Automatic Image Updates with OpenShift Image Change Triggers

Objectives

- Ensure automatic update of application pods by using image streams with Kubernetes workload resources.

Using Triggers to Manage Images

Image stream tags record the SHA ID of the source container image. Thus, an image stream tag always points to an immutable image.

If a new version of the source image becomes available, then you can change the image stream tag to point to that new image. However, a `Deployment` object that uses the image stream tag does not roll out automatically. For an automatic rollout, you must configure the `Deployment` object with an image trigger.

If you update an image stream tag to point to a new image version, and you notice that this version does not work as expected, then you can revert the image stream tag. `Deployment` objects for which you configured a trigger automatically roll back to that previous image.

Other Kubernetes workloads also support image triggers, such as `Pod`, `CronJob`, and `Job` objects.

Configuring Image Triggers for Deployments

Before you can configure image triggers for a `Deployment` object, ensure that the `Deployment` object is using image stream tags for its containers:

- Create the image stream object in the same project as the `Deployment` object.
- Enable the local lookup policy in the image stream object by using the `oc set image-lookup` command.
- In the `Deployment` object, reference the image stream tags by their names, such as `keycloak:20`, and not by the full image names from the source registry.

Image triggers apply at the container level. If your `Deployment` object includes several containers, then you can specify a trigger for each one. Before you can set triggers, retrieve the container names:

```
[user@host ~]$ oc get deployment mykeycloak -o wide
NAME         READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS ...
mykeycloak   0/1     1            0           6s    keycloak   ...
```

Use the `oc set triggers` command to configure an image trigger for the container inside the `Deployment` object. Use the `--from-image` option to specify the image stream tag to watch.

```
[user@host ~]$ oc set triggers deployment/mykeycloak --from-image keycloak:20 \
  --containers keycloak
```

To provide automatic image rollout for `Deployment` objects, OpenShift adds the `image.openshift.io/triggers` annotation to store the configuration in JSON format.

The following example retrieves the content of the `image.openshift.io/triggers` annotation, and then uses the `jq` command to display the configuration in a more readable format:

```
[user@host ~]$ oc get deployment mykeycloak \
  -o jsonpath='{.metadata.annotations.image\.openshift\.io/triggers}' | jq .
[
  {
    "from": {
      "kind": "ImageStreamTag",
      "name": "keycloak:20"
    },
    "fieldPath": "spec.template.spec.containers[?(@.name==\"keycloak\")].image"
  }
]
```

The `fieldPath` attribute is a JSONPath expression that OpenShift uses to locate the attribute that stores the container image name. OpenShift updates that attribute with the new image name and SHA ID whenever the image stream tag changes.

For a more concise view, use the `oc set triggers` command with the name of the `Deployment` object as an argument:

```
[user@host ~]$ oc set triggers deployment/mykeycloak
NAME                    TYPE    VALUE                   AUTO
deployments/mykeycloak  config                          true (1)
deployments/mykeycloak  image   keycloak:20 (keycloak)  true (2)
```

(1) OpenShift uses the configuration trigger to roll out the deployment whenever you change its configuration, such as to update environment variables or to configure the readiness probe.

(2) OpenShift watches the `keycloak:20` image stream tag that the `keycloak` container uses.

The `true` value under the `AUTO` column indicates that the trigger is enabled.

You can disable the configuration trigger by using the `oc rollout pause` command, and you can re-enable it by using the `oc rollout resume` command.

You can disable the image trigger by adding the `--manual` option to the `oc set triggers` command:

```
[user@host ~]$ oc set triggers deployment/mykeycloak --manual \
  --from-image keycloak:20 --containers keycloak
```

You re-enable the trigger by using the `--auto` option:

```
[user@host ~]$ oc set triggers deployment/mykeycloak --auto \
  --from-image keycloak:20 --containers keycloak
```

You can remove the triggers from all the containers in the `Deployment` object by adding the `--remove-all` option to the command:

```
[user@host ~]$ oc set triggers deployment/mykeycloak --remove-all
```

Rolling out Deployments

A `Deployment` object with an image trigger automatically rolls out when the image stream tag changes.

The image stream tag might change because you manually update it to point to a new version of the source image. The image stream tag might also change automatically if you configure it for periodic refresh, by adding the `--scheduled` option to the `oc tag` command. When the image stream tag automatically changes, all the `Deployment` objects with a trigger that refers to that image stream tag also roll out.

Rolling Back Deployments

To roll back the `Deployment` objects, you revert the image stream tag. By reverting the image stream tag, OpenShift rolls out the `Deployment` object to use the previous image that the image stream tag is pointing to again.

Managing Image Stream Tags

You can create image streams and image stream tags in several ways. The following three commands perform the same operation. They all create the `keycloak` image stream if it does not exist, and then create the `keycloak:20.0.2` image stream tag:

```
[user@host ~]$ oc create istag keycloak:20.0.2 \
  --from-image quay.io/keycloak/keycloak:20.0.2

[user@host ~]$ oc import-image keycloak:20.0.2 \
  --from quay.io/keycloak/keycloak:20.0.2 --confirm

[user@host ~]$ oc tag quay.io/keycloak/keycloak:20.0.2 keycloak:20.0.2
```

You can rerun the `oc import-image` and `oc tag` commands to update the image stream tag from the source image. If the source image changes, then the commands update the image stream tag to point to that new version. However, you can use the `oc create istag` command only for the initial creation of the image stream tag. You cannot update tags by using the `oc create istag` command.

The `--help` option shows more details about these commands.

You can create several image stream tags that point to the same image. The following command creates the `keycloak:20` image stream tag, which points to the same image as the `keycloak:20.0.2` image stream tag. In this example, the `keycloak:20` image stream tag is an alias for the `keycloak:20.0.2` image stream tag.

```
[user@host ~]$ oc tag --alias keycloak:20.0.2 keycloak:20
```

The `oc describe is` command reports that both tags point to the same image:

```
[user@host ~]$ oc describe is keycloak
Name:       keycloak
Namespace:  myproject
...output omitted...

20.0.2 (20)
  tagged from quay.io/keycloak/keycloak:20.0.2

  * quay.io/keycloak/keycloak@sha256:5569...b311
      3 minutes ago
```

Using aliases for image tags is a similar concept to the use of floating tags for container images. Suppose that a new image version is available in the Quay.io repository. You can create an image stream tag for that new image:

```
[user@host ~]$ oc create istag keycloak:20.0.3 \
  --from-image quay.io/keycloak/keycloak:20.0.3
imagestreamtag.image.openshift.io/keycloak:20.0.3 created
[user@host ~]$ oc describe is keycloak
Name:       keycloak
Namespace:  myproject
...output omitted...

20.0.3
  tagged from quay.io/keycloak/keycloak:20.0.3

  * quay.io/keycloak/keycloak@sha256:c167...62e9
      36 seconds ago

20.0.2 (20)
  tagged from quay.io/keycloak/keycloak:20.0.2

  * quay.io/keycloak/keycloak@sha256:5569...b311
      About an hour ago
```

The `keycloak:20` image stream tag does not change. Therefore, the `Deployment` objects that use that tag do not roll out.

After testing the new image, you can move the `keycloak:20` tag to point to the new image stream tag:

```
[user@host ~]$ oc tag --alias keycloak:20.0.3 keycloak:20
Tag keycloak:20 set up to track keycloak:20.0.3.
[user@host ~]$ oc describe is keycloak
Name:       keycloak
Namespace:  myproject
...output omitted...

20.0.3 (20)
  tagged from quay.io/keycloak/keycloak:20.0.3

  * quay.io/keycloak/keycloak@sha256:c167...62e9
      10 minutes ago

20.0.2
  tagged from quay.io/keycloak/keycloak:20.0.2

  * quay.io/keycloak/keycloak@sha256:5569...b311
      About an hour ago
```

Because the `keycloak:20` image stream tag points to a new image, OpenShift rolls out all the `Deployment` objects that use that tag. These new deployments that use the new image should be tested to ensure that they perform as expected. If the new application does not work as expected, you can roll back the deployments by resetting the `keycloak:20` tag to the previous image stream tag, as shown here:

```
[user@host ~]$ oc tag --alias keycloak:20.0.2 keycloak:20
```

By providing a level of abstraction, image streams give you control over managing the container images that you use in your OpenShift cluster.

REFERENCES

For more information about image triggers, refer to the *Triggering Updates on Image Stream Changes* chapter in the Red Hat OpenShift Container Platform 4.18 *Images* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/images/index#images-tag_tagging-images

For more information about image stream tags, refer to the *Tagging Images* section in the *Managing Images* chapter in the Red Hat OpenShift Container Platform 4.18 *Images* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/images/index#tagging-images

Using Red Hat OpenShift Image Streams with Kubernetes Deployments