Red Hat Reference Architecture Series

Building JBoss EAP 6 Microservices on OpenShift Enterprise 3

Babak Mozaffari

Version 1.0, December, 2015

Table of Contents

Comments and Feedback

In the spirit of open source, we invite anyone to provide feedback and comments on any of the reference architectures. 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. Please refer to the title within the email.

Staying In Touch

Join us on some of the popular social media sites where we keep our audience informed on new reference architectures as well as offer related information on things we find interesting.

Like us on Facebook:

https://www.facebook.com/rhrefarch

Follow us on Twitter:

https://twitter.com/RedHatRefArch

Plus us on Google+

https://plus.google.com/114152126783830728030/

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 6.4`, to take avantage of `OpenShift Enterprise 3.1 by Red Hat` and build an on-premise cloud environment to host the microservice architecture environment.

`OpenShift Enterprise 3.1 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.

2. Reference Architecture Environment

This reference architecture designs, develops and deploys a simple distributed microservice architecture on Red Hat JBoss Enterprise Application Platform 6.4 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.

3. Creating the Enviornment

3.1. Overview

This reference architecture can be deployed on either a production or a trial environment. In both cases, it is assumed that *ose-master1* refers to one (or the only) OpenShift master host and that the environment includes two OpenShift node hosts with the host names of *ose-node1* and *ose-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 envionment 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 *ose-master1* and IP addresses of *10.xxx.xxx.41*. The two nodes have hostnames of *ose-node1* and *ose-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-3.1-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:

```
OPTIONS='--selinux-enabled --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 ose-master1 ose-node1 ose-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 and continue. 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 add the next two hosts to use them as nodes:

```
Enter hostname or IP address: []: 10.xxx.xxx.41
Will this host be an OpenShift Master? [y/N]: y
Do you want to add additional hosts? [y/N]: y
Enter hostname or IP address: []: 10.xxx.xxx.42
Do you want to add additional hosts? [y/N]: y
Enter hostname or IP address: []: 10.xxx.xxx.43
Do you want to add additional hosts? [y/N]: n
```

Continue to the next step and select *OpenShift Enterprise 3.1* as the OpenShift variant to install:

```
Which variant would you like to install?


(1) OpenShift Enterprise 3.0
(2) OpenShift Enterprise 3.1
(3) Atomic Enterprise Platform 3.1
Choose a variant from above:  [1]: 2
```

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:

```
Do the above facts look correct? [y/N]: y
Ready to run installation process.

If changes are needed to the values recorded by the installer please update
/root/.config/openshift/installer.cfg.yml.

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.31          : ok=206  changed=58  unreachable=0   failed=0
10.xxx.xxx.32          : ok=57   changed=20  unreachable=0   failed=0
10.xxx.xxx.33          : ok=57   changed=20  unreachable=0   failed=0
localhost              : ok=11   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. Configure the default domain for your OpenShift applications 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"
```

Once again, restart the OpenShift service for this change 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, configure it as schedulable:

```
# oc edit node ose-master1.xxx.com
```

Edit the configuration and remove the line that makes it unschedulable:

```
spec:
  ...
  unschedulable: true
```

> ⚠️ 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 Enterprise 3.1 by Red Hat provides an internal, integrated Docker registry that can be deployed in your OpenShift environment to manage images "locally". This registry is used to build Docker images from your source code, deploy the built applications and manage their lifecycle. To deploy the integrated Docker registry:

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

> ❗ The default registry deployment uses an ephemeral volume that is destroyed if the pod exits. Persistent storage must be used for production environments.

> ❗ 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. For this trial environment, deploy a single instance of the router to a host that matches the wildcard DNS entry map.

Deploying a router requires a corresponding service account. The quick installation of OpenShift creates a service account called *router* for this purpose. Verify the existence of this service account through the following command:

```
# oc get serviceaccounts
```

In an environment where this service account is not available, or to create a separate account:

```
# echo '{"kind":"ServiceAccount","apiVersion":"v1","metadata":{"name":"router"}}' | oc create -f -
```

Configure privileged security context constraints (SCC) and add the service account in the list of users. Once again, the quick installation approach of OpenShift manages this aspect of the configuration as well. To add a different user or verify the existence of the *router* user:

```
# oc edit scc privileged
```

The router service account should appear in the list of users:

```
users:
- system:serviceaccount:default:registry
- system:serviceaccount:default:router
- system:serviceaccount:openshift-infra:build-controller
```

With the service account and its privileges in place, deploy the default router:

```
# oadm router router --replicas=1 \
    --credentials='/etc/origin/master/openshift-router.kubeconfig' \
    --service-account=router
```

> ❗ Configuring a single replica for the router results in a single point of failure for incomming 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 acheive 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:          172.30.175.196
Port:           80-tcp  80/TCP
Endpoints:      10.xxx.xxx.41:80
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.

> ℹ️ 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 6" \
      --description="This is a microservice architecture environment built on JBoss EAP 6
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://ose-master1.hostname.xxx.com:8443
Authentication required for https://ose-master1.hostname.xxx.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,MYSQL_PASSWORD=password,MYSQL_DATABASE=product,\
MYSQL_ROOT_PASSWORD=passwd mysql --name=product-db
```

To deploy the sales database service:

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

⚠ 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*.

ℹ Refer to OpenShift Enterprise 3.1 by Red Hat documentation to configure persistent storage.

⚠ 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 6 (msa) on server https://ose-
master1.hostname.xxx.com:8443

service/product-db - 172.30.43.206:3306
  dc/product-db deploys openshift/mysql:latest
    #1 deployed 4 minutes ago - 1 pod

service/sales-db - 172.30.222.4:3306
  dc/sales-db deploys openshift/mysql:latest
    #1 deployment pending 5 seconds ago

To see more, use 'oc describe <resource>/<name>'.
You can use 'oc get all' to see a list of other objects.
```

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.

3.4.4. JBoss EAP 6 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:

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

Query the configured image streams for the project:

```
# oc get imagestreams
NAME                             DOCKER REPO
jboss-amq-62                      registry.access.redhat.com/jboss-amq-6/amq62-
openshift
jboss-eap64-openshift             registry.access.redhat.com/jboss-eap-6/eap64-
openshift
...
```

The *jboss-eap64-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.

The microservice application for this reference architecture is made available in a public git repository at https://github.com/RHsyseng/OpenShift3-MSA.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-eap64-openshift~https://github.com/RHsyseng/OpenShift3-MSA.git
--context-dir=Billing --name=billing-service
imagestreams/billing-service
buildconfigs/billing-service
deploymentconfigs/billing-service
services/billing-service
Build "billing-service" created and started - you can run `oc status` to check the
progress.
Service "billing-service" created at 172.30.136.77 with port mappings 8080->8080, 8443-
>8443.
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      STATUS    POD
billing-service-1   Source    Running   billing-service-1-build

$ oc build-logs 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 depency 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,MYSQL_PASSWORD=password jboss-eap64-
openshift~https://github.com/RHsyseng/OpenShift3-MSA.git --context-dir=Product
--name=product-service
$ oc new-app -e MYSQL_USER=sales,MYSQL_PASSWORD=password jboss-eap64-
openshift~https://github.com/RHsyseng/OpenShift3-MSA.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-eap64-openshift~https://github.com/RHsyseng/OpenShift3-MSA.git
--context-dir=Presentation --name=presentation
```

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
NAME            HOST/PORT       PATH      SERVICE         LABELS              TLS TERMINATION
presentation    msa.example.com                presentation    app=presentation
```

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
Created:       4 days ago
Labels:        app=product-service
Latest Version: 1
Triggers:      Config, Image(product-service@latest, auto=true)
Strategy:      Rolling
Template:
  Selector: app=product-service,deploymentconfig=product-service
  Replicas: 1
  Containers:
  NAME            IMAGE                                                    ENV
...
    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
No events.
```

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.
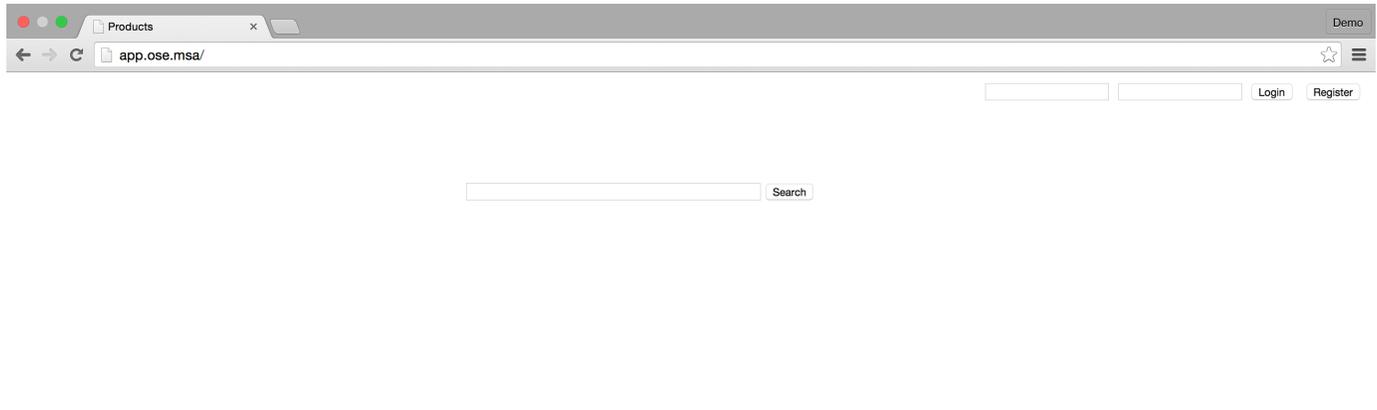

Figure 1. Application Homepage before initialization

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:
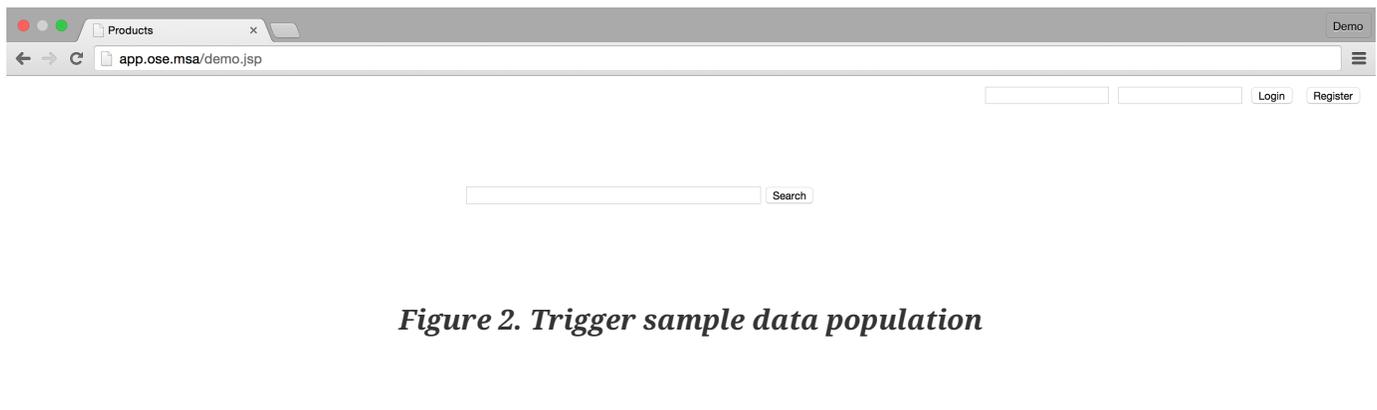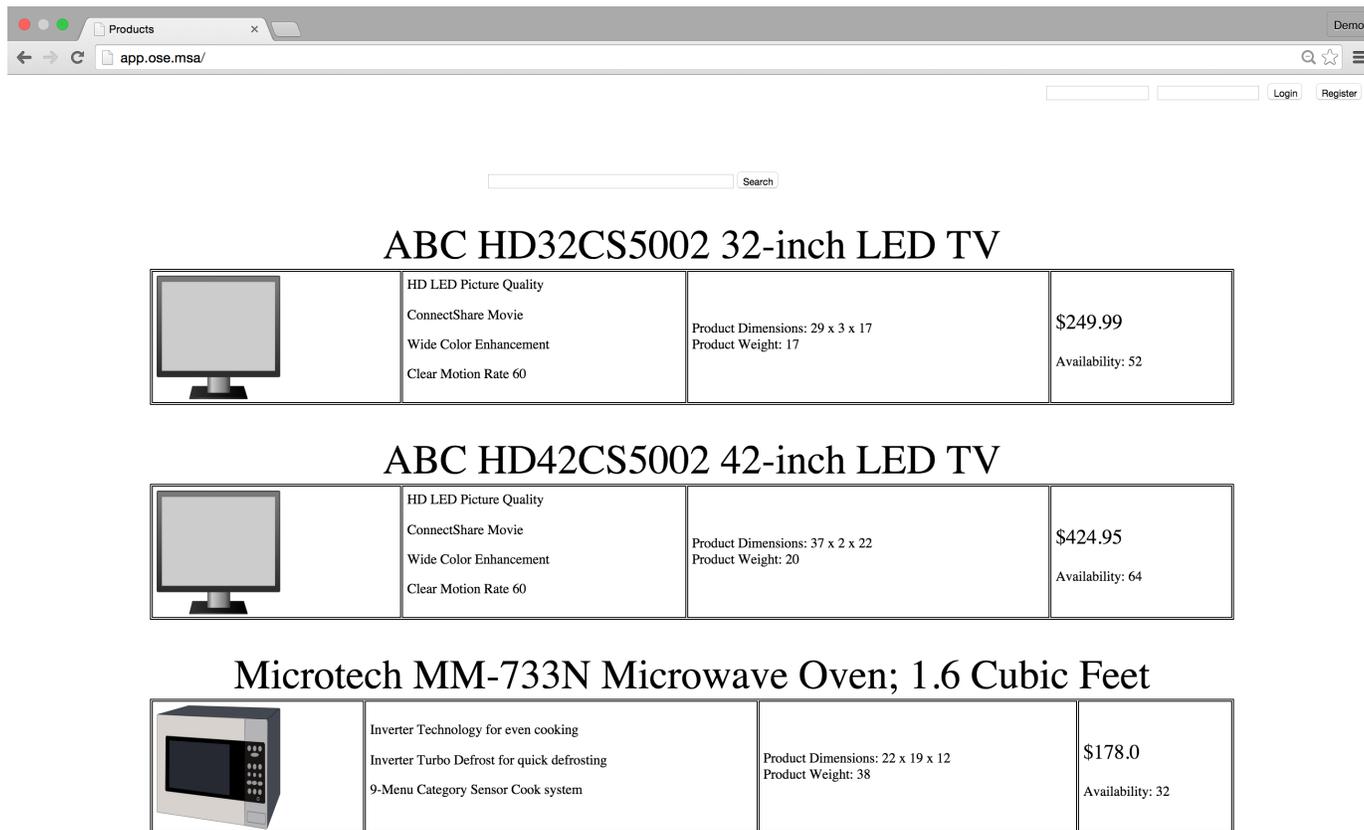

Figure 2. Trigger sample data population

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:

		Search	

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: 52

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: 64

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

Figure 3. Application Homepage after initialization

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 4. Customer Registration Form

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 6.

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 6.

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.

> ⓘ 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-6/eap-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-6/eap64-openshift \
        cat /opt/eap/standalone/configuration/standalone-openshift.xml
<?xml version='1.0' encoding='UTF-8'?>

<server xmlns="urn:jboss:domain:1.7">
    <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 only other required configuration is to deploy the application in the context root of the server. First disable the default welcome application:

```
<subsystem xmlns="urn:jboss:domain:web:2.2" ...>
  <connector name="http" protocol="HTTP/1.1" scheme="http" ...>
  <!-- No HTTPS configuration discovered -→
  <virtual-server name="default-host" enable-welcome-root="false">
```

Now that the default welcome application has been disabled, rename the application to *ROOT* so that it deploys in the root context by creating the proper *openshift* profile in the Maven *pom* file:

```
<profile>
<!-- When built in OpenShift the 'openshift' profile will be used when invoking mvn. -->
<!-- Use this profile for any OpenShift specific customization your app will need. -->
<!-- By default that is to put the resulting archive into the 'deployments' folder. -->
<!-- http://maven.apache.org/guides/mini/guide-building-for-different-environments.html -->
<id>openshift</id>
<build>
 <plugins>
  <plugin>
   <artifactId>maven-war-plugin</artifactId>
   <version>${version.war.plugin}</version>
   <configuration>
    <outputDirectory>deployments</outputDirectory>
    <warName>ROOT</warName>
   </configuration>
  </plugin>
 </plugins>
</build>
</profile>
</profiles>
</project>
```

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)
1.0	December 2015	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
etcd

[OSEv3:vars]
ansible_ssh_user=root
deployment_type=enterprise

openshift_master_cluster_password=openshift_msa_cluster
openshift_master_cluster_vip=10.xxx.xxx.40
openshift_master_cluster_public_vip=10.xxx.xxx.40
openshift_master_cluster_hostname=msa-vip.hostname.xxx.com
openshift_master_cluster_public_hostname=msa-vip.hostname.xxx.com

[masters]
ose-master1.hostname.xxx.com
ose-master2.hostname.xxx.com
ose-master3.hostname.xxx.com

[etcd]
ose-master1.hostname.xxx.com
ose-master2.hostname.xxx.com
ose-master3.hostname.xxx.com

[lb]
lb.example.com

[nodes]
ose-node[1:2].hostname.xxx.com openshift_node_labels="{'region': 'primary', 'zone': 'default'}"
```