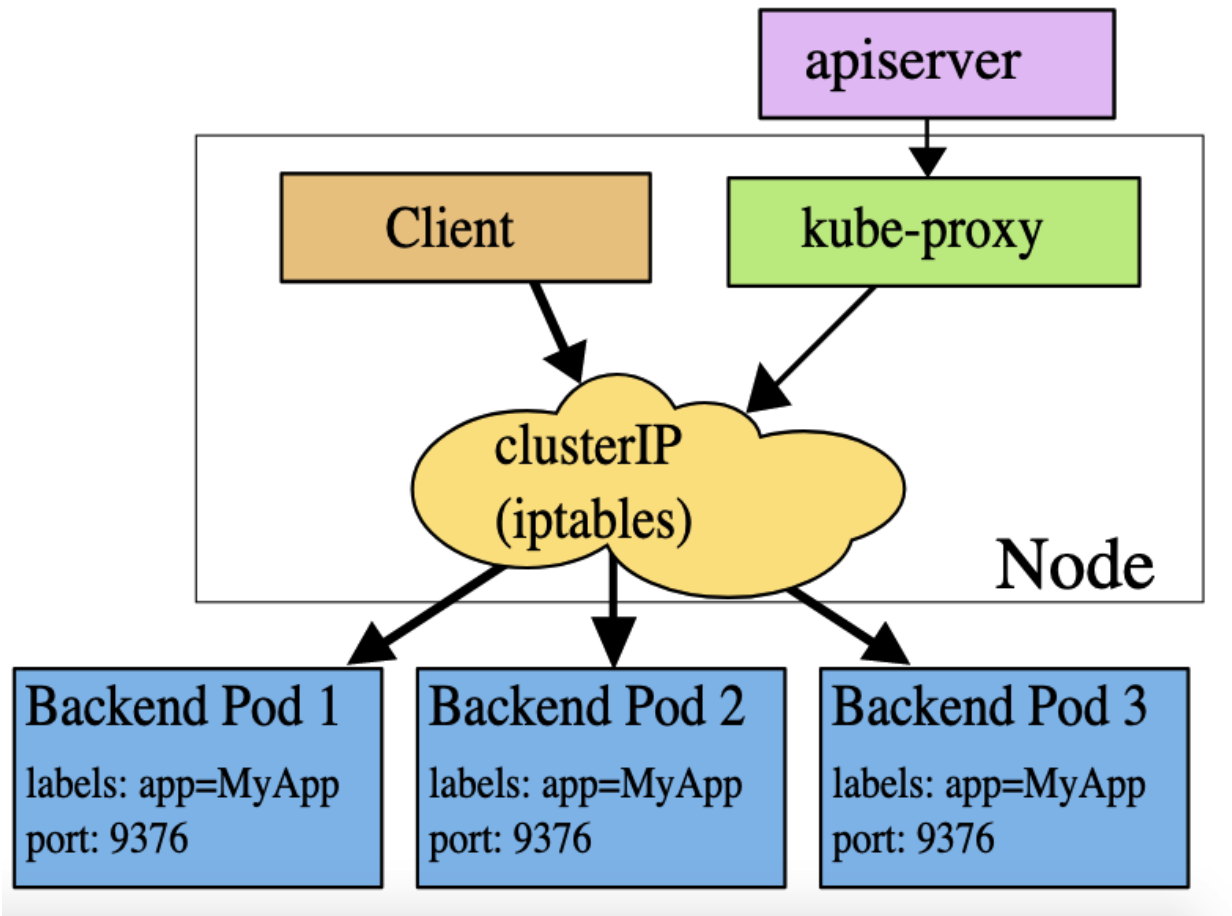
Kubernetes Headless service vs ClusterIP and traffic distribution


ismail yenigül Follow
Apr 21, 2019 · 3 min read



Default Kubernetes service type is `clusterIP` , When you create a headless service by setting clusterIP `None` , no load-balancing is done and no cluster IP is allocated for this service. Only DNS is automatically configured. When you run a DNS query for headless service, you will get the list of the Pods IPs and usually client dns chooses the first DNS record.

Lets do a real quick test for headless and clusterIP services. I will use `stenote/nginx-hostname` image to get hostname of the each pod in http response.

Lets create a nginx deployment

```
$ kubectl create deployment nginx --image=stenote/nginx-hostname
```

Scale to 3 pods.

```
$ kubectl scale --replicas=3 deployment nginx
```

Expose a headless service by setting `--cluster-ip=None`

```
$ kubectl expose deployment nginx --name nginxheadless --cluster-ip=None
service/nginxheadless exposed
```

Expose a standart service (ClusterIP type)

```
$ kubectl expose deployment nginx --name nginxclusterip --port=80  --target-port=80
service/nginxclusterip exposed
```

To test our case we need to run DNS queries and curl command. `arunvelsriram/utils` contains all the tool that we need.

```
$ kubectl run --generator=run-pod/v1 --rm utils -it --image arunvelsriram/utils bash

root@utils:/# host nginxheadless
nginxheadless.default.svc.cluster.local has address 100.64.10.148
nginxheadless.default.svc.cluster.local has address 100.64.10.206
nginxheadless.default.svc.cluster.local has address 100.64.2.87
```

As you can see above, `host nginxheadless` query returns Pods IP list in the response. Let's curl to this service name.

```
root@utils:/# for i in $(seq 1 10) ; do curl nginxheadless; done
nginx-66cf4d99b5-kpqgm
nginx-66cf4d99b5-kpqgm
nginx-66cf4d99b5-kpqgm
nginx-66cf4d99b5-kpqgm
```

```
nginx-66cf4d99b5-kpqgm
nginx-66cf4d99b5-kpqgm
nginx-66cf4d99b5-kpqgm
nginx-66cf4d99b5-kpqgm
nginx-66cf4d99b5-kpqgm
nginx-66cf4d99b5-kpqgm
root@utils:/#

root@utils:/# curl -v nginxheadless
* Rebuilt URL to: nginxheadless/
* Trying 100.64.10.148…
* TCP_NODELAY set
* Connected to nginxheadless (100.64.10.148) port 80 (#0)
> GET / HTTP/1.1
> Host: nginxheadless
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx
< Date: Sat, 20 Apr 2019 19:57:23 GMT
< Content-Type: text/html
< Transfer-Encoding: chunked
< Connection: keep-alive
<
nginx-66cf4d99b5-kpqgm
* Connection #0 to host nginxheadless left intact
root@utils:/#
```

As you can see above, our client pod always connects to the first IP in dns response. there is no true load balancing here :(

Let's test `clusterIP` service

```
root@utils:/# host nginxclusterip
nginxclusterip.default.svc.cluster.local has address 10.100.65.120
root@utils:/#

root@utils:/# for i in $(seq 1 10) ; do curl nginxclusterip; done
nginx-66cf4d99b5-m7knq
nginx-66cf4d99b5-m7knq
nginx-66cf4d99b5-kpqgm
nginx-66cf4d99b5-m7knq
nginx-66cf4d99b5-m7knq
nginx-66cf4d99b5-sfw5h
nginx-66cf4d99b5-m7knq
nginx-66cf4d99b5-m7knq
nginx-66cf4d99b5-sfw5h
nginx-66cf4d99b5-m7knq
root@utils:/#
```

Perfect! `clusterIP` service creates a single cluster IP and distribute the traffic between pods.

If you are using a single pod like a database server(mysql,pgsql), you can use headless service. but if you are going to run multiple pods for a service, it is better to create `clusterIP` type kubernetes service

Ismail YENIGUL

Devops Engineer at Feedstock Inc.

Kubernetes Service Headless Docker