Cluster resource quotas

Before continuing, make sure you've done the LDAP lab.

In a previous lab, you worked with quotas and saw how they could be applied to projects. You also set up default quotas, so anytime someone requests a new project; they get assigned the default quota. These project quotas are great for maintaining control over the resources in your cluster.

But what if you want to apply a quota not to individual projects, but accross projects?

Use cases

There are two primary usecases where you would use `clusterresourcequota` instead of a project based `quota`. One of which, is when you want to set quotas on a specific user. This is useful when you want users to create as much projects as they need (thus achiving great multitenancy), but you want to limit the amount of resource they can consume.

The other use case is if you want to set a quota by application vertical. In this case, you want to set the quota on an application stack wholistically; as an application stack can span multiple OpenShift Projects.

In this lab, we will be exploring both use cases.

Setting quota per user

In order to set a `clusterresourcequota` to a user you need to be `kubeadmin`

```
oc login -u kubeadmin -p {{ KUBEADMIN_PASSWORD }}
```

Now, set a quota for the `normaluser1`. We will be using the `annotation` key of `openshift.io/requester=` to identify the projects that will have these quotas assigned to. For this exercise, we will set a hard quota against creating more than 10 pods.

```
oc create clusterquota for-user-normaluser1 \
    --project-annotation-selector openshift.io/requester=normaluser1 \
    --hard pods=10
```

The syntax is `openshift.io/requester=<username>`.

View the configuration.

```
oc get clusterresourcequotas for-user-normaluser1 -o yaml
```

The configuration should look something like this:

```
piVersion: quota.openshift.io/v1
kind: ClusterResourceQuota
metadata:
  creationTimestamp: "2020-12-02T22:48:38Z"
  generation: 1
  managedFields:
  - apiVersion: quota.openshift.io/v1
    fieldsType: FieldsV1
    fieldsV1:
      f:spec:
        .: {}
        f:quota:
          .: {}
          f:hard:
            .: {}
            f:pods: {}
        f:selector:
          .: {}
          f:annotations:
            .: {}
            f:openshift.io/requester: {}
          f:labels: {}
      f:status:
        .: {}
        f:namespaces: {}
        f:total: {}
    manager: oc
    operation: Update
    time: "2020-12-02T22:48:38Z"
  name: for-user-normaluser1
  resourceVersion: "55396"
  selfLink: /apis/quota.openshift.io/v1/clusterresourcequotas/for-user-normaluser1
  uid: 21670296-e0af-4572-8141-832217531cc3
spec:
  quota:
    hard:
      pods: "10"
  selector:
    annotations:
      openshift.io/requester: normaluser1
    labels: null
```

This user, `normaluser1`, can create no more than 10 pods accross all the projects he creates. This applies only to projects that he as created (based on the `openshift.io/requester: normaluser1` annotation), not any projects he has access to. More on this later.

Now, login as `normaluser1`

```
oc login -u normaluser1 -p Op#nSh1ft
```

List all your current projects

```
oc get projects
```

This user shouldn't have any projects, and you should see output similar to this (don't worry if you do though):

```
No resources found.
```

Create two projects `welcome1` and `welcome2`.

```
oc new-project welcome1
oc new-project welcome2
```

You'll be creating two applications. One in the `welcome1` project and the other in the `welcome2` project.

```
oc new-app -n welcome1 --name=php1 quay.io/redhatworkshops/welcome-php:latest
oc new-app -n welcome2 --name=php2 quay.io/redhatworkshops/welcome-php:latest
```

After the deployment, you should have two running pods. One in each namespace. Check it with the`oc get pods` command (You may have to run this a few times before you see any output):

```
oc get pods -n welcome1 -l deployment=php1
oc get pods -n welcome2 -l deployment=php2
```

The output should look something like this:

```
[~] $ oc get pods -n welcome1 -l deployment=php1
NAME           READY   STATUS    RESTARTS   AGE
php1-1-nww4m   1/1     Running   0          4m20s
[~] $ oc get pods -n welcome2 -l deployment=php2
NAME           READY   STATUS    RESTARTS   AGE
php2-1-ljw9w   1/1     Running   0          4m20s
```

Now we can check the quota by first becoming `kubeadmin`:

```
oc login -u kubeadmin -p {{ KUBEADMIN_PASSWORD }}
```

Now run `oc describe clusterresourcequotas for-user-normaluser1` to see the status of the quota:

```
oc describe clusterresourcequotas for-user-normaluser1
```

You should see the following output:

```
Name:  for-user-normaluser1
Created: 22 minutes ago
Labels:  <none>
Annotations: <none>
Namespace Selector: ["welcome1" "welcome2"]
Label Selector:
AnnotationSelector: map[openshift.io/requester:normaluser1]
Resource Used Hard
-------- ---- ----
pods  2 10
```

You see that not only that 2 out of 10 pods are being used, but that the namespaces the quota is being applied to. Check the namespace manifest for `welcome1` to see the annotation the quota is looking for:

```
oc get ns welcome1 -o yaml
```

The output should look something like this. Take special note of the annotations:

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    openshift.io/description: ""
    openshift.io/display-name: ""
    openshift.io/requester: normaluser1
    openshift.io/sa.scc.mcs: s0:c26,c5
    openshift.io/sa.scc.supplemental-groups: 1000660000/10000
    openshift.io/sa.scc.uid-range: 1000660000/10000
  creationTimestamp: "2020-12-02T22:49:46Z"
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          f:openshift.io/sa.scc.mcs: {}
          f:openshift.io/sa.scc.supplemental-groups: {}
          f:openshift.io/sa.scc.uid-range: {}
    manager: cluster-policy-controller
    operation: Update
    time: "2020-12-02T22:49:46Z"
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          .: {}
          f:openshift.io/description: {}
          f:openshift.io/display-name: {}
          f:openshift.io/requester: {}
      f:status:
        f:phase: {}
    manager: openshift-apiserver
    operation: Update
    time: "2020-12-02T22:49:46Z"
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:spec:
        f:finalizers: {}
    manager: openshift-controller-manager
    operation: Update
    time: "2020-12-02T22:49:46Z"
  name: welcome1
  resourceVersion: "55712"
  selfLink: /api/v1/namespaces/welcome1
  uid: fe1ceda9-51aa-4222-b47b-e25181291f5e
spec:
  finalizers:
  - kubernetes
status:
  phase: Active
```

Now as `normaluser1`, try to scale your apps beyond 10 pods:

```
oc login -u normaluser1 -p Op#nSh1ft
oc scale deploy/php1 -n welcome1 --replicas=5
oc scale deploy/php2 -n welcome2 --replicas=6
```

Take a note of how many pods are running:

```
oc get pods --no-headers -n welcome1 -l deployment=php1 | wc -l
oc get pods --no-headers -n welcome2 -l deployment=php2 | wc -l
```

Both of these commands should return no more than 10 added up together. Check the events to see the quota in action!

```
oc get events -n welcome1 | grep "quota" | head -1
oc get events -n welcome2 | grep "quota" | head -1
```

You should see a message like the following.

```
3m24s       Warning   FailedCreate        replicaset/php1-89fcb8d8b    Error creating: pods "php1-89fcb8d8b-spdw2" is forbid
den: exceeded quota: for-user-normaluser1, requested: pods=1, used: pods=10, limited: pods=10
```

To see the status, switch to the `kubeadmin` account and run the `describe` command from before:

```
oc login -u kubeadmin -p {{ KUBEADMIN_PASSWORD }}
oc describe clusterresourcequotas for-user-normaluser1
```

You should see that the hard pod limit has been reached

```
Name:           for-user-normaluser1
Created:        15 minutes ago
Labels:         <none>
Annotations:    <none>
Namespace Selector: ["welcome1" "welcome2"]
Label Selector:
AnnotationSelector: map[openshift.io/requester:normaluser1]
Resource        Used    Hard
--------        ----    ----
pods            10      10
```

Setting quota by label

In order to set a quota by application stacks that may span multiple projects, you'll have to use labels to identify the project. First, make sure you're `kubeadmin`

```
oc login -u kubeadmin -p {{ KUBEADMIN_PASSWORD }}
```

Now set a quota based on a label. For this lab we will use `appstack=pricelist` key/value based label to identify projects.

```
oc create clusterresourcequota for-pricelist \
    --project-label-selector=appstack=pricelist \
    --hard=pods=5
```

Now create two projects:

```
oc adm new-project pricelist-frontend
oc adm new-project pricelist-backend
```

Assign the `edit` role to the user `normaluser1` for these two projects:

```
oc adm policy add-role-to-user edit normaluser1 -n pricelist-frontend
oc adm policy add-role-to-user edit normaluser1 -n pricelist-backend
```

To identify these two projects to belonging to the `pricelist` application stack, you will need to label the corresponding namespace:

```
oc label ns pricelist-frontend appstack=pricelist
oc label ns pricelist-backend appstack=pricelist
```

Run the `oc describe` command for the `for-pricelist` cluster resource quota:

```
oc describe clusterresourcequotas for-pricelist
```

You should see that both of the projects are now being tracked:

```
Name:           for-pricelist
Created:        21 seconds ago
Labels:         <none>
Annotations:    <none>
Namespace Selector: ["pricelist-frontend" "pricelist-backend"]
Label Selector: appstack=pricelist
AnnotationSelector: map[]
Resource        Used    Hard
--------        ----    ----
pods            0       5
```

Login as `normaluser1` and create the applications in their respective projects:

```
oc login -u normaluser1 -p Op#nSh1ft
oc new-app -n pricelist-frontend --name frontend quay.io/redhatworkshops/pricelist:frontend
oc new-app -n pricelist-backend --name backend quay.io/redhatworkshops/pricelist:backend
```

Check the status of the quota by logging in as `kubeadmin` and running the `describe` command:

```
oc login -u kubeadmin -p {{ KUBEADMIN_PASSWORD }}
oc describe clusterresourcequotas for-pricelist
```

You should see that 2 out of 5 pods are being used against this quota:

```
Name:           for-pricelist
Created:        About a minute ago
Labels:         <none>
Annotations:    <none>
Namespace Selector: ["pricelist-frontend" "pricelist-backend"]
Label Selector: appstack=pricelist
AnnotationSelector: map[]
Resource        Used    Hard
--------        ----    ----
pods            2       5
```

The user `normaluser1` can create more pods because `pricelist-frontend` and `pricelist-backend` were assigned to the user by `kubeadmin`. They don't have the `openshift.io/requester=normaluser1` annotation since `normaluser1` didn't create them. You can already see how you can mix and match quota polices to fit your envrionment.

Test this by logging back in as `normaluser1` and try to scale the applications beyond 5 pods total.

```
oc login -u normaluser1 -p Op#nSh1ft
oc scale -n pricelist-frontend deploy/frontend --replicas=3
oc scale -n pricelist-backend deploy/backend --replicas=3
```

Just like before, you should see an error about not being able to scale:

```
oc get events -n pricelist-frontend | grep "quota" | head -1
oc get events -n pricelist-backend | grep "quota" | head -1
```

The output should be like the other exercise:

```
39s         Warning   FailedCreate        replicaset/backend-577cf89b68   Error creating: pods "backend-577cf89b68-15svw" is
 forbidden: exceeded quota: for-pricelist, requested: pods=1, used: pods=5, limited: pods=5
```

Clean Up

Clean up the work you did by first becoming `kubeadmin`:

```
oc login -u kubeadmin -p {{ KUBEADMIN_PASSWORD }}
```

These quotas may interfere with other labs; so delete both of the `clusterresourcequota` we created in this lab:

```
oc delete clusterresourcequotas for-pricelist for-user-normaluser1
```

Also delete the projects we created for this lab:

```
oc delete projects pricelist-backend pricelist-frontend welcome1 welcome2
```

Make sure you login as `kubeadmin` in an existing project for the next lab.

```
oc login -u kubeadmin -p {{ KUBEADMIN_PASSWORD }}
oc project default
```

Last updated 2020-12-09 11:28:42 +0100