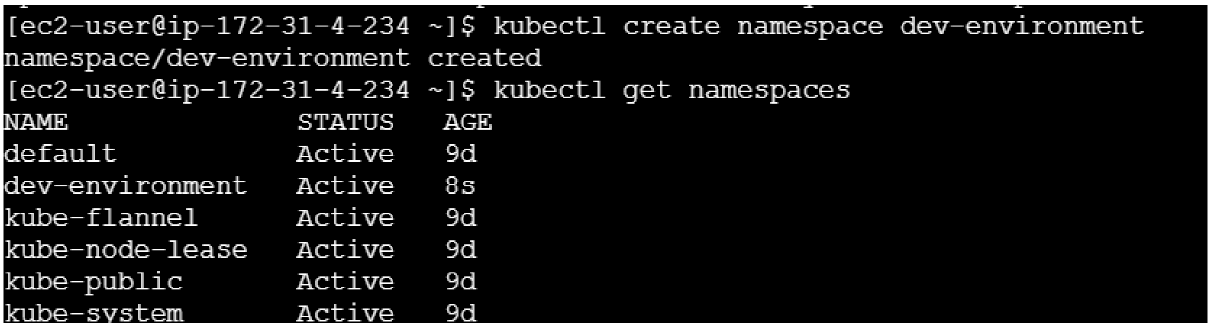
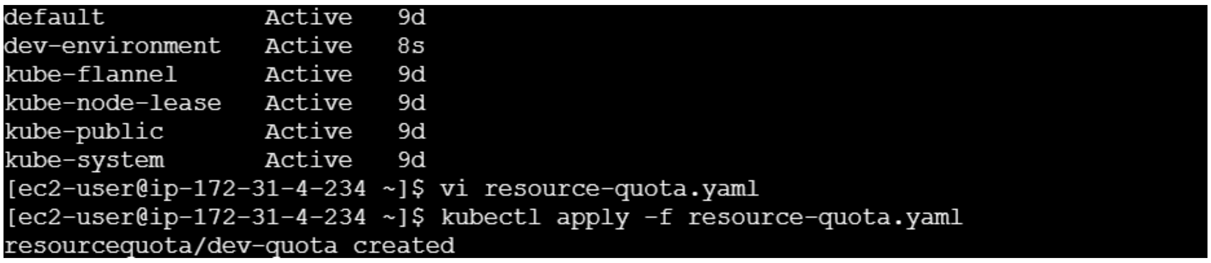
Kubernetes 05

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

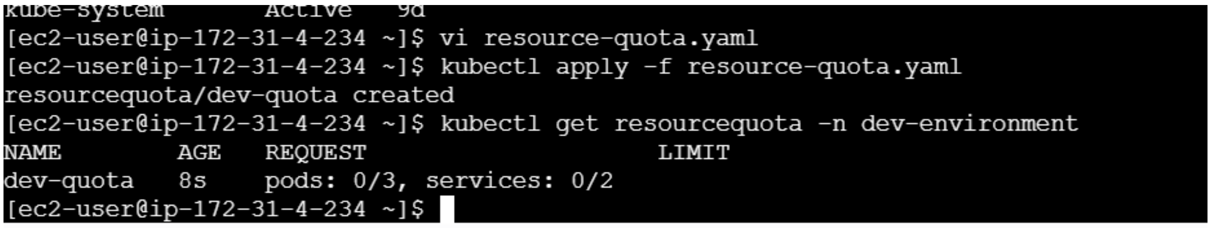
- Create  dev environment
- kubectl create namespace dev-environment
- 

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl create namespace dev-environment
namespace/dev-environment created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get namespaces
NAME              STATUS   AGE
default           Active   9d
dev-environment   Active   8s
kube-flannel      Active   9d
kube-node-lease   Active   9d
kube-public       Active   9d
kube-system       Active   9d
```

- Niw create a yaml file
- Vi resource-quota.yml
- apiVersion: v1
- kind: ResourceQuota
- metadata:
-   name: dev-quota
-   namespace: dev-environment
- spec:
-   hard:
-     pods: "3"
-     services: "2"

- Add deploy it
- 

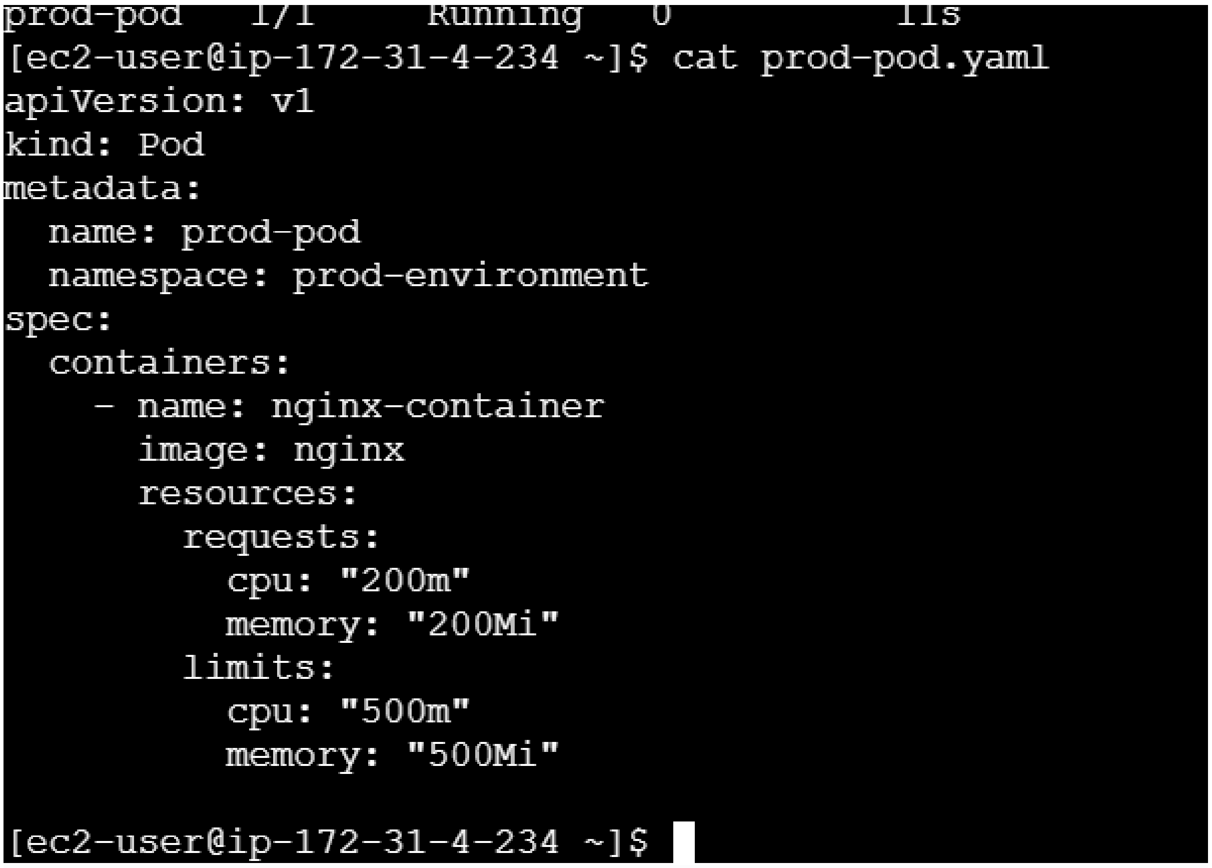
```
default            Active   9d
dev-environment    Active   8s
kube-flannel       Active   9d
kube-node-lease    Active   9d
kube-public        Active   9d
kube-system        Active   9d
[ec2-user@ip-172-31-4-234 ~]$ vi resource-quota.yaml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f resource-quota.yaml
resourcequota/dev-quota created
```

- Verify
- kubectl get resourcequota -n dev-environment
- 

```
kube-system        Active   9d
[ec2-user@ip-172-31-4-234 ~]$ vi resource-quota.yaml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f resource-quota.yaml
resourcequota/dev-quota created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get resourcequota -n dev-environment
NAME        AGE   REQUEST                      LIMIT
dev-quota   8s    pods: 0/3, services: 0/2
[ec2-user@ip-172-31-4-234 ~]$ 
```

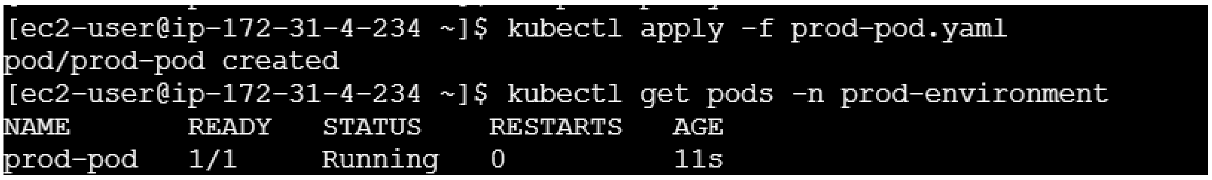
2 . Create a pod in the prod-environment namespace with 0.2 CPU and 200Mi memory requests, and 0.5 CPU and 500Mi memory limits.

- Create prod environment
- kubectl create namespace prod-environment
- now create  a yaml file



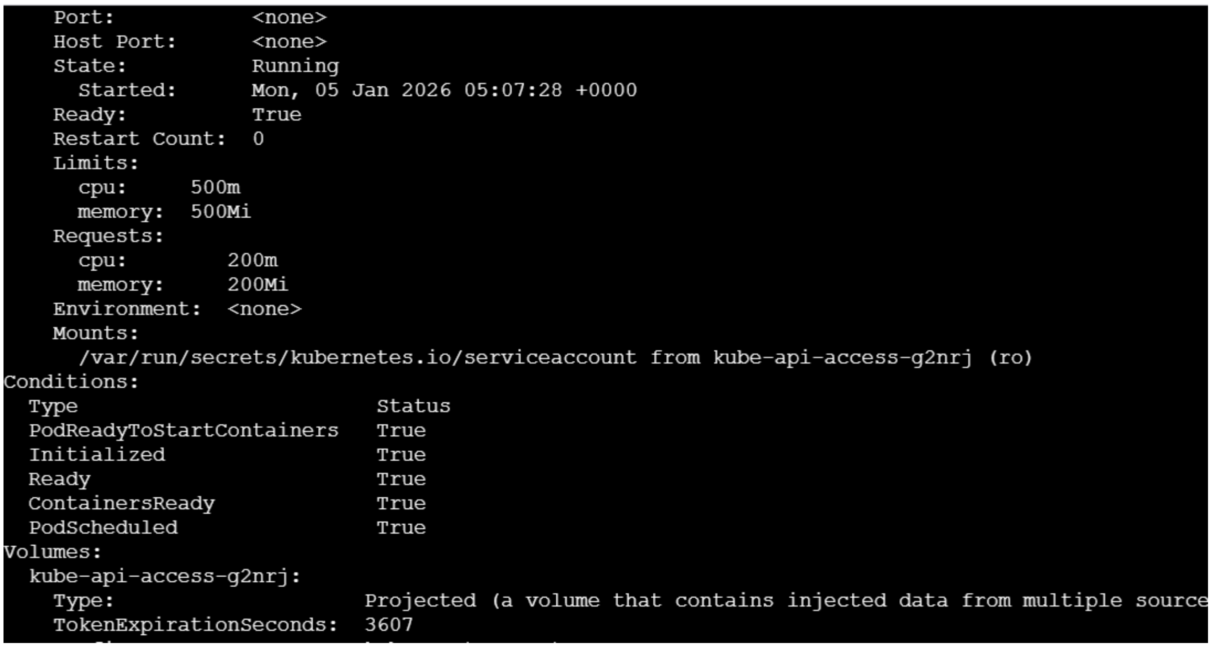
```
prod-pod    1/1     Running   0          11s
[ec2-user@ip-172-31-4-234 ~]$ cat prod-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: prod-pod
  namespace: prod-environment
spec:
  containers:
    - name: nginx-container
      image: nginx
      resources:
        requests:
          cpu: "200m"
          memory: "200Mi"
        limits:
          cpu: "500m"
          memory: "500Mi"

[ec2-user@ip-172-31-4-234 ~]$
```

- 
- Deploy it
- Kubectl appy -f prod-pod.yml
- Verify  it
- kubectl get pods -n prod-environment
- 

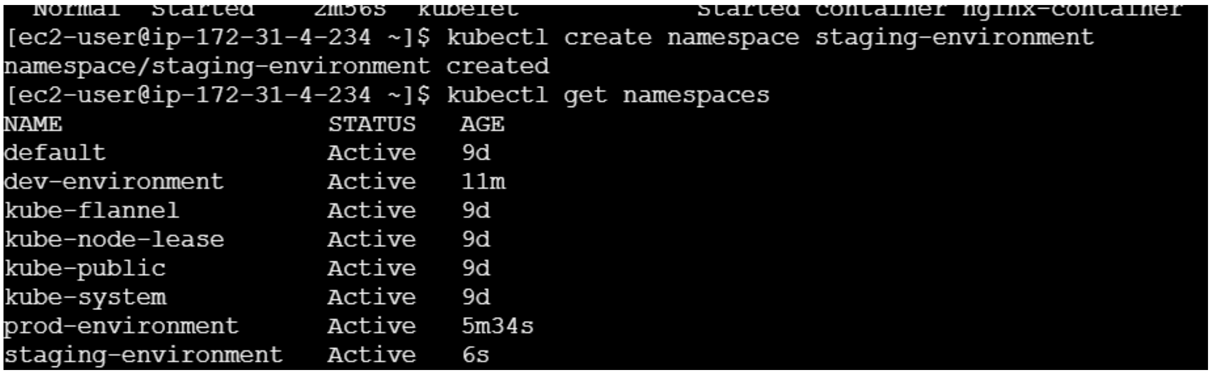
```
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f prod-pod.yaml
pod/prod-pod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pods -n prod-environment
NAME       READY   STATUS    RESTARTS   AGE
prod-pod   1/1     Running   0          11s
```

- kubectl describe pod prod-pod -n prod-environment
- this show the memory and cpu
- - Guaranteed **0.2 CPU / 200Mi memory**
- - Capped at **0.5 CPU / 500Mi memory**
-

- 

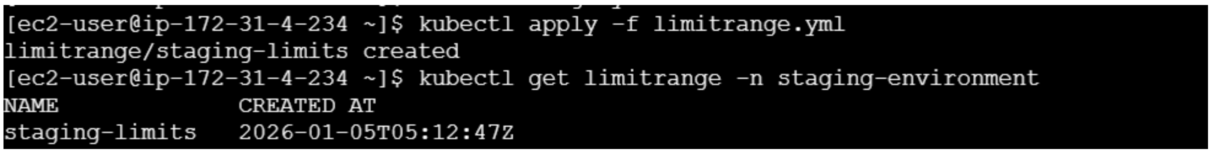
```
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 05 Jan 2026 05:07:28 +0000
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     500m
      memory:  500Mi
    Requests:
      cpu:        200m
      memory:     200Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-g2nrj (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-g2nrj:
    Type:                    Projected (a volume that contains injected data from multiple source
    TokenExpirationSeconds:  3607
```

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  Environmetn
- Kubectl creaste  namespace staging
- 

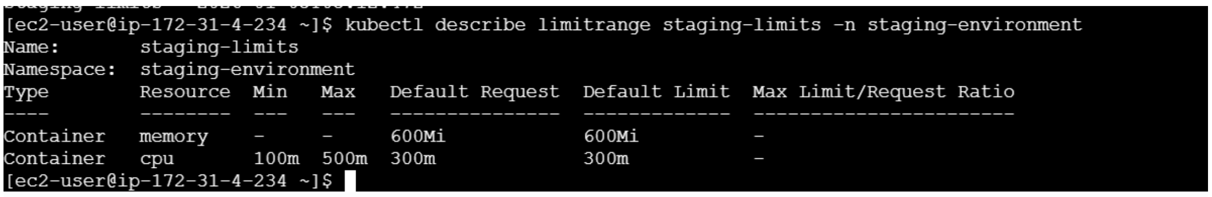
```
  Normal  Started    2m56s  kubelet           Started container nginx-container
[ec2-user@ip-172-31-4-234 ~]$ kubectl create namespace staging-environment
namespace/staging-environment created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get namespaces
NAME                   STATUS   AGE
default                Active   9d
dev-environment        Active   11m
kube-flannel           Active   9d
kube-node-lease        Active   9d
kube-public            Active   9d
kube-system            Active   9d
prod-environment       Active   5m34s
staging-environment    Active   6s
```

- Now,
- Write a limitrange.yml file
- apiVersion: v1
- kind: LimitRange
- metadata:

- name: staging-limits
- namespace: staging-environment
- spec:
- limits:
- - type: Container
- default:
- cpu: "300m"
- memory: "600Mi"
- defaultRequest:
- cpu: "300m"
- memory: "600Mi"
- min:
- cpu: "100m"
- max:
- cpu: "500m"
- Now deploy it
- Kubectl apply -f limitrange.yml
- 

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f limitrange.yml
limitrange/staging-limits created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get limitrange -n staging-environment
NAME             CREATED AT
staging-limits   2026-01-05T05:12:47Z
```

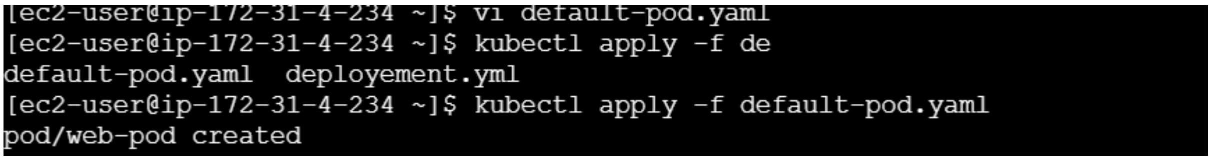
- Now verify it
- kubectl get limitrange -n staging-environment
- get detailed view
- kubectl describe limitrange staging-limits -n staging-environment



```
[ec2-user@ip-172-31-4-234 ~]$ kubectl describe limitrange staging-limits -n staging-environment
Name:       staging-limits
Namespace:  staging-environment
Type        Resource  Min   Max   Default Request  Default Limit  Max Limit/Request Ratio
----        --------  ---   ---   ---------------  -------------  -----------------------
Container   memory    -     -     600Mi            600Mi          -
Container   cpu       100m  500m  300m             300m           -
[ec2-user@ip-172-31-4-234 ~]$
```

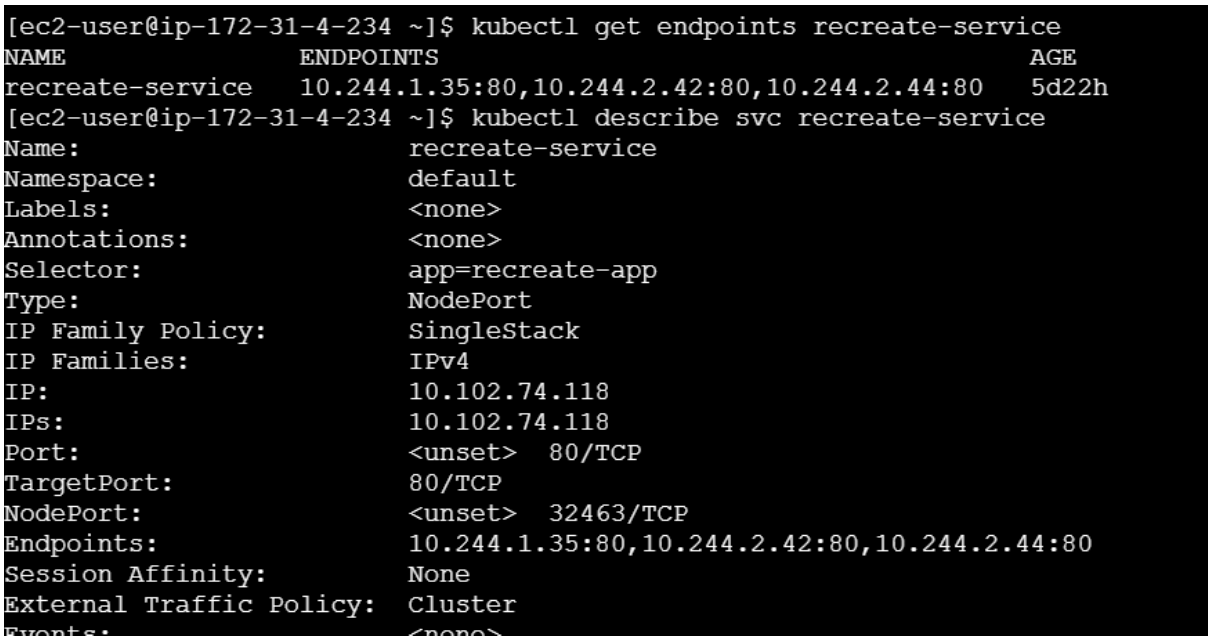
-

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.

- Vi default-pod.yaml
- apiVersion: v1
- kind: Pod
- metadata:
-   name: web-pod
-   labels:
-     app: web
- spec:
-   containers:
-     - name: nginx
-       image: nginx
-       ports:
-         - containerPort: 80
- Deploy it
- Kubectl apply -f default-pod.yml
- 

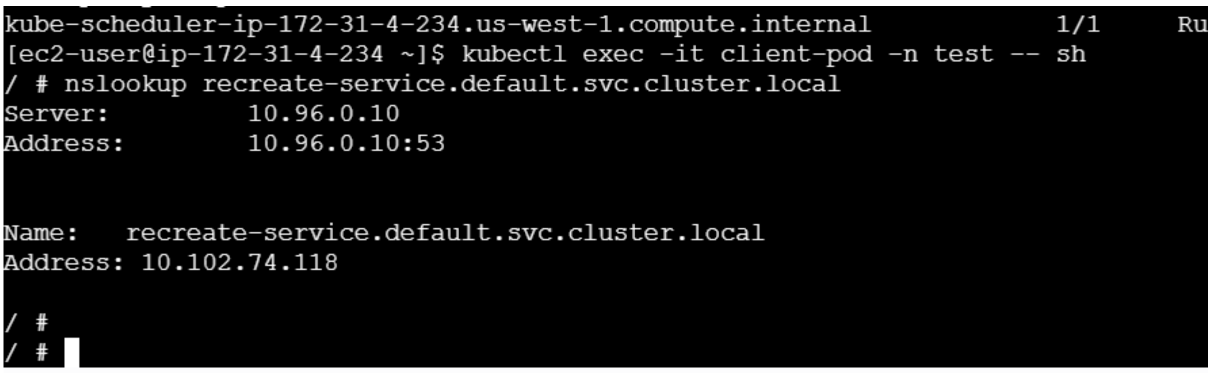
```
[ec2-user@ip-172-31-4-234 ~]$ vi default-pod.yaml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f de
default-pod.yaml  deployement.yml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f default-pod.yaml
pod/web-pod created
```

- Create a NodePort Service in default

- Vi web.service.yml
- apiVersion: v1
- kind: Service
- metadata:
-   name: web-service
- spec:
-   type: NodePort
-   selector:
-     app: web
-   ports:
-     - port: 80
-       targetPort: 80
-       nodePort: 30080
- Deploy it
- kubectl apply -f web-service.yaml
- 

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl get endpoints recreate-service
NAME                ENDPOINTS                                         AGE
recreate-service    10.244.1.35:80,10.244.2.42:80,10.244.2.44:80     5d22h
[ec2-user@ip-172-31-4-234 ~]$ kubectl describe svc recreate-service
Name:                     recreate-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=recreate-app
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.102.74.118
IPs:                      10.102.74.118
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  32463/TCP
Endpoints:                10.244.1.35:80,10.244.2.42:80,10.244.2.44:80
Session Affinity:         None
External Traffic Policy:  Cluster
```

- Dns is working perfect

- 

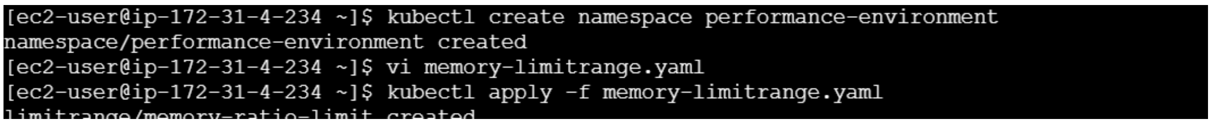
```
kube-scheduler-ip-172-31-4-234.us-west-1.compute.internal          1/1     Ru
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec -it client-pod -n test -- sh
/ # nslookup recreate-service.default.svc.cluster.local
Server:         10.96.0.10
Address:        10.96.0.10:53


Name:   recreate-service.default.svc.cluster.local
Address: 10.102.74.118

/ #
/ #
```

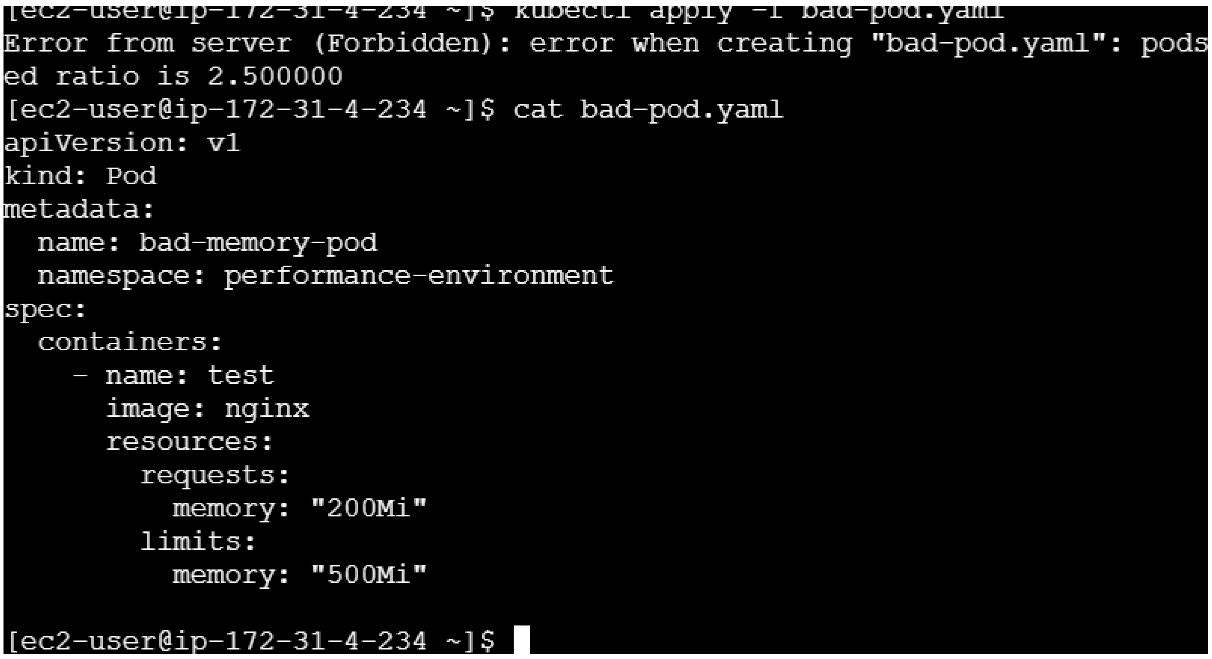
1. 5 . Apply a LimitRange with a max limit/request ratio of 2 for memory in the performance-environment namespace, and test by creating a pod with mismatched resource requests and limits.

- Create a namespace
-  Now wirte a Yaml file
- apiVersion: v1
- kind: LimitRange
- metadata:
-   name: memory-ratio-limit
-   namespace: performance-environment
- spec:
-   limits:
-     - type: Container
-       maxLimitRequestRatio:
-         memory: "2"
- Deploy it
- By
- Kubectl apply -f memory-limitrange.yml

- 

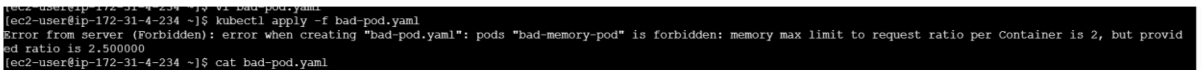
```
[ec2-user@ip-172-31-4-234 ~]$ kubectl create namespace performance-environment
namespace/performance-environment created
[ec2-user@ip-172-31-4-234 ~]$ vi memory-limitrange.yaml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f memory-limitrange.yaml
limitrange/memory-ratio-limit created
```

- Now
- Wtite a testing yaml file as
- Bad-pod.yml
- apiVersion: v1
- kind: Pod
- metadata:
-   name: bad-memory-pod
-   namespace: performance-environment
- spec:
-   containers:
-     - name: test
-       image: nginx
-       resources:
-         requests:
-           memory: "200Mi"
-         limits:
-           memory: "500Mi"when we try to deploy it

- 

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f bad-pod.yaml
Error from server (Forbidden): error when creating "bad-pod.yaml": pods
ed ratio is 2.500000
[ec2-user@ip-172-31-4-234 ~]$ cat bad-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: bad-memory-pod
  namespace: performance-environment
spec:
  containers:
    - name: test
      image: nginx
      resources:
        requests:
          memory: "200Mi"
        limits:
          memory: "500Mi"

[ec2-user@ip-172-31-4-234 ~]$
```

- It doesn’t deploy
- Kubectl apply -f bad-pod.yml
- It throws error
- 

```
[ec2-user@ip-172-31-4-234 ~]$ vi bad-pod.yaml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f bad-pod.yaml
Error from server (Forbidden): error when creating "bad-pod.yaml": pods "bad-memory-pod" is forbidden: memory max limit to request ratio per Container is 2, but provid
ed ratio is 2.500000
[ec2-user@ip-172-31-4-234 ~]$ cat bad-pod.yaml
```