Lab Exercise 14- 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: myns
```

Apply the YAML to create the namespace:

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

```
PS D:\Coding\ClassWork\k8s> kubectl apply -f quota-namespace.yaml
namespace/myns created
```

Verify that the namespace is created:

```
kubectl get namespaces
```

```
PS D:\Coding\ClassWork\k8s> kubectl get namespaces
NAME                   STATUS   AGE
default                Active   14m
kube-node-lease        Active   14m
kube-public            Active   14m
kube-system            Active   14m
kubernetes-dashboard   Active   13m
myns                   Active   13s
```

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: myns-quota    # The name of the Resource Quota.
  namespace: myns # 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 resource-quota.yaml
```

```
PS D:\Coding\ClassWork\k8s> kubectl apply -f resource-quota.yaml
resourcequota/myns-quota created
```

Verify that the Resource Quota is applied:

```
kubectl get resourcequota -n myns
```

```
PS D:\Coding\ClassWork\k8s> kubectl get resourcequota -n myns
NAME         REQUEST                                                                                     LIMIT
                    AGE
myns-quota   configmaps: 1/10, persistentvolumeclaims: 0/5, pods: 0/10, requests.cpu: 0/2, requests.memory: 0/4Gi, services: 0/5   limits.cpu:
 0/4, limits.memory: 0/8Gi   14s
```

To see the details of the applied Resource Quota:

kubectl describe resourcequota myns-quota -n myns

```
PS D:\Coding\ClassWork\k8s> kubectl describe resourcequota myns-quota -n myns
Name:                   myns-quota
Namespace:              myns
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
```

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:

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
  namespace: myns
spec:
  replicas: 5           # Desired number of Pod replicas.
  selector:
```

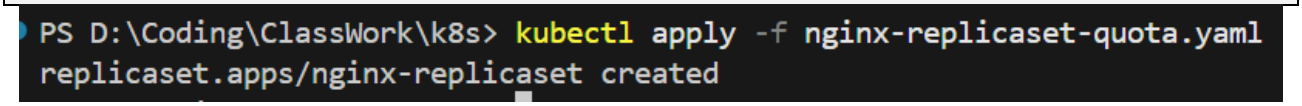
```
  matchLabels:
    app: nginx
template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
    - name: nginx
      image: nginx:latest
      ports:
      - containerPort: 80
      resources:        # Define resource requests and limits.
        requests:
          memory: "100Mi"
          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-replicaset-quota.yaml
```



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

```
kubectl get pods -n myns
```

```
PS D:\Coding\ClassWork\k8s> kubectl get pods -n myns
NAME                     READY   STATUS              RESTARTS   AGE
nginx-replicaset-8614t   0/1     ContainerCreating   0          14s
nginx-replicaset-97858   1/1     Running             0          14s
nginx-replicaset-vmxx5   0/1     ContainerCreating   0          14s
nginx-replicaset-vqjtz   1/1     Running             0          14s
nginx-replicaset-zmrsx   1/1     Running             0          14s
```

To describe the Pods and see their resource allocations:

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

```
PS D:\Coding\ClassWork\k8s> kubectl describe pods -l app=nginx -n myns
Name:             nginx-replicaset-8614t
Namespace:        myns
Priority:         0
Service Account:  default
Node:             minikube/172.28.227.29
Start Time:       Mon, 23 Feb 2026 10:44:20 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.9
IPs:
  IP:           10.244.0.9
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://bcae1799b4103d4ad42dc4da2c1dbb5cd4609886a07a2319129d2e7d6ab72989
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:341bf0f3ce6c5277d6002cf6e1fb0319fa4252add24ab6a0e262e0056d313208
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Mon, 23 Feb 2026 10:44:35 +0530
```

Attempt to Exceed the Resource Quota

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-pod.yaml** with the following content:

```
apiVersion: v1
```

```
kind: Pod
metadata:
  name: nginx-extra-pod
  namespace: myns
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
```

```
PS D:\Coding\ClassWork\k8s> kubectl apply -f nginx-extra-pod.yaml
Error from server (Forbidden): error when creating "nginx-extra-pod.yaml": pods "nginx-extra-pod" is forbidden: exceeded quota: myns-quota, re
quested: requests.cpu=2, used: requests.cpu=500m, limited: requests.cpu=2
```

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

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

```
PS D:\Coding\ClassWork\k8s> kubectl get events -n myns
LAST SEEN   TYPE     REASON            OBJECT                      MESSAGE
6m48s       Normal   Scheduled         pod/nginx-replicaset-8614t   Successfully assigned myns/nginx-replicaset-8614t to minikube
6m48s       Normal   Pulling           pod/nginx-replicaset-8614t   Pulling image "nginx:latest"
6m33s       Normal   Pulled            pod/nginx-replicaset-8614t   Successfully pulled image "nginx:latest" in 3.588s (15.187s including wa
iting). Image size: 160850673 bytes.
6m33s       Normal   Created           pod/nginx-replicaset-8614t   Container created
6m33s       Normal   Started           pod/nginx-replicaset-8614t   Container started
6m48s       Normal   Scheduled         pod/nginx-replicaset-97858   Successfully assigned myns/nginx-replicaset-97858 to minikube
6m48s       Normal   Pulling           pod/nginx-replicaset-97858   Pulling image "nginx:latest"
6m44s       Normal   Pulled            pod/nginx-replicaset-97858   Successfully pulled image "nginx:latest" in 3.993s (3.993s including wai
ting). Image size: 160850673 bytes.
6m44s       Normal   Created           pod/nginx-replicaset-97858   Container created
6m44s       Normal   Started           pod/nginx-replicaset-97858   Container started
6m48s       Normal   Scheduled         pod/nginx-replicaset-vmxx5   Successfully assigned myns/nginx-replicaset-vmxx5 to minikube
6m48s       Normal   Pulling           pod/nginx-replicaset-vmxx5   Pulling image "nginx:latest"
6m29s       Normal   Pulled            pod/nginx-replicaset-vmxx5   Successfully pulled image "nginx:latest" in 3.615s (18.792s including wa
iting). Image size: 160850673 bytes.
6m29s       Normal   Created           pod/nginx-replicaset-vmxx5   Container created
6m29s       Normal   Started           pod/nginx-replicaset-vmxx5   Container started
6m48s       Normal   Scheduled         pod/nginx-replicaset-vqjtz   Successfully assigned myns/nginx-replicaset-vqjtz to minikube
6m48s       Normal   Pulling           pod/nginx-replicaset-vqjtz   Pulling image "nginx:latest"
6m40s       Normal   Pulled            pod/nginx-replicaset-vqjtz   Successfully pulled image "nginx:latest" in 3.929s (7.826s including wai
ting). Image size: 160850673 bytes.
6m40s       Normal   Created           pod/nginx-replicaset-vqjtz   Container created
6m40s       Normal   Started           pod/nginx-replicaset-vqjtz   Container started
6m48s       Normal   Scheduled         pod/nginx-replicaset-zmrsx   Successfully assigned myns/nginx-replicaset-zmrsx to minikube
6m48s       Normal   Pulling           pod/nginx-replicaset-zmrsx   Pulling image "nginx:latest"
6m36s       Normal   Pulled            pod/nginx-replicaset-zmrsx   Successfully pulled image "nginx:latest" in 3.786s (11.599s including wa
iting). Image size: 160850673 bytes.
6m36s       Normal   Created           pod/nginx-replicaset-zmrsx   Container created
6m36s       Normal   Started           pod/nginx-replicaset-zmrsx   Container started
6m48s       Normal   SuccessfulCreate  replicaset/nginx-replicaset  Created pod: nginx-replicaset-97858
6m48s       Normal   SuccessfulCreate  replicaset/nginx-replicaset  Created pod: nginx-replicaset-vqjtz
6m48s       Normal   SuccessfulCreate  replicaset/nginx-replicaset  Created pod: nginx-replicaset-zmrsx
6m48s       Normal   SuccessfulCreate  replicaset/nginx-replicaset  Created pod: nginx-replicaset-vmxx5
6m48s       Normal   SuccessfulCreate  replicaset/nginx-replicaset  Created pod: nginx-replicaset-8614t
```

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:

kubectl delete -f nginx-replicaset-quota.yaml

kubectl delete -f nginx-extra-pod.yaml

kubectl delete -f resource-quota.yaml

kubectl delete namespace myns

```
PS D:\Coding\ClassWork\k8s> kubectl delete -f nginx-replicaset-quota.yaml
replicaset.apps "nginx-replicaset" deleted from myns namespace
PS D:\Coding\ClassWork\k8s> kubectl delete -f nginx-extra-pod.yaml
Error from server (NotFound): error when deleting "nginx-extra-pod.yaml": pods "nginx-extra-pod" not found
PS D:\Coding\ClassWork\k8s> kubectl delete -f resource-quota.yaml
resourcequota "myns-quota" deleted from myns namespace
PS D:\Coding\ClassWork\k8s> kubectl delete namespace myns
namespace "myns" deleted
```