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.
```

quota-namespace.yaml

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

```

Apply the YAML to create the namespace:

kubectl apply -f quota-namespace.yaml

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl apply -f quota-namespace.yaml
namespace/quota-example created
```

Verify that the namespace is created:

kubectl get namespaces

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl get namespaces
NAME              STATUS   AGE
default           Active   21d
kube-node-lease   Active   21d
kube-public       Active   21d
kube-system       Active   21d
quota-example     Active   48s
```

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:

```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.
```

resource-quota.yaml

```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:

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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl apply -f resource-quota.yaml
resourcequota/example-quota created
```

Verify that the Resource Quota is applied:

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

```
NAME            AGE   REQUEST                                                                                      LIMIT
example-quota   48s   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
```

To see the details of the applied Resource Quota:

kubectl describe resourcequota example-quota -n quota-example

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> 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
```

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"
```

nginx-replicaset-quota.yaml nginx-replicaset-quota.yaml

```yaml
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
```

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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl apply -f nginx-replicaset-quota.yaml
replicaset.apps/nginx-replicaset created
```

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

kubectl get pods -n quota-example

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl get pods -n quota-example
NAME                    READY   STATUS    RESTARTS   AGE
nginx-replicaset-2gvkn  1/1     Running   0          28s
nginx-replicaset-fgn94  1/1     Running   0          28s
nginx-replicaset-jp6sb  1/1     Running   0          28s
nginx-replicaset-ng668  1/1     Running   0          28s
nginx-replicaset-snbnx  1/1     Running   0          28s
```

To describe the Pods and see their resource allocations:

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

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl describe pods -l app=nginx -n quota-example
Name:             nginx-replicaset-2gvkn
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Mon, 11 Nov 2024 12:02:03 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.21
IPs:
  IP:           10.1.0.21
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
```

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: 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"
```

nginx-extra-pod.yaml

```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:

kubectl apply -f nginx-extra-pod.yaml

```
PS C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> 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: example-quota, reque
sted: 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 C:\Github Repositores\CDS-LAB-SUBMISSION-2022-26\R2142221383> kubectl get events -n quota-example
LAST SEEN   TYPE     REASON           OBJECT                        MESSAGE
7m          Normal   Scheduled        pod/nginx-replicaset-2gvkn    Successfully assigned quota-example/nginx-replicaset-2gvkn to docker-desktop
6m59s       Normal   Pulling          pod/nginx-replicaset-2gvkn    Pulling image "nginx:latest"
6m38s       Normal   Pulled           pod/nginx-replicaset-2gvkn    Successfully pulled image "nginx:latest" in 5.861s (21.548s including waiting)
6m38s       Normal   Created          pod/nginx-replicaset-2gvkn    Created container nginx
6m37s       Normal   Started          pod/nginx-replicaset-2gvkn    Started container nginx
7m          Normal   Scheduled        pod/nginx-replicaset-fgn94    Successfully assigned quota-example/nginx-replicaset-fgn94 to docker-desktop
6m59s       Normal   Pulling          pod/nginx-replicaset-fgn94    Pulling image "nginx:latest"
6m52s       Normal   Pulled           pod/nginx-replicaset-fgn94    Successfully pulled image "nginx:latest" in 3.347s (7.281s including waiting)
6m52s       Normal   Created          pod/nginx-replicaset-fgn94    Created container nginx
6m52s       Normal   Started          pod/nginx-replicaset-fgn94    Started container nginx
7m          Normal   Scheduled        pod/nginx-replicaset-jp6sb    Successfully assigned quota-example/nginx-replicaset-jp6sb to docker-desktop
6m59s       Normal   Pulling          pod/nginx-replicaset-jp6sb    Pulling image "nginx:latest"
6m43s       Normal   Pulled           pod/nginx-replicaset-jp6sb    Successfully pulled image "nginx:latest" in 4.695s (15.686s including waiting)
6m43s       Normal   Created          pod/nginx-replicaset-jp6sb    Created container nginx
6m43s       Normal   Started          pod/nginx-replicaset-jp6sb    Started container nginx
7m          Normal   Scheduled        pod/nginx-replicaset-ng668    Successfully assigned quota-example/nginx-replicaset-ng668 to docker-desktop
6m59s       Normal   Pulling          pod/nginx-replicaset-ng668    Pulling image "nginx:latest"
6m48s       Normal   Pulled           pod/nginx-replicaset-ng668    Successfully pulled image "nginx:latest" in 3.727s (10.993s including waiting)
6m48s       Normal   Created          pod/nginx-replicaset-ng668    Created container nginx
6m48s       Normal   Started          pod/nginx-replicaset-ng668    Started container nginx
```

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 quota-example
```

```
Error from server (NotFound): error when deleting "nginx-extra-pod.yaml": pods "nginx-extra-pod" not found
resourcequota "example-quota" deleted
namespace "quota-example" deleted
```