

What's the difference between ClusterIP, NodePort and LoadBalancer service types in Kubernetes?

Asked 6 years, 2 months ago Modified 6 months ago Viewed 271k times



Question 1 - I'm reading the documentation and I'm slightly confused with the wording. It says:

493



ClusterIP: Exposes the service on a cluster-internal IP. Choosing this value makes the service only reachable from within the cluster. This is the default ServiceType

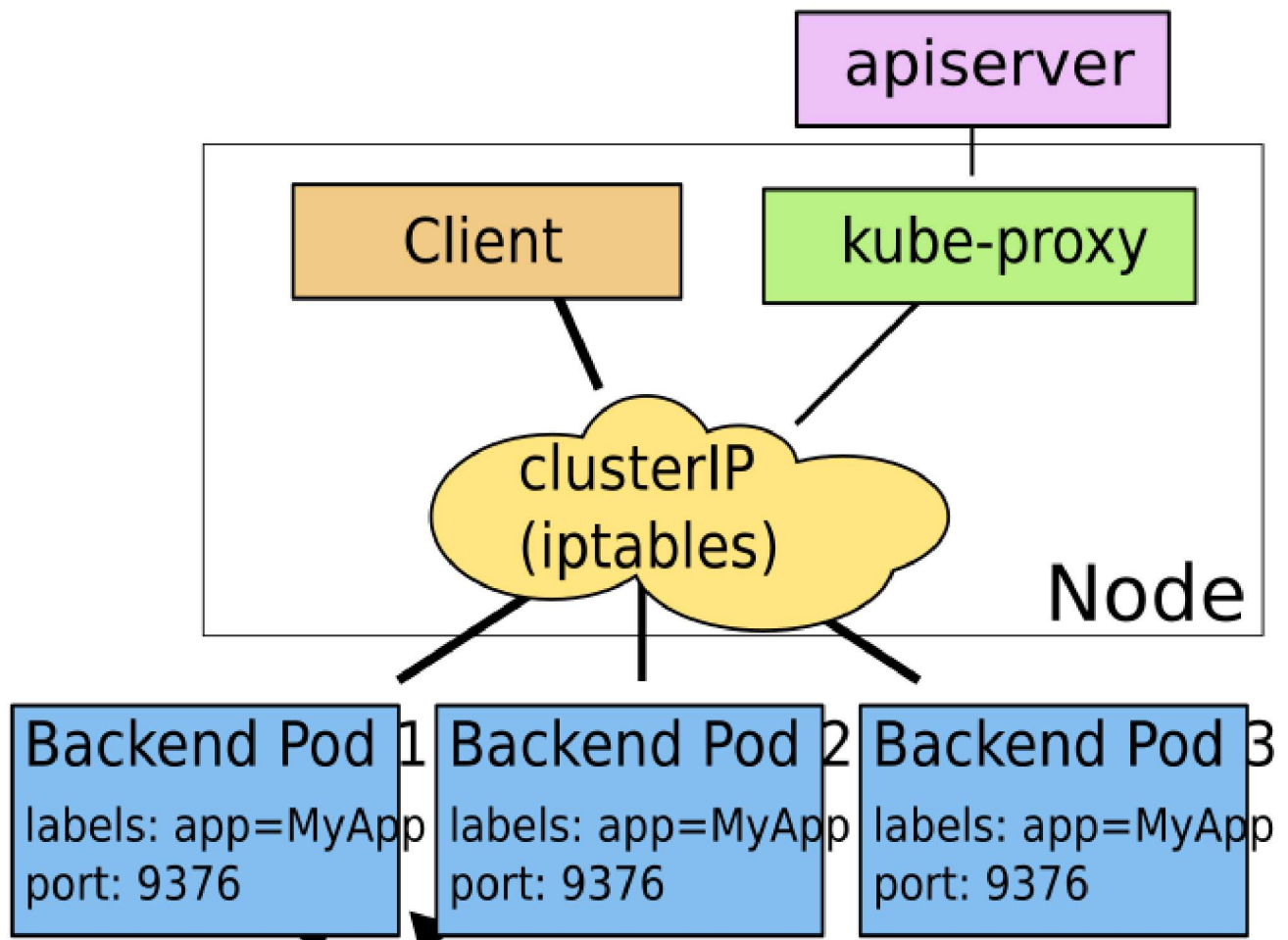
NodePort: Exposes the service on each Node's IP at a static port (the NodePort). A ClusterIP service, to which the NodePort service will route, is automatically created. You'll be able to contact the NodePort service, from outside the cluster, by requesting `<NodeIP>:<NodePort>` .

LoadBalancer: Exposes the service externally using a cloud provider's load balancer. NodePort and ClusterIP services, to which the external load balancer will route, are automatically created.

Does the NodePort service type still use the `ClusterIP` but just at a different port, which is open to external clients? So in this case is `<NodeIP>:<NodePort>` the same as `<ClusterIP>:<NodePort>` ?

Or is the `NodeIP` actually the IP found when you run `kubectl get nodes` and not the virtual IP used for the ClusterIP service type?

Question 2 - Also in the diagram from the link below:



Is there any particular reason why the `client` is inside the `Node`? I assumed it would need to be inside a `cluster` in the case of a `ClusterIP` service type?

If the same diagram was drawn for `NodePort`, would it be valid to draw the client completely outside both the `Node` and `cluster` or am I completely missing the point?

kubernetes containers kubernetes-service

Share Improve this question Follow

edited Jun 16, 2021 at 18:14

asked Jan 6, 2017 at 16:02



dahiya_boy

9,146 1 30 50



AmazingBergkamp

5,508 4 15 23

2 Some really great explanations here - shame no one added Ingress as another point of comparison.

– Paul Hodges Dec 8, 2022 at 22:05

1 explained.. including with ingress and external LBs, and nice diagrams: medium.com/google-cloud/...

– Apurva Singh Dec 23, 2022 at 20:55

Sorted by:

10 Answers

Highest score (default) 

A ClusterIP exposes the following:

440

- `spec.clusterIp:spec.ports[*].port`



You can only access this service while inside the cluster. It is accessible from its `spec.clusterIp` port. If a `spec.ports[*].targetPort` is set it will route from the port to the `targetPort`. The CLUSTER-IP you get when calling `kubectl get services` is the IP assigned to this service within the cluster internally.



A NodePort exposes the following:

- `<NodeIP>:spec.ports[*].nodePort`
- `spec.clusterIp:spec.ports[*].port`

If you access this service on a nodePort from the node's external IP, it will route the request to `spec.clusterIp:spec.ports[*].port`, which will in turn route it to your `spec.ports[*].targetPort`, if set. This service can also be accessed in the same way as ClusterIP.

Your NodeIPs are the external IP addresses of the nodes. You cannot access your service from `spec.clusterIp:spec.ports[*].nodePort`.

A LoadBalancer exposes the following:

- `spec.loadBalancerIp:spec.ports[*].port`
- `<NodeIP>:spec.ports[*].nodePort`
- `spec.clusterIp:spec.ports[*].port`

You can access this service from your load balancer's IP address, which routes your request to a nodePort, which in turn routes the request to the clusterIP port. You can access this service as you would a NodePort or a ClusterIP service as well.

Share Improve this answer Follow

edited Sep 12, 2020 at 19:26



sitaktif

1,494 13 15

answered Jan 6, 2017 at 17:03



kellanburket

11.9k 3 44 70



227



To clarify for anyone who is looking for what is the difference between the 3 on a simpler level. You can expose your service with minimal ClusterIp (within k8s cluster) or larger exposure with NodePort (within cluster external to k8s cluster) or LoadBalancer (external world or whatever you defined in your LB).

ClusterIp exposure < NodePort exposure < LoadBalancer exposure



- **ClusterIp**

Expose service through **k8s cluster** with `ip/name:port`

- **NodePort**

Expose service through **Internal network VM's** also external to k8s `ip/name:port`

- **LoadBalancer**

Expose service through **External world** or whatever you defined in your LB.

Share Improve this answer Follow

edited Apr 2, 2020 at 23:04

answered Jan 16, 2018 at 12:51



200_success

7,204 1 43 73



Tomer Ben David

7,983 1 42 24



111



ClusterIP: Services are reachable by pods/services in the Cluster

If I make a service called myservice in the default namespace of type: ClusterIP then the following predictable static DNS address for the service will be created:

myservice.default.svc.cluster.local (or just myservice.default, or by pods in the default namespace just "myservice" will work)

And that DNS name can only be resolved by pods and services inside the cluster.

NodePort: Services are reachable by clients on the same LAN/clients who can ping the K8s Host Nodes (and pods/services in the cluster) (Note for security your k8s host nodes should be on a private subnet, thus clients on the internet won't be able to reach this service)

If I make a service called mynodeportservice in the mynamespace namespace of type: NodePort on a 3 Node Kubernetes Cluster. Then a Service of type: ClusterIP will be created and it'll be reachable by clients inside the cluster at the following predictable static DNS address:

mynodeportservice.mynamespace.svc.cluster.local (or just mynodeportservice.mynamespace)

For each port that mynodeportservice listens on a nodeport in the range of 30000 - 32767 will be randomly chosen. So that External clients that are outside the cluster can hit that ClusterIP service that exists inside the cluster. Lets say that our 3 K8s host nodes have IPs 10.10.10.1, 10.10.10.2, 10.10.10.3, the Kubernetes service is listening on port 80, and the Nodeport picked at random was 31852.

A client that exists outside of the cluster could visit 10.10.10.1:31852, 10.10.10.2:31852, or 10.10.10.3:31852 (as NodePort is listened for by every Kubernetes Host Node) Kubeproxy will forward the request to mynodeportservice's port 80.

LoadBalancer: Services are reachable by everyone connected to the internet* (Common architecture is L4 LB is publicly accessible on the internet by putting it in a DMZ or giving it both a private and public IP and k8s host nodes are on a private subnet)

(Note: This is the only service type that doesn't work in 100% of Kubernetes implementations, like bare metal Kubernetes, it works when Kubernetes has cloud provider integrations.)

If you make mylb service, then a L4 LB VM will be spawned (a cluster IP service, and a NodePort Service will be implicitly spawned as well). This time our NodePort is 30222. the idea is that the L4 LB will have a public IP of 1.2.3.4 and it will load balance and forward traffic to the 3 K8s host nodes that have private IP addresses. (10.10.10.1:30222, 10.10.10.2:30222, 10.10.10.3:30222) and then Kube Proxy will forward it to the service of type ClusterIP that exists inside the cluster.

You also asked: Does the NodePort service type still use the ClusterIP? Yes*

Or is the NodeIP actually the IP found when you run `kubectl get nodes`? Also Yes*

Lets draw a parrallel between Fundamentals:

A container is inside a pod. a pod is inside a replicaset. a replicaset is inside a deployment.

Well similarly:

A ClusterIP Service is part of a NodePort Service. A NodePort Service is Part of a Load Balancer Service.

In that diagram you showed, the Client would be a pod inside the cluster.

Share Improve this answer Follow

edited Jan 24, 2019 at 2:01

answered Sep 9, 2018 at 4:05



neokyle

3,879 1 28 32



Lets assume you created a Ubuntu VM on your local machine. It's IP address is **192.168.1.104**.

94

You login into VM, and installed Kubernetes. Then you created a pod where nginx image running on it.



1- If you want to access this nginx pod inside your VM, you will create a **ClusterIP** bound to that pod for example:



```
$ kubectl expose deployment nginxapp --name=nginxclusterip --port=80 --target-port=8080
```

Then on your browser you can type ip address of nginxclusterip with port 80, like:

<http://10.152.183.2:80>

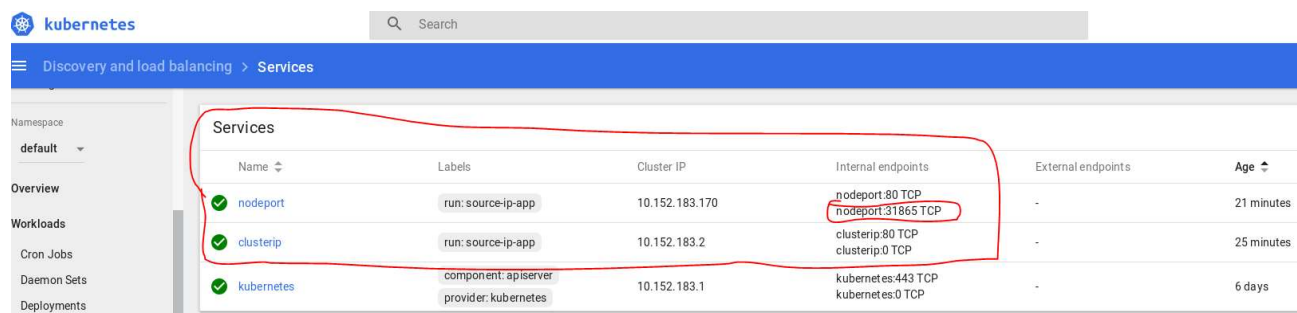
2- If you want to access this nginx pod from your host machine, you will need to expose your deployment with **NodePort**. For example:

```
$ kubectl expose deployment nginxapp --name=nginxnodeport --port=80 --target-port=8080 --type=NodePort
```

Now from your host machine you can access to nginx like:

<http://192.168.1.104:31865/>

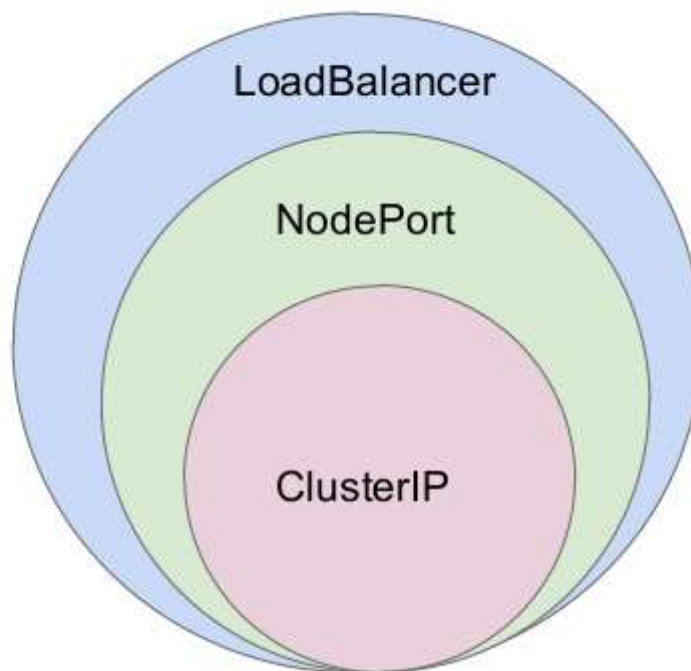
In my dashboard they appear as:



Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
nodeport	run: source-ip-app	10.152.183.170	nodeport:80 TCP nodeport:31865 TCP	-	21 minutes
clusterip	run: source-ip-app	10.152.183.2	clusterip:80 TCP clusterip:0 TCP	-	25 minutes
kubernetes	component: apiserver provider: kubernetes	10.152.183.1	kubernetes:443 TCP kubernetes:0 TCP	-	6 days

Below is a diagram shows basic relationship.

services



Share Improve this answer Follow

answered Jan 2, 2019 at 5:10



Teoman shipahi

46.9k 14 132 154



Feature	ClusterIP	NodePort	LoadBalancer
Exposition	Exposes the Service on an internal IP in the cluster.	Exposing services to external clients	Exposing services to external clients
Cluster	This type makes the Service only reachable from within the cluster	A NodePort service, each cluster node opens a port on the node itself (hence the name) and redirects traffic received on that port to the underlying service.	A LoadBalancer service accessible through a dedicated load balancer, provisioned from the cloud infrastructure Kubernetes is running on
Accessibility	It is default service and Internal clients send requests to a stable internal IP address.	The service is accessible at the internal cluster IP-port, and also through a dedicated port on all nodes.	Clients connect to the service through the load balancer's IP.
Yaml Config	type: ClusterIP	type: NodePort	type: LoadBalancer
Port Range	Any public ip form Cluster	30000 - 32767	Any public ip form Cluster
User Cases	For internal communication	Best for testing public or private access or providing access for a small amount of time.	widely used For External communication


Sources:

- [Kubernetes in Action](#)
- [Kubernetes.io Services](#)
- [Kubernetes Services simply visually explained](#)

Share Improve this answer Follow

edited Sep 8, 2022 at 7:23

answered Jun 11, 2021 at 4:07

 **Gupta**
8,256 4 46 58



1. clusterIP : IP accessible inside cluster (across nodes within d cluster).

```
nodeA : pod1 => clusterIP1, pod2 => clusterIP2
nodeB : pod3 => clusterIP3.
```

pod3 can talk to pod1 via their clusterIP network.

- nodeport : to make pods accessible from outside the cluster via nodeIP:nodeport, it will create/keep clusterIP above as its clusterIP network.

```
nodeA => nodeIPA : nodeportX
nodeB => nodeIPB : nodeportX
```

you might access service on pod1 either via nodeIPA:nodeportX OR nodeIPB:nodeportX. Either way will work because kube-proxy (which is installed in each node) will receive your request and distribute it [redirect it(iptables term)] across nodes using clusterIP network.

3. Load balancer

basically just putting LB in front, so that inbound traffic is distributed to nodeIPA:nodeportX and nodeIPB:nodeportX then continue with the process flow number 2 above.

Share Improve this answer Follow

answered Jan 23, 2019 at 8:27



system programmer

391 3 7

Practical understanding.

16

I have created 2 services 1 for **NodePort** and other for **ClusterIP**

```
[root@master ~]# kubectl get svc
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes           ClusterIP   10.96.0.1     <none>         443/TCP          24h
nginx-svc-cluster-ip ClusterIP   10.96.128.109 <none>         80/TCP           6m34s
nginx-svc-node-port  NodePort    10.103.109.253 <none>         80:30030/TCP     5m12s
[root@master ~]#
```

If I wanted to access the service inside the cluster(from master or any worker node) than both are accessible.


```
[root@master ~]# curl -I 10.96.128.109
HTTP/1.1 200 OK
Server: nginx/1.21.0
Date: Wed, 16 Jun 2021 18:38:04 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 25 May 2021 12:28:56 GMT
Connection: keep-alive
ETag: "60aced88-264"
Accept-Ranges: bytes

[root@master ~]#
[root@master ~]# curl -I 10.103.109.253
HTTP/1.1 200 OK
Server: nginx/1.21.0
Date: Wed, 16 Jun 2021 18:38:14 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 25 May 2021 12:28:56 GMT
Connection: keep-alive
ETag: "60aced88-264"
Accept-Ranges: bytes

[root@master ~]#
```

Now if I wanted to access the services from outside the cluster then **Nodeport** only accessible not **ClusterIP**.

```
[root@master ~]# curl -I localhost:80
curl: (7) Failed connect to localhost:80; Connection refused
[root@master ~]#
[root@master ~]# curl -I localhost:30030
HTTP/1.1 200 OK
Server: nginx/1.21.0
Date: Wed, 16 Jun 2021 18:42:58 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 25 May 2021 12:28:56 GMT
Connection: keep-alive
ETag: "60aced88-264"
Accept-Ranges: bytes

[root@master ~]#
```

Here you can see localhost wont listening on port 80 even my nginx container are listening on port 80.

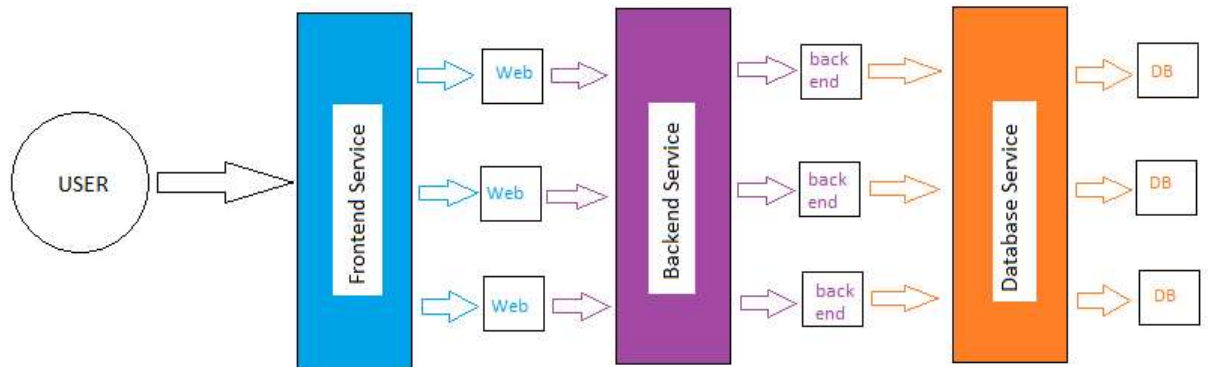
Yes, this is the only difference.

- **ClusterIP**. Exposes a service which is only accessible from within the cluster.
- **NodePort**. Exposes a service via a static port on each node's IP.
- **LoadBalancer**. Exposes the service via the cloud provider's load balancer.

- **ExternalName.** Maps a service to a predefined externalName field by returning a value for the CNAME record.

Practical Use Case

Let be assume you have to create below architecture in your cluster. I guess its pretty common.



Now, user only going to communicate with frontend on some port. Backend and DB services are always hidden to the external world.

Share Improve this answer Follow

edited Jan 2, 2022 at 9:59



G.S

10.1k

7

35

52

answered Jun 16, 2021 at 19:09



dahiya_boy

9,146

1

30

50

- Summary:

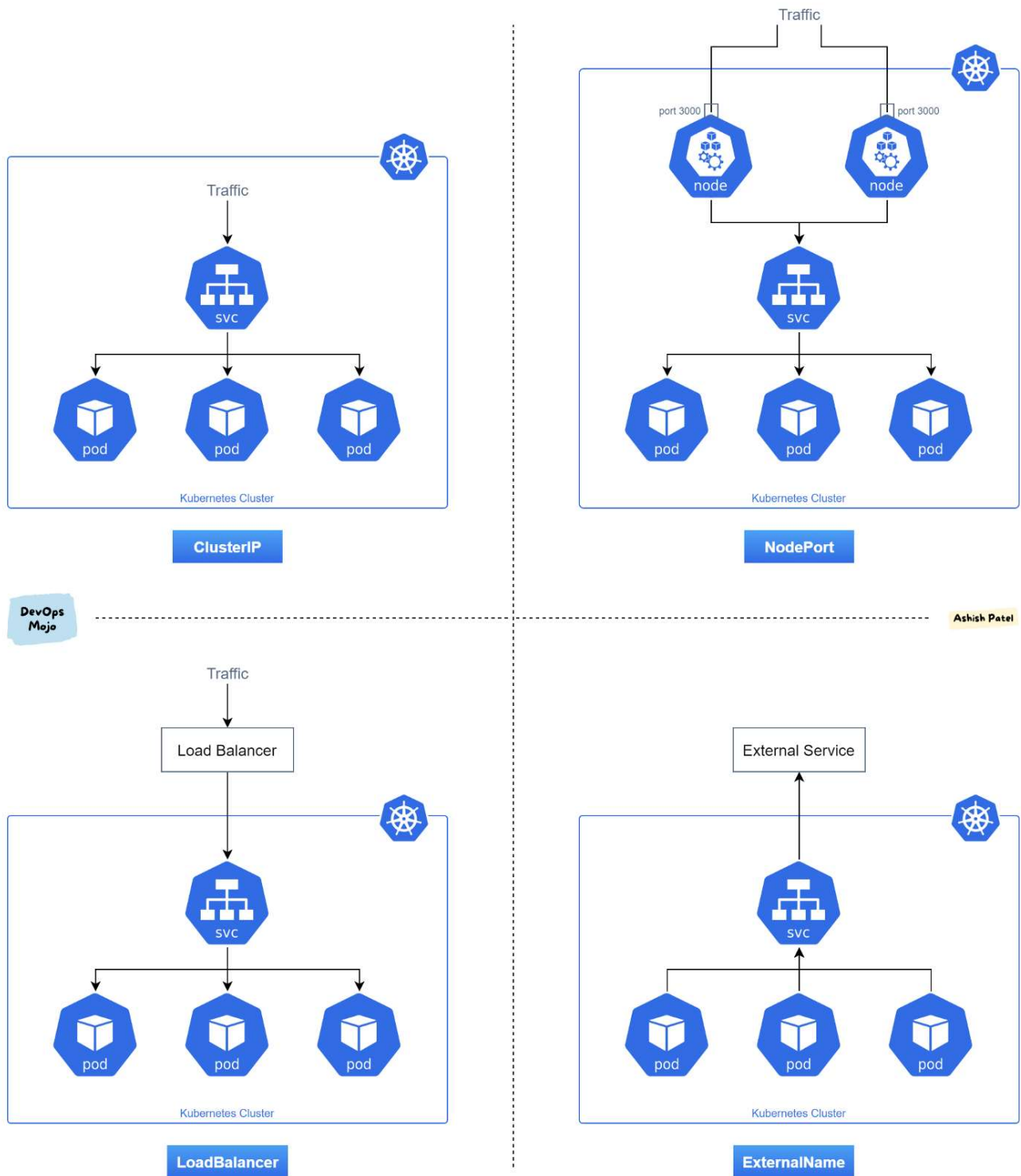
- [There are five types of Services:](#)

- **ClusterIP (default):** Internal clients send requests to a stable internal IP address.
- **NodePort:** Clients send requests to the IP address of a node on one or more nodePort values that are specified by the Service.
- **LoadBalancer:** Clients send requests to the IP address of a network load balancer.
- **ExternalName:** Internal clients use the DNS name of a Service as an alias for an external DNS name.
- **Headless:** You can use a headless service when you want a Pod grouping, but don't need a stable IP address.

The NodePort type is an extension of the ClusterIP type. So a Service of type NodePort has a cluster IP address.

The LoadBalancer type is an extension of the NodePort type. So a Service of type LoadBalancer has a cluster IP address and one or more nodePort values.

- Illustrate through [Image](#)



- [Details](#)
 - ClusterIP

- ClusterIP is the default and most common service type.
- Kubernetes will assign a cluster-internal IP address to ClusterIP service. This makes the service only reachable within the cluster.
- You cannot make requests to service (pods) from outside the cluster. You can optionally set cluster IP in the service definition file.
- Use Cases
 - Inter-service communication within the cluster. For example, communication between the front-end and back-end components of your app.
- NodePort
 - NodePort service is an extension of ClusterIP service. A ClusterIP Service, to which the NodePort Service routes, is automatically created.
 - It exposes the service outside of the cluster by adding a cluster-wide port on top of ClusterIP.
 - NodePort exposes the service on each Node's IP at a static port (the NodePort). Each node proxies that port into your Service. So, external traffic has access to fixed port on each Node. It means any request to your cluster on that port gets forwarded to the service.
 - You can contact the NodePort Service, from outside the cluster, by requesting :.
 - Node port must be in the range of 30000–32767. Manually allocating a port to the service is optional. If it is undefined, Kubernetes will automatically assign one.
 - If you are going to choose node port explicitly, ensure that the port was not already used by another service.
 - Use Cases
 - When you want to enable external connectivity to your service. Using a NodePort gives you the freedom to set up your own load balancing solution, to configure environments that are not fully supported by
 - Kubernetes, or even to expose one or more nodes' IPs directly. Prefer to place a load balancer above your nodes to avoid node failure.
- LoadBalancer
 - LoadBalancer service is an extension of NodePort service. NodePort and ClusterIP Services, to which the external load balancer routes, are automatically created.
 - It integrates NodePort with cloud-based load balancers.
 - It exposes the Service externally using a cloud provider's load balancer.
 - Each cloud provider (AWS, Azure, GCP, etc) has its own native load balancer implementation. The cloud provider will create a load balancer, which then automatically routes requests to your Kubernetes Service.

- Traffic from the external load balancer is directed at the backend Pods. The cloud provider decides how it is load balanced.
- The actual creation of the load balancer happens asynchronously.
- Every time you want to expose a service to the outside world, you have to create a new LoadBalancer and get an IP address.
- Use Cases
 - When you are using a cloud provider to host your Kubernetes cluster.
- ExternalName
 - Services of type ExternalName map a Service to a DNS name, not to a typical selector such as my-service.
 - You specify these Services with the `spec.externalName` parameter. It maps the Service to the contents of the externalName field (e.g. foo.bar.example.com), by returning a CNAME record with its value.
 - No proxying of any kind is established.
 - Use Cases
 - This is commonly used to create a service within Kubernetes to represent an external datastore like a database that runs externally to Kubernetes.
 - You can use that ExternalName service (as a local service) when Pods from one namespace talk to a service in another namespace.

Share Improve this answer Follow

answered Jul 15, 2022 at 3:51

undefined. [zangw](#)

41.7k

19

163

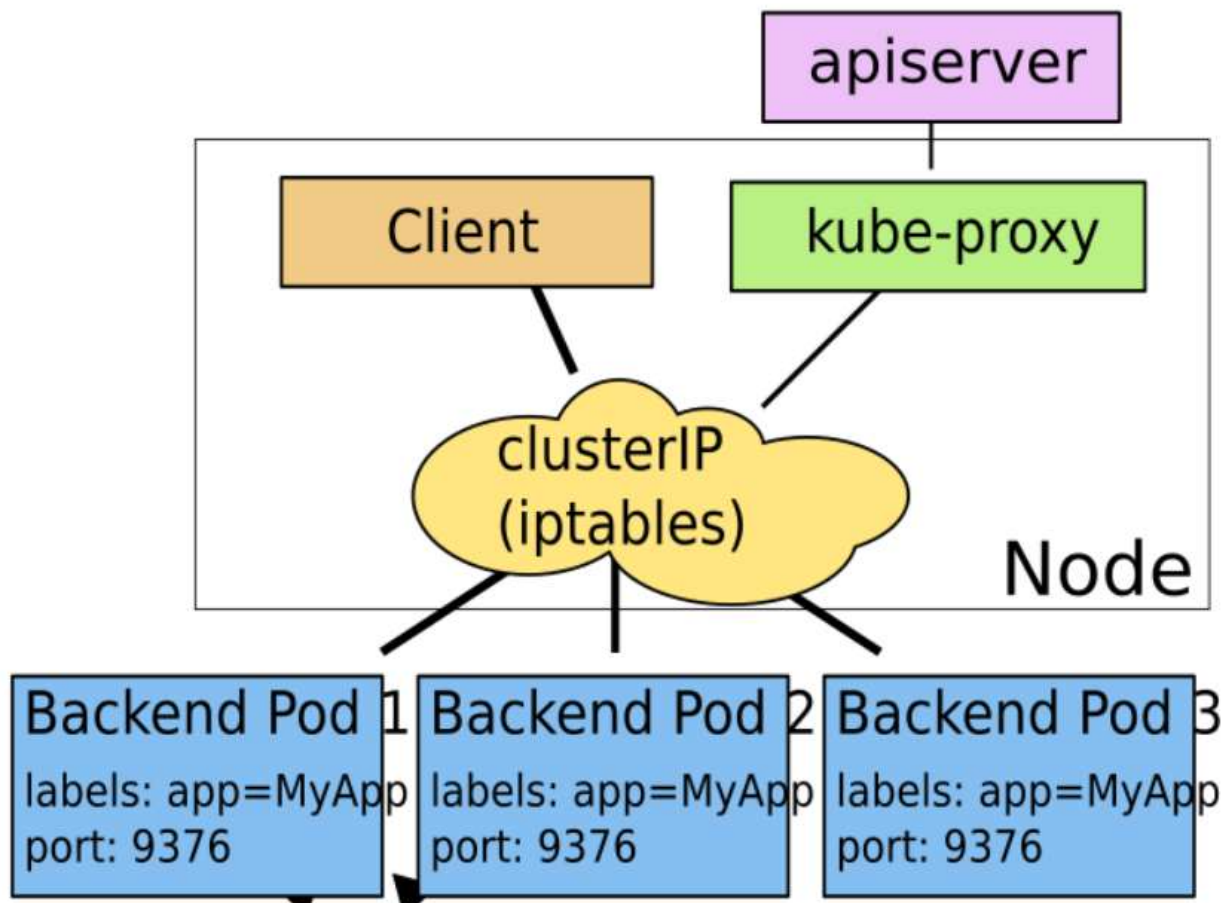
206



3



Here is the answer for the **Question 2** about the diagram, since it still doesn't seem to be answered directly:



Is there any particular reason why the Client is inside the Node? I assumed it would need to be inside a Cluster in the case of a ClusterIP service type?

At the diagram the Client is placed inside the Node to highlight the fact that ClusterIP is only accessible on a machine which has a running kube-proxy daemon. Kube-proxy is responsible for configuring iptables according to the data provided by apiserver (which is also visible at the diagram). So if you create a virtual machine and put it into the network where the Nodes of your cluster are and also properly configure networking on that machine so that individual cluster pods are accessible from there, even with that ClusterIP services will not be accessible from that VM, unless the VM has its iptables configured properly (which doesn't happen without kube-proxy running on that VM).

If the same diagram was drawn for NodePort, would it be valid to draw the client completely outside both the Node and Cluster or am I completely missing the point?

It would be valid to draw client outside the Node and Cluster, because NodePort is accessible from any machine which has access to a cluster Node and the corresponding port, including machines outside the cluster.



victorm1710

1,125 11 11



And do not forget the "new" service type ([from the k8s docu](#)):

1



ExternalName: Maps the Service to the contents of the externalName field (e.g. foo.bar.example.com), by returning a CNAME record with its value. No proxying of any kind is set up.



Note: You need either kube-dns version 1.7 or CoreDNS version 0.0.8 or higher to use the ExternalName type.

Share Improve this answer Follow

answered Sep 13, 2020 at 11:39



tal47

29 4