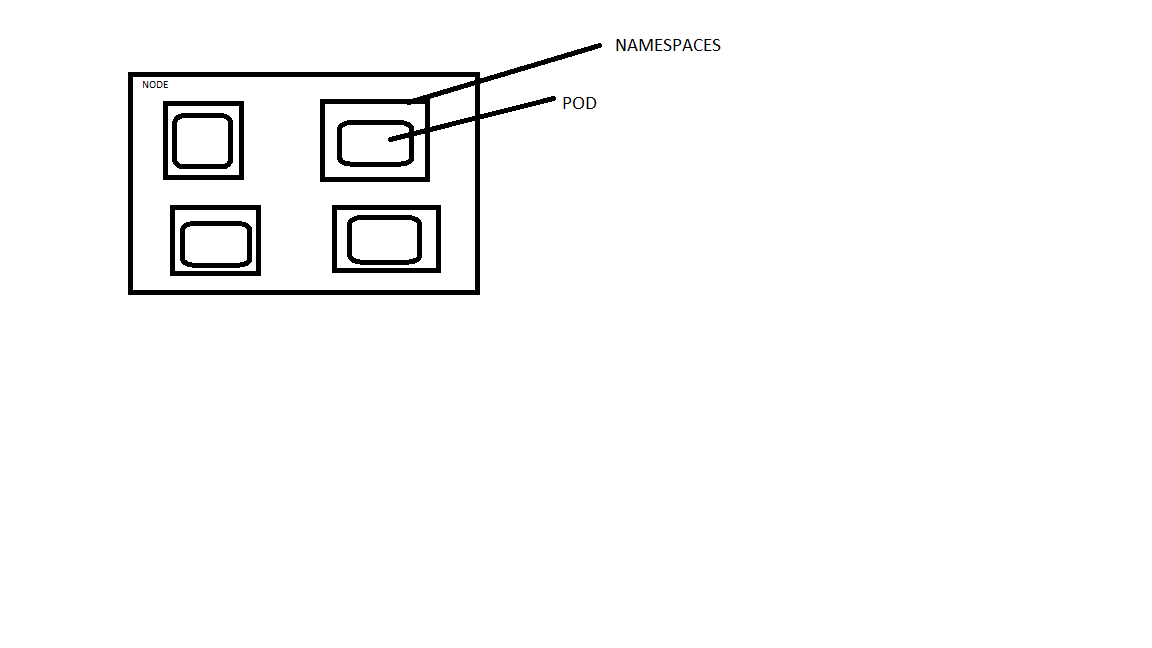
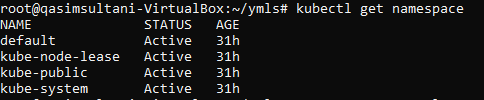
***Namespaces and Resource Quota***

Namespaces is also known as boundary or extra layer outside the pod in cluster. It is used when there are multiple application are running in your cluster so it give tag to your application like this pod belongs to this application



//by default k8s search pods on default namespace





***LAB***

//create namespace and then pod

* nano namespace.yml

apiVersion: v1

kind: Namespace

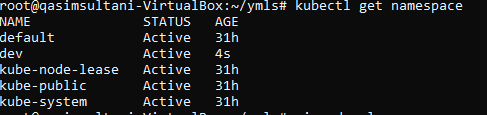
metadata:

name: dev

labels:

name: dev

* kubectl apply -f namespace.yml
* kubectl get namespace // to view namespaces



* nano pod.yml

kind: Pod

apiVersion: v1

metadata:

name: testpod

spec:

containers:

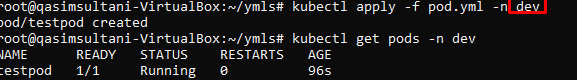
- name: c00

image: ubuntu

command: ["/bin/bash", "-c", "while true; do echo Nice to meet you; sleep 5 ; done"]

restartPolicy: Never

* kubectl apply -f pod.yml -n dev //for creating pods in your specific namespaces
* kubectl get pods -n dev //to view pods in your specific namespaces



kubectl delete pods testpod -n dev or //delete pod

kubectl delete -f pod.yml -n dev //delete all work in yml file

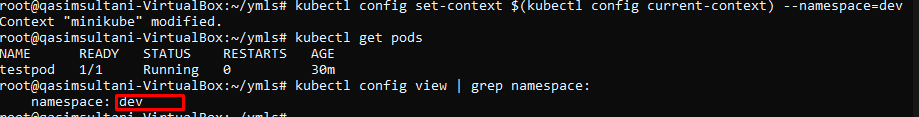


**//whenever we check any pods in namespaces or any information it goes on default but we want to check this on specific namespaces so here we want to map the namespaces so whatever changes we want to view it goes to that specific namespace we map it.**



**//now to change namespaces so that we do,**

* kubectl config set-context $(kubectl config current-context) --namespace=dev
* kubectl get pods
* kubectl config view | grep namespace:



* kubectl delete -f pod.yml

***RESOURCE QUOTA***

Whenever we create pods it will take all resources by defaults suppose we have 30GB it will take all. So, we give some limited resources to pods manually like memory and CPU, RAM as needed

Resources memory limit cpu all things are given to the namespaces.

* nano .resources.yml

apiVersion: v1

kind: Pod

metadata:

name: resources

spec:

containers:

- name: resource

image: centos

command: ["/bin/bash", "-c", "while true; do echo QASIM; sleep 5 ; done"]

resources:

requests:

memory: "64Mi"

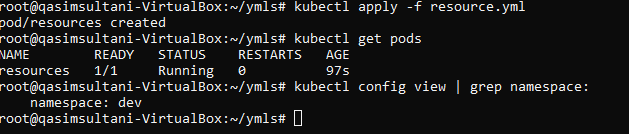
cpu: "100m"

limits:

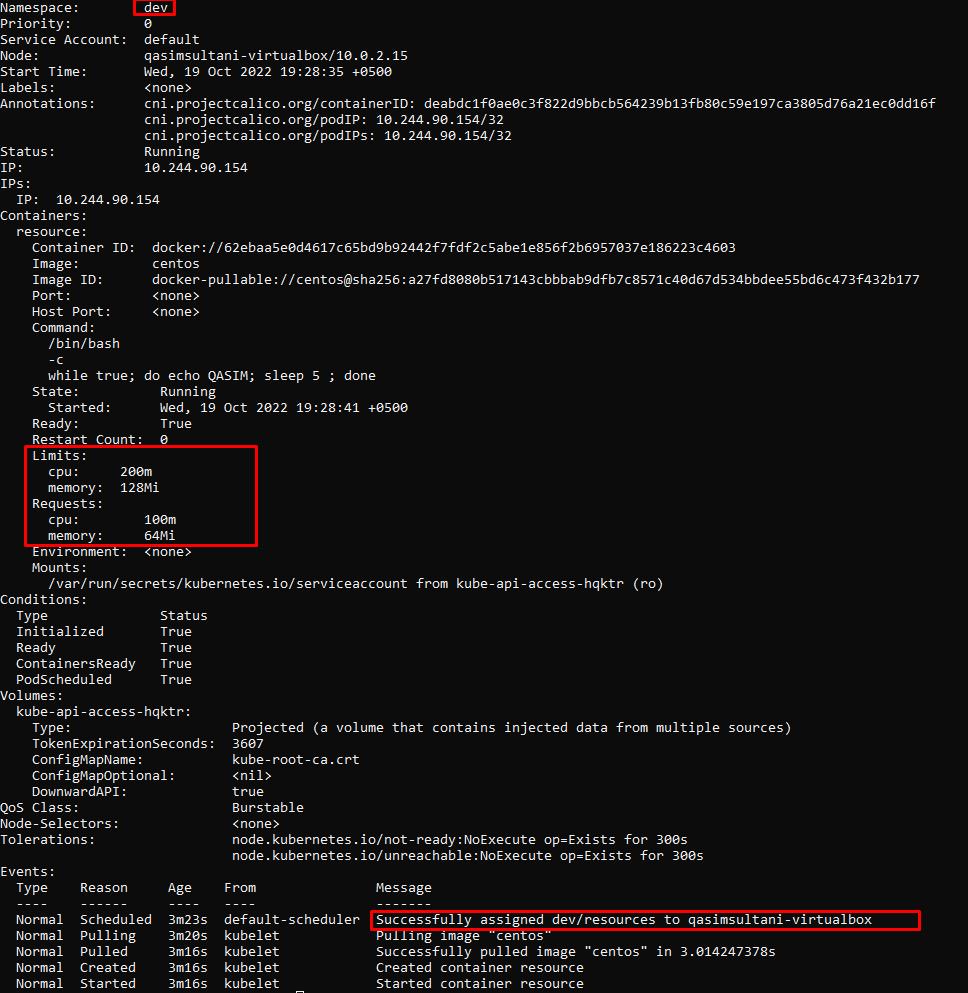
memory: "128Mi"

cpu: "200m"

* kubectl apply -f resource.yml /you have to create dev namespace recently and map it to as default-namespaces so by default pods are created in dev namespaces
* kubectl get pods
* kubectl config view | grep namespace: //to view current namespaces



* kubectl describe pod resources



* kubectl delete -f resource.yml

//creating resource quota to the namespaces

* nano resourcequota.yml

apiVersion: v1

kind: ResourceQuota

metadata:

name: myquota

spec:

hard:

limits.cpu: "400m"

limits.memory: "400Mi"

requests.cpu: "200m"

requests.memory: "200Mi"

* kubectl apply -f resourcequota.yml
* nano testpod.yml

kind: Deployment

apiVersion: apps/v1

metadata:

name: deployments

spec:

replicas: 3

selector:

matchLabels:

objtype: deployment

template:

metadata:

name: testpod8

labels:

objtype: deployment

spec:

containers:

- name: c00

image: ubuntu

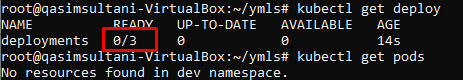
command: ["/bin/bash", "-c", "while true; do echo QASIM; sleep 5 ; done"]

resources:

requests:

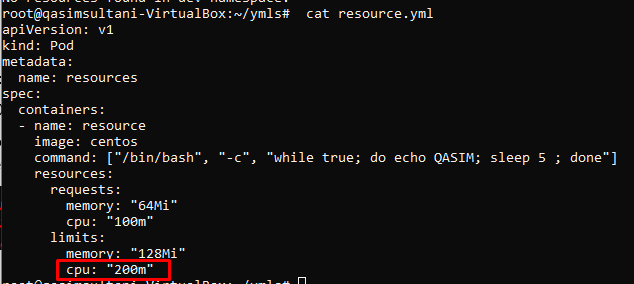
cpu: "200m"

* kubectl apply -f testpod.yml
* kubectl get deploy

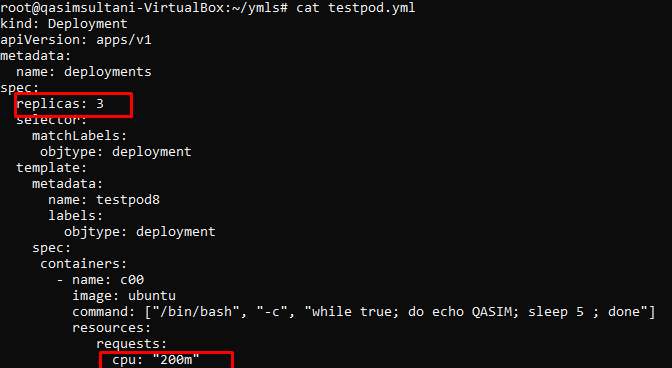


//here container is not working because the reason is resources of CPU are exceed from the given limits on testpod.yml

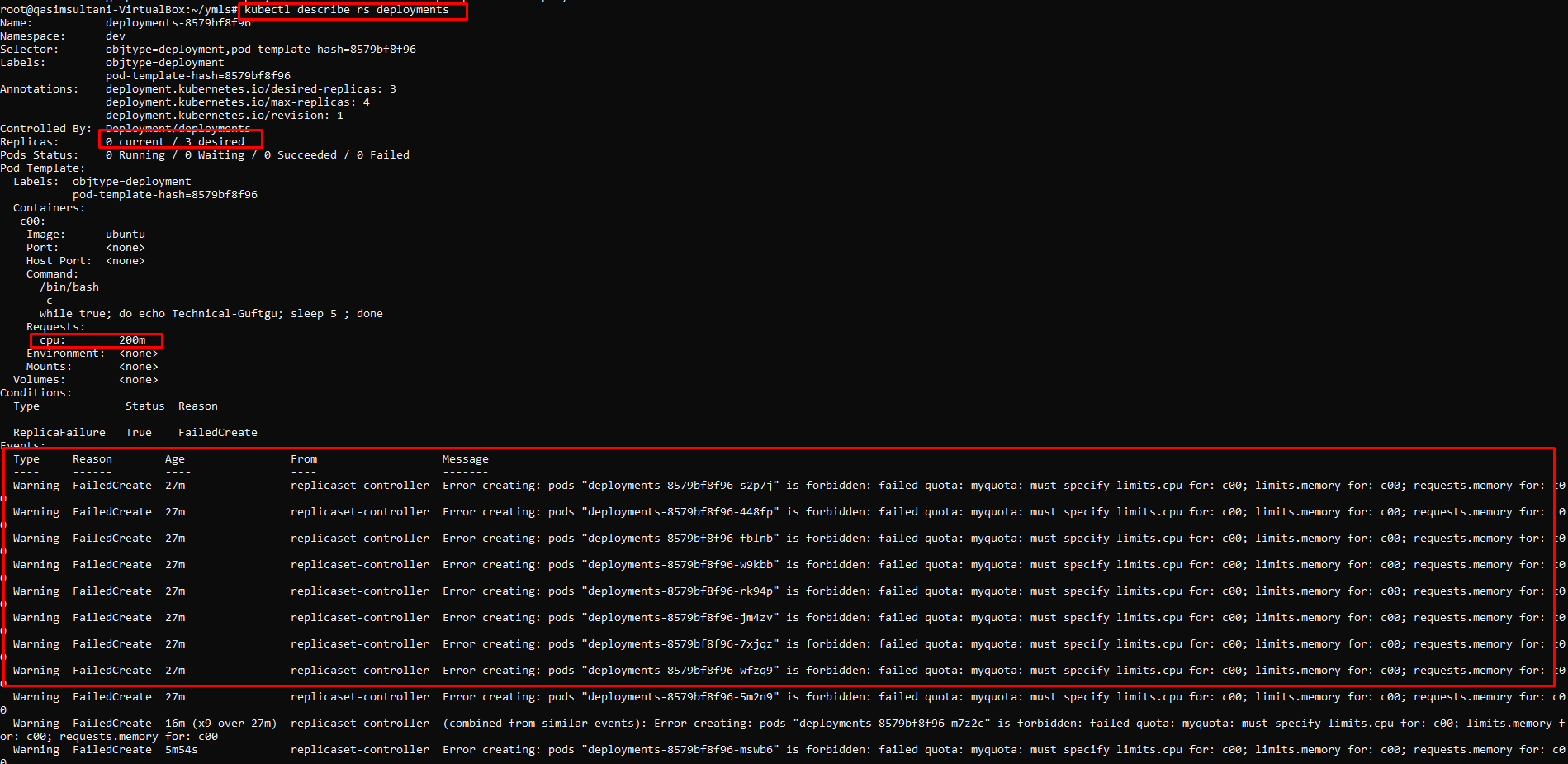
//the current limit is 200



//but in current file pod have 3 replicas & there is CPU limit is 200m so it’s became 600m and we have limit of 200m so that why pod is not created



* kubectl describe rs deployments /to check logs



* kubectl delete -f resourcequota.yml
* kubectl delete -f testpod.yml

//we now already check limit now let’s familiar with request

**//created the scenario with in the limited range resources needed**

* nano cpulimit.yml

apiVersion: v1

kind: LimitRange

metadata:

name: cpu-limit-range

spec:

limits:

- default:

cpu: 1

defaultRequest:

cpu: 0.5

type: Container

* nano pod.yml

kind: Pod

apiVersion: v1

metadata:

name: testpod

spec:

containers:

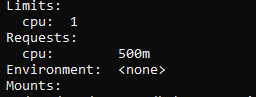
- name: c00

image: ubuntu

command: ["/bin/bash", "-c", "while true; do echo Nice to meet you; sleep 5 ; done"]

restartPolicy: Never

* kubectl apply -f pod.yml
* kubectl get pods
* kubectl describe pod testpod



//here you get enough resources because you required 500m & limit is 1 CPU.

* kubectl delete -f pod.yml

**//now create the scenario where you define the limit and not define the request.**

* nano cpu2.yml

apiVersion: v1

kind: Pod

metadata:

name: default-cpu-demo-2

spec:

containers:

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

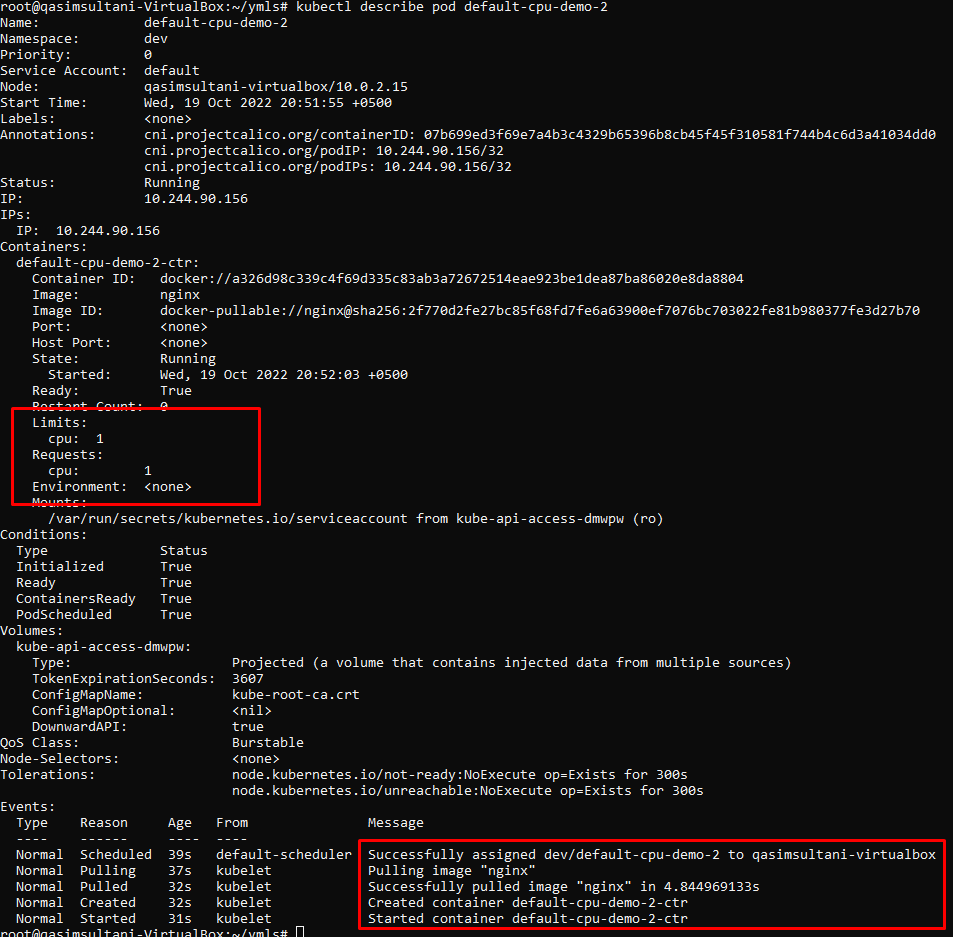
image: nginx

resources:

limits:

cpu: "1"

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



// here you get your request as same as limit if you not define the request because Limit=Request

* kubectl delete -f cpu2.yml

**//now create the scenario where you define the request and not define the limit.**

* nano cpu3.yml

apiVersion: v1

kind: Pod

metadata:

name: default-cpu-demo-3

spec:

containers:

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

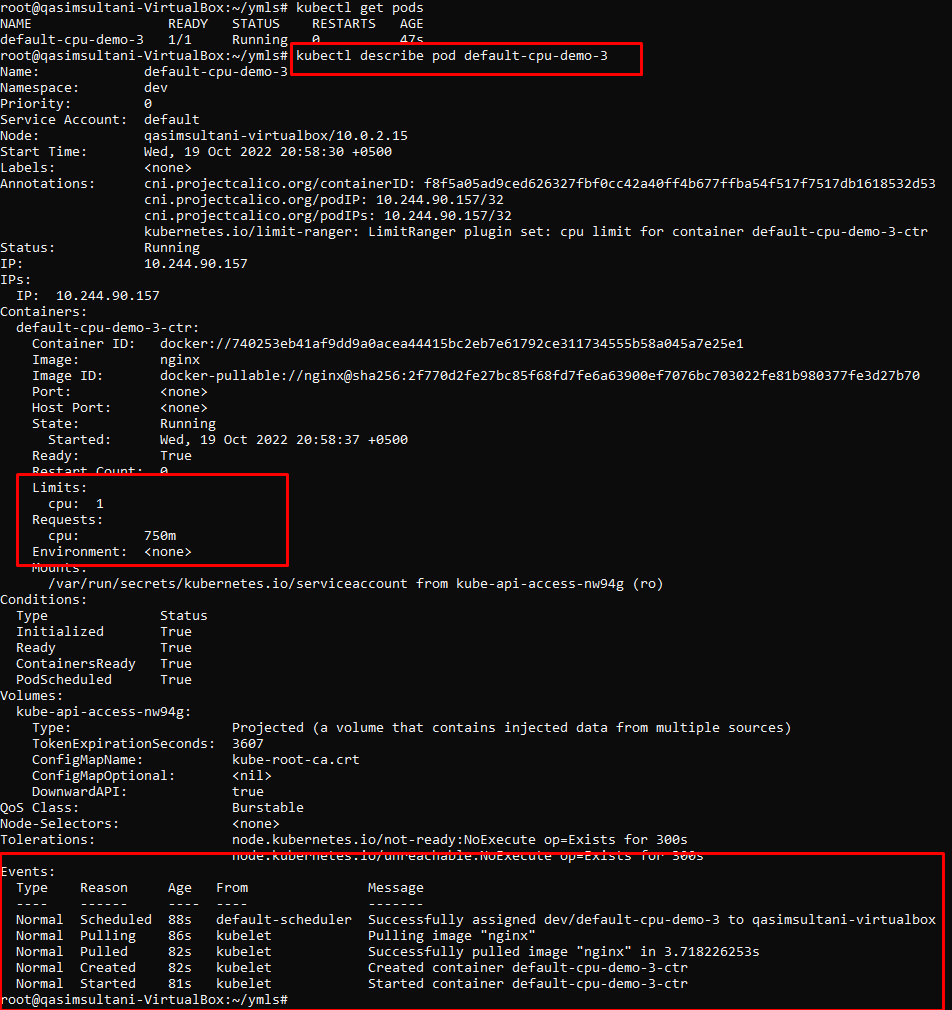
image: nginx

resources:

requests:

cpu: "0.75"

* kubectl apply -f cpu3.yml
* kubectl get pods
* kubectl describe pod default-cpu-demo-3



//here you get default limit that Is 1 cpu if you not define any limit

* kubectl delete -f cpu3.yml