Table of Contents

Templates Lab

This lab includes the following sections:

- **Create and Upload Template**

  In this section, you create a template for a two-tier application (front end and database), upload it into the shared namespace (the `openshift` project), and ensure that users can deploy it from the web console.

- **Use Templates and Template Parameters**

  In this section, you create two separate template instances in two separate projects and establish a front-end-to-database-back-end connection by means of template parameters.

1. Create and Upload Template

1.1. Install Template

The example in this section involves a build of an application and a service with two pods: a front-end web tier and a back-end database tier. This application uses auto-generated parameters and other sleek features of OpenShift Enterprise. Note that this application contains predefined connectivity between the front-end and back-end components as part of its JSON, embedded in the source code. You add resources in a later lab.

This example is, in effect, a "quick start" — a predefined application that comes in a template and that you can immediately use or customize.

1. As `root` on the master host, download the template’s definition file:

```
[root@master00-GUID ~]# wget
```

```
http://www.opentlc.com/download/ose_implementation/3.1/resources/Template_Example.j
son
```

2. Create the template object in the shared `openshift` project. This is also referred to as *uploading* the template.

```
[root@master00-GUID ~]# oc create -f Template_Example.json -n openshift
template "a-quickstart-keyvalue-application" created
```

The `Template_Example.json` file defines a template. By "creating" it, you added it to the `openshift` project. To make the template available only for limited projects, add it to them, not to the `openshift` project.

1.2. Create Instance of Template

1. On your browser, connect to the OpenShift web console at `https://master00-GUID.oslab.opentlc.com:8443`:
   a. If prompted, accept the untrusted certificate.
   b. Log in as `andrew` with the password `r3dh4t1!`.
2. Click the blue **New Project** button in the top right corner.
3. Specify the project name, display name, and description:
   - **Name**: `instant-app`
   - **Display Name**: `instant app example project`
   - **Description**: `A demonstration of an instant app or template`.

   Alternatively, perform this step from the command line:

```
[root@master00-GUID ~]$ oadm new-project instant-app --display-
name="instant app example project" \
    --description='A demonstration of an instant-app/template'
\
    --node-selector='region=primary' --admin=andrew
```

4. From the `instant-app` project's **Overview** screen, click **Add to project**.

   This familiar screen now displays something new: an instant application, a special kind of template with the `instant-app` tag. The idea behind an instant application is that, when you create a template instance, you already have a fully functional application. In

> this example, your instant application is just a simple web page for key-value storage and retrieval.

5. Click **a-quickstart-keyvalue-application**.

   The template configuration screen is displayed. Here, you can specify certain options for instantiating the application components:

   a. Set the `ADMIN_PASSWORD` parameter to your favorite password.

   b. Add a label named `version` with the value `1`.

6. Click **Create** to instantiate the services, pods, replication controllers, etc.
   - The build starts immediately.

7. Wait for the build to finish. You can browse the build logs to follow the progress.

1.3. Use Application

After the build is complete, visit your application at `http://example-route-instant-app.cloudapps-GUID.oslab.opentlc.com/`.

> Be sure to use HTTP and *not* HTTPS. HTTPS does not work for this example because the form submission was coded with HTTP links.

2. Use Templates and Template Parameters

Quick starts are slick. But there are times when developers want to build the components manually. Here, you treat the quick-start example as two separate applications to be wired together.

2.1. Deploy Ephemeral Database Back End

1. Create a project for the database back end:

   a. Use your browser to connect to the OpenShift web console at `https://master00-GUID.oslab.opentlc.com:8443`.

   b. If prompted, accept the untrusted certificate.

   c. Log in as `marina` with the password `r3dh4t1!`.

   d. Click the blue **New Project** button in the top right corner.

   e. Specify the project name, display name, and description:

      - **Name**: `templates`
      - **Display Name**: `Templates Testing Project`
      - **Description**: `Project for testing templates`

Alternatively, perform this step from the command line:

```
[root@master00-GUID ~]$ oadm new-project templates --display-name="Templates
Testing Project" \
    --description='Project used to test templates' \
    --admin=marina
```

2. Deploy an ephemeral MySQL database:
   a. From the `templates` project’s **Overview** screen, click **Add to project**.
   b. Scroll down to **Databases** or type `mysql` in the search field.
   c. Select the `mysql-ephemeral` database template.
   d. Set the template parameters:
      - `DATABASE_SERVICE_NAME`: `database`
      - `MYSQL_USER`: `mysqluser`
      - `MYSQL_PASSWORD`: `redhat`
      - `MYSQL_DATABASE`: `mydb`

      Make sure you set these values correctly, otherwise the application would not connect to the database backend.

   e. Click **Create** and then click **Continue to overview**.

      Alternatively, create the template instance from the command line:

```
[marina@master00-GUID ~]$ oc new-app --template=mysql-ephemeral --
param=MYSQL_USER=mysqluser,MYSQL_PASSWORD=redhat,MYSQL_DATABASE=mydb,DATABASE_SE
RVICE_NAME=database
```

   f. As `marina`, switch to the "templates" project and examine the objects that were created as part of the `mysql-ephemeral` template.

```
[marina@master00-GUID ~]$ oc get projects
NAME                DISPLAY NAME               STATUS
custom-s2i-script   Custom S2I Build Script    Active
templates           Templates Testing Project  Active

[marina@master00-GUID ~]$ oc project templates
Now using project "templates" on server "https://master00-
3191.oslab.opentlc.com:8443".
```

```
[marina@master00-GUID ~]$ oc get dc
NAME       TRIGGERS                    LATEST
database   ConfigChange, ImageChange   1

[marina@master00-GUID ~]$ oc get service
NAME       CLUSTER_IP       EXTERNAL_IP   PORT(S)    SELECTOR        AGE
database   172.30.102.220   <none>        3306/TCP   name=database   1m
```

A deployment configuration is available for your instance. The service name is the same as that of your `DATABASE_SERVICE_NAME` parameter.

g. Verify that the values of the environment variables in the deployment configuration (`dc`) are correct:

```
[marina@master00-GUID ~]$ oc env dc database --list
# deploymentconfigs mysql, container mysql
MYSQL_USER=mysqluser
MYSQL_PASSWORD=redhat
MYSQL_DATABASE=mydb
```

2.2. Deploy Application’s Ruby Front End

1. As `marina`, create an application with the `https://github.com/openshift/ruby-hello-world` Git repository:

```
[marina@master00-GUID ~]$ oc new-app -i openshift/ruby
https://github.com/openshift/ruby-hello-world \
                          MYSQL_USER=mysqluser MYSQL_PASSWORD=redhat
MYSQL_DATABASE=mydb
```

2. Verify that your service is in place:

```
[marina@master00-GUID ~]$ oc get service
mysql              172.30.68.48    <none>       3306/TCP   name=mysql
4m
ruby-hello-world   172.30.78.240   <none>       8080/TCP   app=ruby-hello-
world,deploymentconfig=ruby-hello-world   8s
```

3. Create an external route to your front-end application.

   - If you do not specify a host name, the default subdomain route creates the route.

```
[marina@master00-GUID ~]$ oc expose service ruby-hello-world
route "ruby-hello-world" exposed
```

```
[marina@master00-GUID ~]$ oc get route
NAME                HOST/PORT
PATH      SERVICE            LABELS
ruby-hello-world   ruby-hello-world-templates.cloudapps-GUID.oslab.opentlc.com
ruby-hello-world   app=ruby-hello-world
```

4. Wait for the build to complete. Then test your environment:

```
[marina@master00-GUID ~]$ oc logs -f builds/ruby-hello-world-1
... Omitted Output ...
I1127 09:15:14.147821       1 cleanup.go:23] Removing temporary directory /tmp/s2i-
build846159358
I1127 09:15:14.148009       1 fs.go:99] Removing directory '/tmp/s2i-
build846159358'
I1127 09:15:14.173869       1 sti.go:213] Using provided push secret for pushing
172.30.42.118:5000/templates/ruby-hello-world:latest image
I1127 09:15:14.173963       1 sti.go:217] Pushing
172.30.42.118:5000/templates/ruby-hello-world:latest image ...
I1127 09:23:36.705738       1 sti.go:233] Successfully pushed
172.30.42.118:5000/templates/ruby-hello-world:latest
```

5. Wait for the pods to start and verify that your application is running and connecting to the database:

```
http://ruby-hello-world-templates.cloudapps-GUID.oslab.opentlc.com
```

Build Version: 2.2.1 : Last updated 2016-01-19 02:58:45 EST