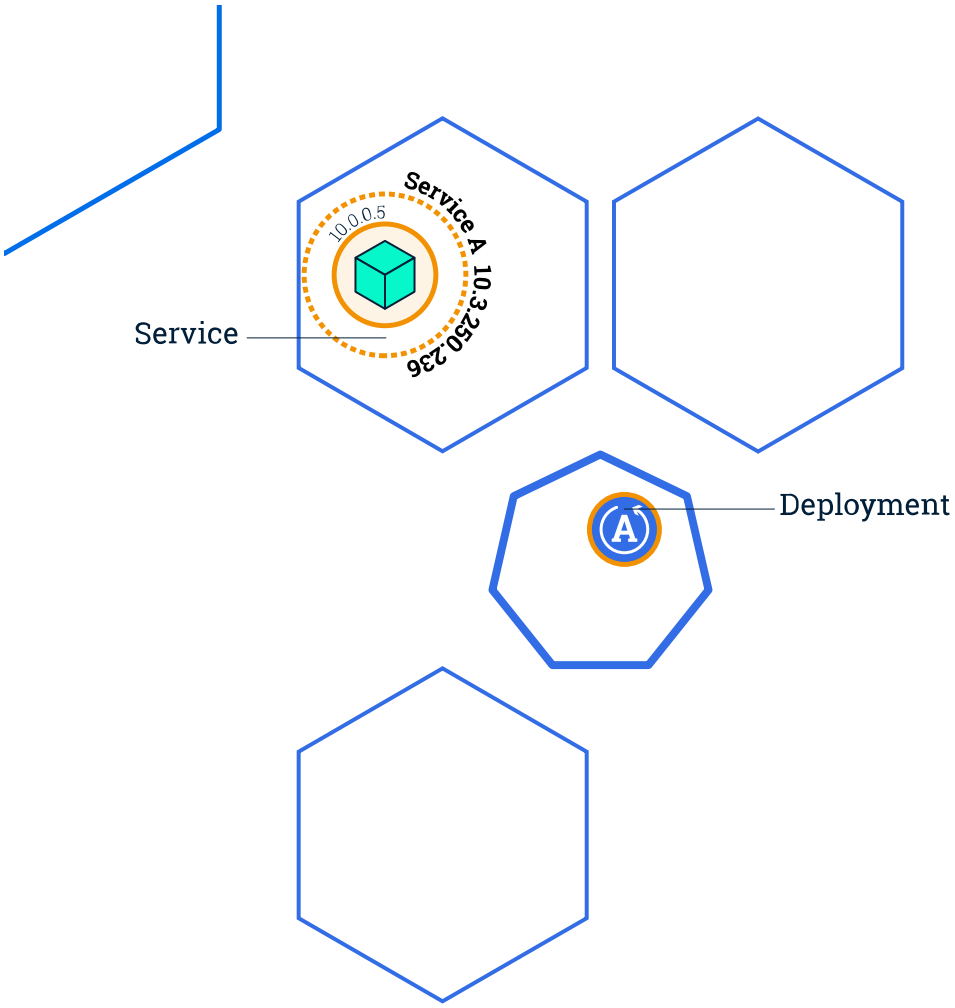
Scaling an App in Kubernetes (K8s)

Scaling

When scaling your app then **you are running multiple instances of your app**. A K8s Deployment creates only one Pod for running the application. When traffic increases, we will need to scale the application to keep up with user demand.

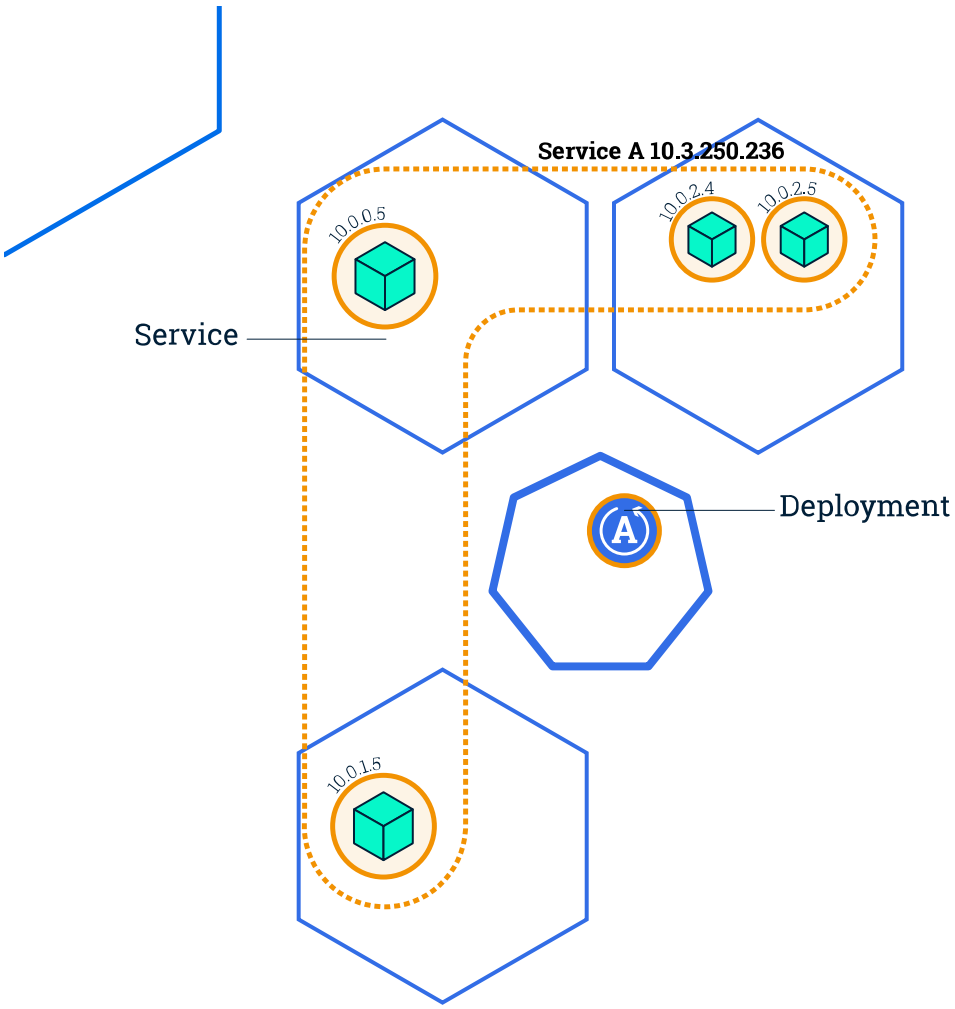


How it works

Scaling out a Deployment will ensure new Pods are created and scheduled to Nodes with available resources. Kubernetes also supports autoscaling of Pods, but it is outside the scope of this tutorial.

Running multiple instances of an application will require a way to distribute the traffic to all of them. Services have an integrated load-balancer that will distribute network traffic to all Pods of an exposed Deployment. Services will monitor continuously the running Pods using endpoints, to ensure the

traffic is sent only to available Pods. Scaling is accomplished by changing the number of replicas in a Deployment.
By the way: Once you have multiple instances of an application running, you're able to update without downtime (so-called rolling updates).



How to read the Deployment output

When you list your Deployments with ```kubectl get deployments``` then the output should be similar to:

```
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/1     1            1           11m
```

We should have 1 Pod. If not, run the command again. The columns are

- NAME: lists the names of the Deployments in the cluster.
- READY: shows the ratio of CURRENT/DESIRED replicas
- UP-TO-DATE: displays the number of replicas that have been updated to achieve the desired state.
- AVAILABLE: displays how many replicas of the application are available to your users.

- AGE: displays the amount of time that the application has been running.

ReplicaSet

Kubernetes Pods are mortal. Pods have a lifecycle. When a worker node dies, the Pods running on the Node are also lost. A ReplicaSet might then dynamically drive the cluster back to the desired state via the creation of new Pods to keep your application running.

To see the ReplicaSet created by the Deployment you run:

```
kubectl get rs
```

The name of the ReplicaSet is always formatted as

```
[DEPLOYMENT-NAME]-[RANDOM-STRING]
```

The random string is randomly generated and uses the pod-template-hash as a seed.

Two important columns of this output are:

- DESIRED displays the desired number of replicas of the application, which you define when you create the Deployment. This is the desired state.
- CURRENT displays how many replicas are currently running.