K8s 05

1) Create a namespace dev-environment and apply a resource-based quota that restricts the number of pods to 3 and services to 2.

→ Create the dev-environment Namespace by using below command:

kubectl create namespace dev-environment

```
ubuntu@master:~$ kubectl create namespace dev-environment
namespace/dev-environment created
```

--check ns..its created

```
ubuntu@master:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   3d23h
dev-environment   Active   14s
kube-node-lease   Active   3d23h
kube-public       Active   3d23h
kube-system       Active   3d23h
tigera-operator   Active   3d22h
```

→YAML File to create aresource-based quota that restricts the number of pods to 3 and services to 2:

ubuntu@master: ~

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: dev-quota
  namespace: dev-environment
spec:
  hard:
    pods: "3"
    services: "2"
```

--run Yaml file

```
ubuntu@master:~$ vi resource-quota.yaml
ubuntu@master:~$ kubectl apply -f resource-quota.yaml
resourcequota/dev-quota created
```

--check

```
ubuntu@master:~$ kubectl get resourcequota -n dev-environment
NAME        AGE   REQUEST                       LIMIT
dev-quota   20s   pods: 0/3, services: 0/2
```

→described a resource-based quota that restricts the number of pods to 3 and services to 2:

kubectl describe ns dev-environment

```
ubuntu@master:~$ kubectl describe ns dev-environment
Name:         dev-environment
Labels:       kubernetes.io/metadata.name=dev-environment
Annotations:  <none>
Status:       Active

Resource Quotas
  Name:      dev-quota
  Resource   Used  Hard
  --------   ---   ---
  pods       0     3
  services   0     2

No LimitRange resource.
ubuntu@master:~$
```

→now checked that when we try to create more than 3 pods got error so its restricted by resource-based quota:

```
ubuntu@master:~$ kubectl run mypod1 --image=nginx -n dev-environment
kubectl run mypod2 --image=nginx -n dev-environment
kubectl run mypod3 --image=nginx -n dev-environment
kubectl run mypod4 --image=nginx -n dev-environment
pod/mypod1 created
pod/mypod2 created
pod/mypod3 created
Error from server (Forbidden): pods "mypod4" is forbidden: exceeded quota: dev-quota, requested: pods=1, used: pods=3, limited: pods=3
```

2) Create a pod in the prod-environment namespace with 0.2 CPU and 200Mi memory requests, and 0.5 CPU and 500Mi memory limits.

→first create namespace by below command:

kubectl create ns prod-environment

```
root@master:~# kubectl create ns prod-environment
namespace/prod-environment created
```

--created:

```
root@master:~# kubectl get ns
NAME              STATUS        AGE
default           Active        3d23h
dev-environment   Terminating   24m
kube-node-lease   Active        3d23h
kube-public       Active        3d23h
kube-system       Active        3d23h
prod-environment  Active        6s
tigera-operator   Active        3d22h
```

→ **yaml file to Create a pod in the prod-environment namespace with 0.2 CPU and 200Mi memory requests, and 0.5 CPU and 500Mi memory limits:**

root@master: ~

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
  namespace: prod-environment
spec:
  containers:
  - name: my-container
    image: nginx
    resources:
      requests:
        cpu: "0.2"
        memory: "200Mi"
      limits:
        cpu: "0.5"
        memory: "500Mi"
```

--run yaml

```
root@master:~# kubectl apply -f prod-pod.yaml
pod/my-pod configured
```

--check

```
root@master:~# kubectl get pods -n prod-environment
NAME      READY   STATUS    RESTARTS       AGE
my-pod    1/1     Running   1 (106s ago)   31m
```

→describe pod:

```
root@master:~# kubectl describe pod my-pod -n prod-environment
Name:             my-pod
Namespace:        prod-environment
```

```
    Ready:          True
    Restart Count:  1
    Limits:
      cpu:     500m
      memory:  500Mi
    Requests:
      cpu:        200m
      memory:     200Mi
    Environment:  <none>
```

3) **In the staging-environment namespace, set a LimitRange that assigns default CPU and memory limits (300m CPU, 600Mi memory) and applies a minimum and maximum CPU.**

→create name space:

```
root@master:~# kubectl create ns staging-environment
namespace/staging-environment created
root@master:~# kubectl get ns
NAME                  STATUS        AGE
default               Active        4d
dev-environment       Terminating   71m
kube-node-lease       Active        4d
kube-public           Active        4d
kube-system           Active        4d
prod-environment      Terminating   47m
staging-environment   Active        6s
tigera-operator       Active        3d23h
```

→yaml file to create limit range:

root@master: ~

```
apiVersion: v1
kind: LimitRange
metadata:
  name: default-limits
  namespace: staging-environment
spec:
  limits:
  - default:
      cpu: "300m"
      memory: "600Mi"
    defaultRequest:
      cpu: "300m"
      memory: "600Mi"
    min:
      cpu: "100m"
      memory: "200Mi"
    max:
      cpu: "1000m"
      memory: "2Gi"
    type: Container

~
```

--run yaml file

```
root@master:~# kubectl apply -f limitrange.yaml
limitrange/default-limits created
```

→describe:

```
root@master:~# kubectl describe ns staging-environment
Name:         staging-environment
Labels:       kubernetes.io/metadata.name=staging-environment
Annotations:  <none>
Status:       Active

No resource quota.

Resource Limits
 Type       Resource  Min    Max  Default Request  Default Limit  Max Limit/Request Ratio
 ----       --------  ---    ---  ---------------  -------------  -----------------------
 Container  cpu       100m   1    300m             300m           -
 Container  memory    200Mi  2Gi  600Mi            600Mi          -
```

4) Create a pod and a NodePort service in the default namespace, then create another pod in the test namespace and communicate between them using Service DNS.

→ yaml file to create a pod in default namespace:

root@master: ~

```
apiVersion: v1
kind: Pod
metadata:
  name: httpd-pod
  namespace: default
spec:
  containers:
  - name: httpd
    image: httpd
    ports:
    - containerPort: 80
~
```

--run the yaml then pod created in default namespace:

```
root@master:~# kubectl apply -f httpd-pod.yaml
pod/httpd-pod created
root@master:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
httpd-pod   1/1     Running   0          23s
testpod     1/1     Running   0          49m
```

→now create the nodeport service:

```
root@master: ~
apiVersion: v1
kind: Service
metadata:
  name: httpd-service
  namespace: default
spec:
  selector:
    app: httpd
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30081
  type: NodePort
```

→run yaml then created nodeport service:

```
root@master:~# vi httpd-service.yaml
root@master:~# kubectl apply -f httpd-service.yaml
service/httpd-service created
```

```
root@master:~# kubectl get services
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
apache-service    ClusterIP   10.108.182.53   <none>        80/TCP         3d1h
echo-service      NodePort    10.107.50.179   <none>        80:30080/TCP   2d22h
firstpod-service  ClusterIP   10.111.36.119   <none>        80/TCP         3d20h
httpd-service     NodePort    10.99.142.63    <none>        80:30081/TCP   2m22s
```

→now Create the Pod in the test Namespace:

--first create test name space:

```
root@master:~# kubectl create ns test
namespace/test created
```

--now create a pod in test ns

root@master: ~

```
apiVersion: v1
kind: Pod
metadata:
  name: curl-pod
  namespace: test
spec:
  containers:
  - name: curl
    image: curlimages/curl
    command:
      - sleep
      - "3600"
```

--run the yaml

```
root@master:~# kubectl apply -f curl-pod.yaml
pod/curl-pod created
```

→now check the connectivity:

--in local

```
root@master:~# curl 3.148.252.247:30081
<html><body><h1>It works!</h1></body></html>
```

-- we can exec into the curl-pod and test the communication with httdpod by the following command:

kubectl exec -it curl-pod -n test -- /bin/sh

now check with curl DNS

curl httpd-service.default.svc.cluster.local

```
~ $ curl httpd-service.default.svc.cluster.local
<html><body><h1>It works!</h1></body></html>
```

5)Apply a LimitRange with a max limit/request ratio of 2 for memory in the performance-environment namespace, and test by creating a pod with mismatched resource requests and limits.

→create a namespce:

```
root@master:~# kubectl create namespace performance-environment
namespace/performance-environment created
```

Check ns

```
root@master:~# kubectl get ns
NAME                      STATUS        AGE
default                   Active        4d1h
dev-environment           Terminating   138m
kube-node-lease           Active        4d1h
kube-public               Active        4d1h
kube-system               Active        4d1h
performance-environment   Active        17s
```

→create yaml

```
root@master: ~
apiVersion: v1
kind: LimitRange
metadata:
  name: memory-limit-range
  namespace: performance-environment
spec:
  limits:
  - max:
      memory: "4Gi"  # Maximum memory limit for any pod in this namespace
    min:
      memory: "512Mi"  # Minimum memory limit for any pod in this namespace
    default:
      memory: "1Gi"  # Default memory request if not specified
    defaultRequest:
      memory: "1Gi"  # Default memory request for the pod
    type: Container
  - maxLimitRequestRatio:
      memory: "2"  # Max memory limit to request ratio (limit can't be more than 2x the request)
    type: Pod
```

--run the yaml

```
root@master:~# kubectl apply -f limitrange-memory.yaml
limitrange/memory-limit-range created
```

--check describe

kubectl describe ns performance-environment

```
root@master:~# kubectl describe ns performance-environment
Name:         performance-environment
Labels:       kubernetes.io/metadata.name=performance-environment
Annotations:  <none>
Status:       Active

No resource quota.

Resource Limits
 Type       Resource  Min    Max  Default Request  Default Limit  Max Limit/Request Ratio
 ----       --------  ---    ---  ---------------  -------------  -----------------------
 Container  memory    512Mi  4Gi  1Gi              1Gi            -
 Pod        memory    -      -    -                -              2
```

→ testing by Creating a Pod with Mismatched Resource Requests and Limits

Yaml file:

root@master: ~

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
  namespace: performance-environment
spec:
  containers:
  - name: test-container
    image: nginx
    resources:
      requests:
        memory: "1Gi"  # Memory request
      limits:
        memory: "3Gi"  # Memory limit (3Gi is greater than 2x the memory request, so it will be rejected)
```

--run the yaml

kubectl apply -f test-pod.yaml

--not created pod got an error because the pod was rejected due to the LimitRange enforcement

```
root@master:~# kubectl apply -f test-pod.yaml
Error from server (Forbidden): error when creating "test
 ratio is 3.000000
```