



What is an Ingress?

Kubernetes - Beginners | Intermediate | Advanced

[View on GitHub](#) [Join Slack](#)

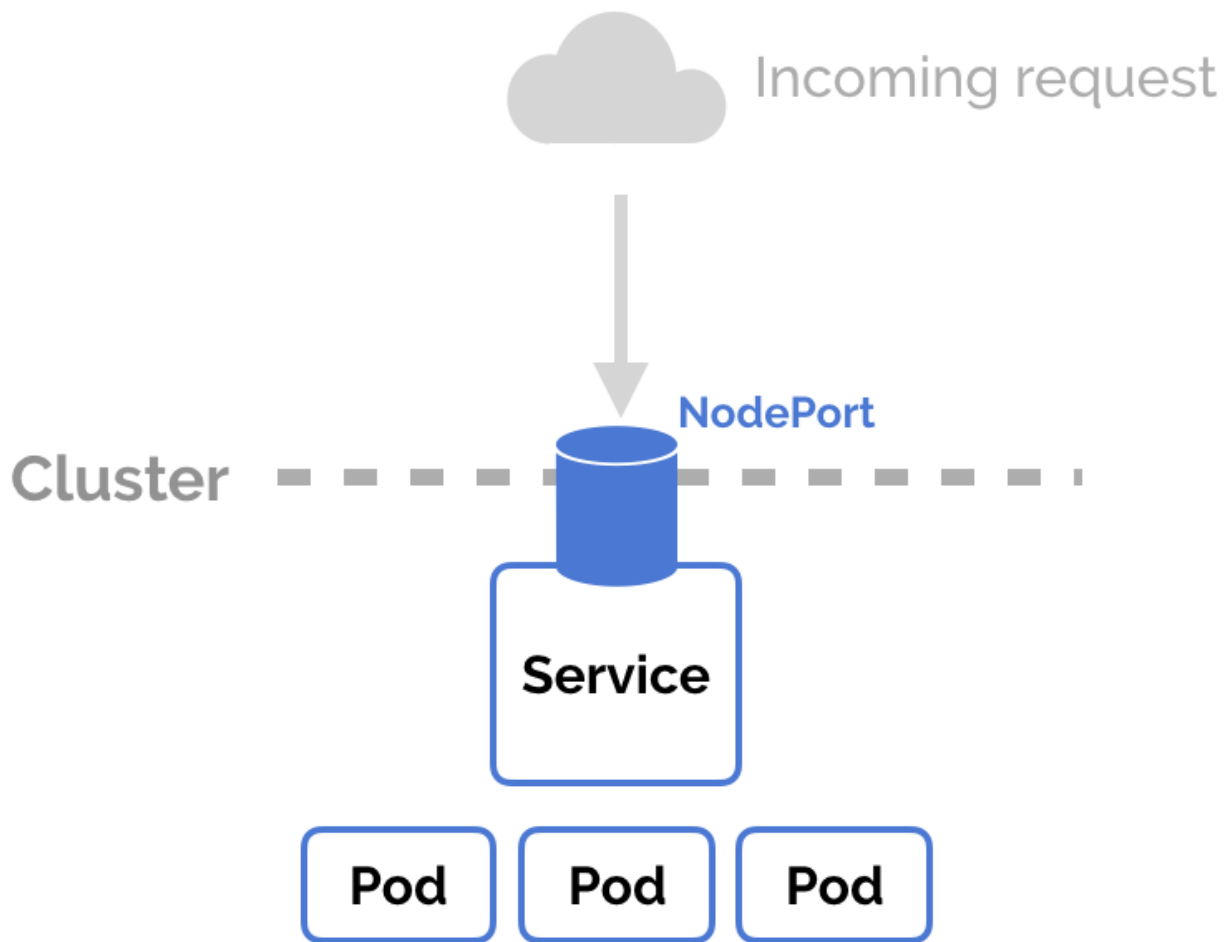
What is an Ingress?

- In Kubernetes, an Ingress is an object that allows access to your Kubernetes services from outside the Kubernetes cluster. You configure access by creating a collection of rules that define which inbound connections reach which services.
- This lets you consolidate your routing rules into a single resource. For example, you might want to send requests to `example.com/api/v1/` to an `api-v1` service, and requests to `example.com/api/v2/` to the `api-v2` service. With an Ingress, you can easily set this up without creating a bunch of LoadBalancers or exposing each service on the Node.

Kubernetes Ingress vs LoadBalancer vs NodePort

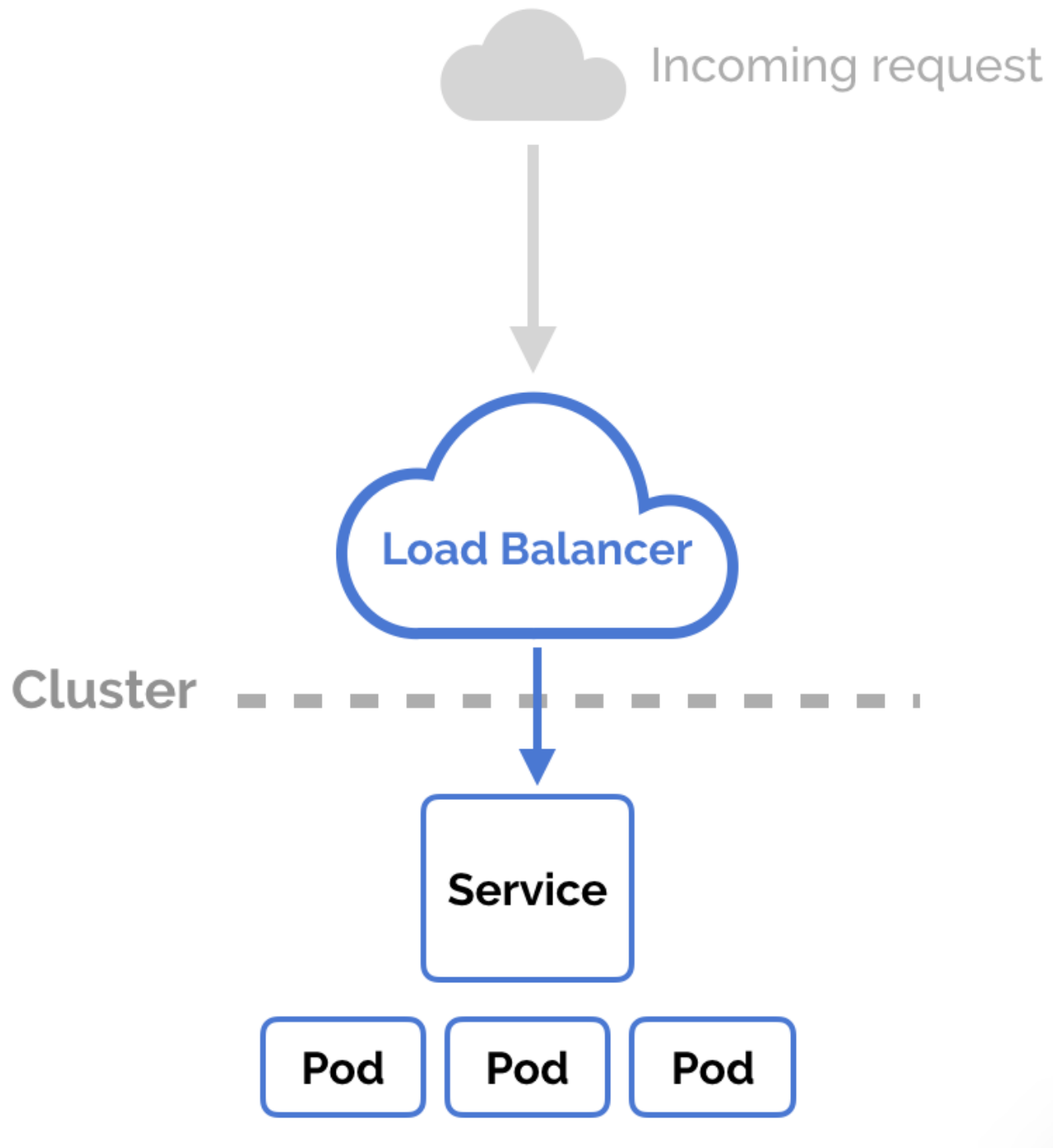
These options all do the same thing. They let you expose a service to external network requests. They let you send a request from outside the Kubernetes cluster to a service inside the cluster.

NodePort



- NodePort is a configuration setting you declare in a service's YAML. Set the service spec's type to NodePort. Then, Kubernetes will allocate a specific port on each Node to that service, and any request to your cluster on that port gets forwarded to the service.
- This is cool and easy, it's just not super robust. You don't know what port your service is going to be allocated, and the port might get re-allocated at some point.

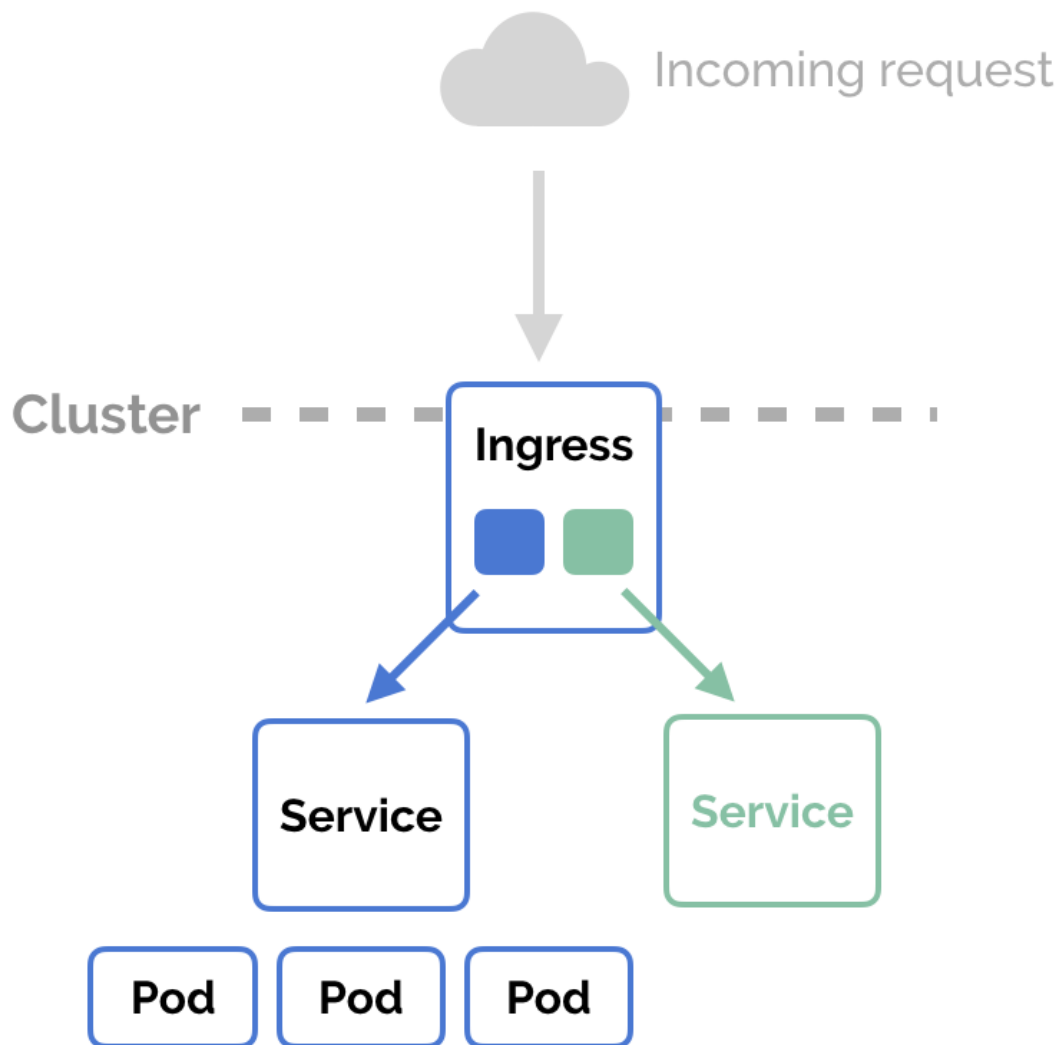
LoadBalancer



- You can set a service to be of type LoadBalancer the same way you'd set NodePort—specify the type property in the service's YAML. There needs to be some external load balancer functionality in the cluster, typically implemented by a cloud provider.
- This is typically heavily dependent on the cloud provider—GKE creates a Network Load Balancer with an IP address that you can use to access your service.

- Every time you want to expose a service to the outside world, you have to create a new LoadBalancer and get an IP address.

Ingress



- NodePort and LoadBalancer let you expose a service by specifying that value in the service's type. Ingress, on the other hand, is a completely independent resource to your service. You declare, create and destroy it separately to your services.
- This makes it decoupled and isolated from the services you want to expose. It also helps you to consolidate routing rules into one place.

- The one downside is that you need to configure an Ingress Controller for your cluster. But that's pretty easy—in this example, we'll use the Nginx Ingress Controller.

How to Use Nginx Ingress Controller

- Start by creating the “mandatory” resources for Nginx Ingress in your cluster.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx
```

- Then, enable the ingress add-on for Minikube

```
minikube addons enable ingress
```

- Or, if you're using Docker for Mac to run Kubernetes instead of Minikube.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx
```

- Check that it's all set up correctly.

```
kubectl get pods --all-namespaces -l app=ingress-nginx
```

Creating a Kubernetes Ingress

- First, let's create two services to demonstrate how the Ingress routes our request. We'll run two web applications that output a slightly different response.

```
git clone https://github.com/collabnix/kubelabs
cd ingress101
$ kubectl apply -f apple.yaml
$ kubectl apply -f banana.yaml
```

- Create the Ingress in the cluster

```
kubectl create -f ingress.yaml
```

Perfect! Let's check that it's working. If you're using Minikube, you might need to replace localhost with minikube IP.

```
$ curl -kL http://localhost/apple  
apple  
  
$ curl -kL http://localhost/banana  
banana  
  
$ curl -kL http://localhost/notfound  
default backend - 404
```

Ingress Controllers and Ingress Resources

- Kubernetes supports a high level abstraction called Ingress, which allows simple host or URL based HTTP routing. An ingress is a core concept (in beta) of Kubernetes, but is always implemented by a third party proxy. These implementations are known as ingress controllers. An ingress controller is responsible for reading the Ingress Resource information and processing that data accordingly. Different ingress controllers have extended the specification in different ways to support additional use cases.
- Ingress is tightly integrated into Kubernetes, meaning that your existing workflows around kubectl will likely extend nicely to managing ingress. Note that an ingress controller typically doesn't eliminate the need for an external load balancer — the ingress controller simply adds an additional layer of routing and control behind the load balancer.

Contributors


Sangam Biradar


Join KubeDaily


7 Members Online

Support


MEMBERS ONLINE

 h3ll_boy


 MEE6

 Nitinkashyap

Apex Legends

 ojaswa

 Parmeshwar

 prasad

 vikas027

Free voice chat from Discord

Connect

[Tweets by collabnix](#)

kubelabs is maintained by **collabnix**.

This page was generated by [GitHub Pages](#).