Name: Aarushi
SAP ID: 500105028
Rollno.: R2142220004
Batch: DevSecOps B1:H

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

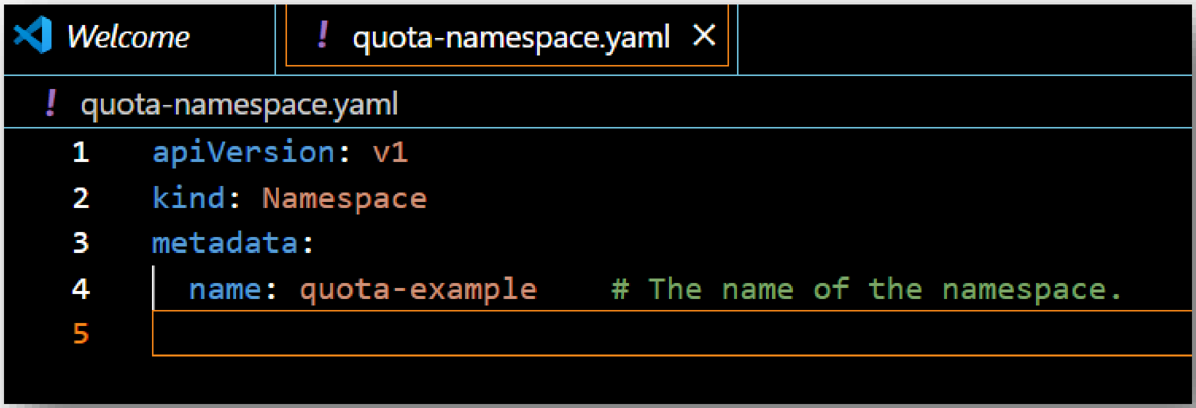
Resource Quotas allow you to:

- 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:

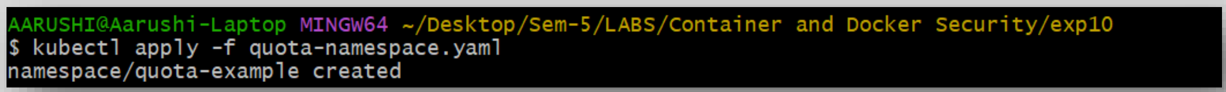
```
apiVersion: v1
kind: Namespace
metadata:
  name: quota-example    # The name of the namespace.
```



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

Apply the YAML to create the namespace:

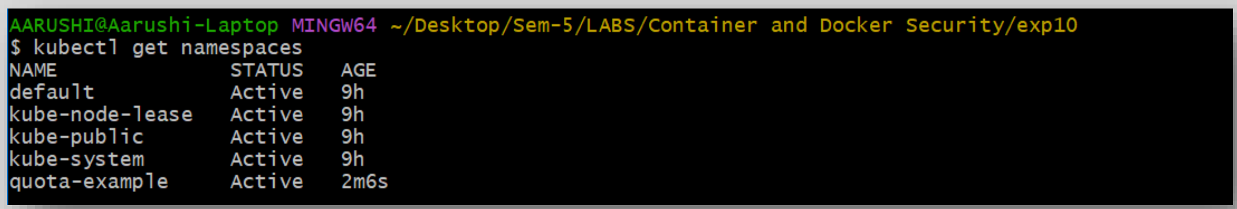
```
kubectl apply -f quota-namespace.yaml
```



```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl apply -f quota-namespace.yaml
namespace/quota-example created
```

Verify that the namespace is created:

```
kubectl get namespaces
```



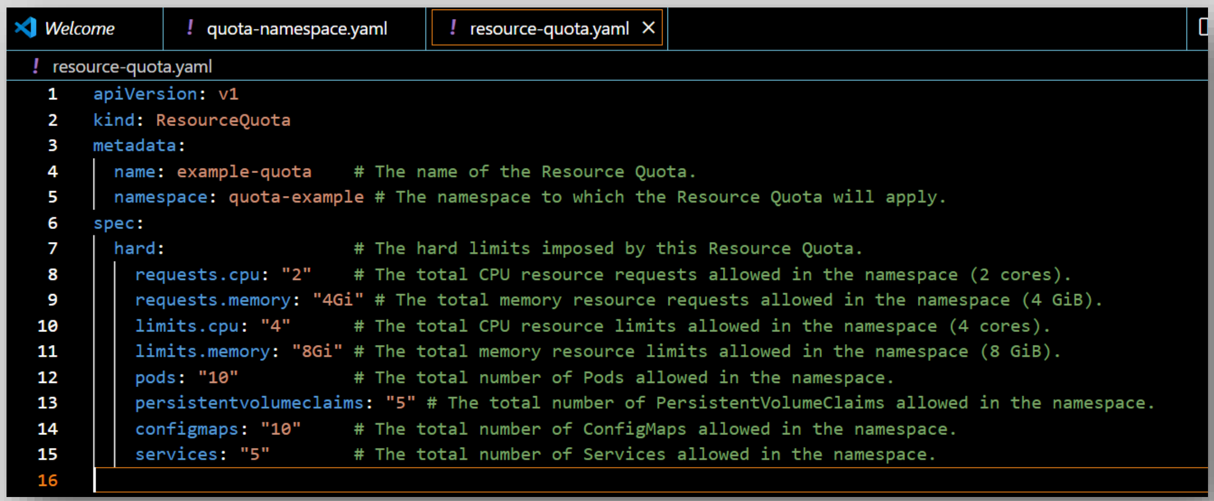
```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl get namespaces
NAME              STATUS   AGE
default           Active   9h
kube-node-lease   Active   9h
kube-public       Active   9h
kube-system       Active   9h
quota-example     Active   2m6s
```

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.
```

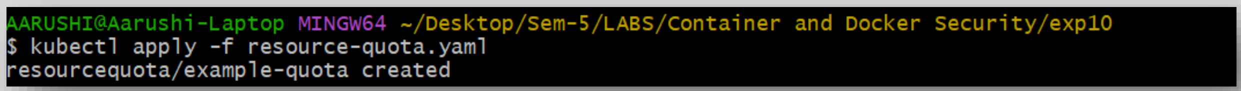


```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: example-quota    # The name of the Resource Quota.
  namespace: quota-example # The namespace to which the Resource Quota will apply.
spec:
  hard:                  # The hard limits imposed by this Resource Quota.
    requests.cpu: "2"    # The 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:

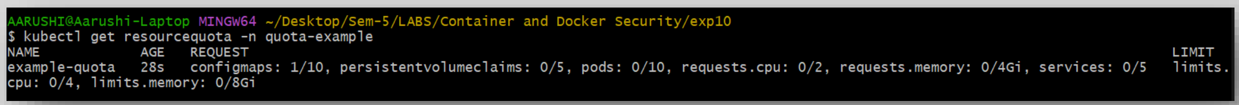
```
kubectl apply -f resource-quota.yaml
```



```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl apply -f resource-quota.yaml
resourcequota/example-quota created
```

Verify that the Resource Quota is applied:

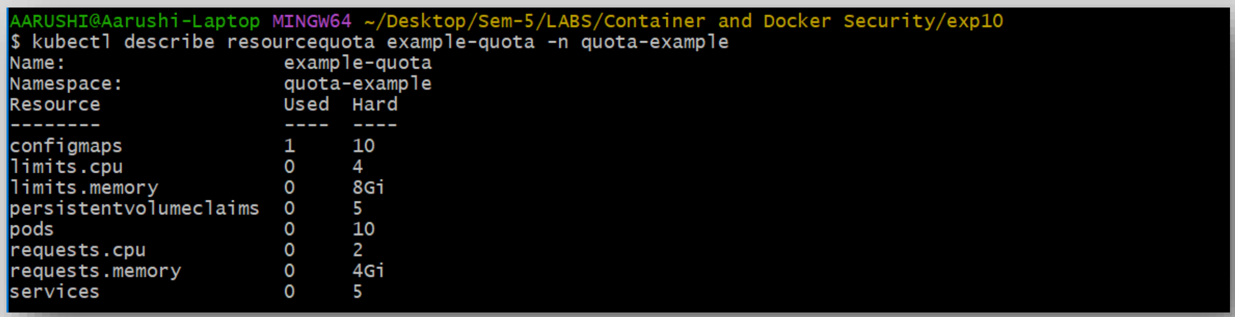
```
kubectl get resourcequota -n quota-example
```



```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl get resourcequota -n quota-example
NAME            AGE   REQUEST                                                                                                               LIMIT
example-quota   28s   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
```



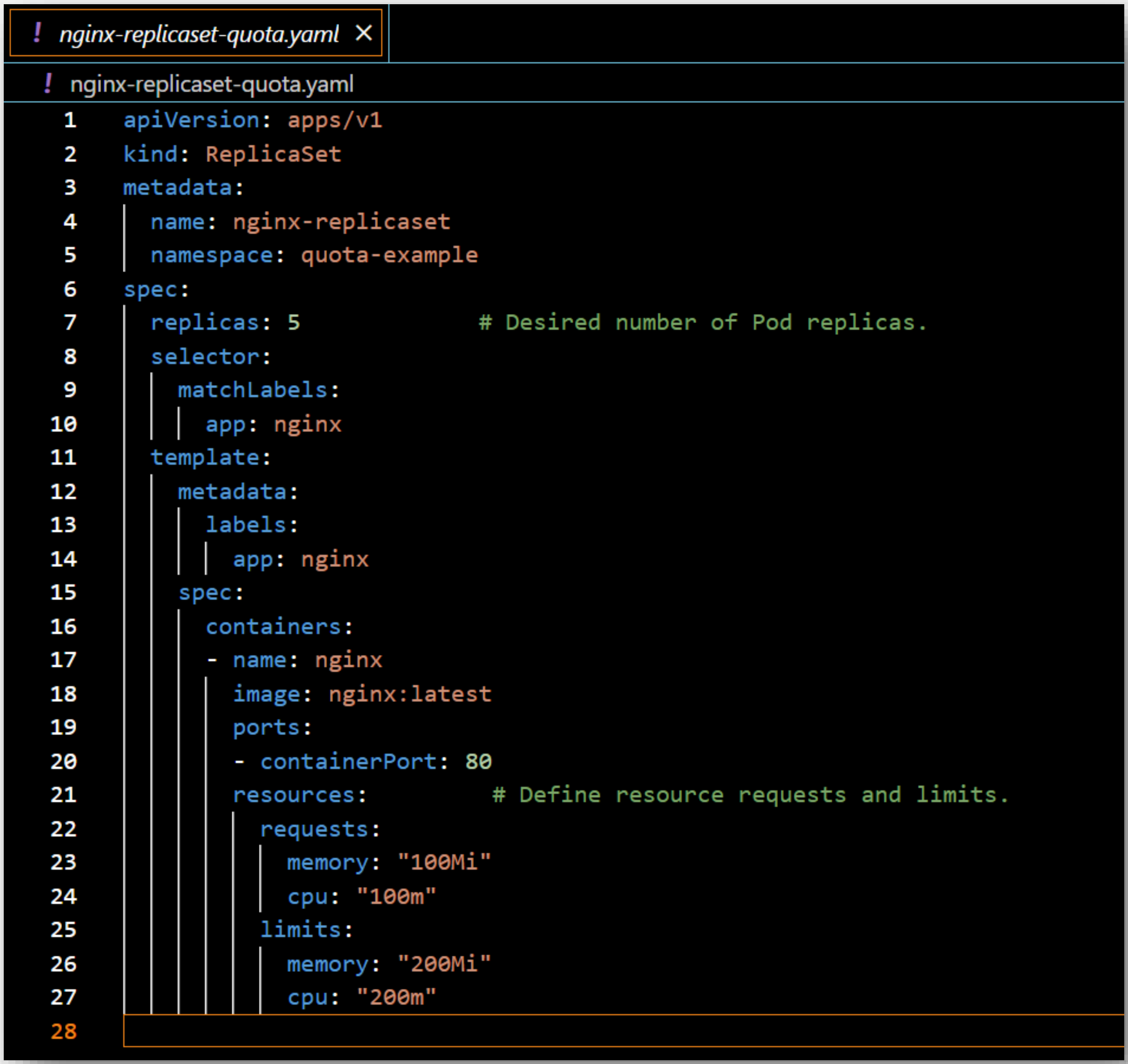
```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl describe resourcequota example-quota -n quota-example
Name:                   example-quota
Namespace:              quota-example
Resource                Used  Hard
--------                ----  ----
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
```

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

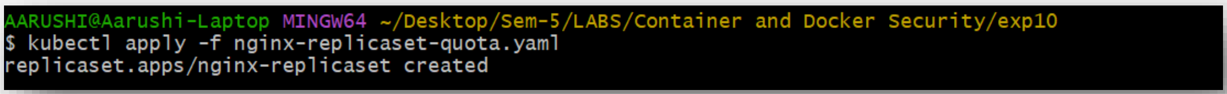
```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
  namespace: quota-example
spec:
  replicas: 5              # Desired number of Pod replicas.
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - 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:

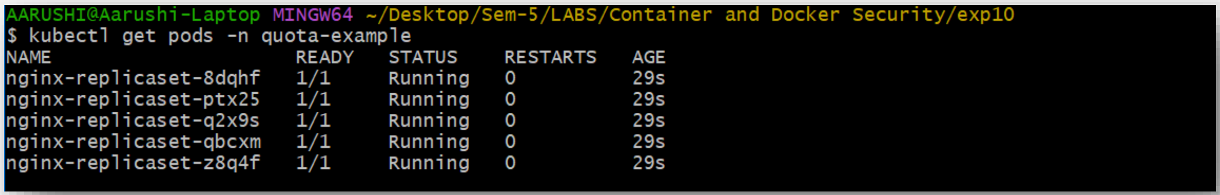
```
kubectl apply -f nginx-replicaset-quota.yaml
```



```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ 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:

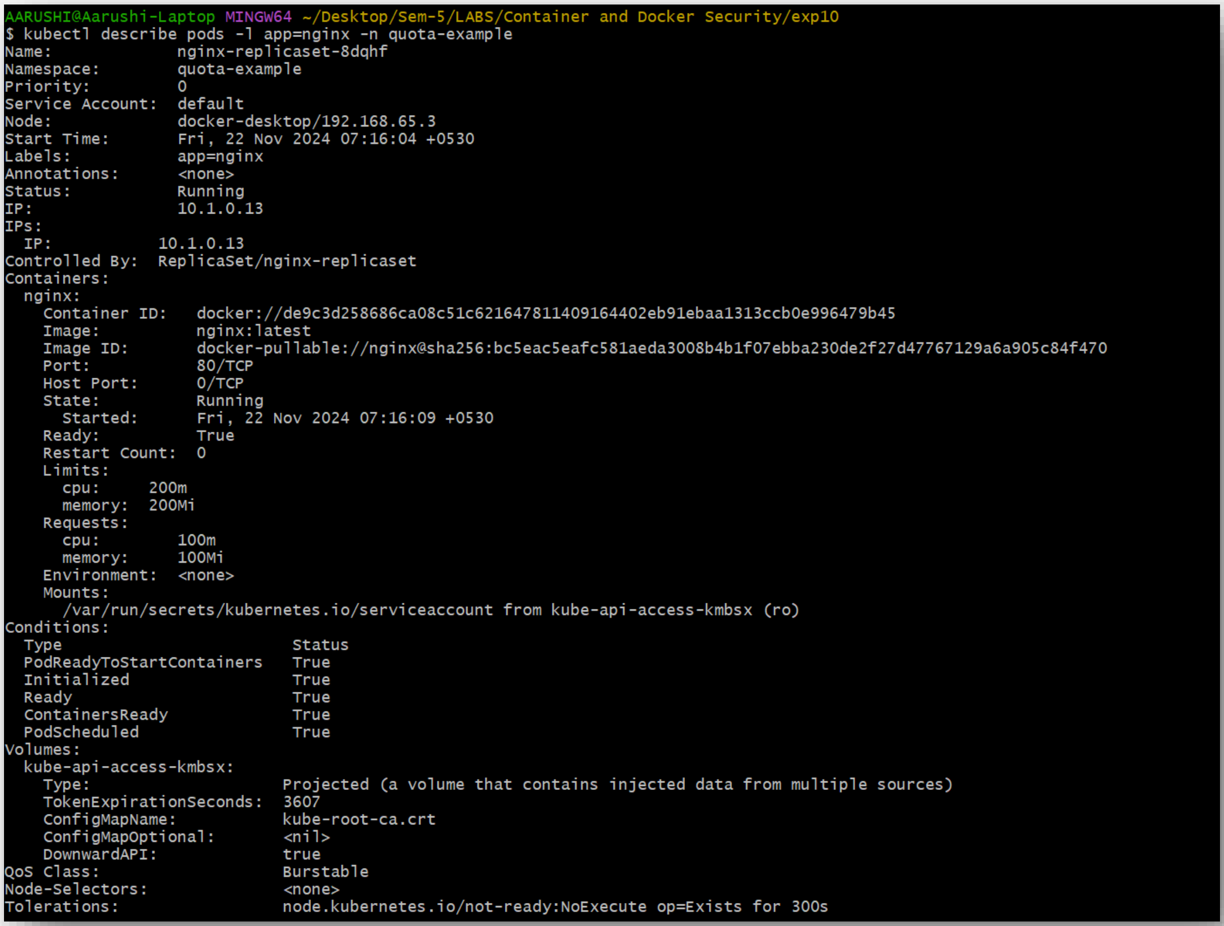
```
kubectl get pods -n quota-example
```



```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl get pods -n quota-example
NAME                     READY   STATUS    RESTARTS   AGE
nginx-replicaset-8dqhf   1/1     Running   0          29s
nginx-replicaset-ptx25   1/1     Running   0          29s
nginx-replicaset-q2x9s   1/1     Running   0          29s
nginx-replicaset-qbcxm   1/1     Running   0          29s
nginx-replicaset-z8q4f   1/1     Running   0          29s
```

To describe the Pods and see their resource allocations:

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



```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl describe pods -l app=nginx -n quota-example
Name:             nginx-replicaset-8dqhf
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Fri, 22 Nov 2024 07:16:04 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.13
IPs:
  IP:           10.1.0.13
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://de9c3d258686ca08c51c621647811409164402eb91ebaa1313ccb0e996479b45
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 22 Nov 2024 07:16:09 +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-kmbsx (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-kmbsx:
    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
```

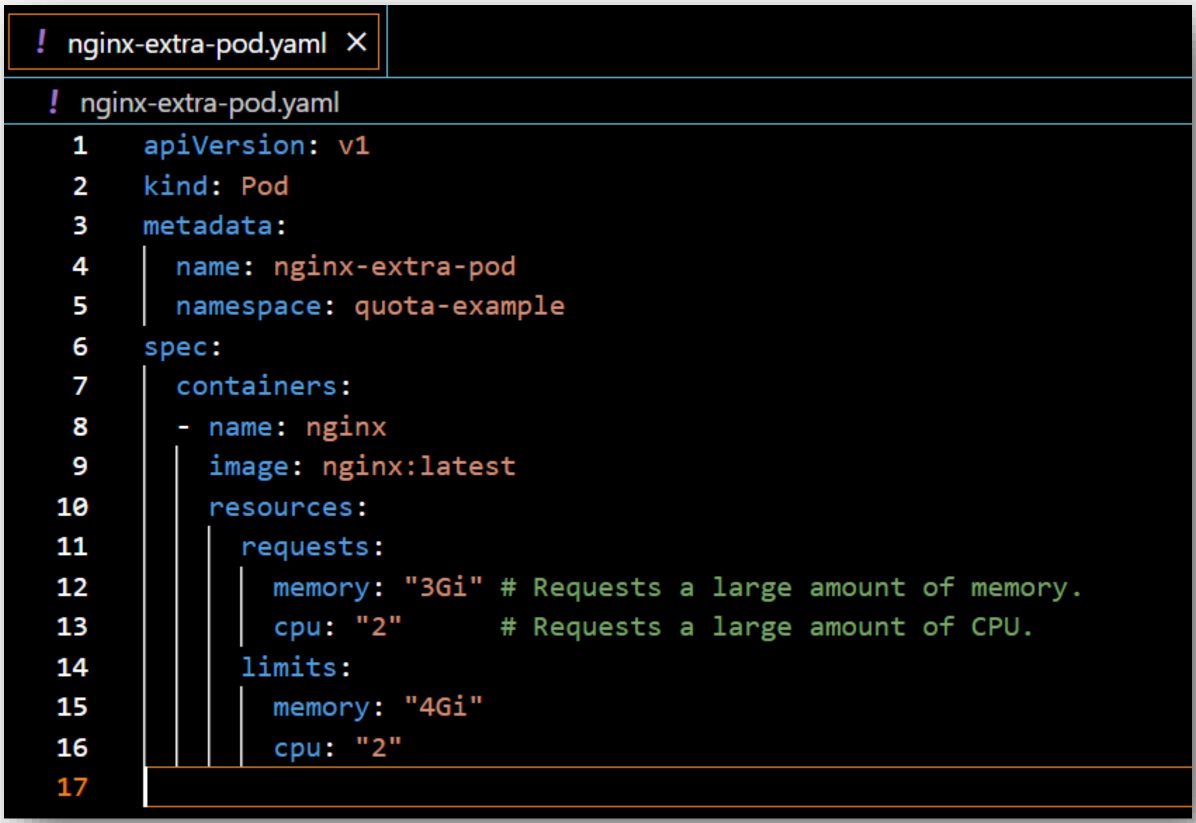
Attempt to Exceed the Resource Quota

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:
```

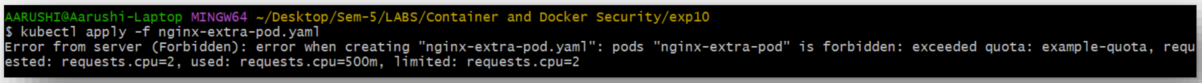
```yaml
  name: nginx-extra-pod
  namespace: quota-example
spec:
  containers:
  - name: nginx
    image: nginx:latest
    resources:
      requests:
        memory: "3Gi" # Requests a large amount of memory.
        cpu: "2"      # Requests a large amount of CPU.
      limits:
        memory: "4Gi"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

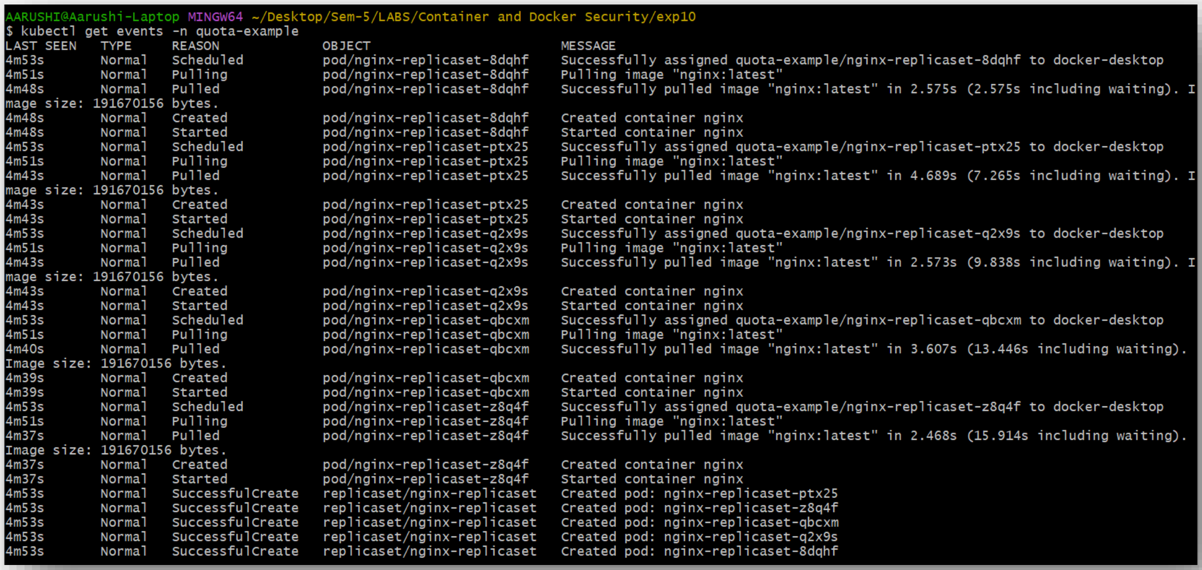
```
kubectl apply -f nginx-extra-pod.yaml
```



```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/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, requ
ested: 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
```



```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl get events -n quota-example
LAST SEEN   TYPE     REASON             OBJECT                          MESSAGE
4m53s       Normal   Scheduled          pod/nginx-replicaset-8dqhf      Successfully assigned quota-example/nginx-replicaset-8dqhf to docker-desktop
4m51s       Normal   Pulling            pod/nginx-replicaset-8dqhf      Pulling image "nginx:latest"
4m48s       Normal   Pulled             pod/nginx-replicaset-8dqhf      Successfully pulled image "nginx:latest" in 2.575s (2.575s including waiting). I
mage size: 191670156 bytes.
4m48s       Normal   Created            pod/nginx-replicaset-8dqhf      Created container nginx
4m48s       Normal   Started            pod/nginx-replicaset-8dqhf      Started container nginx
4m53s       Normal   Scheduled          pod/nginx-replicaset-ptx25      Successfully assigned quota-example/nginx-replicaset-ptx25 to docker-desktop
4m51s       Normal   Pulling            pod/nginx-replicaset-ptx25      Pulling image "nginx:latest"
4m43s       Normal   Pulled             pod/nginx-replicaset-ptx25      Successfully pulled image "nginx:latest" in 4.689s (7.265s including waiting). I
mage size: 191670156 bytes.
4m43s       Normal   Created            pod/nginx-replicaset-ptx25      Created container nginx
4m43s       Normal   Started            pod/nginx-replicaset-ptx25      Started container nginx
4m53s       Normal   Scheduled          pod/nginx-replicaset-q2x9s      Successfully assigned quota-example/nginx-replicaset-q2x9s to docker-desktop
4m51s       Normal   Pulling            pod/nginx-replicaset-q2x9s      Pulling image "nginx:latest"
4m43s       Normal   Pulled             pod/nginx-replicaset-q2x9s      Successfully pulled image "nginx:latest" in 2.573s (9.838s including waiting). I
mage size: 191670156 bytes.
4m43s       Normal   Created            pod/nginx-replicaset-q2x9s      Created container nginx
4m43s       Normal   Started            pod/nginx-replicaset-q2x9s      Started container nginx
4m53s       Normal   Scheduled          pod/nginx-replicaset-qbcxm      Successfully assigned quota-example/nginx-replicaset-qbcxm to docker-desktop
4m51s       Normal   Pulling            pod/nginx-replicaset-qbcxm      Pulling image "nginx:latest"
4m40s       Normal   Pulled             pod/nginx-replicaset-qbcxm      Successfully pulled image "nginx:latest" in 3.607s (13.446s including waiting).
Image size: 191670156 bytes.
4m39s       Normal   Created            pod/nginx-replicaset-qbcxm      Created container nginx
4m39s       Normal   Started            pod/nginx-replicaset-qbcxm      Started container nginx
4m53s       Normal   Scheduled          pod/nginx-replicaset-z8q4f      Successfully assigned quota-example/nginx-replicaset-z8q4f to docker-desktop
4m51s       Normal   Pulling            pod/nginx-replicaset-z8q4f      Pulling image "nginx:latest"
4m37s       Normal   Pulled             pod/nginx-replicaset-z8q4f      Successfully pulled image "nginx:latest" in 2.468s (15.914s including waiting).
Image size: 191670156 bytes.
4m37s       Normal   Created            pod/nginx-replicaset-z8q4f      Created container nginx
4m37s       Normal   Started            pod/nginx-replicaset-z8q4f      Started container nginx
4m53s       Normal   SuccessfulCreate   replicaset/nginx-replicaset     Created pod: nginx-replicaset-ptx25
4m53s       Normal   SuccessfulCreate   replicaset/nginx-replicaset     Created pod: nginx-replicaset-z8q4f
4m53s       Normal   SuccessfulCreate   replicaset/nginx-replicaset     Created pod: nginx-replicaset-qbcxm
4m53s       Normal   SuccessfulCreate   replicaset/nginx-replicaset     Created pod: nginx-replicaset-q2x9s
4m53s       Normal   SuccessfulCreate   replicaset/nginx-replicaset     Created pod: nginx-replicaset-8dqhf
```

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

Step 6: Clean Up Resources

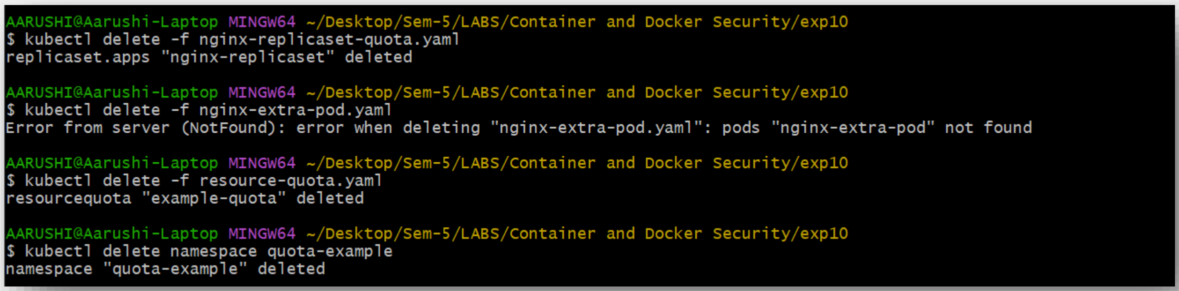
To delete the resources you created:

```
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



```
AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl delete -f nginx-replicaset-quota.yaml
replicaset.apps "nginx-replicaset" deleted

AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/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

AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl delete -f resource-quota.yaml
resourcequota "example-quota" deleted

AARUSHI@Aarushi-Laptop MINGW64 ~/Desktop/Sem-5/LABS/Container and Docker Security/exp10
$ kubectl delete namespace quota-example
namespace "quota-example" deleted
```