Open in app Get started

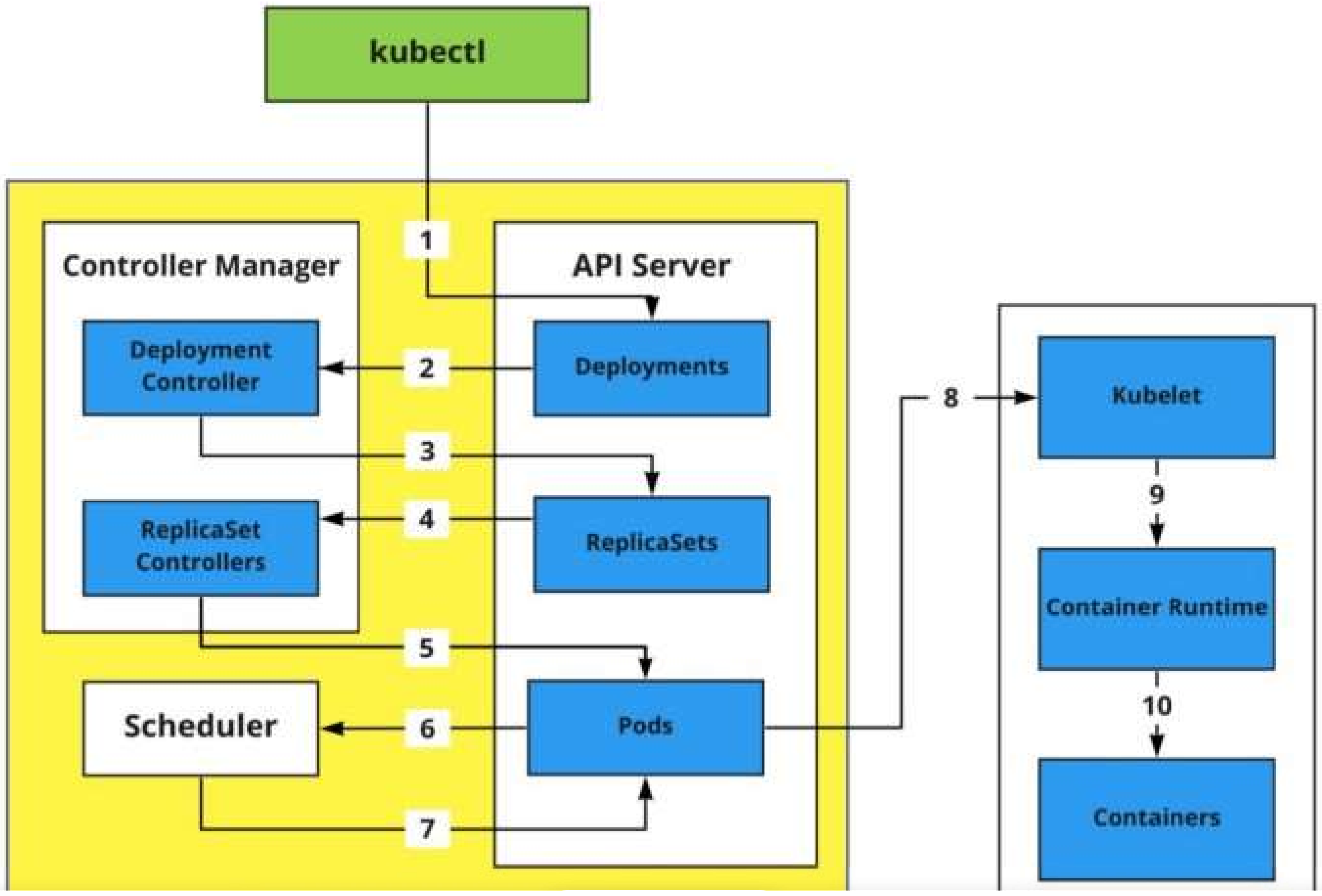
Published in techbeatly

Nived Velayudhan Follow
Jan 19 · 3 min read · Listen

Save

Chain of events behind a running Pod

What exactly happens behind the scenes when you create a pod/deployment? I'll try to cover the chain of events on a high level. I do not intend to put each and everything that's happening behind the scenes but I'll definitely cover all the important ones.



You may notice in the diagram above that I haven't included etcd here. The reason is fairly simple, the API server is the only component that can directly talk to etcd, and only it can make changes to it. So you can assume that etcd is sitting behind the API server and is hidden in this diagram. Also, I'm talking about only two main controllers here, others would also work in a similar manner.

That being said, let's understand the chain of events once the kubectl create command is executed:

Step 1: When you pass the kubectl create command, a HTTP POST request is sent to the API server which contains the deployment manifest. The API server stores this in the etcd data store and returns a response to the kubectl.

Step 2 & 3: API server has a watch mechanism and all the clients watching this get notified. One of those clients is the Deployment controller. The Deployment controller detects a Deployment object and it creates a ReplicaSet with the current specification of the Deployment. This resource is sent back to the API Server which stores it in the etcd datastore.

Step 4 & 5: Similar to the previous step, all the watchers get notified about the change made in the API Server, and this time the change is picked up by the ReplicaSet Controller. The controller understands the desired replica counts and the pod selectors defined in the object specification, creates the pod resources, and sends this information back to the API server which stores it in the etcd datastore.

Step 6 & 7: We have all the required information to run the pod but the question is which node should the pods run on? The scheduler watches for pods that don't have a node assigned to them yet and starts its process of filtering and ranking all nodes to choose the best node to run the pod on. Once the node is selected, that information would be added to the pod specification and is sent back to the API Server to store it in the etcd datastore.

Step 8, 9 & 10: Notice that all the steps up until now happened in the control plane itself and no work has been done by the worker node. The pod's creation is not handled by the instruct the container runtime in the worker node to create the container. This is when the container images are downloaded (if not already present) and the container starts running.

I think this itself should fairly help you understand the flow of events in Kubernetes. Just think about a DemonSet or StatefulSet in Kubernetes, apart from the use of different controllers — the pod creation process remains the same. Ask yourself this, if the deployment is modified to have 3 replicas of an app, what would be the chain of events that lead to the creation of the pods (refer the diagram above) and if you can figure that out — I believe I have done my job right! ;)

And of course, there would be folks like me who would still want to go way deeper into each component to understand how everything works together. For you people, read through this article by @jamiehannaford — he has done an amazing job explaining things.

Also, check out my previous article on Kubernetes here, I'm confident you would be more captivated by Kubernetes.