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.

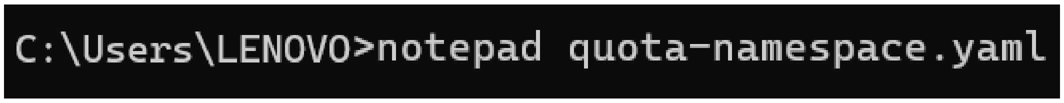
Step 1: Understand Resource Quotas

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

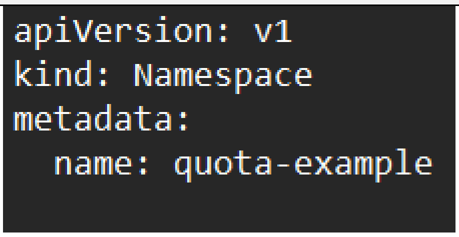
First, create a namespace where you will apply the Resource Quota. This helps in isolating and controlling resource usage within that specific namespace.
Create a YAML file named ***quota-namespace.yaml*** with the following content:



```
C:\Users\LENOVO>notepad quota-namespace.yaml
```

```
apiVersion: v1
kind: Namespace
```

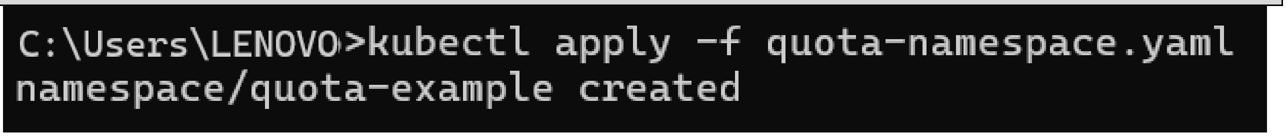
```
metadata:
  name: quota-example    # The name of the namespace.
```



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

Apply the YAML to create the namespace:

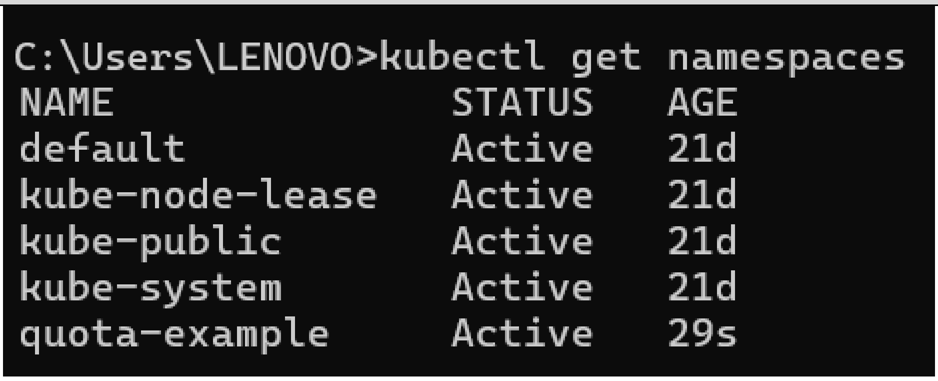
```
kubectl apply -f quota-namespace.yaml
```



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

Verify that the namespace is created:

```
kubectl get namespaces
```



```
C:\Users\LENOVO>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   29s
```

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:

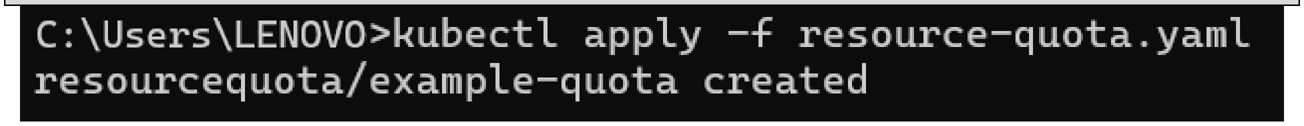
```
apiVersion: v1
```

```
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.
```

Step 4: Apply the Resource Quota

Apply the Resource Quota YAML to the namespace:

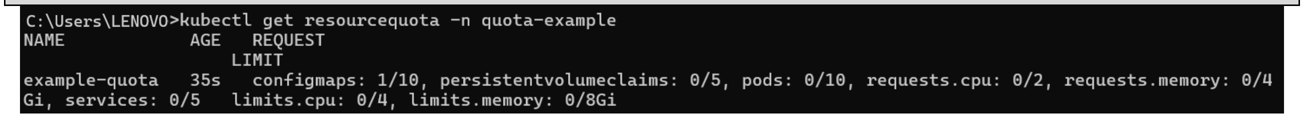
```
kubectl apply -f resource-quota.yaml
```



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

Verify that the Resource Quota is applied:

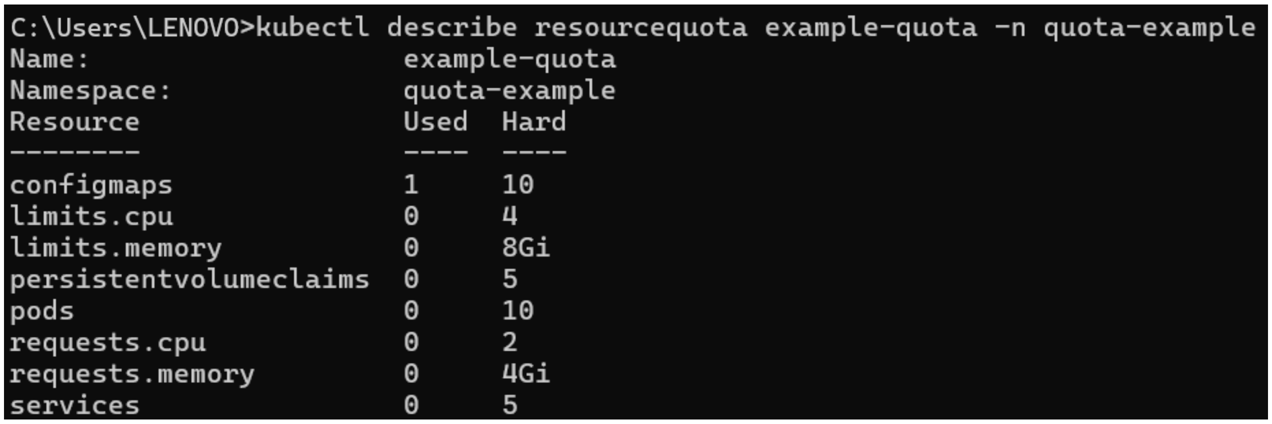
```
kubectl get resourcequota -n quota-example
```



```
C:\Users\LENOVO>kubectl get resourcequota -n quota-example
NAME            AGE   REQUEST
                   LIMIT
example-quota   35s   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\LENOVO>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:
```

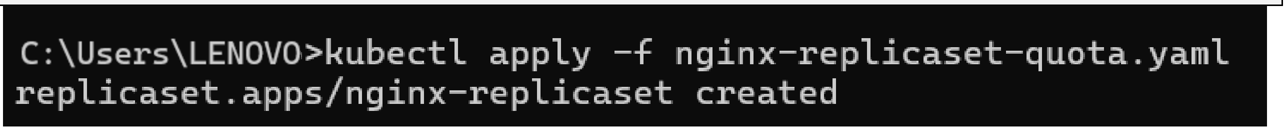
```
    containers:
    - name: nginx
      image: nginx:latest
      ports:
      - containerPort: 80
      resources:        # Define resource requests and limits.
        requests:
          memory: "100Mi"
          cpu: "100m"
        limits:
          memory: "200Mi"
          cpu: "200m"
```

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:

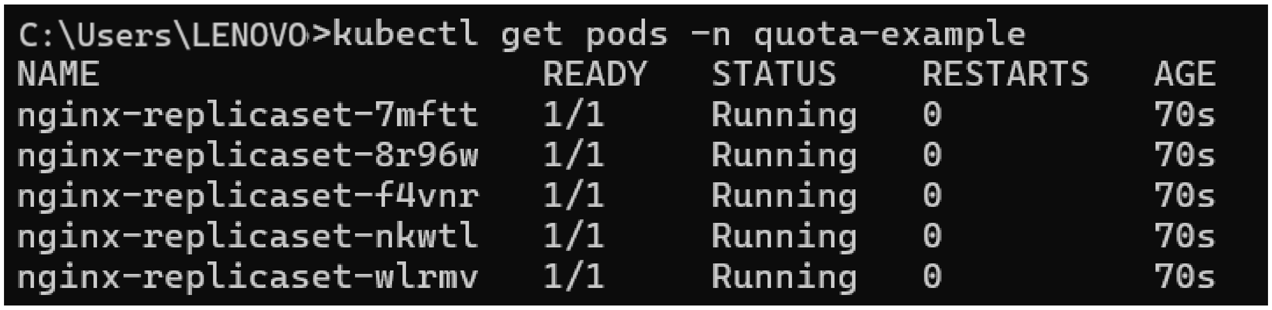
```
kubectl apply -f nginx-replicaset-quota.yaml
```



```
C:\Users\LENOVO>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:

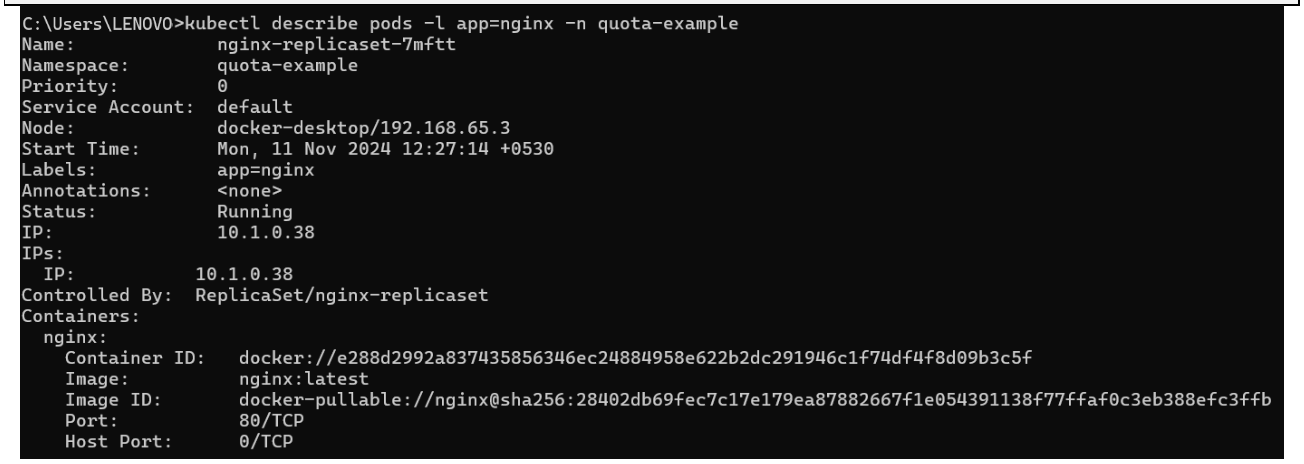
```
kubectl get pods -n quota-example
```



```
C:\Users\LENOVO>kubectl get pods -n quota-example
NAME                     READY   STATUS    RESTARTS   AGE
nginx-replicaset-7mftt   1/1     Running   0          70s
nginx-replicaset-8r96w   1/1     Running   0          70s
nginx-replicaset-f4vnr   1/1     Running   0          70s
nginx-replicaset-nkwtl   1/1     Running   0          70s
nginx-replicaset-wlrmv   1/1     Running   0          70s
```

To describe the Pods and see their resource allocations:

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



```
C:\Users\LENOVO>kubectl describe pods -l app=nginx -n quota-example
Name:             nginx-replicaset-7mftt
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Mon, 11 Nov 2024 12:27:14 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.38
IPs:
  IP:           10.1.0.38
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://e288d2992a837435856346ec24884958e622b2dc291946c1f74df4f8d09b3c5f
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:28402db69fec7c17e179ea87882667f1e054391138f77ffaf0c3eb388efc3ffb
    Port:           80/TCP
    Host Port:      0/TCP
```

Attempt to Exceed the Resource Quota

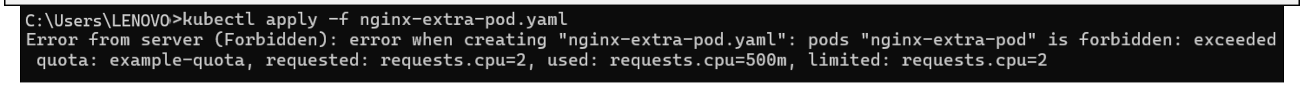
Try creating additional resources to see if they are rejected when exceeding the quota. For example, create more Pods or increase the CPU/memory requests to exceed the quota limits.

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"
```

Apply this YAML to create the Pod:

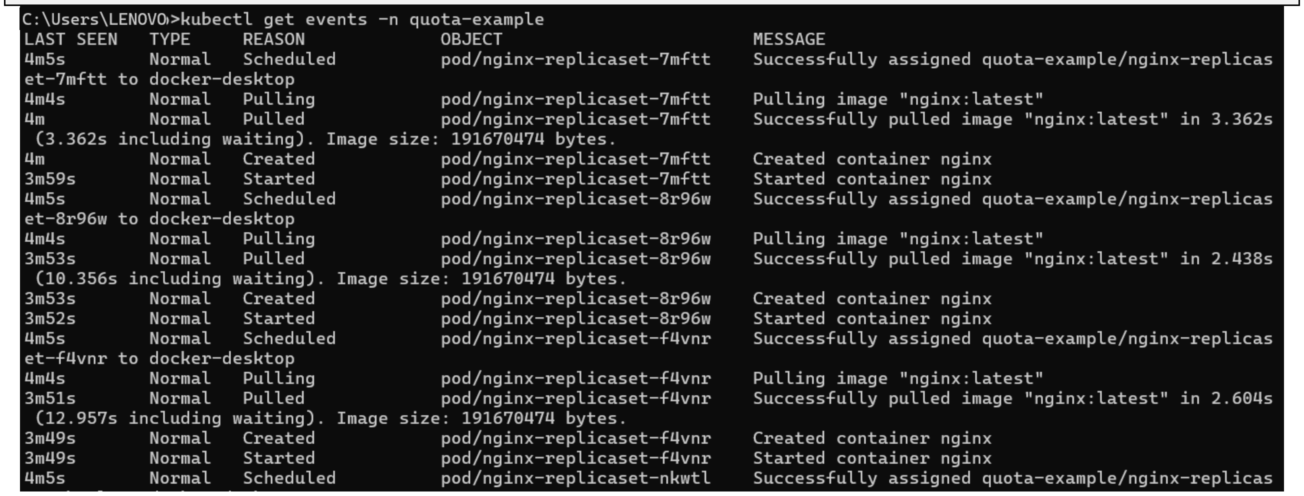
```
kubectl apply -f nginx-extra-pod.yaml
```



```
C:\Users\LENOVO>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
```

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

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



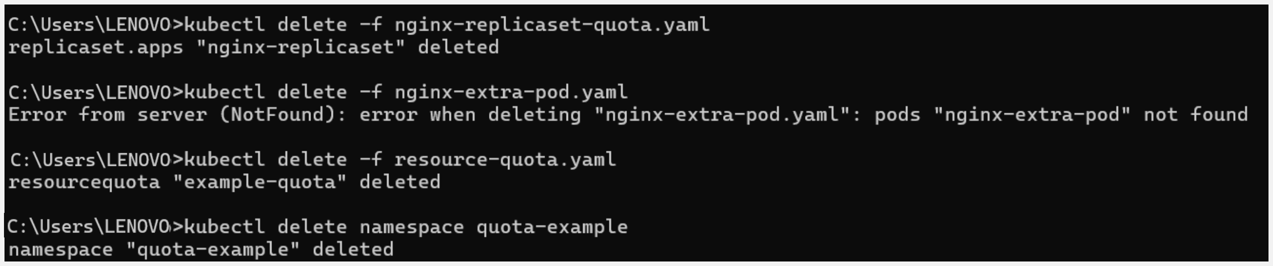
```
C:\Users\LENOVO>kubectl get events -n quota-example
LAST SEEN   TYPE     REASON              OBJECT                          MESSAGE
4m5s        Normal   Scheduled           pod/nginx-replicaset-7mftt    Successfully assigned quota-example/nginx-replicas
et-7mftt to docker-desktop
4m4s        Normal   Pulling             pod/nginx-replicaset-7mftt    Pulling image "nginx:latest"
4m          Normal   Pulled              pod/nginx-replicaset-7mftt    Successfully pulled image "nginx:latest" in 3.362s
 (3.362s including waiting). Image size: 191670474 bytes.
4m          Normal   Created             pod/nginx-replicaset-7mftt    Created container nginx
3m59s       Normal   Started             pod/nginx-replicaset-7mftt    Started container nginx
4m5s        Normal   Scheduled           pod/nginx-replicaset-8r96w    Successfully assigned quota-example/nginx-replicas
et-8r96w to docker-desktop
4m4s        Normal   Pulling             pod/nginx-replicaset-8r96w    Pulling image "nginx:latest"
3m53s       Normal   Pulled              pod/nginx-replicaset-8r96w    Successfully pulled image "nginx:latest" in 2.438s
 (10.356s including waiting). Image size: 191670474 bytes.
3m53s       Normal   Created             pod/nginx-replicaset-8r96w    Created container nginx
3m52s       Normal   Started             pod/nginx-replicaset-8r96w    Started container nginx
4m5s        Normal   Scheduled           pod/nginx-replicaset-f4vnr    Successfully assigned quota-example/nginx-replicas
et-f4vnr to docker-desktop
4m4s        Normal   Pulling             pod/nginx-replicaset-f4vnr    Pulling image "nginx:latest"
3m51s       Normal   Pulled              pod/nginx-replicaset-f4vnr    Successfully pulled image "nginx:latest" in 2.604s
 (12.957s including waiting). Image size: 191670474 bytes.
3m49s       Normal   Created             pod/nginx-replicaset-f4vnr    Created container nginx
3m49s       Normal   Started             pod/nginx-replicaset-f4vnr    Started container nginx
4m5s        Normal   Scheduled           pod/nginx-replicaset-nkwtl    Successfully assigned quota-example/nginx-replicas
```

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\LENOVO>kubectl delete -f nginx-replicaset-quota.yaml
replicaset.apps "nginx-replicaset" deleted

C:\Users\LENOVO>kubectl delete -f nginx-extra-pod.yaml
Error from server (NotFound): error when deleting "nginx-extra-pod.yaml": pods "nginx-extra-pod" not found

C:\Users\LENOVO>kubectl delete -f resource-quota.yaml
resourcequota "example-quota" deleted

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