Kubernetes service types (ClusterIP, NodePort, LoadBalancer)

In Kubernetes, there are three commonly used Service types: ClusterIP, NodePort, and LoadBalancer. These Services provide different ways to make Pods accessible to other Pods within the cluster and to clients outside of it.

In this blog post, first, we're going to talk about Kubernetes Services — what they are and what problems they solve. Next, we'll get into the details of three Service types: ClusterIP, NodePort, and LoadBalancer. Lastly, we'll point out the main differences between them and discuss when to use which type.

Let's get started!

What Is a Kubernetes Service?

Kubernetes assigns Pods private IP addresses as soon as they are created in the cluster. However, there is a catch: These IP addresses are not permanent. If you delete or recreate a Pod, it gets a new IP address that is different from the one it had before. This is problematic for a client that needs to connect to a Pod. If the IP address keeps changing, which one would the client keep track of and connect to?

Imagine if you had a friend who kept changing their phone number every day. You wouldn't be able to call them or text them because you wouldn't know which number to use.

Try the Kubernetes Services Lab for free.
Kubernetes Services Lab

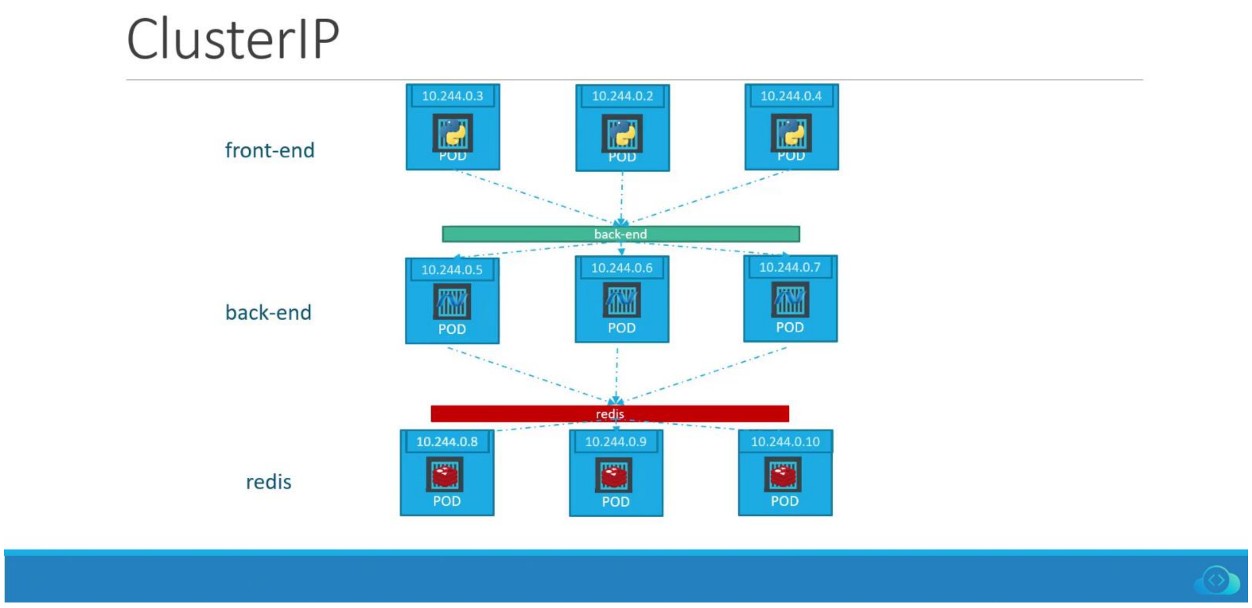
This is where Kubernetes Services comes in. A service helps clients reach one (or more) of the Pods that can fulfill their request. The Service can be reached at the same place at any point in time, so it serves as a **stable destination** that the client can use to get access to what it needs. The client doesn't have to worry about the Pods' dynamic IP addresses.

Now that we understand the basic purpose of a Kubernetes Service, let's take a closer look at how different types of Services work and what they're used for.

ClusterIP

In Kubernetes, the **ClusterIP Service is used for Pod-to-Pod communication within the same cluster**. This means that a client running outside of the cluster, such as a user accessing an application over the internet, cannot directly access a ClusterIP Service.

When a ClusterIP Service is created, it is assigned a static IP address. This address remains the same for the service's lifetime. When a client sends a request to the IP address, the request is automatically routed to one of the Pods behind the Service. If multiple Pods are associated, the ClusterIP Service uses load balancing to distribute traffic equally among them.



ClusterIP

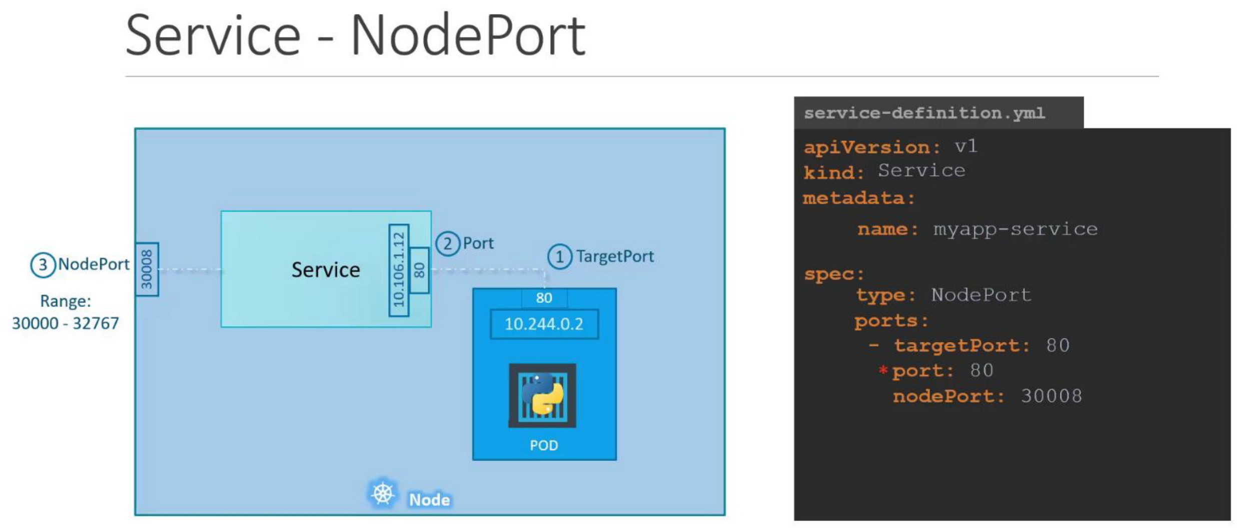
In the image above, the green bar titled "back-end" represents a ClusterIP Service. It sits in front of all the Pods labeled "back-end" and redirects incoming traffic to one of them.

Now you know what a Kubernetes ClusterIP Service is and how it works. Next, let's dive into NodePort Service.

NodePort

The **NodePort Service provides a way to expose your application to external clients**. An external client is anyone who is trying to access your application from outside of the Kubernetes cluster.

The NodePort Service does this by opening the port you choose (in the range of 30000 to 32767) on all worker nodes in the cluster. This port is what external clients will use to connect to your app. So, if the nodePort is set to 30020, for example, anyone who wants to use your app can just connect to any worker node’s IP address on port 30020, and voila! They're in.



```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
      *port: 80
       nodePort: 30008
```

NodePort

Note that a NodePort Service builds on top of the ClusterIP Service type. This means that when you create a NodePort Service, Kubernetes automatically creates a ClusterIP Service for it as well. The node receives the request, the NodePort Service picks it up, and it sends it to the ClusterIP Service, which, in turn, sends it to one of the Pods behind it (External Client -> Node ->Node->**NodePort**->ClusterIP->Pod).

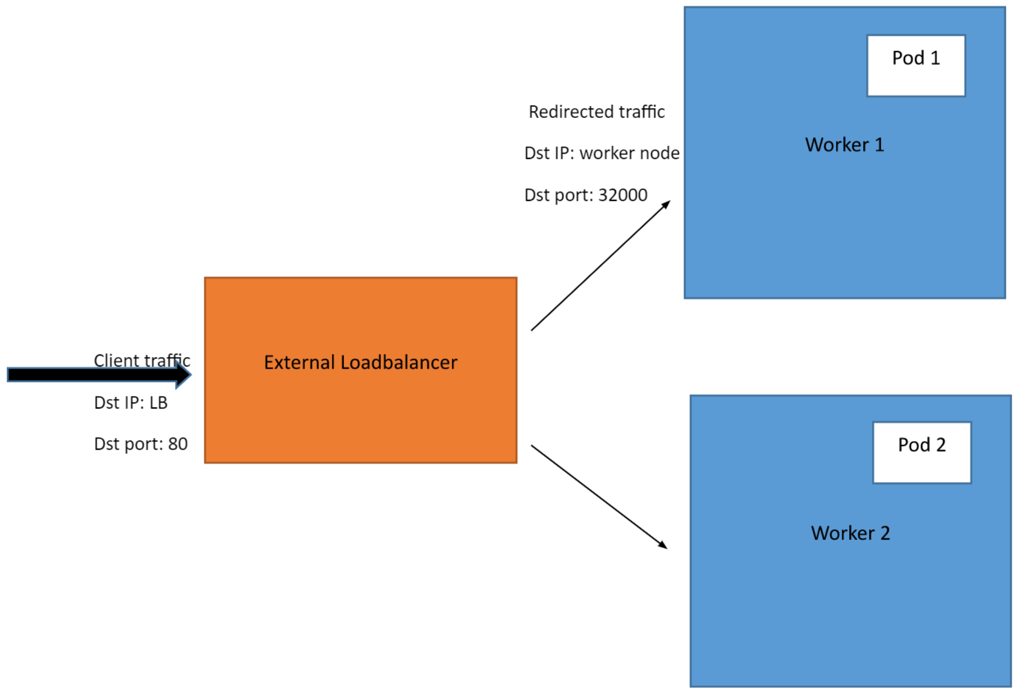
The extra benefit is that internal clients can still access those Pods quickly. They can skip going through the NodePort and reach the ClusterIP directly to connect to one of the Pods.

One disadvantage of the NodePort Service is that it doesn't do any kind of load balancing across multiple nodes. It simply directs traffic to whichever node the client connected to. This can create a problem: Some nodes can get overwhelmed with requests while others sit idle.

Now that you understand the NodePort Service well, it’s time to examine the LoadBalancer Service.

LoadBalancer

A LoadBalancer Service is another way you can expose your applications to external clients. However, it only works if you're using Kubernetes on a cloud platform that supports this Service type.



LoadBlancer

Now, when you create a LoadBalancer Service, Kubernetes detects which cloud computing platform your cluster is running on and creates a load balancer in the infrastructure of the cloud provider. The load balancer will have its own unique, publicly accessible IP address that clients can use to connect to your application.

For example, if you're running a Kubernetes cluster on a cloud platform like Amazon Web Services (AWS), you can create a LoadBalancer Service. When you do this, Kubernetes will create an Elastic Load Balancer in AWS to route traffic to the nodes in your cluster.

Note that the LoadBalancer Service, this time, builds on top of the NodePort Service, with an added benefit: It adds load-balancing functionality to distribute traffic between nodes. This reduces the negative effects of any one node failing or becoming overloaded with requests.

The traffic coming from external clients goes through a path like this: External client -> Loadbalancer -> Worker node IP -> NodePort -> ClusterIP Service -> Pod.

ClusterIP vs. NodePort vs. LoadBalancer: Key Differences & Use Cases

Now, let’s discuss the key differences among ClusterIP, NodePort, and LoadBalancer Service types and when to use which type.

Key difference

	ClusterIP	**NodePort**	**LoadBalancer**
Communication	Pod-to-Pod	External client-to-Pod (**No** load balancing between nodes)	External client-to-Pod (Load balancing between nodes)
Cloud platform required?	No	No	Yes

ClusterIP is used for Pod-to-Pod communication within the same Kubernetes cluster. In contrast, NodePort and LoadBalancer Services are used for communication between applications within the cluster and external clients outside the cluster.

When NodePort is used, the client connects **directly** to a certain node (client->node). When LoadBalancer is used, the client connects to the cloud platform's load balancer instead (client->load balancer->node). Then, the load balancer picks a node and connects the client to it.

So it sits in the middle, so to speak. Clients don't connect directly to nodes anymore when this is used. The LoadBalancer can do its job and actually load-balance traffic. This ensures nodes are somewhat evenly used and no single node is hammered with all the requests.

Another thing worth mentioning is that the LoadBalancer Service can only be used when the Kubernetes cluster is provisioned on a cloud platform that supports this Service type.

Now that you understand the key differences among ClusterIP, NodePort, and LoadBalancer Service types, let's discuss when to use which type.

Use cases

	ClusterIP	**NodePort**	**LoadBalancer**
Use case	To allow Pod-to-Pod communication within the same cluster	To expose app(s) inside the cluster to external clients (outside the cluster). Client requests go to the same node they connected to.	To expose app(s) inside the cluster to external clients (outside the cluster). Client requests are load balanced across multiple nodes.

The choice of which Kubernetes Service type to use depends on the specific requirements of your application and the environment where it is running. Having said that here is a brief overview of when to use which Service type:

ClusterIP:

Use this Service type when you want to expose an application within the cluster and allow other Pods within the cluster to access it.

NodePort:

Use this service type when you want to expose your application to a specific port on each worker node in the cluster, making it accessible to external connections (coming from outside the cluster). NodePort Services are often used for development and testing purposes.

LoadBalancer:

It sounds like the same thing mentioned for NodePort, but there's an added benefit. You take advantage of a cloud provider's load balancing capabilities and like the same thing mentioned for NodePort. But there's the added benefit. You take advantage of a cloud provider's load-balancing capabilities. All client requests can be smoothly load-balanced to multiple nodes in your cluster.

LoadBalancer Services are typically used in production environments. Why? One big reason is the increased reliability. When clients connect to one node specifically (through NodePort), if that node fails, the clients will be left hanging. Their requests will remain unfulfilled as the node is unreachable.

But, with a LoadBalancer, if one node fails, the LoadBalancer doesn't rely on a single node (it sends traffic to all). So, only a few requests hitting the problematic node will fail, not all.

With proper health checks in place, the LoadBalancer can stop sending traffic to the failed node, so future client requests can all land on healthy nodes.