Lab Exercise 14- Implementing Resource Quota in Kubernetes

Name – Vishal Pandey
500125280
B2 DevOps

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:

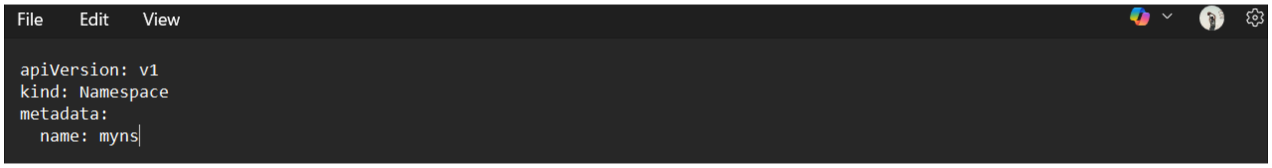
- Limit the amount of CPU and memory a namespace can use.
- Control the number of certain types of resources (e.g., Pods, Services, PersistentVolumeClaims) in a namespace.
- Prevent a namespace from consuming more resources than allocated, ensuring fair usage across multiple teams or applications.

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:

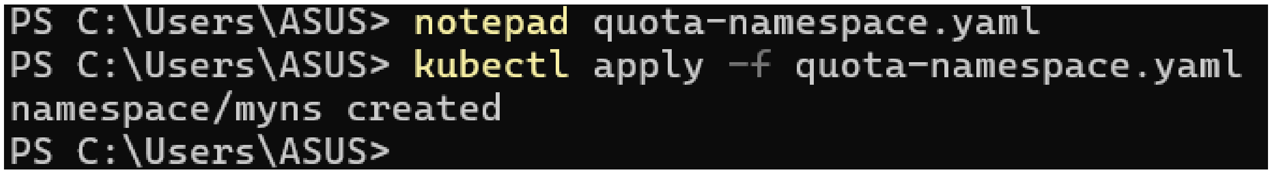
```
apiVersion: v1
kind: Namespace
metadata:
  name: myns
```



```
apiVersion: v1
kind: Namespace
metadata:
  name: myns
```

Apply the YAML to create the namespace:

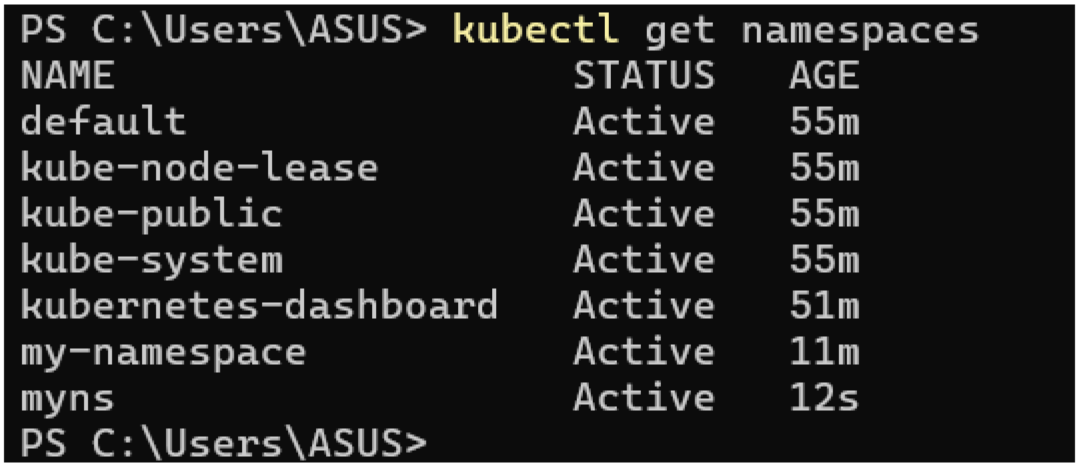
```
kubectl apply -f quota-namespace.yaml
```



```
PS C:\Users\ASUS> notepad quota-namespace.yaml
PS C:\Users\ASUS> kubectl apply -f quota-namespace.yaml
namespace/myns created
PS C:\Users\ASUS>
```

Verify that the namespace is created:

```
kubectl get namespaces
```



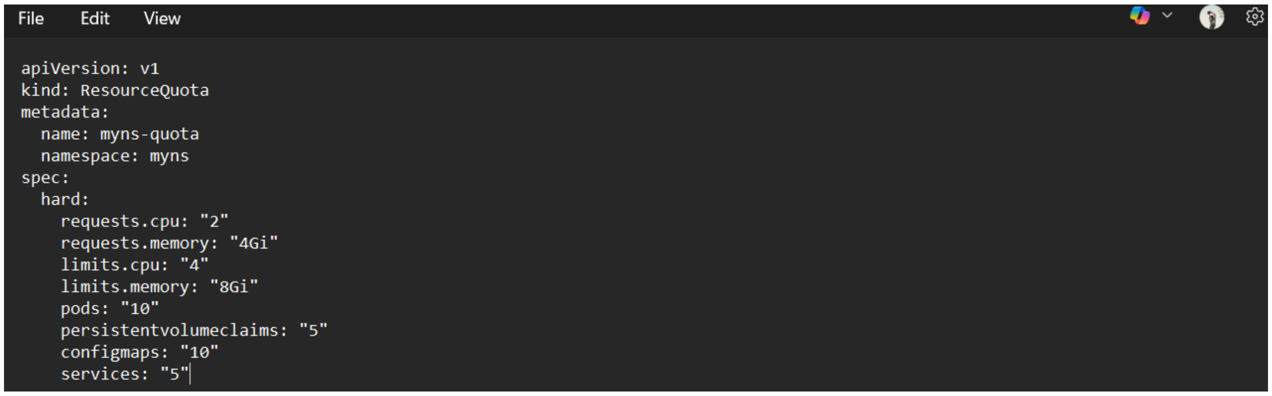
```
PS C:\Users\ASUS> kubectl get namespaces
NAME                   STATUS   AGE
default                Active   55m
kube-node-lease        Active   55m
kube-public            Active   55m
kube-system            Active   55m
kubernetes-dashboard   Active   51m
my-namespace           Active   11m
myns                   Active   12s
PS C:\Users\ASUS>
```

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: myns-quota    # The name of the Resource Quota.
  namespace: myns # 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.
```

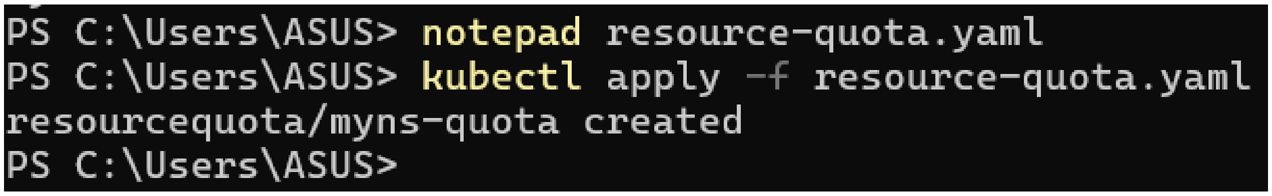


```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: myns-quota
  namespace: myns
spec:
  hard:
    requests.cpu: "2"
    requests.memory: "4Gi"
    limits.cpu: "4"
    limits.memory: "8Gi"
    pods: "10"
    persistentvolumeclaims: "5"
    configmaps: "10"
    services: "5"
```

Step 4: Apply the Resource Quota

Apply the Resource Quota YAML to the namespace:

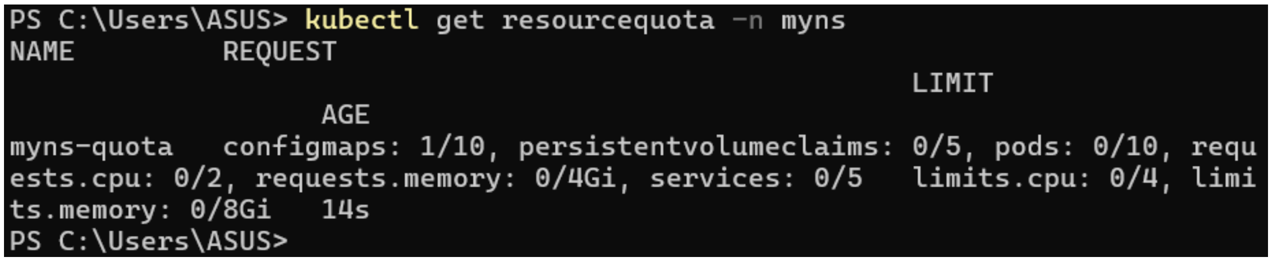
```
kubectl apply -f resource-quota.yaml
```



```
PS C:\Users\ASUS> notepad resource-quota.yaml
PS C:\Users\ASUS> kubectl apply -f resource-quota.yaml
resourcequota/myns-quota created
PS C:\Users\ASUS>
```

Verify that the Resource Quota is applied:

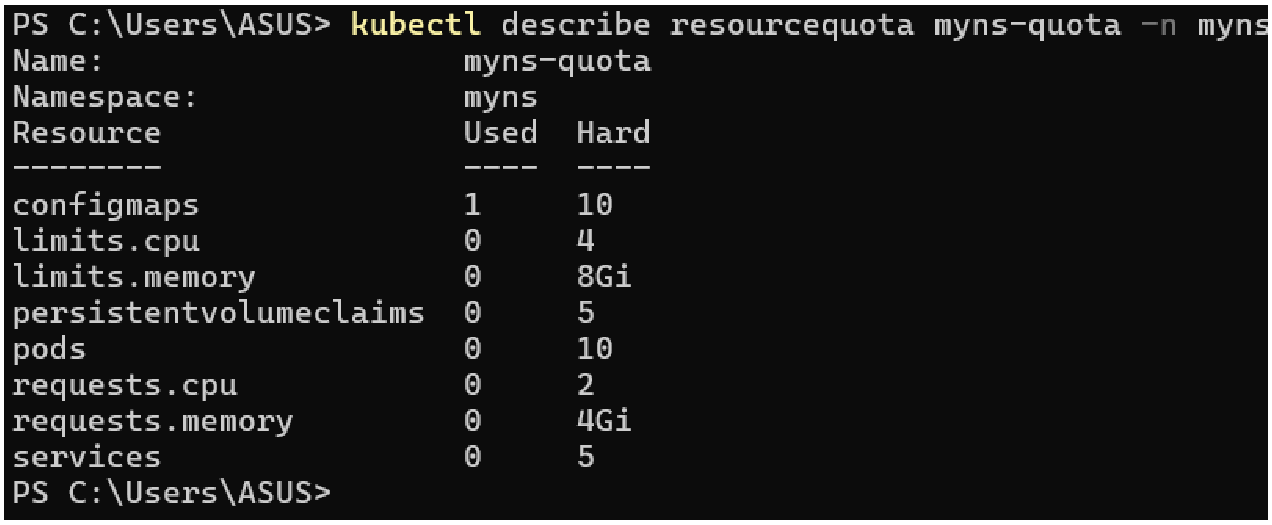
```
kubectl get resourcequota -n myns
```



```
PS C:\Users\ASUS> kubectl get resourcequota -n myns
NAME         REQUEST
                                                           LIMIT
                    AGE
myns-quota   configmaps: 1/10, persistentvolumeclaims: 0/5, pods: 0/10, requ
ests.cpu: 0/2, requests.memory: 0/4Gi, services: 0/5   limits.cpu: 0/4, limi
ts.memory: 0/8Gi   14s
PS C:\Users\ASUS>
```

To see the details of the applied Resource Quota:

```
kubectl describe resourcequota myns-quota -n myns
```



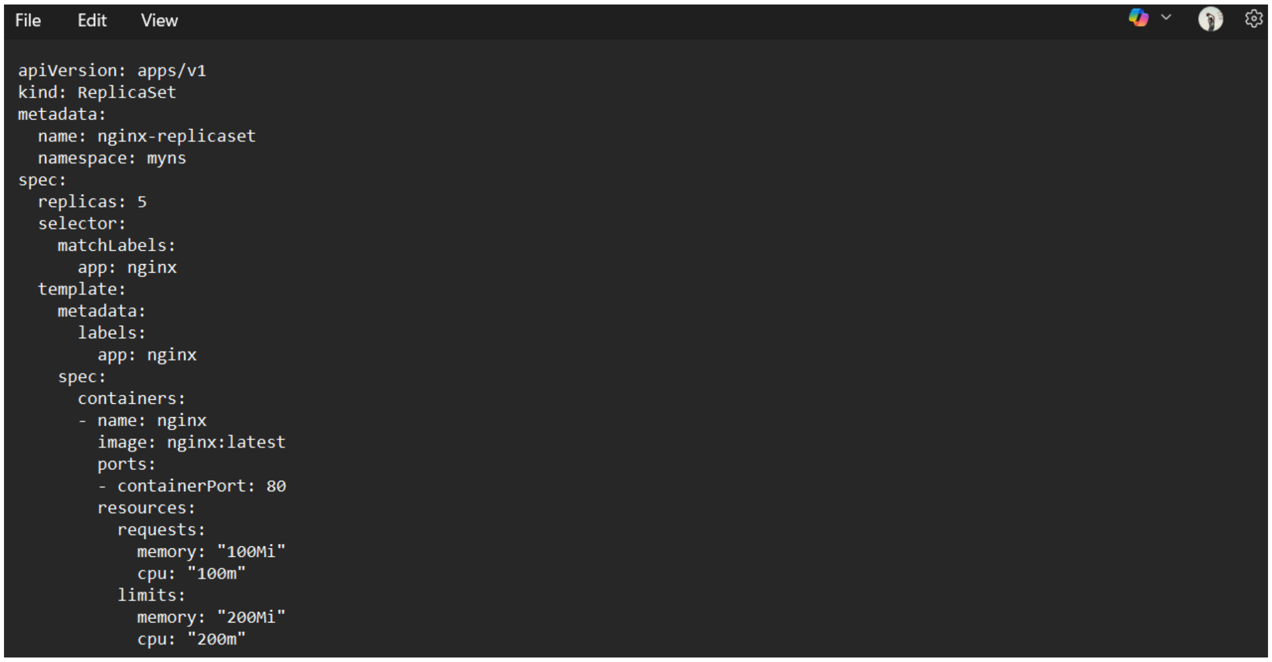
```
PS C:\Users\ASUS> kubectl describe resourcequota myns-quota -n myns
Name:                   myns-quota
Namespace:              myns
Resource                Used  Hard
--------                ----  ----
configmaps              1     10
limits.cpu              0     4
limits.memory           0     8Gi
persistentvolumeclaims  0     5
pods                    0     10
requests.cpu            0     2
requests.memory         0     4Gi
services                0     5
PS C:\Users\ASUS>
```

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: myns
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"
```



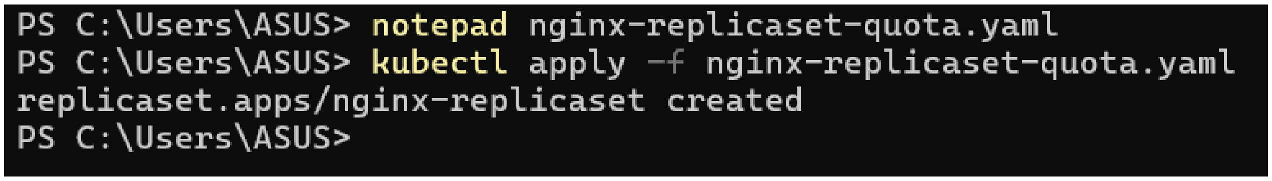
```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
  namespace: myns
spec:
  replicas: 5
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        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:

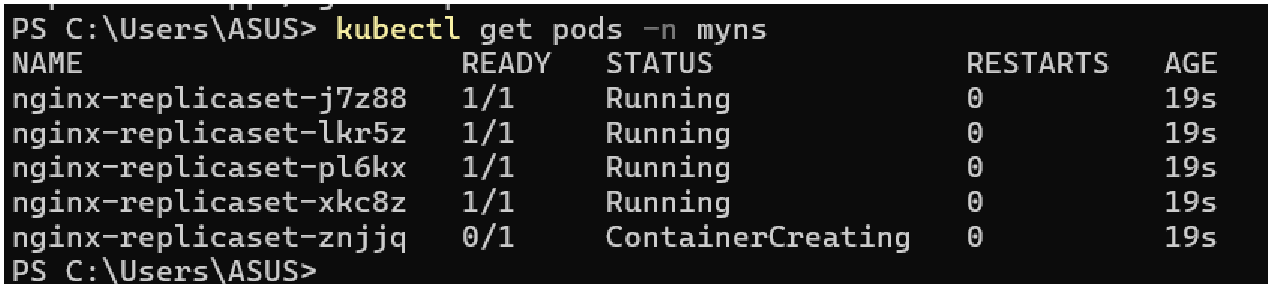
```
kubectl apply -f nginx-replicaset-quota.yaml
```



```
PS C:\Users\ASUS> notepad nginx-replicaset-quota.yaml
PS C:\Users\ASUS> kubectl apply -f nginx-replicaset-quota.yaml
replicaset.apps/nginx-replicaset created
PS C:\Users\ASUS>
```

Check the status of the Pods and ensure they are created within the constraints of the Resource Quota:

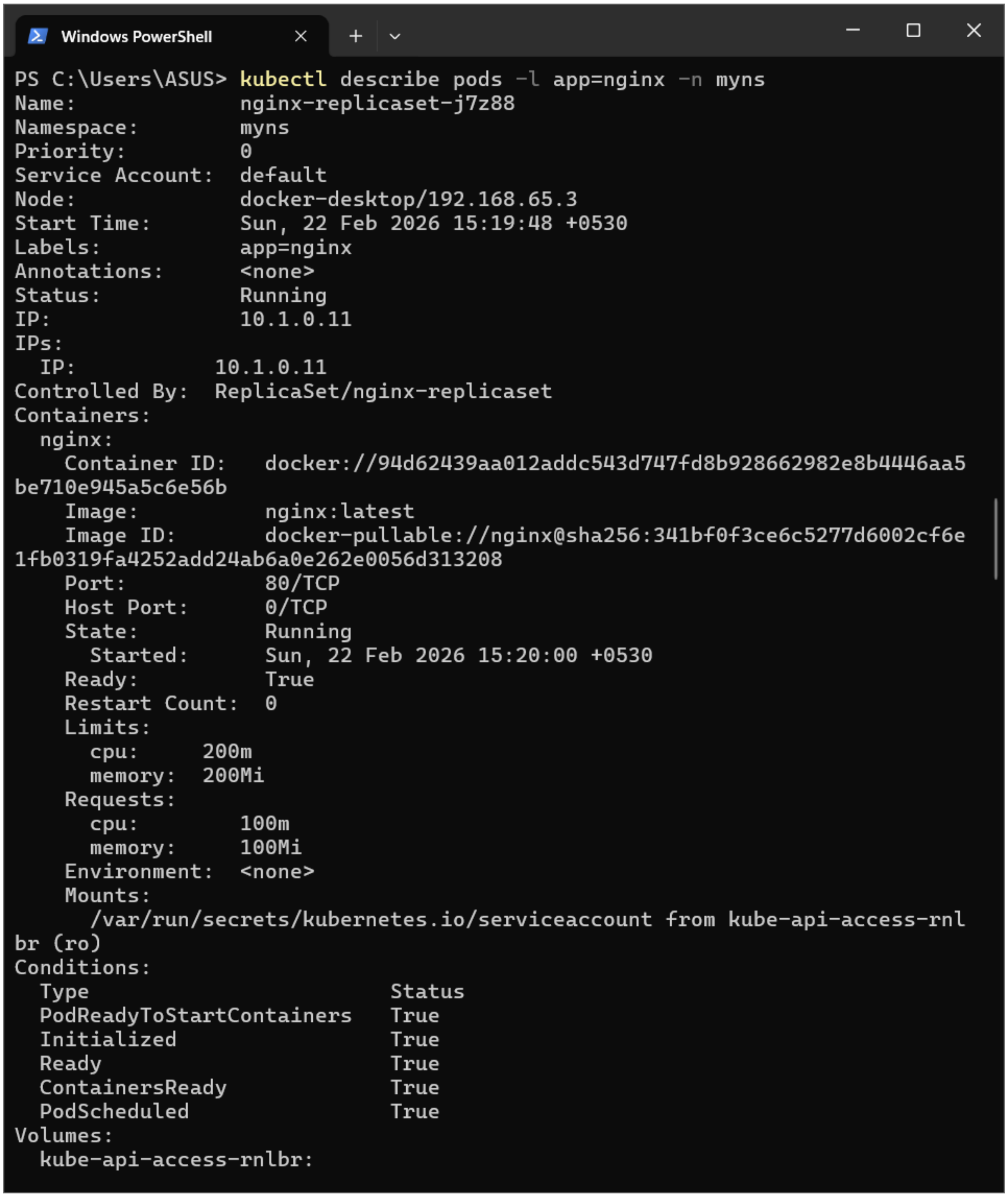
```
kubectl get pods -n myns
```



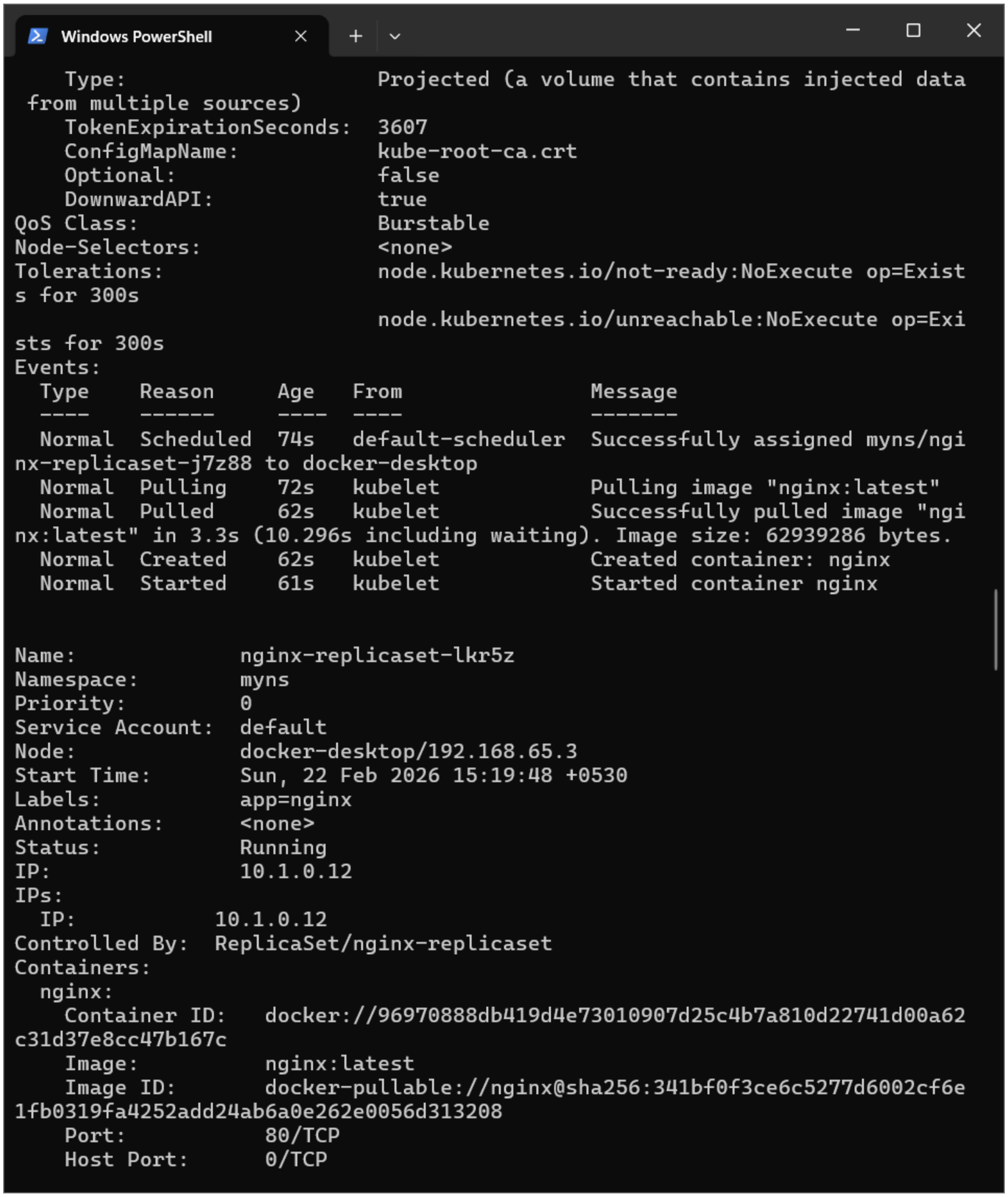
```
PS C:\Users\ASUS> kubectl get pods -n myns
NAME                       READY   STATUS              RESTARTS   AGE
nginx-replicaset-j7z88     1/1     Running             0          19s
nginx-replicaset-lkr5z     1/1     Running             0          19s
nginx-replicaset-pl6kx     1/1     Running             0          19s
nginx-replicaset-xkc8z     1/1     Running             0          19s
nginx-replicaset-znjjq     0/1     ContainerCreating   0          19s
PS C:\Users\ASUS>
```

To describe the Pods and see their resource allocations:

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

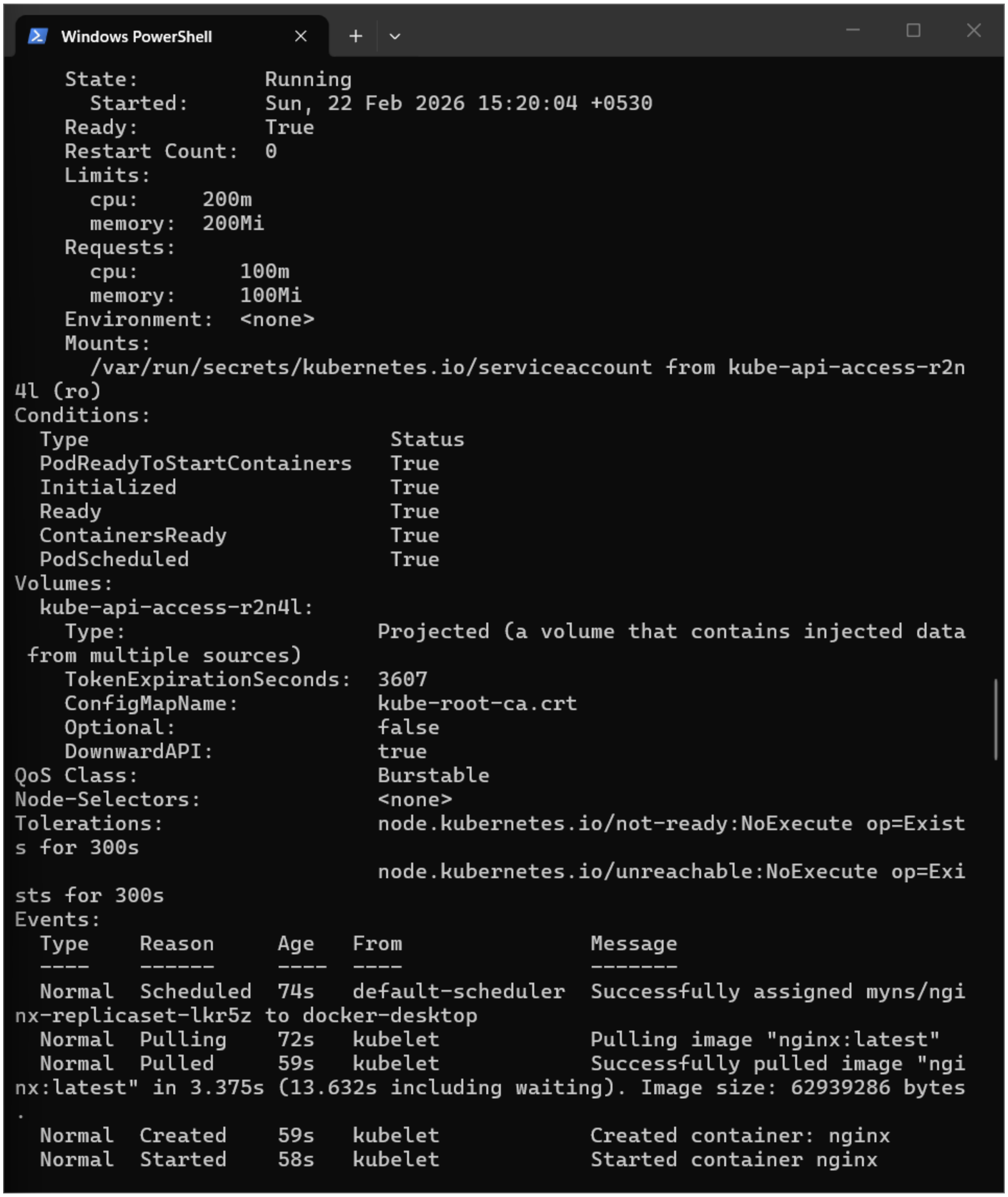

Windows PowerShell

```
PS C:\Users\ASUS> kubectl describe pods -l app=nginx -n myns
Name:             nginx-replicaset-j7z88
Namespace:        myns
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Sun, 22 Feb 2026 15:19:48 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.11
IPs:
  IP:           10.1.0.11
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://94d62439aa012addc543d747fd8b928662982e8b4446aa5
be710e945a5c6e56b
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:341bf0f3ce6c5277d6002cf6e
1fb0319fa4252add24ab6a0e262e0056d313208
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 22 Feb 2026 15:20:00 +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-rnl
br (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-rnlbr:
```

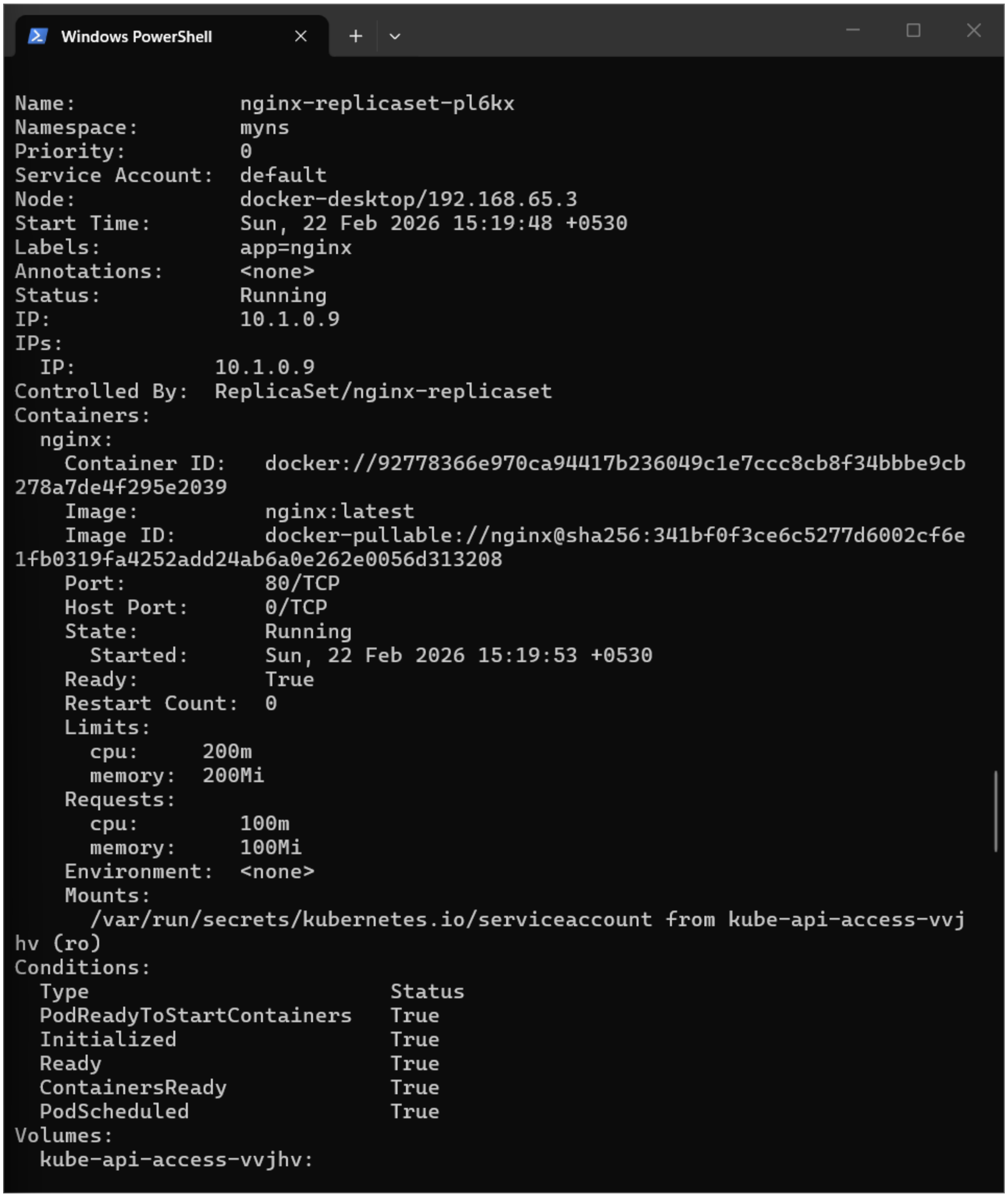


```
    Type:                    Projected (a volume that contains injected data
 from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    Optional:                false
    DownwardAPI:             true
QoS Class:                   Burstable
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exist
s for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exi
sts for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  74s   default-scheduler  Successfully assigned myns/ngi
nx-replicaset-j7z88 to docker-desktop
  Normal  Pulling    72s   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     62s   kubelet            Successfully pulled image "ngi
nx:latest" in 3.3s (10.296s including waiting). Image size: 62939286 bytes.
  Normal  Created    62s   kubelet            Created container: nginx
  Normal  Started    61s   kubelet            Started container nginx


Name:             nginx-replicaset-lkr5z
Namespace:        myns
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Sun, 22 Feb 2026 15:19:48 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.12
IPs:
  IP:           10.1.0.12
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://96970888db419d4e73010907d25c4b7a810d22741d00a62
c31d37e8cc47b167c
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:341bf0f3ce6c5277d6002cf6e
1fb0319fa4252add24ab6a0e262e0056d313208
    Port:           80/TCP
    Host Port:      0/TCP
```

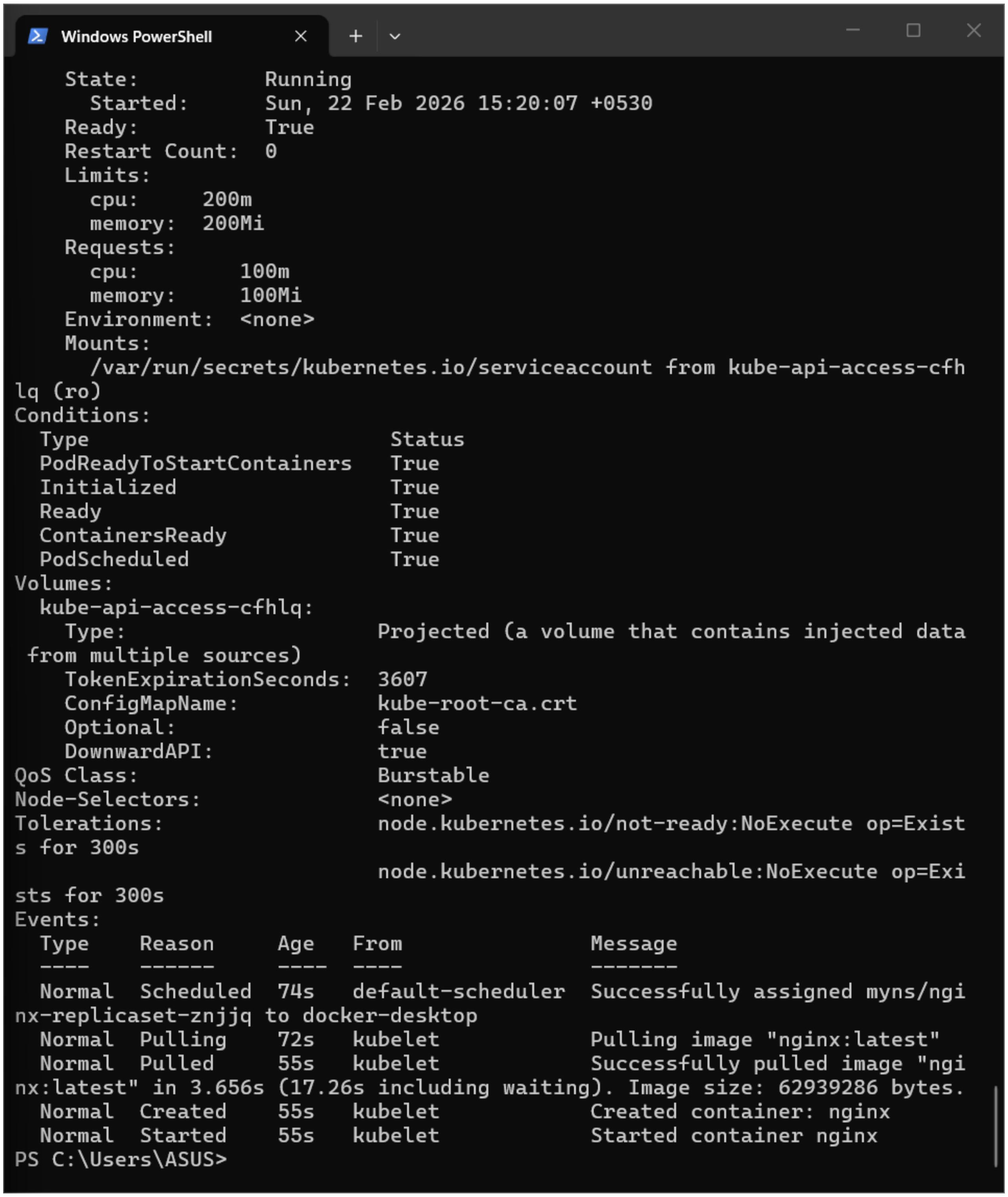

Windows PowerShell

```
    State:          Running
      Started:      Sun, 22 Feb 2026 15:20:04 +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-r2n
4l (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-r2n4l:
    Type:                    Projected (a volume that contains injected data
 from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    Optional:                false
    DownwardAPI:             true
QoS Class:                   Burstable
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exist
s for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exi
sts for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  74s   default-scheduler  Successfully assigned myns/ngi
nx-replicaset-lkr5z to docker-desktop
  Normal  Pulling    72s   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     59s   kubelet            Successfully pulled image "ngi
nx:latest" in 3.375s (13.632s including waiting). Image size: 62939286 bytes
.
  Normal  Created    59s   kubelet            Created container: nginx
  Normal  Started    58s   kubelet            Started container nginx
```



Windows PowerShell

```
Name:             nginx-replicaset-pl6kx
Namespace:        myns
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.3
Start Time:       Sun, 22 Feb 2026 15:19:48 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.1.0.9
IPs:
  IP:           10.1.0.9
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://92778366e970ca94417b236049c1e7ccc8cb8f34bbbe9cb
278a7de4f295e2039
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:341bf0f3ce6c5277d6002cf6e
1fb0319fa4252add24ab6a0e262e0056d313208
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 22 Feb 2026 15:19:53 +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-vvj
hv (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-vvjhv:
```



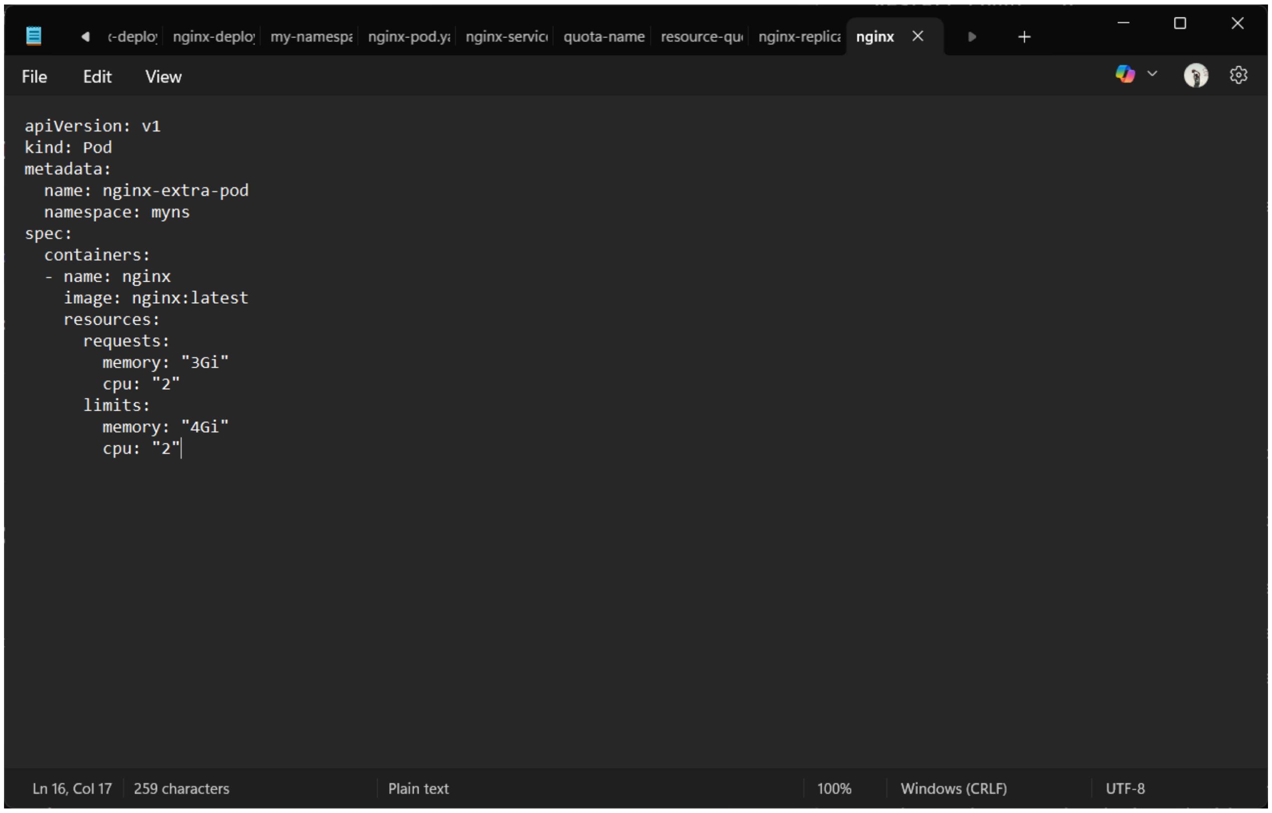
```
    State:          Running
      Started:      Sun, 22 Feb 2026 15:20:07 +0530
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     200m
      memory:  200Mi
    Requests:
      cpu:        100m
      memory:     100Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-cfh
lq (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-cfhlq:
    Type:                    Projected (a volume that contains injected data
 from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    Optional:                false
    DownwardAPI:             true
QoS Class:                   Burstable
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exist
s for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exi
sts for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  74s   default-scheduler  Successfully assigned myns/ngi
nx-replicaset-znjjq to docker-desktop
  Normal  Pulling    72s   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     55s   kubelet            Successfully pulled image "ngi
nx:latest" in 3.656s (17.26s including waiting). Image size: 62939286 bytes.
  Normal  Created    55s   kubelet            Created container: nginx
  Normal  Started    55s   kubelet            Started container nginx
PS C:\Users\ASUS>
```

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:

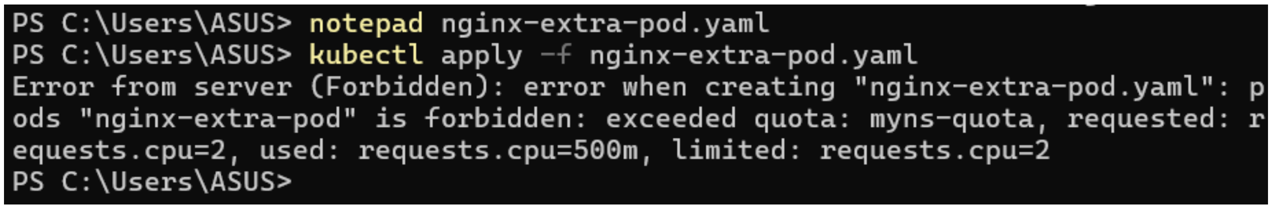
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-extra-pod
  namespace: myns
spec:
  containers:
  - name: nginx
    image: nginx:latest
    resources:
      requests:
        memory: "3Gi" # Requests a large amount of memory.
        cpu: "2"      # Requests a large amount of CPU.
      limits: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



```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-extra-pod
  namespace: myns
spec:
  containers:
  - name: nginx
    image: nginx:latest
    resources:
      requests:
        memory: "3Gi"
        cpu: "2"
      limits:
        memory: "4Gi"
        cpu: "2"
```

Apply this YAML to create the Pod:

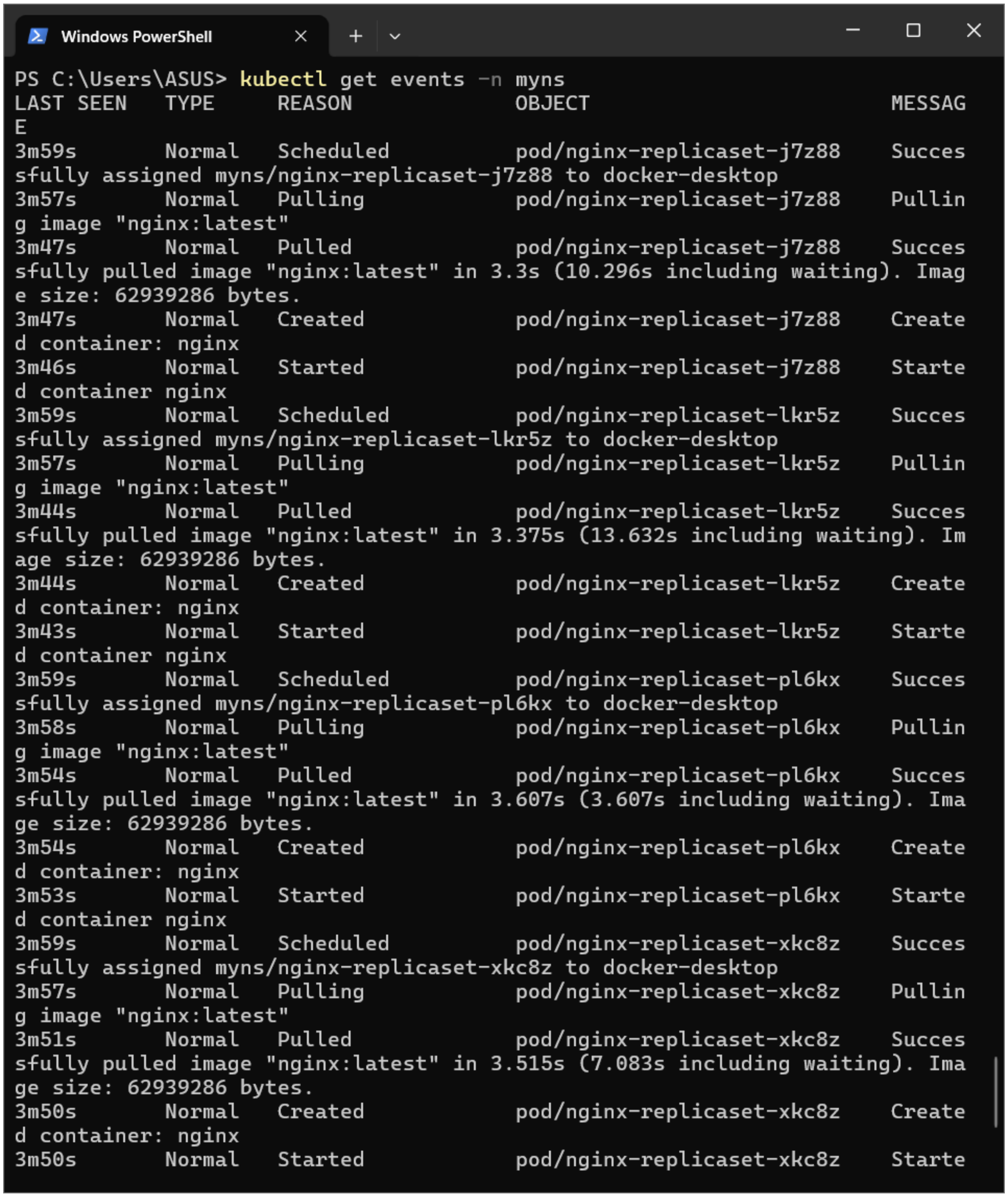
```
kubectl apply -f nginx-extra-pod.yaml
```



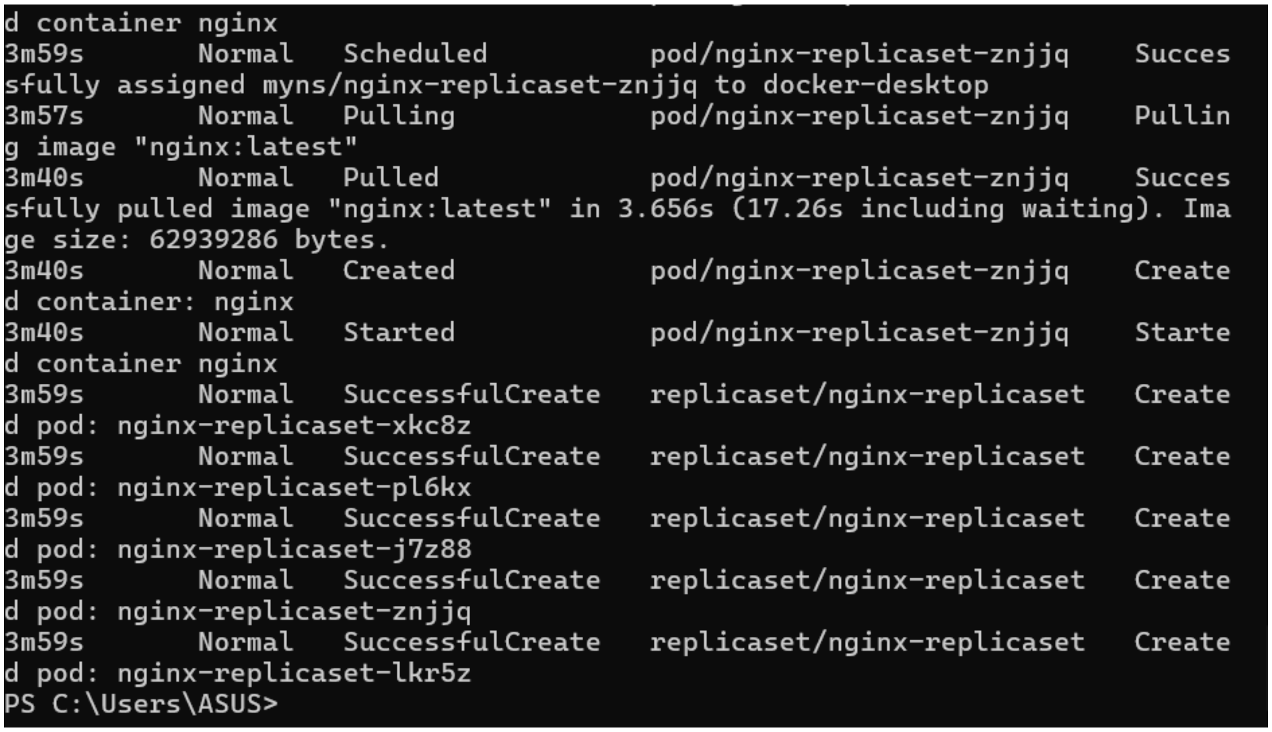
```
PS C:\Users\ASUS> notepad nginx-extra-pod.yaml
PS C:\Users\ASUS> kubectl apply -f nginx-extra-pod.yaml
Error from server (Forbidden): error when creating "nginx-extra-pod.yaml": p
ods "nginx-extra-pod" is forbidden: exceeded quota: myns-quota, requested: r
equests.cpu=2, used: requests.cpu=500m, limited: requests.cpu=2
PS C:\Users\ASUS>
```

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

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


Windows PowerShell

```
PS C:\Users\ASUS> kubectl get events -n myns
LAST SEEN   TYPE     REASON             OBJECT                        MESSAG
E
3m59s       Normal   Scheduled          pod/nginx-replicaset-j7z88    Succes
sfully assigned myns/nginx-replicaset-j7z88 to docker-desktop
3m57s       Normal   Pulling            pod/nginx-replicaset-j7z88    Pullin
g image "nginx:latest"
3m47s       Normal   Pulled             pod/nginx-replicaset-j7z88    Succes
sfully pulled image "nginx:latest" in 3.3s (10.296s including waiting). Imag
e size: 62939286 bytes.
3m47s       Normal   Created            pod/nginx-replicaset-j7z88    Create
d container: nginx
3m46s       Normal   Started            pod/nginx-replicaset-j7z88    Starte
d container nginx
3m59s       Normal   Scheduled          pod/nginx-replicaset-lkr5z    Succes
sfully assigned myns/nginx-replicaset-lkr5z to docker-desktop
3m57s       Normal   Pulling            pod/nginx-replicaset-lkr5z    Pullin
g image "nginx:latest"
3m44s       Normal   Pulled             pod/nginx-replicaset-lkr5z    Succes
sfully pulled image "nginx:latest" in 3.375s (13.632s including waiting). Im
age size: 62939286 bytes.
3m44s       Normal   Created            pod/nginx-replicaset-lkr5z    Create
d container: nginx
3m43s       Normal   Started            pod/nginx-replicaset-lkr5z    Starte
d container nginx
3m59s       Normal   Scheduled          pod/nginx-replicaset-pl6kx    Succes
sfully assigned myns/nginx-replicaset-pl6kx to docker-desktop
3m58s       Normal   Pulling            pod/nginx-replicaset-pl6kx    Pullin
g image "nginx:latest"
3m54s       Normal   Pulled             pod/nginx-replicaset-pl6kx    Succes
sfully pulled image "nginx:latest" in 3.607s (3.607s including waiting). Ima
ge size: 62939286 bytes.
3m54s       Normal   Created            pod/nginx-replicaset-pl6kx    Create
d container: nginx
3m53s       Normal   Started            pod/nginx-replicaset-pl6kx    Starte
d container nginx
3m59s       Normal   Scheduled          pod/nginx-replicaset-xkc8z    Succes
sfully assigned myns/nginx-replicaset-xkc8z to docker-desktop
3m57s       Normal   Pulling            pod/nginx-replicaset-xkc8z    Pullin
g image "nginx:latest"
3m51s       Normal   Pulled             pod/nginx-replicaset-xkc8z    Succes
sfully pulled image "nginx:latest" in 3.515s (7.083s including waiting). Ima
ge size: 62939286 bytes.
3m50s       Normal   Created            pod/nginx-replicaset-xkc8z    Create
d container: nginx
3m50s       Normal   Started            pod/nginx-replicaset-xkc8z    Starte
```

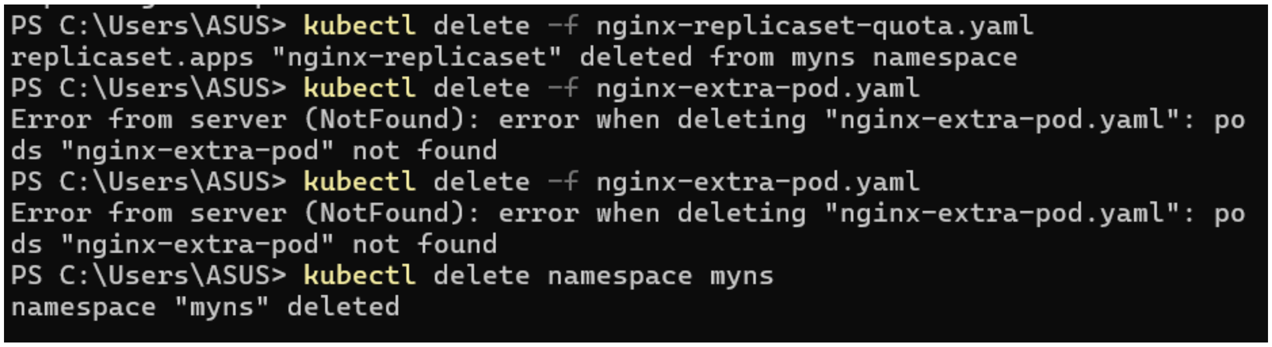

```
d container nginx
3m59s       Normal   Scheduled          pod/nginx-replicaset-znjjq    Succes
sfully assigned myns/nginx-replicaset-znjjq to docker-desktop
3m57s       Normal   Pulling            pod/nginx-replicaset-znjjq    Pullin
g image "nginx:latest"
3m40s       Normal   Pulled             pod/nginx-replicaset-znjjq    Succes
sfully pulled image "nginx:latest" in 3.656s (17.26s including waiting). Ima
ge size: 62939286 bytes.
3m40s       Normal   Created            pod/nginx-replicaset-znjjq    Create
d container: nginx
3m40s       Normal   Started            pod/nginx-replicaset-znjjq    Starte
d container nginx
3m59s       Normal   SuccessfulCreate   replicaset/nginx-replicaset   Create
d pod: nginx-replicaset-xkc8z
3m59s       Normal   SuccessfulCreate   replicaset/nginx-replicaset   Create
d pod: nginx-replicaset-pl6kx
3m59s       Normal   SuccessfulCreate   replicaset/nginx-replicaset   Create
d pod: nginx-replicaset-j7z88
3m59s       Normal   SuccessfulCreate   replicaset/nginx-replicaset   Create
d pod: nginx-replicaset-znjjq
3m59s       Normal   SuccessfulCreate   replicaset/nginx-replicaset   Create
d pod: nginx-replicaset-lkr5z
PS C:\Users\ASUS>
```

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

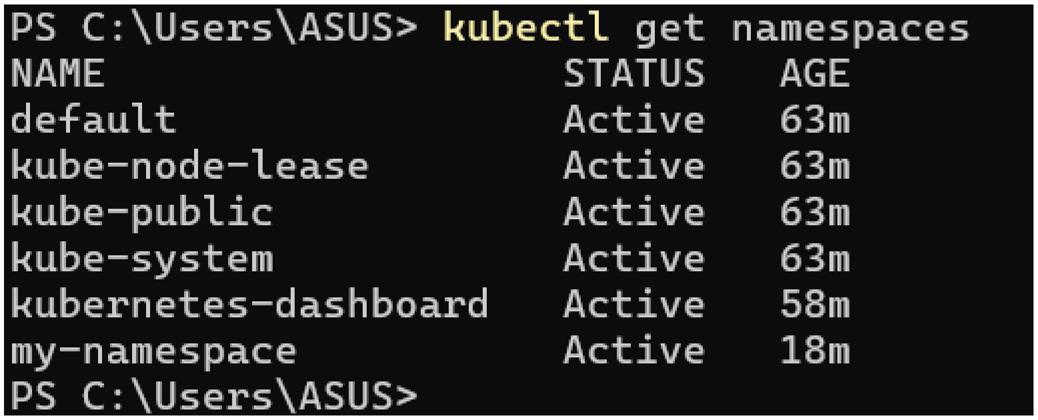
Step 6: Clean Up Resources

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 myns
```



```
PS C:\Users\ASUS> kubectl delete -f nginx-replicaset-quota.yaml
replicaset.apps "nginx-replicaset" deleted from myns namespace
PS C:\Users\ASUS> kubectl delete -f nginx-extra-pod.yaml
Error from server (NotFound): error when deleting "nginx-extra-pod.yaml": po
ds "nginx-extra-pod" not found
PS C:\Users\ASUS> kubectl delete -f nginx-extra-pod.yaml
Error from server (NotFound): error when deleting "nginx-extra-pod.yaml": po
ds "nginx-extra-pod" not found
PS C:\Users\ASUS> kubectl delete namespace myns
namespace "myns" deleted
```



```
PS C:\Users\ASUS> kubectl get namespaces
NAME                      STATUS   AGE
default                   Active   63m
kube-node-lease           Active   63m
kube-public               Active   63m
kube-system               Active   63m
kubernetes-dashboard      Active   58m
my-namespace              Active   18m
PS C:\Users\ASUS>
```