Guided Exercise: Reserving Compute Capacity for Applications

Configure an application with compute resource requests that allow and prevent successful scheduling and scaling of its pods.

Outcomes

- Inspect how memory resource requests allocate cluster node memory.
- Explore how adjusting resource requests impacts the number of replicas that the cluster can schedule on a node.

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

This command ensures that the following conditions are true:

- The `reliability-requests` project exists.
- The resource files are available in the course directory.
- The classroom registry has the `registry.ocp4.example.com:8443/redhattraining/long-load:v1` container image.

The `registry.ocp4.example.com:8443/redhattraining/long-load:v1` container image contains an application with utility endpoints. These endpoints perform such tasks as crashing the process and toggling the server's health status.

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

Instructions

1. As the `admin` user, deploy the `long-load` application by applying the `long-load-deploy.yaml` file in the `reliability-requests` project.

   Log in to the OpenShift cluster as the `admin` user with `redhatocp` as the password.

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

   > **NOTE**
   >
   > In general, use accounts with the least required privileges to perform a task. In the classroom environment, this account is the `developer` user. However, you need cluster administrator privileges to view the cluster node metrics in this exercise.

   View the total memory request allocation for the node.

   ```
   [student@workstation ~]$ oc describe node master01
   ...output omitted...
   Allocated resources:
     (Total limits may be over 100 percent, i.e., overcommitted.)
     Resource           Requests       Limits
     --------           --------       ------
     cpu                3158m (42%)    980m (13%)
     memory             12667Mi (66%)  1250Mi (6%)
   ...output omitted...
   ```

   The command output shows that the pods that are currently running on the node requested a total of 12667 MiB of memory. That value might be slightly different on your system.

   > **IMPORTANT**
   >
   > Projects and objects from previous exercises can cause the memory usage from this exercise to mismatch the intended results. Delete any unrelated projects before continuing.
   >
   > If you still experience issues, then re-create your classroom environment and try this exercise again.

   Select the `reliability-requests` project.

   ```
   [student@workstation ~]$ oc project reliability-requests
   Now using project "reliability-requests" on server "https://api.ocp4.example.com:6443".
   ```

Go to the `~/DO180/labs/reliability-requests` directory. Create a deployment, a service, and a route by using the `oc apply` command and the `long-load-deploy.yaml` file.

```
[student@workstation ~]$ cd DO180/labs/reliability-requests
[student@workstation reliability-requests]$ oc apply -f long-load-deploy.yaml
deployment.apps/long-load created
service/long-load created
route.route.openshift.io/long-load created
```

2. Add a resource request to the pod definition and scale the deployment beyond the cluster's capacity.

   Modify the `long-load-deploy.yaml` file by adding a resource request. The request allocates one gibibyte (1 GiB) to each of the application pods.

```
spec:
  ...output omitted...
  template:
  ...output omitted...
    spec:
      containers:
      - image: registry.ocp4.example.com:8443/redhattraining/long-load:v1
        resources:
          requests:
            memory: 1Gi
...output omitted...
```

   Apply the YAML file to modify the deployment with the resource request.

```
[student@workstation reliability-requests]$ oc apply -f long-load-deploy.yaml
deployment.apps/long-load configured
service/long-load unchanged
route.route.openshift.io/long-load unchanged
```

   Scale the deployment to have 10 replicas.

```
[student@workstation reliability-requests]$ oc scale deploy/long-load \
  --replicas 10
deployment.apps/long-load scaled
```

   Observe that the cluster cannot schedule all pods on the single node. The cluster cannot schedule pods with the `Pending` status.

```
[student@workstation reliability-requests]$ oc get pods
NAME                         READY   STATUS    RESTARTS   AGE
...output omitted...
long-load-86bb4b79f8-44zwd   0/1     Pending   0          58s
...output omitted...
```

   Retrieve the cluster event log, and observe that insufficient memory is the cause of the failed scheduling.

```
[student@workstation reliability-requests]$ oc get events \
  --field-selector reason="FailedScheduling"
... pod/long-load-...   0/1 nodes are available: 1 Insufficient memory. ...
```

   Alternatively, view the events for a pending pod to see the reason. In the following command, replace the pod name with one of the pending pods in your classroom.

```
[student@workstation reliability-requests]$ oc describe \
  pod/long-load-86bb4b79f8-44zwd
...output omitted...
Events:
...output omitted...  0/1 nodes are available: 1 Insufficient memory. ...
```

   Observe that the node's requested memory usage is high.

```
[student@workstation reliability-requests]$ oc describe node master01
...output omitted...
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests          Limits
  --------           --------          ------
  cpu                3158m (42%)       980m (13%)
  memory             18811Mi (99%)     1250Mi (6%)
...output omitted...
```

The command output shows that the pods from the `long-load` deployment requested most of the remaining memory from the node. However, not enough memory is available to accommodate the 10 replicas.

3. Reduce the requested memory per pod so that the replicas can run on the node.

   Manually set the resource request to `250Mi`.

```
[student@workstation reliability-requests]$ oc set resources deploy/long-load \
  --requests memory=150Mi
deployment.apps/long-load resource requirements updated
```

   Delete the pods so that the deployment controller re-creates the pods with the new resource request.

```
[student@workstation reliability-requests]$ oc delete pod -l app=long-load
pod "long-load-557b4d94f5-29brx" deleted
...output omitted...
```

   Observe that all pods can start with the lowered memory request. Within one minute, the cluster marks the pods as `Ready` and places them in the `Running` state, with no pods in the `Pending` status.

```
[student@workstation reliability-requests]$ oc get pods
NAME                         READY   STATUS    RESTARTS   AGE
long-load-557b4d94f5-68hbb   1/1     Running   0          3m14s
long-load-557b4d94f5-bfk7c   1/1     Running   0          3m21s
long-load-557b4d94f5-bnpzh   1/1     Running   0          3m21s
long-load-557b4d94f5-chtv9   1/1     Running   0          3m21s
long-load-557b4d94f5-drg2p   1/1     Running   0          3m14s
long-load-557b4d94f5-hwsz6   1/1     Running   0          3m12s
long-load-557b4d94f5-k5vqj   1/1     Running   0          3m21s
long-load-557b4d94f5-lgstq   1/1     Running   0          3m21s
long-load-557b4d94f5-r8hq4   1/1     Running   0          3m21s
long-load-557b4d94f5-xrg7c   1/1     Running   0          3m21s
```

   Observe that the memory usage of the node is lower.

```
[student@workstation reliability-requests]$ oc describe node master01
...output omitted...
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests           Limits
  --------           --------           ------
  cpu                3158m (42%)        980m (13%)
  memory             15167Mi (80%)      1250Mi (6%)
...output omitted...
```

   Return to the `/home/student/` directory.

```
[student@workstation reliability-requests]$ cd /home/student/
[student@workstation ~]$
```

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-requests
```