IMPLEMENTING RESOURCE QUOTA IN KUBERNETES

Implementing Resource Quota in Kubernetes

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.

```
controlplane ~ ➜ kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
controlplane   Ready    control-plane   43m   v1.33.0
node01         Ready    <none>          42m   v1.33.0
node02         Ready    <none>          42m   v1.33.0
```

Step 1: 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 namespace_quota.yaml and apply the YAML to create the namespace:

```
controlplane ~ ➜ vi namespace_quota.yaml

controlplane ~ ➜ kubectl apply -f namespace_quota.yaml
namespace/mynamespace created

controlplane ~ ➜ cat namespace_quota.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: mynamespace
```

Verify that the namespace is created:

```
controlplane ~ ➜ kubectl get namespaces
NAME              STATUS   AGE
default           Active   46m
kube-node-lease   Active   46m
kube-public       Active   46m
kube-system       Active   46m
mynamespace       Active   37s
```

Step 2: Define a Resource Quota

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

```
controlplane ~ ➜ cat resourcequota.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: myns-quota    # The name of the Resource Quota.
  namespace: mynamespace # 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.
```

Apply the Resource Quota YAML to the namespace:

```
controlplane ~ ➜ vi resourcequota.yaml

controlplane ~ ➜ kubectl apply -f resourcequota.yaml
resourcequota/myns-quota created
```

Verify that the Resource Quota is applied:

```
controlplane ~ ➜ kubectl describe resourcequota myns-quota -n mynamespace
Name:                   myns-quota
Namespace:              mynamespace
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 3: Test the Resource Quota

Let's create some resources in the mynamespace to see how the Resource Quota affects them.

Deploy a ReplicaSet with Resource Requests and Limits

Create a YAML file named **replicasetquota.yaml** with the following content:

```
controlplane ~ ✖ vi replicasetquota.yaml

controlplane ~ ➜ kubectl apply -f replicasetquota.yaml
replicaset.apps/nginx-replicaset created

controlplane ~ ➜ kubectl get pods -n mynamespace
NAME                    READY   STATUS    RESTARTS   AGE
nginx-replicaset-c7mmr  1/1     Running   0          89s
nginx-replicaset-kqxw4  1/1     Running   0          89s
nginx-replicaset-mhjwg  1/1     Running   0          89s
nginx-replicaset-p6xnj  1/1     Running   0          89s
nginx-replicaset-tps7j  1/1     Running   0          89s
```

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 **checkquotaapplying.yaml** with the following content:

```
controlplane ~ ➜ cat checkquotaapplying.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-extra-pod
  namespace: mynamespace
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"
```

```
controlplane ~ ➜ vi checkquotaapplying.yaml

controlplane ~ ➜ vi checkquotaapplying.yaml

controlplane ~ ➜ kubectl apply -f checkquotaapplying.yaml
Error from server (Forbidden): error when creating "checkquotaapplying.yaml": pods "nginx-extra-pod" is forbidden: exceeded quota: myns-quota, requested: requests.cpu=2, used: requests.cpu=500m, limited: requests.cpu=2
```