Guided Exercise: Manage Long-lived and Short-lived Applications by Using the Kubernetes Workload API

Deploy a batch application managed by a job resource and a database server that a deployment resource manages by using the Kubernetes command-line interface.

Outcomes

In this exercise, you deploy a database server and a batch application, both managed by workload resources that the deployment resources manage.

- Create deployments.
- Update environment variables on a pod template.
- Create and run job resources.
- Retrieve the logs and termination status of a job.
- View the pod template of a job resource.

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

This command ensures resource availability for the exercise.

```
[student@workstation ~]$ lab start deploy-workloads
```

Instructions

1. As the `developer` user, create a MySQL deployment in a new project.

   Log in as the `developer` user with the `developer` password.

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

   Create a project named `deploy-workloads`.

   ```
   [student@workstation ~]$ oc new-project deploy-workloads
   Now using project "deploy-workloads" on server "https://api.ocp4.example.com:6443".
   ...output omitted...
   ```

   Create a deployment that runs an ephemeral MySQL server.

   ```
   [student@workstation ~]$ oc create deployment my-db \
     --image registry.ocp4.example.com:8443/rhel9/mysql-80:1
   deployment.apps/my-db created
   ```

   Retrieve the status of the deployment.

   ```
   [student@workstation ~]$ oc get deployments
   NAME    READY   UP-TO-DATE   AVAILABLE   AGE
   my-db   0/1     1            0           67s
   ```

   The deployment never shows a ready instance.

   Retrieve the status of the created pod. Your pod name might differ from the output.

   ```
   [student@workstation ~]$ oc get pods
   NAME                     READY   STATUS             RESTARTS      AGE
   my-db-8567b478dd-d28f7   0/1     CrashLoopBackOff   4 (60s ago)   2m35s
   ```

   The pod fails to start and repeatedly crashes. If the pod status shows `Pending`, allow a few minutes for it to crash before proceeding.

   Review the logs for the pod to determine why it fails to start.

```
[student@workstation ~]$ oc logs deploy/my-db
...output omitted...
You must either specify the following environment variables:
  MYSQL_USER (regex: '^$')
  MYSQL_PASSWORD (regex: '^[a-zA-Z0-9_~!@#$%^&*()-=<>,.?;:|]$')
  MYSQL_DATABASE (regex: '^$')
Or the following environment variable:
  MYSQL_ROOT_PASSWORD (regex: '^[a-zA-Z0-9_~!@#$%^&*()-=<>,.?;:|]$')
...output omitted...
```

Note that the container fails to start due to missing environment variables.

2. Fix the database deployment and verify that the server runs.

   Set the `MYSQL_USER`, `MYSQL_PASSWORD`, and `MYSQL_DATABASE` environment variables.

```
[student@workstation ~]$ oc set env deployment/my-db \
  MYSQL_USER=developer \
  MYSQL_PASSWORD=developer \
  MYSQL_DATABASE=sampledb
deployment.apps/my-db updated
```

   Retrieve the list of deployments and observe that the `my-db` deployment has a running pod.

```
[student@workstation ~]$ oc get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
my-db   1/1     1            1           4m50s
```

   Retrieve the internal IP address of the MySQL pod within the list of all pods.

```
[student@workstation ~]$ oc get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP          ...
my-db-748c97d478-g8xc9   1/1     Running   0          64s   10.8.0.91 ...
```

   The `-o wide` option provides additional output, such as IP addresses. Your IP address value might differ from the previous output.

   Store the IP address of the MySQL pod as a variable for use in the next step.

```
[student@workstation ~]$ DB_IP=$(oc get pods my-db-7459cb94-zdb5p \
  -o=jsonpath='{.status.podIP}')
```

   Verify that the database server runs by executing a query. This command uses the variable that contains the IP address from preceding step.

```
[student@workstation ~]$ oc run -it db-test --restart=Never \
  --image registry.ocp4.example.com:8443/rhel9/mysql-80:1 \
  -- mysql sampledb -h $DB_IP -u developer --password=developer \
  -e "select 1;"
...output omitted...
---
| 1 |
---
| 1 |
---
```

3. Delete the database server pod and observe that the deployment causes the pod to be re-created.

   Delete the existing MySQL pod by using the label that is associated with the deployment.

```
[student@workstation ~]$ oc delete pod -l app=my-db
pod "my-db-84c8995d5-2sssl" deleted
```

   Retrieve the information for the MySQL pod and observe that it is newly created. Your pod name might differ in your output.

```
[student@workstation ~]$ oc get pod -l app=my-db
NAME                    READY   STATUS    RESTARTS   AGE
my-db-fbccb9447-p99jd   1/1     Running   0          6s
```

4. Create and apply a job resource that prints the time and date repeatedly.

   Create a job resource called `date-loop` that runs a script. Ignore the warning.

```
[student@workstation ~]$ oc create job date-loop \
  --image registry.ocp4.example.com:8443/ubi9/ubi \
  -- /bin/bash -c "for i in {1..30}; do date; done"
job.batch/date-loop created
```

Retrieve the job resource to review the pod specification.

```
[student@workstation ~]$ oc get job date-loop -o yaml
...output omitted...
    spec:
      containers:
      - command: (1)
        - /bin/bash
        - -c
        - for i in {1..30}; do date; done
        image: registry.ocp4.example.com:8443/ubi9/ubi (2)
        imagePullPolicy: Always
        name: date-loop
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Never (3)
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
...output omitted...
```

(1) The `command` object, which specifies the defined script to execute within the pod.

(2) Sets the container image for the pod.

(3) Defines the restart policy for the pod. Kubernetes does not restart the job pod after the pod exits.

List the jobs to see that the `date-loop` job completed successfully.

```
[student@workstation ~]$ oc get jobs
NAME          COMPLETIONS   DURATION   AGE
date-loop     1/1           7s         8s
```

You might need to wait for the script to finish and run the command again.

Retrieve the logs for the associated pod. The log values might differ in your output.

```
[student@workstation ~]$ oc logs job/date-loop
Fri Nov 18 14:50:56 UTC 2022
Fri Nov 18 14:50:59 UTC 2022
...output omitted...
```

5. Delete the pod for the `date-loop` job and observe that the pod is not created again.

   Delete the associated pod.

```
[student@workstation ~]$ oc delete pod -l job-name=date-loop
pod "date-loop-wvn2q" deleted
```

   View the list of pods and observe that the pod is not re-created for the job.

```
[student@workstation ~]$ oc get pod -l job-name=date-loop
No resources found in deploy-workloads namespace.
```

   Verify that the job status is still listed as successfully completed.

```
[student@workstation ~]$ oc get job -l job-name=date-loop
NAME        COMPLETIONS   DURATION   AGE
date-loop   1/1           7s         7m36s
```

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 deploy-workloads
```