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:

```
PS C:\Users\upes\docker> notepad quota-namespace.yaml
```

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

Apply the YAML to create the namespace:

```
PS C:\Users\upes\docker> kubectl apply -f quota-namespace.yaml
namespace/quota-example created
PS C:\Users\upes\docker> kubectl get namespaces
```

kubectl apply -f quota-namespace.yaml

Verify that the namespace is created:

```
PS C:\Users\upes\docker> kubectl get namespaces
NAME              STATUS   AGE
default           Active   49m
kube-node-lease   Active   49m
kube-public       Active   49m
kube-system       Active   49m
quota-example     Active   4s
```

kubectl get namespaces

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:

```
PS C:\Users\upes\docker> notepad resource-quota.yaml
```

```
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:

```
PS C:\Users\upes\docker> kubectl apply -f resource-quota.yaml
resourcequota/example-quota created
```

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

Verify that the Resource Quota is applied:

```
PS C:\Users\upes\docker> kubectl get resourcequota -n quota-example
NAME            AGE   REQUEST                                                 LIMIT
example-quota   2s    configmaps: 1/10, persistentvolumeclaims: 0/5, pods: 0/10, requests.cp
 0/2, requests.memory: 0/4Gi, services: 0/5   limits.cpu: 0/4, limits.memory: 0/8Gi
```

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

To see the details of the applied Resource Quota:

```
PS C:\Users\upes\docker> 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
PS C:\Users\upes\docker> notepad nginx-replicaset-quota.yaml
```

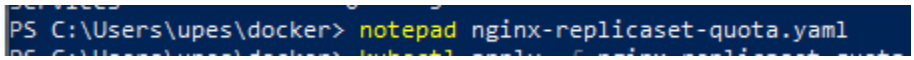
```
kubectl describe resourcequota example-quota -n quota-example
```

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:




```
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:

```
PS C:\Users\upes\docker> kubectl apply -f nginx-replicaset-quota.yaml
replicaset.apps/nginx-replicaset created
```

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:

```
PS C:\Users\upes\docker> kubectl get pods -n quota-example
NAME                    READY   STATUS              RESTARTS   AGE
nginx-replicaset-9hjk5  0/1     ContainerCreating   0          5s
nginx-replicaset-kbvl9  0/1     ContainerCreating   0          4s
nginx-replicaset-kxbpw  0/1     Pending             0          4s
nginx-replicaset-pfptj  0/1     ContainerCreating   0          5s
nginx-replicaset-xjnmv  0/1     ContainerCreating   0          5s
```

kubectl get pods -n quota-example

To describe the Pods and see their resource allocations:

```
PS C:\Users\upes\docker> kubectl describe pods -l app=nginx -n quota-example
Name:             nginx-replicaset-9hjk5
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Mon, 11 Nov 2024 13:15:35 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Controlled By:    ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:
```

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

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:

```
PS C:\Users\upes\docker> notepad nginx-extra-pod.yaml
```

```
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:

```
PS C:\Users\upes\docker> kubectl apply -f nginx-extra-pod.yaml
Error from server (Forbidden): error when creating "nginx-extra-pod.y
od" is forbidden: exceeded quota: example-quota, requested: requests.
=500m, limited: requests.cpu=2
```

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

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

```
PS C:\Users\upes\docker> kubectl get events -n quota-example
LAST SEEN   TYPE     REASON             OBJECT
52s         Normal   Scheduled          pod/nginx-replicaset-9hj
ota-example/nginx-replicaset-9hjk5 to docker-desktop
36s         Normal   Pulling            pod/nginx-replicaset-9hj
est"
29s         Normal   Pulled             pod/nginx-replicaset-9hj
e "nginx:latest" in 6.993s (6.993s including waiting). Image siz
25s         Normal   Created            pod/nginx-replicaset-9hj
24s         Normal   Started            pod/nginx-replicaset-9hj
```

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

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:

```
PS C:\Users\upes\docker> kubectl delete -f nginx-extra-pod.yaml
Error from server (NotFound): error when deleting "nginx-extra-po
d" not found
PS C:\Users\upes\docker> kubectl delete -f resource-quota.yaml
resourcequota "example-quota" deleted
PS C:\Users\upes\docker> kubectl delete namespace quota-example
namespace "quota-example" deleted
```

```
kubectl delete -f nginx-replicaset-quota.yaml
kubectl delete -f nginx-extra-pod.yaml
kubectl delete -f resource-quota.yaml
kubectl delete namespace quota-example
```