Table of Contents

Create Your First S2I Build Lab

In this lab, you create a simple S2I build. As part of this process, you create a `BuildConfig` object and build an image using the S2I build process. You also create the pod, service, and route for your S2I build image.

Although this training currently covers OpenShift Enterprise 3.0, the labs and environment have already been updated to OpenShift Enterprise 3.1.

---

1. Prepare Your Environment

`If you have not done so in previous labs`, follow these steps before starting this lab.

1. As you did in the first lab, connect to **oselab** using your key and OPENTLC username.
2. From **oselab** connect to your **master** as the `root` user.
3. Run the "Create-Users" script, follow this example, **using your own name and key.**

```
[sborenst@desktop01 ~]$ ssh -i ~/.ssh/sborenstkey.pub shacharb-redhat.com@oselab-
*GUID*.oslab.opentlc.com

[bash-4.2$ ~] ssh root@192.168.0.100
root@192.168.0.100's password: ******** (r3dh4t1!)
[root@master00~]# bash /root/Create_Users_And_Projects.sh
```

Did you know? Reading the entire lab increases you chances of success!

2. Authenticate to OpenShift Enterprise and Choose Your Project

1. **`If you have not done so in a previous lab`**, authenticate to OpenShift Enterprise as user **`andrew`**:

```
[root@master00-GUID ~]# su - andrew
[andrew@master00-GUID ~]$ oc login -u andrew --insecure-skip-tls-verify --
server=https://master00-${GUID}.oslab.opentlc.com:8443
```

2. As Andrew, Change the context to the **`hello-s2i`** project:

```
[andrew@master00-GUID ~]$ oc project hello-s2i
Now using project "hello-s2i" on server "https://master00-
GUID.oslab.opentlc.com:8443".
```

3. Check your current context: (That means, what project you are working on)
   a. View the **`~/.kube/config`** file to review the information.
   b. Run the following command for a quick test:

```
[andrew@master00-GUID ~]$ grep current ~/.kube/config
current-context: hello-s2i/master00-GUID-oslab-opentlc-com:8443/andrew
```

3. Create Your S2I image

For this activity, you use a prebuilt and preconfigured code repository. This repository is an extremely simple application of the **`Hello World`** type.

1. Go to https://github.com/openshift/simple-openshift-sinatra-sti. You will use this application's source code.
2. Take a minute to review the repository.
3. To create the instructions and configuration for your image, use the **`oc new-app`** command as follows:

```
[andrew@master00-GUID ~]$ oc new-app https://github.com/openshift/simple-openshift-
sinatra-sti.git -o json > ~/simple-sinatra.json
```

4. Look at the JSON that you generated, if we didn't use the **`-o json`** flag, this application would have been created according to this definition file.

```
[andrew@master00~]$ cat ~/simple-sinatra.json
```

4. Start Your Build

1. To create the build components, use the **`oc create`** command on the **`BuildConfig`** file:

```
[andrew@master00-GUID ~]$  oc create -f ~/simple-sinatra.json
imagestream "simple-openshift-sinatra-sti" created
buildconfig "simple-openshift-sinatra-sti" created
deploymentconfig "simple-openshift-sinatra-sti" created
service "simple-openshift-sinatra" created
```

2. To see the the items created by the last command, run the following:
   a. Check your **`buildconfigs`**

```
[andrew@master00-GUID ~]$ oc get bc
NAME                          TYPE      FROM          LATEST
simple-openshift-sinatra-sti   Source    Git           1
```

   b. Check your **`services`**

```
[andrew@master00-GUID ~]$ oc get service
NAME                       CLUSTER_IP      EXTERNAL_IP   PORT(S)    SELECTOR
AGE
simple-openshift-sinatra   172.30.151.21   <none>        8080/TCP   app=simple-
openshift-sinatra-sti,deploymentconfig=simple-openshift-sinatra-sti   35s
```

   c. Check your **`deploymentconfigs`**

```
[andrew@master00-GUID ~]$ oc get dc
NAME                          TRIGGERS                    LATEST
simple-openshift-sinatra-sti   ConfigChange, ImageChange   0
```

   d. Check your **`replicationcontrollers`**

```
[andrew@master00-GUID ~]$ oc get rc # for replication controllers
CONTROLLER   CONTAINER(S)   IMAGE(S)
SELECTOR                                    REPLICAS   AGE
test1-1      test1          172.30.186.232:5000/hello-
s2i/test1@sha256:fc60b4c124859cdb1448caf32bfdf0cd3ea8705973dc4661f32fa7a53b2ed58
7   deployment=test1-1,deploymentconfig=test1   1          3h
```

3. Display your running pods

```
[andrew@master00-GUID ~]$ oc get pods
NAME                                     READY     STATUS      RESTARTS   AGE
simple-openshift-sinatra-sti-1-build     0/1       Completed   0          3m
simple-openshift-sinatra-sti-1-xfkww     1/1       Running     0          43s
```

Don't be alarmed if you see that the pod has failed to deploy, that happens before our image is created and will rectify itself once the image build process is complete.

4. To view the current build status and build logs, run the following:

```
[andrew@master00-GUID ~]$  oc get builds
NAME                              TYPE      FROM         STATUS     STARTED
DURATION
simple-openshift-sinatra-sti-1    Source    Git          Complete   3 minutes ago
2m19s
```

5. View the Build log

```
[andrew@master00-GUID ~]$ oc logs builds/simple-openshift-sinatra-sti-1
 ... Skipped output ...
I1209 22:38:35.736651       1 fs.go:99] Removing directory '/tmp/s2i-
build514028186'
I1209 22:38:35.760747       1 sti.go:213] Using provided push secret for pushing
172.30.186.232:5000/hello-s2i/simple-openshift-sinatra-sti:latest image
I1209 22:38:35.760832       1 sti.go:217] Pushing 172.30.186.232:5000/hello-
s2i/simple-openshift-sinatra-sti:latest image ...
I1209 22:40:06.762013       1 sti.go:233] Successfully pushed
172.30.186.232:5000/hello-s2i/simple-openshift-sinatra-sti:latest
```

6. Make sure to check the progress on the web console.

5. Create Your First Image (Sinatra)

1. After your build is complete, to verify your pod and service, run the following:

```
[andrew@master00-GUID ~]$ ServiceIPandPORT=`oc get services | grep sin | awk
'{print $2":"$4}' | awk -F'/' '{print $1}'`
[andrew@master00-GUID ~]$ curl $ServiceIPandPORT
Hello, Sinatra!
```

2. Your last step is to add a route to make the application publicly accessible. To do this, run the following:

```
[andrew@master00-GUID ~]$ oc expose service simple-openshift-sinatra --
hostname=mysinatra.cloudapps-$GUID.oslab.opentlc.com
route "simple-openshift-sinatra" exposed
[andrew@master00-GUID ~]$ oc get routes
NAME                      HOST/PORT                                                PATH
SERVICE                   LABELS                              INSECURE POLICY   TLS
TERMINATION
simple-openshift-sinatra   mysinatra.cloudapps-GUID.oslab.opentlc.com
simple-openshift-sinatra   app=simple-openshift-sinatra-sti
```

3. Test that your **route** is working:

```
[andrew@master00-GUID ~]$ curl http://mysinatra.cloudapps-$GUID.oslab.opentlc.com
Hello, Sinatra!
```

6. Challenge lab (Optional)

This lab requires you to figure stuff out on your own and might present a challenge for new users. It might require looking in the manual.

1. Using what you learned in this chapter, create an application using the Web Console and the command line.
2. Create a project called "nodejs"
3. You can use nodejs for your example, or any other builder based application (Ruby, Perl PHP, up to you)
   a. The Application repository is https://github.com/openshift/nodejs-ex
   b. Use the "nodejs:0.10" image
4. Create a route and expose the service to the world under the name : http://nodejs.cloudapps-GUID.oslab.opentlc.com/
   a. Try to explore the **oc edit route [routename]** command
5. Make sure Application has 4 replicas, using the `oc scale --replicas=4 dc [DCname]` command or the web console.

Last updated 2015-12-10 17:51:56 EST