Guided Exercise: Automatic Image Updates with OpenShift Image Change Triggers

Update an application that references container images indirectly though image streams.

Outcomes

- Add an image trigger to a deployment.
- Modify an image stream tag to point to a new image.
- Watch the rollout of the application.
- Roll back a deployment to the previous image.

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

This command ensures that all resources are available for this exercise. It also creates the `updates-triggers` project and deploys a web application with 10 replicas.

The command creates the `/home/student/DO180/labs/updates-triggers/resources.txt` file. The `resources.txt` file contains the name of the images and some commands that you use during the exercise. You can use the file to copy and paste these image names and commands.

```
[student@workstation ~]$ lab start updates-triggers
```

Instructions

1. Log in to the OpenShift cluster as the `developer` user with the `developer` password. Use the `updates-triggers` project.

   Log in to the OpenShift cluster.

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

   Set the `updates-triggers` project as the active project.

   ```
   [student@workstation ~]$ oc project updates-triggers
   ...output omitted...
   ```

2. Inspect the `versioned-hello` image stream that the `lab` command created.

   Verify that the `lab` command enabled the local lookup policy for the `versioned-hello` image stream.

   ```
   [student@workstation ~]$ oc set image-lookup
   NAME             LOCAL
   versioned-hello  true
   ```

   Verify that the `lab` command created the `versioned-hello:1` image stream tag. The image stream tag refers to the image in the classroom registry by its SHA ID.

   > **NOTE**
   >
   > To improve readability, the instructions truncate the SHA-256 strings.
   >
   > On your system, the commands return the full SHA-256 strings. Also, you must type the full SHA-256 string, to provide such a parameter to a command.

   ```
   [student@workstation ~]$ oc get istag
   NAME               IMAGE REFERENCE ...
   versioned-hello:1  ...:8443/redhattraining/versioned-hello@sha256:66e0...105e ...
   ```

   Verify that the `lab` command created the `versioned-hello:1` image stream tag from the `registry.ocp4.example.com:8443/redhattraining/versioned-hello:1-123` image.

```
[student@workstation ~]$ oc get istag versioned-hello:1 \
  -o jsonpath='{.tag.from.name}{"\n"}'
registry.ocp4.example.com:8443/redhattraining/versioned-hello:1-123
```

3. Inspect the `Deployment` object that the `lab` command created. Verify that the application is available from outside the cluster.

   List the `Deployment` objects. The `version` deployment retrieved the SHA image ID from the `versioned-hello:1` image stream tag. The `Deployment` object includes a container named `versioned-hello`. You use that information in a later step, when you configure the trigger.

   ```
   [student@workstation ~]$ oc get deployment -o wide
   NAME     READY ... CONTAINERS        IMAGES ...
   version  10/10 ... versioned-hello   .../versioned-hello:1 ...
   ```

   Open a new terminal.

   Run the `/home/student/DO180/labs/updates-triggers/curl_loop.sh` script that the `lab` command prepared. The script sends web requests to the application in a loop. Leave the script running and do not interrupt it.

   ```
   [student@workstation ~]$ /home/student/DO180/labs/updates-triggers/curl_loop.sh
   Hi!
   Hi!
   Hi!
   Hi!
   ...output omitted...
   ```

4. Add an image trigger to the `Deployment` object.

   Return to the first terminal window, and then use the `oc set triggers` command to add the trigger for the `versioned-hello:1` image stream tag to the `versioned-hello` container.

   ```
   [student@workstation ~]$ oc set triggers deployment/version \
     --from-image versioned-hello:1 --containers versioned-hello
   deployment.apps/version triggers updated
   ```

   Review the definition of the trigger from the `image.openshift.io/triggers` annotation.

   ```
   [student@workstation ~]$ oc get deployment version \
     -o jsonpath='{.metadata.annotations.image\.openshift\.io/triggers}' | jq .
   [
     {
       "from": {
         "kind": "ImageStreamTag",
         "name": "versioned-hello:1"
       },
       "fieldPath": "spec.template.spec.containers[?(@.name==\"versioned-hello\")].image"
     }
   ]
   ```

5. Update the `versioned-hello:1` image stream tag to point to the `1-125` tag of the `registry.ocp4.example.com:8443/redhattraining/versioned-hello` image. Watch the output of the `curl_loop.sh` script to verify that the `Deployment` object automatically rolls out.

   Use the `oc tag` command to update the `versioned-hello:1` tag.

   ```
   [student@workstation ~]$ oc tag \
     registry.ocp4.example.com:8443/redhattraining/versioned-hello:1-125 \
     versioned-hello:1
   Tag versioned-hello:1 set to registry.ocp4.example.com:8443/redhattraining/versioned-hello:1-125.
   ```

   Changing the image stream tag triggered a rolling update. Watch the output of the `curl_loop.sh` script in the second terminal.

   Before the update, only pods that use the earlier version of the image reply. During the rolling updates, both old and new pods respond. After the update, only pods that run the latest version of the image reply. The following output probably differs on your system, and after the rollout completes, shows only the `v1.1` version.

```
...output omitted...
Hi!
Hi!
Hi!
Hi!
Hi! v1.1
Hi! v1.1
Hi!
Hi! v1.1
Hi!
Hi! v1.1
Hi! v1.1
Hi! v1.1
Hi! v1.1
...output omitted...
```

Do not stop the script.

6. Inspect the `Deployment` object and the image stream.

   List the `version` deployment and notice that the image changed.

```
[student@workstation ~]$ oc get deployment version -o wide
NAME     READY ... CONTAINERS        IMAGES ...
version  10/10 ... versioned-hello   .../versioned-hello@sha256:834d...fcb4 ...
```

   Display the details of the `versioned-hello` image stream. The `versioned-hello:1` image stream tag points to the image with the same SHA ID as in the `Deployment` object.

   Notice that the preceding image is still available. In the following step, you roll back to that image by specifying its SHA ID.

```
[student@workstation ~]$ oc describe is versioned-hello
Name:           versioned-hello
Namespace:      updates-triggers
...output omitted...

1
  tagged from registry.ocp4.example.com:8443/redhattraining/versioned-hello:1-125

  * registry.ocp4.example.com:8443/.../versioned-hello@sha256:834d...fcb4
      6 minutes ago
    registry.ocp4.example.com:8443/.../versioned-hello@sha256:66e0...105e
      37 minutes ago
```

7. Roll back the `Deployment` object by reverting the `versioned-hello:1` image stream tag.

   Use the `oc tag` command. For the source image, copy and paste the old image name and the SHA ID from the output of the preceding command.

```
[student@workstation ~]$ oc tag \
registry.ocp4.example.com:8443/redhattraining/versioned-hello@sha256:66e0...105e \
versioned-hello:1
Tag versioned-hello:1 set to registry.ocp4.example.com:8443/redhattraining/versioned-hello@sha256:66e0...105e.
```

   Watch the output of the `curl_loop.sh` script in the second terminal. The pods that run the `v1.0` version of the application are responding again. The following output probably differs on your system, and after the rollout completes, shows only the `v1.0` version.

```
...output omitted...
Hi! v1.1
Hi! v1.1
Hi! v1.1
Hi! v1.1
Hi!
Hi! v1.1
Hi! v1.1
Hi! v1.1
Hi!
Hi! v1.1
Hi! v1.1
Hi!
Hi!
Hi!
...output omitted...
```

Press **Ctrl**+**C** to end the script execution. Close the second terminal.

Finish

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

```
[student@workstation ~]$ lab finish updates-triggers
```