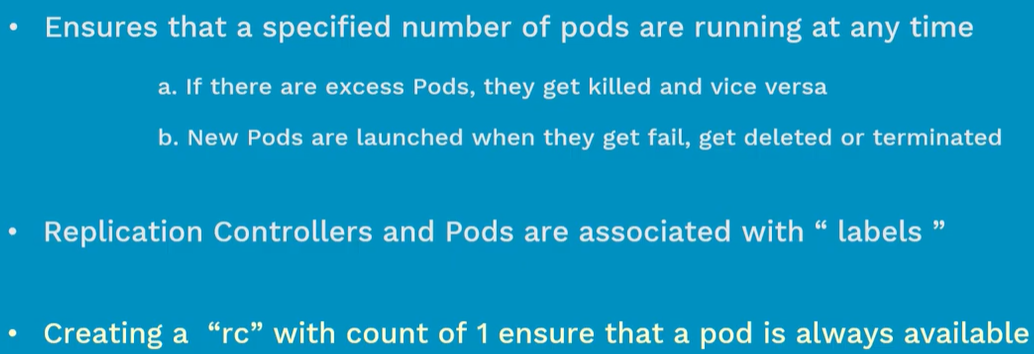
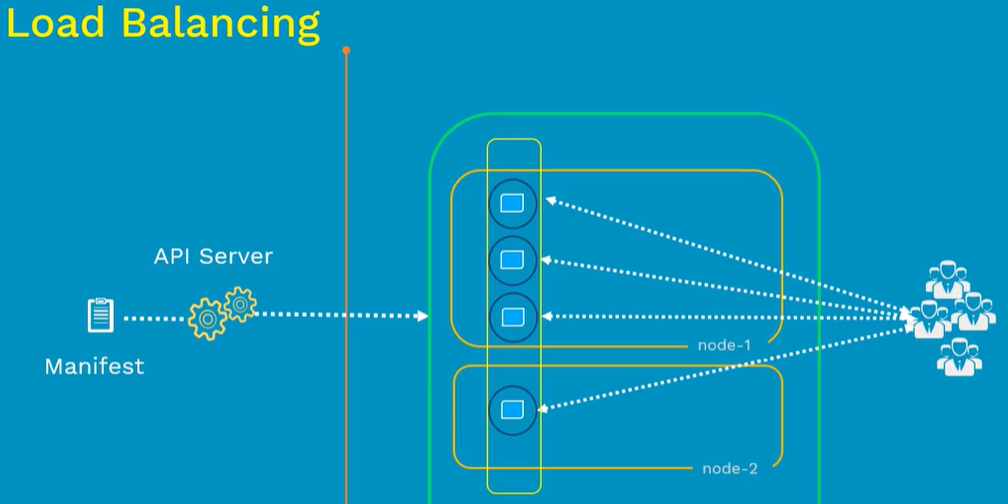
**Introduction:**

* If the pods got killed or failed, kubernetes cannot do anything with those pods. So, we need replication controllers to maintain the desired state
* Replication controller will always try to maintain the desired state
* If there are extra pods running than desired state, then the replication controller will terminate those extra ones
* If there are less pods running than desired state, then it will create new pods to make the desired state
* If u always want single pods to be running, then it’s better to mention replication controller as 1. So that it will always maintain the desired state
* Replication controller manages the pods associated with labels

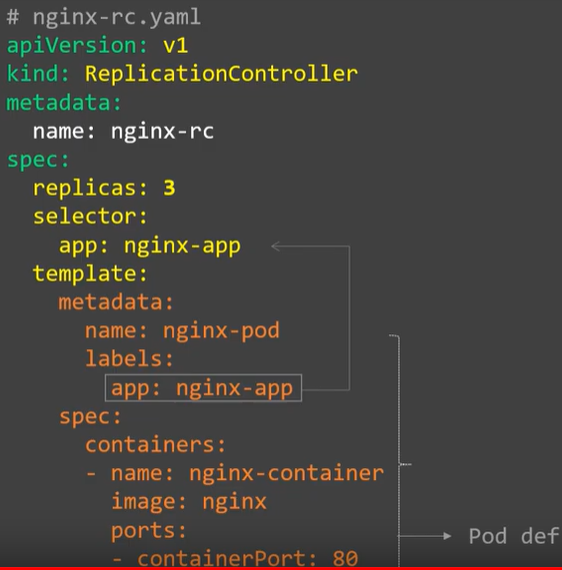


* It maintains the high availability always
* If the node1 gets failed and then the controller manager in master detects these changes and moves the pods from node1 to any healthier node within 2 to 3 minutes
* Replication controller is old and replicasets is the new now
* Replication controller selects equality base selectors and whereas replicasets selects set base selectors



* Let’s say we have 3 pods schedules in node1 and the resources of node1 are not sufficient enough
* So, if we add another node to the cluster and increase the replication controller to 4. Then the 4th POD will be created in node and load balancing will be done to the 4th POD as well

**Replication controller yaml file:**

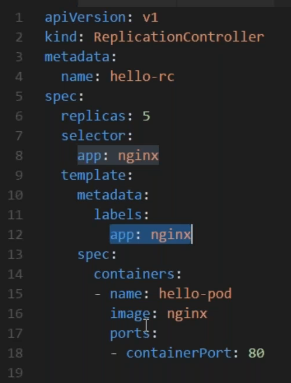


We can also define the lables under metadata section also.

Under “spec” we need to define the replicas number and then we can go for the template.

Now, under templates, we do the same configuration as POD.

After this, we need to add selector. So that the replication controller will select the POD based on labels.



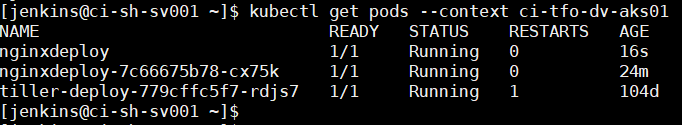
* Here we have 4 top level sections which are “apiversion” “kind” “metadata” and “spec”

we can also have multiple labels as below.

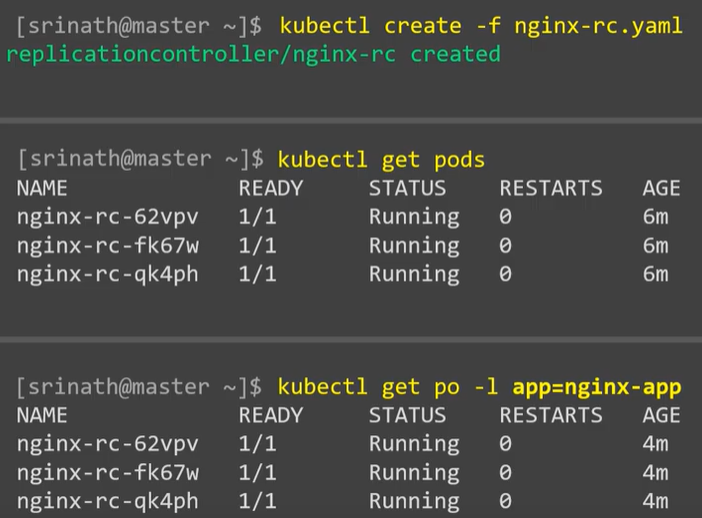


**Note:**

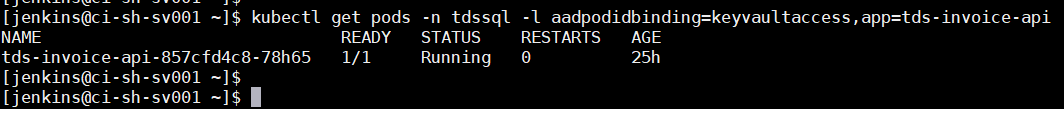
* **If we have two labels for a pod. The same can be attached to RC or RS by using anyone label of pod. And we can also filter the pods using anyone label. No need to use both all the time**
* **We can’t assign different name for RC and different for pod. The name option which we use under “template” is of no use. Be default the pod gets created with the deployment name by adding some extra autogenerated name as below.**
* **And if a pod is already running. We can’t assign a RC to that pod even if we give the same name of the pod in RC spec. pod will be running separately and RC or deployment object pod will be running separately. This is same for RC, RS and Deployment objects.**



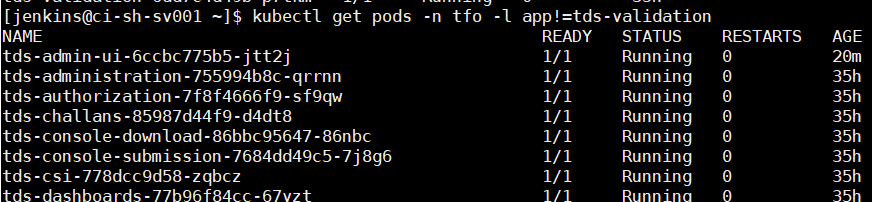
**Commands:**



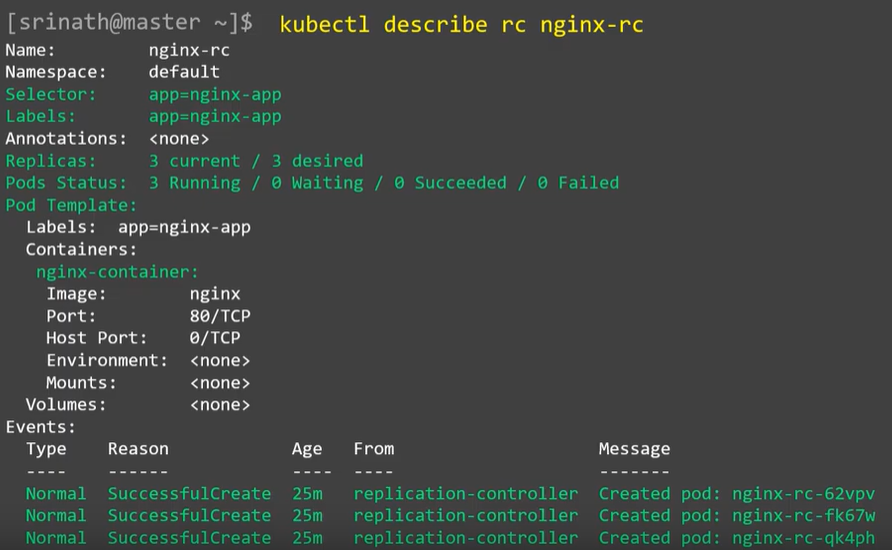
* We can use -l and the label name to filter the pods as above
* We can use below command to filter with multiple labels.



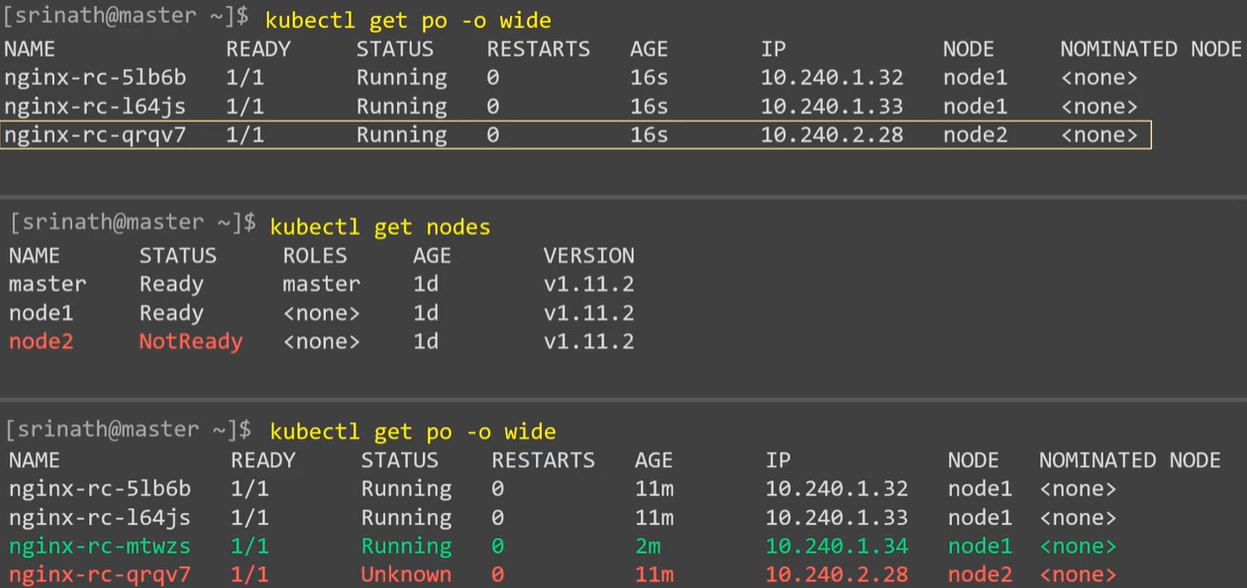
* = & == represents equality but != represents inequality. We can use that in command as below.



* As above, it displays the pods which are containing “app” as key but not “tds-validation” as value.
* This same can also be use in selectors as well.

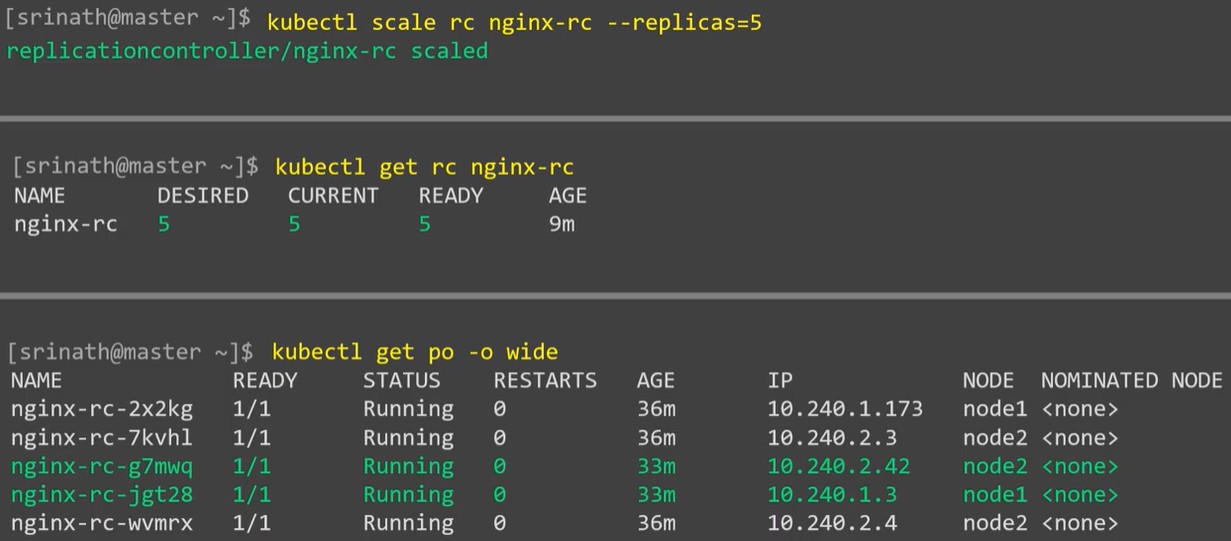


**When node fails:**



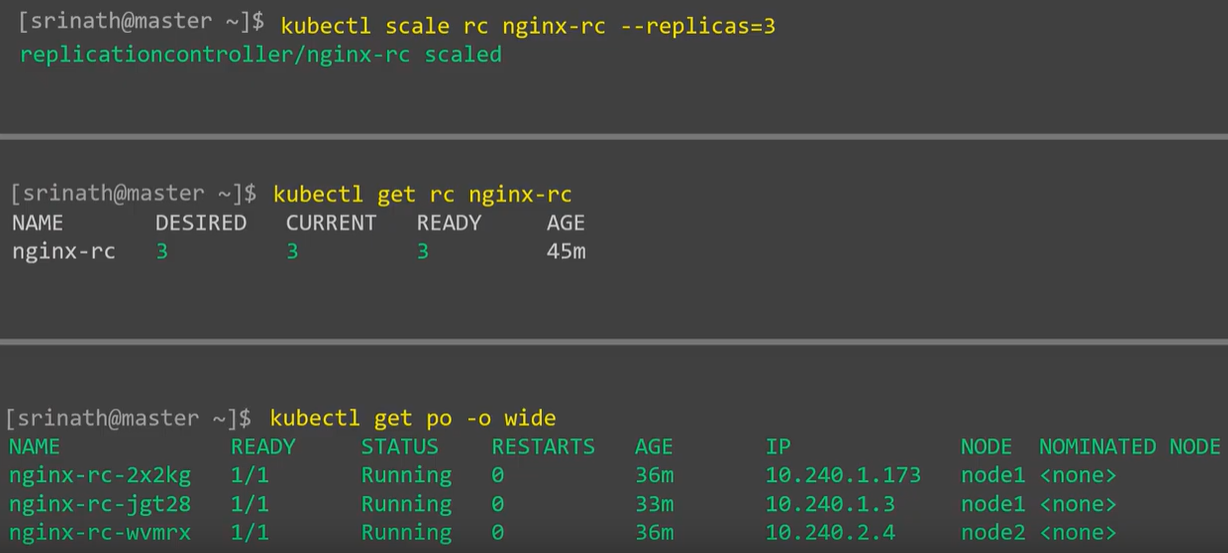
* As above, if node2 fails then the master will move the pods to the healthy node

**Scaling up:**



* We can use the “scale” down command to scale up the replication controller as above

**Scaling down:**



**Deleting RC:**

* If we delete RC, then all the pods under that will also be deleted

