# Containers & Docker Security LAB

SUBMITTED TO
Dr. Hitesh Kumar Sharma

SUBMITTED BY
Siddharth Agarwal
500107594
R2142220663
Btech CSE DevOps B1

# 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

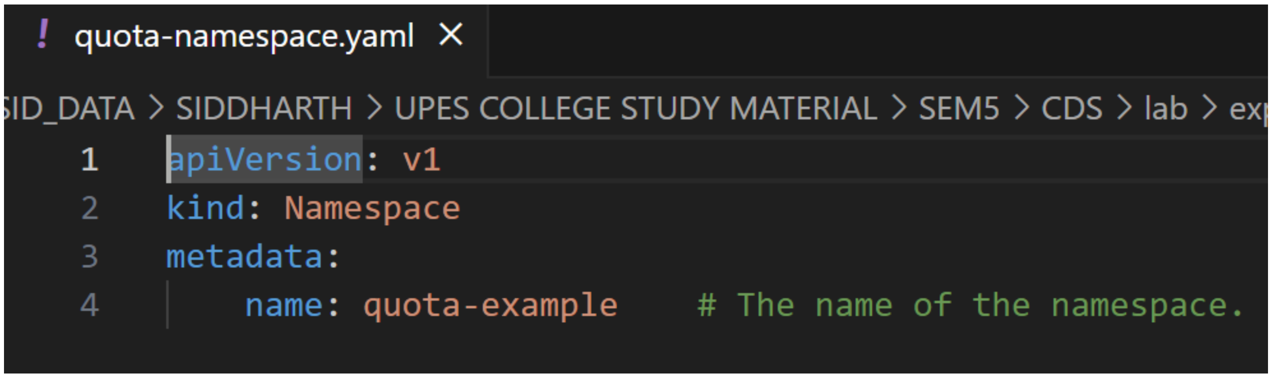
Resource Quotas allow you to:

- 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:

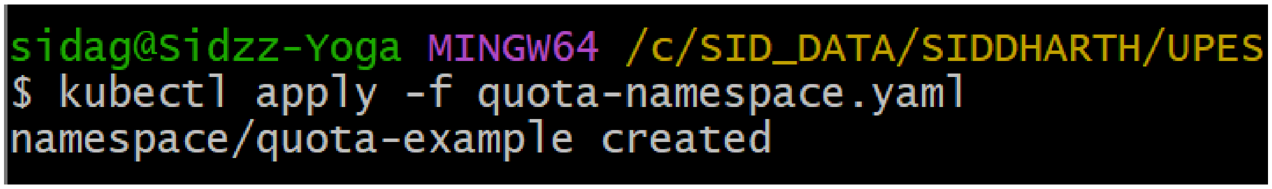
```
apiVersion: v1
kind: Namespace
metadata:
  name: quota-example    # The name of the namespace.
```


quota-namespace.yaml
ID_DATA > SIDDHARTH > UPES COLLEGE STUDY MATERIAL > SEM5 > CDS > lab > ex

```yaml
apiVersion: v1
kind: Namespace
metadata:
    name: quota-example     # The name of the namespace.
```

Apply the YAML to create the namespace:

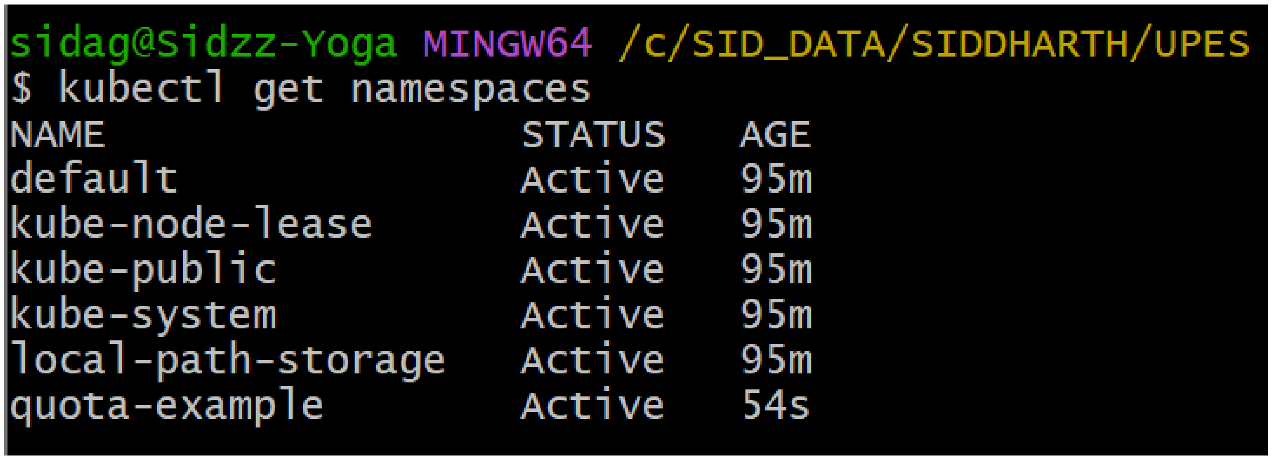
```
kubectl apply -f quota-namespace.yaml
```



```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES
$ kubectl apply -f quota-namespace.yaml
namespace/quota-example created
```

Verify that the namespace is created:

```
kubectl get namespaces
```



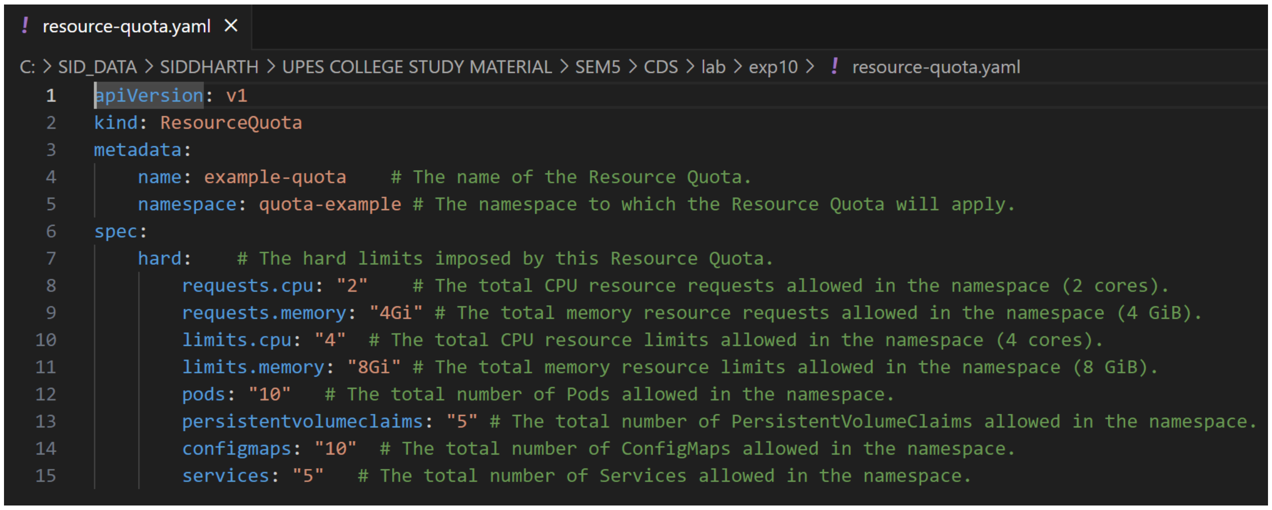
```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES
$ kubectl get namespaces
NAME                 STATUS   AGE
default              Active   95m
kube-node-lease      Active   95m
kube-public          Active   95m
kube-system          Active   95m
local-path-storage   Active   95m
quota-example        Active   54s
```

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.
```

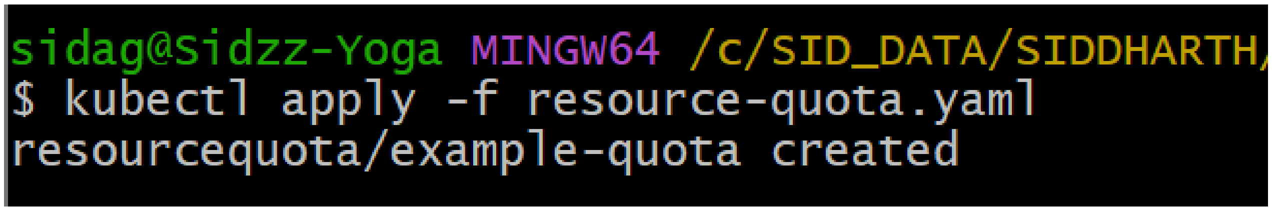

! resource-quota.yaml ×
C: > SID_DATA > SIDDHARTH > UPES COLLEGE STUDY MATERIAL > SEM5 > CDS > lab > exp10 > ! resource-quota.yaml

```
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 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:

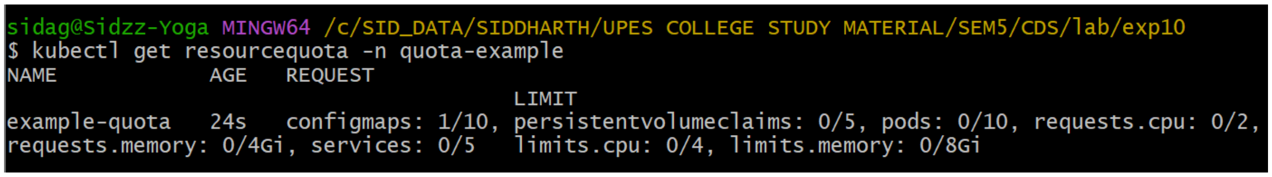
```
kubectl apply -f resource-quota.yaml
```



```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH
$ kubectl apply -f resource-quota.yaml
resourcequota/example-quota created
```

Verify that the Resource Quota is applied:

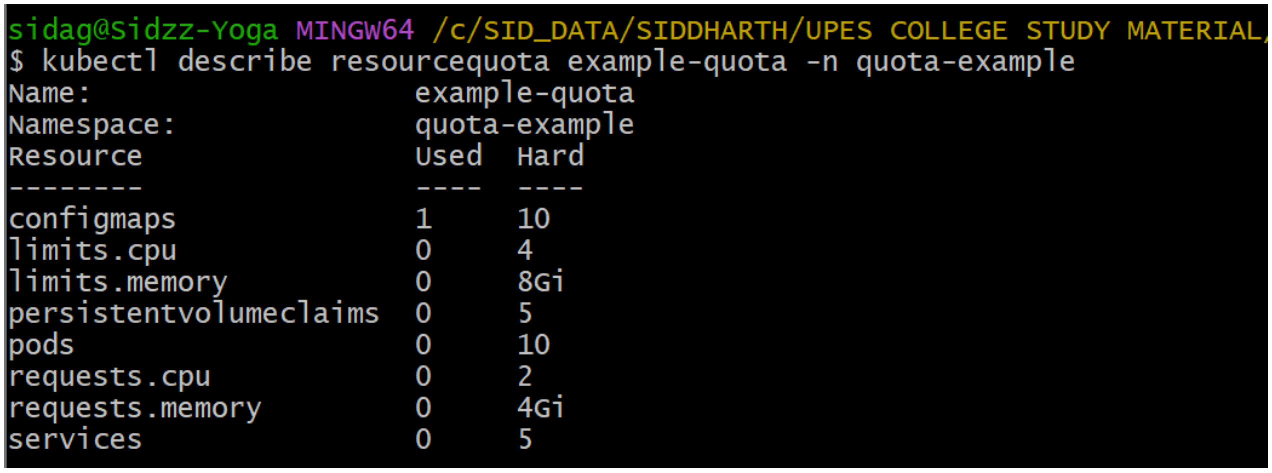
```
kubectl get resourcequota -n quota-example
```



```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES COLLEGE STUDY MATERIAL/SEM5/CDS/lab/exp10
$ 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: 0/4Gi, 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
```



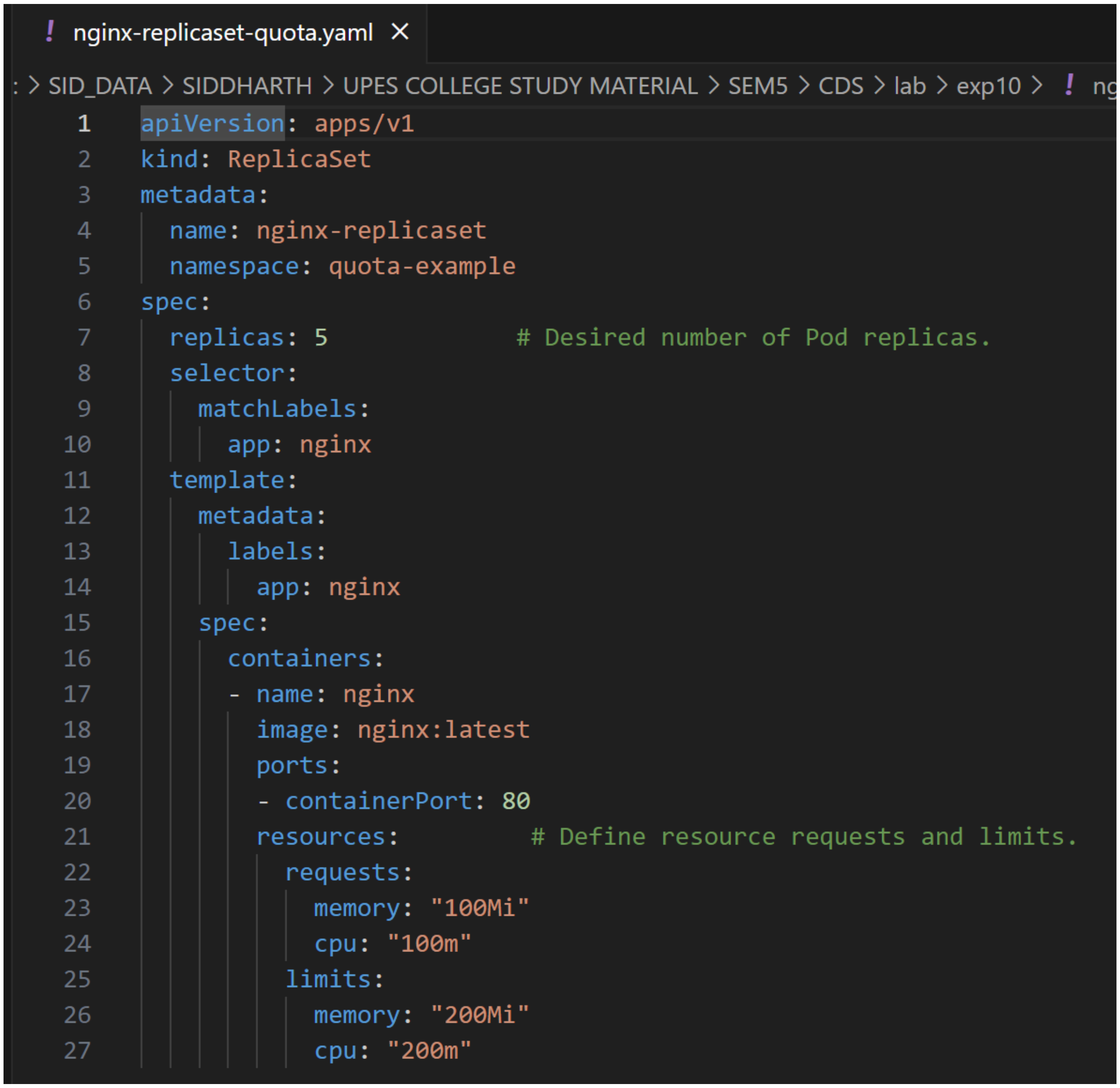
```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES COLLEGE STUDY MATERIAL
$ kubectl describe resourcequota example-quota -n quota-example
Name:                   example-quota
Namespace:              quota-example
Resource                Used  Hard
--------                ----  ----
configmaps              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"
```



nginx-replicaset-quota.yaml

: > SID_DATA > SIDDHARTH > UPES COLLEGE STUDY MATERIAL > SEM5 > CDS > lab > exp10 > ng

```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"
```

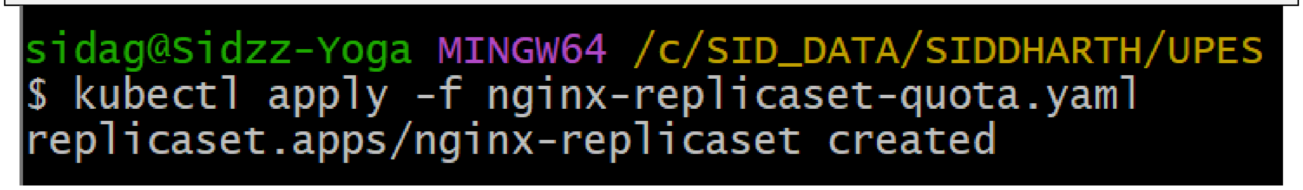
**Explanation:**

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:

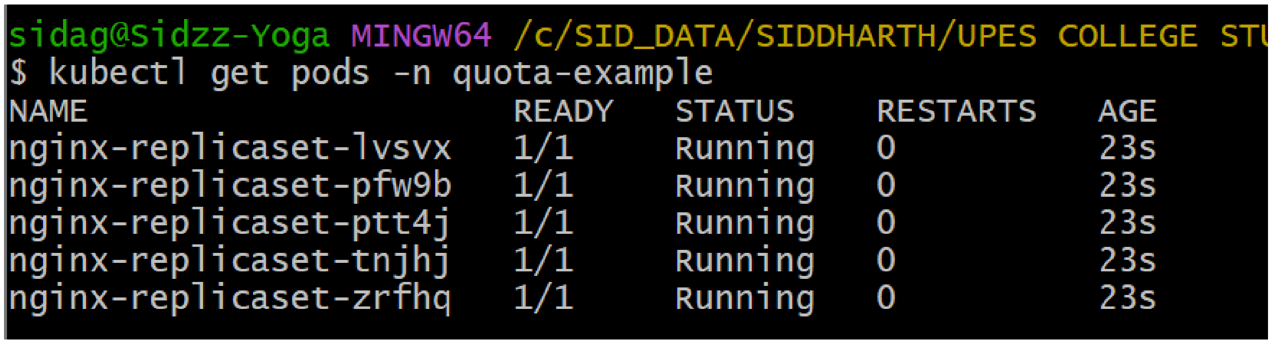
```
kubectl apply -f nginx-replicaset-quota.yaml
```



```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES
$ 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:

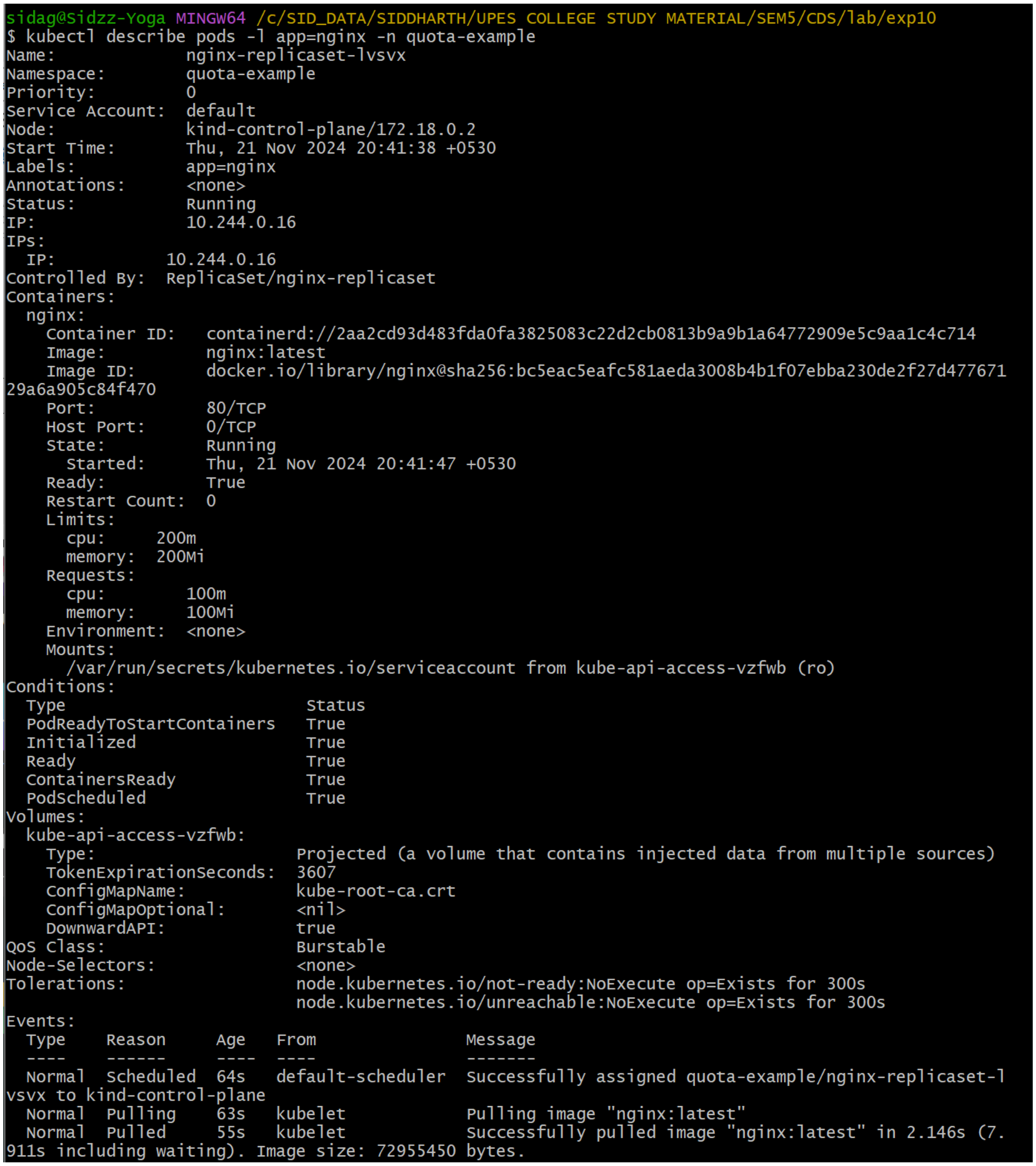
```
kubectl get pods -n quota-example
```



```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES COLLEGE STU
$ kubectl get pods -n quota-example
NAME                     READY   STATUS    RESTARTS   AGE
nginx-replicaset-lvsvx   1/1     Running   0          23s
nginx-replicaset-pfw9b   1/1     Running   0          23s
nginx-replicaset-ptt4j   1/1     Running   0          23s
nginx-replicaset-tnjhj   1/1     Running   0          23s
nginx-replicaset-zrfhq   1/1     Running   0          23s
```

To describe the Pods and see their resource allocations:

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



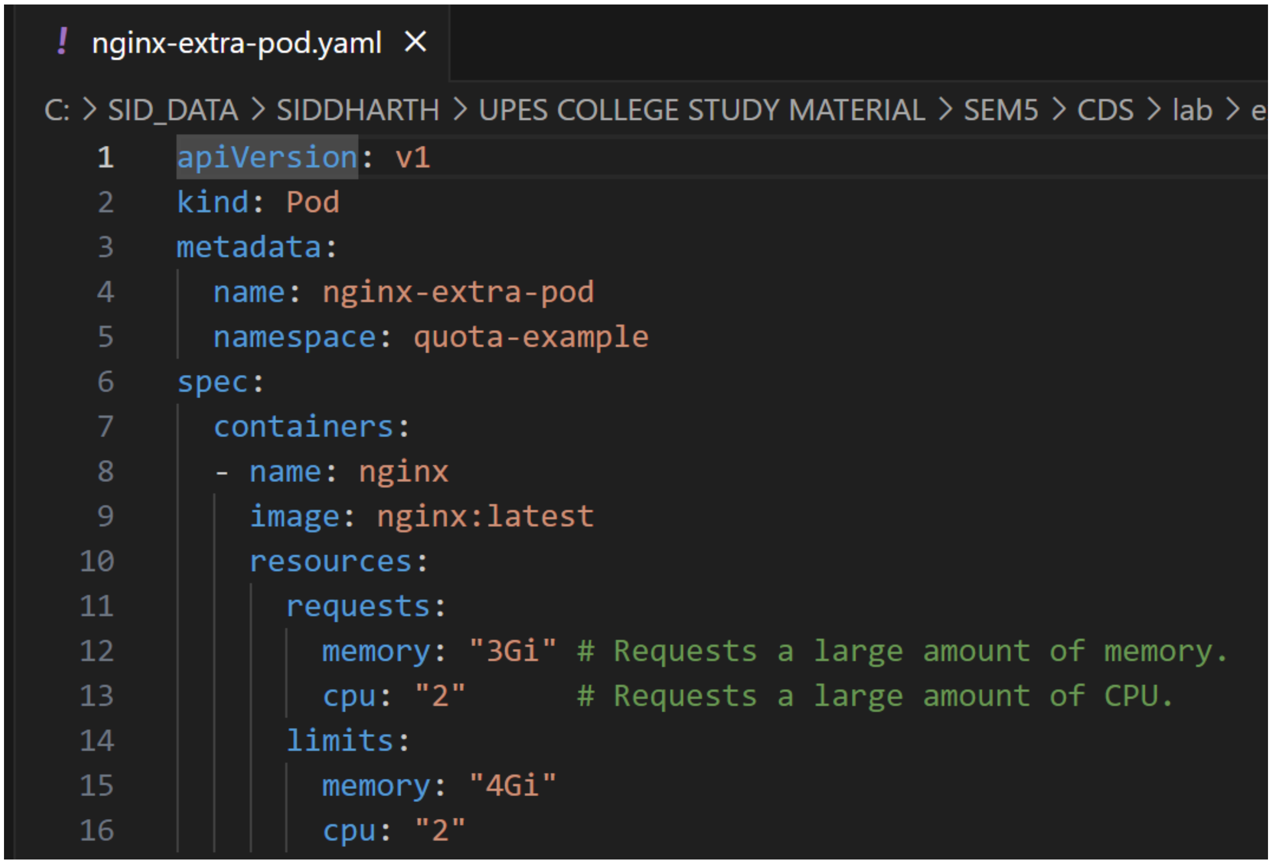
```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES COLLEGE STUDY MATERIAL/SEM5/CDS/lab/exp10
$ kubectl describe pods -l app=nginx -n quota-example
Name:             nginx-replicaset-lvsvx
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             kind-control-plane/172.18.0.2
Start Time:       Thu, 21 Nov 2024 20:41:38 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.16
IPs:
  IP:           10.244.0.16
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   containerd://2aa2cd93d483fda0fa3825083c22d2cb0813b9a9b1a64772909e5c9aa1c4c714
    Image:          nginx:latest
    Image ID:       docker.io/library/nginx@sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d477671
29a6a905c84f470
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 21 Nov 2024 20:41:47 +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-vzfwb (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-vzfwb:
    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>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  64s   default-scheduler  Successfully assigned quota-example/nginx-replicaset-l
vsvx to kind-control-plane
  Normal  Pulling    63s   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     55s   kubelet            Successfully pulled image "nginx:latest" in 2.146s (7.
911s including waiting). Image size: 72955450 bytes.
```

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:

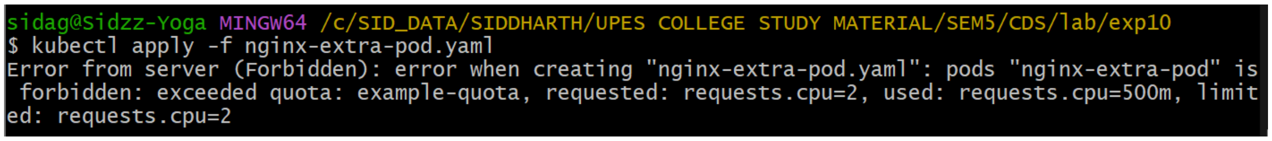
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-extra-pod
  namespace: quota-example
spec:
  containers:
  - name: nginx
    image: nginx:latest
    resources:
      requests:
        memory: "3Gi" # Requests a large amount of memory.
        cpu: "2"      # Requests a large amount of CPU.
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


! nginx-extra-pod.yaml
C: > SID_DATA > SIDDHARTH > UPES COLLEGE STUDY MATERIAL > SEM5 > CDS > lab > e

```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:

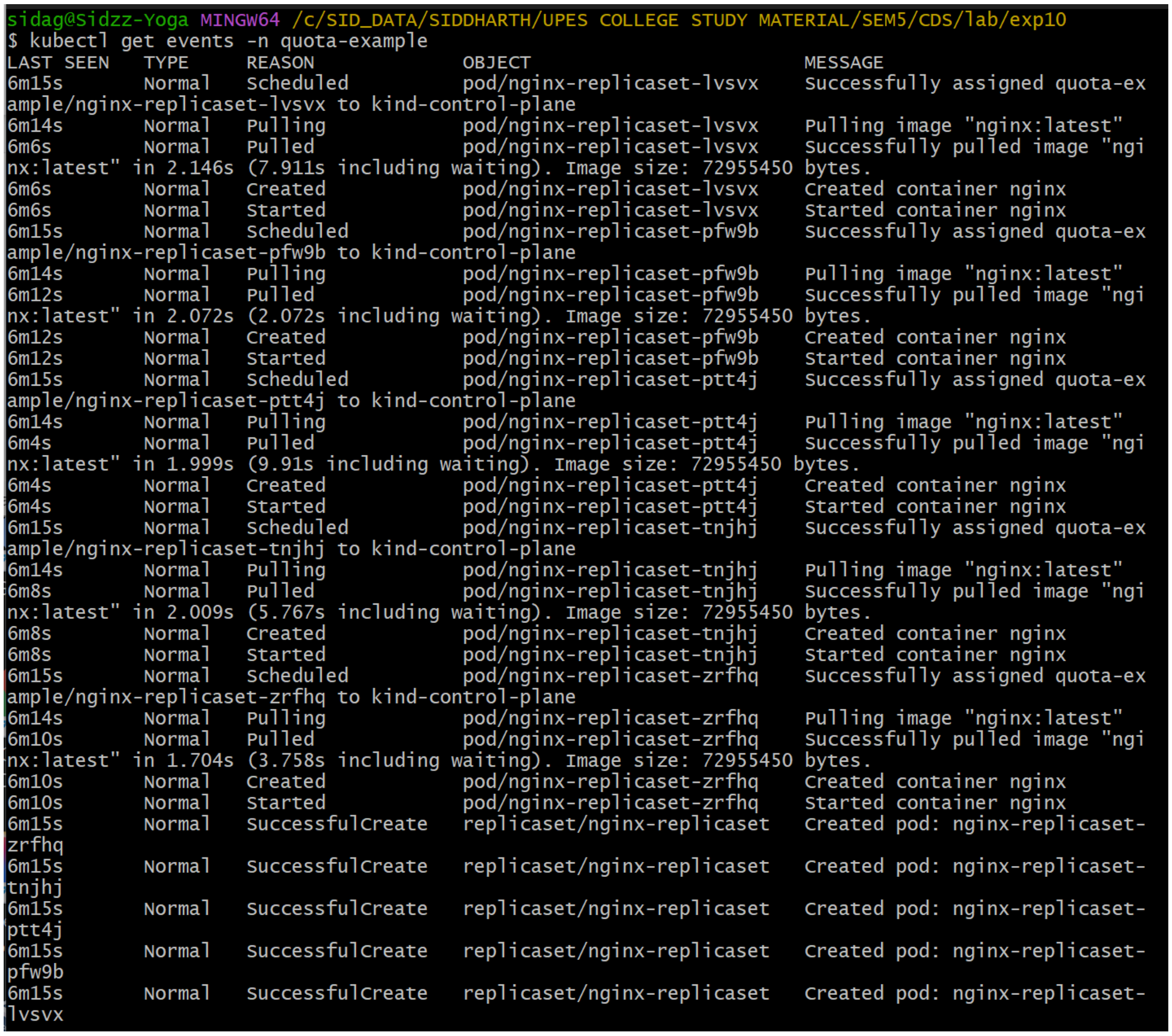
```
kubectl apply -f nginx-extra-pod.yaml
```



```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES COLLEGE STUDY MATERIAL/SEM5/CDS/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, limit
ed: 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
```



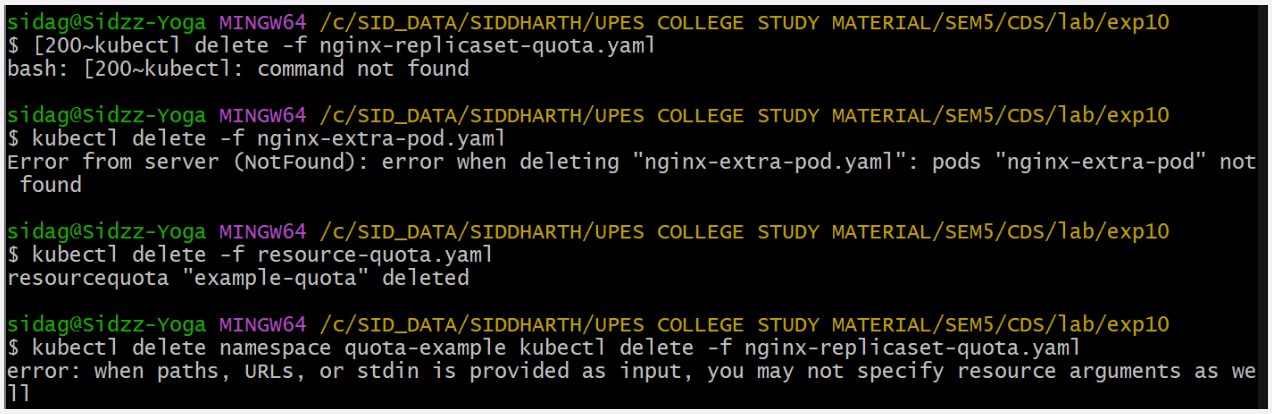
```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES COLLEGE STUDY MATERIAL/SEM5/CDS/lab/exp10
$ kubectl get events -n quota-example
LAST SEEN   TYPE     REASON             OBJECT                         MESSAGE
6m15s       Normal   Scheduled          pod/nginx-replicaset-lvsvx     Successfully assigned quota-ex
ample/nginx-replicaset-lvsvx to kind-control-plane
6m14s       Normal   Pulling            pod/nginx-replicaset-lvsvx     Pulling image "nginx:latest"
6m6s        Normal   Pulled             pod/nginx-replicaset-lvsvx     Successfully pulled image "ngi
nx:latest" in 2.146s (7.911s including waiting). Image size: 72955450 bytes.
6m6s        Normal   Created            pod/nginx-replicaset-lvsvx     Created container nginx
6m6s        Normal   Started            pod/nginx-replicaset-lvsvx     Started container nginx
6m15s       Normal   Scheduled          pod/nginx-replicaset-pfw9b     Successfully assigned quota-ex
ample/nginx-replicaset-pfw9b to kind-control-plane
6m14s       Normal   Pulling            pod/nginx-replicaset-pfw9b     Pulling image "nginx:latest"
6m12s       Normal   Pulled             pod/nginx-replicaset-pfw9b     Successfully pulled image "ngi
nx:latest" in 2.072s (2.072s including waiting). Image size: 72955450 bytes.
6m12s       Normal   Created            pod/nginx-replicaset-pfw9b     Created container nginx
6m12s       Normal   Started            pod/nginx-replicaset-pfw9b     Started container nginx
6m15s       Normal   Scheduled          pod/nginx-replicaset-ptt4j     Successfully assigned quota-ex
ample/nginx-replicaset-ptt4j to kind-control-plane
6m14s       Normal   Pulling            pod/nginx-replicaset-ptt4j     Pulling image "nginx:latest"
6m4s        Normal   Pulled             pod/nginx-replicaset-ptt4j     Successfully pulled image "ngi
nx:latest" in 1.999s (9.91s including waiting). Image size: 72955450 bytes.
6m4s        Normal   Created            pod/nginx-replicaset-ptt4j     Created container nginx
6m4s        Normal   Started            pod/nginx-replicaset-ptt4j     Started container nginx
6m15s       Normal   Scheduled          pod/nginx-replicaset-tnjhj     Successfully assigned quota-ex
ample/nginx-replicaset-tnjhj to kind-control-plane
6m14s       Normal   Pulling            pod/nginx-replicaset-tnjhj     Pulling image "nginx:latest"
6m8s        Normal   Pulled             pod/nginx-replicaset-tnjhj     Successfully pulled image "ngi
nx:latest" in 2.009s (5.767s including waiting). Image size: 72955450 bytes.
6m8s        Normal   Created            pod/nginx-replicaset-tnjhj     Created container nginx
6m8s        Normal   Started            pod/nginx-replicaset-tnjhj     Started container nginx
6m15s       Normal   Scheduled          pod/nginx-replicaset-zrfhq     Successfully assigned quota-ex
ample/nginx-replicaset-zrfhq to kind-control-plane
6m14s       Normal   Pulling            pod/nginx-replicaset-zrfhq     Pulling image "nginx:latest"
6m10s       Normal   Pulled             pod/nginx-replicaset-zrfhq     Successfully pulled image "ngi
nx:latest" in 1.704s (3.758s including waiting). Image size: 72955450 bytes.
6m10s       Normal   Created            pod/nginx-replicaset-zrfhq     Created container nginx
6m10s       Normal   Started            pod/nginx-replicaset-zrfhq     Started container nginx
6m15s       Normal   SuccessfulCreate   replicaset/nginx-replicaset    Created pod: nginx-replicaset-
zrfhq
6m15s       Normal   SuccessfulCreate   replicaset/nginx-replicaset    Created pod: nginx-replicaset-
tnjhj
6m15s       Normal   SuccessfulCreate   replicaset/nginx-replicaset    Created pod: nginx-replicaset-
ptt4j
6m15s       Normal   SuccessfulCreate   replicaset/nginx-replicaset    Created pod: nginx-replicaset-
pfw9b
6m15s       Normal   SuccessfulCreate   replicaset/nginx-replicaset    Created pod: nginx-replicaset-
lvsvx
```

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
```



```
sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES COLLEGE STUDY MATERIAL/SEM5/CDS/lab/exp10
$ [200~kubectl delete -f nginx-replicaset-quota.yaml
bash: [200~kubectl: command not found

sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES COLLEGE STUDY MATERIAL/SEM5/CDS/lab/exp10
$ kubectl delete -f nginx-extra-pod.yaml
Error from server (NotFound): error when deleting "nginx-extra-pod.yaml": pods "nginx-extra-pod" not
 found

sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES COLLEGE STUDY MATERIAL/SEM5/CDS/lab/exp10
$ kubectl delete -f resource-quota.yaml
resourcequota "example-quota" deleted

sidag@Sidzz-Yoga MINGW64 /c/SID_DATA/SIDDHARTH/UPES COLLEGE STUDY MATERIAL/SEM5/CDS/lab/exp10
$ kubectl delete namespace quota-example kubectl delete -f nginx-replicaset-quota.yaml
error: when paths, URLs, or stdin is provided as input, you may not specify resource arguments as we
ll
```