School of Computer Science

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES DEHRADUN, UTTARAKHAND

Containers & Docker Security

Lab File (2022-2026)
5th Semester

Submitted To:

Dr. Hitesh Kumar Sharma

Submitted By:

Akshat Pandey
(500101788)
B Tech CSE
DevOps[5th Semester]
R2142220306
Batch - 1

EXPERIMENT 10

AIM: 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.
```

File Edit View

```
apiVersion: v1
kind: Namespace
metadata:
  name: quota-example
```

Apply the YAML to create the namespace:

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

```
C:\Users\madha>kubectl apply -f quota-namespace.yaml
namespace/quota-example created
```

Verify that the namespace is created:

```
kubectl get namespaces
```

```
C:\Users\madha>kubectl get namespaces
NAME                 STATUS   AGE
default              Active   17d
kube-node-lease      Active   17d
kube-public          Active   17d
kube-system          Active   17d
quota-example        Active   39s
```

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.
```

File Edit View

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: example-quota
  namespace: quota-example
spec:
  hard:
    requests.cpu: "2"                # 2 CPUs requested in total.
    requests.memory: "4Gi"           # 4Gi memory requested in total.
    limits.cpu: "4"                  # 4 CPUs total limit.
    limits.memory: "8Gi"             # 8Gi memory limit.
    pods: "10"                       # Maximum number of Pods.
    persistentvolumeclaims: "5"      # Maximum PersistentVolumeClaims.
    configmaps: "10"                 # Maximum ConfigMaps.
    services: "5"                    # Maximum Services.
```

Step 4: Apply the Resource Quota

Apply the Resource Quota YAML to the namespace:

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

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

Verify that the Resource Quota is applied:

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

```
C:\Users\madha>kubectl get resourcequota -n quota-example
NAME            AGE   REQUEST
                      LIMIT
example-quota   24s   configmaps: 1/10, persistentvolumeclaims: 0/5, pods: 0/10, requests.cpu: 0/2, requests.memory: 0/4
Gi, 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
```

```
C:\Users\madha>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"
```

File Edit View

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
  namespace: quota-example
spec:
  replicas: 5
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
        resources:
          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
```

```
C:\Users\madha>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
```

```
C:\Users\madha>kubectl get pods -n quota-example
NAME                    READY   STATUS              RESTARTS   AGE
nginx-replicaset-8xb67  1/1     Running             0          21s
nginx-replicaset-crxp2  1/1     Running             0          21s
nginx-replicaset-msfp9  1/1     Running             0          21s
nginx-replicaset-snfh6  1/1     Running             0          21s
nginx-replicaset-x6485  0/1     ContainerCreating   0          21s
```

To describe the Pods and see their resource allocations:

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

```
C:\Users\madha>kubectl describe pods -l app=nginx -n quota-example
Name:             nginx-replicaset-8xb67
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Thu, 21 Nov 2024 12:39:07 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.5
IPs:
  IP:           10.244.0.5
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://8a1f50c749a0019ed88d5e2a0a8391a4c0916fce91ea566f43b2060b5e9837f5
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 21 Nov 2024 12:39:26 +0530
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     200m
      memory:  200Mi
    Requests:
```

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"
```

File Edit View

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-extra-pod
  namespace: quota-example
spec:
  containers:
  - name: nginx
    image: nginx:latest
    resources:
      requests:
        memory: "3Gi"    # Requests more memory than the quota allows
        cpu: "2"         # Requests more CPU than the quota allows
      limits:
        memory: "4Gi"
        cpu: "2"
```

Apply this YAML to create the Pod:

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

```
C:\Users\madha>kubectl apply -f nginx-extra-pod.yaml
resourcequota/example-quota configured
```

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

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

```
C:\Users\madha>kubectl get events -n quota-example
LAST SEEN   TYPE     REASON             OBJECT                          MESSAGE
5m37s       Normal   Scheduled          pod/nginx-replicaset-8xb67      Successfully assigned quota-example/nginx-replicas
et-8xb67 to minikube
5m37s       Normal   Pulling            pod/nginx-replicaset-8xb67      Pulling image "nginx:latest"
5m19s       Normal   Pulled             pod/nginx-replicaset-8xb67      Successfully pulled image "nginx:latest" in 3.08s
(14.404s including waiting). Image size: 191670156 bytes.
5m19s       Normal   Created            pod/nginx-replicaset-8xb67      Created container nginx
5m19s       Normal   Started            pod/nginx-replicaset-8xb67      Started container nginx
5m37s       Normal   Scheduled          pod/nginx-replicaset-crxp2      Successfully assigned quota-example/nginx-replicas
et-crxp2 to minikube
5m37s       Normal   Pulling            pod/nginx-replicaset-crxp2      Pulling image "nginx:latest"
5m22s       Normal   Pulled             pod/nginx-replicaset-crxp2      Successfully pulled image "nginx:latest" in 3.547s
 (11.323s including waiting). Image size: 191670156 bytes.
5m22s       Normal   Created            pod/nginx-replicaset-crxp2      Created container nginx
5m22s       Normal   Started            pod/nginx-replicaset-crxp2      Started container nginx
5m37s       Normal   Scheduled          pod/nginx-replicaset-msfp9      Successfully assigned quota-example/nginx-replicas
et-msfp9 to minikube
5m37s       Normal   Pulling            pod/nginx-replicaset-msfp9      Pulling image "nginx:latest"
5m29s       Normal   Pulled             pod/nginx-replicaset-msfp9      Successfully pulled image "nginx:latest" in 3.663s
 (7.775s including waiting). Image size: 191670156 bytes.
5m29s       Normal   Created            pod/nginx-replicaset-msfp9      Created container nginx
5m29s       Normal   Started            pod/nginx-replicaset-msfp9      Started container nginx
5m37s       Normal   Scheduled          pod/nginx-replicaset-snfh6      Successfully assigned quota-example/nginx-replicas
et-snfh6 to minikube
5m37s       Normal   Pulling            pod/nginx-replicaset-snfh6      Pulling image "nginx:latest"
5m33s       Normal   Pulled             pod/nginx-replicaset-snfh6      Successfully pulled image "nginx:latest" in 4.111s
 (4.111s including waiting). Image size: 191670156 bytes.
5m33s       Normal   Created            pod/nginx-replicaset-snfh6      Created 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
```

```
C:\Users\madha>kubectl delete -f nginx-replicaset-quota.yaml
replicaset.apps "nginx-replicaset" deleted

C:\Users\madha>kubectl delete -f nginx-extra-pod.yaml
resourcequota "example-quota" deleted

C:\Users\madha>kubectl delete -f resource-quota.yaml
Error from server (NotFound): error when deleting "resource-quota.yaml": resourcequotas "example-quota" not found

C:\Users\madha>kubectl delete namespace quota-example
namespace "quota-example" deleted

C:\Users\madha>kubectl get namespaces
NAME              STATUS   AGE
default           Active   17d
kube-node-lease   Active   17d
kube-public       Active   17d
kube-system       Active   17d

C:\Users\madha>kubectl get resourcequota -n quota-example
No resources found in quota-example namespace.
```