Name- Aayush Mishra
SAP ID: 500107141

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.

```
PS C:\Users\Slayer\nginx-html-app> code .
```

quota-namespace.yaml

```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

```
PS C:\Users\Slayer\nginx-html-app> kubectl apply -f quota-namespace.yaml
namespace/quota-example created
```

Verify that the namespace is created:

kubectl get namespaces

```
PS C:\Users\Slayer\nginx-html-app> kubectl get namespaces
NAME              STATUS   AGE
default           Active   15d
kube-node-lease   Active   15d
kube-public       Active   15d
kube-system       Active   15d
quota-example     Active   21s
```

You should see the 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

```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
```

```
PS C:\Users\Slayer\nginx-html-app> kubectl apply -f resource-quota.yaml
resourcequota/example-quota created
```

Verify that the Resource Quota is applied:

kubectl get resourcequota -n quota-example

```
PS C:\Users\Slayer\nginx-html-app> kubectl get resourcequota -n quota-exam
ple
NAME            AGE   REQUEST                                                 LIMIT
example-quota   21s   configmaps: 1/10, persistentvolumeclaims: 0/5, pods:
 0/10, requests.cpu: 0/2, requests.memory: 0/4Gi, services: 0/5   limits.c
pu: 0/4, limits.memory: 0/8Gi
PS C:\Users\Slayer\nginx-html-app>
```

To see the details of the applied Resource Quota:

kubectl describe resourcequota example-quota -n quota-example

```
PS C:\Users\Slayer\nginx-html-app> 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
PS C:\Users\Slayer\nginx-html-app>
```

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
```

```
PS C:\Users\Slayer\nginx-html-app> 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
```

```
PS C:\Users\Slayer\nginx-html-app> kubectl get pods -n quota-example
NAME                           READY   STATUS    RESTARTS   AGE
nginx-replicaset-49s6s         1/1     Running   0          29s
nginx-replicaset-jvj78         1/1     Running   0          29s
nginx-replicaset-krqft         1/1     Running   0          29s
nginx-replicaset-w8nxj         1/1     Running   0          29s
nginx-replicaset-zmpr8         1/1     Running   0          29s
PS C:\Users\Slayer\nginx-html-app>
```

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

```
C:\Users\Slayer\nginx-html-app>kubectl describe pods -l app=nginx -n quota-example
Name:             nginx-replicaset-49s6s
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Sun, 24 Nov 2024 20:30:42 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.18
IPs:
  IP:           10.244.0.18
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://1157397a4eac18bb41808d10f0536b334e93fc6d1ec8cd31858a25b88219dd91
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 24 Nov 2024 20:30:55 +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-8p7hp (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-8p7hp:
    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-49s6s to minikube
  Normal  Pulling    63s   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     51s   kubelet            Successfully pulled image "nginx:latest" in 2.861s (11.607s including waiting). Image size: 191670156 bytes.
  Normal  Created    51s   kubelet            Created container nginx
  Normal  Started    51s   kubelet            Started container nginx
```

```
Name:             nginx-replicaset-zmpr8
Namespace:        quota-example
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Sun, 24 Nov 2024 20:30:42 +0530
Labels:           app=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.15
IPs:
  IP:           10.244.0.15
Controlled By:  ReplicaSet/nginx-replicaset
Containers:
  nginx:
    Container ID:   docker://f2b1d22df9d3fd228d43e623a0773527c55600f98cb8a3bc82690e12b590fc79
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 24 Nov 2024 20:30:49 +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-5gxxk (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-5gxxk:
    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  6m47s  default-scheduler  Successfully assigned quota-example/nginx-replicaset-zmpr8 to minikube
  Normal  Pulling    6m46s  kubelet            Pulling image "nginx:latest"
  Normal  Pulled     6m40s  kubelet            Successfully pulled image "nginx:latest" in 2.861s (5.786s including waiting). Image size: 191670156 bytes.
  Normal  Created    6m40s  kubelet            Created container nginx
  Normal  Started    6m40s  kubelet            Started container nginx

C:\Users\Slayer\nginx-html-app>
```

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: 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

```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

```
PS C:\Users\Slayer\nginx-html-app> kubectl apply -f nginx-extra-pod.yaml
pod/nginx-extra-pod unchanged
```

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

kubectl get events -n quota-example

```
C:\Users\Slayer\nginx-html-app>kubectl get events -n quota-example
LAST SEEN   TYPE     REASON             OBJECT                          MESSAGE
23m         Normal   Scheduled          pod/nginx-replicaset-49s6s      Successfully assigned quota-example/nginx-replicaset-49s6s to minikube
23m         Normal   Pulling            pod/nginx-replicaset-49s6s      Pulling image "nginx:latest"
23m         Normal   Pulled             pod/nginx-replicaset-49s6s      Successfully pulled image "nginx:latest" in 2.861s (11.607s including waiting). Image size: 191670156 bytes.
23m         Normal   Created            pod/nginx-replicaset-49s6s      Created container nginx
23m         Normal   Started            pod/nginx-replicaset-49s6s      Started container nginx
23m         Normal   Scheduled          pod/nginx-replicaset-jvj78      Successfully assigned quota-example/nginx-replicaset-jvj78 to minikube
23m         Normal   Pulling            pod/nginx-replicaset-jvj78      Pulling image "nginx:latest"
23m         Normal   Pulled             pod/nginx-replicaset-jvj78      Successfully pulled image "nginx:latest" in 2.87s (14.476s including waiting). Image size: 191670156 bytes.
23m         Normal   Created            pod/nginx-replicaset-jvj78      Created container nginx
23m         Normal   Started            pod/nginx-replicaset-jvj78      Started container nginx
23m         Normal   Scheduled          pod/nginx-replicaset-krqft      Successfully assigned quota-example/nginx-replicaset-krqft to minikube
23m         Normal   Pulling            pod/nginx-replicaset-krqft      Pulling image "nginx:latest"
23m         Normal   Pulled             pod/nginx-replicaset-krqft      Successfully pulled image "nginx:latest" in 3.075s (8.862s including waiting). Image size: 191670156 bytes.
23m         Normal   Created            pod/nginx-replicaset-krqft      Created container nginx
23m         Normal   Started            pod/nginx-replicaset-krqft      Started container nginx
23m         Normal   Scheduled          pod/nginx-replicaset-w8nxj      Successfully assigned quota-example/nginx-replicaset-w8nxj to minikube
23m         Normal   Pulling            pod/nginx-replicaset-w8nxj      Pulling image "nginx:latest"
23m         Normal   Pulled             pod/nginx-replicaset-w8nxj      Successfully pulled image "nginx:latest" in 2.925s (2.925s including waiting). Image size: 191670156 bytes.
23m         Normal   Created            pod/nginx-replicaset-w8nxj      Created container nginx
23m         Normal   Started            pod/nginx-replicaset-w8nxj      Started container nginx
23m         Normal   Scheduled          pod/nginx-replicaset-zmpr8      Successfully assigned quota-example/nginx-replicaset-zmpr8 to minikube
23m         Normal   Pulling            pod/nginx-replicaset-zmpr8      Pulling image "nginx:latest"
23m         Normal   Pulled             pod/nginx-replicaset-zmpr8      Successfully pulled image "nginx:latest" in 2.861s (5.786s including waiting). Image size: 191670156 bytes.
23m         Normal   Created            pod/nginx-replicaset-zmpr8      Created container nginx
23m         Normal   Started            pod/nginx-replicaset-zmpr8      Started container nginx
23m         Normal   SuccessfulCreate   replicaset/nginx-replicaset     Created pod: nginx-replicaset-zmpr8
23m         Normal   SuccessfulCreate   replicaset/nginx-replicaset     Created pod: nginx-replicaset-w8nxj
23m         Normal   SuccessfulCreate   replicaset/nginx-replicaset     Created pod: nginx-replicaset-krqft
23m         Normal   SuccessfulCreate   replicaset/nginx-replicaset     Created pod: nginx-replicaset-jvj78
23m         Normal   SuccessfulCreate   replicaset/nginx-replicaset     Created pod: nginx-replicaset-49s6s
```

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

```
PS C:\Users\Slayer\nginx-html-app> kubectl delete -f nginx-replicaset-quota.yaml
replicaset.apps "nginx-replicaset" deleted
PS C:\Users\Slayer\nginx-html-app> kubectl delete -f nginx-extra-pod.yaml
pod "nginx-extra-pod" deleted
PS C:\Users\Slayer\nginx-html-app> kubectl delete -f resource-quota.yaml
resourcequota "example-quota" deleted
PS C:\Users\Slayer\nginx-html-app> kubectl delete namespace quota-example
namespace "quota-example" deleted
PS C:\Users\Slayer\nginx-html-app>
```