How to Demonstrate OpenShift Enterprise 3.0

Table of Contents

1. Set up your Demonstration Environment

1.1. Provision Your Demonstration Environment

To perform this demonstration, you must provision the demonstration environment. Provisioning ensures that you have access to an environment containing all the components you need to perform this demonstration.

1. Go to https://rhpds.redhat.com and use your credentials to log in to the OPENTLC labs provisioning system, which is built on top of Red Hat CloudForms to provide a self-service portal.

If you forgot your username or password, go to https://www.opentlc.com/pwm to reset your password or obtain a username reminder.

2. After you log in, navigate to **Services** → **Catalogs** → **All Services** → **Cloud Infrastructure Demos**.
3. On the left side of the screen, locate the **OpenShift 3.1 Demo** catalog item and select it.
4. Read the demo's description text for updates other important information

The root user credentials for the demonstration environment will be listed in the Demonstration description box, make sure you note them down as they will not be mentioned anywhere in this document.

1. Click the **Order** button that appears on the right.
2. On the next screen, click **Submit** (on the lower right side) to order your environment.

   In a few minutes you will receive an email containing the details on how to connect to the environment.
3. Wait about 30 minutes to allow the environment to build. Your environment includes the following entities:
   - One host for the Master
   - One Infrastructure node (Not required) <1>
   - Two hosts for nodes
   - One administration host for connection into the environment and miscellaneous tasks
   - IPA identity management

1 We use this server to demonstrate a highly scalable architecture

Because the environment is all cloud-based, you can access it over the WAN from anywhere. You should not, however, expect performance to match a bare-metal environment.

2. Set Up SSH

2.1. Share your public key with OPENTLC for authentication

To access any of your lab systems via SSH, you must use your personal OPENTLC SSO user name and public SSH key. *(These are the credentials you used to connect to *labs.opentlc.com* and *rhpds.redhat.com*`)

`If you have not already done so`, you must provide a public SSH key to the OPENTLC authentication system.

1. Go to https://www.opentlc.com/update and log in. (Using your Opentlc credentials)

For more information on generating an SSH key, see the following: https://www.opentlc.com/ssh.html

2. Paste your public key in that location.

2.2. Test your ability to connect to your servers

The **`oselab`** host, (aka: the administration host) is just there to serve as an access point into the environment and is not part of the OpenShift environment. (We do actually use th oselab as our dedicated DNS server, but that is not an OpenShift component.)

An email with all the host Public IPs and DNS host names was sent to you as soon as the environment started provisioning, if you did not recieve this email please check your spam folder and verify in RHPDS that your environment has been deployed.

1. Connect to your administration host and make sure you can access each of your provisioned hosts:

```
ssh -i ~/.ssh/yourprivatekey.key opentlcuser@oselab-GUID.oslab.opentlc.com
```

The text **GUID** is a 4-character unique identifier generated for your lab environment. Your GUID is at the top of the lab provisioning email.

+

In these labs, you will see "$guid" and $GUID" used to signify your unique identifier, We create a environment variable so you don't have to type it each time.

Example

```
Laptop$ ssh -i ~/.ssh/mykey.key shacharb-redhat.com@oselab-c3po.oslab.opentlc.com
(root password is: r3dh4t1!)
```

1. Once connected to **oselab**, test the connection to each server:
   a. use the same **root** password as before : **r3dh4t1!**, when prompted

```
-bash-4.2$ ssh root@192.168.0.100
-bash-4.2$ ssh root@192.168.0.101
-bash-4.2$ ssh root@192.168.0.200
-bash-4.2$ ssh root@192.168.0.201
```

3. Deploy From Prebuilt Containers (`hello-openshift` version)

In this scenario, David wants to deploy an application based on a container created by someone.

3.1. Login and Authenticate

1. **Action** - Login to your server and switch to the user `david`.

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

#From you administration host (oselab)
[bash-4.2$ ~] ssh root@192.168.0.100
root@192.168.0.100's password: ******** (r3dh4t1!)

# From the Master server (Master00)
[root@master00-GUID ~]# su - david
[david@master00-GUID ~]$ #you're in
```

2. **Action** - As user `david` log in to OpenShift (Password: **r3dh4t1!**)and select the `hello-openshift-demo` project.
   - **Caution** - If you have *already logged in* to OpenShift Enterprise, *do not* run the `oc login` command again. Because you are already logged in, this will result in an error on screen.
   - **Explain** that you are currently logging in to the master as part of this demonstration, but consider that every command that `david` issues is a command that a developer would do on his or her laptop or workstation, or from wherever he or she is working.
   - **Explain** what projects are and how different projects could have different user permissions and quotas attached to them.

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

```
(Password is: r3dh4t1!)
```

You should expect output similar to this one:

```
Login successful.

Using project "hello-openshift-demo".

You have access to the following projects and can switch between them with 'oc project
<projectname>':

  * hello-openshift-demo (current)
  * instantapps-demo
  * sourcetoimage-demo
  * weightwatcher-demo

Welcome to OpenShift! See 'oc help' to get started.
```

1. If you are not already using the **hello-openshift-demo** project, switch to it:

```
[david@master00]$ oc project hello-openshift-demo
Now using project "hello-openshift-demo" on server "https://master00-
8675.oslab.opentlc.com:8443".
```

3.2. Define the Pod

Here you define the pod, but not the service, replication controllers, or routes.

1. **Show** the `hello-openshift-podonly.json` file.
   - **Explain** that this is a very simple pod definition example and it does not cover services, routes, and other resources.
   - **Point out** the following:
     - `name` - This is the name of the pod.
     - `image` - This is the container image that this pod is running. This can be a local registry or an external one (like `docker.io`).
     - `ports` - These are the ports that the Docker container exposes. Your code and application need to listen on those ports as well.
     - `labels` - This is perhaps the most important component. Labels are "tags" that you apply, so that you can refer to a group of resources (pods, services, and so on).

```
[david@master00]$ cat hello-openshift-pod.json
{
  "kind": "Pod",
  "apiVersion": "v1",
  "metadata": {
    "name": "hello-openshift",
    "creationTimestamp": null,
    "labels": {
      "name": "hello-openshift"
    }
  },
  "spec": {
    "containers": [
      {
        "name": "hello-openshift",
        "image": "openshift/hello-openshift:latest",
        "ports": [
          {
            "containerPort": 8080,
            "protocol": "TCP"
          }
        ],
        "resources": {
          "limits": {
            "cpu": "10m",
            "memory": "16Mi"
          }
        },
        "terminationMessagePath": "/dev/termination-log",
        "imagePullPolicy": "IfNotPresent",
        "capabilities": {},
        "securityContext": {
          "capabilities": {},
          "privileged": false
        },
        "nodeSelector": {
          "region": "primary"
        }
      }
    ],
    "restartPolicy": "Always",
    "dnsPolicy": "ClusterFirst",
    "serviceAccount": ""
  },
  "status": {}
}
```

3.3. Create the Pod

Here you create the pod, but not the service, replication controllers, or routes.

1. **Action** - Use the `oc create` command to create the pod from the `hello-openshift-pod.json` file.
    - **Explain** that during this process, OpenShift Enterprise reviews and processes the file. You could easily have added other pods or resources into the file, and OpenShift Enterprise would have processed them together.
2. **Optional** - You can use the `docker ps` command to show the running container and the `docker logs -f $DOCKERPID&` command to show the internal Docker log for the container being built.

```
[david@master00~]$ oc create -f hello-openshift-pod.json
```

3. Expect the following output:

```
pods/hello-openshift-pod
```

1. **Action** - Run **oc get pods** to show the pod status and that you can access the pod locally.
    - **Explain** the output to the audience.
    - **Point out** the following:
        - `NAME` - The pod name.
        - `REASON` - The pod's status or last error message
        - `AGE` - The pod age since it was first launched.

```
[david@master00~]$ oc get pods
NAME              READY     REASON    RESTARTS   AGE
hello-openshift   1/1       Running   0          20s
```

2. **Action** get the pod's information using **oc describe**

```
[david@master00~]$ oc describe pod hello-openshift
```

3. You will see output similar to this one:
    - **Point out** the following:
        - `Image` - This is the Docker image that is used to deploy this pod.
        - `Host` - This is the host that our pods resides/runs in.
        - `IP` - This is the internal IP address accessible on the local network.

```
Name:                           hello-openshift
Image(s):                       openshift/hello-openshift:v1.0.6
Host:                           node00-f4fc.oslab.opentlc.com/192.168.0.200
Labels:                         name=hello-openshift
Status:                         Running
IP:                             10.1.0.9
Replication Controllers:        <none>
Containers:
  hello-openshift:
    Image:              openshift/hello-openshift:v1.0.6
    State:              Running
      Started:          Fri, 03 Jul 2015 02:11:24 -0400
    Ready:              True
    Restart Count:      0
Conditions:
  Type          Status
  Ready         True
Events:
```

4. **Action** - Test your pod
5. **CATION** You need to use the `IP`of `your` own pod from the last output

```
[david@master00~]$ curl http://10.1.0.9:8080
```

6. Expect the following output:

```
Hello OpenShift!
```

The container will be up in a few seconds, but the application in the container might take a few minutes to load.

3.4. Deploy a Complete Application With Service and Route to Match

1. **Action** - Review the following complete application example `hello-openshift-complete.json` file.
   - **Ask** how complicated or simple it would be to define a full application stack in the audience's current environment.
   - **Point out** the following:

You should understand every line in the file in case the audience asks questions. However, do not explain each line.

- `"kind": "Service"` - Explain what a service is and that here you are creating a "front end" for this pod or group of pods.
- `"kind": "Route"` - Explain that a route resource allows external access using a HAProxy container. You could have many routes to the same application.
- `"replicas": 1` - Explain that you currently set this pod to have a single replica. If you want to deploy many replicas or scale out at any time, you can simply change this value.
- `"labels":` - The label you enter here is applied to each resource item you create for this application. This simplifies management.
- `"triggers":` - This is an optional component. Explain that you can set triggers to redeploy containers under certain conditions—for example, if a newer image is available.

```
[david@master00~]$ cat hello-openshift-complete.json
{
  "kind": "List",
  "apiVersion": "v1",
  "metadata": {
    "name": "hello-openshift-complete-example"
  },
  "items": [
    {
      "kind": "Service",
      "apiVersion": "v1",
      "metadata": {
        "name": "hello-openshift-service"
      },
      "spec": {
        "selector": {
          "name": "hello-openshift"
        },
        "ports": [
          {
            "protocol": "TCP",
            "port": 27017,
            "targetPort": 8080
          }
        ]
      }
    },
    {
      "kind": "Route",
      "apiVersion": "v1",
      "metadata": {
        "name": "hello-openshift-route"
      },
      "spec": {
```

```
      "host": "hello-openshift.cloudapps-GUID.oslab.opentlc.com",
      "to": {
        "name": "hello-openshift-service"
      },
      "tls": {
        "termination": "edge"
      }
    }
  },
  {
    "kind": "DeploymentConfig",
    "apiVersion": "v1",
    "metadata": {
      "name": "hello-openshift"
    },
    "spec": {
      "strategy": {
        "type": "Recreate",
        "resources": {}
      },
      "triggers": [
        {
          "type": "ConfigChange"
        }
      ],
      "replicas": 1,
      "selector": {
        "name": "hello-openshift"
      },
      "template": {
        "metadata": {
          "creationTimestamp": null,
          "labels": {
            "name": "hello-openshift"
          }
        },
        "spec": {
          "containers": [
            {
              "name": "hello-openshift",
              "image": "openshift/hello-openshift:v0.4.3",
              "ports": [
                {
                  "name": "hello-openshift-tcp-8080",
                  "containerPort": 8080,
                  "protocol": "TCP"
                }
              ],
              "resources": {
                "limits": {
                  "cpu": "10m",
                  "memory": "16Mi"
                }
              },
```

```
                "livenessProbe": {
                  "tcpSocket": {
                    "port": 8080
                  },
                  "timeoutSeconds": 1,
                  "initialDelaySeconds": 10
                }
              }
            ],
            "restartPolicy": "Always",
            "dnsPolicy": "ClusterFirst",
            "serviceAccount": "",
            "nodeSelector": {
              "region": "primary"
            }
          }
        }
      }
    }
  ]
}
```

2. **Action** - Create your application using the **oc create** command.
   - **Explain** that by passing the `.json` file to OpenShift Enterprise, you are requesting all the resource items in the file to be created.
   - **Optional** - Show your audience the web console.
     a. **Ask** the audience for any questions. This is a good time to find out, for example, if the process is clear, or if they see themselves using this tool.

```
[david@master00~]$ oc create -f hello-openshift-complete.json
[david@master00~]$ oc expose service hello-openshift-service --
hostname=hello-openshift.cloudapps-$GUID.oslab.opentlc.com
```

3.5. Show the Application Using Your Browser

1. **Action** - Browse to: http://hello-openshift.cloudapps-$GUID.oslab.opentlc.com
   - **Explain** that you have now deployed a container that is externally accessible. You could scale the application at any time, and the route and service ensure that traffic always routes to the application.
   - **Big finish** - Offer some closing words on this demo.

3.6. Show the Application Using the Command Line

1. **Action** - You can run the following commands to show the application resources from the command line
   a. **Show** The audience that you now have a Pod, a ReplicationController (RC) and a

DeploymentConfig (DC)

b. **Explain** the roles of the RC and DC resources.

c. **Point Out:**

i. The differences between the single sad pod from the beginning of this demonstration and the pod that was generated by our DC and RC

ii. That when we run **oc get rc** and see our RCs we can see how many replicas we are running.

```
[david@master00-70ac ~]$ oc get pods
POD                       IP         CONTAINER(S)                  IMAGE(S)
HOST                                            LABELS
STATUS    CREATED
hello-openshift-1-ok0aa   10.1.0.6   hello-openshift               openshift/hello-
openshift   master00-70ac.oslab.opentlc.com/192.168.0.100   deployment=hello-openshift-
1,deploymentconfig=hello-openshift,name=hello-openshift   Running   About a minute
hello-openshift-pod       10.1.0.5   hello-openshift-singlesadpod   openshift/hello-
openshift   master00-70ac.oslab.opentlc.com/192.168.0.100   name=hello-openshift-
singlesadpod                                                    Running   2 minutes
[david@master00-70ac ~]$ oc get dc
NAME              TRIGGERS      LATEST VERSION
hello-openshift   ImageChange   1
[david@master00-70ac ~]$ oc get rc
CONTROLLER          CONTAINER(S)      IMAGE(S)                    SELECTOR
REPLICAS
hello-openshift-1   hello-openshift   openshift/hello-openshift   deployment=hello-
openshift-1,deploymentconfig=hello-openshift,name=hello-openshift   1

[david@master00-70ac ~]$ curl http://hello-openshift.cloudapps-$GUID.oslab.opentlc.com
Hello OpenShift!
```

3.7. Optional - Show how to increase the replicas of the deployed pods.

1. **Action** - Run the following command.

a. *Explain the role of the DC (DeploymentConfig)

b. **Point Out:**

i. Triggers - What makes the DC redeploy the pods

ii. Replicas - How many replicas are required of this pod - This is where we will make a permanent change to an environment

```
[david@master00-70ac ~]$ oc describe dc hello-openshift
Name:           hello-openshift
Created:        3 minutes ago
Labels:         <none>
Latest Version: 1
```

```
Triggers:       Config
Strategy:       Recreate
Template:
        Selector:       name=hello-openshift
        Replicas:       1
        Containers:
                NAME            IMAGE                                   ENV
                hello-openshift openshift/hello-openshift:v1.0.6
Deployment #1 (latest):
        Name:           hello-openshift-1
        Created:        3 minutes ago
        Status:         Complete
        Replicas:       1 current / 1 desired
        Selector:       deployment=hello-openshift-1,deploymentconfig=hello-
openshift,name=hello-openshift
        Labels:         openshift.io/deployment-config.name=hello-openshift
        Pods Status:    1 Running / 0 Waiting / 0 Succeeded / 0 Failed
No events.
```

1. **Action** - Run the following command.
   a. **Action** - Use the **oc scale** command to increase the **replica** count from 1 to 10.
   b. **Explain** By changing the DeploymentConfig we are raising the "desired state" of the replica count from 1 to 10, this will result in immediate change.
   c. **Note** How efficient OpenShift 3 is, the output of the **oc scale** command is the simple and concise **"scaled"**

```
[david@master00-70ac ~]$ oc scale dc hello-openshift --replicas=10
scaled
```

1. **Action** - Run the following commands to show the the new replicas that were created and that the service has updated with the new pods.
   a. **Explain** that new pods are created by the RC the next time it syncs with the desired state defined in the DC …**Explain** The Service will be updated with the new pod names as they appear.

```
[david@master00-70ac ~]$ oc get pods
NAME                      READY     REASON    RESTARTS   AGE
hello-openshift-1-0dxco   1/1       Running   0          4m
hello-openshift-1-0zyoj   1/1       Running   0          4m
hello-openshift-1-17j8o   1/1       Running   0          8m
hello-openshift-1-8rfly   1/1       Running   0          8m
hello-openshift-1-9ve89   1/1       Running   0          4m
hello-openshift-1-bcw8z   1/1       Running   0          8m
hello-openshift-1-dtfos   1/1       Running   0          14m
hello-openshift-1-mtv6s   1/1       Running   0          8m
hello-openshift-1-r1lbp   1/1       Running   0          4m
```

```
hello-openshift-1-y8ffs   1/1       Running   0          4m
```

1. **Action** Use the **oc describe** command to display the service.
   a. **Explain** That the service is automatically listing all the new pods that have the label: **"name=hello-openshift"**

```
[david@master00-70ac ~]$ oc describe service hello-openshift-service
Name:                   hello-openshift-service
Labels:                 <none>
Selector:               name=hello-openshift
Type:                   ClusterIP
IP:                     172.30.47.5
Port:                   <unnamed>       27017/TCP
Endpoints:              10.1.0.14:8080,10.1.0.15:8080,10.1.0.16:8080 + 7 more...
Session Affinity:       None
No events.
```

2. **Optional Action** - If you have a room full of syntax geeks you can show this example of getting the pods to display with their nodes and ips
   a. **Explain** that we can use edit the `template` of our output on the fly and call on different attributes in the object

```
[david@master00-f4fc ~]$ oc get pod -t '{{range .items}}{{.metadata.name}}|
{{.status.phase}} | {{.spec.host}} | {{.status.podIP}} {{"\n"}}{{end}}'

hello-openshift-1-0dxco| Running | node01-f4fc.oslab.opentlc.com | 10.1.1.10
hello-openshift-1-0zyoj| Running | node01-f4fc.oslab.opentlc.com | 10.1.1.9
hello-openshift-1-17j8o| Running | node00-f4fc.oslab.opentlc.com | 10.1.0.15
hello-openshift-1-8rfly| Running | node01-f4fc.oslab.opentlc.com | 10.1.1.7
hello-openshift-1-9ve89| Running | node00-f4fc.oslab.opentlc.com | 10.1.0.16
hello-openshift-1-bcw8z| Running | node01-f4fc.oslab.opentlc.com | 10.1.1.8
hello-openshift-1-dtfos| Running | node00-f4fc.oslab.opentlc.com | 10.1.0.14
hello-openshift-1-mtv6s| Running | node01-f4fc.oslab.opentlc.com | 10.1.1.6
hello-openshift-1-r1lbp| Running | node00-f4fc.oslab.opentlc.com | 10.1.0.17
hello-openshift-1-y8ffs| Running | node00-f4fc.oslab.opentlc.com | 10.1.0.18
```

4. Deploy From an Existing Git Repository (Web Console, S2I Build)

In this scenario, David wants to deploy and test an application from an existing Git repository.

4.1. Review the Target Git Repository and Log In to the OpenShift Web Console

1. **Action** - Browse to https://github.com/openshift/simple-openshift-sinatra-sti.
   - **Explain** that what you see here is a Git repository containing a sample Ruby application using the Sinatra Ruby Gem.
   - **Optional** - Review the files briefly with the audience if you think it would help them understand.
2. **Action** - Browse to https://master00-GUID.oslab.opentlc.com:8443 and log in using the `david` account with password `R3dh4t1!`

   

   GUID in the URL refers to your Global Unique Identifier.

   - **Explain** that you are currently logging into the OpenShift web console as the user `david`.
   - **Point out** the following:
     - You can create users locally or link to an enterprise directory.
     - You can group users and create working teams.
     - You can use quotas to set resource limits for users, projects, and teams.

4.2. Deploy Your S2I Application

1. **Action** - Select the **SourceToImage** project.
2. **Action** - Click the **Create** button.
3. **Action** - Paste the Git repository into the **Source Repository** text box: https://github.com/openshift/simple-openshift-sinatra-sti
   - **Explain** that you are creating a new application. To do that, you need to provide OpenShift Enterprise with two key pieces of information:
     - The source code repository
     - The builder image or the base image on which to build the container
4. **Action** - Click the blue arrow to progress to the next step, and select **"ruby-20-rhel7"**.
   - **Explain** that you picked the **ruby-20-rhel7** image as your builder image. The code and all of its dependencies will be layered on top of this image.
   - **Explain** that you can have OpenShift Enterprise automatically rebuild and redeploy the entire application if an image update occurs.
   - **Explain** that you can have different `ImageStreams`. You can deploy from either certified Red Hat builder images or your own.
5. **Action** - Confirm your selection by clicking **Select this Image**.
   - **Show** and **explain** that next you select the application attributes, such as ports, routes, triggers, and more.

6. **Action** - Set the name of the application to `simplerubyapp`.
   - **Show** that you can select to have a route for the application or not.
   - **Show** that you can select the number of replicas the application has.
   - **Show** that you can set a label for the application to manage it by label.
7. **Action** - Click **Create**.
   - **Show** that you got a successful message stating "All resources for application `simplerubyapp` were created successfully."
   - **Show** that there are currently no pods created.
8. **Action** - Click **Browse** and then **Builds**
   - **Show** That you can start a build or wait for the build to start automaticaly

The web console should refresh shortly to indicate that a build was started.

4.3. The Build Process

1. **Action** - Connect as user `david` to your master host and authenticate to OpenShift Enterprise using the `oc login` command.
   - **Caution** - If you have *already logged in* to OpenShift Enterprise, *do not* run the `oc login` command again. You are already logged in, and this will result in an error on the screen.

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

2. If you are not already using the **sourcetoimage-demo** project, switch to it:

```
[david@master00~]$ oc project sourcetoimage-demo
Using project "sourcetoimage-demo"
```

3. **Action** - get information about the build using the **oc get builds** and o
   - **Explain** that you can see that you requested a build process and that you can follow the build log using simple commands.
   - **Point out** a few lines to explain to your audience if they are so inclined. For example, you can point out the following:
     - The image that OpenShift Enterprise is selecting and importing
     - The repository read and dependencies installed (Sinatra Gem)

```
[david@master00-31c5]$ oc get builds
```

```
NAME TYPE STATUS POD
simplerubyapp-1 S2I Running simplerubyapp-1

[david@master00-31c5 openshift]$ oc build-logs simplerubyapp-1
....
....
I0703 09:21:34.916120       1 docker.go:180] Image
registry.access.redhat.com/openshift3/ruby-20-rhel7:latest available locally
I0703 09:21:34.916257       1 docker.go:267] Image contains io.s2i.scripts-url
set to 'image:///usr/local/sti'
I0703 09:21:34.916472       1 download.go:56] Using image internal scripts
from: image:///usr/local/sti/assemble
I0703 09:21:34.916889       1 download.go:56] Using image internal scripts
from: image:///usr/local/sti/run
I0703 09:21:34.943521       1 docker.go:180] Image
registry.access.redhat.com/openshift3/ruby-20-rhel7:latest available locally
....
.....
I0703 09:21:36.932550       1 docker.go:357] Attaching to container
I0703 09:21:36.952808       1 docker.go:414] Starting container
I0703 09:21:37.596081       1 docker.go:424] Waiting for container
I0703 09:21:38.109326       1 sti.go:388] ---> Installing application source
I0703 09:21:38.132331       1 sti.go:388] ---> Building your Ruby application
from source
I0703 09:21:38.132537       1 sti.go:388] ---> Running 'bundle install --
deployment'
I0703 09:21:43.225774       1 sti.go:388] Fetching gem metadata from
https://rubygems.org/..........
I0703 09:21:49.860178       1 sti.go:388] Installing rack (1.5.2)
I0703 09:21:50.158742       1 sti.go:388] Installing rack-protection (1.5.3)
I0703 09:21:50.670381       1 sti.go:388] Installing tilt (1.4.1)
I0703 09:21:52.292218       1 sti.go:388] Installing sinatra (1.4.5)
I0703 09:21:52.292271       1 sti.go:388] Using bundler (1.3.5)
I0703 09:21:52.297487       1 sti.go:388] Your bundle is complete!
....
....
I0703 09:22:08.108088       1 sti.go:96] Using provided push secret for
pushing 172.30.133.153:5000/sourcetoimage/simplerubyapp image
I0703 09:22:08.108117       1 sti.go:99] Pushing
172.30.133.153:5000/sourcetoimage/simplerubyapp image ...
I0703 09:27:07.204498       1 sti.go:103] Successfully pushed
172.30.133.153:5000/sourcetoimage/simplerubyapp
```

4. **Explain** While you wait for the build to complete, **explain** the concepts of *service resources* and *route resources*.
   a. **Show** the service created for this application under **Browse** → **Services** in the web console.
   b. **Explain** services.
   c. **Show** that the route for the application was set.

5. **Action** Use the **oc get pods** command to display the pods
   a. *Show the status, or REASON, of the pod, it might still be "Pending" if the image is being deployed.

```
[david@master00-31c5 ~]$  oc get pods
NAME                     READY     REASON       RESTARTS   AGE
simplerubyapp-1-build    0/1       ExitCode:0   0          7m
simplerubyapp-1-toei3    1/1       Running      0          1m
```

4.4. Expose the service and create the route for the environment

1. **Action** - Run the **oc expose** command to create a route for the application.
   a. **Caution** - Make sure that the GUID value is populated correctly. Review the file and make sure that the `host:` value is set correctly.
   b. **Explain** that in the current version, you do not use the web console to set routes. In the near future, you will be able to do all this in the web console.
   c. **Explain** that in this scenario, you decided to add another route to your application, so it is available under another URL.
   d. **Explain** that you are creating a route so that when a user resolves `simplerubyapp.cloudapps-$GUID.oslab.opentlc.com`, you will route the user to one of the pods under the `simplerubyapp` service.
   e. **Show** - you can use curl or your browser to see the application at http://simplerubyapp.cloudapps-$GUID.oslab.opentlc.com.

```
[david@master00 ~]$ oc expose service simplerubyapp --name=simplerubyapp-route -
-hostname=simplerubyapp.cloudapps-$GUID.oslab.opentlc.com
```

2. **Optional** - Add the route manually for the environment.
   a. **Action** - Run the **oc expose** command to create a route for the application.

4.5. Complete the Demonstration

1. **Action** - Browse to: http://simplerubyapp.cloudapps-$GUID.oslab.opentlc.com.
   - **Explain** what you did, and that this is a very common workflow for every development environment.
   - **Point out** the following:
     - You created an image from a Git repository and a builder image.
     - You created a service that acts as a list that represents all of your pods.
     - You created a route to direct to that service.
     - S2I builds *do not* need to recreate the image every time. When the code changes,

the builds just "add a layer" with the code.

5. Deploy a Two-Tiered Application From a Template (Web Console and Command Line)

In this scenario, David wants to deploy a two-tiered `Web-DB` application using an `Instant Apps` template.

5.1. Deploy Your Application Using the Web Console

1. **Action** - Browse to the OpenShift Enterprise web console: https://master00-GUID.oslab.opentlc.com:8443.

   

   Remember that **GUID** in the URL refers to your Global Unique Identifier.

2. **Action** - Log in using the `david` account with password `R3dh4t1!`
   - **Explain** - I am currently logging into the OpenShift Enterprise web console as the user `david`.
   - **Point out** the following:
     - You can create users locally or link to an enterprise directory.
     - You can group users and create teams.
     - You can use quotas to set resource limits on users, projects, and teams.
3. **Action** - Select the `Instant Apps Demonstration` project.
4. **Action** - Click the **Create** button.
5. **Action** Click `instantapp-2tier-application`.
   - **Explain** that you are now creating a new application from a template that was loaded in the OpenShift Enterprise environment.
6. **Action** - Click the **Select Template** box.
   - **Explain** that you need to review the images and edit the application attributes, such as labels and parameters.
   - **Show** that you can set a label for the application to manage it by label.
   - **Show** that parameters such as usernames and credentials are generated for each template, but you can also set them manually.
7. **Action** - Click **Create**.
   - **Explain** what is about to happen: Builds are getting started and services are being created for the front end and back end.

8. **Optional** - Select **Browse** on the left side of the screen and show the **Builds**, **Services**, and **Pods** panes.

5.2. Log In and Authenticate

1. **Action** - Log in to your server and switch to the user `david`:

```
[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-GUID ~]# su - david
```

2. **Action:** - As user `david`, log in to OpenShift Enterprise and select the `instantapps-demo` project.
   - **Caution** - If you have *already logged in* to OpenShift Enterprise, *do not* run the `oc login` command again. You are already logged in, and this will result in an error on the screen.
   - **Explain** that you are currently logging in to the master as part of this demonstration, but consider that every command that `david` issues is a command that the developer could do on his or her laptop or workstation, or from wherever he or she is working.
   - **Explain** what projects are and how different projects could have different user permissions and quotas attached to them.

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

3. If you are not already using the **instantapps-demo** project, switch to it:

```
[david@master00~]$ oc project instantapps-demo
Using project "instantapps-demo"
```

5.3. Review the Build Process

1. **Action** - Run the following commands to display the current process.
   - **Explain** the process the audience is seeing and the different resources that you created.
   - **Point out** the following
     - The `service` resource created for `frontend` and `backend`

- The `route` resource created for the `frontend`

```
[david@master00~]$ oc get builds
NAME                  TYPE      STATUS     POD
ruby-sample-build-1   Source    Complete   ruby-sample-build-1-build
```

2. **Action** Look at the build logs using the **oc build-logs** command

```
[david@master00~]$ oc build-logs ruby-sample-build-1
I0703 09:57:49.921355       1 sti.go:388] ---> Installing application source
I0703 09:57:49.990848       1 sti.go:388] ---> Building your Ruby application from
source
I0703 09:57:49.990927       1 sti.go:388] ---> Running 'bundle install --
deployment'
I0703 09:57:56.212277       1 sti.go:388] Fetching gem metadata from
https://rubygems.org/.........
I0703 09:58:00.672821       1 sti.go:388] Installing rake (10.3.2)
I0703 09:58:02.017834       1 sti.go:388] Installing i18n (0.6.11)
I0703 09:58:09.992863       1 sti.go:388] Installing json (1.8.1)

...
...
I0703 09:58:57.122259       1 cfg.go:64] Using serviceaccount user for Docker
authentication
I0703 09:58:57.122318       1 sti.go:96] Using provided push secret for pushing
172.30.133.153:5000/instantapps/ruby-sample image
I0703 09:58:57.122351       1 sti.go:99] Pushing
172.30.133.153:5000/instantapps/ruby-sample image ...
I0703 10:02:01.730922       1 sti.go:103] Successfully pushed
172.30.133.153:5000/instantapps/ruby-sample
```

3. **Action** While you wait for the build to complete, expose the **service** and create the route for the application.
   - **CAUTION** Don't skip this step!, if you don't **expose** the service, the application will **NOT be accessible from the outside** world.
     - **Explain** that in this scenario, you decided to add another route to your application, so it is available under another URL.
     - **Explain** that you are creating a route so that when a user resolves `myinstant.cloudapps-$GUID.oslab.opentlc.com`, you will "route" (actualy its more like "proxy") the user to one of the pods under the `frontend` service.

```
[david@master00~]$ oc expose service frontend --name=myinst-route --
hostname=myinst.cloudapps-$GUID.oslab.opentlc.com
```

4. **Show** that the pods were all created, 2 for the **frontend** and 1 **database** backend.

```
[david@master00~]$ oc get pods
NAME                        READY     REASON       RESTARTS   AGE
database-1-3vjjb            1/1       Running      0          5m
frontend-1-akq23            1/1       Running      0          25s
frontend-1-yiivo            1/1       Running      0          24s
ruby-sample-build-1-build   0/1       ExitCode:0   0          5m
```

5. **Show** that the Services were all created, 1 for the **frontend** and 1 **database** backend service.

```
[david@master00~]$ oc get services
NAME       LABELS                                    SELECTOR        IP(S)
PORT(S)
database   template=application-template-stibuild   name=database   172.30.176.104
5434/TCP
frontend   template=application-template-stibuild   name=frontend   172.30.149.55
5432/TCP
```

5.4. Complete the Demonstration

1. **Action** - Browse to: instantapp.cloudapps-$GUID.oslab.opentlc.com.
   - **Explain** what you did, and that this is a very common workflow for every development environment.
   - **Point out** the following:
     - You created a Ruby front end and a database backend.
     - Using the template, both parts of the application environment can share values like usernames and passwords.
     - You can randomize and generate values for each template.
     - You created a route to direct to the front end service.

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