

# Kubernetes NodePort vs LoadBalancer vs Ingress? When should I use what?



Sandeep Dinesh

Follow

Mar 11, 2018 · 5 min read



Recently, someone asked me what the difference between NodePorts, LoadBalancers, and Ingress were. They are all different ways to get external traffic into your cluster, and they all do it in different ways. Let's take a look at how each of them work, and when you would use each.

**Note:** *Everything here applies to Google Kubernetes Engine. If you are running on another cloud, on prem, with minikube, or something else, these will be slightly different. I'm also not going into deep technical details. If you are interested in learning more, the official documentation is a great resource!*

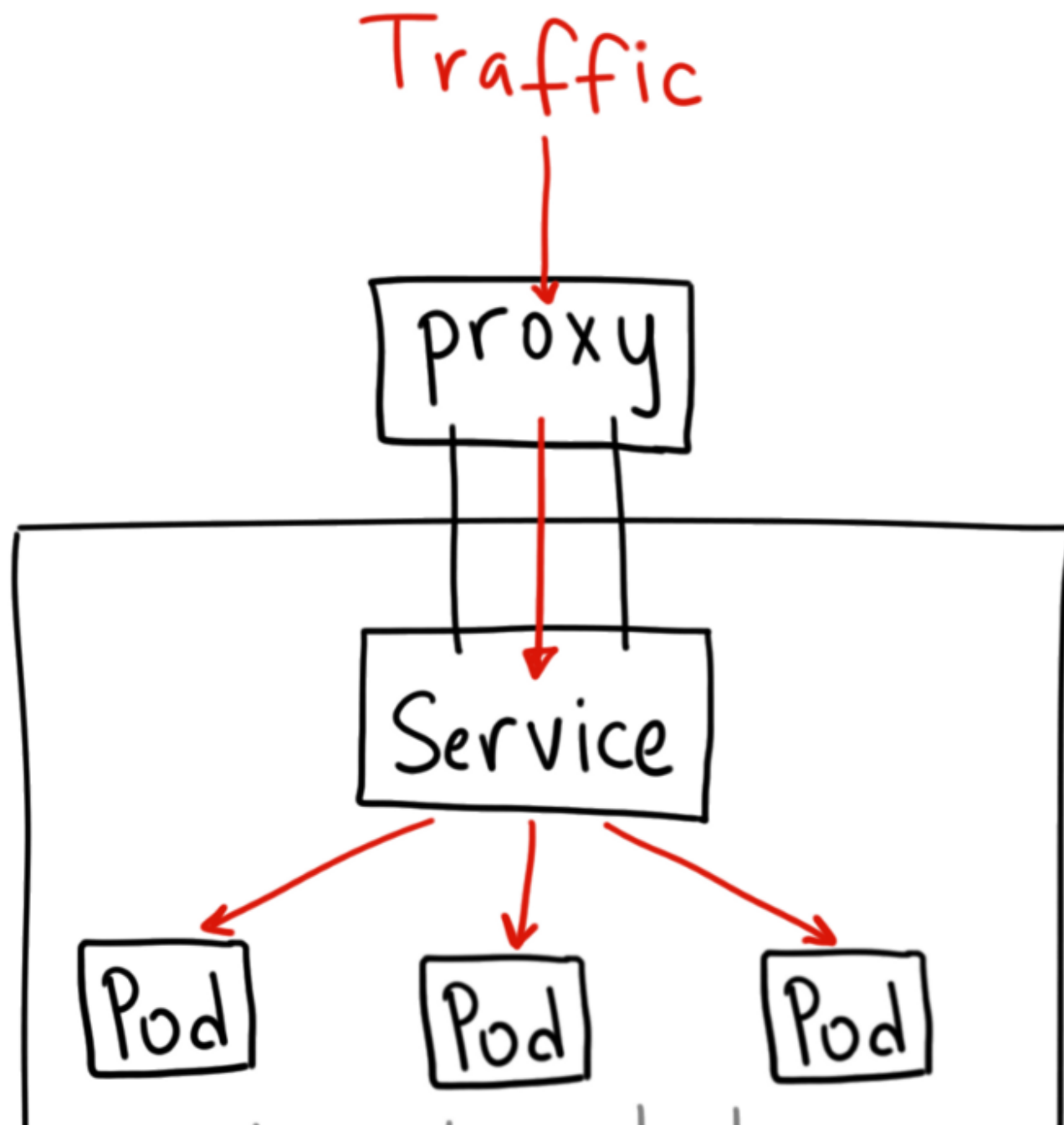
# ClusterIP

A ClusterIP service is the default Kubernetes service. It gives you a service inside your cluster that other apps inside your cluster can access. There is no external access.

The YAML for a ClusterIP service looks like this:

```
apiVersion: v1
kind: Service
metadata:
  name: my-internal-service
spec:
  selector:
    app: my-app
  type: ClusterIP
  ports:
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP
```

If you can't access a ClusterIP service from the internet, why am I talking about it? Turns out you can access it using the Kubernetes proxy!



# Kubernetes cluster

Thanks to [Ahmet Alp Balkan](#) for the diagrams

Start the Kubernetes Proxy:

```
$ kubectl proxy --port=8080
```

Now, you can navigate through the Kubernetes API to access this service using this scheme:

`http://localhost:8080/api/v1/proxy/namespaces/<NAMESPACE>/services/<SERVICE-NAME>:<PORT-NAME>/`

So to access the service we defined above, you could use the following address:

`http://localhost:8080/api/v1/proxy/namespaces/default/services/my-internal-service:http/`

## When would you use this?

There are a few scenarios where you would use the Kubernetes proxy to access your services.

1. Debugging your services, or connecting to them directly from your laptop for some reason
2. Allowing internal traffic, displaying internal dashboards, etc.

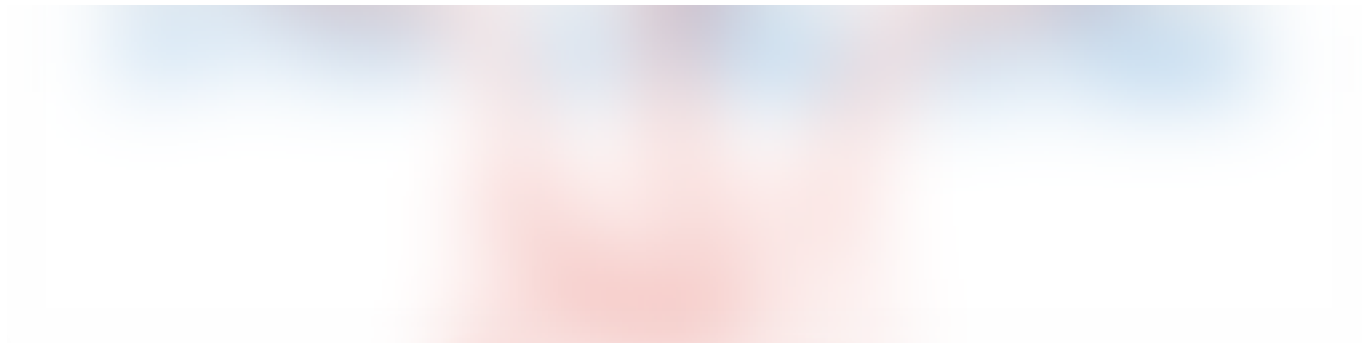
Because this method requires you to run `kubectl` as an authenticated user, you should NOT use this to expose your service to the internet or use it for production services.

## NodePort

A NodePort service is the most primitive way to get external traffic directly to your service. NodePort, as the name implies, opens a specific port on all

the Nodes (the VMs), and any traffic that is sent to this port is forwarded to the service.





This isn't the most technically accurate diagram, but I think it illustrates the point of how a NodePort works

The YAML for a NodePort service looks like this:

```
apiVersion: v1
kind: Service
metadata:
  name: my-nodeport-service
spec:
  selector:
    app: my-app
  type: NodePort
  ports:
    - name: http
      port: 80
      targetPort: 80
      nodePort: 30036
      protocol: TCP
```

Basically, a NodePort service has two differences from a normal “ClusterIP” service. First, the type is “NodePort.” There is also an additional port called the nodePort that specifies which port to open on the nodes. If you don’t specify this port, it will pick a random port. Most of the time you should let Kubernetes choose the port; as [thockin](#) says, there are many caveats to what ports are available for you to use.

## When would you use this?

There are many downsides to this method:

1. You can only have one service per port
2. You can only use ports 30000–32767
3. If your Node/VM IP address change, you need to deal with that

For these reasons, I don’t recommend using this method in production to directly expose your service. If you are running a service that doesn’t have to be always available, or you are very cost sensitive, this method will work for you. A good example of such an application is a demo app or something temporary.



# LoadBalancer

A LoadBalancer service is the standard way to expose a service to the internet. On GKE, this will spin up a Network Load Balancer that will give you a single IP address that will forward all traffic to your service.





Thanks to [Ahmet Alp Balkan](#) for the diagrams

## When would you use this?

If you want to directly expose a service, this is the default method. All traffic on the port you specify will be forwarded to the service. There is no filtering, no routing, etc. This means you can send almost any kind of traffic to it, like HTTP, TCP, UDP, Websockets, gRPC, or whatever.

The big downside is that each service you expose with a LoadBalancer will get its own IP address, and you have to pay for a LoadBalancer per exposed service, which can get expensive!

# Ingress

Unlike all the above examples, Ingress is actually NOT a type of service. Instead, it sits in front of multiple services and act as a “smart router” or entrypoint into your cluster.

You can do a lot of different things with an Ingress, and there are many types of Ingress controllers that have different capabilities.

The default GKE ingress controller will spin up a HTTP(S) Load Balancer for you. This will let you do both path based and subdomain based routing to backend services. For example, you can send everything on `foo.yourdomain.com` to the foo service, and everything under the `yourdomain.com/bar/` path to the bar service.





Thanks to [Ahmet Alp Balkan](#) for the diagrams

The YAML for a Ingress object on GKE with a L7 HTTP Load Balancer might look like this:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-ingress
spec:
  backend:
    serviceName: other
    servicePort: 8080
  rules:
  - host: foo.mydomain.com
    http:
      paths:
      - backend:
```

```
        serviceName: foo
        servicePort: 8080
- host: mydomain.com
  http:
    paths:
    - path: /bar/*
      backend:
        serviceName: bar
        servicePort: 8080
```

## When would you use this?

Ingress is probably the most powerful way to expose your services, but can also be the most complicated. There are many types of Ingress controllers, from the [Google Cloud Load Balancer](#), [Nginx](#), [Contour](#), [Istio](#), and more. There are also plugins for Ingress controllers, like the [cert-manager](#), that can automatically provision SSL certificates for your services.

Ingress is the most useful if you want to expose multiple services under the same IP address, and these services all use the same L7 protocol (typically HTTP). You only pay for one load balancer if you are using the native GCP integration, and because Ingress is “smart” you can get a lot of features out of the box (like SSL, Auth, Routing, etc)

[Kubernetes](#)[Microservices](#)[Services](#)[Load Balancing](#)[Ingress](#)

13.6K claps



WRITTEN BY

**Sandeep Dinesh**

Follow



## Google Cloud - Community

Follow

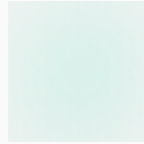
A collection of technical articles published or curated by Google Cloud Developer Advocates. The views expressed are those of the authors and don't necessarily reflect those of Google.

[See responses \(61\)](#)

## More From Medium

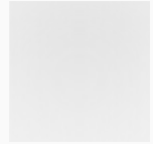
### Should we avoid "Header Interfaces"?

Kristoffer Olsson



### Chronic early-adopter tech-debt syndrome

Rotem Tamir in The Startup



### Sass finally introduced a new module system

Fynn in COYO Tech



### How to use Velocity and Capacity to get reliable forecasts

Alexander Postnikov



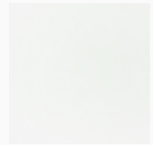
### No, the Cloud is not too expensive

Phil Goerd



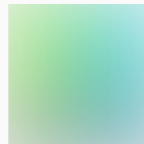
### Running through the Google GCP Cloud Foundation Toolkit Setup

pascal.brokmeier in datamindedbe



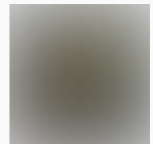
### BlueHost vs A2Hosting [2019]

David Cross



### Decoupling your technical code from your business logic with the Hexagonal Architecture (HexArch)

Julien Topçu



### Discover Medium

Welcome to a place where words matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. [Watch](#)

### Make Medium yours

Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. [Explore](#)

### Become a member

Get unlimited access to the best stories on Medium — and support writers while you're at it. Just \$5/month. [Upgrade](#)

---

# Medium

About

Help

Legal