

Openshift 4 Workshop JavaEE8 (Part 1 of 2)

1. Login to Openshift as your assigned user. For example, if your assigned username is user1, use the command below to login.

```
oc login <Openshift URL> -u user1 -p r3dh4t1!
```

2. Create a new project, substitute your username in the command below

```
oc new-project javaee8user<user#>
```

3. Clone the weather project onto your machine using the "git clone" command.

```
git clone https://github.com/tqvarnst/weather-app.git
```

4. Review the weather application

Background

To help illustrate the most common features of Java EE 8, you will build a web application called "Weather App". The application allows users to see weather information in the largest cities per country.

England

London

 **9°** Max: 11°, Min 7°

Feels like: 9°, Humidity: %, Wind: 3 m/s

Manchester

 **7°** Max: 8°, Min 5°

Feels like: 3°, Humidity: %, Wind: 10 m/s

Edinburgh

 **-7°** Max: -6°, Min -9°

Feels like: -13°, Humidity: %, Wind: 7 m/s

A user can switch country by clicking on one of the flags in the menu.

Weather App



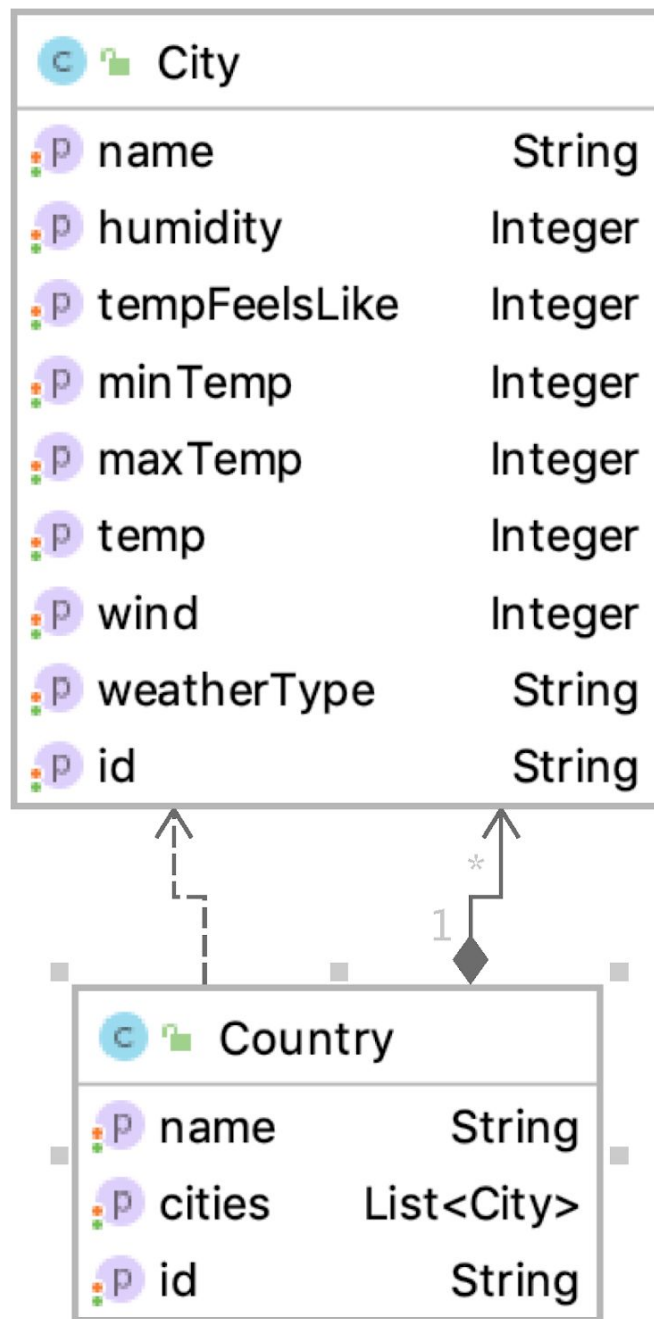
To save time there is a skeleton project with a prototype version that returns a hard-coded list of cities. What you will achieve are:

1. Deploy the prototype to OpenShift
2. Read the weather information from a database (internal at first)
3. Allow the user to choose a country
4. Connect the application to an external database with weather information

Review the current state of the prototype app

Front-end The web application is a single page `src/main/webapp/index.html`. In short, the front-end uses Bootstrap and jQuery. It uses REST to contact the back-end.

Domain model Two Java classes represent the domain model for our application, `src/main/java/com/redhat/example/weather/Country.java` and `src/main/java/com/redhat/example/weather/City.java`.



REST application using JAX-RS The prototype already implements a simple REST based service to allow the front-end to retrieve a country object with a list of cities to display. To enable JAX-RS it's required to provide a class that extends the `javax.ws.rs.core.Application` and configure the base URI (path) that is used to expose services. In

the prototype, this is done in the `src/main/java/com/redhat/example/weather/RestApplication.java`. In that class, we set the base URI to `/api`, which means that any call to the back-end using that URI is routed to the JAX-RS subsystem.

Weather REST Service

The current weather service `src/main/java/com/redhat/example/weather/WeatherService.java` already implements a simple REST service using JAX-RS and returns a List with one country that is currently hard-coded. We will change that later in this service.

User state

Finally, for convenience, the prototype also includes `src/main/java/com/redhat/example/weather/SelectedCountry.java`. This class is not in use at the moment, but we will make use of it later. The class is annotated with `@SessionScope` which tells the application server that the state of this class should be stored in the session scope. E.g., the state is maintained between different requests from the front-end to the back-end. However, if the user closes the browser or is inactive for too long, the state is removed. Using the session is a convenient way to store temporary state for a user that is ephemeral. For a persistent state, a database is typically used.

Build configuration The prototype also includes build configuration in the `pom.xml`. The build configuration is pretty simple but does include the `wildfly-maven-plugin`. Since we will be deploying to OpenShift, we will not use that plugin, but it's included here if you would like to use this project as a base for local development.

Lab Instructions

1. Deploy JBoss EAP image

To deploy our application we first have to create a container. JBoss EAP comes with a S2I image that packages all the binaries. S2I stands for Source-To-Image and is a convenient way for developers to provide either an artifact (E.g. WAR, JAR, etc) or the source code directly; and OpenShift will then build a container image for you, so there is no need to have a local container build system and as a developer you do not have to worry about patching and maintaining the base image.

```
oc new-build --image-stream=jboss-eap72-openshift:1.0 --binary=true --name=weather-app
```

2. Examine the Image Stream and Build Configs. Locate the Builds in the Web Console.

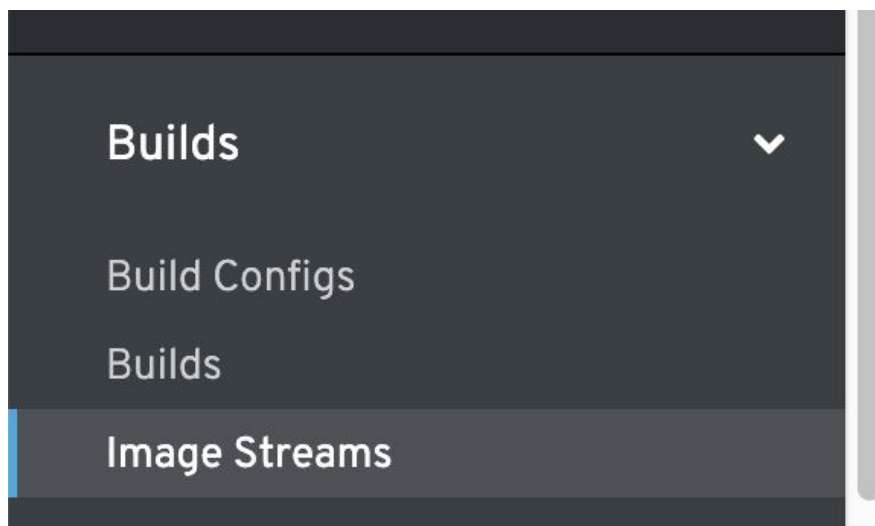






Image Streams

Create Image Stream				<input type="text" value="Filter Image Streams by name..."/>
NAME ↑	NAMESPACE	LABELS	CREATED	
 weather-app	 javaee8user23	build=weather-app	a few seconds ago	⋮

Build Configs

Create Build Config				<input type="text" value="Filter Build Configs by name..."/>
<div>0 Docker 0 JenkinsPipeline 1 Source 0 Custom Select All Filters</div>				1 Item
NAME ↑	NAMESPACE	LABELS	CREATED	
 weather-app	 javaee8user23	build=weather-app	less than a minute ago	⋮

3. Build the Application to deploy.

First move to the project directory:

```
cd weather-app
```

Now, build the application by executing the following maven command.

```
mvn clean package
```

4. We are now ready to upload our artifact (a WAR file) to the build "product-service" process that we created earlier.

```
oc start-build weather-app --from-file=target/ROOT.war --wait
```

This command will upload the artifact to the build config that we created before and start building a container. Please note that this might take a while the first time you execute it since openshift will have to download the EAP image. Next time it will go much faster.

5. When the build is finished, we are ready to create your application.

```
oc new-app weather-app
```

The command new-app will create a running instance based on the container that we built before .

6. Expose the route

```
oc expose svc/weather-app
```

7. Let's look at the details of the weather app route.

oc describe route weather-app






8. Check out the route

Project: javaee8user23 ▾ ⊕ Add ▾

Routes

Create Route Filter Routes by name...

1 Accepted 0 Rejected 0 Pending Select All Filters 1 Item

NAME ↑	NAMESPACE	LOCATION	SERVICE	STATUS	
 weather-app	 javaee8user23	http://weather-app-javaee8user23.apps.cluster-vegas-c8bc.vegas-c8bc.openshiftworkshop.com/	 weather-app	 Accepted	

9. Click on the location to try the route (next steps are in Part2)

Weather App   

England

London



8°C Max: 10°C, Min 6°C

Feels like: 8°C, Humidity: 30%, Wind: 2 m/s

Manchester



12°C Max: 14°C, Min 10°C

Feels like: 10°C, Humidity: 0%, Wind: 7 m/s

See Part 2 for adding JPA.