Controllers are the brain behind Kubernetes. They are the processes that monitor Kubernetes objects and respond accordingly. We are going to discuss about Replication Controller in this lecture. What is a Replica and why do we need a Replication Controller?

Let’s go back to our first scenario where we had a single pod running our application. If for some reason our application crashes and the pod fail, users will not be able to access our application. To prevent users from losing access to our application, we would like to have more than one instance or pod running at the same time. That way, if one fails, we still have our application running on the other one.

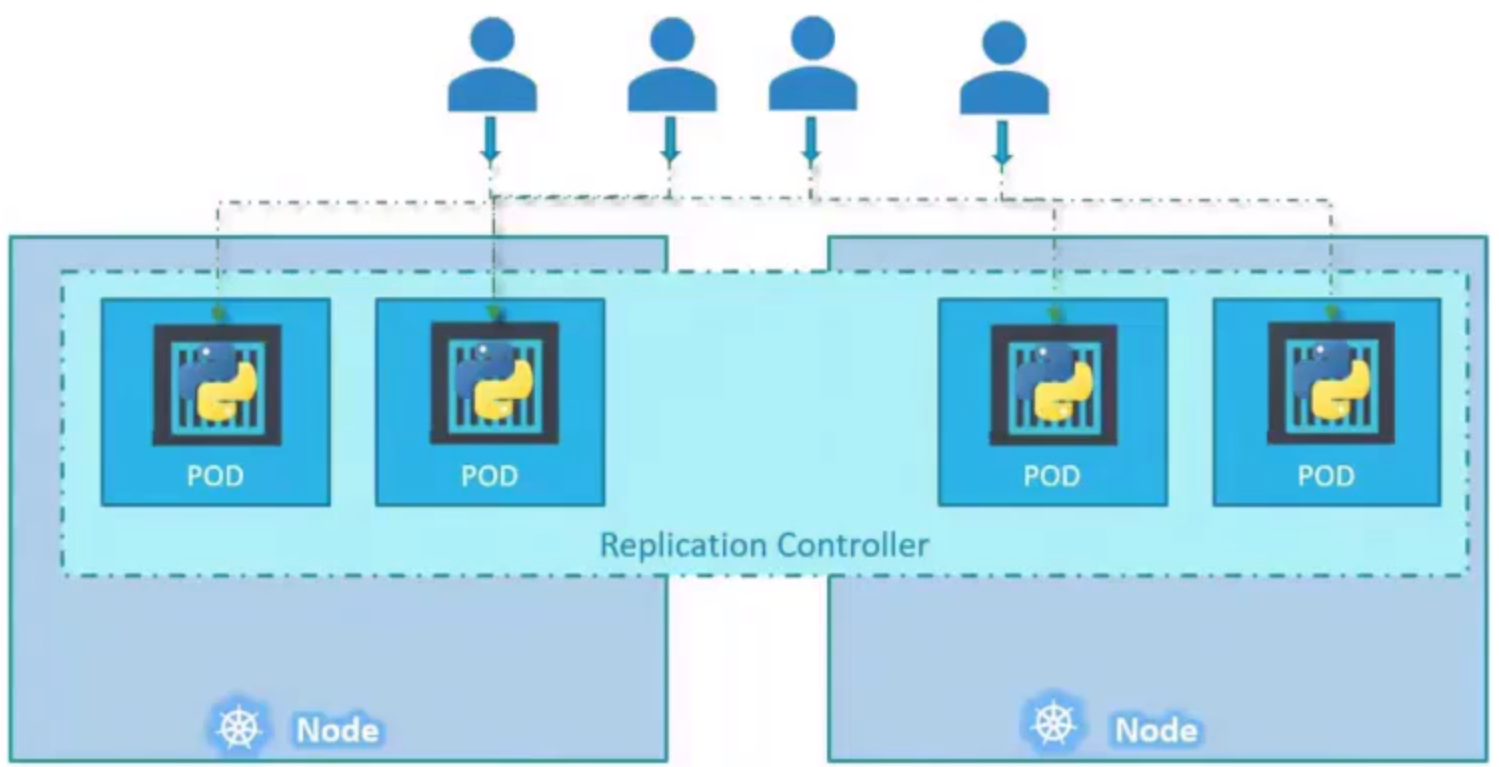
The Replication Controller helps us run multiple instances of a single pod in the Kubernetes cluster, thus providing high availability. So, does that mean you can’t use a replication controller if you plan to have a single pod?

Even if you have a single pod, the Replication Controller can help by automatically bringing up a new pod when the existing one fails. Thus, the Replication Controller ensures that the specified number of pods are running at all times, even if it’s just 1 or 100.



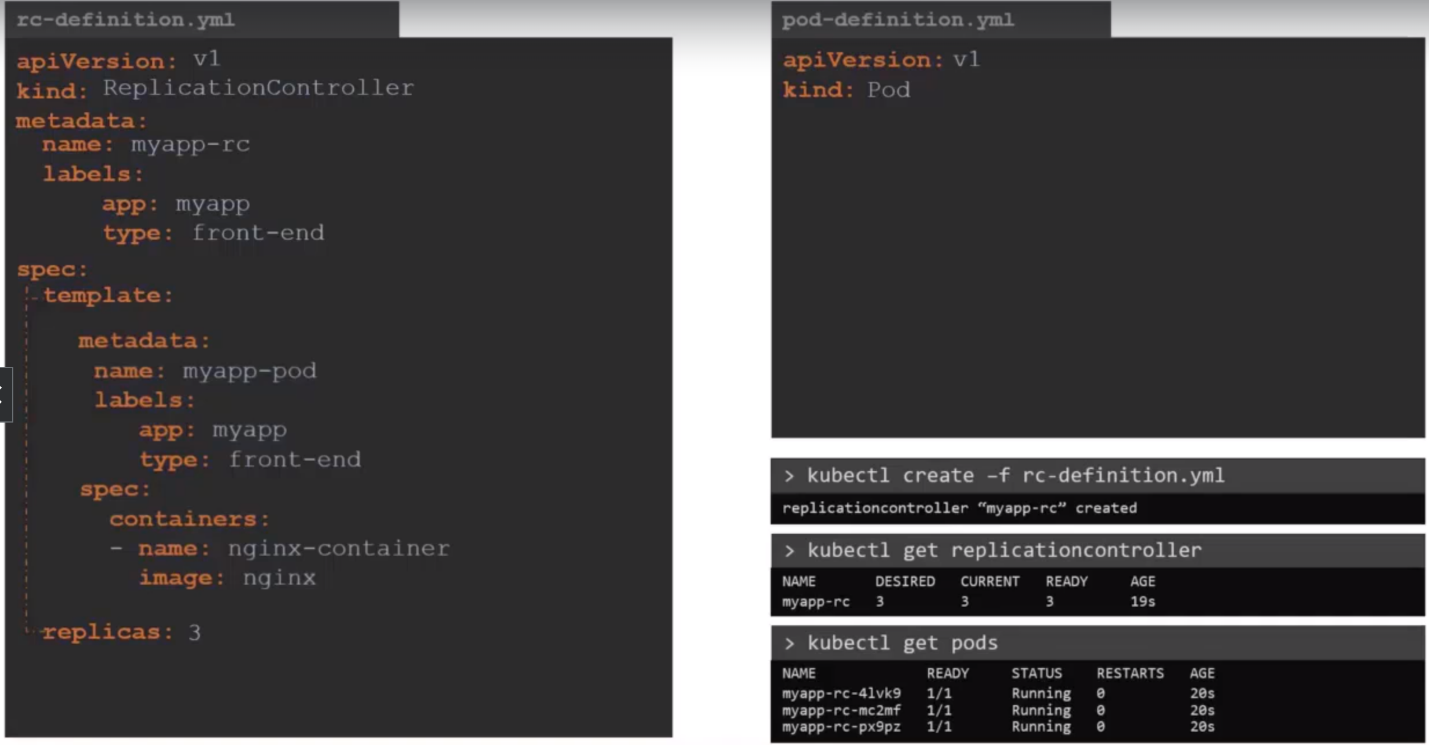
Another reason we need Replication Controller is to create multiple pods to share the load across them. For example, in a simple scenario, we have a single pod serving a set of users, when the number of users increase, we deploy additional pod to balance the load across the two pods. If the demand further increases and if we were to run out of resources on the first node, we could deploy additional pods across the other nodes in the cluster.

The Replication Controller spans across multiple nodes in the cluster. It helps us balance the load across multiple pods on different nodes as well as scale our application when the demand increases. It’s important to note that there are two similar terms Replication Controller and Replica Set, both have the same purpose but they are not the same. Replication Controller is the older technology that is being replaced by Replica Set. Replica Set is the new recommended way to set up replication.

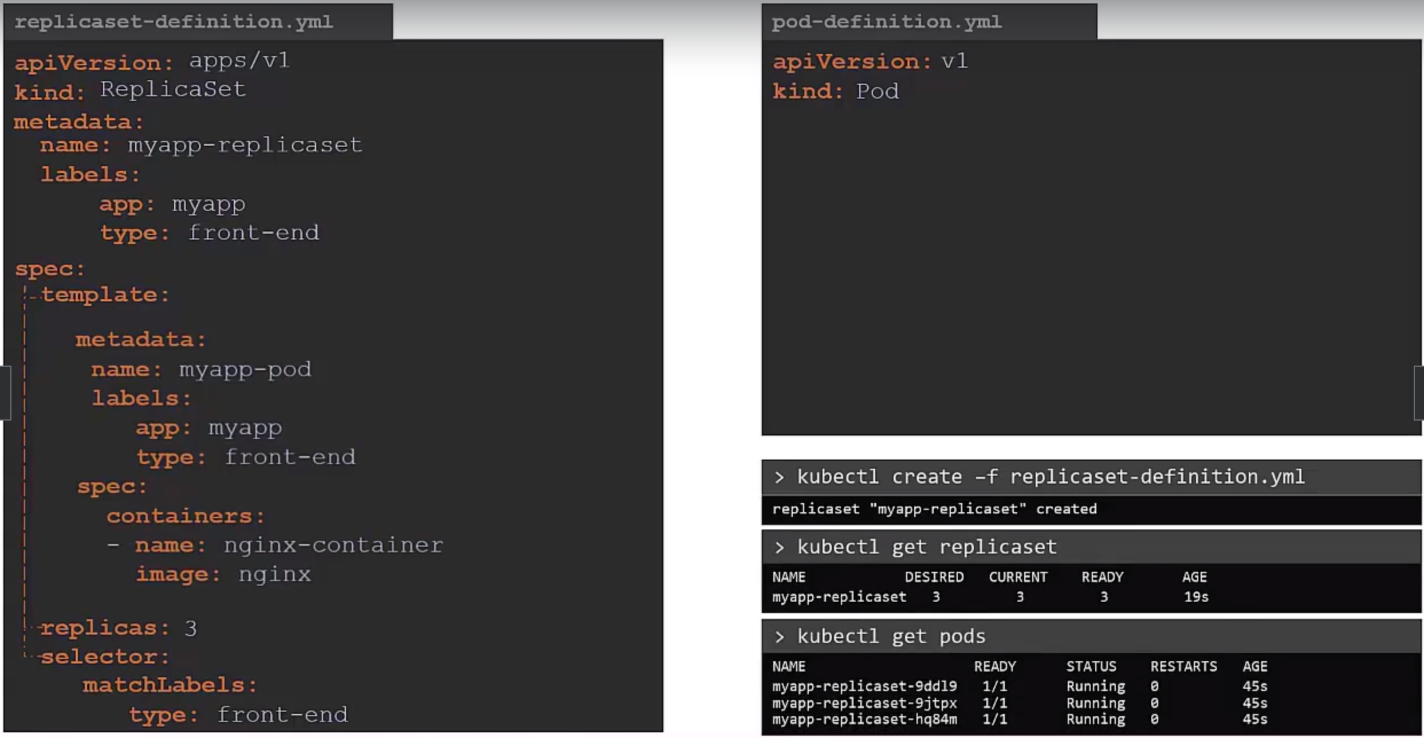


However, whatever we discussed in the previous few slides remain applicable to both these technologies. There are minor differences in the way each work. As such, we will try to stick to Replica Sets going forward. Replication Controller is supported in Kubernetes API version V1.

The template and replicas are direct children of spec sections, so they are siblings and must be on the same vertical line, which means having equal number of spaces before them.



**Replica Set:** It is very similar to Replication Controller. The API Version is different than Replication Controller (compare the image below with the above one). The selector is not a required field in case of a Replication Controller.



**Labels and Selectors**

Say we deploy three instances of our front-end web application as three pods. We would like to create a Replication Controller or Replica Set to ensure that we have three active pods at any time. You can use it to monitor existing pods if you have them already created. In case they were not created, the replica set will create them for you.

The role of the Replica Set is to monitor the pods and if any of them were to fail deploy new ones. The Replica Set is in fact a process that monitor the parts. How does the Replica Set know what parts to monitor?

There could be hundreds of other parts in the cluster running different applications. This is where labeling our pods during creation comes in handy. We could now provide these labels as a filter for replica set. Under the selector section we use the match labels filter and provide the same label that we used while creating the pods.

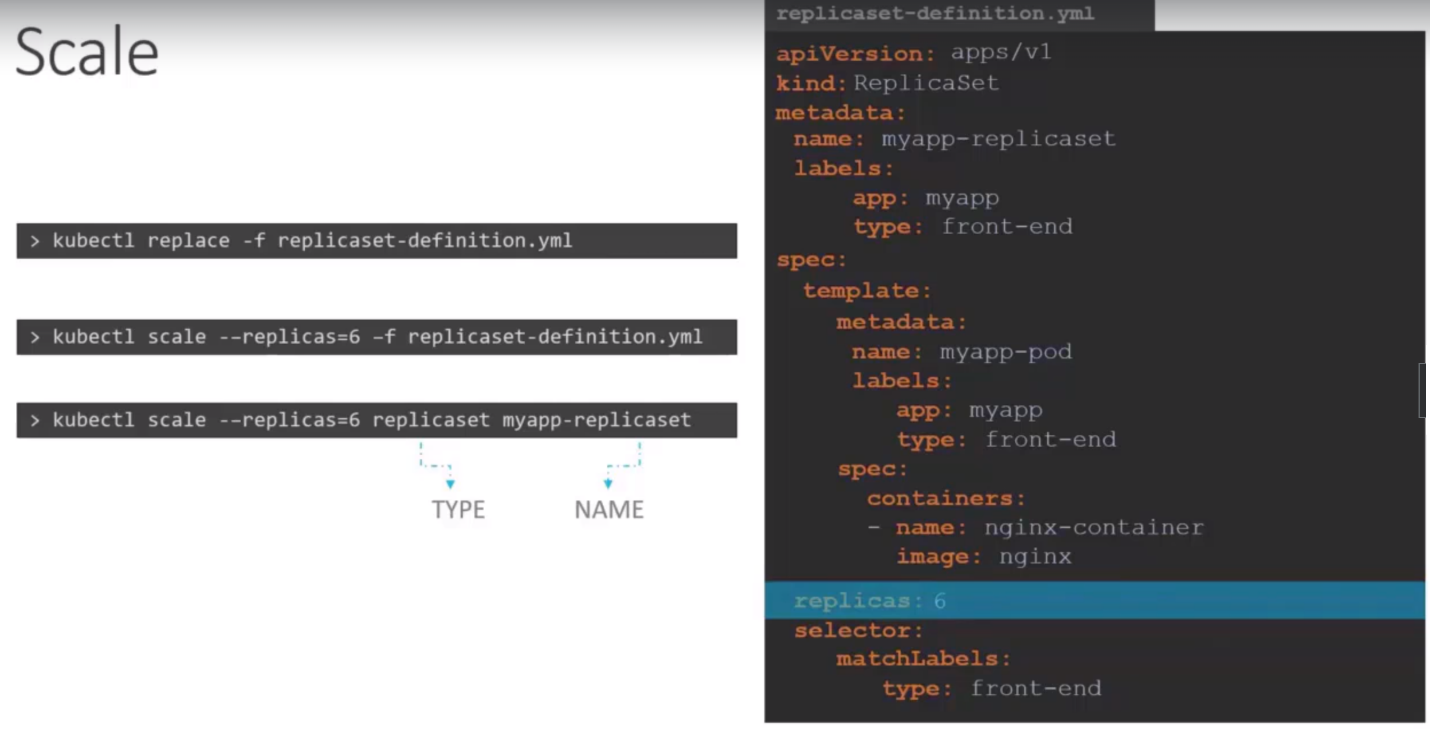


This way the Replica Set knows which pods to monitor. The same concept of labels and selectors is used in many other places throughout Kubernetes.

**Question Time**

In the Replica Set specification section, we learned that there are three sections: template, replicas and selector. We need three replicas and we have updated our selector. For instance, we have the same scenario as in the previous slide where we have three existing pods that were created already and we need to create a Replica set to monitor the pods to ensure there are a minimum of three running at all times. When the Replication Controller is created, it is not going to deploy a new instance of Pod as three of them with matching labels are already created. In that case, do we really need to provide a template section in the Replica Set specification, since we are not expecting the Replica Set to create a new Pod on deployment?

Yes, we do because in case one of the pods were to fail in future, the Replica Set needs to create a new one to maintain the desired number of pods. And for Replica Set to create a new Pod, the template definition section is required. Let’s look at how we scale the Replica Set. Assuming you want to scale from 3 to 6.



There are two ways to scale a Replica Set:

1. In the Replica Set file, update the replicas to 6 from 3 and then you have to run the replace command of Kubectl.
2. Second is just to run the scale command with updated value but it will not update the value in file.

