

# Lab Exercise 10- Implementing Resource Quota in Kubernetes

## Objective:

In Kubernetes, Resource Quotas are used to control the resource consumption of namespaces. They help in managing and enforcing limits on the usage of resources like CPU, memory, and the number of objects (e.g., Pods, Services) within a namespace. This exercise will guide you through creating and managing Resource Quotas to limit the resources used by applications in a specific namespace.

## Step 1: Understand Resource Quotas

Resource Quotas allow you to:

- Limit the amount of CPU and memory a namespace can use.
- Control the number of certain types of resources (e.g., Pods, Services, PersistentVolumeClaims) in a namespace.
- Prevent a namespace from consuming more resources than allocated, ensuring fair usage across multiple teams or applications.

## Step 2: Create a Namespace

First, create a namespace where you will apply the Resource Quota. This helps in isolating and controlling resource usage within that specific namespace.

Create a YAML file named ***quota-namespace.yaml*** with the following content:

```
apiVersion: v1
kind: Namespace
metadata:
  name: quota-example    # The name of the namespace.
```

Apply the YAML to create the namespace:

```
kubectl apply -f quota.yaml
```

Verify that the namespace is created:

```
kubectl get namespaces
```

```
vagrant@controlplane:~$ kubectl apply -f quota.yaml
namespace/quota-example created
vagrant@controlplane:~$ kubectl get ns
NAME                STATUS   AGE
default             Active   7d19h
kube-node-lease     Active   7d19h
kube-public         Active   7d19h
kube-system         Active   7d19h
kubernetes-dashboard Active   7d19h
quota-example       Active   22s
vagrant@controlplane:~$ apiVersion: v1
kind: Namespace
metadata:
  name: quota-example    # The name of the namespace.
```

You should see quota-example listed in the output.

### Step 3: Define a Resource Quota

Next, create a Resource Quota YAML file named ***resource-quota.yaml*** with the following content:

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: example-quota # The name of the Resource Quota.
  namespace: quota-example # The namespace to which the Resource Quota will apply.
spec:
  hard:
    # The hard limits imposed by this Resource Quota.
    requests.cpu: "2" # The total CPU resource requests allowed in the namespace (2
cores).
    requests.memory: "4Gi" # The total memory resource requests allowed in the namespace
(4 GiB).
    limits.cpu: "4" # The total CPU resource limits allowed in the namespace (4 cores).
    limits.memory: "8Gi" # The total memory resource limits allowed in the namespace (8
GiB).
    pods: "10" # The total number of Pods allowed in the namespace.
    persistentvolumeclaims: "5" # The total number of PersistentVolumeClaims allowed in the
namespace.
    configmaps: "10" # The total number of ConfigMaps allowed in the namespace.
    services: "5" # The total number of Services allowed in the namespace.
```

### Step 4: Apply the Resource Quota

Apply the Resource Quota YAML to the namespace:

```
kubectl apply -f quota.yaml
```

Verify that the Resource Quota is applied:

```
kubectl get resourcequota -n quota-example
```

```
vagrant@controlplane:~$ kubectl get resourcequotas
No resources found in my-ns namespace.

vagrant@controlplane:~$ kubectl get resourcequotas -n quota-example
NAME          AGE   REQUEST
example-quota 39s   configmaps: 1/10, persistentvolumeclaims: 0/5, pods: 0/10, requests.cpu: 0/2, requests.memory: 0/4Gi, services: 0/5
vagrant@controlplane:~$
```

Step 4: Apply the Resource Quota

Apply the Resource Quota YAML to the namespace:

```
kubectl apply -f resource-quota.yaml
```

LIMIT

limits.cpu: 0/4, limits.memory: 0/8Gi

To see the details of the applied Resource Quota:

```
kubectl describe resourcequota example-quota -n quota-example
```

```
vagrant@controlplane:~$ kubectl get resourcequotas -n quota-example
NAME          AGE   REQUEST
example-quota 39s   configmaps: 1/10, persistentvolumeclaims: 0/5, pods: 0/10, requests.cpu: 0/2, requests.memory: 0/4Gi, services: 0/5
vagrant@controlplane:~$ kubectl describe resourcequota example-quota -n quota-example
Name:          example-quota
Namespace:     quota-example
Resource       Used   Hard
-----
configmaps     1     10
limits.cpu     0      4
limits.memory  0     8Gi
persistentvolumeclaims 0      5
pods           0     10
requests.cpu   0      2
requests.memory 0     4Gi
services       0      5
vagrant@controlplane:~$
```

Applied Resource Quota:

```
kubectl describe resourcequota example-quota -n quota-example
```

Step 5: Test the Resource Quota

Let's create some resources in the quota-example namespace to see how the Resource Quota affects them.

Deploy a ReplicaSet with Resource Requests and Limits

Create a YAML file named *nginx-replicaset-quota.yaml* with the following content:

## Step 5: Test the Resource Quota

Let's create some resources in the quota-example namespace to see how the Resource Quota affects them.

Deploy a ReplicaSet with Resource Requests and Limits

Create a YAML file named ***nginx-quota.yaml*** with the following content:

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
  namespace: quota-example
spec:
  replicas: 5          # Desired number of Pod replicas.
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - containerPort: 80
          resources:      # Define resource requests and limits.
            requests:
              memory: "100Mi"
              cpu: "100m"
```

```
limits:
  memory: "200Mi"
  cpu: "200m"
```

### Explanation:

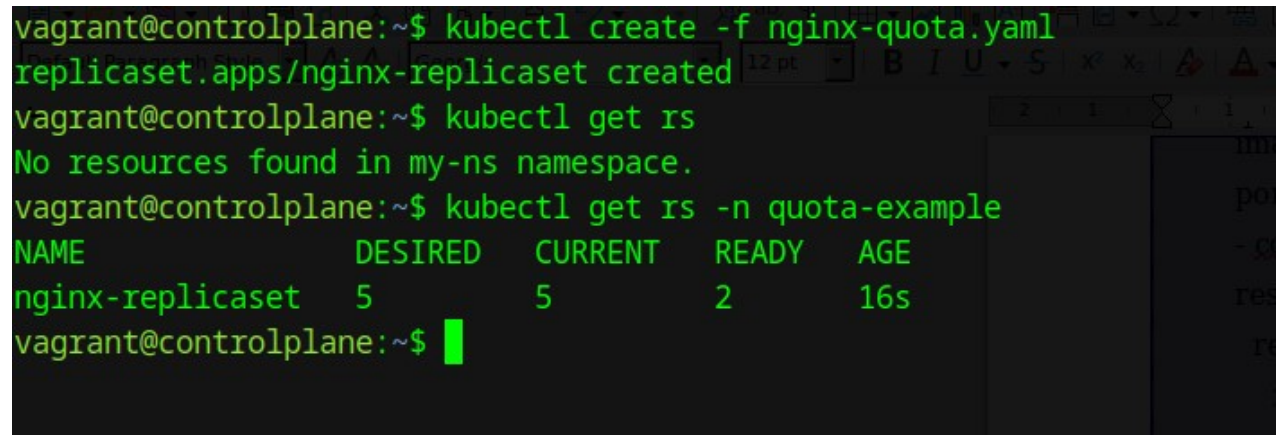
This ReplicaSet requests a total of 500m CPU and 500Mi memory across 5 replicas.

It also limits each replica to use a maximum of 200m CPU and 200Mi memory.

Apply this YAML to create the ReplicaSet:

```
kubectl apply -f nginx-quota.yaml
```

Check the status of the Pods and ensure they are created within the constraints of the Resource Quota:

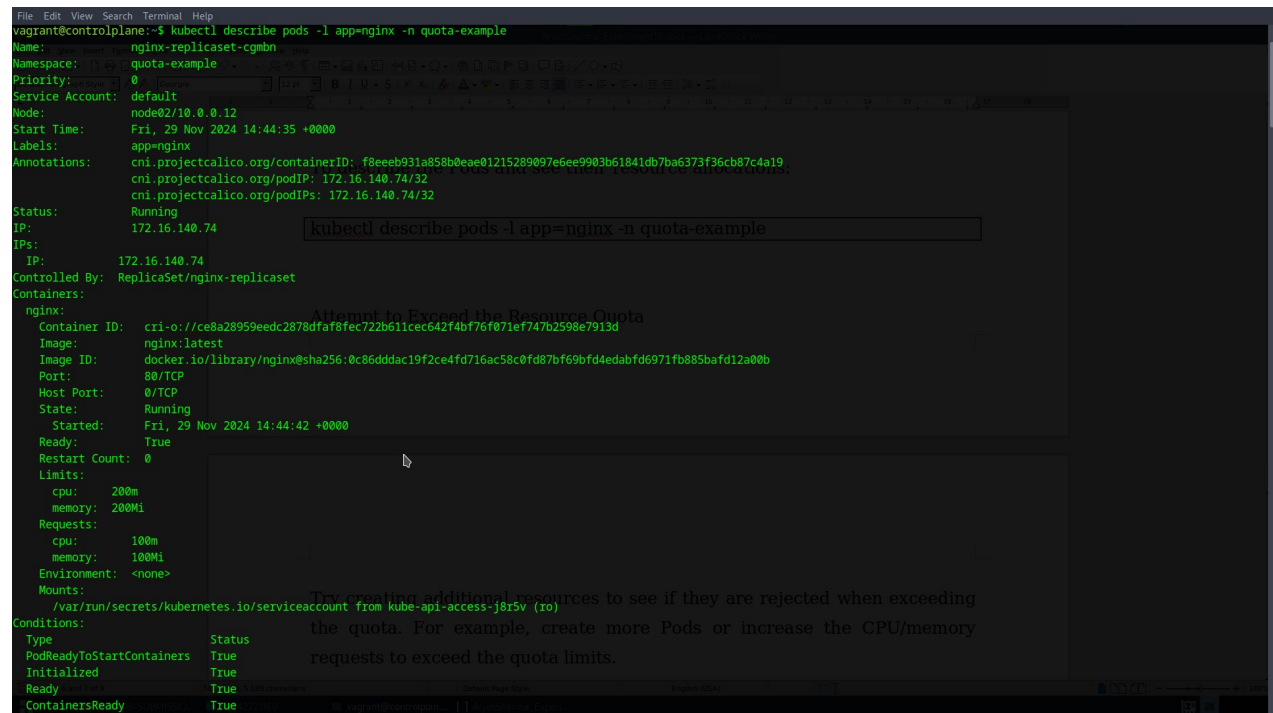


```
vagrant@controlplane:~$ kubectl create -f nginx-quota.yaml
replicaset.apps/nginx-replicaset created
vagrant@controlplane:~$ kubectl get rs
No resources found in my-ns namespace.
vagrant@controlplane:~$ kubectl get rs -n quota-example
NAME                DESIRED   CURRENT   READY   AGE
nginx-replicaset    5         5         2       16s
vagrant@controlplane:~$
```

To describe the Pods and see their resource allocations:

```
kubectl describe pods -l app=nginx -n quota-example
```

## Attempt to Exceed the Resource Quota



```
File Edit View Search Terminal Help
vagrant@controlplane:~$ kubectl describe pods -l app=nginx -n quota-example
Name:          nginx-replicaset-cgmbn
Namespace:     quota-example
Priority:       0
Service Account: default
Node:          node02/10.0.0.12
Start Time:    Fri, 29 Nov 2024 14:44:35 +0000
Labels:        app=nginx
Annotations:   cni.projectcalico.org/containerID: f8eeeb931a858b0eae01215289097e6ee9903b61841db7ba6373f36cb87c4a19
               cni.projectcalico.org/podIP: 172.16.140.74/32
               cni.projectcalico.org/podIPs: 172.16.140.74/32
Status:        Running
IP:            172.16.140.74
IPs:           IP: 172.16.140.74
Controlled By: ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:  cri-o://ce8a28959eedc2878dfaf8fec722b611cec642f4bf76f071ef747b2598e7913d
    Image:          nginx:latest
    Image ID:       docker.io/library/nginx@sha256:0c86dddc19f2ce4fd716ac58c0fd87bf69b7d4edabfd6971fb885bafd12a00b
    Port:          80/TCP
    Host Port:     0/TCP
    State:         Running
      Started:     Fri, 29 Nov 2024 14:44:42 +0000
    Ready:         True
    Restart Count: 0
    Limits:        cpu: 200m
                  memory: 200Mi
    Requests:      cpu: 100m
                  memory: 100Mi
    Environment:   <none>
    Mounts:        /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-j8r5v (ro)
Conditions:
  Type              Status
  PodReadyToStartContainers  True
  Initialized         True
  Ready               True
  ContainersReady     True
```

Try creating additional resources to see if they are rejected when exceeding the quota. For example, create more Pods or increase the CPU/memory requests to exceed the quota limits.

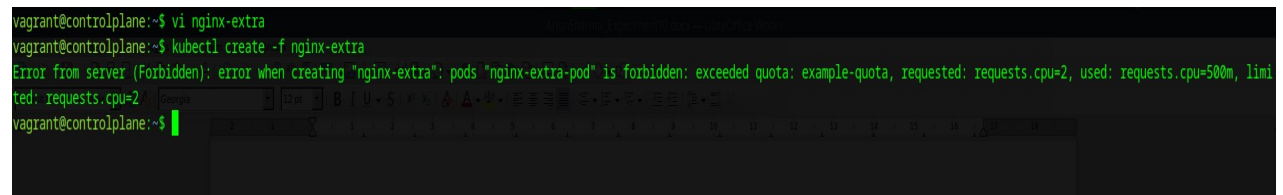
Create a YAML file named **nginx-extra.yaml** with the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-extra-pod
  namespace: quota-example
```

```
spec:
  containers:
  - name: nginx
    image: nginx:latest
  resources:
    requests:
      memory: "3Gi" # Requests a large amount of memory.
      cpu: "2"      # Requests a large amount of CPU.
    limits:
      memory: "4Gi"
      cpu: "2"
```

Apply this YAML to create the Pod:

```
kubectl apply -f nginx-extra-pod.yaml
```



```
vagrant@controlplane:~$ vi nginx-extra
vagrant@controlplane:~$ kubectl create -f nginx-extra
Error from server (Forbidden): error when creating "nginx-extra": pods "nginx-extra-pod" is forbidden: exceeded quota: example-quota, requested: requests.cpu=2, used: requests.cpu=500m, limited: requests.cpu=2
vagrant@controlplane:~$
```

This should fail due to exceeding the Resource Quota. Check the events to see the failure reason:

```
kubectl get events -n quota-example
```



```
File Edit View Search Terminal Help
vagrant@controlplane:~$ kubectl get events -n quota-example
LAST SEEN   TYPE      REASON      OBJECT                                MESSAGE
6m43s       Normal    Scheduled    pod/nginx-replicaset-cgmbn           Successfully assigned quota-example/nginx-replicaset-cgmbn to node02
6m42s       Warning   FailedMount  pod/nginx-replicaset-cgmbn           MountVolume.SetUp failed for volume "kube-api-access-j8r5v": failed to sync configmap cache: timed out waiting for the
condition
6m41s       Normal    Pulling      pod/nginx-replicaset-cgmbn           Pulling image "nginx:latest"
6m37s       Normal    Pulled       pod/nginx-replicaset-cgmbn           Successfully pulled image "nginx:latest" in 4.03s (4.03s including waiting). Image size: 195866137 bytes.
6m36s       Normal    Created      pod/nginx-replicaset-cgmbn           Created container nginx
6m36s       Normal    Started      pod/nginx-replicaset-cgmbn           Started container nginx
6m43s       Normal    Scheduled    pod/nginx-replicaset-d7jd4           Successfully assigned quota-example/nginx-replicaset-d7jd4 to node01
6m42s       Normal    Pulling      pod/nginx-replicaset-d7jd4           Pulling image "nginx:latest"
6m22s       Normal    Pulled       pod/nginx-replicaset-d7jd4           Successfully pulled image "nginx:latest" in 3.126s (20.127s including waiting). Image size: 195866137 bytes.
6m22s       Normal    Created      pod/nginx-replicaset-d7jd4           Created container nginx
6m22s       Normal    Started      pod/nginx-replicaset-d7jd4           Started container nginx
6m43s       Normal    Scheduled    pod/nginx-replicaset-kd8cz           Successfully assigned quota-example/nginx-replicaset-kd8cz to node02
6m42s       Warning   FailedMount  pod/nginx-replicaset-kd8cz           MountVolume.SetUp failed for volume "kube-api-access-hvngs": failed to sync configmap cache: timed out waiting for the
condition
6m40s       Normal    Pulling      pod/nginx-replicaset-kd8cz           Pulling image "nginx:latest"
6m33s       Normal    Pulled       pod/nginx-replicaset-kd8cz           Successfully pulled image "nginx:latest" in 3.8s (7.736s including waiting). Image size: 195866137 bytes.
6m33s       Normal    Created      pod/nginx-replicaset-kd8cz           Created container nginx
6m33s       Normal    Started      pod/nginx-replicaset-kd8cz           Started container nginx
6m43s       Normal    Scheduled    pod/nginx-replicaset-tghgv           Successfully assigned quota-example/nginx-replicaset-tghgv to node01
6m42s       Normal    Pulling      pod/nginx-replicaset-tghgv           Pulling image "nginx:latest"
6m19s       Normal    Pulled       pod/nginx-replicaset-tghgv           Successfully pulled image "nginx:latest" in 3.107s (23.134s including waiting). Image size: 195866137 bytes.
6m19s       Normal    Created      pod/nginx-replicaset-tghgv           Created container nginx
6m19s       Normal    Started      pod/nginx-replicaset-tghgv           Started container nginx
6m43s       Normal    Scheduled    pod/nginx-replicaset-wfsnc           Successfully assigned quota-example/nginx-replicaset-wfsnc to node01
6m42s       Normal    Pulling      pod/nginx-replicaset-wfsnc           Pulling image "nginx:latest"
6m25s       Normal    Pulled       pod/nginx-replicaset-wfsnc           Successfully pulled image "nginx:latest" in 17.085s (17.085s including waiting). Image size: 195866137 bytes.
6m25s       Normal    Created      pod/nginx-replicaset-wfsnc           Created container nginx
6m25s       Normal    Started      pod/nginx-replicaset-wfsnc           Started container nginx
6m44s       Normal    SuccessfulCreate  replicaset/nginx-replicaset      Created pod: nginx-replicaset-wfsnc
6m44s       Normal    SuccessfulCreate  replicaset/nginx-replicaset      Created pod: nginx-replicaset-cgmbn
6m44s       Normal    SuccessfulCreate  replicaset/nginx-replicaset      Created pod: nginx-replicaset-tghgv
6m44s       Normal    SuccessfulCreate  replicaset/nginx-replicaset      Created pod: nginx-replicaset-kd8cz
6m44s       Normal    SuccessfulCreate  replicaset/nginx-replicaset      Created pod: nginx-replicaset-d7jd4
vagrant@controlplane:~$
```

Look for error messages indicating that the Pod creation was denied due to resource constraints.

## Step 6: Clean Up Resources

To delete the resources you created:

```
vagrant@controlplane:~$ kubectl delete -f nginx-quota.yaml
kubectl delete -f nginx-extra.yaml
kubectl delete -f quota.yaml
kubectl delete namespace quota-example
replicaset.apps "nginx-replicaset" deleted
```

```
kubectl delete -f nginx-quota.yaml
kubectl delete -f nginx-extra.yaml
kubectl delete -f quota.yaml
```

```
kubectl delete namespace quota-example
```