



# **Reference Architectures 2017 Building JBoss EAP 7 Microservices on OpenShift Container Platform**

---

Babak Mozaffari



# Reference Architectures 2017 Building JBoss EAP 7 Microservices on OpenShift Container Platform

---

Babak Mozaffari  
refarch-feedback@redhat.com

## Legal Notice

Copyright © 2017 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

This reference architecture builds on previous work on designing, developing and deploying microservices on JBoss EAP 7, and builds a microservice architecture environment on OpenShift Container Platform 3.4. The audience for this paper includes Java EE software architects and developers engaged in either Greenfield or Brownfield projects with the goal of creating a microservice architecture and/or using JBoss EAP 7 on OpenShift. While working knowledge of Java EE and JBoss EAP is a prerequisite, no OpenShift experience is assumed. The reader is walked through the installation of configuration of oCP 3.4 in a trial environment using the quick installation approach, while referred to official documentation and other resources for a supported production environment setup. The paper proceeds to describe, step by step, the process of building and deploying an application that runs on OCP 3.4, leveraging supported xPaaS and database images.

---

## Table of Contents

<b>COMMENTS AND FEEDBACK</b> .....	<b>3</b>
<b>CHAPTER 1. EXECUTIVE SUMMARY</b> .....	<b>4</b>
<b>CHAPTER 2. REFERENCE ARCHITECTURE ENVIRONMENT</b> .....	<b>5</b>
<b>CHAPTER 3. CREATING THE ENVIORNMENT</b> .....	<b>6</b>
3.1. OVERVIEW	6
3.2. PRODUCTION ENVIRONMENT SETUP	6
3.3. TRIAL ENVIRONMENT SETUP	6
3.4. BUILD AND DEPLOY	14
3.5. RUNNING THE APPLICATION	20
<b>CHAPTER 4. DESIGN AND DEVELOPMENT</b> .....	<b>23</b>
4.1. OVERVIEW	23
4.2. APPLICATION STRUCTURE	23
4.3. CUSTOMIZING THE APPLICATION SERVER	23
<b>CHAPTER 5. CONCLUSION</b> .....	<b>25</b>
<b>APPENDIX A. REVISION HISTORY</b> .....	<b>26</b>
<b>APPENDIX B. CONTRIBUTORS</b> .....	<b>27</b>
<b>APPENDIX C. ANSIBLE HOSTS</b> .....	<b>28</b>
<b>APPENDIX D. REVISION HISTORY</b> .....	<b>29</b>



## COMMENTS AND FEEDBACK

In the spirit of open source, we invite anyone to provide feedback and comments on any reference architecture. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers. Feedback on the papers can be provided by emailing [refarch-feedback@redhat.com](mailto:refarch-feedback@redhat.com). Please refer to the title within the email.

## CHAPTER 1. EXECUTIVE SUMMARY

This reference architecture builds upon previous work describing microservices and a pattern for building a microservice architecture using **Red Hat JBoss Enterprise Application Platform 7.0**, to take advantage of **OpenShift Container Platform 3.4 by Red Hat** and build an on-premise cloud environment to host the microservice architecture environment.

**OpenShift Container Platform 3.4 by Red Hat** is designed for on-premise, public, or hybrid cloud deployments. Built with proven open source technologies, OpenShift is a [Platform-as-a-Service](#) (PaaS) that helps application development and IT operations teams create and deploy apps with the speed and consistency that business demands.

This effort assumes a properly configured OpenShift production environment and refers the reader to documents and resources to help install and configure such an environment. For convenience, the setup and configuration of a testing environment for OpenShift is described using the quick installation approach. The microservice architecture project is largely imported from [previous reference architecture work](#) but formatted to take advantage of the host environment.



## CHAPTER 2. REFERENCE ARCHITECTURE ENVIRONMENT

This reference architecture designs, develops and deploys a simple distributed microservice architecture on Red Hat JBoss Enterprise Application Platform 7.0 on top of OpenShift. The application architecture is based on the design laid out and explained in depth in the reference architecture document on [EAP Microservice Architecture](#).

The OpenShift environment includes a highly available master and two nodes. An alternatively trial configuration is described to use quick installation and configure a single stand-alone master, which is not intended for use in a production environment. The application is set up as a single OpenShift project, called *msa*.

This project includes six services, where two of these are database services based on the standard and supported [MySQL image](#). The remaining four images are based on the supported JBoss EAP [xPaaS image](#) and rely on the [Source to Image \(S2I\)](#) functionality to build and deploy. The EAP images are customized for two of these services to add a MySQL datasource.

All OpenShift services can be configured and scaled with the desired number of replicas. In a default OpenShift installation, the nodes are the only targets for service deployment. However, the masters can also be configured as scheduleable in order to host service replicas.

The *presentation* service is the only entry point to the application and serves HTML to the client browser over HTTP. This web tier is stateless, and can therefore be replicated without concern for multicast communication or alternative EAP clustering setup. This services relies on the *product*, *sales*, and *billing* services, while the first two in turn rely on the MySQL *product-db* and *sales-db* services.

## CHAPTER 3. CREATING THE ENVIRONMENT

### 3.1. OVERVIEW

This reference architecture can be deployed on either a production or a trial environment. In both cases, it is assumed that *ocp-master1* refers to one (or the only) OpenShift master host and that the environment includes two OpenShift node hosts with the host names of *ocp-node1* and *ocp-node2*.

It is further assumed that OpenShift has been installed by the *root* user and that a regular user has been created with basic access to the host machine, as well as access to OpenShift through its *identity providers*.

### 3.2. PRODUCTION ENVIRONMENT SETUP

Using this reference architecture for a production environment relies on an existing installation and configuration of an OpenShift environment. Refer to the official OpenShift [documentation](#) for further details.

A production environment requires [highly-available OpenShift masters](#), described and documented in a [reference architecture](#) on Deploying an OpenShift Enterprise 3 Distributed Architecture.

The complete configuration of a production environment depends on various network and server details, which are beyond the scope of this document. The product should be installed through the [advanced installation](#) approach. Refer to [Appendix C: Ansible Host Configuration File](#) for a sample **Ansible** hosts configuration file. This file is provided as a starting point and should be customized based on your environment.

### 3.3. TRIAL ENVIRONMENT SETUP

For faster and easier setup, OpenShift can be installed with a single master and multiple nodes. This configuration introduces a single point of failure for the master, but not for services, as they can have the desired number of replicas and be deployed on multiple OpenShift nodes as well as the master. However, failure of master services [reduces the ability](#) of the system to respond to application failures or creation of new applications.

This section describes the installation and configuration of a trial environment of OpenShift, as validated in a lab environment with three virtual machines running Red Hat Enterprise Linux 7.1, where one is configured as the master and the other two as nodes. The master is configured with a hostname of ***ocp-master1*** and IP addresses of *10.xxx.xxx.41*. The two nodes have hostnames of ***ocp-node1*** and ***ocp-node2***, with respective IP addresses of *10.xxx.xxx.42* and *10.xxx.xxx.43*.

#### 3.3.1. Operating System

Perform a basic and standard installation of Red Hat Enterprise Linux 7.1 on all machines. After successful installation, configure and satisfy the [software prerequisites](#). Execute all the following instructions as the *root* user.

Start by using **Red Hat Subscription-Manager** to register each system:

```
# subscription-manager register --username USER --password PASSWORD --  
auto-attach
```

Begin by disabling all repositories:

```
# subscription-manager repos --disable="*"
```

Then, explicitly enabled repositories for the server and OpenShift:

```
# subscription-manager repos \
  --enable="rhel-7-server-rpms" \
  --enable="rhel-7-server-extras-rpms" \
  --enable="rhel-7-server-ose-3.4-rpms"
```

Use **yum** to install the dependencies:

```
# yum -y install wget git net-tools bind-utils iptables-services
bridge-utils bash-completion
# yum -y update
```

Also install the OpenShift utilities, required by the installation process:

```
# yum -y install atomic-openshift-utils
```

### 3.3.2. Network Configuration

The [network requirements](#) for OpenShift include a shared network between the master and node hosts. In this trial setup, all hosts are simply set up in the same subnet.

A wildcard for a DNS zone must ultimately resolve to the IP address of the server that hosts the OpenShift router. This trial environment includes a single router deployed on the master node. Find the configured name server on the master and node hosts:

```
# cat /etc/resolv.conf
```

To look up each name server, use **nslookup** for a given *IP address*, for example:

```
# nslookup 10.xxx.xxx.247
```

Configure the name server for these hosts and create a wildcard DNS entry to point to the master. The steps to create a DNS entry vary for each naming server software. For example, to configure **Berkeley Internet Name Domain** (BIND), also known as **named**, edit the configuration file:

```
# vi /etc/named.conf
```

Add a zone file at the bottom of the configuration file:

```
zone "example.com" {
  type master;
  file "example.com.db";
};
```

In the zone file, configure the desired subdomain to point to the OpenShift master:

```
example.com. 300 IN SOA example.com. root.example.com. (
  2008031801 ; Serial
  15 ; Refresh every 15 minutes
```

```
3600 ; Retry every hour
3000000 ; Expire after a month+
86400 ) ; Minimum ttl of 1 day
IN NS example.com.

IN MX 10 example.com.

IN A 10.xxx.xxx.41
*.example.com. IN A 10.xxx.xxx.41
```

Restart the name server to have the changes take effect:

```
# service named restart
```

Validate the wildcard DNS entry by pinging a subdomain from one of the servers and making sure it resolves to the configured IP address:

```
# ping test.example.com
```

### 3.3.3. Docker Setup

Install docker:

```
# yum -y install docker
```

To allow the Docker daemon to trust any Docker registry on the subnet, rather than requiring a certificate, edit the `/etc/sysconfig/docker` file and add an option to allow an insecure registry on the indicated addresses:

```
INSECURE_REGISTRY='--insecure-registry 172.30.0.0/16'
```

Docker containers and images are created in an ephemeral storage space by default. This default storage back end is a thin pool on loopback devices which is not supported for production use and only appropriate for proof of concept environments. For production environments, you must create a thin pool logical volume and [re-configure Docker](#) to use that volume.

This requires leaving free space available when provisioning your host. If free space is not available, add a *Virtual IO Disk* to the virtual machine. After starting the server, find the newly created partition:

```
# cat /proc/partitions
```

A second virtual disk would typically be called `vdb`. Create a physical driver from this partition:

```
# pvcreate /dev/vdb
```

Extend your volume group to include this physical drive:

```
# vgextend myvg /dev/vdb
```

The above steps, if necessary, would create free space in your host volume group and enable docker storage to be created.

Configure docker storage to use LVM thin pool:

```
# echo <<EOF > /etc/sysconfig/docker-storage-setup
SETUP_LVM_THIN_POOL=yes
EOF
```

Then use **docker-storage-setup** to create the LVM:

```
# docker-storage-setup
```

Remember to start, or restart docker:

```
# systemctl start docker
```

### 3.3.4. Ensuring Host Access

For both the quick and advanced installation methods, the installing user must have access to all hosts. Generate an *SSH* key pair on each host:

```
# ssh-keygen
```

Accept all default options and do not provide a password. Once the key is generated, copy it to all hosts:

```
# for host in ocp-master1 ocp-node1 ocp-node2; \
do ssh-copy-id -i ~/.ssh/id_rsa.pub $host; \
done
```

### 3.3.5. Quick Installation

[Install](#) OpenShift using the quick installation approach by running an interactive CLI utility:

```
# atomic-openshift-installer install
```

Confirm the installation and press *y* to continue. Then elect to install OpenShift as the root user.

Continue to the next step and select the default choice of *OpenShift Container Platform* as the variant to install:

```
Which variant would you like to install?
```

- (1) OpenShift Container Platform
- (2) Registry

```
Choose a variant from above: [1]:
```

Proceed to host configuration and when prompted, enter the IP address of the OpenShift master, press enter, and answer in the affirmative to indicate that this host will be the mater. Then select the default *rpm* installation approach for OpenShift:

```
Enter hostname or IP address: 10.xxx.xxx.41
Will this host be an OpenShift Master? [y/N]: y
Will this host be RPM or Container based (rpm/container)? [rpm]:
```

Continue and add the next two hosts to use them as nodes. Then continue to provide default, blank or optional answers to questions about storage host, default subdomain and proxy hostname.

Finally, verify that internal and public IP addresses of all hosts along with their hostnames have been correctly resolved. Note that after confirmation, an installation script will be generated and stored locally. You can use this script to later re-run the installation, add hosts to the cluster, or [upgrade your cluster](#):

```
A list of the facts gathered from the provided hosts follows. Because
it is
often the case that the hostname for a system inside the cluster is
different
from the hostname that is resolveable from command line or web clients
these settings cannot be validated automatically.
```

```
For some cloud providers the installer is able to gather metadata
exposed in
the instance so reasonable defaults will be provided.
```

```
Please confirm that they are correct before moving forward.
```

```
10.xxx.xxx.41,10.xxx.xxx.41,10.xxx.xxx.41,ocp-
master1.hostname.xxx.redhat.com,ocp-master1.hostname.xxx.redhat.com
10.xxx.xxx.42,10.xxx.xxx.42,10.xxx.xxx.42,ocp-
node1.hostname.xxx.redhat.com,ocp-node1.hostname.xxx.redhat.com
10.xxx.xxx.43,10.xxx.xxx.43,10.xxx.xxx.43,ocp-
node2.hostname.xxx.redhat.com,ocp-node2.hostname.xxx.redhat.com
```

```
Format:
```

```
connect_to,IP,public IP,hostname,public hostname
```

```
Notes:
```

- \* The installation host is the hostname from the installer's perspective.
- \* The IP of the host should be the internal IP of the instance.
- \* The public IP should be the externally accessible IP associated with the instance
- \* The hostname should resolve to the internal IP from the instances themselves.
- \* The public hostname should resolve to the external ip from hosts outside of the cloud.

```
Do the above facts look correct? [y/N]: y
```

```
Wrote atomic-openshift-installer config:
```

```
/root/.config/openshift/installer.cfg.yml
```

```
Wrote ansible inventory: /root/.config/openshift/hosts
```

```
Ready to run installation process.
```

```
If changes are needed please edit the config file above and re-run.
```

```
Are you ready to continue? [y/N]: y
```

Confirm the installation to move forward. The script will proceed to install OpenShift.

Verify successful installation by making sure that the script reports no *failed* status for any of the hosts:

#### PLAY RECAP

```
10.xxx.xxx.41 : ok=451 changed=10 unreachable=0 failed=0
10.xxx.xxx.42 : ok=103 changed=1 unreachable=0 failed=0
10.xxx.xxx.43 : ok=103 changed=1 unreachable=0 failed=0
localhost : ok=23 changed=0 unreachable=0 failed=0
```

### 3.3.6. Configuring Authentication

Developers and administrators authenticate against OpenShift through its built-in **OAuth** Server. A default installation of OpenShift has a security policy of *Deny All*, which denies access to all users.

[Configure authentication](#) by editing the master configuration file, set to `/etc/origin/master/master-config.yaml` by default:

```
# vi /etc/origin/master/master-config.yaml
```

Note that OpenShift supports several authentication mechanisms, including [LDAP Authentication](#).

Select a proper identity provider, for example, the **HTPasswdPasswordIdentityProvider**:

oauthConfig:

```
...
identityProviders:
- name: htpasswd
  challenge: True
  login: True
  provider:
    apiVersion: v1
    kind: HTPasswdPasswordIdentityProvider
    file: /etc/origin/master/users.htpasswd
```

To use this identity provider, install the **htpasswd** package on the master host:

```
# yum -y install httpd-tools.x86_64
```

Configure the desired users and passwords with this tool, using the `-c` flag the first time to create a new authentication file:

```
# htpasswd -c /etc/origin/master/users.htpasswd babak
```

Add corresponding Linux users:

```
# useradd babak
# passwd babak
```

Restart the service for changes to take effect:

```
# systemctl restart atomic-openshift-master
```

### 3.3.7. Default Application Subdomain

When a route is created for a service, the default address is constructed by appending the configured domain to the service name. The default subdomain can be specified during installation, or verified or changed by editing the master configuration file:

```
# vi /etc/origin/master/master-config.yaml
```

Set the *subdomain* attribute to the value used for the wildcard DNS entry:

```
routingConfig:
  subdomain: "example.com"
```

Restart the OpenShift service for any potential change to the default subdomain value to take effect:

```
# systemctl restart atomic-openshift-master
```

### 3.3.8. Deploying to the OpenShift Master

By default, any hosts you designate as masters during the installation process will also be configured as nodes that are marked unschedulable. To deploy the router or any other service on the master host, you can [configure it as schedulable](#):

```
# oadm manage-node ocp-master1.hostname.example.com --schedulable=true
```

#### Warning

In production environments and for production use, it is best to exclusively use node hosts for application deployment and avoid deploying services to master hosts.

### 3.3.9. Deploying the Docker Registry

OpenShift Container Platform 3.4 by Red Hat [provides](#) an internal, integrated registry that can be deployed in your OpenShift environment to manage container images *locally*. This registry is used to build Docker images from your source code, deploy the built applications and manage their lifecycle. The registry is deployed as part of the installation, but to deploy it manually:

```
# oadm registry \ --config=/etc/origin/master/admin.kubeconfig \ --
credentials=/etc/origin/master/openshift-registry.kubeconfig \ --
images='registry.access.redhat.com/openshift3/ose-
${component}:${version}'
```

#### Important

The default registry deployment uses an ephemeral volume that is destroyed if the pod exits. [Persistent storage](#) must be used for production environments.



**Important**

In production environments, it is recommended that you [secure the registry](#) so that it only serves traffic via TLS.

**3.3.10. Deploying a Router**

The OpenShift router is the entry point for all external traffic to OpenShift services. The router and its dependencies are created as part of the installation process.

Deploying a router requires a corresponding [service account](#). You can verify the existence of this service account through the following command:

```
# oc get serviceaccounts
```

The *router* service account is granted a [security context constraint](#) (SCC) that allows it to specify hosts, or otherwise create and configure the router. To verify the SCC assignment:

```
# oc describe scc hostnetwork

Name:          hostnetwork
Priority:       <none>
Access:
  Users:       system:serviceaccount:default:router
  Groups:      <none>
Settings:
  Allow Privileged:    false
  Default Add Capabilities:  <none>
  Required Drop Capabilities:  KILL,MKNOD,SYS_CHROOT,SETUID,SETGID
  Allowed Capabilities:    <none>
  Allowed Volume Types:
configMap,downwardAPI,emptyDir,persistentVolumeClaim,secret
  Allow Host Network:    true
  Allow Host Ports:     true
  Allow Host PID:       false
  Allow Host IPC:       false
  Read Only Root Filesystem:  false
  Run As User Strategy: MustRunAsRange
    UID:                <none>
    UID Range Min:      <none>
    UID Range Max:      <none>
  SELinux Context Strategy: MustRunAs
    User:               <none>
    Role:               <none>
    Type:               <none>
    Level:              <none>
  FSGroup Strategy: MustRunAs
    Ranges:             <none>
  Supplemental Groups Strategy: MustRunAs
    Ranges:             <none>
```

With the service account and its privileges in place, you can manually [deploy](#) a router:

```
# oadm router router --replicas=1 \ --service-account=router
```



### Important

Configuring a single replica for the router results in a single point of failure for incoming traffic and should be avoided in production environment.

Using the naming service to create a wildcard DNS entry that maps to the router requires that the router would be deployed to a known host. One way to achieve this in a trial environment is to have the number of replicas match the total number of hosts so that the router is deployed to every host. Regardless of the approach, to verify where the router is deployed, use the following command to describe the service:

```
# oc describe service router
Name:      router
Namespace: default
Labels:    router=router
Selector:  router=router
Type:      ClusterIP
IP:        172.30.234.170
Port:      80-tcp 80/TCP
Endpoints:
Port:      443-tcp 443/TCP
Endpoints:
Port:      1936-tcp 1936/TCP
Endpoints:
Session Affinity: None
No events.
```

Various load balancing and virtual IP solutions may be used to enabling mapping to multiple router replicas, but their configuration is outside the scope of this document.



### Note

The default router in OpenShift is the [HAProxy Template Router](#). In environments with an existing **F5 BIG-IP®** system, use the [F5 Router](#) plugin instead.

## 3.4. BUILD AND DEPLOY

### 3.4.1. Creating a New Project

Log in to a master host as the *root* user and create a new project, assigning the administrative rights of the project to the previously created OpenShift user:

```
# oadm new-project msa \ --display-name="OpenShift 3 MSA on EAP 7" \ --
description="This is a microservice architecture environment built on
JBoss EAP 7 on OpenShift v3" \ --admin=babak
Created project msa
```

### 3.4.2. OpenShift Login

Once the project has been created, all remaining steps can be performed as the regular OpenShift user. Log in to the master host machine or switch the user, and use the `oc` utility to authenticate against OpenShift:

```
# su - babak
$ oc login -u babak --certificate-authority=/etc/origin/master/ca.crt \
--server=https://ocp-master1.hostname.example.com:8443
Authentication required for https://ocp-
master1.hostname.example.com:8443 (openshift)
Username: babak
Password: PASSWORD
Login successful.

Using project "msa".
Welcome! See 'oc help' to get started.
```

The recently created `msa` project is the only project for this user, which is why it is automatically selected as the default working project of the user.

### 3.4.3. MySQL Images

This reference architecture includes two database services built on the supported [MySQL image](#).

To deploy the database services, use the `new-app` command and provide a number of required and optional environment variables along with the desired service name.

To deploy the product database service:

```
$ oc new-app -e MYSQL_USER=product -e MYSQL_PASSWORD=password -e
MYSQL_DATABASE=product -e MYSQL_ROOT_PASSWORD=passwd mysql --
name=product-db
```

To deploy the sales database service:

```
$ oc new-app -e MYSQL_USER=sales -e MYSQL_PASSWORD=password -e
MYSQL_DATABASE=sales -e MYSQL_ROOT_PASSWORD=passwd mysql --name=sales-
db
```

#### Warning

Database images created with this simple command are ephemeral and result in data loss in the case of a pod restart. Run the image with mounted volumes to enable persistent storage for the database. The data directory where MySQL stores database files is located at `/var/lib/mysql/data`.



#### Note

Refer to OpenShift Container Platform 3.4 by Red Hat documentation to configure [persistent storage](#).

**Warning**

Enabling clustering for database images is currently in [Technology Preview](#) and not intended for production use.

These commands create two OpenShift services, each running **MySQL** in its own container. In each case, a MySQL user is created with the value specified by the `MYSQL_USER` attribute and the associated password. The `MYSQL_DATABASE` attribute results in a database being created and set as the default user database. To monitor the provisioning of the services, use `oc status`. You can use `oc get events` for further information and troubleshooting.

```
$ oc status
In project OpenShift 3 MSA on EAP 7 (msa) on server https://ocp-
master1.hostname.example.com:8443

svc/product-db - 172.30.192.152:3306
  dc/product-db deploys openshift/mysql:5.6
  deployment #1 deployed 2 minutes ago - 1 pod
svc/sales-db - 172.30.216.251:3306
  dc/sales-db deploys openshift/mysql:5.6
  deployment #1 deployed 37 seconds ago - 1 pod
4 warnings identified, use 'oc status -v' to see details.
```

The warnings can be further inspected by using the `-v` flag. In this case, they simply refer to the fact that the image *has no liveness (or readiness) probe to verify pods are ready to accept traffic or ensure deployment is successful*.

Make sure the database services are successfully deployed before deploying other services that may depend on them. The service log clearly shows if the database has been successfully deployed. Use `tab` to complete the pod name, for example:

```
$ oc logs product-db-1-3drkp
---> 21:21:50      Processing MySQL configuration files ...
---> 21:21:50      Initializing database ...
---> 21:21:50      Running mysql_install_db ...
...
2016-06-04 21:21:58 1 [Note] /opt/rh/rh-
mysql56/root/usr/libexec/mysqld: ready for connections.
Version: '5.6.30'  socket: '/var/lib/mysql/mysql.sock'  port: 3306
MySQL Community Server (GPL)
```

**3.4.4. JBoss EAP 7 xPaaS Images**

This application relies on OpenShift S2I for Java EE applications and uses the [Image Streams for xPaaS Middleware Images](#). You can verify the presence of these image streams in the `openshift` project as the `root` user, but first switch from the default to the `openshift` project as the `root` user:

```
# oc project openshift
Now using project "openshift" on server "https://ocp-
master1.hostname.example.com:8443".
```

Query the configured image streams for the project:

```
# oc get imagestreams
NAME          DOCKER REPO
...
jboss-eap64-openshift      registry.access.redhat.com/jboss-eap-
6/eap64-openshift  1.1,latest,1.4 + 2 more...
jboss-eap70-openshift      registry.access.redhat.com/jboss-eap-
7/eap70-openshift  1.3,1.3-22,1.4-10 + 2 more...
...
```

The *jboss-eap70-openshift* image stream is used for EAP applications, but in the absence of an image called *jee* or a local image stream with a supports label value of *jee*, this image must be named explicitly. Also verify that the image stream points to a proper version of an OpenShift EAP 7 image.

The microservice application for this reference architecture is made available in a public git repository at <https://github.com/RHsyseng/MSA-EAP7-OSE.git>. This includes four distinct services, provided as subdirectories of this repository: *Billing*, *Product*, *Sales*, and *Presentation*.

Start by building and deploying the *Billing* service, which has no dependencies on either a database or another Java EE service. Switch back to the regular user with the associated *msa* project and run:

```
$ oc new-app jboss-eap70-openshift~https://github.com/RHsyseng/MSA-
EAP7-OSE.git --context-dir=Billing --name=billing-service
--> Found image ae4332e (2 weeks old) in image stream
"openshift/jboss-eap70-openshift" under tag "latest" for "jboss-eap70-
openshift"

      JBoss EAP 7.0
      -----
      Platform for building and running JavaEE applications on JBoss EAP
      7.0

      Tags: builder, javaee, eap, eap7

      * A source build using source code from
      https://github.com/RHsyseng/MSA-EAP7-OSE.git will be created
      * The resulting image will be pushed to image stream "billing-
      service:latest"
      * Use 'start-build' to trigger a new build
      * This image will be deployed in deployment config "billing-
      service"
      * Ports 8080/tcp, 8443/tcp, 8778/tcp will be load balanced by
      service "billing-service"
      * Other containers can access this service through the hostname
      "billing-service"

--> Creating resources ...
      imagestream "billing-service" created
      buildconfig "billing-service" created
      deploymentconfig "billing-service" created
      service "billing-service" created
--> Success
      Build scheduled, use 'oc logs -f bc/product-service' to track its
      progress.
      Run 'oc status' to view your app.
```

Once again, `oc status` can be used to monitor the progress of the operation. To monitor the build and deployment process more closely, find the running build and follow the build log:

```
$ oc get builds
NAME              TYPE      FROM      STATUS      STARTED
DURATION
billing-service-1 Source    Git        Running     1 seconds ago
1s

$ oc logs -f build/billing-service-1
```

Once this service has successfully deployed, use similar commands to deploy the *Product* and *Sales* services, bearing in mind that both have a database dependency and rely on previous MySQL services. Change any necessary default database parameters by passing them as environment variables:

```
$ oc new-app -e MYSQL_USER=product -e MYSQL_PASSWORD=password jboss-
eap70-openshift~https://github.com/RHsyseng/MSA-EAP7-0SE.git --context-
dir=Product --name=product-service
$ oc new-app -e MYSQL_USER=sales -e MYSQL_PASSWORD=password jboss-
eap70-openshift~https://github.com/RHsyseng/MSA-EAP7-0SE.git --context-
dir=Sales --name=sales-service
```

Finally, deploy the *Presentation* service, which exposes a web tier and an aggregator that uses the three previously deployed services to fulfill the business request:

```
$ oc new-app jboss-eap70-openshift~https://github.com/RHsyseng/MSA-
EAP7-0SE.git --context-dir=Presentation --name=presentation
```

Note that the *Maven* build file for this project specifies a *war* file name of *ROOT*, which results in this application being deployed to the root context of the server.

Once all four services have successfully deployed, the *presentation* service can be accessed through a browser to verify application functionality. First create a route to expose this service to clients outside the OpenShift environment:

```
$ oc expose service presentation --hostname=msa.example.com
route "presentation" exposed
```



#### Note

Please note that the `example.com` domain name successfully resolves to a public IP address. You can use any hostname, but it must resolve to the host running the router

The route tells the [deployed router](#) to load balance any requests with the given host name among the replicas of the *presentation* service. This host name should map to the IP address of the hosts where the router has been deployed, not necessarily where the service is hosted. For any servers in the same subnet or configured with the same name servers, the previously performed network configuration and [DNS wildcard entry](#) would resolve the host name correctly. For clients outside of this network and for testing purposes, simply modify your `/etc/hosts` file to map this host name to the IP address of the master host.

### 3.4.5. Replicas

Describe the deployment configuration of your services and verify how many instances have been configured and deployed. For example:

```
$ oc describe dc product-service
Name: product-service
Namespace: msa
Created: 10 minutes ago
Labels: app=product-service
Annotations: openshift.io/generated-by=OpenShiftNewApp
Latest Version: 1
...
Deployment #1 (latest):
Name: product-service-1
Created: 7 minutes ago
Status: Complete
Replicas: 1 current / 1 desired
Selector: app=product-service,deployment=product-service-1,deploymentconfig=product-service
Labels: app=product-service,openshift.io/deployment-config.name=product-service
Pods Status: 1 Running / 0 Waiting / 0 Succeeded / 0 Failed

Events:
  FirstSeen LastSeen Count From          SubobjectPath Type Reason      Message
  -----
  7m 7m 1 {deploymentconfig-controller } Normal DeploymentCreated
Created new replication controller "product-service-1" for version 1
```

Based on this default configuration, each service will not have any more than one replica, which means the OpenShift service will be backed by a single pod. In the event of the failure of a service container or OpenShift node, as long as there is an active master host, a new pod for the service will be deployed to a healthy node. However, it is often desirable to balance load between multiple pods of a service and also avoid a lengthy downtime while a failed pod is replaced.

Refer to the OpenShift [Developer Guide](#) for deployment configuration details and properly specifying the number of desired replicas.

To [manually scale](#) a service and verify this feature, use the `oc scale` command and provide the number of desired replicas for a given *deployment configuration*, for example:

```
$ oc scale dc product-service --replicas=3
deploymentconfig "product-service" scaled
```

There will now be 3 separate pods running this service. To verify, query all pods configured for *product-service*:

```
$ oc get pods -l app=product-service
NAME                                READY    STATUS    RESTARTS    AGE
product-service-1-8ag1z             1/1     Running   0           40s
product-service-1-mhyfu             1/1     Running   0           40s
product-service-1-mjis5             1/1     Running   0           5m
```

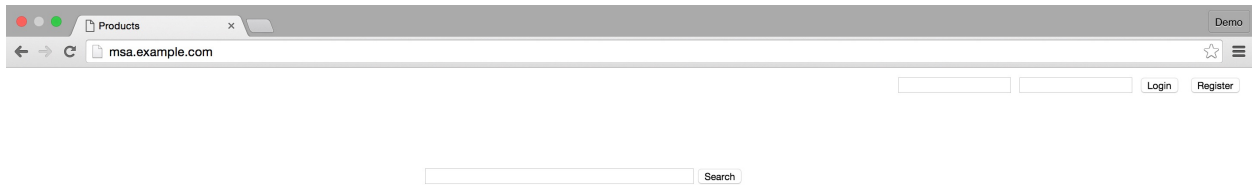
Traffic that is received through an exposed route or from an existing OpenShift service referencing another service by its service / host name (as is the case with the *presentation* service calling the other services) is handled by an internal proxy and balances the load between available replicas, while failing over when necessary.

## 3.5. RUNNING THE APPLICATION

### 3.5.1. Browser Access

To use the application, simply point your browser to the address exposed by the route. This address should ultimately resolve to the IP address of the OpenShift host where the router is deployed.

Figure 3.1. Application Homepage before initialization

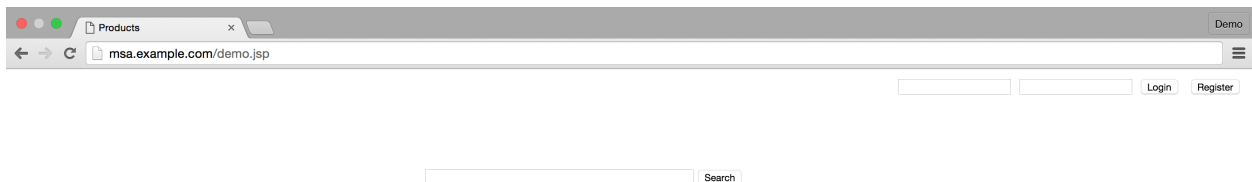


While **Hibernate** development settings are turned on and used to create the required database schema, the tables are still empty and content needs to be created for the application to function properly.

### 3.5.2. Sample Data

The application includes a demo page that when triggered, populates the database with sample data. To use this page and populate the sample data, point your browser to <http://msa.example.com/demo.jsp>:

Figure 3.2. Trigger sample data population

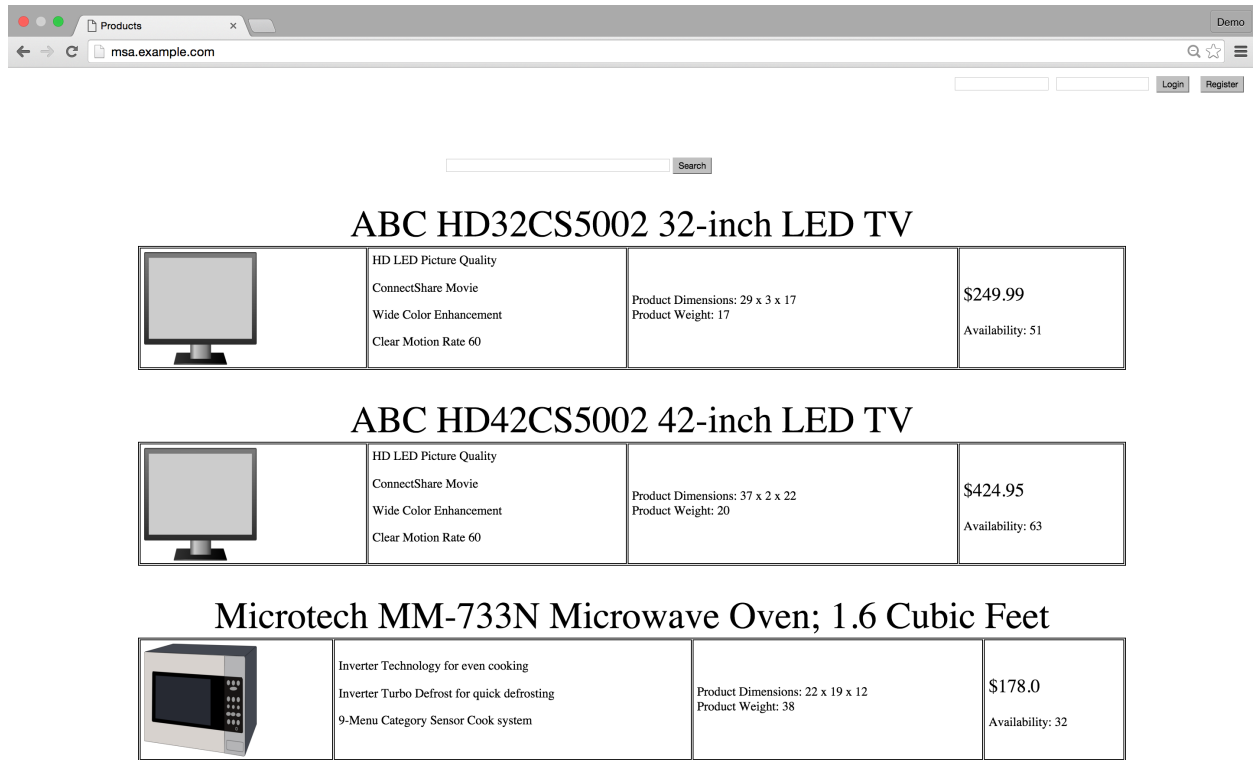


### 3.5.3. Featured Product Catalog


After populating the product database, the demo page redirects your browser to the route address, but this time you will see the featured products listed:

Figure 3.3. Application Homepage after initialization







The screenshot shows a web browser window with the address bar containing 'msa.example.com'. The page has a search bar and a 'Demo' label in the top right. Below the search bar, there are three product listings, each presented in a table format with four columns: product image, features, specifications, and price/availability.

ABC HD32CS5002 32-inch LED TV			
	HD LED Picture Quality ConnectShare Movie Wide Color Enhancement Clear Motion Rate 60	Product Dimensions: 29 x 3 x 17 Product Weight: 17	\$249.99 Availability: 51

ABC HD42CS5002 42-inch LED TV			
	HD LED Picture Quality ConnectShare Movie Wide Color Enhancement Clear Motion Rate 60	Product Dimensions: 37 x 2 x 22 Product Weight: 20	\$424.95 Availability: 63

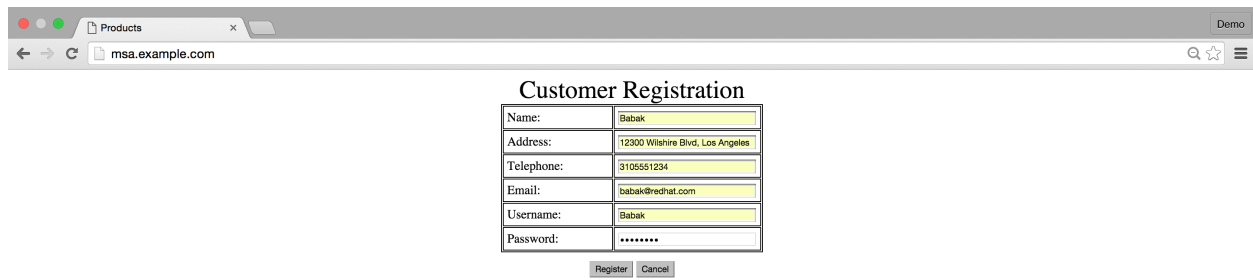
Microtech MM-733N Microwave Oven; 1.6 Cubic Feet			
	Inverter Technology for even cooking Inverter Turbo Defrost for quick defrosting 9-Menu Category Sensor Cook system	Product Dimensions: 22 x 19 x 12 Product Weight: 38	\$178.0 Availability: 32

### 3.5.4. User Registration

Anonymous users are constrained to browsing inventory, viewing featured products and searching the catalog. To use other features that result in calls to the *Sales* and *Billing* services, a valid customer must be logged in.

To register a customer and log in, click on the *Register* button in the top-right corner of the screen and fill out the registration form:

**Figure 3.4. Customer Registration Form**



Customer Registration

Name:	Babak
Address:	12300 Wilshire Blvd, Los Angeles
Telephone:	3105551234
Email:	babak@redhat.com
Username:	Babak
Password:	*****

Register Cancel

After registration, the purchase button allows customers to add items to their shopping cart, to subsequently visit the shopping cart and check out, and review their order history.

For details on application functionality and how each feature is implemented by the provided services and exposes through their REST API, refer to the previously published reference architecture on [Building microservices with JBoss EAP 7](#).

## CHAPTER 4. DESIGN AND DEVELOPMENT

### 4.1. OVERVIEW

For a discussion of microservices as a software architectural style, as well as the design and composition of the sample application, refer to the previously published reference architecture on [Building microservices with JBoss EAP 7](#).

OpenShift provides an ideal platform for deploying, hosting and managing microservices. By deploying each service as an individual **docker** container, OpenShift helps isolate each service and decouples its lifecycle and deployment from that of other services. OpenShift can configure the desired number of replicas for each service and provide intelligent scaling to respond to varying load.

This sample application uses the [Source-to-Image](#) (S2I) mechanism to build and assemble reproducible container images from the application source and on top of supported OpenShift images.

### 4.2. APPLICATION STRUCTURE

The source code for the sample application is checked in to a public GitHub [repository](#). The code is organized as four separate directories, each structured as a **Maven** project with a *pom* file at the root. An aggregation *POM* file is provided at the top level of the root directory to build all four project if needed, although this build file is neither required nor used by OpenShift.

### 4.3. CUSTOMIZING THE APPLICATION SERVER

The *Product* and *Sales* services each have a database dependency and use their respective MySQL database to store and retrieve data. The supported [xPaaS Middleware Images](#) bundle MySQL JDBC drivers, but the driver would have to be declared in the server configuration file and a datasource would need to be described in order for the application to access the database through a connection pool.

To make customizations to a server, provide an updated server configuration file. The replacement configuration file should be named [standalone-openshift.xml](#) and placed in a directory called *configuration* at the root of the project.



#### Note

Some configuration be performed by simply providing descriptive environment variables. For example, supplying the `DB_SERVICE_PREFIX_MAPPING` variable and value instructs the script to add MySQL and/or PostgreSQL datasources to the EAP instance. Refer to the documentation for OpenShift images for further details.

In order to make the required changes to the correct baseline, obtain the latest server configuration file. The supported image is located at `registry.access.redhat.com/jboss-eap-7/eap70-openshift`. To view the original copy of the file, you can run this docker container directly:

```
# docker run -it registry.access.redhat.com/jboss-eap-7/eap70-openshift
\ cat /opt/eap/standalone/configuration/standalone-openshift.xml
<?xml version="1.0" ?>
```

```
<server xmlns="urn:jboss:domain:4.0">
  <extensions>
    ...
```

Declare the datasource with parameterized variables for the database credentials. For example, to configure the product datasource for the product service:

```
<subsystem xmlns="urn:jboss:domain:datasources:1.2">
  <datasources>
    <datasource jndi-name="java:jboss/datasources/ProductDS" enabled="true"
      use-java-context="true" pool-name="ProductDS">
      <connection-url>
        jdbc:mysql://${env.DATABASE_SERVICE_HOST:product-db}
          :${env.DATABASE_SERVICE_PORT:3306}/${env.MYSQL_DATABASE:product}
      </connection-url>
      <driver>mysql</driver>
      <security>
        <user-name>${env.MYSQL_USER:product}</user-name>
        <password>${env.MYSQL_PASSWORD:password}</password>
      </security>
    </datasource>
```

The datasource simply refers to the database driver as *mysql*. Declare the driver class in the same section after the datasource:

```
...
</datasource>
<drivers>
  <driver name="mysql" module="com.mysql">
    <xa-datasource-class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-datasource-
class>
  </driver>
```

With the above configuration, environment variables are substituted to specify connection details and the database host name is resolved to the name of the OpenShift service hosting the database.

The default EAP welcome application is [disabled](#) in the Red Hat xPaaS EAP image. To deploy the *Presentation* application to the root context, rename the *warName* to *ROOT* in the **Maven pom** file:

```
<build>
  <finalName>${project.artifactId}</finalName>
  <plugins>
    <plugin>
      <artifactId>maven-war-plugin</artifactId>
      <version>${version.war.plugin}</version>
      <configuration>
        <warName>ROOT</warName>
        <!-- Java EE 7 doesn't require web.xml, Maven needs to catch up! -->
        <failOnMissingWebXml>>false</failOnMissingWebXml>
      </configuration>
    </plugin>
  </plugins>
</build>
```

## CHAPTER 5. CONCLUSION

Docker provides numerous advantages and is a natural fit for microservices. By leveraging Kubernetes to manage a cluster of Docker containers, OpenShift expands on the capabilities of container technologies and helps govern and simplify various stages of the software lifecycle.

This reference architecture demonstrates an easy path for Java EE developers and architects to migrate their applications to a microservice architecture that can be deployed and distributed in a secure environment and on-premise cloud, while benefiting from enterprise support and quality of service.

For a discussion of planning, deployment and operation of an Open Source Platform as a Service, refer to the reference architecture on [OpenShift Enterprise 3 Architecture Guide](#).

## APPENDIX A. REVISION HISTORY

Revision	Release Date	Author(s)
1.1	Feb 2017	Babak Mozaffari
1.0	Jun 2016	Babak Mozaffari

## APPENDIX B. CONTRIBUTORS

We would like to thank the following individuals for their time and patience as we collaborated on this process. This document would not have been possible without their many contributions.

Contributor	Title	Contribution
Kevin Conner	Manager, Software Engineering	OpenShift architecture, xPaaS images, Review
Daniel McPherson	Senior Principal Software Engineer	Technical Content Review
Scott Collier	Senior Principal Software Engineer	Technical Content Review
Christoph Görn	Principal Software Engineer	Technical Content Review
John Clingan	Senior Principal Product Manager	Subject Matter Review

## APPENDIX C. ANSIBLE HOSTS

[OSEv3:children]

masters

nodes

[OSEv3:vars]

ansible\_ssh\_user=root

openshift\_master\_default\_subdomain=example.com

deployment\_type=openshift-enterprise

openshift\_master\_identity\_providers=[{'name': 'htpasswd\_auth', 'login': 'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd'}]

[masters]

ocp-master1

[nodes]

ocp-master1

ocp-node1

ocp-node2



## APPENDIX D. REVISION HISTORY

---

Revision 1.1-0

Feb 2017

BM