ANSHIKA SRIVASTAVA

ROLL NUMBER – R2142220907

SAP ID – 500107049

DEVSECOPS BATCH B1 HONS.

EXPERIMENT – 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.

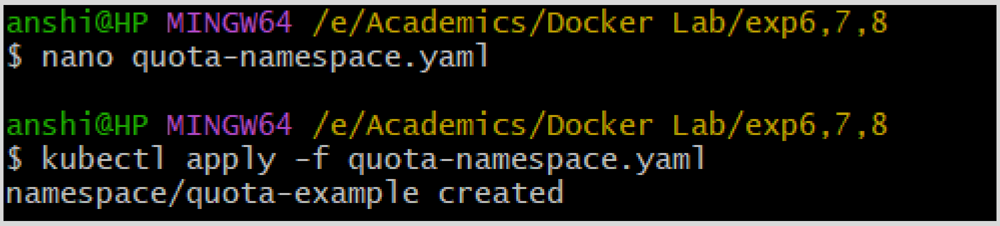
Create a YAML file named ***quota-namespace.yaml*** with the following content:

```
apiVersion: v1

kind: Namespace
```

```
metadata:

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

Apply the YAML to create the namespace:

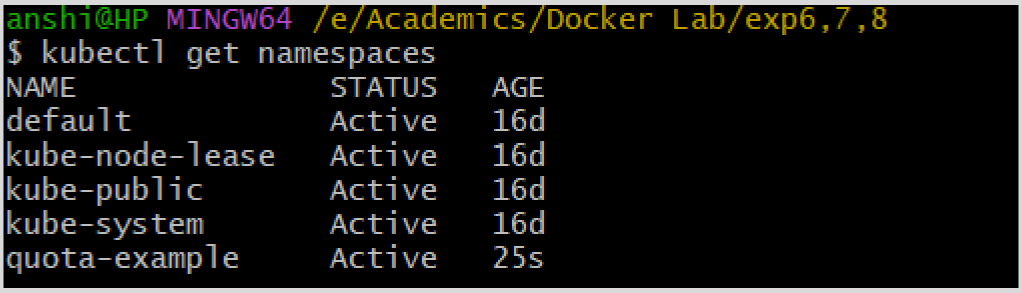
```
kubectl apply -f quota-namespace.yaml
```



```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ nano quota-namespace.yaml

anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ kubectl apply -f quota-namespace.yaml
namespace/quota-example created
```

Verify that the namespace is created:

```
kubectl get namespaces
```



```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ kubectl get namespaces
NAME              STATUS   AGE
default           Active   16d
kube-node-lease   Active   16d
kube-public       Active   16d
kube-system       Active   16d
quota-example     Active   25s
```

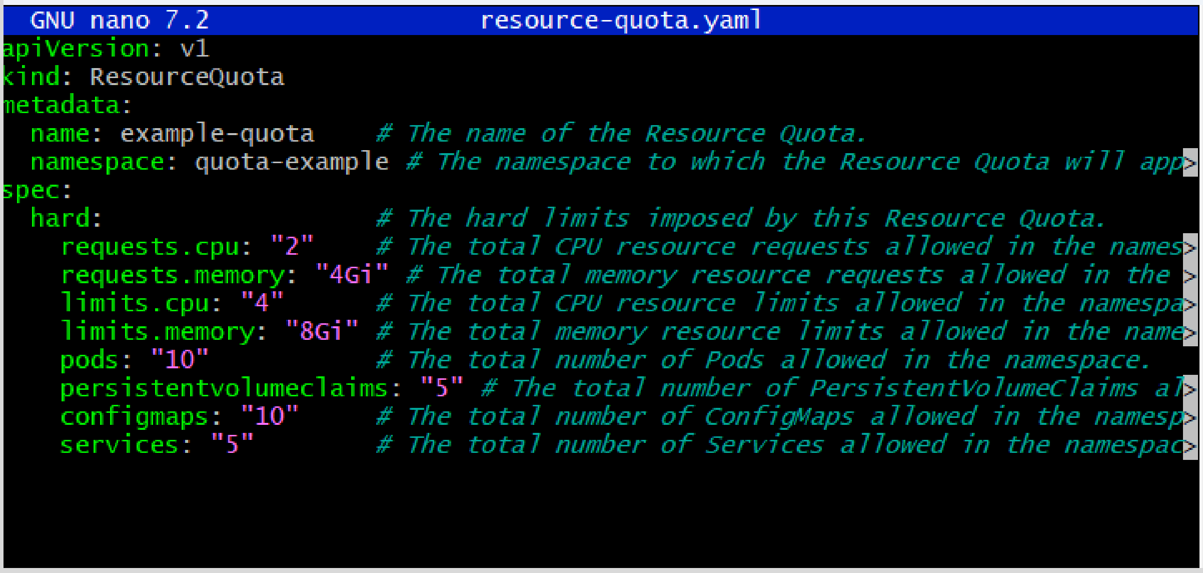
You should see quota-example listed in the output.

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.
```

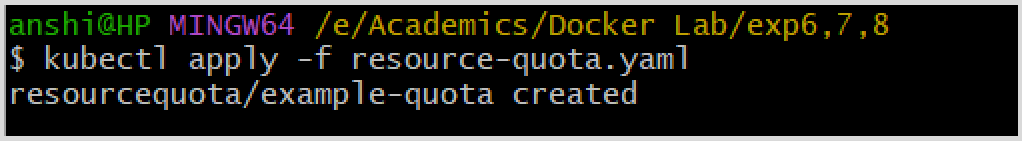


```
  GNU nano 7.2                              resource-quota.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: example-quota    # The name of the Resource Quota.
  namespace: quota-example # The namespace to which the Resource Quota will app>
spec:
  hard:                  # The hard limits imposed by this Resource Quota.
    requests.cpu: "2"    # The total CPU resource requests allowed in the names>
    requests.memory: "4Gi" # The total memory resource requests allowed in the >
    limits.cpu: "4"      # The total CPU resource limits allowed in the namespa>
    limits.memory: "8Gi" # The total memory resource limits allowed in the name>
    pods: "10"           # The total number of Pods allowed in the namespace.
    persistentvolumeclaims: "5" # The total number of PersistentVolumeClaims al>
    configmaps: "10"     # The total number of ConfigMaps allowed in the namesp>
    services: "5"        # The total number of Services allowed in the namespac>
```

Step 4: Apply the Resource Quota

Apply the Resource Quota YAML to the namespace:

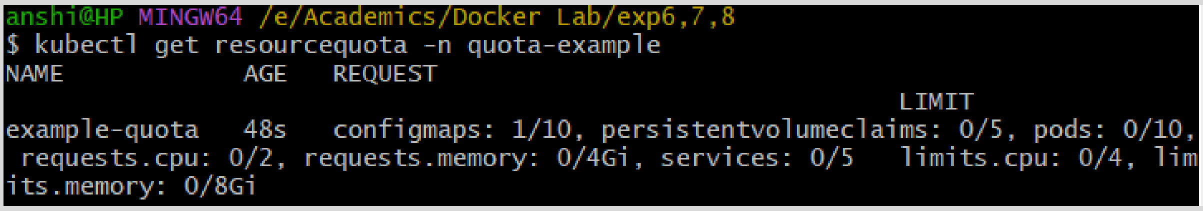
```
kubectl apply -f resource-quota.yaml
```



```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ kubectl apply -f resource-quota.yaml
resourcequota/example-quota created
```

Verify that the Resource Quota is applied:

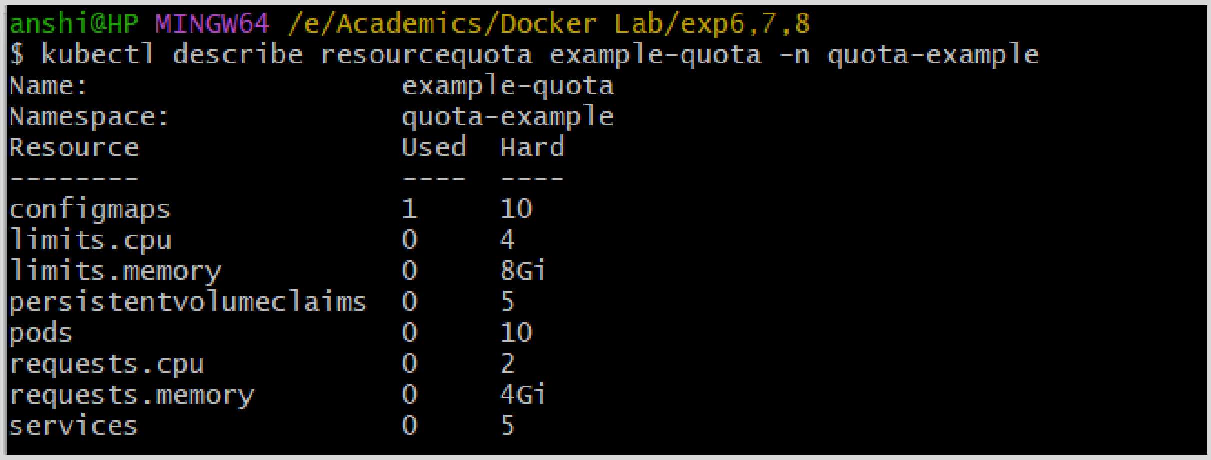
```
kubectl get resourcequota -n quota-example
```



```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ 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, lim
its.memory: 0/8Gi
```

To see the details of the applied Resource Quota:

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



```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ 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
```

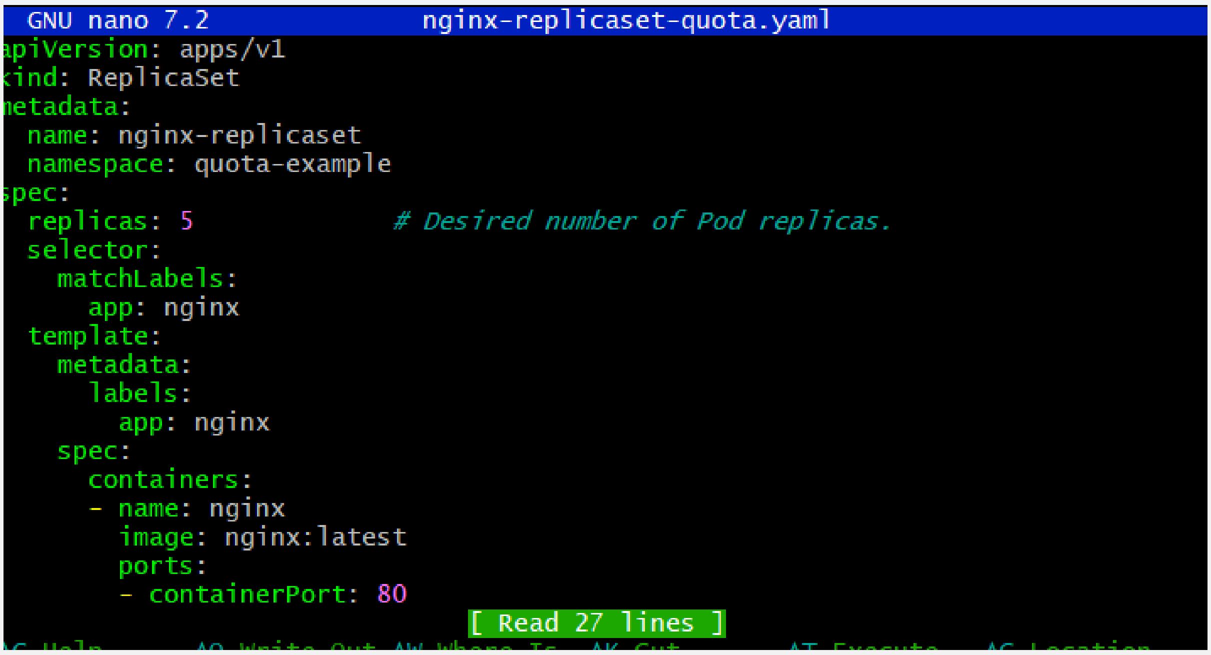
Step 5: Test the Resource Quota

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"
```



```
  GNU nano 7.2              nginx-replicaset-quota.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
                               [ Read 27 lines ]
^G Help      ^O Write Out ^W Where Is ^K Cut       ^T Execute   ^C Location
```

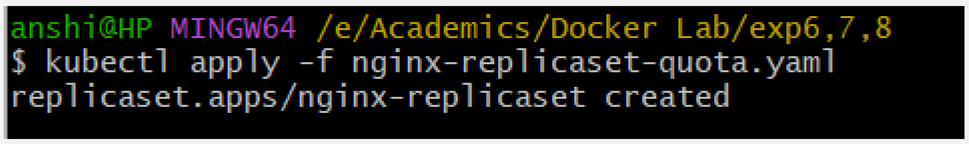
Explanation:

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:

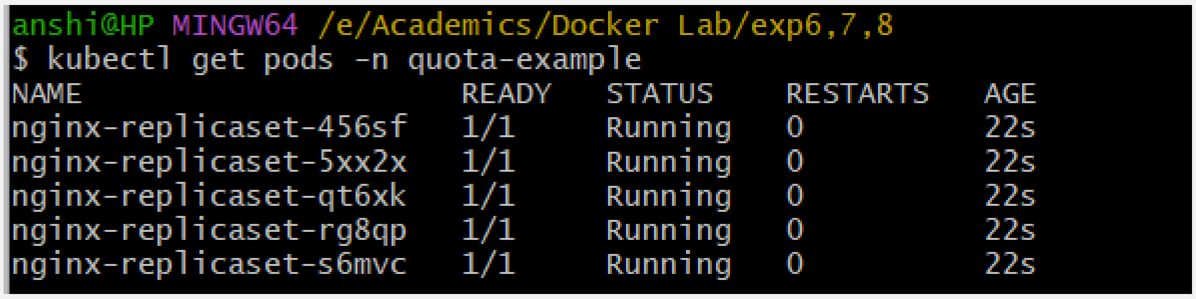
```
kubectl apply -f nginx-replicaset-quota.yaml
```



```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ 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:

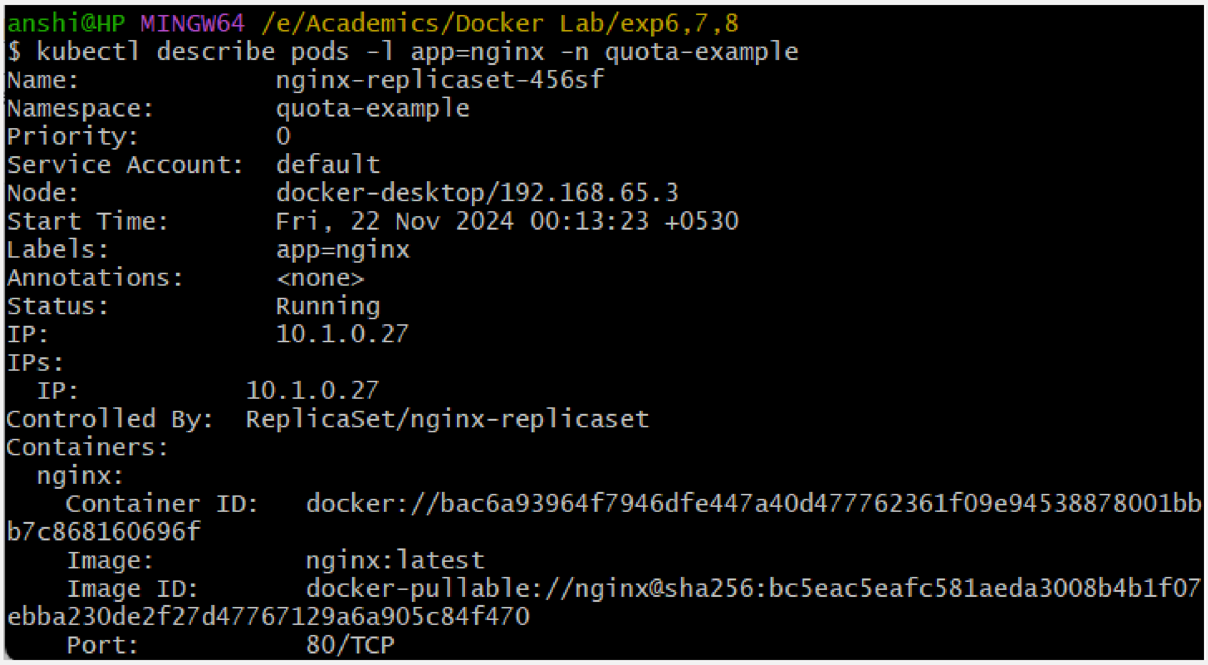
```
kubectl get pods -n quota-example
```



```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ kubectl get pods -n quota-example
NAME                     READY   STATUS    RESTARTS   AGE
nginx-replicaset-456sf   1/1     Running   0          22s
nginx-replicaset-5xx2x   1/1     Running   0          22s
nginx-replicaset-qt6xk   1/1     Running   0          22s
nginx-replicaset-rg8qp   1/1     Running   0          22s
nginx-replicaset-s6mvc   1/1     Running   0          22s
```

To describe the Pods and see their resource allocations:

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



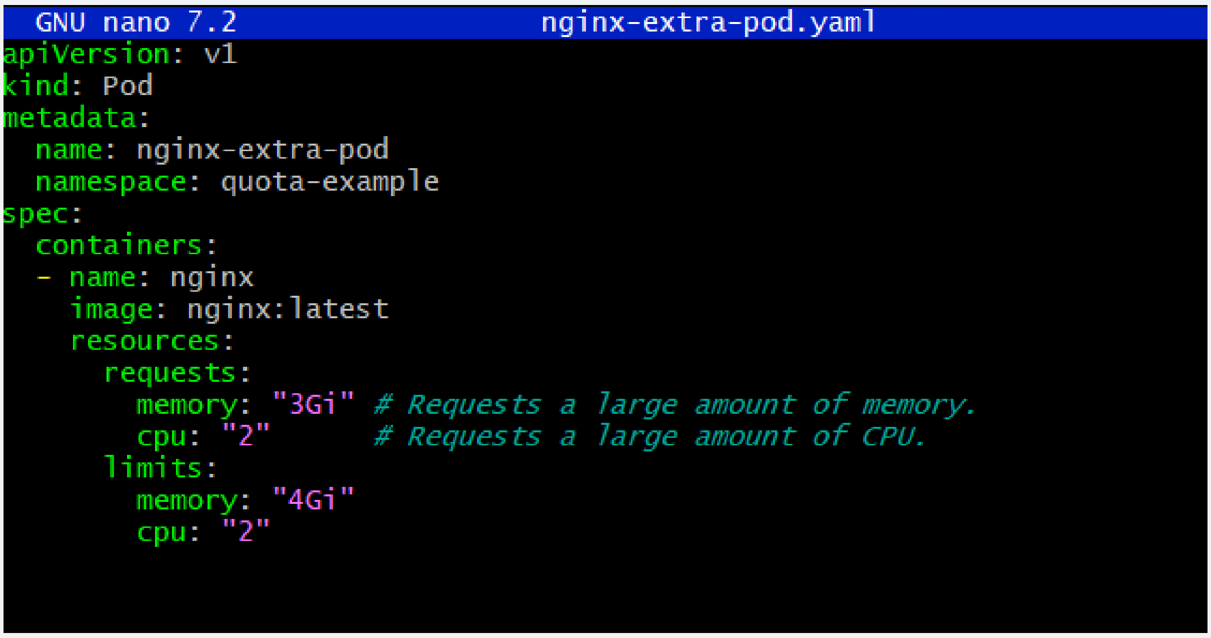
```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ kubectl describe pods -l app=nginx -n quota-example
Name:             nginx-replicaset-456sf
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Fri, 22 Nov 2024 00:13:23 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.27
IPs:
  IP:           10.1.0.27
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://bac6a93964f7946dfe447a40d477762361f09e94538878001bb
b7c868160696f
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:bc5eac5eafc581aeda3008b4b1f07
ebba230de2f27d47767129a6a905c84f470
    Port:           80/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:

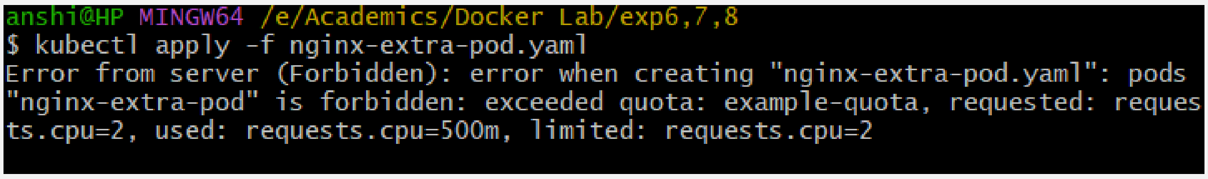
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-extra-pod
  namespace: quota-example
spec:
  containers:
  - name: nginx
    image: nginx:latest
    resources:
      requests:
        memory: "3Gi" # Requests a large amount of memory.
        cpu: "2"      # Requests a large amount of CPU.
      limits:
        memory: "4Gi"
        cpu: "2"
```



```
  GNU nano 7.2                    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:

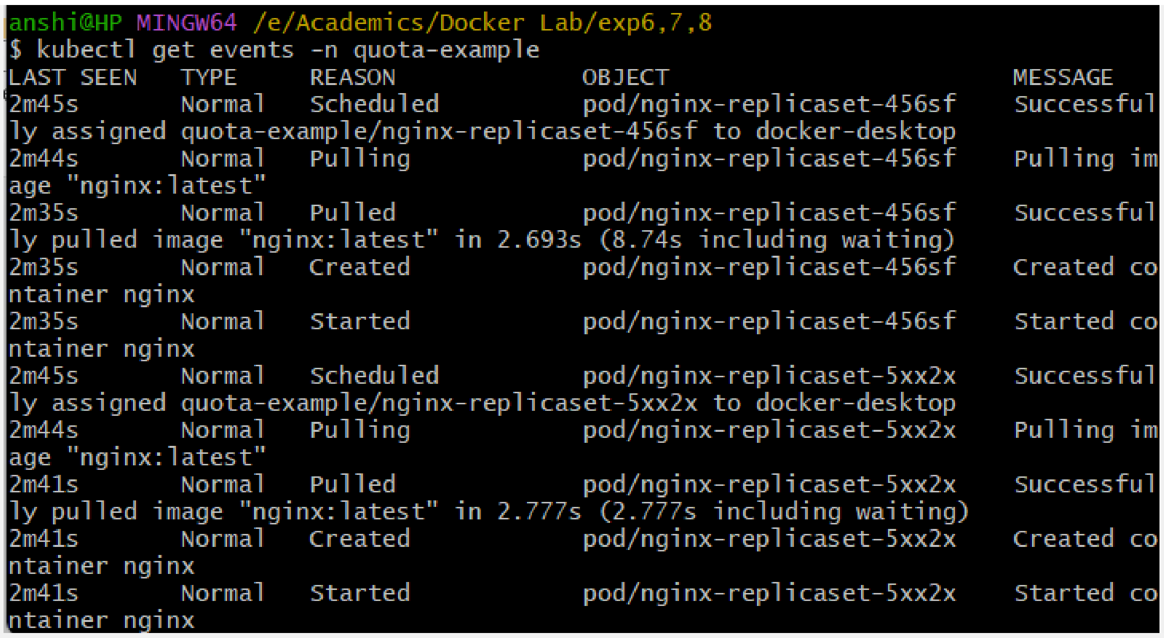
```
kubectl apply -f nginx-extra-pod.yaml
```



```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ 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
```



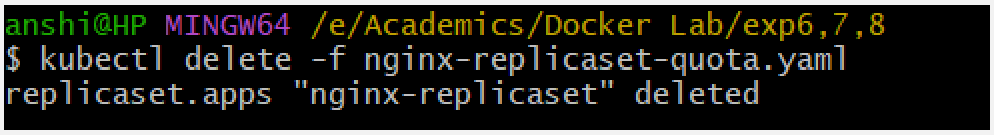
```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ kubectl get events -n quota-example
LAST SEEN   TYPE     REASON              OBJECT                          MESSAGE
2m45s       Normal   Scheduled           pod/nginx-replicaset-456sf      Successful
ly assigned quota-example/nginx-replicaset-456sf to docker-desktop
2m44s       Normal   Pulling             pod/nginx-replicaset-456sf      Pulling im
age "nginx:latest"
2m35s       Normal   Pulled              pod/nginx-replicaset-456sf      Successful
ly pulled image "nginx:latest" in 2.693s (8.74s including waiting)
2m35s       Normal   Created             pod/nginx-replicaset-456sf      Created co
ntainer nginx
2m35s       Normal   Started             pod/nginx-replicaset-456sf      Started co
ntainer nginx
2m45s       Normal   Scheduled           pod/nginx-replicaset-5xx2x      Successful
ly assigned quota-example/nginx-replicaset-5xx2x to docker-desktop
2m44s       Normal   Pulling             pod/nginx-replicaset-5xx2x      Pulling im
age "nginx:latest"
2m41s       Normal   Pulled              pod/nginx-replicaset-5xx2x      Successful
ly pulled image "nginx:latest" in 2.777s (2.777s including waiting)
2m41s       Normal   Created             pod/nginx-replicaset-5xx2x      Created co
ntainer nginx
2m41s       Normal   Started             pod/nginx-replicaset-5xx2x      Started co
ntainer nginx
```

Look for error messages indicating that the Pod creation was denied due to resource constraints.

Step 6: Clean Up Resources

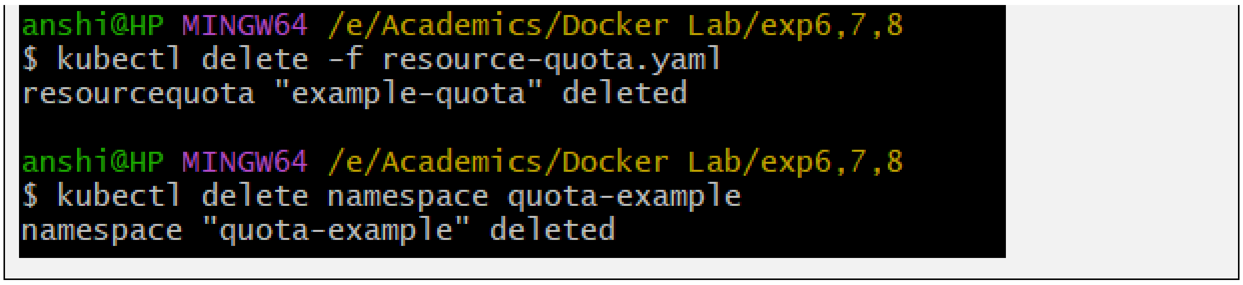
To delete the resources you created:

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



```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ kubectl delete -f nginx-replicaset-quota.yaml
replicaset.apps "nginx-replicaset" deleted
```

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

kubectl delete -f resource-quota.yaml

kubectl delete namespace quota-example
```



```
anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ kubectl delete -f resource-quota.yaml
resourcequota "example-quota" deleted

anshi@HP MINGW64 /e/Academics/Docker Lab/exp6,7,8
$ kubectl delete namespace quota-example
namespace "quota-example" deleted
```