EXPERIMENT 10

AIM: Implementing Resource Quota in Kubernetes

Objective:

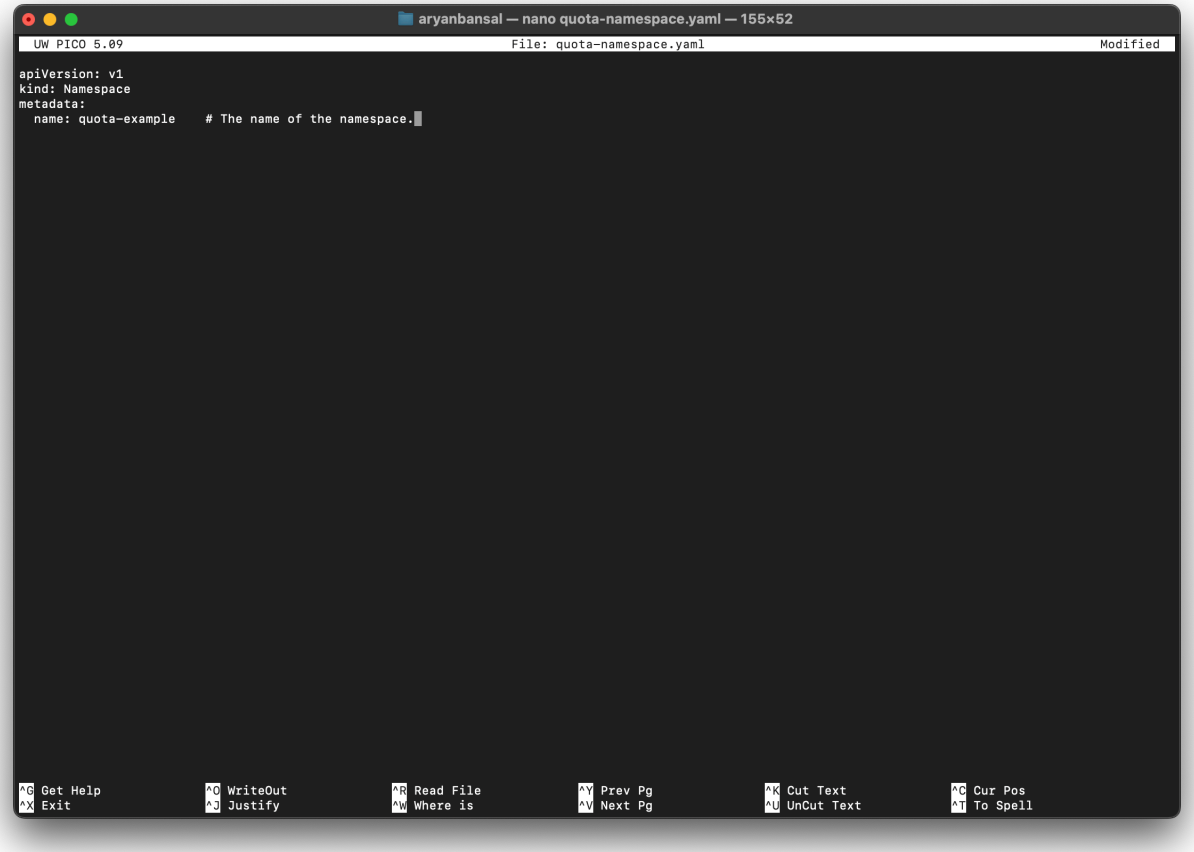
In Kubernetes, Resource Quotas are used to control the resource consumption of namespaces. They help in managing and enforcing limits on the usage of resources like CPU, memory, and the number of objects (e.g., Pods, Services) within a namespace. This exercise will guide you through creating and managing 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:



Apply the YAML to create the namespace:
kubectl apply -f quota-namespace.yaml

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl apply -f quota-namespace.yaml
namespace/quota-example created
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

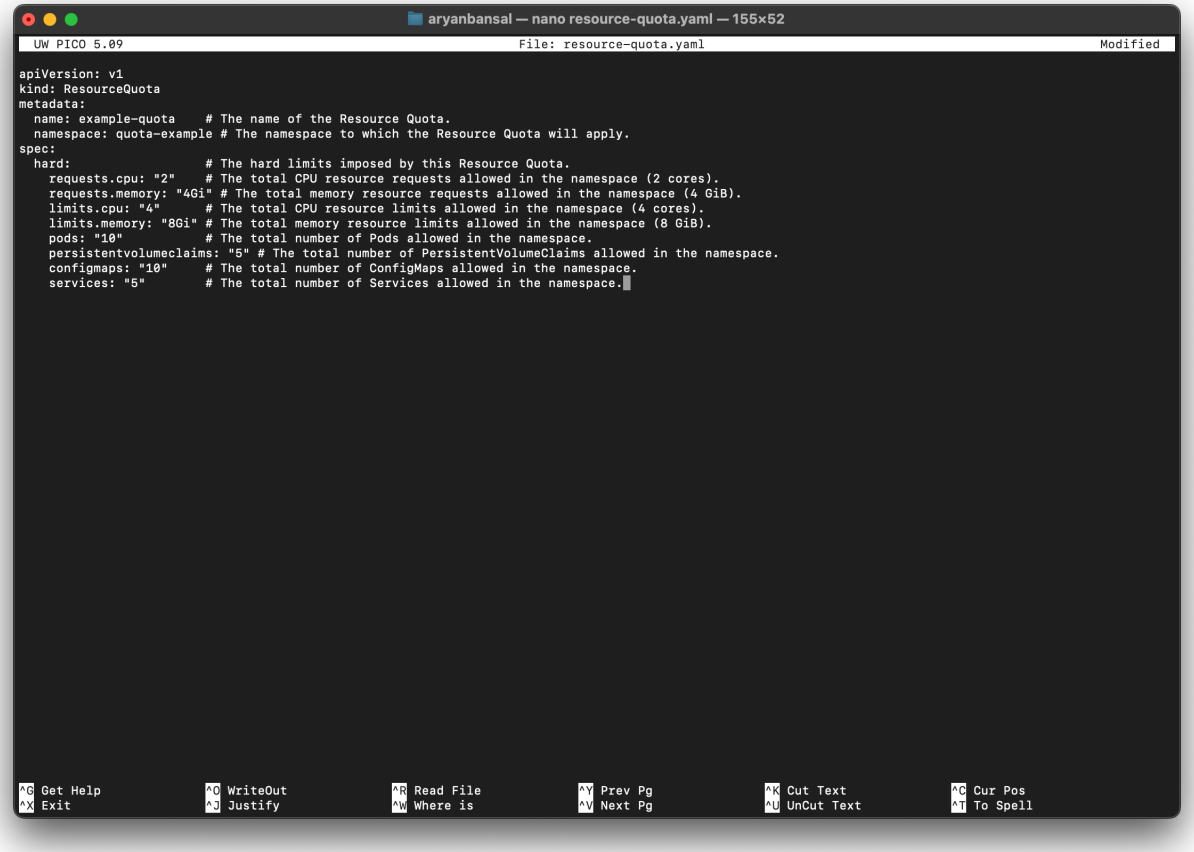
Verify that the namespace is created:
kubectl get namespaces

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get namespaces
NAME              STATUS   AGE
default           Active   11d
kube-node-lease   Active   11d
kube-public       Active   11d
kube-system       Active   11d
quota-example     Active   35s
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

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:



Step 4: Apply the Resource Quota

Apply the Resource Quota YAML to the namespace:
kubectl apply -f resource-quota.yaml

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl apply -f resource-quota.yaml
resourcequota/example-quota created
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

Verify that the Resource Quota is applied:
kubectl get resourcequota -n quota-example

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get resourcequota -n quota-example
NAME            AGE    REQUEST                                                                                                                LIMIT
example-quota   2m6s   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
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % 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
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

To describe the Pods and see their resource allocations:
kubectl describe pods -l app=nginx -n quota-example

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl describe pods -l app=nginx -n quota-example
No resources found in quota-example namespace.
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

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:

```
aryanbansal — nano nginx-replicaset-quota.yaml — 186×52
  UW PICO 5.09                    File: nginx-replicaset-quota.yaml                    Modified

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"

^G Get Help    ^O WriteOut    ^R Read File    ^Y Prev Pg    ^K Cut Text     ^C Cur Pos
^X Exit        ^J Justify     ^W Where is     ^V Next Pg    ^U UnCut Text   ^T To Spell
```

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

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl apply -f nginx-replicaset-quota.yaml
replicaset.apps/nginx-replicaset created
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

Check the status of the Pods and ensure they are created within the constraints of the Resource Quota:
kubectl get pods -n quota-example

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get pods -n quota-example
NAME                     READY   STATUS    RESTARTS   AGE
nginx-replicaset-2fckj   1/1     Running   0          32s
nginx-replicaset-t5zwx   1/1     Running   0          32s
nginx-replicaset-t6ww9   1/1     Running   0          32s
nginx-replicaset-v8dwk   1/1     Running   0          32s
nginx-replicaset-vgrzz   1/1     Running   0          32s
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

To describe the Pods and see their resource allocations:
kubectl describe pods -l app=nginx -n quota-example

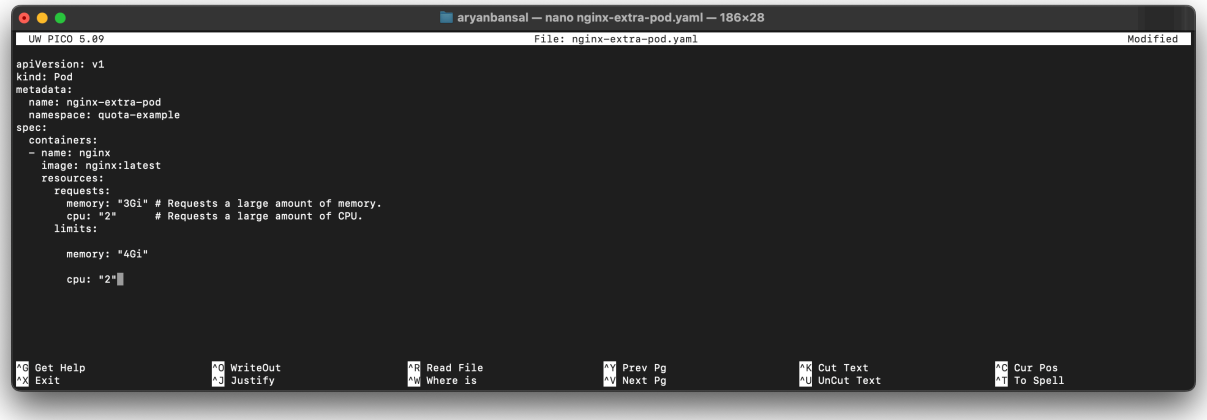
```
aryanbansal — -zsh — 186×58
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl apply -f resource-quota.yaml
resourcequota/example-quota created
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get resourcequota -n quota-example
NAME            AGE   REQUEST                                                                                                                  LIMIT
example-quota   41s   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
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl describe pods -l app=nginx -n quota-example
No resources found in quota-example namespace.
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get resourcequota -n quota-example
NAME            AGE    REQUEST                                                                                                                  LIMIT
example-quota   2m6s   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
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % 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
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl describe pods -l app=nginx -n quota-example
No resources found in quota-example namespace.
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % touch nginx-replicaset-quota.yaml
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % nano nginx-replicaset-quota.yaml
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl apply -f nginx-replicaset-quota.yaml
replicaset.apps/nginx-replicaset created
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get pods -n quota-example
NAME                     READY   STATUS    RESTARTS   AGE
nginx-replicaset-2fckj   1/1     Running   0          32s
nginx-replicaset-t5zwx   1/1     Running   0          32s
nginx-replicaset-t6ww9   1/1     Running   0          32s
nginx-replicaset-v8dwk   1/1     Running   0          32s
nginx-replicaset-vgrzz   1/1     Running   0          32s
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl describe pods -l app=nginx -n quota-example

Name:             nginx-replicaset-2fckj
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Thu, 21 Nov 2024 19:30:44 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.24
IPs:
  IP:           10.244.0.24
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://e5dc13f3fee5cbb83a4784b1f129355277b1f5bfe421c58dad694429f53ac09c
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470
    Port:           80/TCP
    Host Port:      0/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:



Apply this YAML to create the Pod:
kubectl apply -f nginx-extra-pod.yaml

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl apply -f nginx-extra-pod.yaml
pod/nginx-extra-pod created
```

This should fail due to exceeding the Resource Quota. Check the events to see the failure reason:
kubectl get events -n quota-example

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

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl get events -n quota-example
LAST SEEN   TYPE     REASON      OBJECT              MESSAGE
2m15s       Normal   Scheduled   pod/nginx-extra-pod   Successfully assigned quota-example/nginx-extra-pod to minikube
2m14s       Normal   Pulling     pod/nginx-extra-pod   Pulling image "nginx:latest"
2m8s        Normal   Pulled      pod/nginx-extra-pod   Successfully pulled image "nginx:latest" in 6.021s (6.021s including waiting). Image size: 196880043 bytes.
2m8s        Normal   Created     pod/nginx-extra-pod   Created container nginx
2m8s        Normal   Started     pod/nginx-extra-pod   Started container nginx
2m5s        Normal   Killing     pod/nginx-extra-pod   Stopping container nginx
44s         Normal   Scheduled   pod/nginx-extra-pod   Successfully assigned quota-example/nginx-extra-pod to minikube
43s         Normal   Pulling     pod/nginx-extra-pod   Pulling image "nginx:latest"
38s         Normal   Pulled      pod/nginx-extra-pod   Successfully pulled image "nginx:latest" in 4.846s (4.846s including waiting). Image size: 196880043 bytes.
38s         Normal   Created     pod/nginx-extra-pod   Created container nginx
38s         Normal   Started     pod/nginx-extra-pod   Started container nginx
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```

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

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl delete -f nginx-replicaset-quota.yaml
replicaset.apps "nginx-replicaset" deleted
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl delete -f nginx-extra-pod.yaml
Error from server (NotFound): error when deleting "nginx-extra-pod.yaml": pods "nginx-extra-pod" not found
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl delete -f resource-quota.yaml
kubectl delete namespace quota-example
resourcequota "example-quota" deleted
namespace "quota-example" deleted
```

```
(base) aryanbansal@Aryans-MacBook-Air-10 ~ % kubectl delete -f nginx-extra-pod.yaml
pod "nginx-extra-pod" deleted
(base) aryanbansal@Aryans-MacBook-Air-10 ~ %
```