Java Web Server Booster

IMPORTANT	This booster requires Java 8 JDK or greater and Maven 3.3.x or greater.

In this lab we will create a WAR deployment within openshift using JBoss Web Server. In this example we will call the deployment and related artifact jws.

Running the Booster on a Single-node OpenShift Cluster

If you have a single-node OpenShift cluster, such as Minishift or Red Hat Container Development Kit, installed and running, you can also deploy your booster there. A single-node OpenShift cluster provides you with access to a cloud environment that is similar to a production environment.

Connect your local machine to the openshift environmentT:

```
$ oc login -u developer -p developer

$ oc new-project MY_PROJECT_NAME
```

Create a build

Create a binary build configuration based on JBoss Web Server

```
$ oc new-build --binary=true --image-stream=jboss-webserver31-tomcat8-openshift:1.2
--name=jws
```

Build the application locally

Use Maven to build a WAR file locally

```
$ mvn clean install
```

Create a structure the binary build will understand

The build created in the previous step will expect the WAR to be in a file structure 'deployments'. The WAR will also be rename to use the root path structure.

```
$ cp target/jws-1.0.0-SNAPSHOT.war deploy/deployments/ROOT.war
```

Start the build

Manually start the build process with the structure created in the previous step with the build created earlier

```
$ oc start-build jws --from-dir=deploy
```

Create a new application

Create an application based on the build and create an externally accessable route.

```
$ oc new-app jws

$ oc expose svc/jws
```

Add the health checks

Add the health checks to the to the deployment configuration

- A liveness probe checks if the container in which it is configured is still running. If the liveness probe fails, the kubelet kills the container, which will be subjected to its restart policy.
- A readiness probe determines if a container is ready to service requests. If the readiness probe fails a container, the endpoints controller ensures the container has its IP address removed from the endpoints of all services.

```
$ oc set probe dc/jws --liveness --get-url=http://:8080/health --initial-delay
-seconds=10 --timeout-seconds=1

$ oc set probe dc/jws --readiness --get-url=http://:8080/health --initial-delay
-seconds=60 --timeout-seconds=1
```

Interacting with the Booster on a Single-node OpenShift Cluster

To interact with your booster while it's running on a Single-node OpenShift Cluster, you first need to obtain it's URL:

```
$ oc get route jws -o jsonpath={$.spec.host}

MY_APP_NAME-MY_PROJECT_NAME.LOCAL_OPENSHIFT_HOSTNAME
```

Interacting with the application

You can use the form at your application's url or you can use the `curl` command:

```
$ curl http://jws-MY_PROJECT_NAME.LOCAL_OPENSHIFT_HOSTNAME/api/greeting
{"content":"Hello World!"}

$ curl http://jws-MY_PROJECT_NAME.LOCAL_OPENSHIFT_HOSTNAME/api/greeting?name=Sarah
{"content":"Hello Sarah!"}
```

More Information

Related reading:

- https://blog.openshift.com/binary-input-sources-openshift-3-2/
- https://blog.openshift.com/deploying-war-file-openshift-online-3/
- https://blog.openshift.com/getting-started-with-jboss-web-server/
- https://blog.openshift.com/binary-input-sources-openshift-3-2/