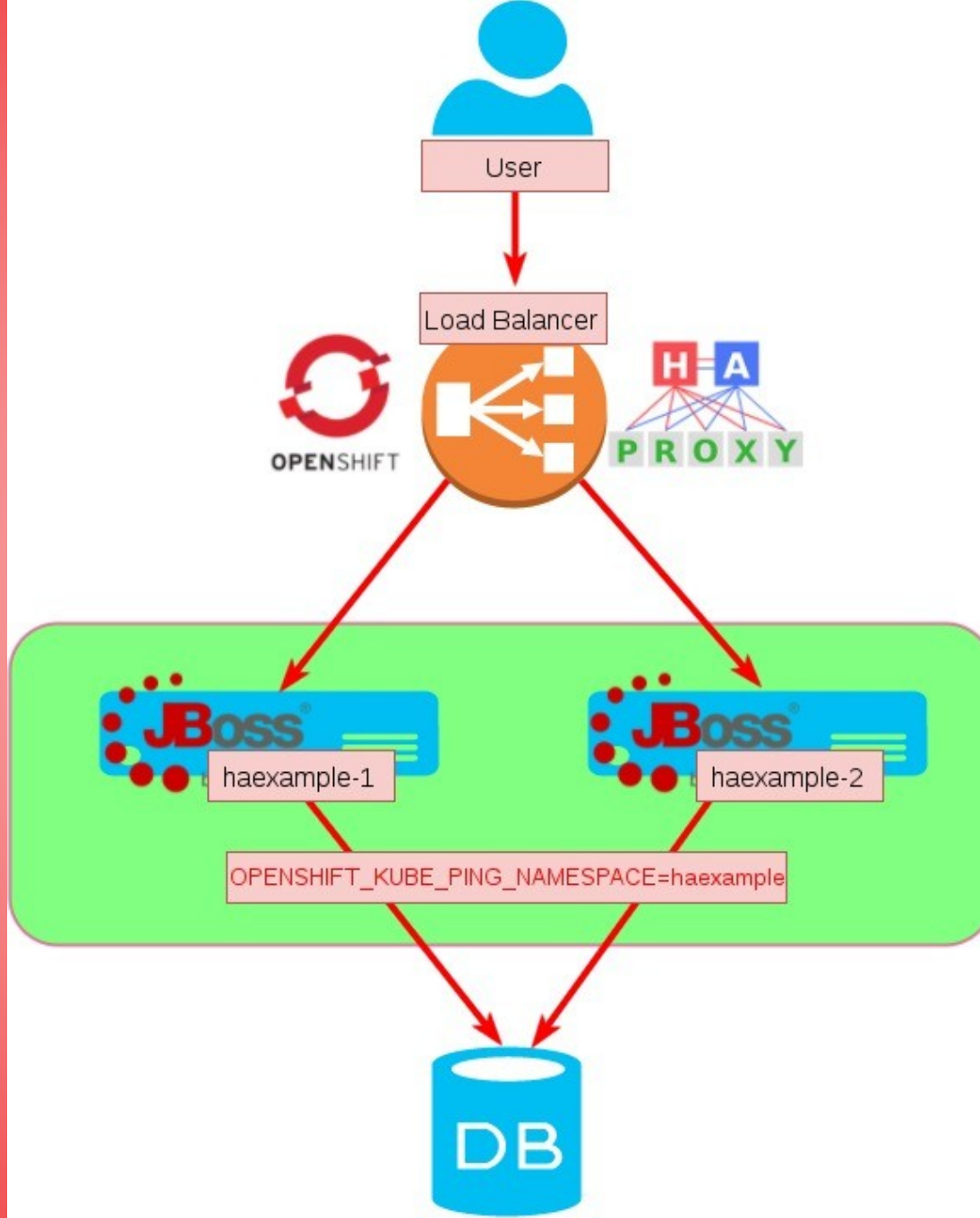
```
oc expose service haexample --name haexample
oc get svc
oc create -f limits.json -n haexample
oc describe limits mylimits
oc autoscale dc/haexample --min 1 --max 10 --cpu-percent=90

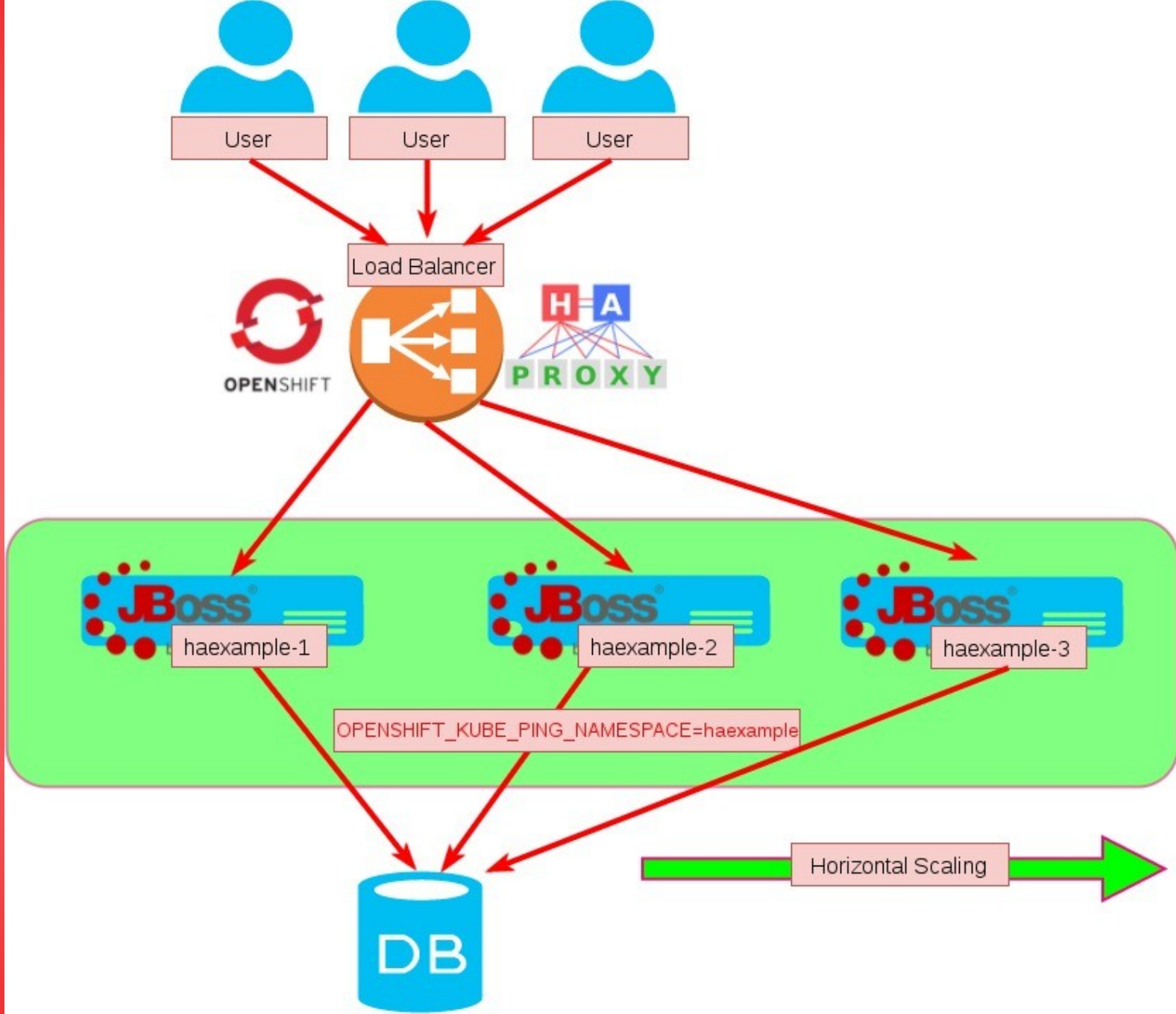
oc describe dc/haexample
oc get hpa
```

Testing Auto Scale

```
for i in {1..500}; do curl http://haexample-haexample.smartfintech.i3-
cloud.com/haexample ; done;

while true; do curl -s http://haexample-haexample.smartfintech.i3-
cloud.com/haexample | grep "haexample" | cut -c 55-95; sleep 2; done
```





More Detail

Complete Solution,
Please visit:

<https://github.com/isnuryusuf/haexample>

Thnks 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>



Graha BIP 6th Floor

Jl. Jend. Gatot Subroto Kav. 23
Jakarta 12930, Indonesia

 Phone : (62) 21 290 23393

 Fax : (62) 21 525 8065

 info@i-3.co.id

 www.i-3.co.id



Inovasi Informatika Indonesia



I3_ID