***Resource Quota Horizontal Pod Autoscaling***

**Limit= Request**

(BY DEFAULT VALUES OF CPU & MEMORY IN PODS)

**cpu**

request= 0.5 (min)

limit= 1 (max)

**memory**

request= 500MB (min)

limit= 1GB (max)

***LAB***

//if limit is mention and request is not mention so the limt=request let do this with example

nano cpu2.yml

apiVersion: v1

kind: Pod

metadata:

name: default-cpu-demo-2

spec:

containers:

- name: default-cpu-demo-2-ctr

image: nginx

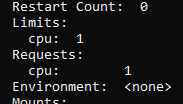
resources:

limits:

cpu: "1"

* kubectl apply -f cpu2.yml
* kubectl get pods
* kubectl describe pod default-cpu-demo-2

//we did not mentioned request here so by default we get same value limit =request



* kubectl delete -f cpu2.yml

//now we are mention request value here and exulted limit

nano cpu1.yml

apiVersion: v1

kind: Pod

metadata:

name: default-cpu-demo-3

spec:

containers:

- name: default-cpu-demo-3-ctr

image: nginx

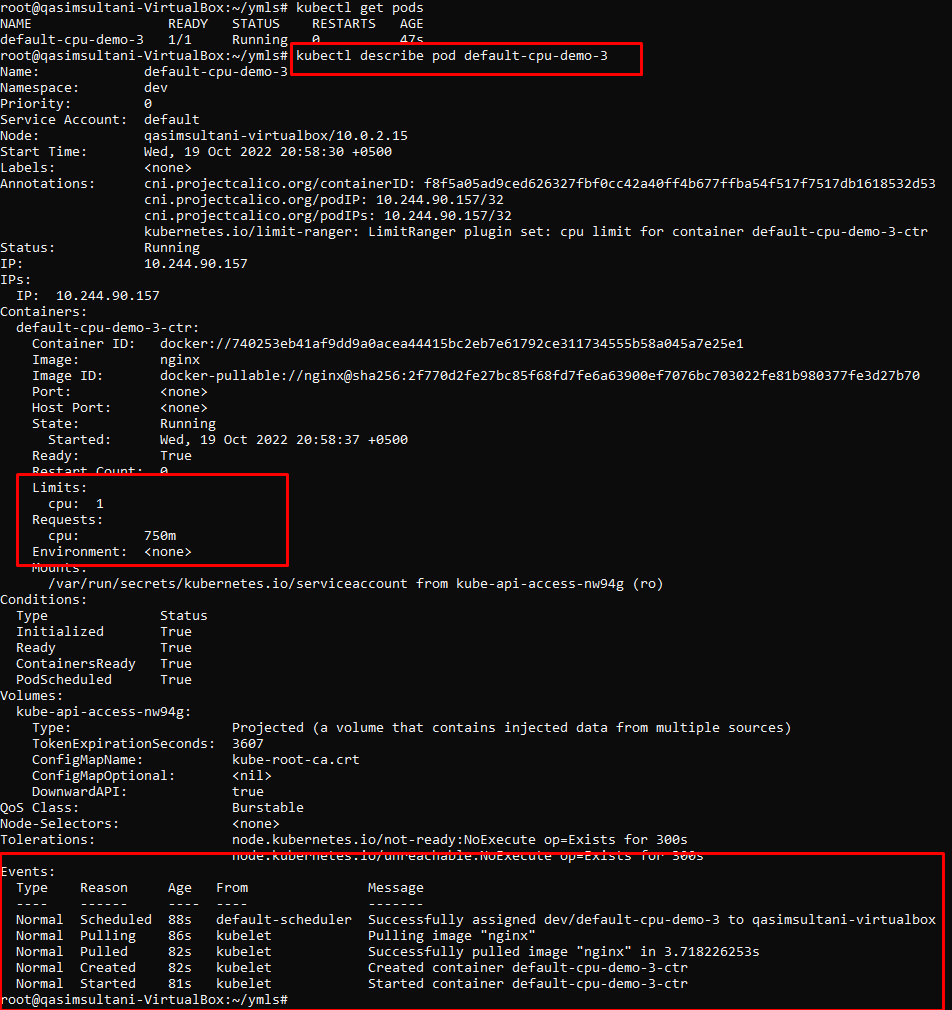
resources:

requests:

cpu: "0.75"

* kubectl apply -f cpu1.yml
* kubectl get pods
* kubectl describe pod default-cpu-demo-3

//sometimes they did not show the limit but when you not give any limit it’s directly imagine that they have 1



**//now we are working same request & limit scenario in memory**

* nano mem.yml

apiVersion: v1

kind: LimitRange

metadata:

name: mem-min-max-demo-lr

spec:

limits:

- max:

memory: 1Gi

min:

memory: 500Mi

type: Container

* kubectl apply -f mem.yml

**//now creating pod with in the limit and check respons wherther is working fine**

* nano mem1.yml

apiVersion: v1

kind: Pod

metadata:

name: constraints-mem-demo

spec:

containers:

- name: constraints-mem-demo-ctr

image: nginx

resources:

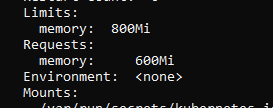
limits:

memory: "800Mi"

requests:

memory: "600Mi"

* kubectl apply -f mem1.yml
* kubectl get pods
* kubectl describe pod constraints-mem-demo



* kubectl delete -f mem1.yml

**//now creating pod out of the limit memory and check response whether is working fine**

apiVersion: v1

kind: Pod

metadata:

name: constraints-mem-demo

spec:

containers:

- name: constraints-mem-demo-ctr

image: nginx

resources:

limits:

memory: "1400Mi"

requests:

memory: "600Mi"

* kubectl apply -f mem2.yml

hence pod is not created because the givien memory is out of the resource

**//now creating pod out of the request in memory and check response whether is working fine**

apiVersion: v1

kind: Pod

metadata:

name: constraints-mem-demo

spec:

containers:

- name: constraints-mem-demo-ctr

image: nginx

resources:

limits:

memory: "800Mi"

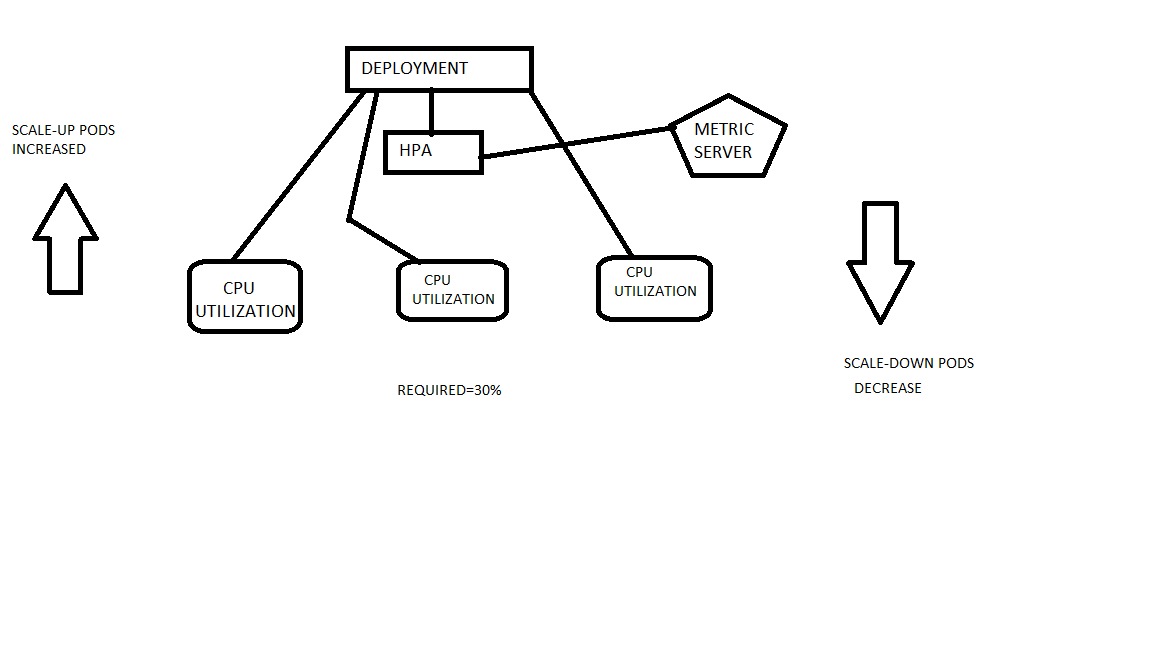
requests:

memory: "300Mi"

* kubectl apply -f mem2.yml



***Horizontal Pod Autoscaling***

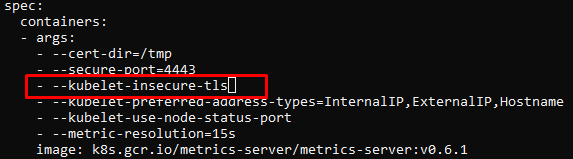


//download the metric server

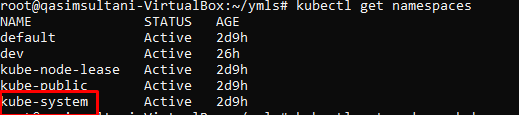
wget -O metricserver.yml <https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml>

//now goto metrics yml file and put this command so they did not ask for the certificate for you to install.

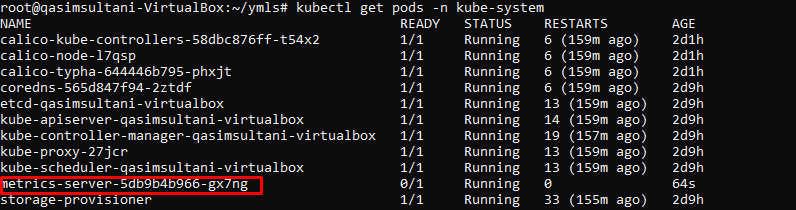
* - --kubelet-insecure-tls



* kubectl apply -f metricserver.yml
* kubectl get namespaces



* kubectl get pods -n kube-system



* kubectl logs -f metrics-server-5db9b4b966-gx7ng -n kube-system /to check it’s working

**//now creating file to auto scale the on HPA**

* nano deployhpa.yml

kind: Deployment

apiVersion: apps/v1

metadata:

name: mydeploy

spec:

replicas: 1

selector:

matchLabels:

name: deployment

template:

metadata:

name: testpod8

labels:

name: deployment

spec:

containers:

- name: c00

image: httpd

ports:

- containerPort: 80

resources:

limits:

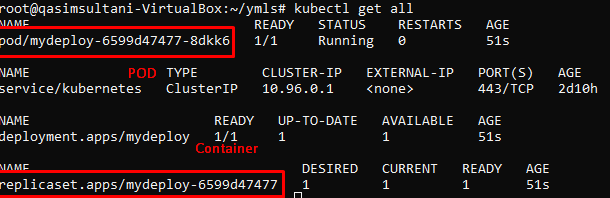
cpu: 500m

requests:

cpu: 200m

* kubectl apply -f deployhpa.yml
* kubectl apply all

//here you see pods container object



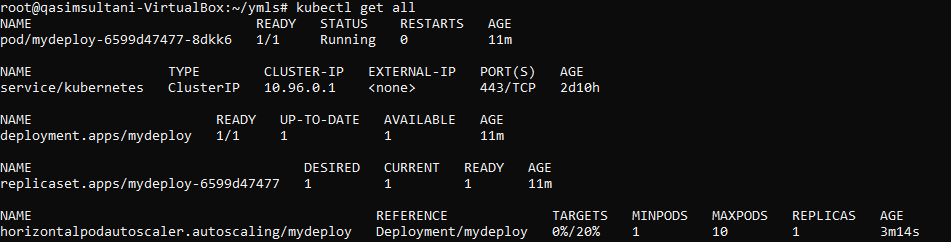
//enabling auto-scaling object

* kubectl autoscale deployment mydeploy --cpu-percent=20 --min=1 --max=10



//now check is your autoscaled is enable & working

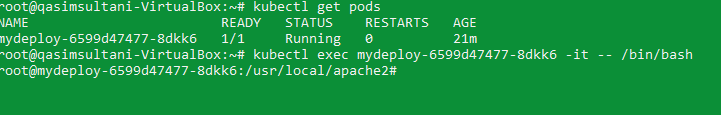
* kubectl get all



//now giving load to pod so that target value increased on horizontalpod right now it’s 0% and it’s increased once giving load

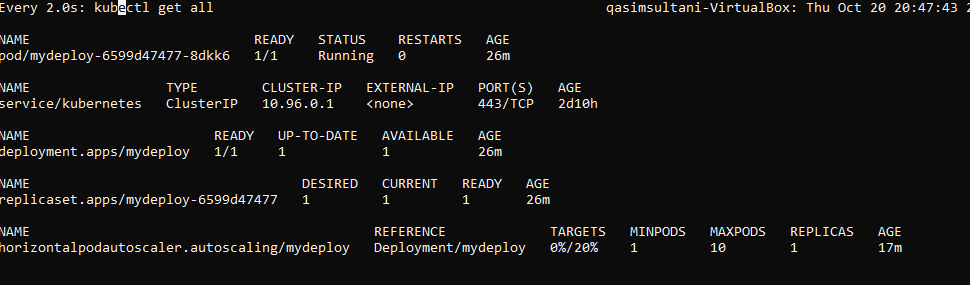
//take the access of the same node again on your terminal and change the screen colour

* kubectl get pods
* kubectl exec mydeploy-6599d47477-8dkk6 -it -- /bin/bash

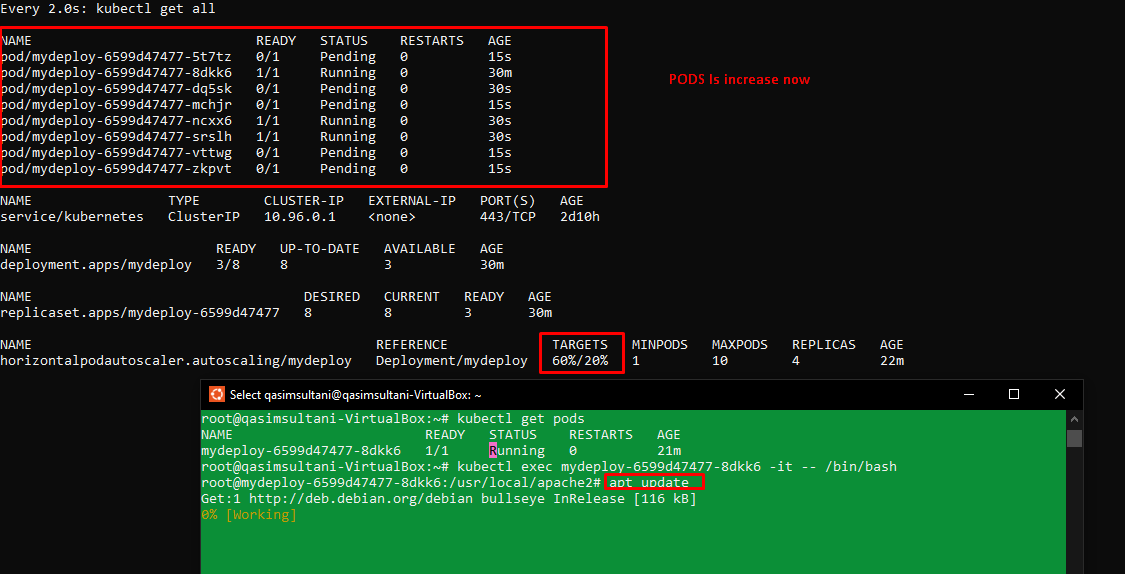


//go to the terminal 1 and run the command to check the pod is creating in your terminal and cpu usage is increased “it’s check in every 2 sec if the load is increased so pods is also increased ”

* watch kubectl get all //to check pods is creating in run time



//go to the terminal 2 which you change colour and update it or try to give load so that pods in increased in terminal 1



//hence you can see that pods is increased once load is increased, the pod is automatically down-scaled once the CPU usage is decreased