Guided Exercise: Deploy Applications in OpenShift

Create and modify Red Hat OpenShift Container Platform (RHOCP) objects.

Outcomes

You should be able to:

- Use the RHOCP Web Console to verify the status of applications.
- Use the RHOCP Web Console to modify the `Service` objects.
- Use the `oc` command-line interface to verify the status of applications.
- Use the `oc` command-line interface to deploy new applications.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command deploys the `podman-hello` application into the `ocp-applications` project.

```
[student@workstation ~]$ lab start openshift-applications
```

Instructions

1. Log in to the cluster as the `developer` user, and ensure that you use the `ocp-applications` project.

   Log in to the cluster as the `developer` user.

   ```
   [student@workstation ~]$ oc login -u developer -p developer \
     https://api.ocp4.example.com:6443
   Login successful.

   ...output omitted...
   ```

   Ensure that you use the `ocp-applications` project.

   ```
   [student@workstation ~]$ oc project ocp-applications
   Already on project "ocp-applications" on server "https://api.ocp4.example.com:6443".
   ```

2. Explore the `podman-hello-client` application.

   Open the `/home/student/DO188/labs/openshift-applications/podman-hello-client/Containerfile` file in a text editor, such as `gedit`.

   Note the default environment variable values and the `CMD` instruction:

   ```
   FROM registry.access.redhat.com/ubi8/ubi-minimal:8.6

   ARG PROTO="http" \
       URL="hello-server-svc" \
       PORT="3000" \
       ENDPOINT="greet"

   ENV PROTO=${PROTO} \
       URL=${URL} \
       PORT=${PORT} \
       ENDPOINT=${ENDPOINT}

   ...Containerfile omitted...

   CMD ["./client.sh"]
   ```

   Open the `/home/student/DO188/labs/openshift-applications/podman-hello-client/client.sh` file in a text editor, such as `gedit`.

   Note that the client uses the environment variables in the following format:

   ```
   ...script omitted...
   curl ${OPTS} "${PROTO}://${URL}:${PORT}/${ENDPOINT}"
   ...script omitted...
   ```

   This means that the client sends requests to the `http://hello-server-svc:3000/greet` URL by default.

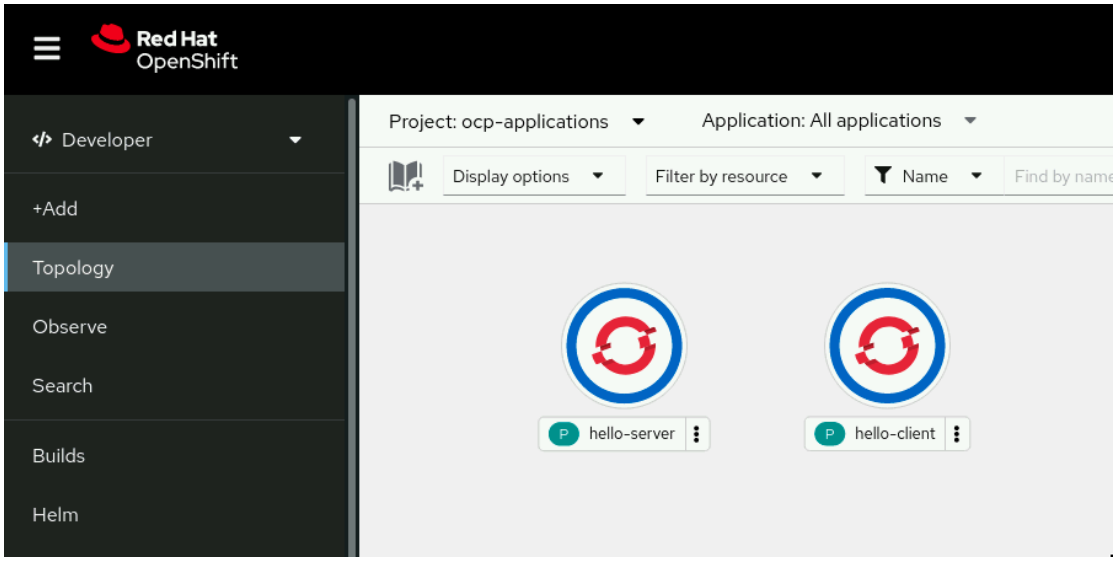
3. Explore the deployed application in the `ocp-applications` project by using the RHOCP Web Console.

In a web browser, go to `https://console-openshift-console.apps.ocp4.example.com`. Select the **htpasswd_provider** user provider, and log in with the following credentials:

- **Username**: `developer`
- **Password**: `developer`

Click **Skip tour** if prompted.

Click **ocp-applications**, then click the overflow menu and select **Topology** on the sidebar. Note the `hello-server` and `hello-client` application pods.



Click the **hello-server** pod. Notice the `hello-server-svc` service.

The service serves on port `3000` and routes requests to the port `3000` inside the `hello-server` pod.

Click the **hello-client** pod. Then, view the application logs by clicking **View logs**.

The `hello-client` application sends requests to the `http://hello-server-svc:3000/greet` URL and prints the `{"hello": "world"}` response to the console.

Return to the **Topology** view by clicking the **Red Hat OpenShift** logo.

4. Modify the `hello-server-svc` service to serve on the `8080` port.

   Click the **hello-server** pod. Then, click the **hello-server-svc** service.

   Click the **YAML** tab. In the text editor, scroll to the `spec.ports` object and change the `port` property to the `8080` value:

   ```
   ...YAML omitted...
   spec:
     ports:
       - protocol: TCP
         port: 8080
         targetPort: 3000
   ...YAML omitted...
   ```

   Click **Save**.

   Verify the service port mapping.

   Click the **Details** tab and verify that the **Service port Mapping** section contains the `8080` **Port** and the `3000` **Pod port or name** values.

5. Modify the `hello-client` pod to use the `8080` port.

   Return to the terminal window and verify that the `oc` command is logged in with the RHOCP API.

   ```
   [student@workstation ~]$ oc whoami --show-context
   ocp-applications/api-ocp4-example-com:6443/developer
   ```

   Display the last line of the `hello-client` pod logs.

```
[student@workstation ~]$ oc logs hello-client | tail -n 1
empty line expected
```

In the preceding instruction, the output of the `oc logs hello-client` command is used as the input for the `tail` command. The `tail -n 1` command displays the last line of the output.

The result is empty because `hello-client` cannot reach the `hello-server` pod on port `3000`.

Delete the `hello-client` pod.

```
[student@workstation ~]$ oc delete pod hello-client
pod "hello-client" deleted
```

Recreate the `hello-client` pod. Set the `PORT` environment variable to the `8080` value and use the `registry.ocp4.example.com:8443/redhattraining/podman-hello-client:latest` image.

```
[student@workstation ~]$ oc run hello-client --env PORT=8080 \
  --image registry.ocp4.example.com:8443/redhattraining/podman-hello-client:latest
pod/hello-client created
```

Verify that the `hello-client` pod is in the `Running` status.

```
[student@workstation ~]$ oc get pod
NAME           READY   STATUS    RESTARTS   AGE
hello-client   1/1     Running   0          32s
hello-server   1/1     Running   0          83m
```

Display the last line of the `hello-client` pod logs.

```
[student@workstation ~]$ oc logs hello-client | tail -n 1
{"hello":"world"}
```

This means that the `hello-client` pod can reach the `hello-server` pod.

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish openshift-applications
```