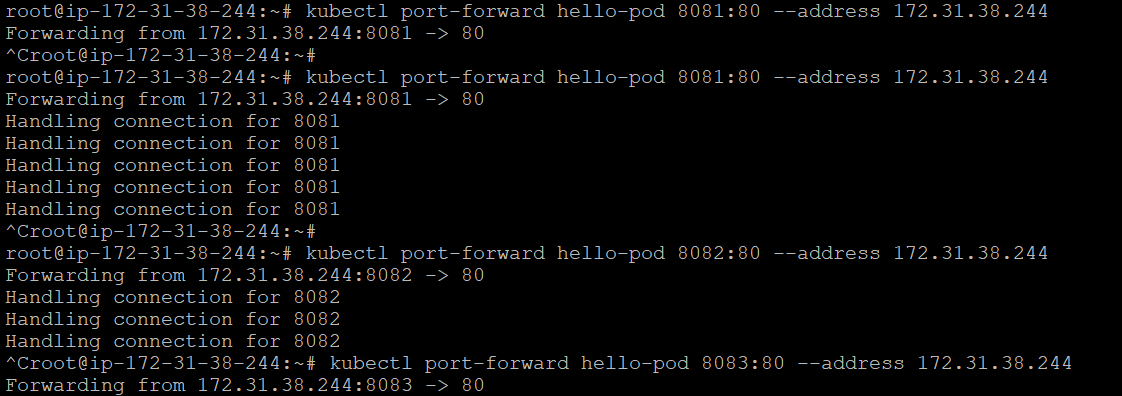
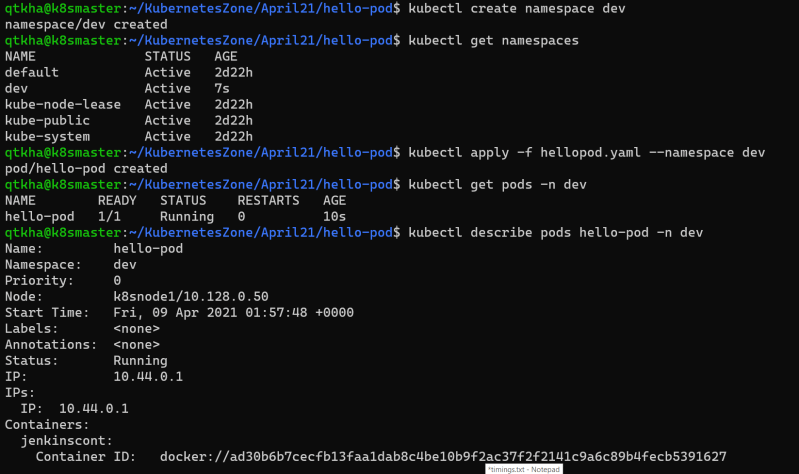
Pods port forwarding

kubectl port-forward <pod-name> <hostport:containerport> --address <private ip>

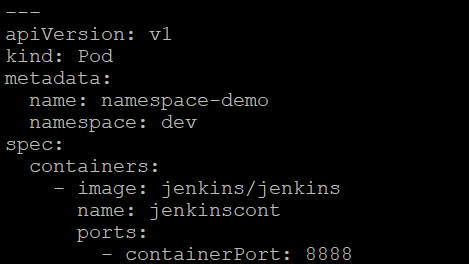
kubectl port-forward hello-pod 8081:8080 --address 10.128.0.49

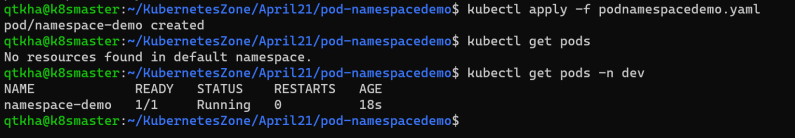
Can we have multiple port forwarding request to same pod? Yes



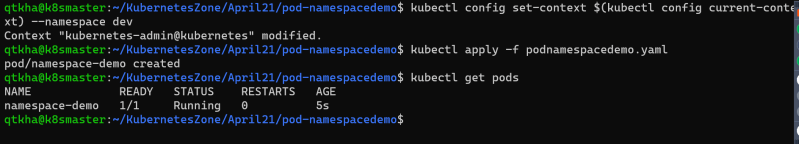
* Our Pod configuration is divided into four components
  + apiVersion: version of k8s api
  + kind: This is kind of kubernetes object in our case it is *Pod*
  + metadata: metadata or information that uniquely identifies the object which we are creating
  + spec: Specification of our pod with container name image, volume and resource requests
* Let’s create a namespace and run a pod within the namespace 

Namespace demo pod spec

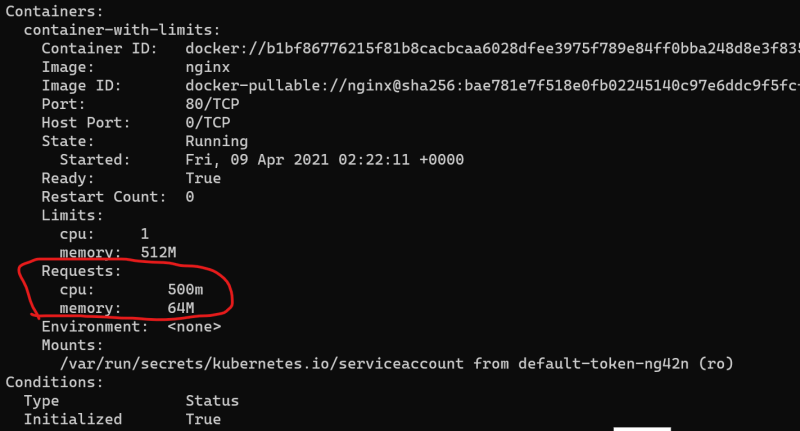
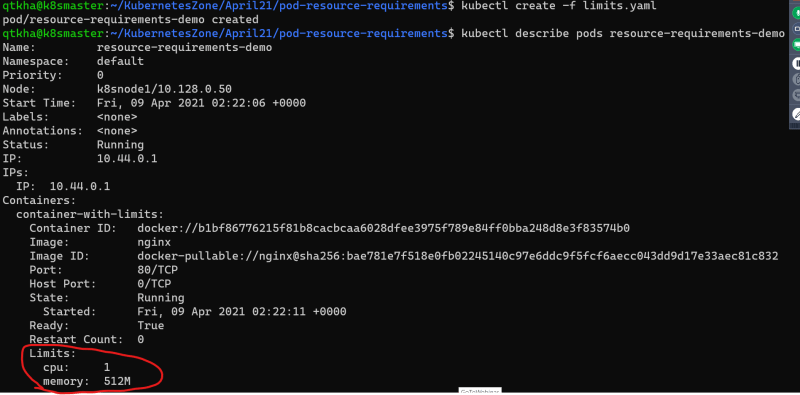


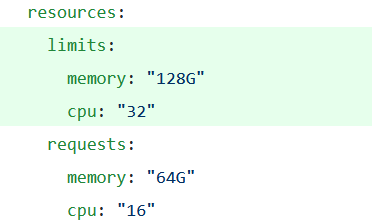
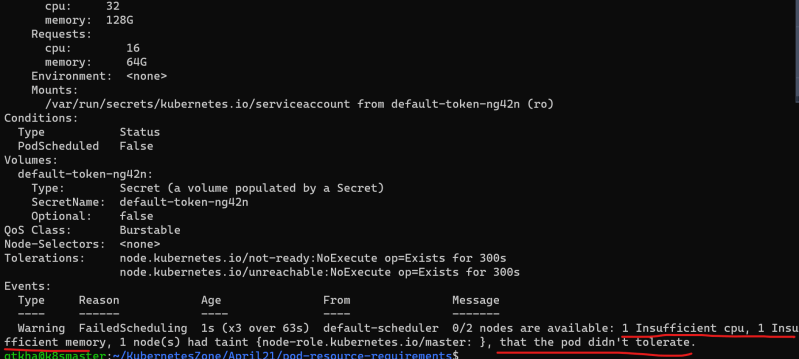
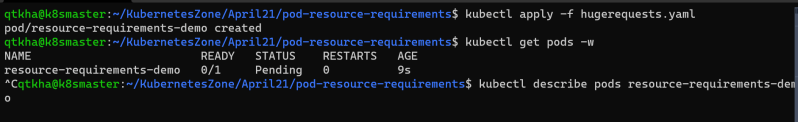


To change the default namespace to dev (or any other namespace)



Scenario: Create a Pod running a Container with Resource Requirements

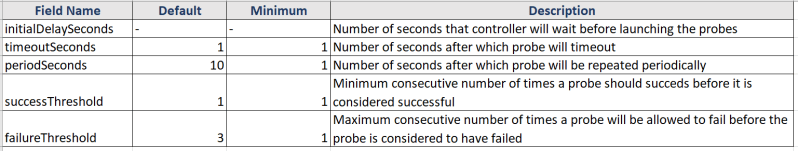


* Now let’s try to add the resource request which are very much huge that we don’t have any node suitable for the limit
* 
* 

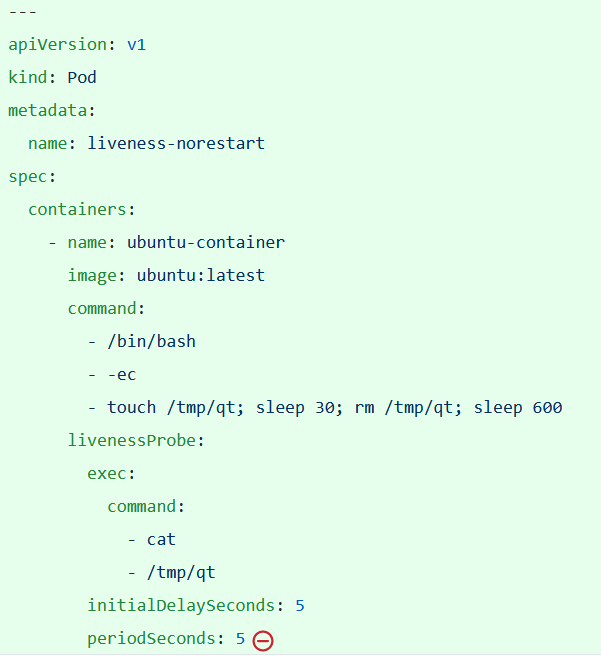
Kubernetes Pod Lifecycle

* Pod can have the following states
  + *Pending*: This means the pod has been submitted to the cluster, but the controller hasn’t created all its containers yets. It may be downloading images or waiting for the pod to be scheduled on one of its cluster nodes
  + *Running*: This state means pod is assigned to one of the cluster nodes and at least one of the containers is either running or is in the process startup
  + *Succeded*: This state means the pods has run and all of the container have been terminated with success
  + *Failed*: This state means that the pod has run & at least one of the containers has terminated with non-zero exit code (failure code)
  + *unknown*: This means theat the state of the pod could not be found

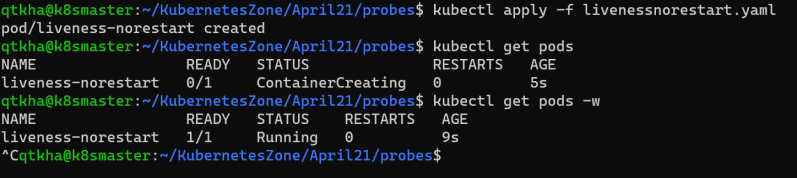
Probes/Health Checks

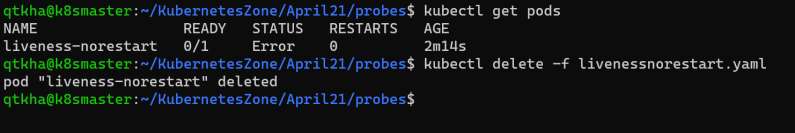
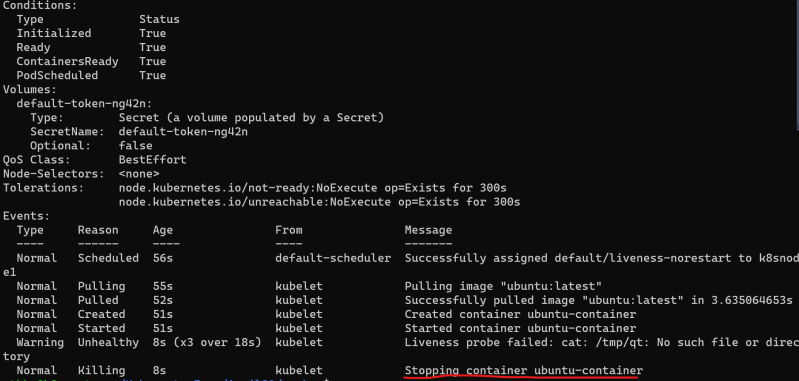
* A probe is a health check that can be configured to check the health of containers running in a Pod. A probe may return the following results
  + Success
  + Failure
  + Unknown
* Types of Probe
  + Liveness Probe: This is used to determine if the particular container is running or not. If a container fails the liveness probe, the controller will try to restart the pod on the same node according to the restart policy configured for the pod
  + Readiness Probe: This is used to determine whether a particular container is ready to receive requests or not. If this fails Kubernetes controller will ensure that the pod doesn’t receive any requests. If container specifies a readiness probe, its default state will be Failure until readiness probe succeeds.
* Configuration of probes 
* Implementation of Probes
  + Command Probe: In this probe controller will get the contaienr to execute specific command in order to perform probe on the container.
  + HTTP Request Probe: In this probe controller will send a GET HTTP request to the give address (host and port) to perform probe on the container. We can set the following fields to configure the HTTP Probe
    - host
    - path
    - port
    - httpHeaders
    - scheme:
  + TCP Socket Probe: In this probe, the controller will try to establish a connection on the given host and the specified port number. We can set the following fields for this probe
    - host
    - port
* RESTART policy:
  + We can specify *restartPolicy* in the pod specification to instruct controller about the conditions to restart the pod. The default value of *restartPolicy* is *Always*. The possible values are
    - Always: Always restart the pod when it terminates
    - OnFailure: Restart the pod only when it terminates with Failure
    - Never: Never restart the pod

Scenario 1: Create a Pod running a container with a liveness probe and no restart policy

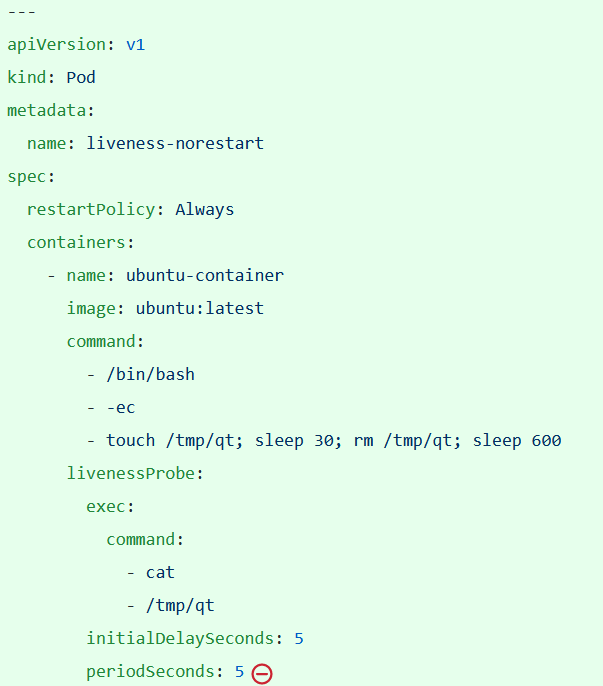


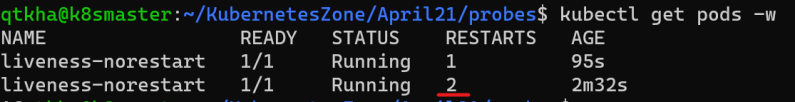
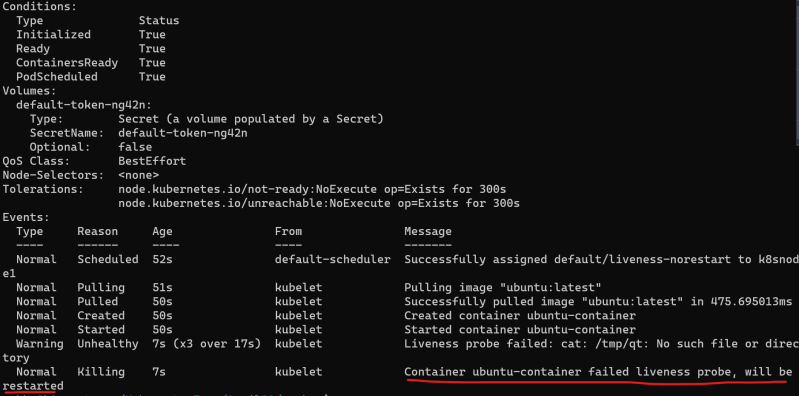
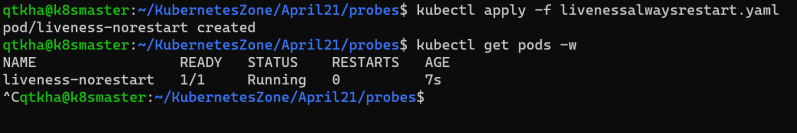


Create the pod 

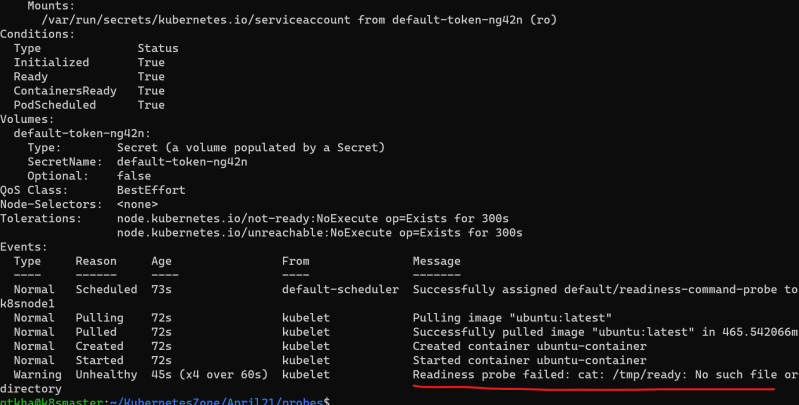
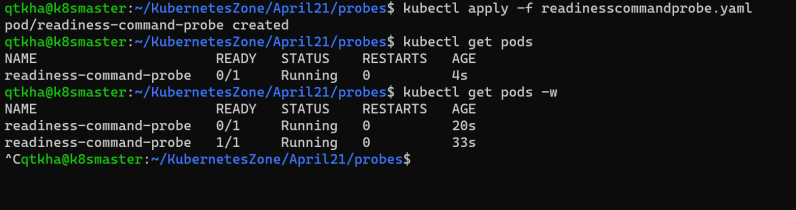
Describe the pod 

Scenario 2: Create a pod running a container with restartPolicy Always



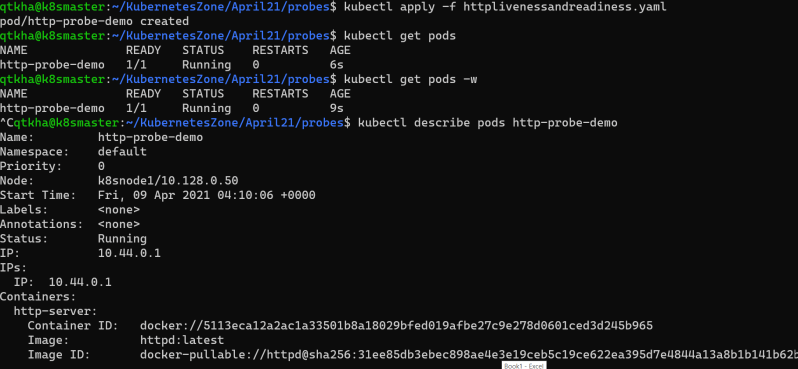


Scenario 3: Create a pod running a container with Readiness Probe 



Scenario 4: Create a pod with http Probe for liveness and readiness



Apply the manifest and get pods info 

Probe link: <https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/>