Guided Exercise: Application Autoscaling

Configure an autoscaler for an application and then load test that application to observe scaling up.

Outcomes

- Deploy an application.
- Manually scale the application.
- Configure application resource requests.
- Deploy the application with a Horizontal Pod Autoscaler.

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

This command ensures that the cluster is accessible and that all resources are available for this exercise. It also creates the `reliability-autoscaling` project.

```
[student@workstation ~]$ lab start reliability-autoscaling
```

Instructions

1. Log in to the OpenShift cluster as the `developer` user with `developer` as the password. Use the `reliability-autoscaling` 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 `reliability-autoscaling` project as the active project.

   ```
   [student@workstation ~]$ oc project reliability-autoscaling
   ...output omitted...
   ```

2. Create the `loadtest` deployment, service, and route. The deployment uses the `registry.ocp4.example.com:8443/redhattraining/loadtest:v1.0` container image, which provides a web application. The web application exposes an API endpoint that creates a CPU-intensive task when queried.

   Review the `~/DO180/labs/reliability-autoscaling/loadtest.yml` resource file that the `lab` command prepared. The container specification does not include the `resources` section that you use to specify CPU requests and limits. You configure that section in another step. Do not change the file for now.

   ```
   apiVersion: v1
   kind: List
   metadata: {}
   items:
     - apiVersion: apps/v1
       kind: Deployment
   ...output omitted...
           spec:
             containers:
             - image: registry.ocp4.example.com:8443/redhattraining/loadtest:v1.0
               name: loadtest
               readinessProbe:
                 failureThreshold: 3
                 httpGet:
                   path: /api/loadtest/v1/healthz
                   port: 8080
                   scheme: HTTP
                 periodSeconds: 10
                 successThreshold: 1
                 timeoutSeconds: 1

     - apiVersion: v1
       kind: Service
   ...output omitted...

     - apiVersion: route.openshift.io/v1
       kind: Route
   ...output omitted...
   ```

Use the `oc apply` command to create the application.

```
[student@workstation ~]$ oc apply -f \
~/DO180/labs/reliability-autoscaling/loadtest.yml
deployment.apps/loadtest created
service/loadtest created
route.route.openshift.io/loadtest created
```

Run the watch command and wait for the pod to report the `Running` status. The name of the pod is different in your cluster.

```
[student@workstation ~]$ watch oc get pods
Every 2.0s: oc get pods

NAME                        READY   STATUS    RESTARTS   AGE
loadtest-86d987b9bf-w66x9   1/1     Running   0          40s
```

Press **Ctrl**+**C** to end the `watch` command.

3. Configure a horizontal pod autoscaler resource for the `loadtest` deployment. Set the minimum number of replicas to 2 and the maximum to 20. Set the average CPU usage to 50% of the CPU requests attribute.

   The horizontal pod autoscaler does not work, because the `loadtest` deployment does not specify requests for CPU usage.

   Use the `oc autoscale` command to create the horizontal pod autoscaler resource.

```
[student@workstation ~]$ oc autoscale deployment/loadtest --min 2 --max 20 \
--cpu-percent 50
horizontalpodautoscaler.autoscaling/loadtest autoscaled
```

   Retrieve the status of the `loadtest` horizontal pod autoscaler resource. The `unknown` value in the `TARGETS` column indicates that OpenShift cannot compute the current CPU usage of the `loadtest` deployment. The deployment must include the CPU requests attribute for OpenShift to be able to compute the CPU usage.

```
[student@workstation ~]$ oc get hpa loadtest
NAME      REFERENCE            TARGETS         MINPODS  MAXPODS   REPLICAS   AGE
loadtest  Deployment/loadtest  <unknown>/50%   2        20        2          74s
```

   Get more details about the resource status. You might have to rerun the command several times. Wait up to three minutes for the command to report the warning message.

```
[student@workstation ~]$ oc describe hpa loadtest
Name:                                                  loadtest
Namespace:                                             reliability-autoscaling
...output omitted...
Conditions:
  Type           Status  Reason                   Message
  ----           ------  ------                   -------
  AbleToScale    True    SucceededGetScale        the HPA controller was able to get the target's current scale
  ScalingActive  False   FailedGetResourceMetric  the HPA was unable to compute the replica count: failed to get cpu utili
zation: missing request for cpu
Events:
  Type     ... Message
  ----     ... -------
...output omitted...
  Warning  ... failed to get cpu utilization: missing request for cpu
...output omitted...
```

   Delete the horizontal pod autoscaler resource. You re-create the resource in another step, after you fix the `loadtest` deployment.

```
[student@workstation ~]$ oc delete hpa loadtest
horizontalpodautoscaler.autoscaling "loadtest" deleted
```

   Delete the `loadtest` application.

```
[student@workstation ~]$ oc delete -f \
~/DO180/labs/reliability-autoscaling/loadtest.yml
deployment.apps "loadtest" deleted
service "loadtest" deleted
route.route.openshift.io "loadtest" deleted
```

4. Add a CPU resource section to the `~/DO180/labs/reliability-autoscaling/loadtest.yml` file. Redeploy the application from the file.

Edit the `~/DO180/labs/reliability-autoscaling/loadtest.yml` file, and configure the CPU limits and requests for the `loadtest` deployment. The pod needs 25 millicores to operate, and must not consume more that 100 millicores.

You can compare your work with the completed `~/DO180/solutions/reliability-autoscaling/loadtest.yml` file that the `lab` command prepared.

```
...output omitted...
        spec:
          containers:
          - image: registry.ocp4.example.com:8443/redhattraining/loadtest:v1.0
            name: loadtest
            readinessProbe:
              failureThreshold: 3
              httpGet:
                path: /api/loadtest/v1/healthz
                port: 8080
                scheme: HTTP
              periodSeconds: 10
              successThreshold: 1
              timeoutSeconds: 1
            resources:
              requests:
                cpu: 25m
              limits:
                cpu: 100m
...output omitted...
```

Use the `oc apply` command to deploy the application from the file.

```
[student@workstation ~]$ oc apply -f \
~/DO180/labs/reliability-autoscaling/loadtest.yml
deployment.apps/loadtest created
service/loadtest created
route.route.openshift.io/loadtest created
```

Run the watch command and wait until the pod is running. The name of the pod is different in your cluster.

```
[student@workstation ~]$ watch oc get pods
NAME                        READY   STATUS    RESTARTS   AGE
loadtest-667bdcdc99-vhc9x   1/1     Running   0          36s
```

Press **Ctrl**+**C** to end the watch command after the pod displays `Running` in the STATUS column.

5. Manually scale the `loadtest` deployment by first increasing and then decreasing the number of running pods.

   Scale up the `loadtest` deployment to five pods.

```
[student@workstation ~]$ oc scale deployment/loadtest --replicas 5
deployment.apps/loadtest scaled
```

Run the watch command and wait until the five pods are running. The names of the pods are different in your cluster.

```
[student@workstation ~]$ watch oc get pods
NAME                        READY   STATUS    RESTARTS   AGE
loadtest-667bdcdc99-5fcvh   1/1     Running   0          43s
loadtest-667bdcdc99-dpspr   1/1     Running   0          42s
loadtest-667bdcdc99-hkssk   1/1     Running   0          43s
loadtest-667bdcdc99-vhc9x   1/1     Running   0          8m11s
loadtest-667bdcdc99-z5n9q   1/1     Running   0          43s
```

Press **Ctrl**+**C** to end the watch command after the pods display `Running` in the STATUS column.

Scale down the `loadtest` deployment back to one pod.

```
[student@workstation ~]$ oc scale deployment/loadtest --replicas 1
deployment.apps/loadtest scaled
```

Run the watch command and wait until the pod is running. The name of the pod is different in your cluster.

```
[student@workstation ~]$ watch oc get pods
NAME                        READY   STATUS    RESTARTS   AGE
loadtest-667bdcdc99-vhc9x   1/1     Running   0          11m
```

Press **Ctrl**+**C** to end the watch command after the pod displays `Running` in the STATUS column.

6. Configure a horizontal pod autoscaler resource for the `loadtest` deployment. Set the minimum number of replicas to 2 and the maximum to 20. Set the average CPU usage to 50% of the CPU request attribute.

   Use the `oc autoscale` command to create the horizontal pod autoscaler resource.

```
[student@workstation ~]$ oc autoscale deployment/loadtest --min 2 --max 20 \
--cpu-percent 50
horizontalpodautoscaler.autoscaling/loadtest autoscaled
```

   Open a new terminal window and run the `watch` command to monitor the `oc get hpa loadtest` command. Wait up to five minutes for the `loadtest` horizontal pod autoscaler to report usage in the `TARGETS` column.

   Notice that the horizontal pod autoscaler scales up the deployment to two replicas, to conform with the minimum number of pods that you configured.

```
[student@workstation ~]$ watch oc get hpa loadtest
Every 2.0s: oc get hpa loadtest            workstation: Fri Mar  3 06:26:24 2023

NAME       REFERENCE             TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
loadtest   Deployment/loadtest   0%/50%    2         20        2          52s
```

   Leave the command running, and do not interrupt it.

7. Increase the CPU usage by sending requests to the `loadtest` application API.

   Use the `oc get route` command to retrieve the URL of the application.

```
[student@workstation ~]$ oc get route loadtest
NAME       HOST/PORT                                           ...
loadtest   loadtest-reliability-autoscaling.apps.ocp4.example.com   ...
```

   Send a request to the application API to simulate additional CPU pressure on the container. Do not wait for the `curl` command to complete, and continue with the exercise. After one minute, the command reports a timeout error that you can ignore.

```
[student@workstation ~]$ curl \
loadtest-reliability-autoscaling.apps.ocp4.example.com/api/loadtest/v1/cpu/1
<html><body><h1>504 Gateway Time-out</h1>
The server didn't respond in time.
</body></html>
```

   Watch the output of the `oc get hpa loadtest` command in the second terminal. After a minute, the horizontal pod autoscaler detects an increase in the CPU usage and deploys additional pods.

   NOTE

   The increased activity of the application does not immediately trigger the autoscaler. Wait a few moments if you do not see any changes to the number of replicas.

   You might need to run the `curl` command multiple times before the application uses enough CPU to trigger the autoscaler.

   The CPU usage and the number of replicas on your system probably differ.

```
Every 2.0s: oc get hpa loadtest            workstation: Fri Mar  3 07:20:19 2023

NAME      REFERENCE             TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
loadtest  Deployment/loadtest   220%/50%   2         20        9          16m
```

   Wait five minutes after the `curl` command completes. The `oc get hpa loadtest` command shows that the CPU load decreases.

   NOTE

   Although the horizontal pod autoscaler resource can be quick to scale up, it is slower to scale down.

```
Every 2.0s: oc get hpa loadtest            workstation: Fri Mar  3 07:23:11 2023

NAME      REFERENCE             TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
loadtest  Deployment/loadtest   0%/50%    2         20        9          18m
```

Optional: Wait for the `loadtest` application to scale down. It takes five additional minutes for the horizontal pod autoscaler to scale down to two replicas.

```
Every 2.0s: oc get hpa loadtest            workstation: Fri Mar  3 07:29:12 2023

NAME      REFERENCE             TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
loadtest  Deployment/loadtest   0%/50%    2         20        2          24m
```

Press **Ctrl**+**C** to quit the `watch` command. Close that second terminal when done.

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 reliability-autoscaling
```