Learn how to use Kubernetes External IP service type

Mohamad Fadhil Follow
Dec 21, 2019 · 7 min read ★

This article originally published here

When building a bare metal Kubernetes cluster, you might face a common problem as I do. You don't know how to expose your Kubernetes service to the Internet other than using NodePort. If you are using the NodePort service type, it will assign a high port number, and you have to allow your firewall rule to connect to those ports. That is not good for your infrastructure, especially when you're exposing the server to the outside Internet. Well, there is another neat way to expose your Kubernetes service to the world, and you can use its original port number. For example, you can expose MySQL service that resides in your Kubernetes cluster to the outside world on port 3306 rather than port 32767. The answer is using Kubernetes External IP service type.

Personally, I find that this topic not widely discussed among the Kubernetes community, probably because many are using cloud providers' Load Balancer or use Metal LB for on-

prem deployments.

What is External IP Service

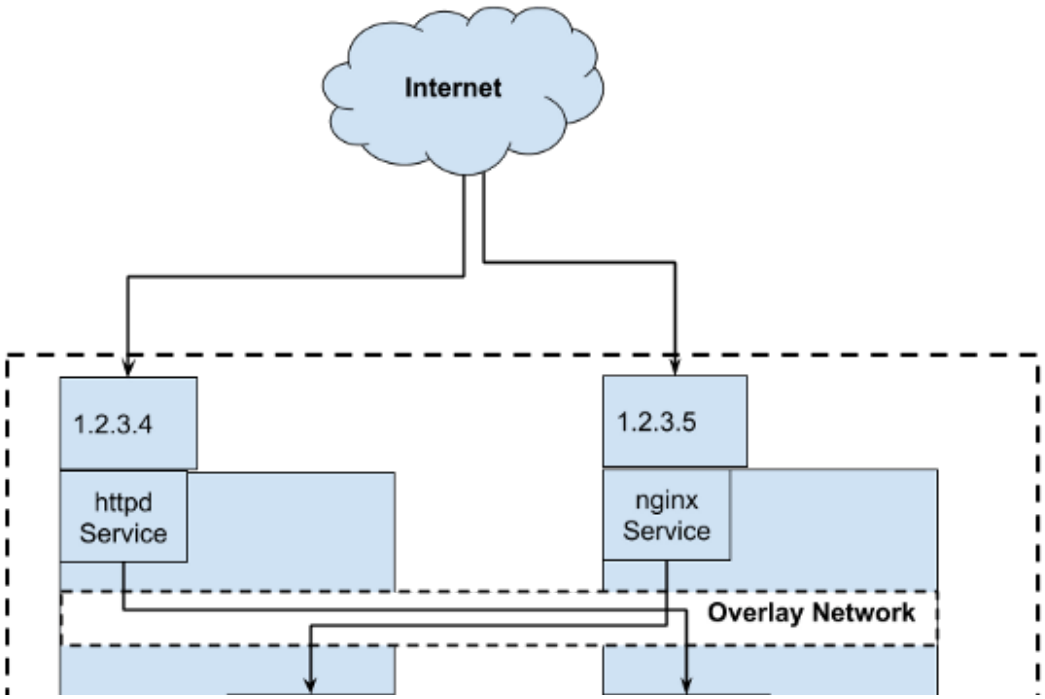
From official Kubernetes documentation, this is how External IP is described

> If there are external IPs that route to one or more cluster nodes, Kubernetes Services can be exposed on those `externalIPs`. Traffic that ingresses into the cluster with the external IP (as destination IP), on the Service port, will be routed to one of the Service endpoints. `externalIPs` are not managed by Kubernetes and are the responsibility of the cluster administrator.

The explanation is understandable for most people. The most important thing here is to be sure which IP is used to reach the Kubernetes cluster. Using external IP service type, we can *bind* the service to the IP used to connect to the cluster.

It is good if you briefly know how Kubernetes networking works. If you are not familiar with it, do check out this blog post written by Mark Betz to understand them in detail. The most important this to know here is that the Kubernetes network works with the Overlay network. That means once you reach any of the nodes (master or worker node) in the cluster, you can virtually reach everything in the cluster.

This is how the network is illustrated



In the diagram above, both Node 1 and Node 2 has 1 IP address. The IP address 1.2.3.4 on Node 1 is bind to httpd service where the actual pod resides in Node 2, and the IP address 1.2.3.6 is bind to Nginx service in that the real pod live in Node 1. The underlying Overlay network makes this possible. When we curl IP address 1.2.3.4, we should see a response from httpd service while curl 1.2.3.5, we should receive a response from Nginx service.

Why not use Ingress?

Even though Ingress is used to expose service to outside, Ingress is built for L7 routing. That means it built to support HTTP (port 80) and/or HTTPS (port 443) traffic and not for other ports. Ingress works as host-based routing or similar to virtual host in the Web Server world. Some ingress controllers able to serve other ports or may provides a workaround for L4 routing, but I have never actually experimented with them.

Advantages & Disadvantages of External IP

The advantage of using External IP is:

- You have full control of the IP that you use. You can use IP that belongs to your ASN instead of the cloud provider's ASN.

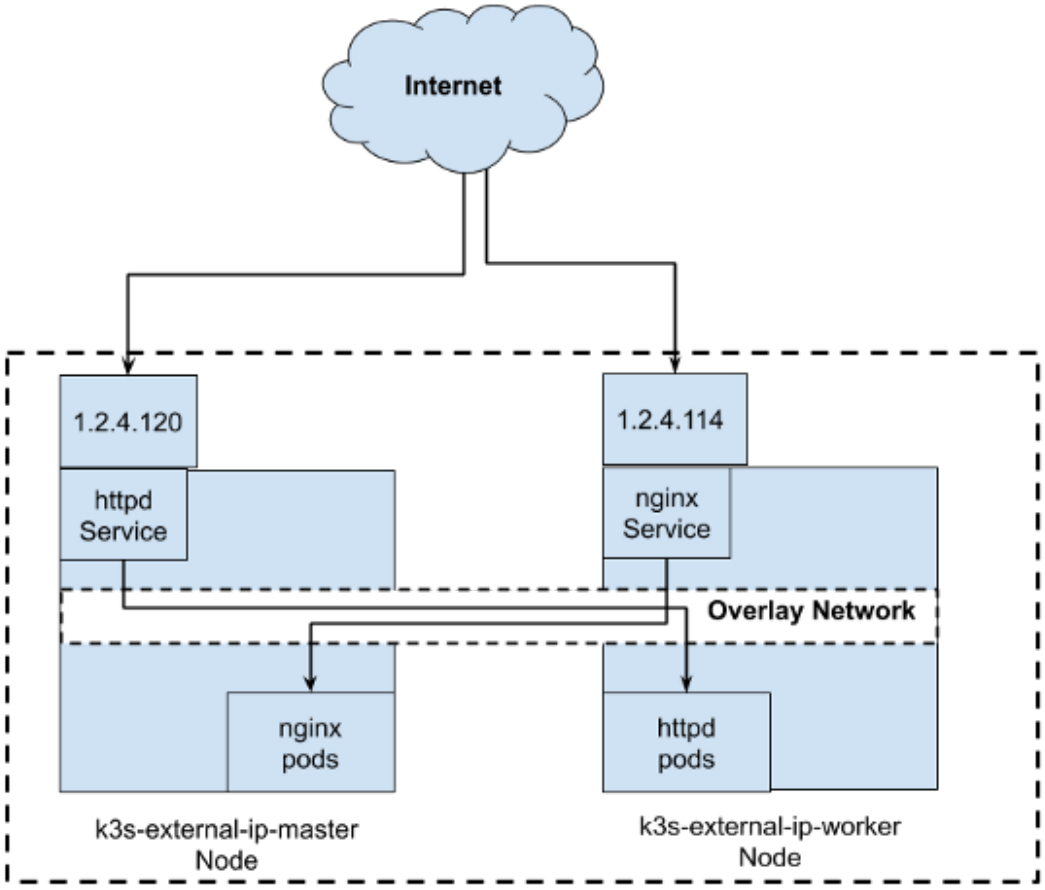
The disadvantages of External IP are:

- The simple setup that we will go thru right now is **NOT** highly available. That means if the node dies, the service is no longer reachable, and you'll need to remediate the issue manually.
- There are some manual works needs to be done to manage the IPs. The IPs are not dynamically provisioned for you; thus, it requires manual human intervention.

How to use External IP service

Our setup

Again, we will use the same diagram as our reference for our cluster setup, except with different IP and different hostname. This setup is not a really meaningful example. Still, it's easy to distinguish which is which when we're verifying the configuration. In real life example, you might want to expose MySQL DB on one external IP and Kafka cluster on another external IP.



I have provisioned 2 VMs for this tutorial. `k3s-external-ip-master` will be our Kubernetes master node and has an IP of 1.2.4.120. `k3s-external-ip-worker` will be Kubernetes worker and has an IP of 1.2.4.114.

Step 1: Setup Kubernetes cluster

Let's install k3s on the master node and let another node to join the cluster.

```
$ k3sup install --ip <master node ip> --user <username>
$ k3sup join --server-ip <master node ip> --ip <worker node ip> --
user <username>
```

You should be seeing something like this now

```
$ kubectl get nodes
NAME                                STATUS   ROLES    AGE
VERSION
k3s-external-ip-master              Ready    master   7m24s
v1.16.3-k3s.2
k3s-external-ip-worker              Ready    <none>   2m21s
v1.16.3-k3s.2
```

Step 2: Create Kubernetes deployments

Let's create Nginx deployment and httpd deployment.

```
$ kubectl create deployment nginx --image=nginx
$ kubectl create deployment httpd --image=httpd
```

You should be seeing this now

```
$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-86c57db685-fzxn5   1/1     Running   0          22s
httpd-7bddd4bd85-zk8ks   1/1     Running   0          16s
```

Step 3: Expose the deployments as External IP type

Let's expose the Nginx deployment

```
$ cat << EOF > nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
```

```
  selector:
    app: nginx
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 80
  externalIPs:
    - 1.2.4.114
EOF
```

And expose httpd deployment

```
$ cat << EOF > httpd-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: httpd-service
spec:
  selector:
    app: httpd
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 80
  externalIPs:
    - 1.2.4.120
EOF
```

Kubectl them

```
$ kubectl create -f nginx-service.yaml
$ kubectl create -f httpd-service.yaml
```

Now your Kubernetes services should look like this

```
$ kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
kubernetes      ClusterIP   10.43.0.1       <none>        443/TCP
18m
```

```
httpd-service   ClusterIP   10.43.240.149   1.2.4.120     80/TCP
32s
nginx-service   ClusterIP   10.43.13.149    1.2.4.114     80/TCP
26s
```

You might see the service type is ClusterIP here. I am not sure why it does not says External IP instead.

Step 4: Voila!

Let's curl the httpd service, and you should see the Apache default page response.

```
$ curl -i 1.2.4.120
HTTP/1.1 200 OK
Date: Fri, 20 Dec 2019 03:36:23 GMT
Server: Apache/2.4.41 (Unix) <------
Last-Modified: Mon, 11 Jun 2007 18:53:14 GMT
ETag: "2d-432a5e4a73a80"
Accept-Ranges: bytes
Content-Length: 45
Content-Type: text/html

<html><body><h1>It works!</h1></body></html>
```

Next, let's curl Nginx service, and you should see the Nginx default page response.

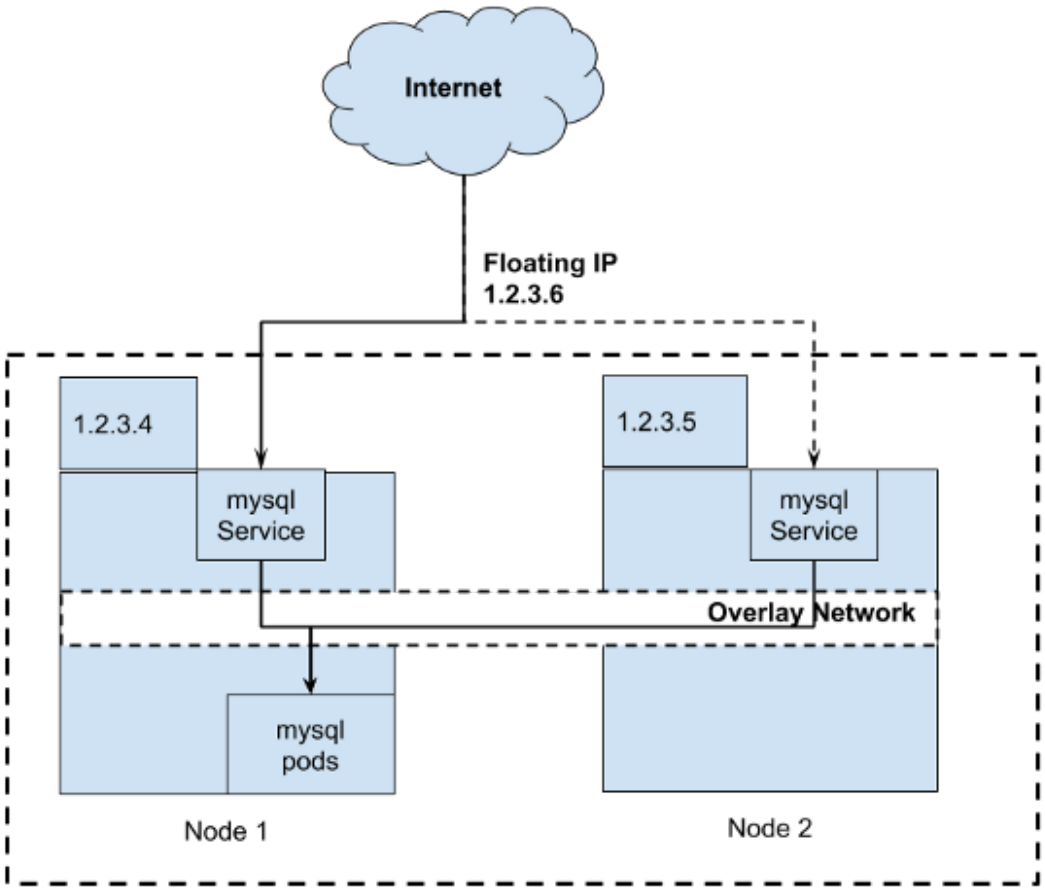
```
$ curl -i 1.2.4.114
HTTP/1.1 200 OK
Server: nginx/1.17.6 <------
Date: Fri, 20 Dec 2019 03:36:01 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 19 Nov 2019 12:50:08 GMT
Connection: keep-alive
ETag: "5dd3e500-264"
Accept-Ranges: bytes

<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
....
```

What's next

Floating IP dedicated to the service

Most cloud providers nowadays offer Floating IP service. Floating IP allows you to have 1 IP and assign that IP dynamically to any IP that you want. In this case, the IP can be assigned to any worker node in the Kubernetes cluster.

In DigitalOcean (I believe other providers also allow this), you can reassign the IP to other VM using an API call. This means that you can quickly reassign the IP to other VM proactively when they're down, or maybe rotate the IP periodically.

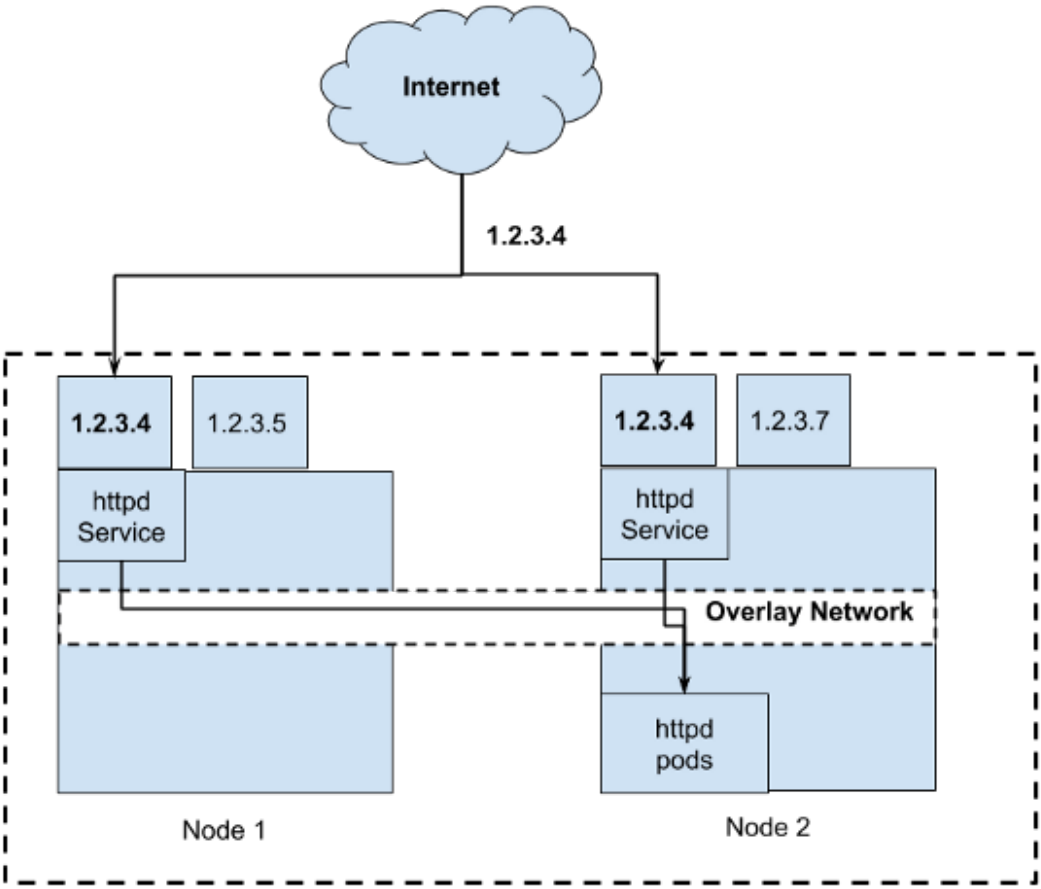


From the diagram, we can have 1 Floating IP 1.2.3.6 that is first assigned to Node 1 and will be switched to Node 2 when Node 1 becomes unavailable. IP 1.2.3.6 is meant for our MySQL service and place that in our application configurations.

I have not tried this set up yet, so I can't confirm it works. I will update the result in a future blog post.

Update 2020–01–03:
I have tried this setup. Unfortunately, I was unable to make this work on neither AWS nor DigitalOcean. The nodes did not respond to the Floating IP with the designated Kubernetes services. If you are able to make it run, feel free to update me.

Anycast IP



You could use Anycast IP as External IP so that they are highly available. For those who are not familiar with Anycast IP, it means that 1 IP may be routed to 2 or more servers. You can read more here. Personally, I am not sure how to set up this. However, I believe it is technically viable. I think this is the best way to run an External IP service.

Conclusion

There are many options that you can get an IP for the bare metal Kubernetes cluster. For example, you can use Inlets and Metal LB for that purpose. This setup might not be the best suit your organization needs. However, it is good to know how you can use this approach.

Disclaimer

I only use this for experimentation and testing, and this article is not meant for production use. Please consult your solution architect or CTO if you're planning to use this in production.

Kubernetes Service External Ip