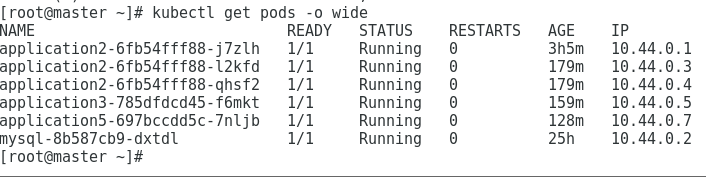
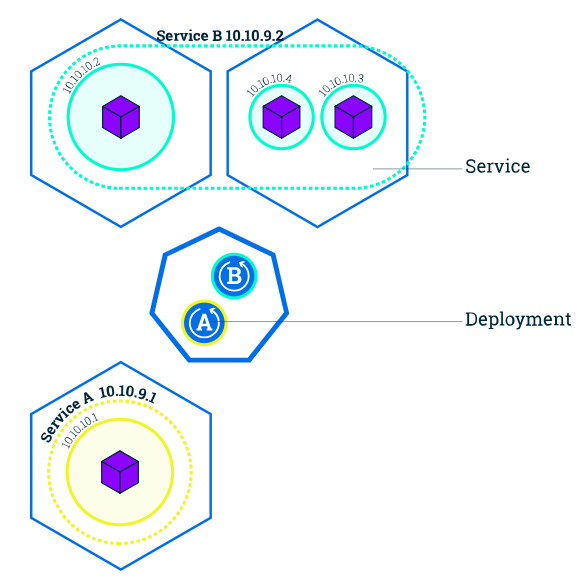
We have seen that **Pods have their own unique IP across the cluster, those IP’s are not exposed outside Kubernetes**.

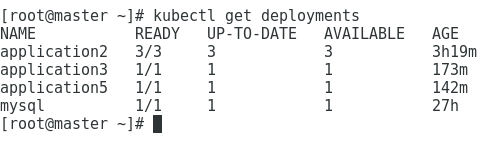


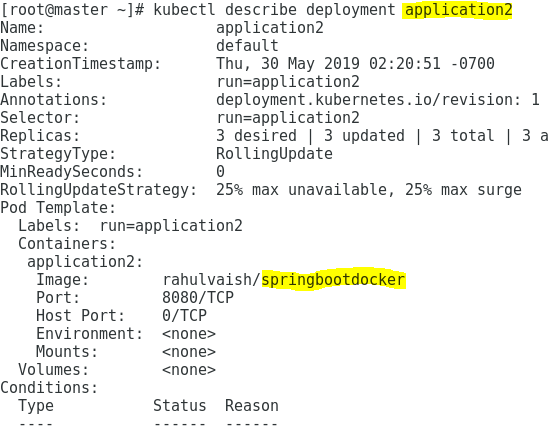
Taking this into account that over time *Pods may be terminated*, *deleted* or *replaced* by other Pods, we need a way to let other Pods and applications automatically discover each other. Kubernetes addresses this by **grouping Pods in Services**. A Kubernetes Service is an abstraction layer which defines a logical set of Pods and enables external traffic exposure, load balancing and service discovery for those Pods.

This abstraction will allow us to expose Pods to traffic originating from outside the cluster. **Services have their own unique cluster-private IP address and expose a port to receive traffic. In other words, exposing a deployment makes a service!**

**Example:**

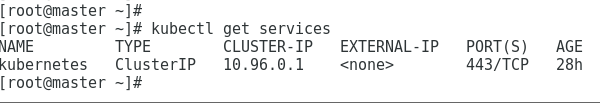




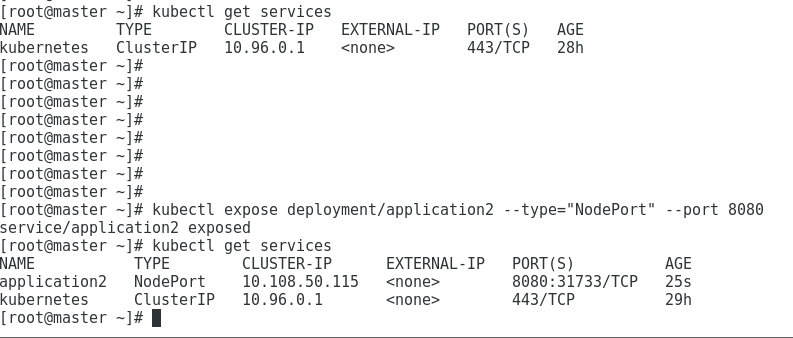


So we can see that deployment **application2** is **springbootdocker** image. Now we need to expose this deployment as service.

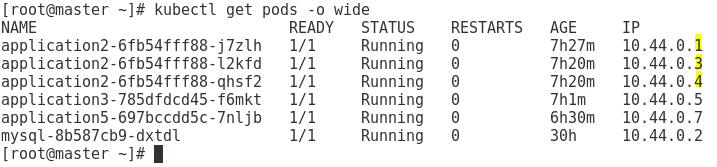
There is a default service running on the cluster, named as **kubernetes**.



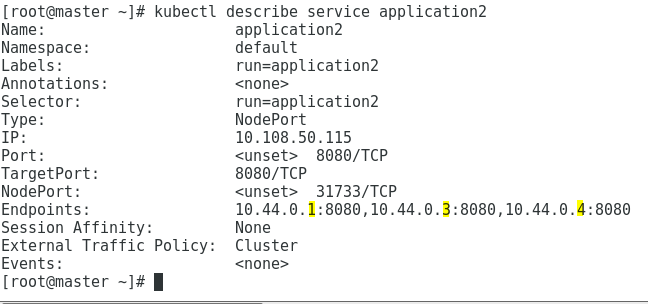
Now, we have exposed our deployment as service:



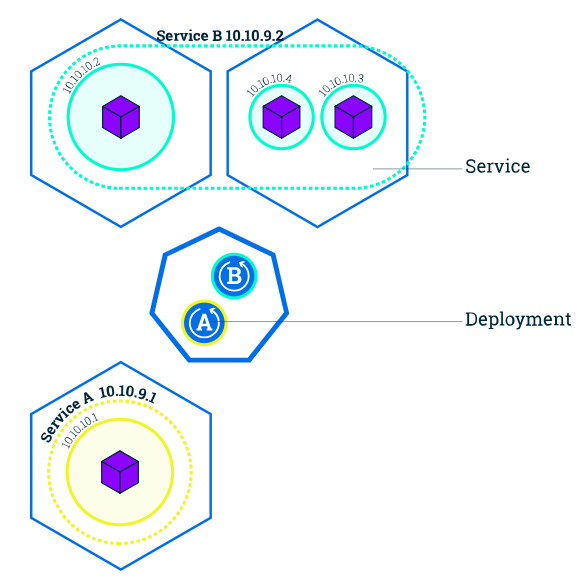
For application2 deployment we have 3 pods running:



Now when we expose the deployment, it becomes a service. The service has all the pods registered. We can confirm the same by the below command:



**Example: The below example is justified!**



Now, let’s try to export the service outside the cluster:

