Lab Exercise 10- Implementing Resource Quota in Kubernetes

Name – Sujal Bhandari
SAP_ID – 500106865
Roll_No – R2142220181

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.
```

```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:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> kubectl apply -f quota-namespace.yaml
namespace/quota-example created
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10>
```

Verify that the namespace is created:

```
kubectl get namespaces
```

```
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> kubectl get namespaces
NAME              STATUS   AGE
default           Active   27d
kube-node-lease   Active   27d
kube-public       Active   27d
kube-system       Active   27d
quota-example     Active   26s
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10>
```

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.
```

```
! quota-namespace.yaml    ! resource-quota.yaml ×
! resource-quota.yaml
1  apiVersion: v1
2  kind: ResourceQuota
3  metadata:
4    name: example-quota    # The name of the Resource Quota.
5    namespace: quota-example # The namespace to which the Resource Quota will apply.
6  spec:
7    hard:                  # The hard limits imposed by this Resource Quota.
8      requests.cpu: "2"    # The total CPU resource requests allowed in the namespace (2 cores).
9      requests.memory: "4Gi" # The total memory resource requests allowed in the namespace (4 GiB).
10     limits.cpu: "4"      # The total CPU resource limits allowed in the namespace (4 cores).
11     limits.memory: "8Gi" # The total memory resource limits allowed in the namespace (8 GiB).
12     pods: "10"           # The total number of Pods allowed in the namespace.
13     persistentvolumeclaims: "5" # The total number of PersistentVolumeClaims allowed in the namespace.
14     configmaps: "10"     # The total number of ConfigMaps allowed in the namespace.
15     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:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> kubectl apply -f resource-quota.yaml
resourcequota/example-quota created
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10>
```

Verify that the Resource Quota is applied:

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

```
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> kubectl get resourcequota -n quota-example
NAME            AGE   REQUEST                                                                                     LIMIT
example-quota   12s   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
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> S
```

To see the details of the applied Resource Quota:

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

```
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> 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\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10>
```

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:

```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"
```

```yaml
spec:
  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
```

```
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> kubectl apply -f nginx-replicaset-quota.yaml
replicaset.apps/nginx-replicaset created
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10>
```

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:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> kubectl get pods -n quota-example
NAME                      READY   STATUS              RESTARTS   AGE
nginx-replicaset-hcxjs    0/1     ContainerCreating   0          15s
nginx-replicaset-jkd96    0/1     ContainerCreating   0          15s
nginx-replicaset-r6j6r    1/1     Running             0          15s
nginx-replicaset-t578x    1/1     Running             0          15s
nginx-replicaset-wcbwc    1/1     Running             0          15s
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10>
```

To describe the Pods and see their resource allocations:

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

```
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> kubectl describe pods -l app=nginx -n quota-example
Name:             nginx-replicaset-hcxjs
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Thu, 21 Nov 2024 17:12:12 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.20
IPs:
  IP:           10.1.0.20
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://2a1b71667d5b9b9d6735c61973d67cebe8e124c78777ad4fb3a1628f0fea7b41
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 21 Nov 2024 17:12:30 +0530
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     200m
      memory:  200Mi
    Requests:
      cpu:        100m
      memory:     100Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-fktsd (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-fktsd:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   Burstable
Node-Selectors:              <none>
```

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
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:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> 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: requests.cpu=2, used: requests.
cpu=500m, limited: requests.cpu=2
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10>
```

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:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10> kubectl get events -n quota-example
LAST SEEN   TYPE     REASON            OBJECT                        MESSAGE
2m26s       Normal   Scheduled         pod/nginx-replicaset-hcxjs    Successfully assigned quota-example/nginx-replicaset-hcxjs to docker-desktop
2m26s       Normal   Pulling           pod/nginx-replicaset-hcxjs    Pulling image "nginx:latest"
2m9s        Normal   Pulled            pod/nginx-replicaset-hcxjs    Successfully pulled image "nginx:latest" in 4.644s (17.061s including waiting)
2m9s        Normal   Created           pod/nginx-replicaset-hcxjs    Created container nginx
2m9s        Normal   Started           pod/nginx-replicaset-hcxjs    Started container nginx
2m26s       Normal   Scheduled         pod/nginx-replicaset-jkd96    Successfully assigned quota-example/nginx-replicaset-jkd96 to docker-desktop
2m26s       Normal   Pulling           pod/nginx-replicaset-jkd96    Pulling image "nginx:latest"
2m5s        Normal   Pulled            pod/nginx-replicaset-jkd96    Successfully pulled image "nginx:latest" in 4.518s (21.577s including waiting)
2m5s        Normal   Created           pod/nginx-replicaset-jkd96    Created container nginx
2m5s        Normal   Started           pod/nginx-replicaset-jkd96    Started container nginx
2m26s       Normal   Scheduled         pod/nginx-replicaset-r6j6r    Successfully assigned quota-example/nginx-replicaset-r6j6r to docker-desktop
2m26s       Normal   Pulling           pod/nginx-replicaset-r6j6r    Pulling image "nginx:latest"
2m14s       Normal   Pulled            pod/nginx-replicaset-r6j6r    Successfully pulled image "nginx:latest" in 3.464s (12.417s including waiting)
2m14s       Normal   Created           pod/nginx-replicaset-r6j6r    Created container nginx
2m14s       Normal   Started           pod/nginx-replicaset-r6j6r    Started container nginx
2m26s       Normal   Scheduled         pod/nginx-replicaset-t578x    Successfully assigned quota-example/nginx-replicaset-t578x to docker-desktop
2m26s       Normal   Pulling           pod/nginx-replicaset-t578x    Pulling image "nginx:latest"
2m20s       Normal   Pulled            pod/nginx-replicaset-t578x    Successfully pulled image "nginx:latest" in 6.379s (6.379s including waiting)
2m20s       Normal   Created           pod/nginx-replicaset-t578x    Created container nginx
2m20s       Normal   Started           pod/nginx-replicaset-t578x    Started container nginx
2m26s       Normal   Scheduled         pod/nginx-replicaset-wcbwc    Successfully assigned quota-example/nginx-replicaset-wcbwc to docker-desktop
2m26s       Normal   Pulling           pod/nginx-replicaset-wcbwc    Pulling image "nginx:latest"
2m18s       Normal   Pulled            pod/nginx-replicaset-wcbwc    Successfully pulled image "nginx:latest" in 2.573s (8.953s including waiting)
2m17s       Normal   Created           pod/nginx-replicaset-wcbwc    Created container nginx
2m17s       Normal   Started           pod/nginx-replicaset-wcbwc    Started container nginx
2m27s       Normal   SuccessfulCreate  replicaset/nginx-replicaset   Created pod: nginx-replicaset-wcbwc
2m27s       Normal   SuccessfulCreate  replicaset/nginx-replicaset   Created pod: nginx-replicaset-t578x
2m27s       Normal   SuccessfulCreate  replicaset/nginx-replicaset   Created pod: nginx-replicaset-r6j6r
2m27s       Normal   SuccessfulCreate  replicaset/nginx-replicaset   Created pod: nginx-replicaset-hcxjs
2m27s       Normal   SuccessfulCreate  replicaset/nginx-replicaset   Created pod: nginx-replicaset-jkd96
PS C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10>
```

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
```

```
>> kubectl delete -f nginx-extra-pod.yamleplicaset/nginx-replicaset   Created pod: nginx-replicaset-r6j6r
>> kubectl delete -f resource-quota.yamlreplicaset/nginx-replicaset   Created pod: nginx-replicaset-hcxjs
>> kubectl delete namespace quota-exampleeplicaset/nginx-replicaset   Created pod: nginx-replicaset-jkd96
>> C:\Users\sujal\OneDrive\Desktop\Sem_5\CnD_Security_Lab\Exp10>
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
```