

Lab Exercise 14- Implementing Resource Quota in Kubernetes

Name:- Vansh Bhatt

SapID:- 500125395

R.No:- R2142231689

Batch:- DevOps B1

To:- Hitesh Sharma Sir

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:

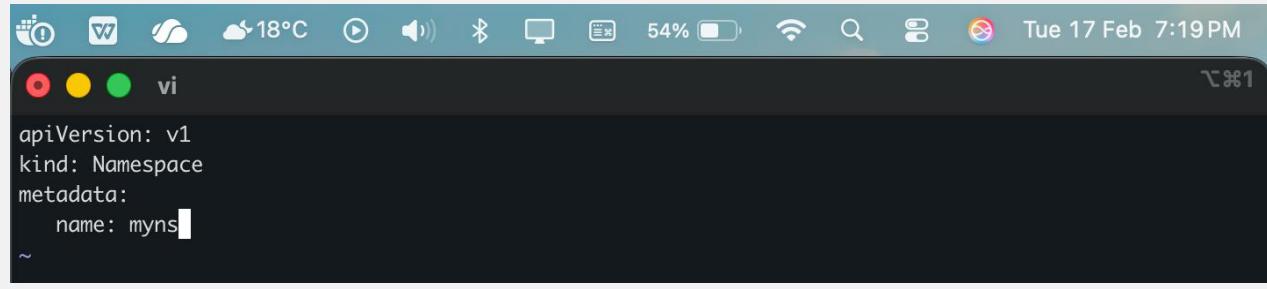
- Limit the amount of CPU and memory a namespace can use.
- Control the number of certain types of resources (e.g., Pods, Services, PersistentVolumeClaims) in a namespace.
- Prevent a namespace from consuming more resources than allocated, ensuring fair usage across multiple teams or applications.

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:

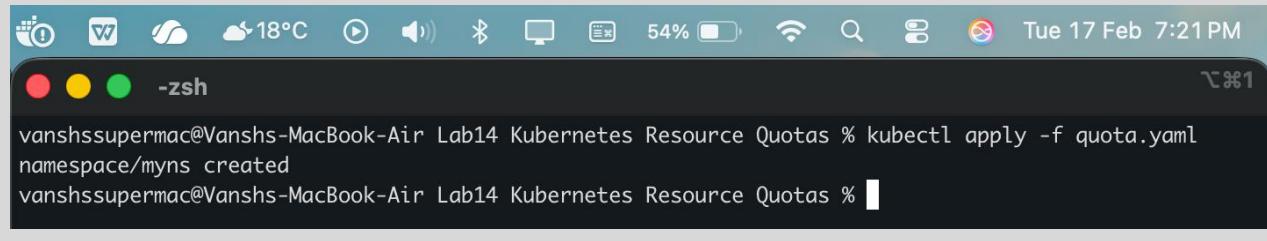
```
apiVersion: v1
kind: Namespace
metadata:
  name: myns
```



A screenshot of a macOS desktop environment. At the top, there's a menu bar with icons for battery, temperature (18°C), volume, and network. The date and time are shown as 'Tue 17 Feb 7:19 PM'. Below the menu bar is a dark-themed terminal window titled 'vi'. The terminal contains the YAML code for creating a namespace, with the 'name: myns' line currently selected. The prompt '~' is visible at the bottom of the terminal.

Apply the YAML to create the namespace:

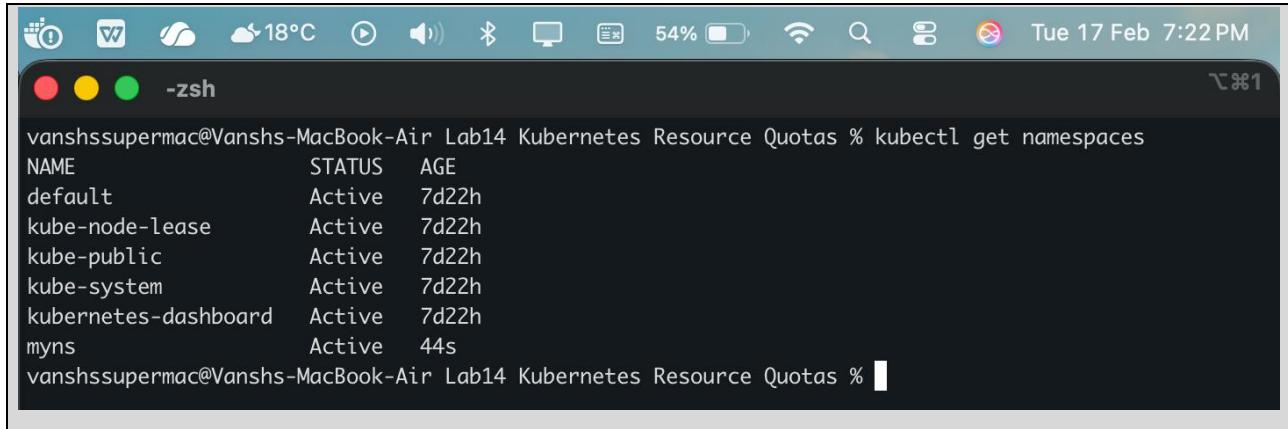
```
kubectl apply -f quota-namespace.yaml
```



A screenshot of a macOS desktop environment. The terminal window is titled '-zsh'. The command 'kubectl apply -f quota-namespace.yaml' is being run. The output shows that a namespace named 'myns' has been created successfully. The prompt '% ' is visible at the bottom of the terminal.

Verify that the namespace is created:

```
kubectl get namespaces
```



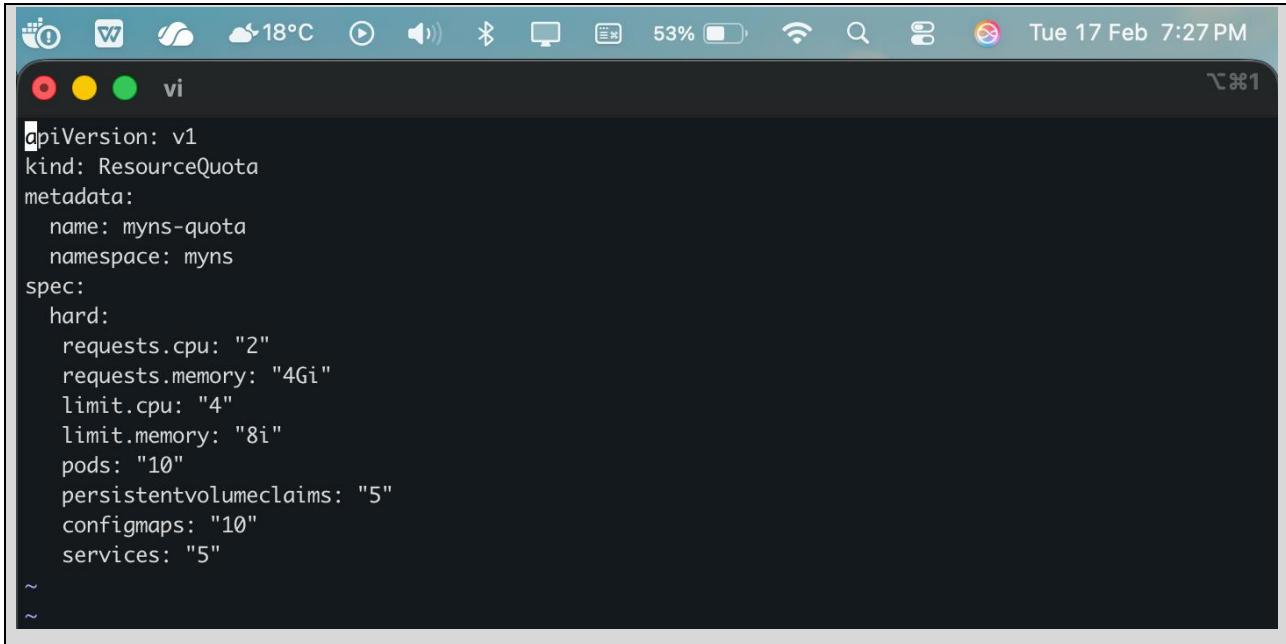
```
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas % kubectl get namespaces
NAME        STATUS  AGE
default     Active  7d22h
kube-node-lease  Active  7d22h
kube-public   Active  7d22h
kube-system   Active  7d22h
kubernetes-dashboard  Active  7d22h
myns         Active  44s
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas %
```

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:
    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.
```

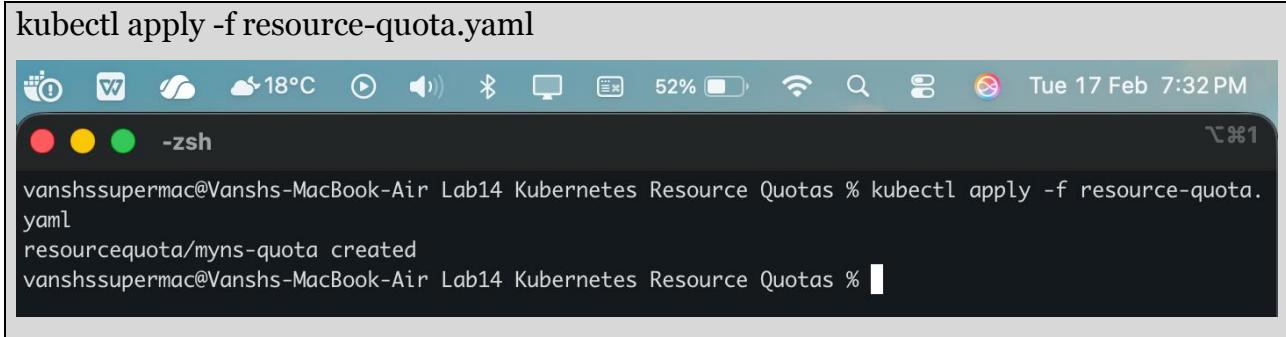


A screenshot of a macOS desktop environment. At the top, there's a menu bar with icons for system status and a date/time indicator ('Tue 17 Feb 7:27 PM'). Below the menu bar is a dark terminal window titled 'vi'. The terminal contains a single line of text: 'vi'. The main body of the terminal shows a ResourceQuota YAML configuration:

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

Step 4: Apply the Resource Quota

Apply the Resource Quota YAML to the namespace:



A screenshot of a macOS desktop environment. At the top, there's a menu bar with icons for system status and a date/time indicator ('Tue 17 Feb 7:32 PM'). Below the menu bar is a dark terminal window titled '-zsh'. The terminal shows the command 'kubectl apply -f resource-quota.yaml' being run. The output indicates that a ResourceQuota named 'myns-quota' was created successfully.

```
kubectl apply -f resource-quota.yaml
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas % kubectl apply -f resource-quota.yaml
resourcequota/myns-quota created
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas %
```

Verify that the Resource Quota is applied:

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

```
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas % kubectl get resourcequota -n myns
NAME      REQUEST           LIMIT          AGE
myns-quota  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  38s
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas %
```

To see the details of the applied Resource Quota:

```
kubectl describe resourcequota myns-quota -n myns
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas % 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
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas %
```

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"
```

```
vi
```

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: vb-nginx
  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:

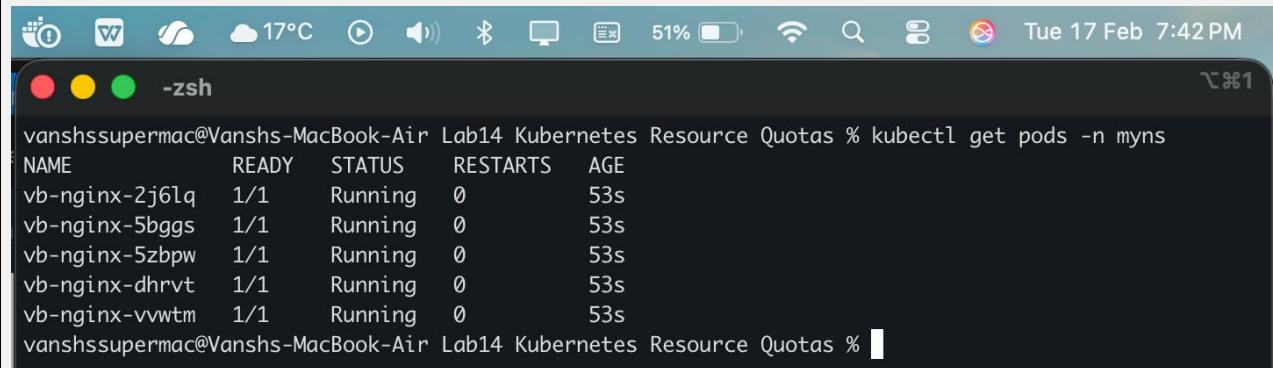
```
kubectl apply -f nginx-replicaset-quota.yaml
```

```
-zsh
```

```
vanshssupermac@Vanshs-MacBook-Air ~ % kubectl apply -f nginx-rq.yaml
replicaset.apps/vb-nginx created
vanshssupermac@Vanshs-MacBook-Air ~ %
```

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

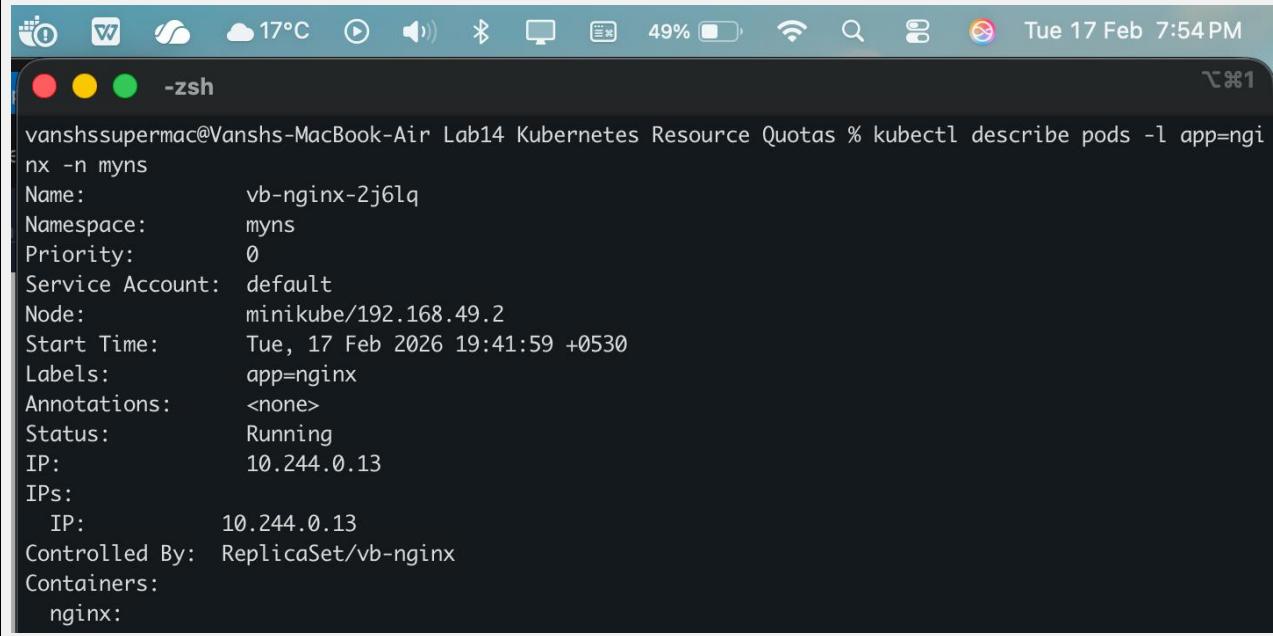
```
kubectl get pods -n myns
```



```
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas % kubectl get pods -n myns
NAME        READY   STATUS    RESTARTS   AGE
vb-nginx-2j6lq  1/1     Running   0          53s
vb-nginx-5bggs  1/1     Running   0          53s
vb-nginx-5zbpw  1/1     Running   0          53s
vb-nginx-dhrvt  1/1     Running   0          53s
vb-nginx-vwwtm  1/1     Running   0          53s
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas %
```

To describe the Pods and see their resource allocations:

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



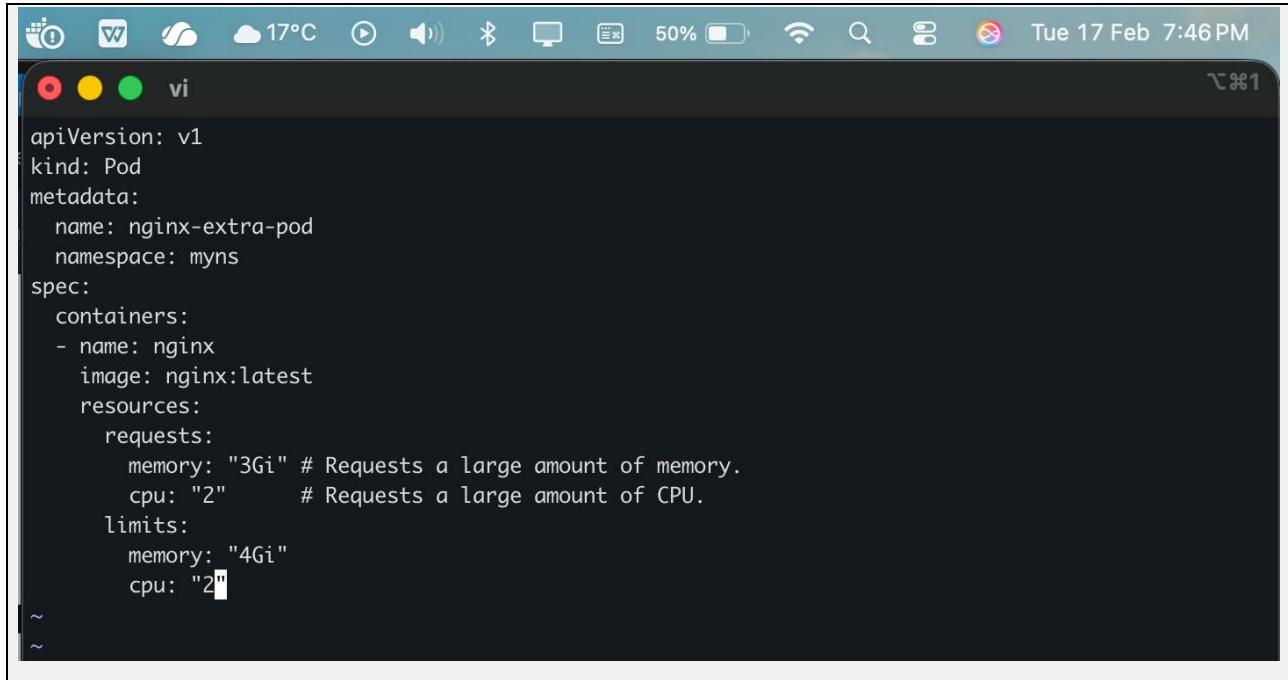
```
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas % kubectl describe pods -l app=nginx -n myns
Name:           vb-nginx-2j6lq
Namespace:      myns
Priority:       0
Service Account: default
Node:           minikube/192.168.49.2
Start Time:     Tue, 17 Feb 2026 19:41:59 +0530
Labels:         app=nginx
Annotations:    <none>
Status:         Running
IP:             10.244.0.13
IPs:
  IP:           10.244.0.13
Controlled By: ReplicaSet/vb-nginx
Containers:
  nginx:
```

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:

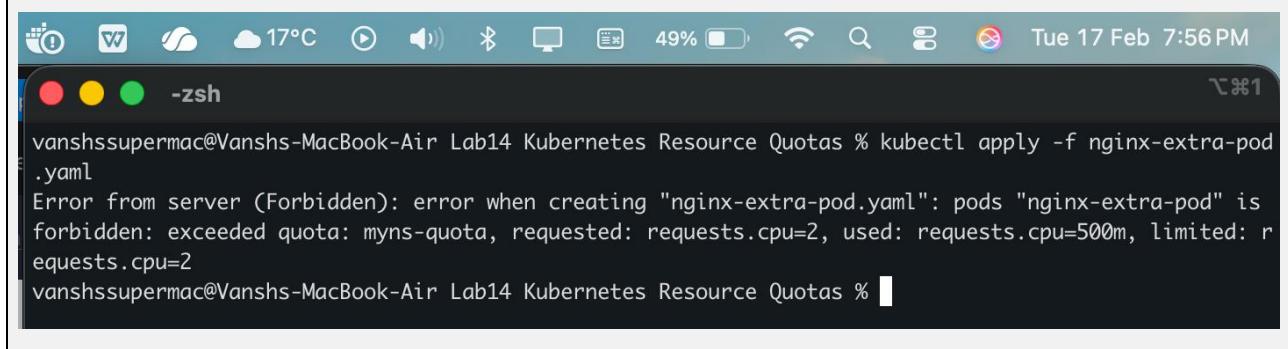
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-extra-pod
  namespace: myns
spec:
  containers:
  - name: nginx
    image: nginx:latest
    resources:
      requests:
        memory: "3Gi" # Requests a large amount of memory.
        cpu: "2"      # Requests a large amount of CPU.
      limits: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



```
vanshssupermac@Vanshs-MacBook-Air ~ % 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: myns-quota, requested: requests.cpu=2, used: requests.cpu=500m, limited: requests.cpu=2
vanshssupermac@Vanshs-MacBook-Air ~ %
```

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

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

```

vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas % kubectl get events -n myns
LAST SEEN   TYPE      REASON          OBJECT                MESSAGE
14m         Normal    Scheduled       pod/vb-nginx-2j6lq  Successfully assigned myns/vb-nginx-2j6lq to minikube
14m         Normal    Pulling         pod/vb-nginx-2j6lq  Pulling image "nginx:latest"
14m         Normal    Pulled          pod/vb-nginx-2j6lq  Successfully pulled image "nginx:latest"
" in 3.013s (3.013s including waiting). Image size: 180544671 bytes.
14m         Normal    Created         pod/vb-nginx-2j6lq  Container created
14m         Normal    Started         pod/vb-nginx-2j6lq  Container started
14m         Normal    Scheduled       pod/vb-nginx-5bggs  Successfully assigned myns/vb-nginx-5bggs to minikube
14m         Normal    Pulling         pod/vb-nginx-5bggs  Pulling image "nginx:latest"
14m         Normal    Pulled          pod/vb-nginx-5bggs  Successfully pulled image "nginx:latest"
" in 2.967s (8.706s including waiting). Image size: 180544671 bytes.
14m         Normal    Created         pod/vb-nginx-5bggs  Container created
14m         Normal    Started         pod/vb-nginx-5bggs  Container started
14m         Normal    Scheduled       pod/vb-nginx-5zbpbw  Successfully assigned myns/vb-nginx-5zbpbw to minikube
14m         Normal    Pulling         pod/vb-nginx-5zbpbw  Pulling image "nginx:latest"
14m         Normal    Pulled          pod/vb-nginx-5zbpbw  Successfully pulled image "nginx:latest"
" in 2.738s (5.751s including waiting). Image size: 180544671 bytes.
14m         Normal    Created         pod/vb-nginx-5zbpbw  Container created
14m         Normal    Started         pod/vb-nginx-5zbpbw  Container started
14m         Normal    Scheduled       pod/vb-nginx-dhrvt  Successfully assigned myns/vb-nginx-dhrvt to minikube
14m         Normal    Pulling         pod/vb-nginx-dhrvt  Pulling image "nginx:latest"
14m         Normal    Pulled          pod/vb-nginx-dhrvt  Successfully pulled image "nginx:latest"
" in 2.552s (11.259s including waiting). Image size: 180544671 bytes.
14m         Normal    Created         pod/vb-nginx-dhrvt  Container created
14m         Normal    Started         pod/vb-nginx-dhrvt  Container started
14m         Normal    Scheduled       pod/vb-nginx-vvwtm  Successfully assigned myns/vb-nginx-vvwtm to minikube
14m         Normal    Pulling         pod/vb-nginx-vvwtm  Pulling image "nginx:latest"
14m         Normal    Pulled          pod/vb-nginx-vvwtm  Successfully pulled image "nginx:latest"
" in 2.874s (14.133s including waiting). Image size: 180544671 bytes.
14m         Normal    Created         pod/vb-nginx-vvwtm  Container created
14m         Normal    Started         pod/vb-nginx-vvwtm  Container started
14m         Normal    SuccessfulCreate replicaset/vb-nginx  Created pod: vb-nginx-2j6lq
14m         Normal    SuccessfulCreate replicaset/vb-nginx  Created pod: vb-nginx-vvwtm
14m         Normal    SuccessfulCreate replicaset/vb-nginx  Created pod: vb-nginx-dhrvt
14m         Normal    SuccessfulCreate replicaset/vb-nginx  Created pod: vb-nginx-5zbpbw
14m         Normal    SuccessfulCreate replicaset/vb-nginx  Created pod: vb-nginx-5bggs
vanshssupermac@Vanshs-MacBook-Air Lab14 Kubernetes Resource Quotas %

```

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
```

Thank You