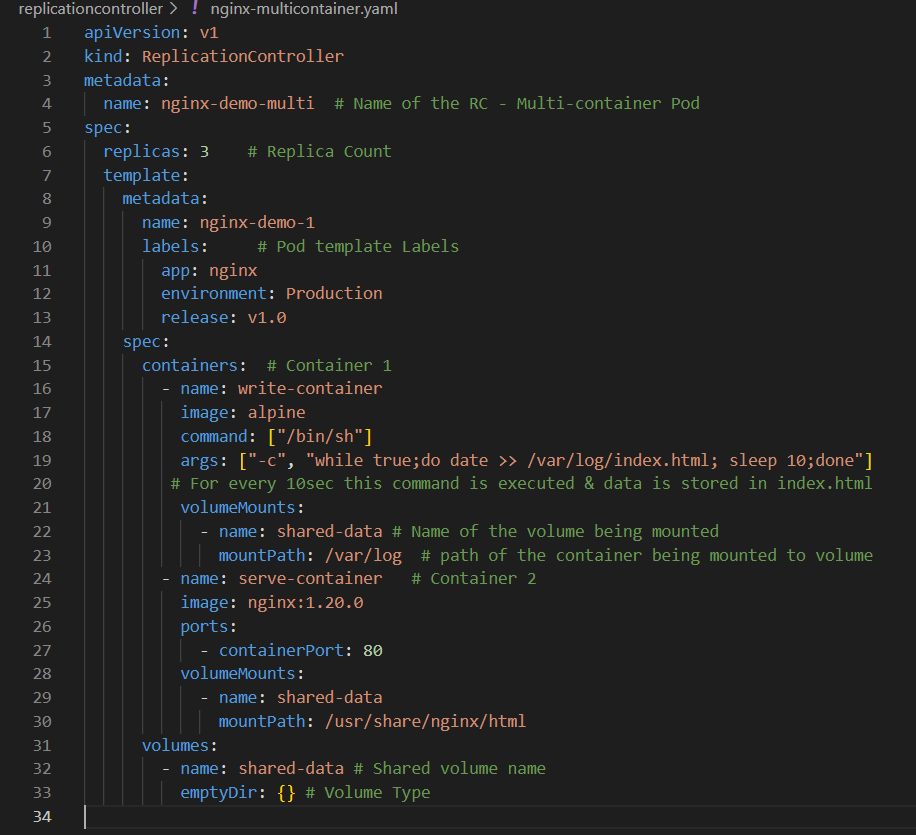
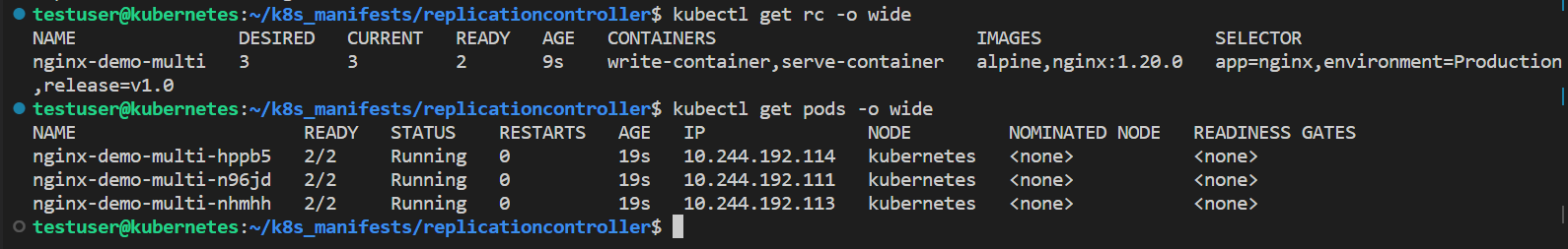
**RC using Cascade option**

**Cascade option:** Cascade flag is used fordeleting the Replication Controller without deleting the Pods available in it. Basically, cascade option is used to reapply the yaml file with new changes.

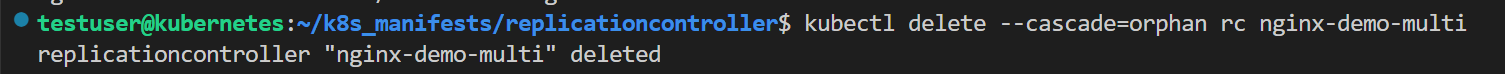
Create a Replicaton Controller with multi-container Pods. Use the command –   
**kubectl create -f nginx-multicontainer.yaml**



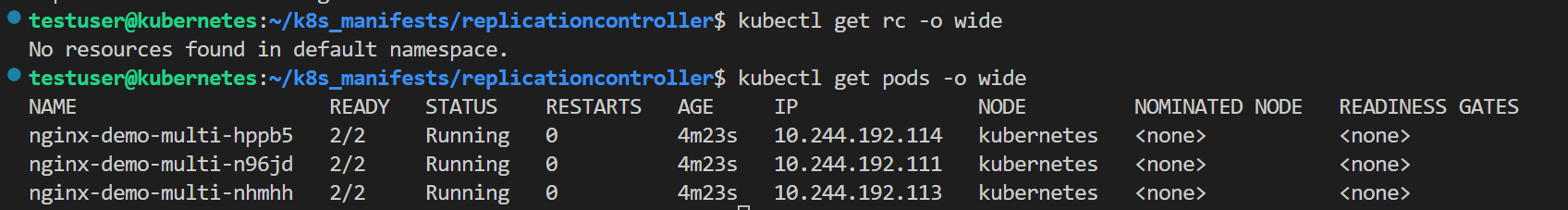
Then check the status of both RC and the Pods for running status.



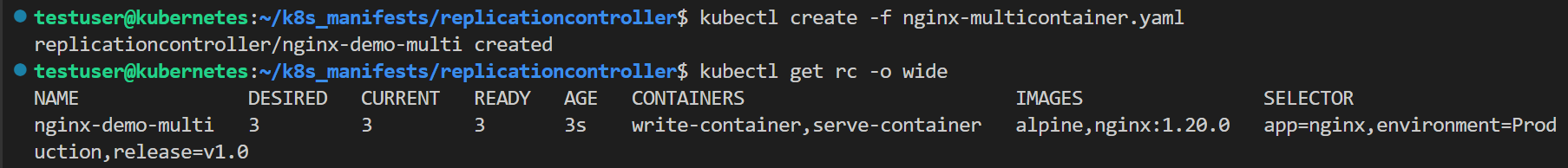
To delete the RC without deleting the Pods, we use cascade Flag. Use the command -  
**kubectl delete** --**cascade=orphan rc <rc\_name>**



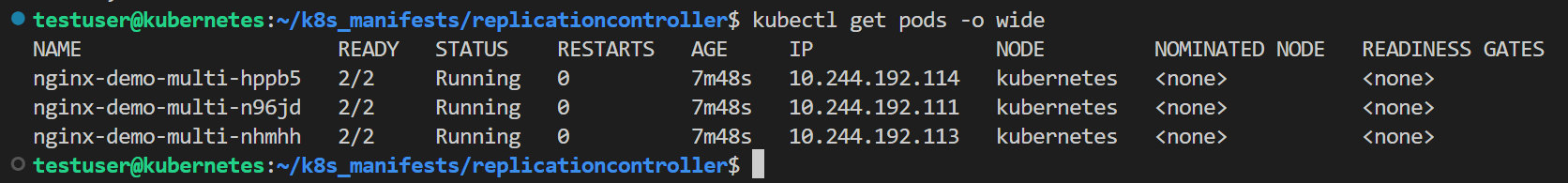
Now check the status of RC, use the command – **kubectl get rc -o wide,** which shows that the RC is deleted. But the Pods status if checked, all the three pods would be available & running.



If Replication controller is recreated with the same yaml file having replica count as 3, it would be expecting 3 pods which are already available. So, the RC will not create new Pods but will take the existing three pods under its control. Use the command to recreate the RC – **kubectl create -f nginx-multicontainer.yaml**

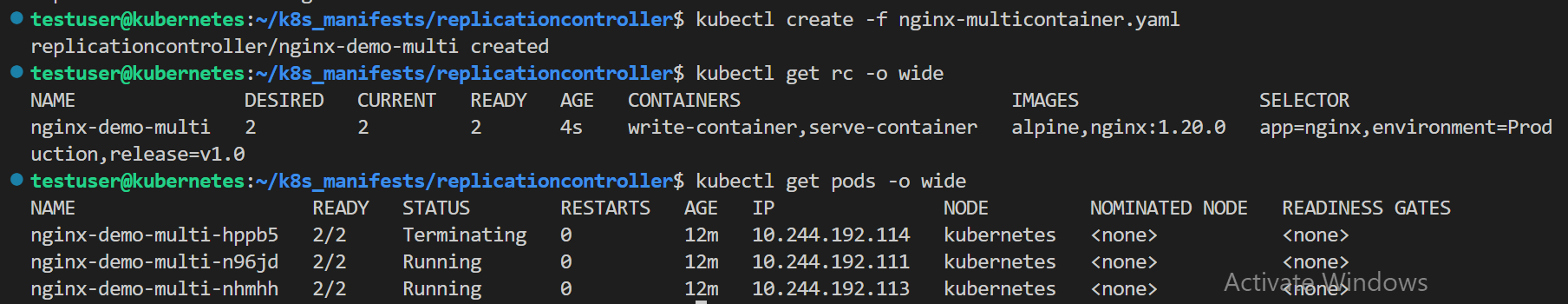


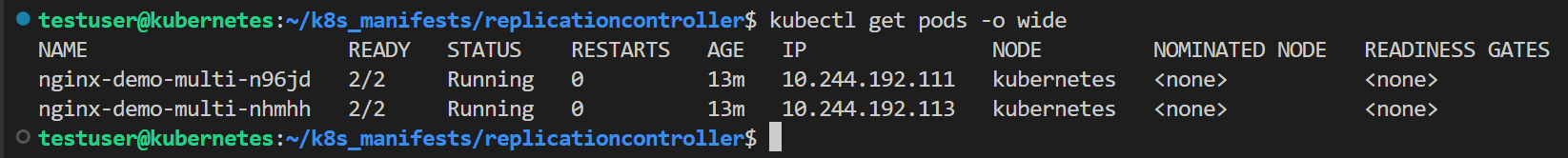
If checked the Pods age, it will not be 3s same as RC age, it would be more than that as Pods aren’t new created only RC is recreated.



**Note: Only if the replica count is changed in the yaml file then the RC tries to create new Pods or delete existing Pods to match the desired state.**

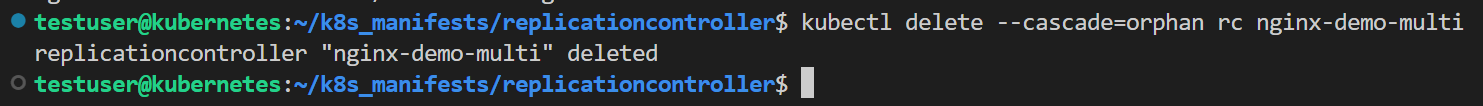
Make a change to the replica count (replicas =2) in the yaml file. Again create the RC using this yaml file and check the RC & Pods status. One out of the three Pods would be terminating as the desired state is changed to 2.

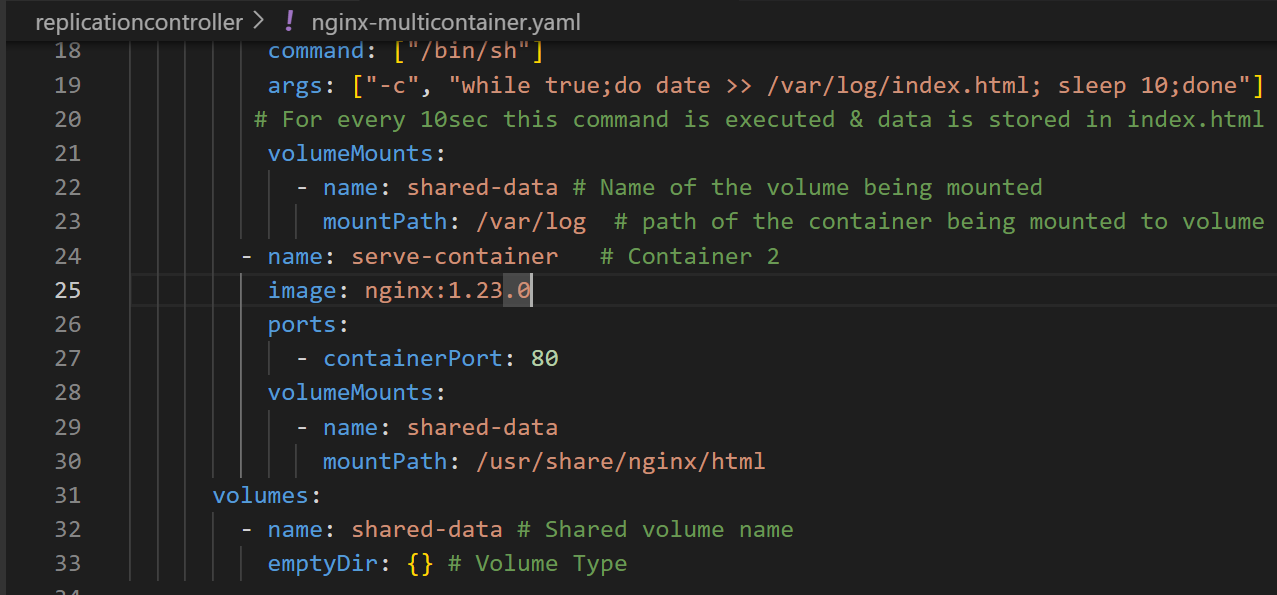


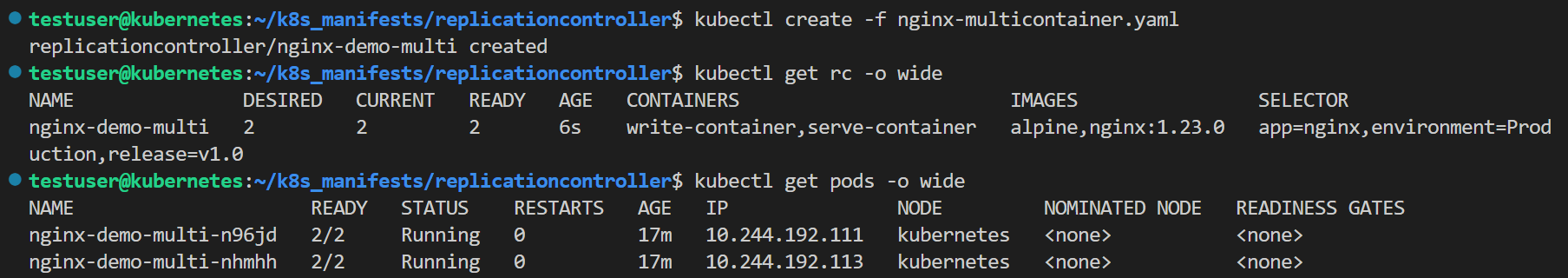


**Note : Use case of cascade option is to reapply the changes made in yaml file to RC and Pods.**

Assume a RC is deleted using cascade option and the Pods of RC are still available and running. Now, if the container image tag mentioned in the yaml file is changed and then a RC is created with the changed yaml file. Here, a new RC will be created but reuses the existing Pods.





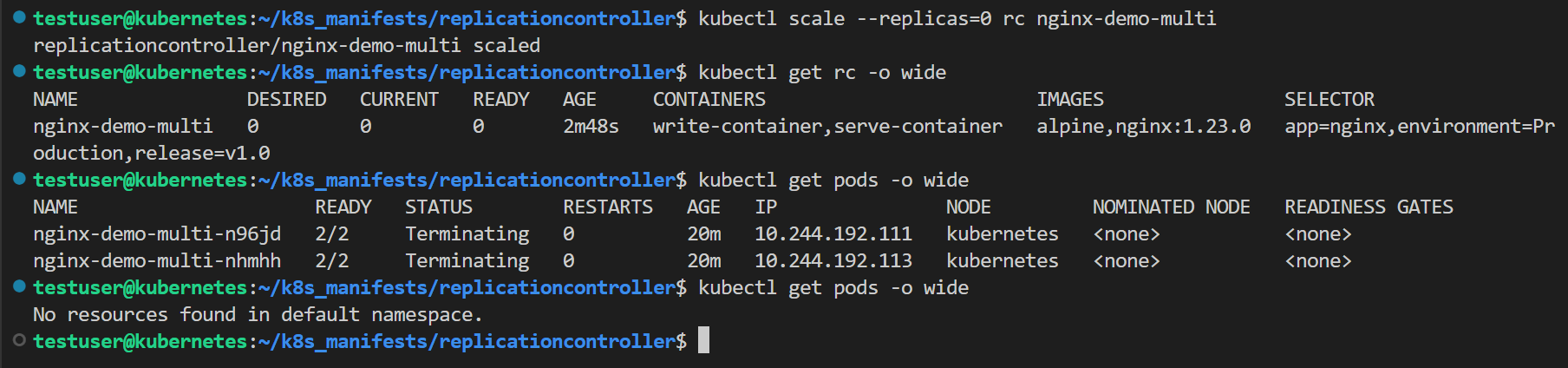


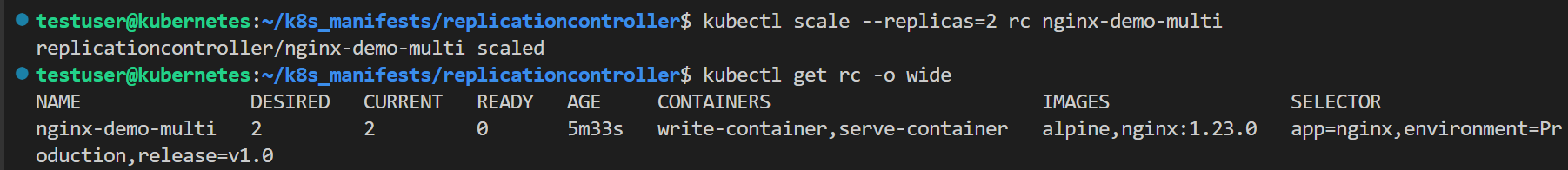
If the container image tag of the Pod is checked, use the command **curl -I <Pod IP>**, the Pod would still be having the old image tag not the new image tag mentioned in the yaml file because the RC didn’t recreate the Pods, it has just taken the existing Pods under its control.

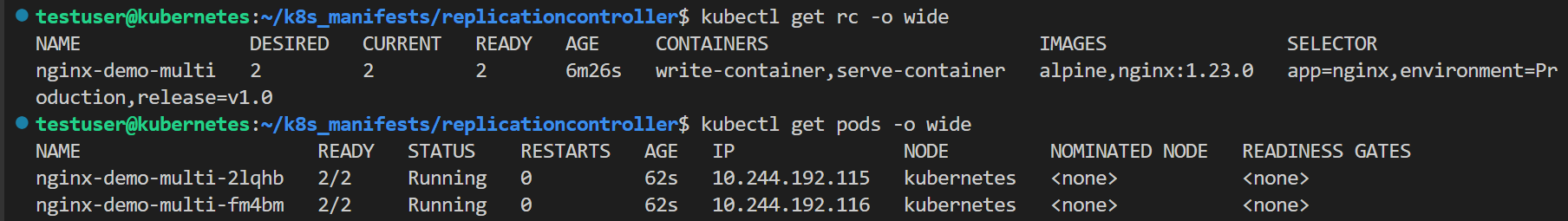


To make sure that the Pods of RC have same image tag as mentioned in the yaml file when the RC is cascaded and recreated, there are two option –

1. Scalein the Pods i.e., make the replica count 0, which will delete all the existing Pods. Then Scaleout the Pods i.e., increase the replica count to 2 or 3. All the Pods that are being created now, will have same image tag as mentioned in the yaml file.







Once the pods are deleted and newly created then check the image tag used by the containers in the Pod, use the command – **curl -I <Pod IP>.** This will clearly show that the Pods have same image tag as mentioned yaml file because here new Pods are created by RC from the information mentioned in the yaml file. Even the name of the Pods and IP address of it would be new as they are newly created.



1. Delete the existing Pods one after the other and immediately, the RC starts creating new Pods to match the desired state and these Pods will have same image tag as mentioned in the yaml file.

Once the pods are deleted (using any method) and newly created then check the image tag used by the containers in the Pod, use the command – **curl -I <Pod IP>.** This will clearly show that the Pods have same image tag as mentioned yaml file because here new Pods are created by RC from the information mentioned in the yaml file.