Lab Exercise 10- Implementing Resource Quota in Kubernetes

Name: Aditya Tomar

SAP:500106015

Batch:2

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
1  apiVersion: v1
2  kind: Namespace
3  metadata:
4    name: quota-example    # The name of the namespace.
5
```

Apply the YAML to create the namespace:

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

```
[adityatomar@Mac Kubernetes % kubectl apply -f quota-namespace.yaml
namespace/quota-example created
```

Verify that the namespace is created:

```
kubectl get namespaces
```

```
[adityatomar@Mac Kubernetes % kubectl get namespaces
NAME              STATUS   AGE
default           Active   20d
kube-node-lease   Active   20d
kube-public       Active   20d
kube-system       Active   20d
quota-example     Active   2m20s
```

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.
```

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
```

```
adityatomar@Mac Kubernetes % kubectl apply -f resource-quota.yaml
resourcequota/example-quota created
```

Verify that the Resource Quota is applied:

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

```
adityatomar@Mac Kubernetes % kubectl get resourcequota -n quota-example
NAME            AGE   REQUEST                                                 LIMIT
example-quota   36s   configmaps: 1/10, persistentvolumeclaims: 0/5, pods: 0/10,
 requests.cpu: 0/2, requests.memory: 0/4Gi, services: 0/5   limits.cpu: 0/4, lim
its.memory: 0/8Gi
```

To see the details of the applied Resource Quota:

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

```
adityatomar@Mac Kubernetes % kubectl describe resourcequota example-quota -n quo
ta-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"
```

x-pod.yaml | nginx-service.yaml | quota-namespace.yaml | resource-quota.yaml | 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
    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
```

```
adityatomar@Mac Kubernetes % 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
```

```
adityatomar@Mac Kubernetes % kubectl get pods -n quota-example
NAME                      READY   STATUS    RESTARTS   AGE
nginx-replicaset-cllvs    1/1     Running   0          45s
nginx-replicaset-glhqf    1/1     Running   0          45s
nginx-replicaset-jppv9    1/1     Running   0          45s
nginx-replicaset-lzzw2    1/1     Running   0          45s
nginx-replicaset-p5ml8    1/1     Running   0          45s
```

To describe the Pods and see their resource allocations:

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

```
adityatomar@Mac Kubernetes % kubectl describe pods -l app=nginx -n quota-example

Name:             nginx-replicaset-cllvs
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Mon, 11 Nov 2024 13:10:46 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.16
IPs:
  IP:           10.1.0.16
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://c46d385b174ffe801a55d40c4ed1d78f2fe5cf782bbe525b496
1022a8a8d7c6e
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:28402db69fec7c17e179ea8788266
7f1e054391138f77ffaf0c3eb388efc3ffb
```

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"
```

```yaml
! 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:

kubectl apply -f nginx-extra-pod.yaml

```
adityatomar@Mac Kubernetes % 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, requested: reques
ts.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

```
adityatomar@Mac Kubernetes % kubectl get events -n quota-example
LAST SEEN   TYPE     REASON           OBJECT                        MESSAGE
8m40s       Normal   Scheduled        pod/nginx-replicaset-cllvs    Successfully assigned quota-example/nginx-replicaset-cllvs to docker-desktop
8m40s       Normal   Pulling          pod/nginx-replicaset-cllvs    Pulling image "nginx:latest"
8m38s       Normal   Pulled           pod/nginx-replicaset-cllvs    Successfully pulled image "nginx:latest" in 2.202s (2.202s including waiting). Image size: 69600252 bytes.
8m38s       Normal   Created          pod/nginx-replicaset-cllvs    Created container nginx
8m38s       Normal   Started          pod/nginx-replicaset-cllvs    Started container nginx
8m40s       Normal   Scheduled        pod/nginx-replicaset-glhqf    Successfully assigned quota-example/nginx-replicaset-glhqf to docker-desktop
8m40s       Normal   Pulling          pod/nginx-replicaset-glhqf    Pulling image "nginx:latest"
8m36s       Normal   Pulled           pod/nginx-replicaset-glhqf    Successfully pulled image "nginx:latest" in 1.398s (3.599s including waiting). Image size: 69600252 bytes.
8m36s       Normal   Created          pod/nginx-replicaset-glhqf    Created container nginx
8m36s       Normal   Started          pod/nginx-replicaset-glhqf    Started container nginx
8m40s       Normal   Scheduled        pod/nginx-replicaset-jppv9    Successfully assigned quota-example/nginx-replicaset-jppv9 to docker-desktop
8m40s       Normal   Pulling          pod/nginx-replicaset-jppv9    Pulling image "nginx:latest"
8m35s       Normal   Pulled           pod/nginx-replicaset-jppv9    Successfully pulled image "nginx:latest" in 1.381s (4.979s including waiting). Image size: 69600252 bytes.
8m35s       Normal   Created          pod/nginx-replicaset-jppv9    Created container nginx
8m35s       Normal   Started          pod/nginx-replicaset-jppv9    Started container nginx
8m40s       Normal   Scheduled        pod/nginx-replicaset-lzzw2    Successfully assigned quota-example/nginx-replicaset-lzzw2 to docker-desktop
8m40s       Normal   Pulling          pod/nginx-replicaset-lzzw2    Pulling image "nginx:latest"
8m30s       Normal   Pulled           pod/nginx-replicaset-lzzw2    Successfully pulled image "nginx:latest" in 3.929s (10.328s including waiting). Image size: 69600252 bytes.
8m30s       Normal   Created          pod/nginx-replicaset-lzzw2    Created container nginx
8m30s       Normal   Started          pod/nginx-replicaset-lzzw2    Started container nginx
8m40s       Normal   Scheduled        pod/nginx-replicaset-p5ml8    Successfully assigned quota-example/nginx-replicaset-p5ml8 to docker-desktop
```

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
```

```
adityatomar@Mac Kubernetes % 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
replicaset.apps "nginx-replicaset" deleted
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
```