RESOURCE QUOTAS

<u>About Resource</u>

<u>Requests and Limits:-</u>

Resource limits can be set for cpu and memory, default pod will utilize entire resource from the node, incase pod is getting more hits the it may hang other pods in the node, so to avoid such situation we can limits the resource for pod. normal virtual machine will support minimum core 1 and max upto 64, it depends on hardware but in pod we can limits the core in ratio of 1:10 or half core etc...

Request -->minimum we can call it as soft limits

limits --> maximum we can call it as hard limits

if cpu is full then pod will be stale or hang, but if memory is full the it will recycle. In production this will be used to avoid interruption of another pod service. Quotas work on namespaces only not in node level.

<u>Actual readme</u>

<u>Resource Quotas</u>

When several users or teams share a cluster with a fixed number of nodes, there is a concern that one team could use more than its fair share of resources.Resource quotas are a tool for administrators to address this concern.

A resource quota, defined by a ResourceQuota object, provides constraints that limit aggregate resource consumption per namespace. It can limit the quantity of objects that can be created in a namespace by type, as well as the total amount of compute resources that may be consumed by resources in that project.

<u>Resource quotas work like this:</u>

1) Different teams work in different namespaces. Currently this is voluntary, but support for making this mandatory via ACLs is planned.

2) The administrator creates one ResourceQuota for each namespace.

3) Users create resources (pods, services, etc.) in the namespace, and the quota system tracks usage to ensure it does not exceed hard resource limits defined in a ResourceQuota.

4) If creating or updating a resource violates a quota constraint, the request will fail with HTTP status code 403 FORBIDDEN with a message explaining the constraint that would have been violated.

5) If quota is enabled in a namespace for compute resources like cpu and memory, users must specify requests or limits for those values; otherwise, the quota system may reject pod creation. Hint: Use the LimitRanger admission controller to force defaults for pods that make no compute resource requirements. Neither contention nor changes to quota will affect already created resources.

<u>Compute Resource Quota</u>

You can limit the total sum of compute resources that can be requested in a given namespace.

The following resource types are supported: CPU and MEMORY

<u>Storage Resource Quota</u>

You can limit the total sum of storage resources that can be requested in a given namespace.

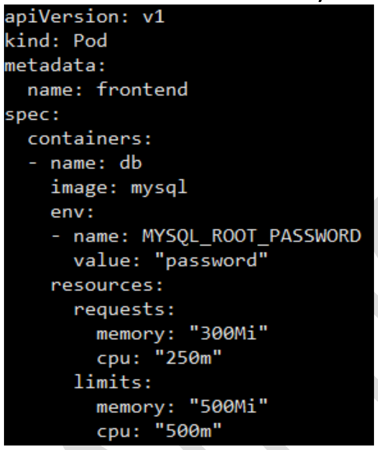
<u>Object Count Quota</u>

Here is an example set of resources users may want to put under object count quota:

count/persistentvolumeclaims, count/services, count/secrets, count/configmaps, count/replicationcontrollers, count/deployments.apps, count/replicasets.apps, count/statefulsets.apps count/jobs.batch, count/cronjobs.batch, count/deployments.extensions

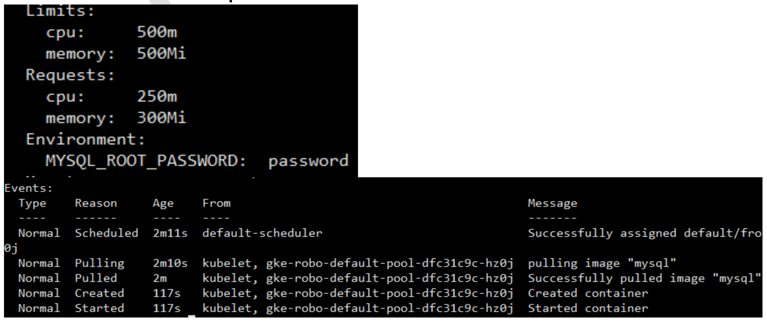
Requests and Limits
Requests and limits are the mechanisms Kubernetes uses to control resources such as CPU and memory. Requests are what the container is guaranteed to get. If a container requests a resource, Kubernetes will only schedule it on a node that can give it that resource. Limits, on the other hand, make sure a container never goes above a certain value. The container is only allowed to go up to the limit, and then it is restricted. It is important to remember that the limit can never be lower than the request. If you try this, Kubernetes will throw an error and won't let you run the container.Requests and limits are on a per-container basis. While Pods usually contain a single container, it's common to see Pods with multiple containers as well. Each container in the Pod gets its own individual limit and request, but because Pods are always scheduled as a group, you need to add the limits and requests for each container together to get an aggregate value for the Pod. To control what requests and limits a container can have, you can set quotas at the Container level and at the Namespace level.
GKE's per-project limits are:
Maximum of 50 clusters per zone, plus 50 regional clusters per region.
GKE's per-cluster limits are:
Maximum of 5000 nodes per cluster.
Maximum of 1000 nodes per cluster if you use the GKE ingress controller.
100 Pods per node.
300,000 containers.
The rate limit for the GKE API is 10 requests per second.

##Create a resource limits on pod level##
# cat container-resources.yml



```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: db
    image: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: "password"
    resources:
      requests:
        memory: "300Mi"
        cpu: "250m"
      limits:
        memory: "500Mi"
        cpu: "500m"
```

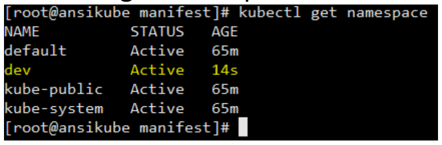
# kubectl apply -f container-resources.yml
# kubectl get pods -o wide
# kubectl describe pod frontend



```
  Limits:
    cpu:      500m
    memory:   500Mi
  Requests:
    cpu:      250m
    memory:   300Mi
  Environment:
    MYSQL_ROOT_PASSWORD:  password
Events:
  Type    Reason     Age    From                                        Message
  ----    ------     ----   ----                                        -------
  Normal  Scheduled  2m11s  default-scheduler                           Successfully assigned default/fro
0j
  Normal  Pulling    2m10s  kubelet, gke-robo-default-pool-dfc31c9c-hz0j  pulling image "mysql"
  Normal  Pulled     2m     kubelet, gke-robo-default-pool-dfc31c9c-hz0j  Successfully pulled image "mysql"
  Normal  Created    117s   kubelet, gke-robo-default-pool-dfc31c9c-hz0j  Created container
  Normal  Started    117s   kubelet, gke-robo-default-pool-dfc31c9c-hz0j  Started container
```

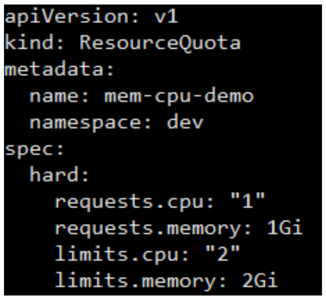
##Create a ResourceQuota on namespace level##

# kubectl create ns dev
kubectl get namespace



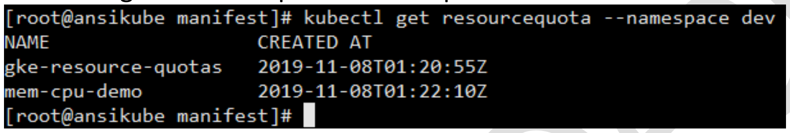
```
[root@ansikube manifest]# kubectl get namespace
NAME          STATUS   AGE
default       Active   65m
dev           Active   14s
kube-public   Active   65m
kube-system   Active   65m
[root@ansikube manifest]#
```

# cat container-resources-2.yml



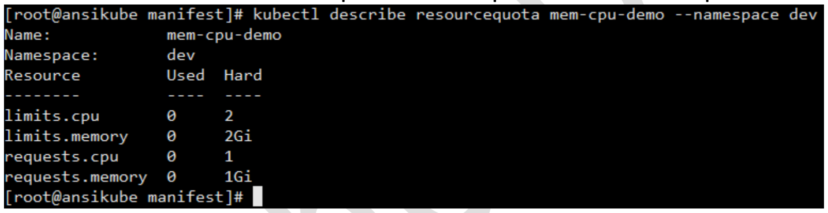
```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mem-cpu-demo
  namespace: dev
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

# kubectl apply -f container-resources-2.yml
kubectl get resourcequota --namespace dev



```
[root@ansikube manifest]# kubectl get resourcequota --namespace dev
NAME                    CREATED AT
gke-resource-quotas     2019-11-08T01:20:55Z
mem-cpu-demo            2019-11-08T01:22:10Z
[root@ansikube manifest]#
```

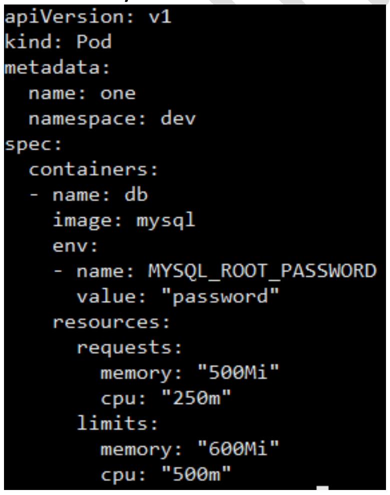
kubectl describe resourcequota mem-cpu-demo --namespace dev



```
[root@ansikube manifest]# kubectl describe resourcequota mem-cpu-demo --namespace dev
Name:            mem-cpu-demo
Namespace:       dev
Resource         Used  Hard
--------         ----  ----
limits.cpu       0     2
limits.memory    0     2Gi
requests.cpu     0     1
requests.memory  0     1Gi
[root@ansikube manifest]#
```

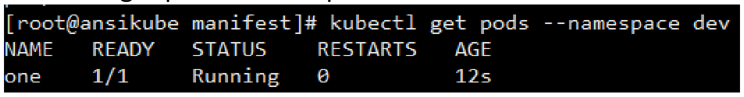
@@Creating pods under dev namespace@@

# cat one.yml



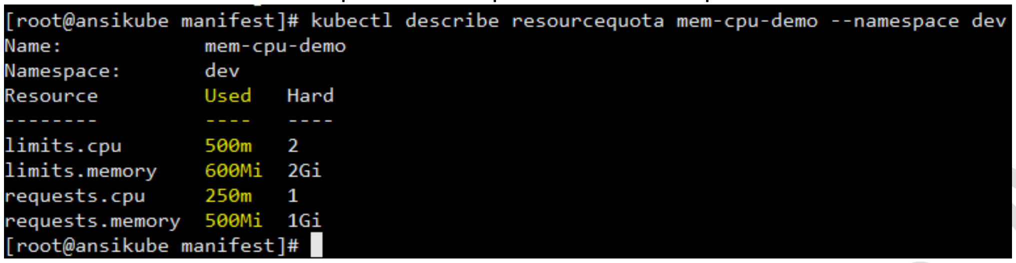
```
apiVersion: v1
kind: Pod
metadata:
  name: one
  namespace: dev
spec:
  containers:
  - name: db
    image: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: "password"
    resources:
      requests:
        memory: "500Mi"
        cpu: "250m"
      limits:
        memory: "600Mi"
        cpu: "500m"
```

# kubectl apply -f one.yml
kubectl get pods --namespace dev



```
[root@ansikube manifest]# kubectl get pods --namespace dev
NAME    READY    STATUS     RESTARTS    AGE
one     1/1      Running    0           12s
```

kubectl describe resourcequota mem-cpu-demo --namespace dev

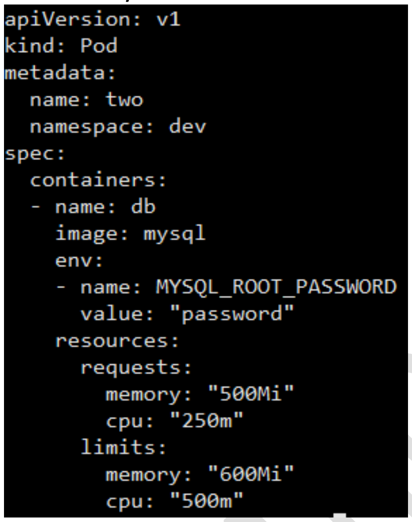


```
[root@ansikube manifest]# kubectl describe resourcequota mem-cpu-demo --namespace dev
Name:              mem-cpu-demo
Namespace:         dev
Resource           Used   Hard
--------           ----   ----
limits.cpu         500m   2
limits.memory      600Mi  2Gi
requests.cpu       250m   1
requests.memory    500Mi  1Gi
[root@ansikube manifest]#
```

Note: - After creating one pod, we can able to see the resource usage.

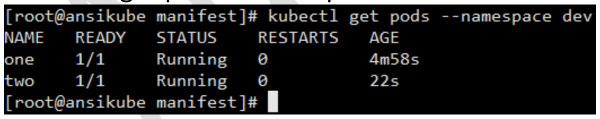
@@Create 2nd pod on dev namespace@@

# cat two.yml



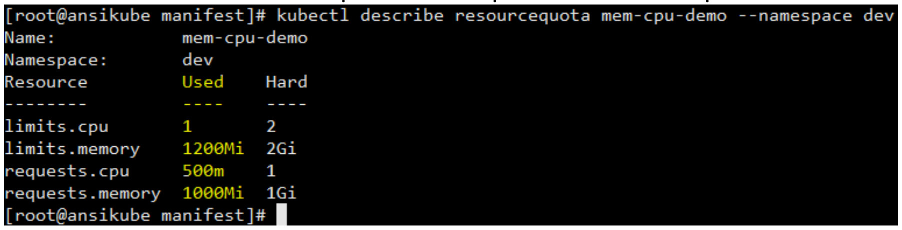
```
apiVersion: v1
kind: Pod
metadata:
  name: two
  namespace: dev
spec:
  containers:
  - name: db
    image: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: "password"
    resources:
      requests:
        memory: "500Mi"
        cpu: "250m"
      limits:
        memory: "600Mi"
        cpu: "500m"
```

# kubectl apply -f two.yml
kubectl get pods --namespace dev



```
[root@ansikube manifest]# kubectl get pods --namespace dev
NAME    READY    STATUS     RESTARTS    AGE
one     1/1      Running    0           4m58s
two     1/1      Running    0           22s
[root@ansikube manifest]#
```

kubectl describe resourcequota mem-cpu-demo --namespace dev

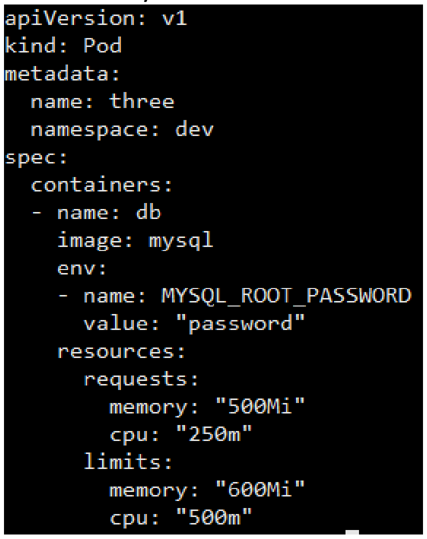


```
[root@ansikube manifest]# kubectl describe resourcequota mem-cpu-demo --namespace dev
Name:              mem-cpu-demo
Namespace:         dev
Resource           Used    Hard
--------           ----    ----
limits.cpu         1       2
limits.memory      1200Mi  2Gi
requests.cpu       500m    1
requests.memory    1000Mi  1Gi
[root@ansikube manifest]#
```

Note: - Resource usage has been increased.

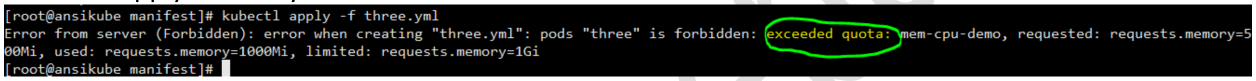
@@Let's try to create 3rd pod@@

# cat three.yml



```
apiVersion: v1
kind: Pod
metadata:
  name: three
  namespace: dev
spec:
  containers:
  - name: db
    image: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: "password"
    resources:
      requests:
        memory: "500Mi"
        cpu: "250m"
      limits:
        memory: "600Mi"
        cpu: "500m"
```

kubectl apply -f three.yml

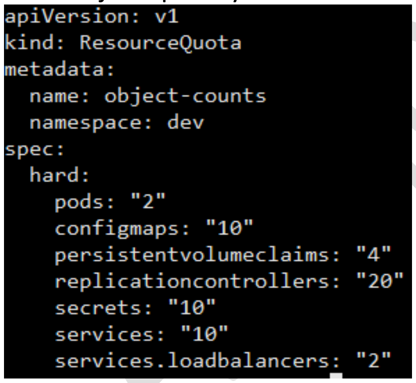


```
[root@ansikube manifest]# kubectl apply -f three.yml
Error from server (Forbidden): error when creating "three.yml": pods "three" is forbidden: exceeded quota: mem-cpu-demo, requested: requests.memory=5
00Mi, used: requests.memory=1000Mi, limited: requests.memory=1Gi
[root@ansikube manifest]#
```

Note: - while trying to create 3rd pod on dev namespace, but its failed because there is no enough resources available. So, the conclusion is we can limit the resource on namespace boundary level.
Note1: - Delete created pods and resourcequota.

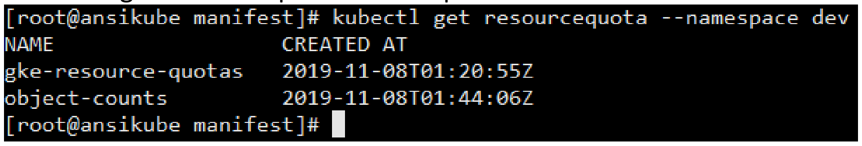
@@Create a resourcequota for object level@@

# cat object-quota.yml



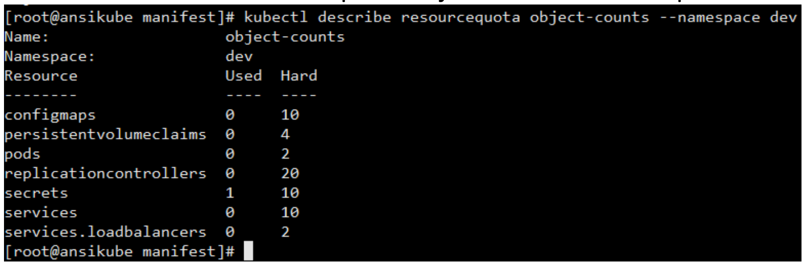
```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: object-counts
  namespace: dev
spec:
  hard:
    pods: "2"
    configmaps: "10"
    persistentvolumeclaims: "4"
    replicationcontrollers: "20"
    secrets: "10"
    services: "10"
    services.loadbalancers: "2"
```

# kubectl apply -f object-quota.yml
kubectl get resourcequota --namespace dev



```
[root@ansikube manifest]# kubectl get resourcequota --namespace dev
NAME                   CREATED AT
gke-resource-quotas    2019-11-08T01:20:55Z
object-counts          2019-11-08T01:44:06Z
[root@ansikube manifest]#
```

kubectl describe resourcequota object-counts --namespace dev



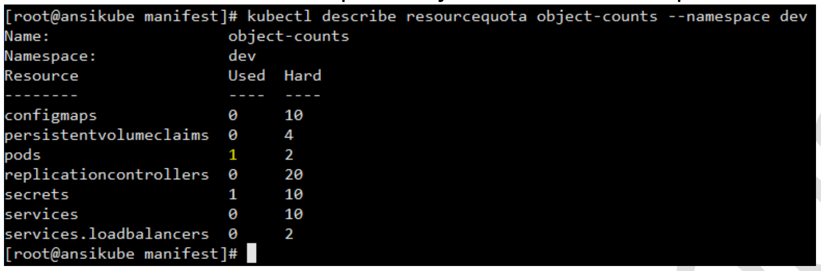
```
[root@ansikube manifest]# kubectl describe resourcequota object-counts --namespace dev
Name:                   object-counts
Namespace:              dev
Resource                Used  Hard
--------                ----  ----
configmaps              0     10
persistentvolumeclaims  0     4
pods                    0     2
replicationcontrollers  0     20
secrets                 1     10
services                0     10
services.loadbalancers  0     2
[root@ansikube manifest]#
```

@@Create a pod and check the resource quota usage@@

# kubectl apply -f one.yml

# kubectl get pods --namespace dev

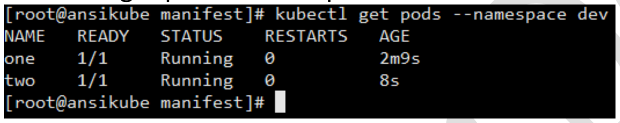
kubectl describe resourcequota object-counts --namespace dev



```
[root@ansikube manifest]# kubectl describe resourcequota object-counts --namespace dev
Name:                   object-counts
Namespace:              dev
Resource                Used  Hard
--------                ----  ----
configmaps              0     10
persistentvolumeclaims  0     4
pods                    1     2
replicationcontrollers  0     20
secrets                 1     10
services                0     10
services.loadbalancers  0     2
[root@ansikube manifest]#
```

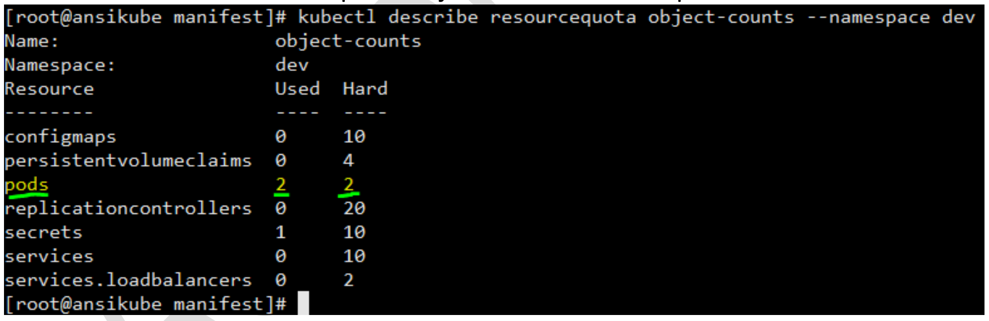
# kubectl apply -f two.yml

# kubectl get pods --namespace dev



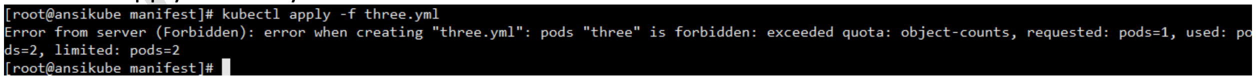
```
[root@ansikube manifest]# kubectl get pods --namespace dev
NAME    READY   STATUS    RESTARTS   AGE
one     1/1     Running   0          2m9s
two     1/1     Running   0          8s
[root@ansikube manifest]#
```

kubectl describe resourcequota object-counts --namespace dev



```
[root@ansikube manifest]# kubectl describe resourcequota object-counts --namespace dev
Name:                   object-counts
Namespace:              dev
Resource                Used  Hard
--------                ----  ----
configmaps              0     10
persistentvolumeclaims  0     4
pods                    2     2
replicationcontrollers  0     20
secrets                 1     10
services                0     10
services.loadbalancers  0     2
[root@ansikube manifest]#
```

kubectl apply -f three.yml



```
[root@ansikube manifest]# kubectl apply -f three.yml
Error from server (Forbidden): error when creating "three.yml": pods "three" is forbidden: exceeded quota: object-counts, requested: pods=1, used: po
ds=2, limited: pods=2
[root@ansikube manifest]#
```

Note:- Above error clearly shows that we cannot create more than two pods, becuase resource quota has been set for pod =2 on object level.