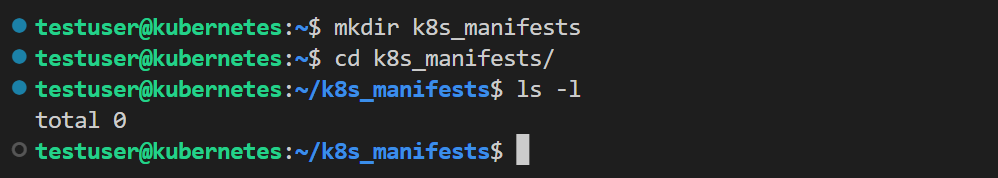
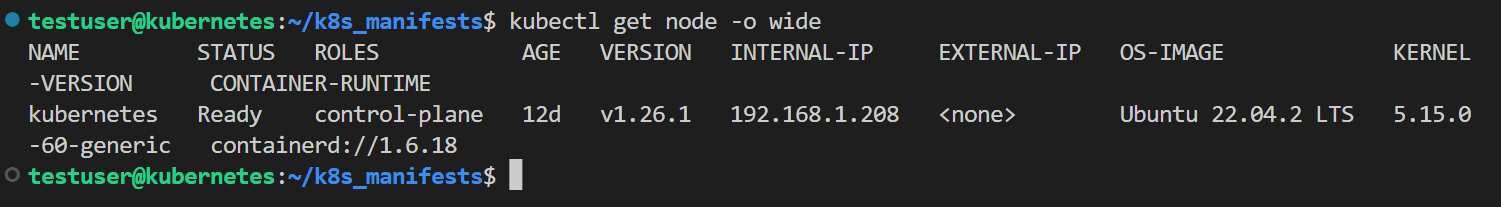
**RC using declarative approach – Single node**

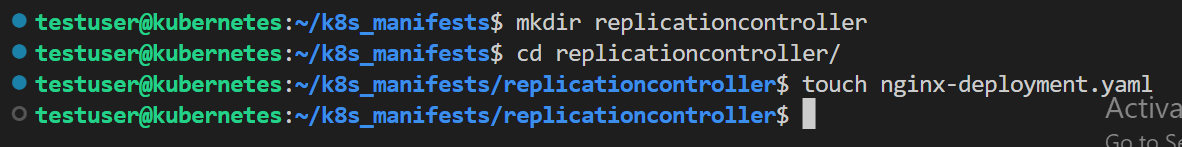
Open VS code and connect to the host machine. Open a terminal, create a directory named k8s\_manifests and cd into that directory.



Check the node status of the cluster, use the command – **kubectl get node -o wide**

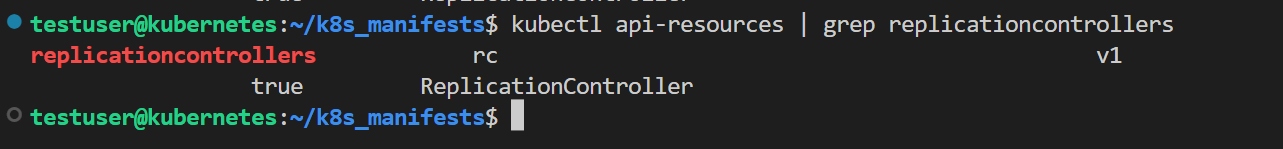


In the declarative approach, the yaml files are used to create the workload resources. In the k8s\_manifests folder create another folder for each workload resource and write all the related yaml files in the same folder to avoid confusion.

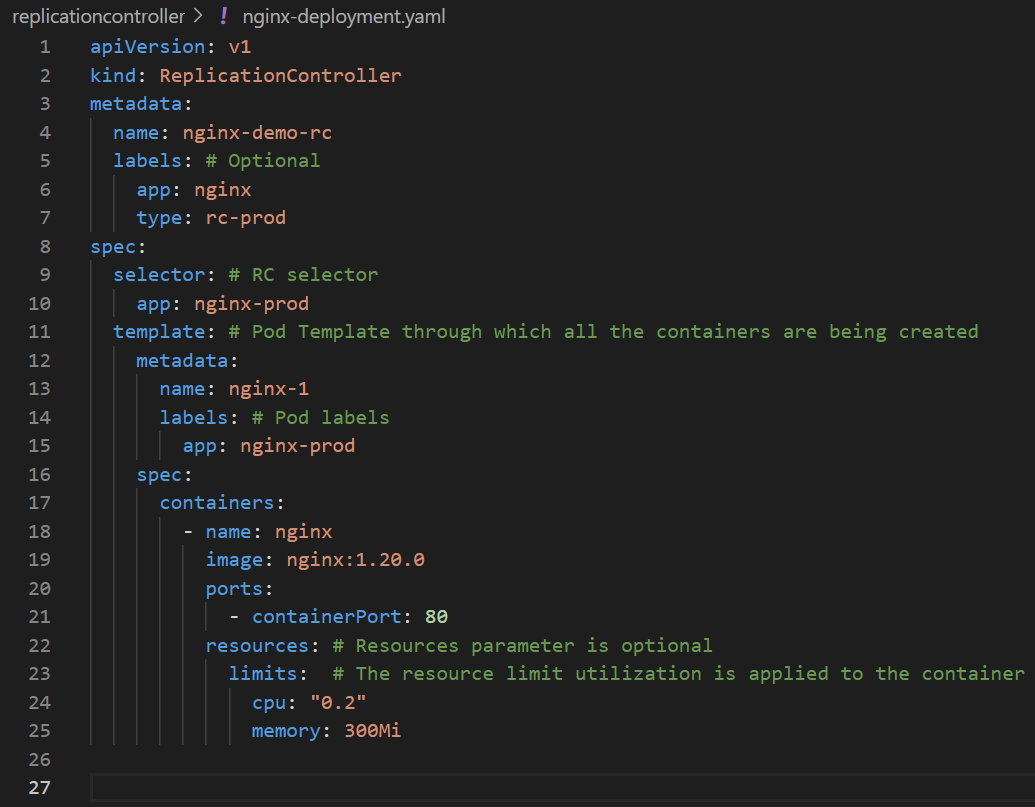


Note: To write into this yaml file in VScode, use ctrl + tilde (~) which will minimize the terminal and allow you to write into a specified yaml file.

To know what is the api-version to used for Replication Controllers use the command – **kubectl api-resources | grep replicationcontrollers** , in the terminal.



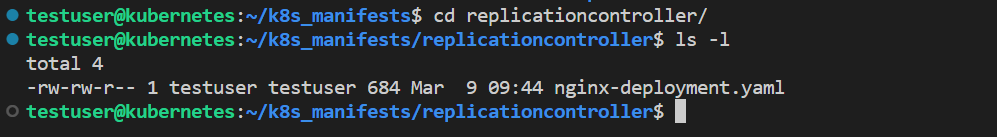
**Note: Always use ctrl + tilde (~) to shuffle between Terminal and yaml file VS code editor.**



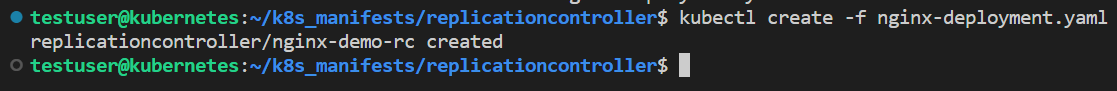
* Mention the api-version of the RC according to the api-resources file.
* The labels given to RC are just metadata and doesn’t effect the application. A replicas parameter is newly added to the spec of RC to define count of replications of Pods. If replicas parameter isn’t mentioned by default the value will be 1.
* The selector given under spec of the RC is used to match with the labels given to the Pods in the template metadata. Pod template is nothing but the template used to create all the pods in this RC. Make sure that the labels given in the Pod template metadata are same as RC Selector, then only the RC can identify and control the Pods.
* The Spec data given parallel to metadata of the template (pod template), is used to create all the container in each Pod.
* The default port for nginx is 80 and if needed we can also define the resource limit that can be used by the Pods i.e., if cpu = “0.2” is given that means on the node where the Pod is being created use 20% of 1corecpus and the memory to be utilized be 300Mi.

Save the nginx-deployment.yaml file and go back to terminal. cd into the replicationcontroller folder and check the list of files available in it. The yaml file should be available.

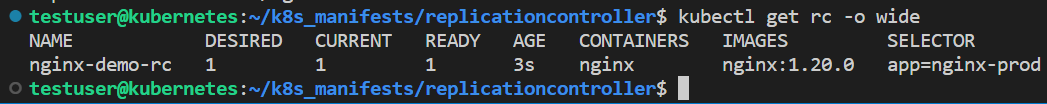
**Note: Make sure that you are in the right path of the folder where the yaml file is available**.



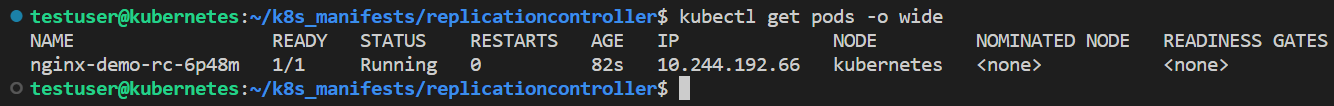
Create a RC with the yaml file written. Use the command – **kubectl create -f nginx-deployment**.



Once the RC is create successfully, check the status of RC first. Use the command – **kubectl get rc -o wide** . This displays the RC name, desired (number of Pods you want), Current number of Pods created, Ready status of RC, age (how much time has it been that the RC is created), image used by the containers, Selector of RC.

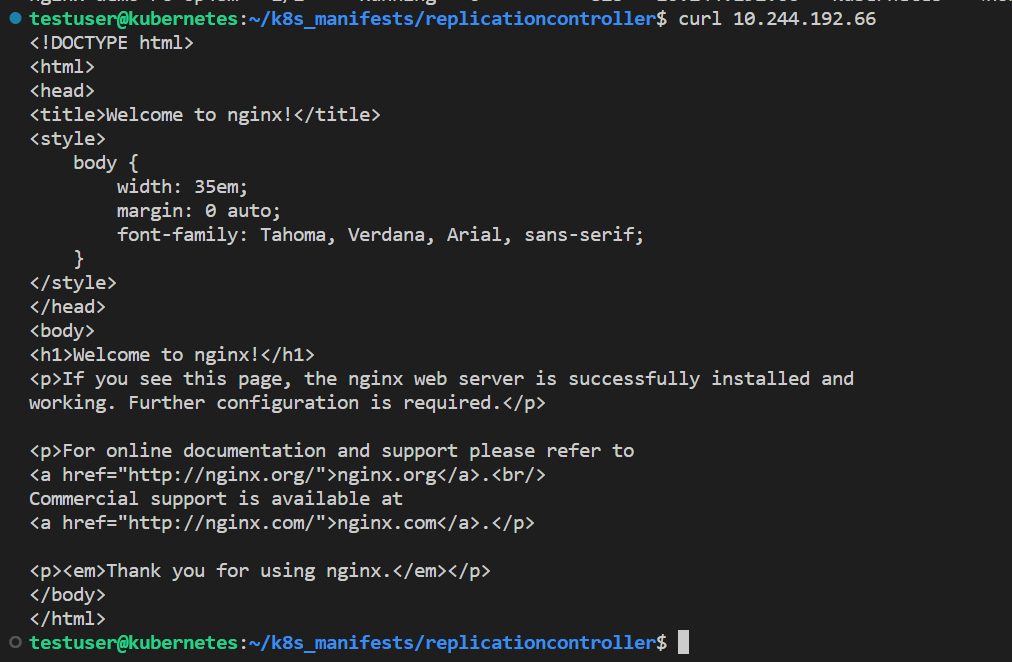


Next check the Pods’ status created from this RC. Use the command – **kuebctl get pods -o wide** . This displays the pod name, ready status, IP address. The name of the pod will be formed by RC name followed by a random number as a suffix.



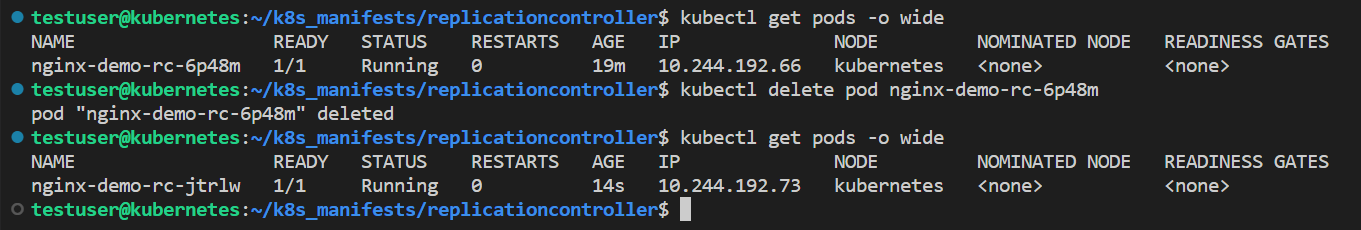
In the yaml file, the replicas parameter isn’t mentioned, so the RC has taken the value by default as 1 that why the desired is 1.

Check the application running inside the container of the POD using pod IP. Use the command – **curl <IP of the Pod>**



**Note: Replication Controller doesn’t have any IP address. RC is just a logical layer to implement the replication of Pods concept.**

If you try to delete the Pods, the RC will recreate the Pods because the desired state is 1 and when the Pod is deleted the actual state becomes 0. As kube control manager always checks for the match of desired state with actual state, immediately a new pod is created by RC with the same Pod template.



The Pods replication is not about data replication, it is all about same metadata is used for all Pods creation. Each Pod has its own data.

To delete the Pods delete RC, not the Pod because the Pods will be recreated by the repolication concept.

