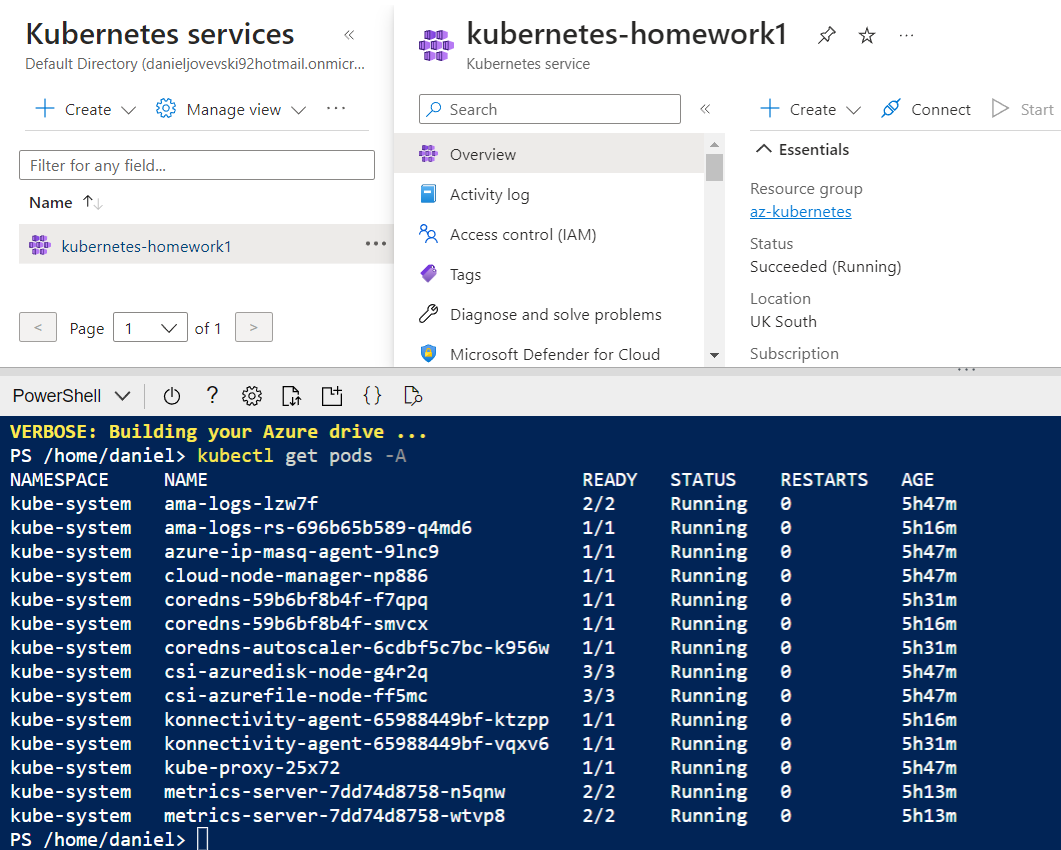
**Practice1: Simple pods operations**

Check how many pods run under the default namespace. Run kubectl get pods.

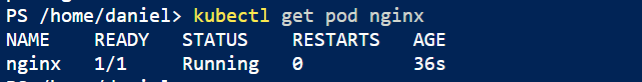
C:\Users\Daniel\Desktop\kubernetes\azure0.png

You should not see any pod under the default namespace. Now check all namespaces. Run kubectl get pods –all-namespace.

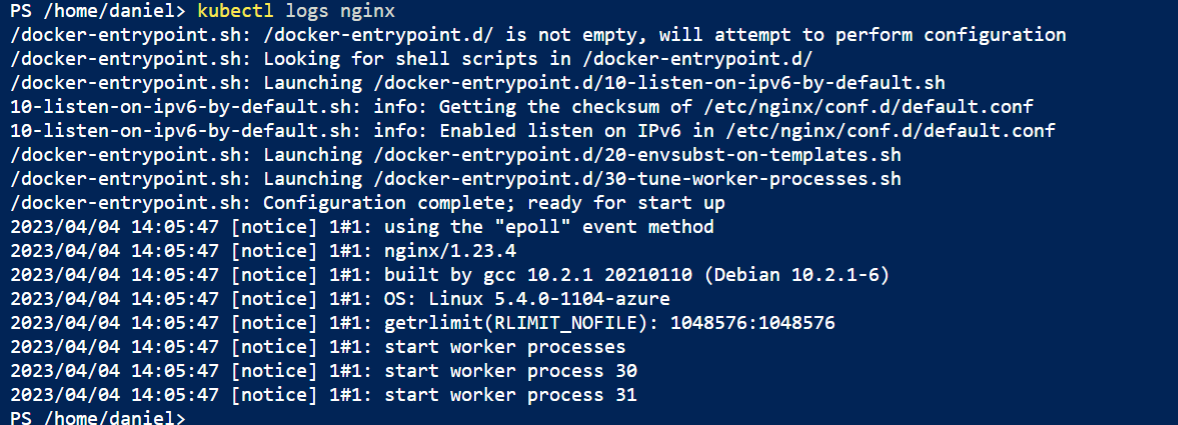


Now deploy you first pod using the imperative approach. Run kubectl run nginx --image=nginx.  
C:\Users\Daniel\Desktop\kubernetes\azure2.png

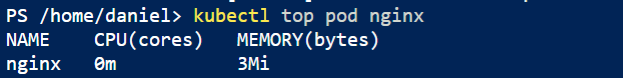
Validate if the pods has been created. What is the status of your pod?



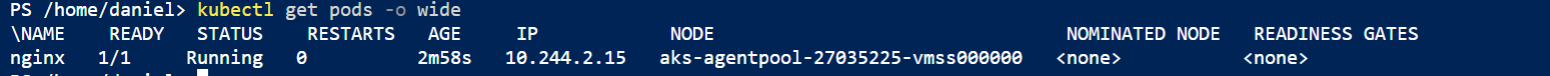
Check the logs coming out of your pod. Run kubectl logs nginx.



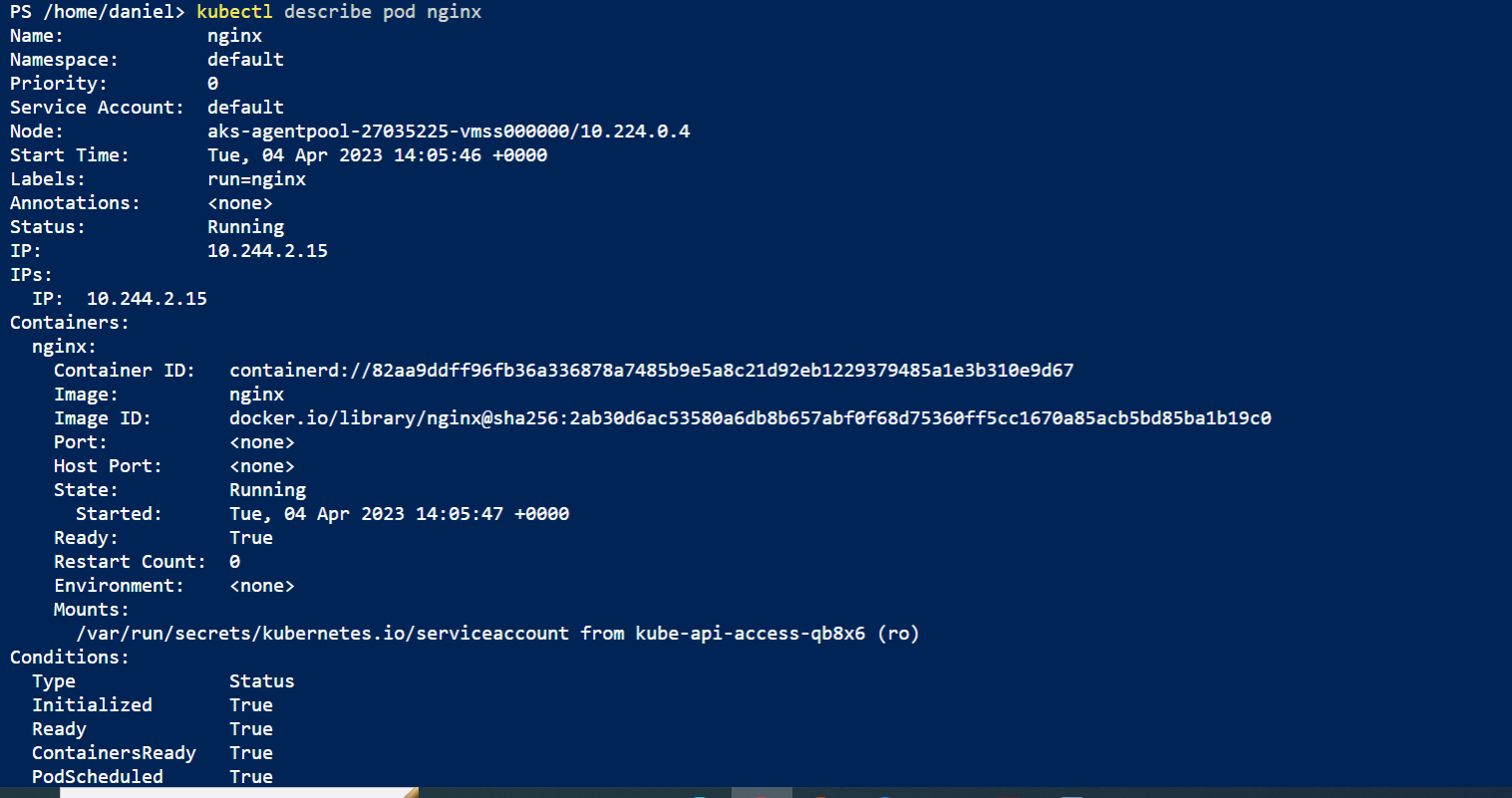
Run following command to check current resource consumption of your pod: kubectl top pod nginx.



Check on which Node your pods has been scheduled. Run kubectl get pods –o wide.

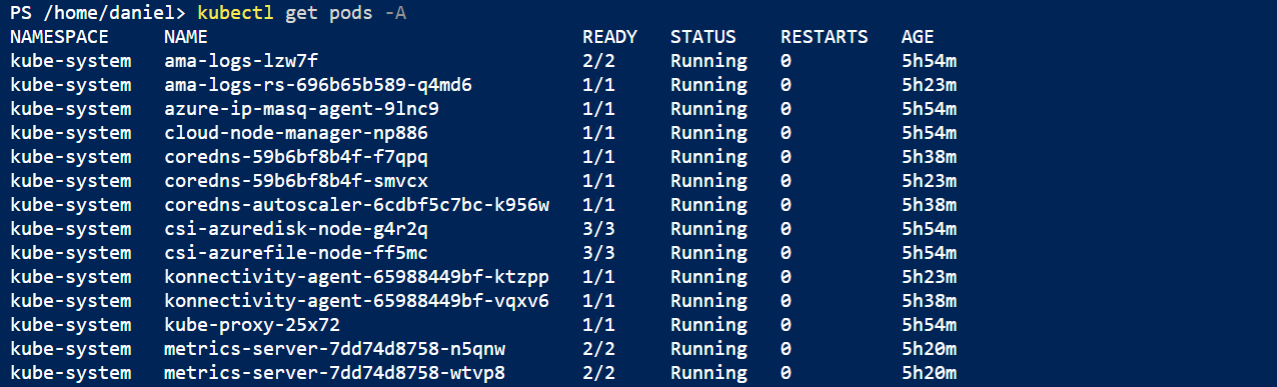


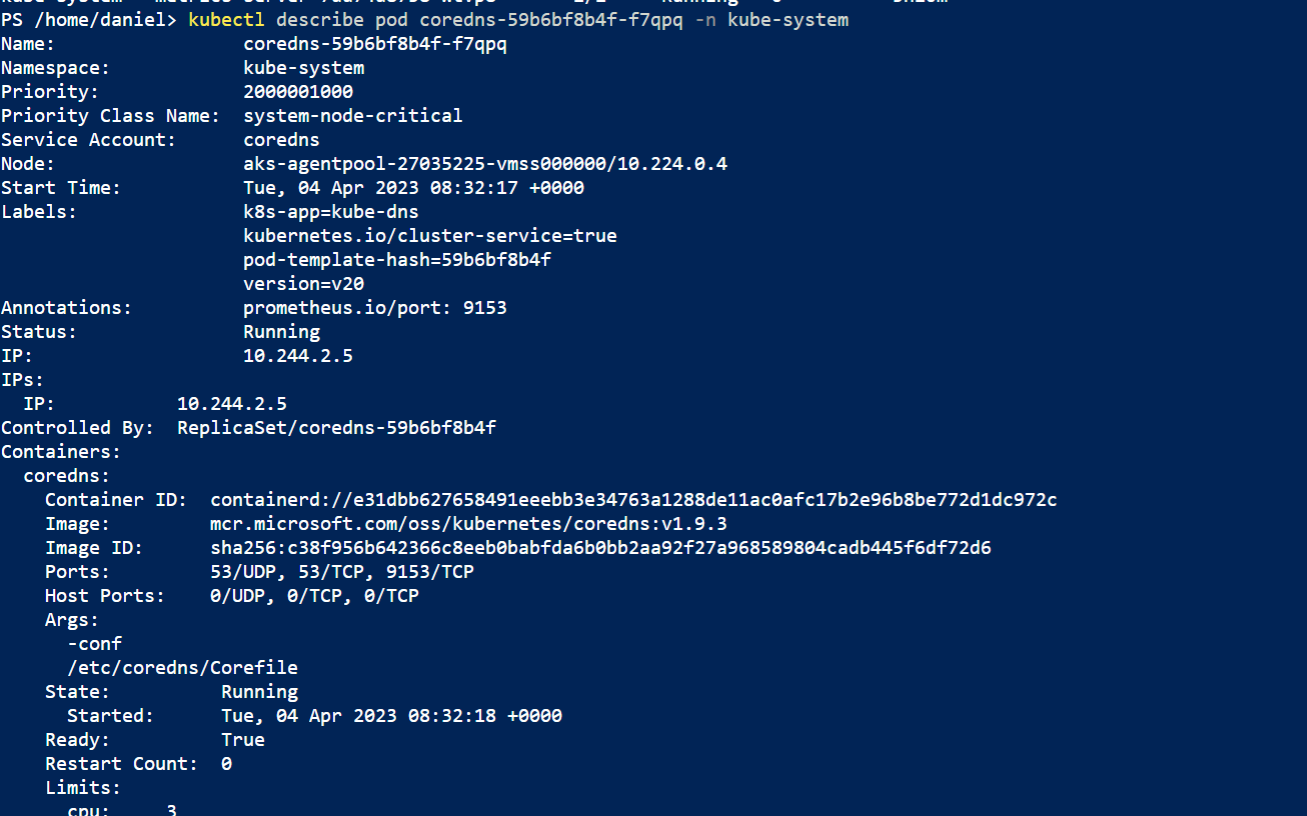
Try to find the same information but this time running kubectl describe pod nginx.



Delete your pod using kubectl delete pod nginx.

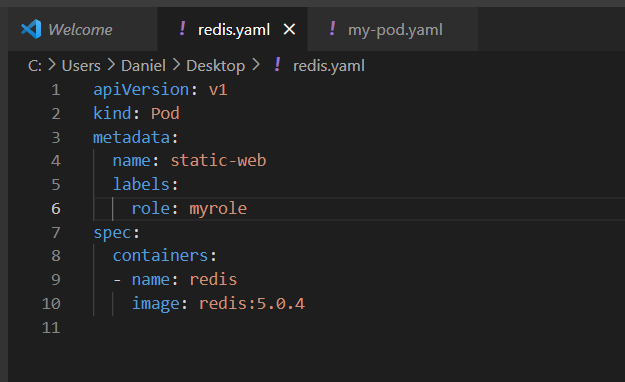
C:\Users\Daniel\Desktop\kubernetes\azure8.png

Let’s find the image used on one of the coredns pods under the kube-system namespace. Once again list all pods under all namespaces.

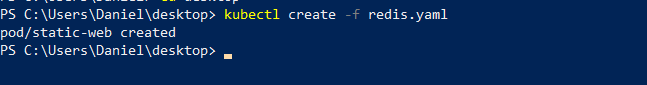
Note one of the coredns pods. Now run kubectl describe pod <coredns-name> -n kube-system. Replace th <coredns-name> place holder with noted name.

**Practice2: Working with pod manifest files**

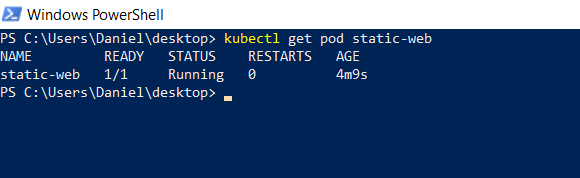
* Created redis.yaml file and corrected it.



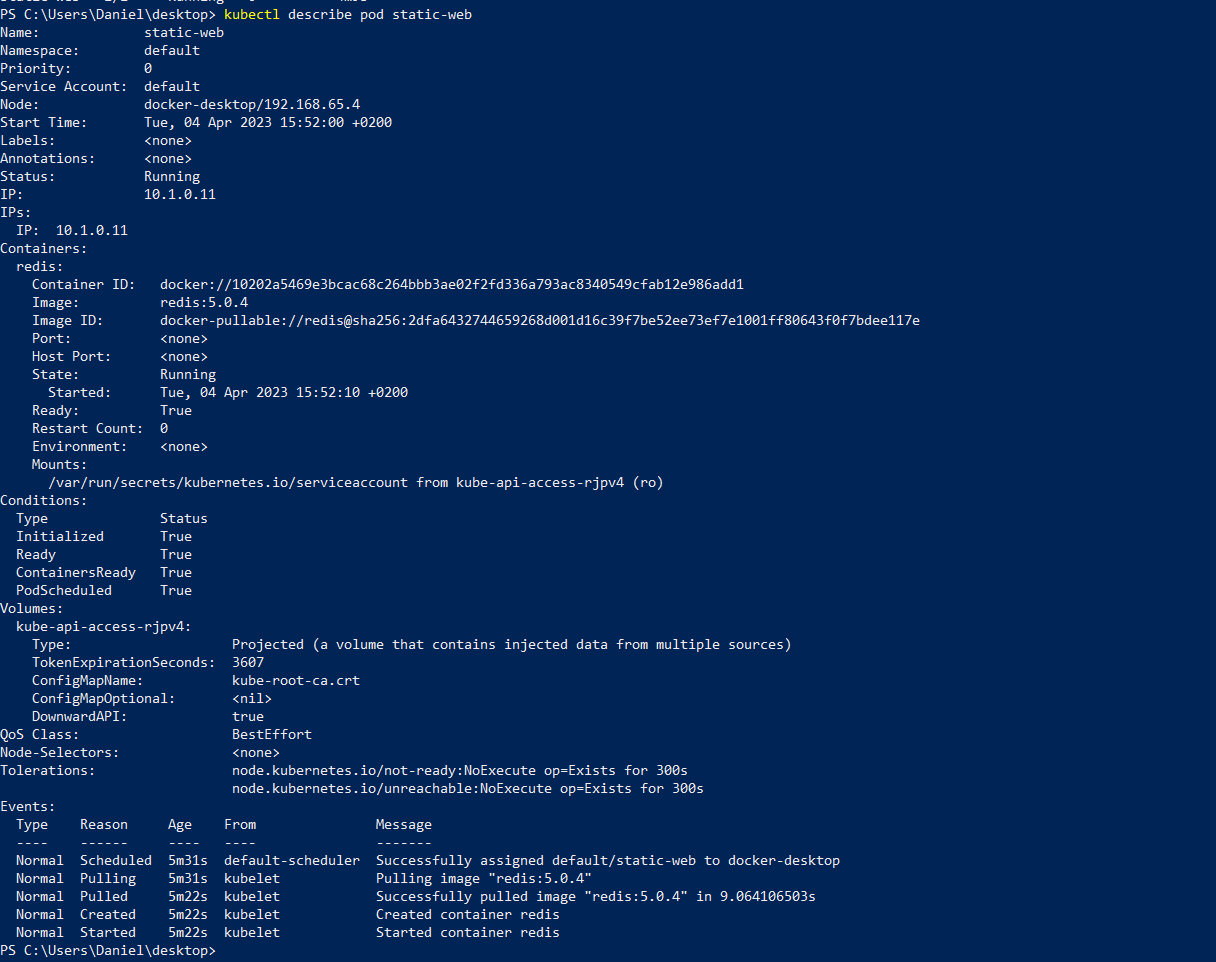
* Deployed redis.yaml file in powershell



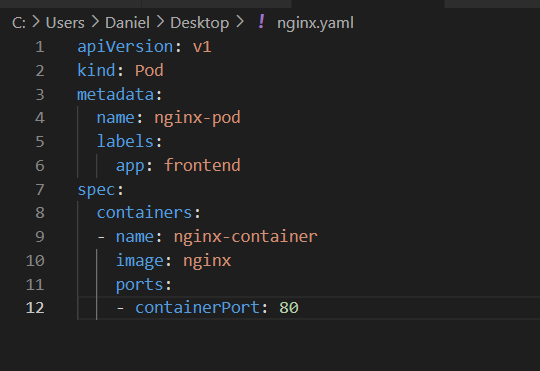
* Get status of this pod

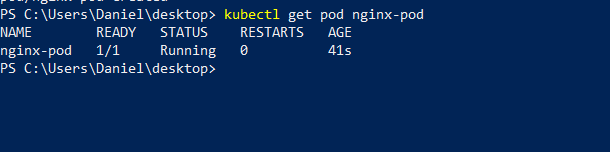


* Status of the pod described

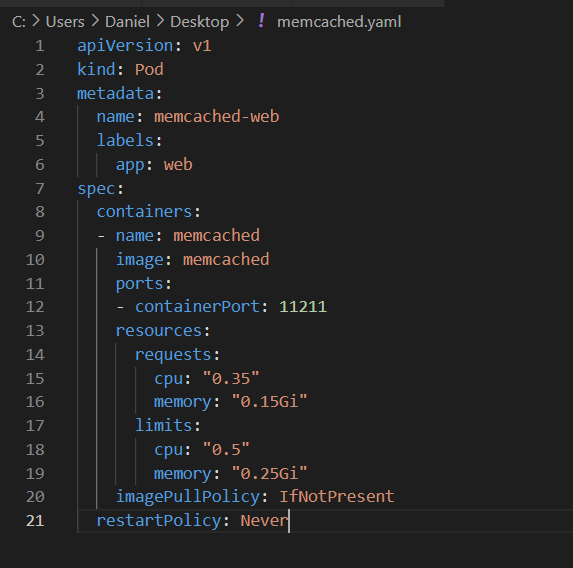


Your next task is to create and test nginx pod definition. Your definition should use the nginx official image, should use label named app with value frontend and should publish port 80. Make sure you complete this task because we will use this template in our next Labs. Your nginx pod should be running without any issues.

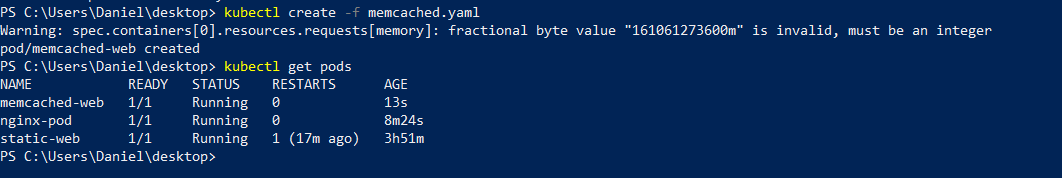




Final task of this practice will be to define pod definition with following details:

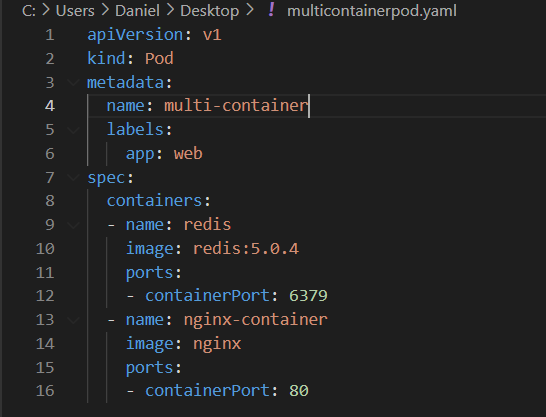


Don’t forget to try your pod definition.

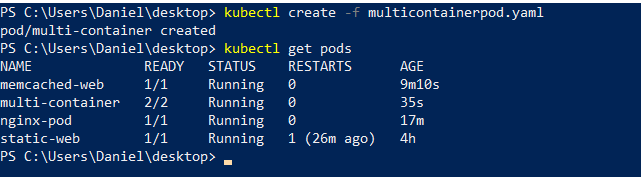


**Practice3: Multi-container pods**

Once finished you can try to create multi-container pod definition. Your multi-container pod should use redis and nginx containers with port 6379 and 80 published respectively. Label name should be app with value web.



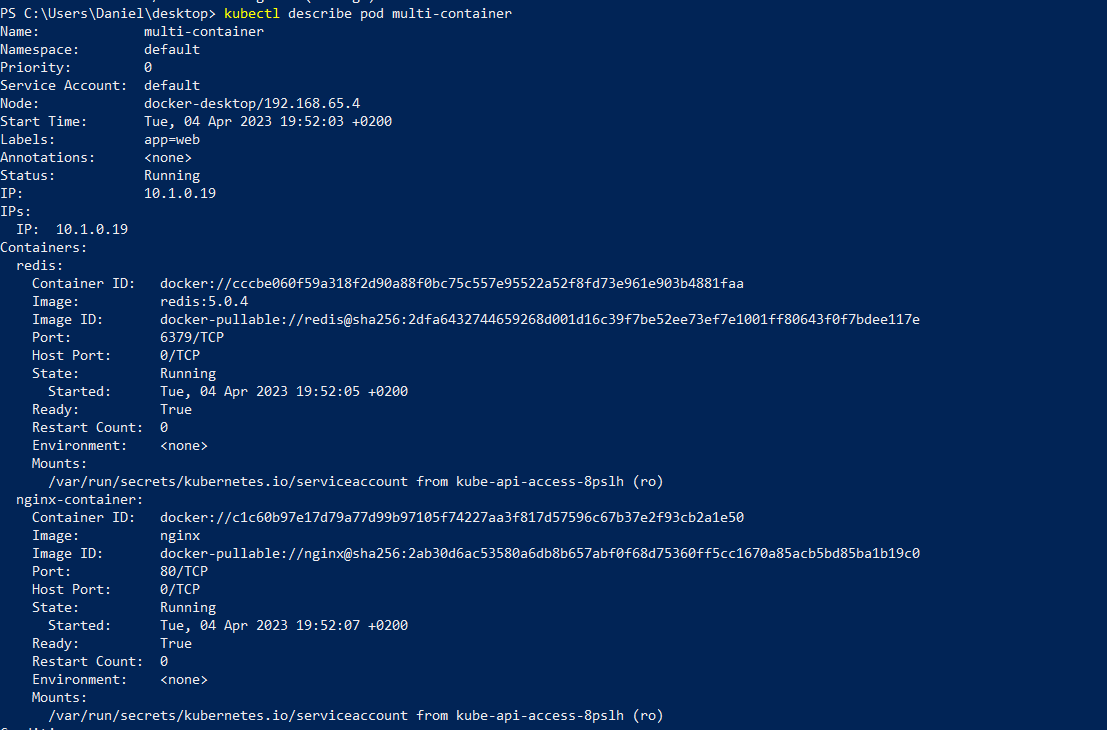
Deploy your multi-container pod. It should have running status. What is written under Ready column when you kubectl get the pods? Why your pod displays different values for ready?



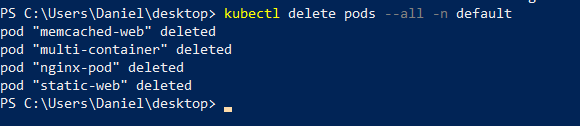
It written 2/2 because have two containers in the pod.

Kubectl describe you new pod, and locate the containers section. How many containers are listed?

-two containers are listed

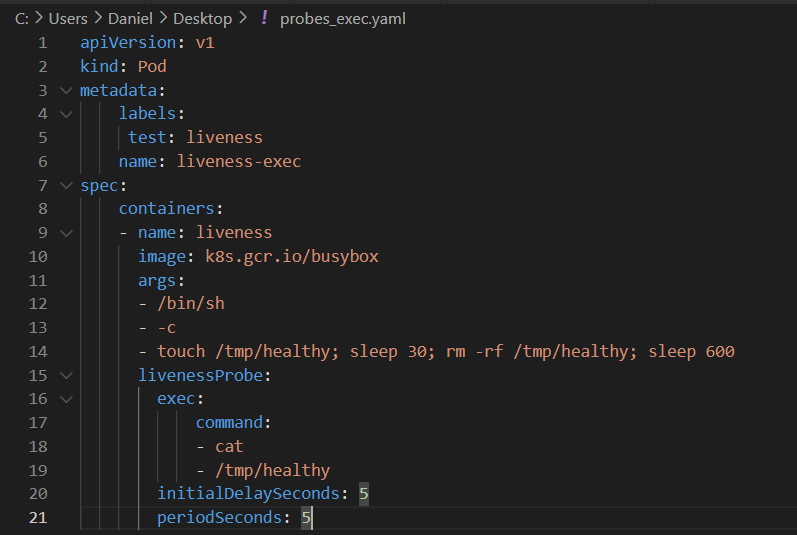


Delete all the pods under the default namespace.



**Practice4: Probes**

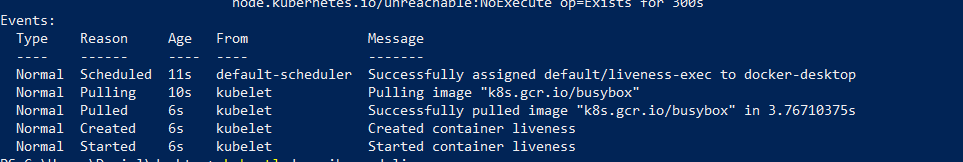
First we will create and test liveness probe with exec test. Create a file named probes\_exec.yaml with following content:



Run kubectl create –f probes\_exec.yaml.

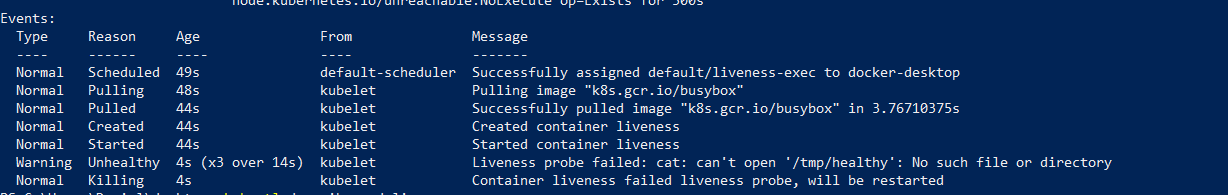
C:\Users\Daniel\Desktop\kubernetes\created probe_exec.png

Run kubectl describe pod liveness-exec immediately after you deploy the pod. The output should indicate that no liveness probes have failed yet.



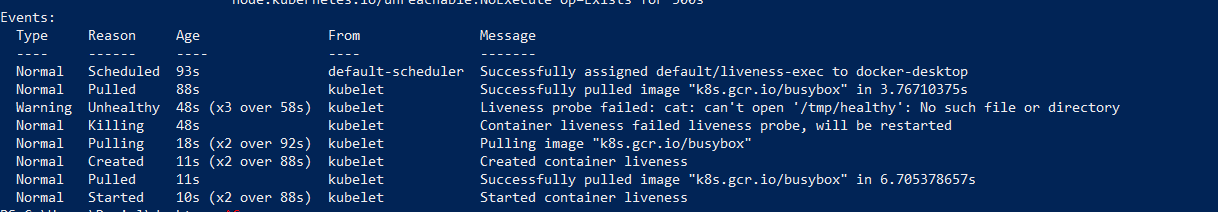
After 35 seconds, view the Pod events again. Run kubectl describe pod liveness-exec.

At the bottom of the output, there should be a messages indicating that the liveness probes have failed, and the containers have been killed and recreated.

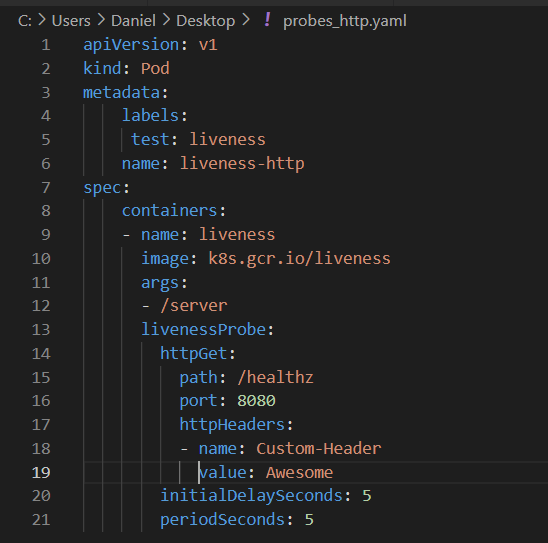


Wait another 30 seconds, and verify that the container has been restarted. Run kubectl get pod liveness-exec.

The output should show that RESTARTS has been incremented.



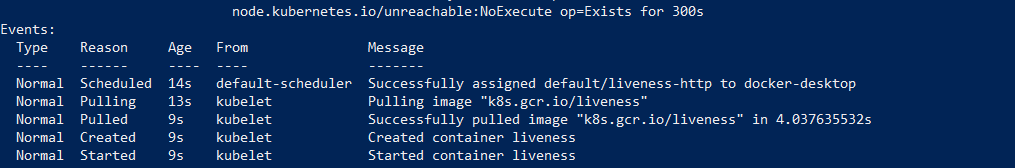
We will continue with HTTP probe. Create file named probes\_http.yaml with following content:



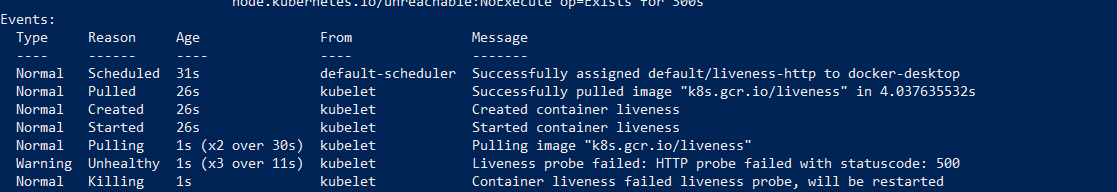
Run kubectl create –f probes\_http.yaml.

C:\Users\Daniel\Desktop\kubernetes\create probe_http.png

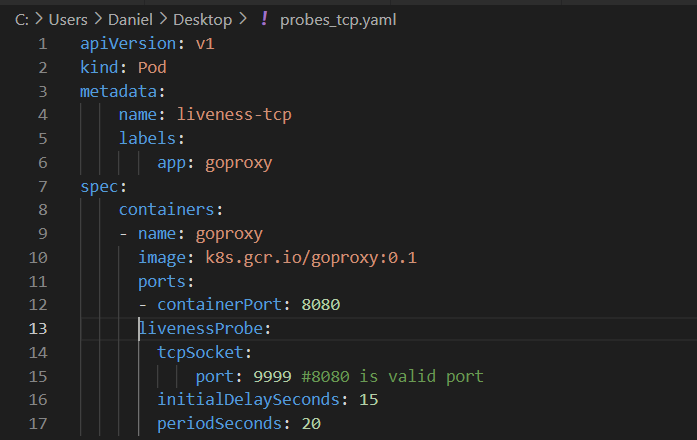
Immediately run (you only have 10 secs to run this command) kubectl describe pod liveness-http.



After 10 seconds, view Pod events to verify that liveness probes have failed and the container has been restarted. Run again kubectl describe pod liveness-http.



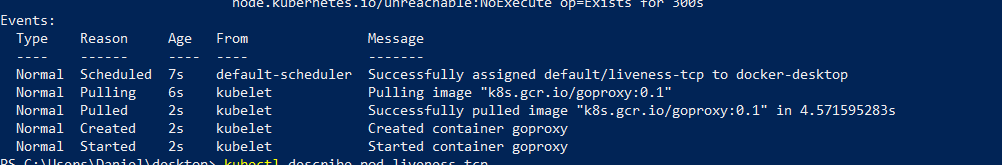
We continue with TCP probes. Create file named probes\_tcp.yaml with following content:



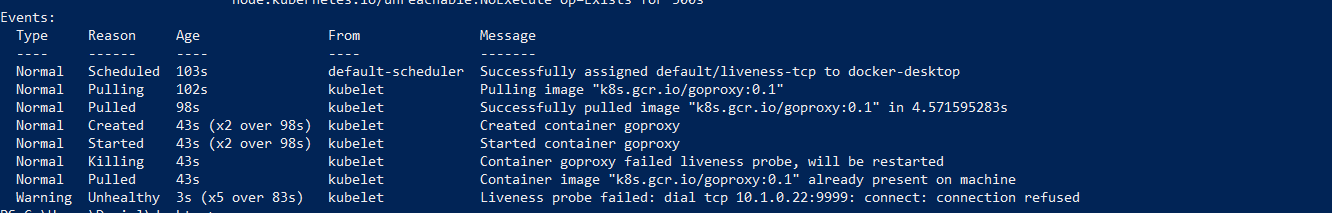
Run kubectl create –f probes\_tcp.yaml.

C:\Users\Daniel\Desktop\kubernetes\create probe_tcp.png

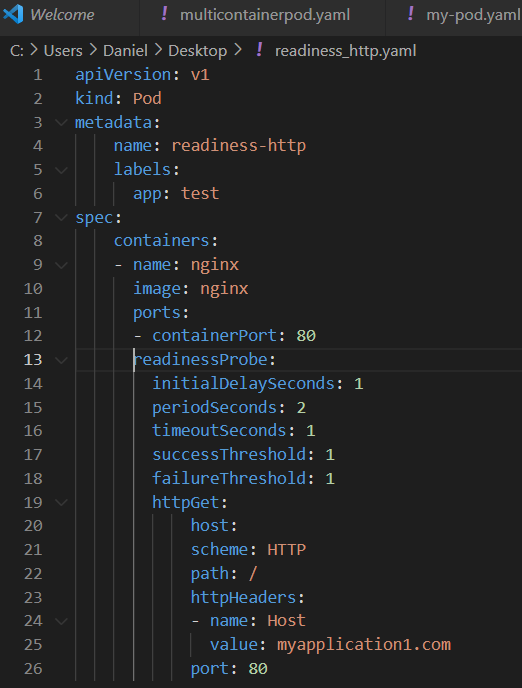
Immediately run (you only have 10 secs to run this command) kubectl describe pod liveness-tcp.



After 10 seconds, view Pod events to verify that liveness probes have failed and the container has been restarted. Run again kubectl describe pod liveness-tcp.



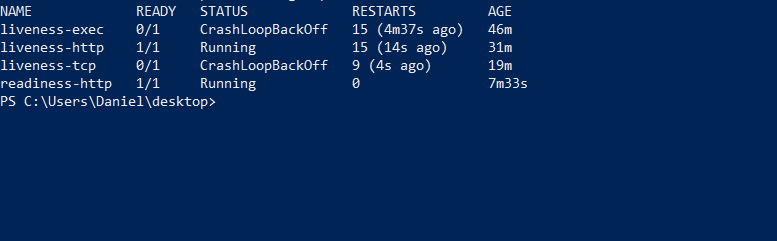
Create file named readiness\_http.yaml with following content:



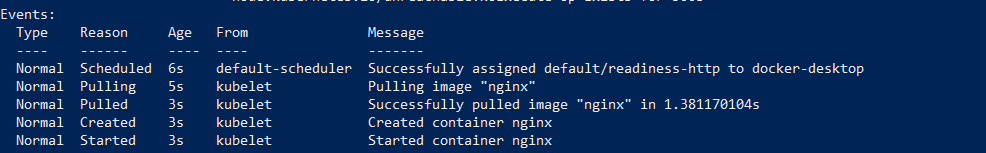
Run kubectl create –f readiness\_http.yaml.

C:\Users\Daniel\Desktop\kubernetes\create readiness_pod.png

Pods and their status and ready states will be displayed; our pod should be in running state.

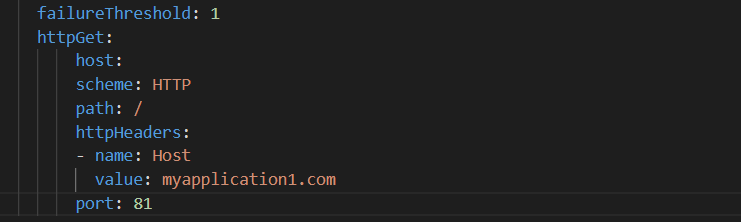


Run kubectl describe pod readiness-http. Examine the events for this pod. Everything should be OK.



Now delete the pod and edit the readiness\_http.yaml so that the port parameter has 81 value.

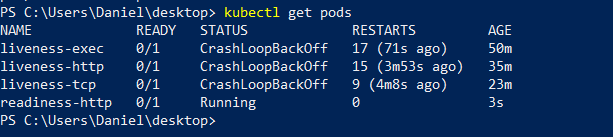
C:\Users\Daniel\Desktop\kubernetes\delete pod.png



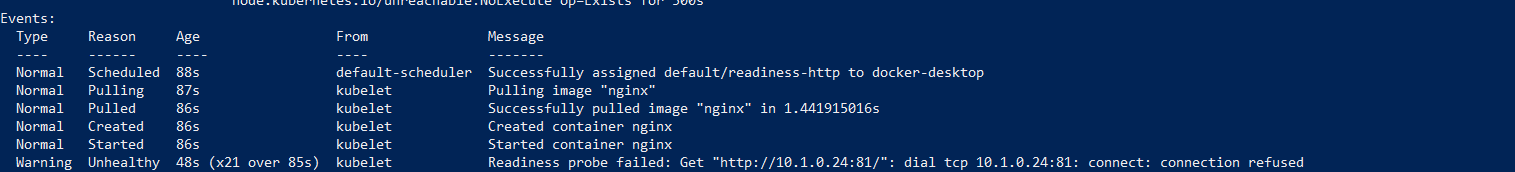
Run again kubectl create –f readiness\_http.yaml.

C:\Users\Daniel\Desktop\kubernetes\create readiness_pod.png

Run kubectl get pods –A to see the status of your pod. You should see that the pod is running but it is not in ready state.



Describe the pod. Run kubectl describe pod readiness-http.



From the events we can see that readiness probe failed due to the connection being refused therefore pod will not receive any traffic.

Delete all pods under the default namespace.

