

Working With ClusterIP Service Type In Kubernetes

Working with services in Kubernetes Using ClusterIP



@pramodAIML

Mar 24 · 6 min read



Sign in to Medium with Google



yashas samaga

yashas224@gmail.com



yashas samaga

yashas2020.samaga@gmail.com

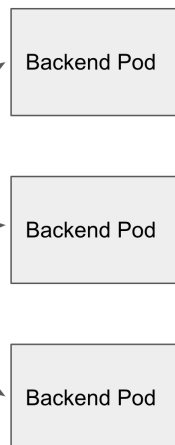


ClusterIP Service Type

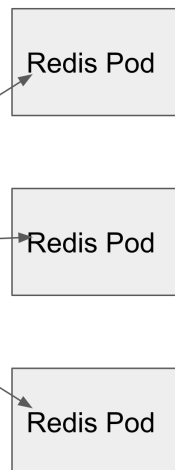
Front-End Pods



Backend Pods



Redis Pods



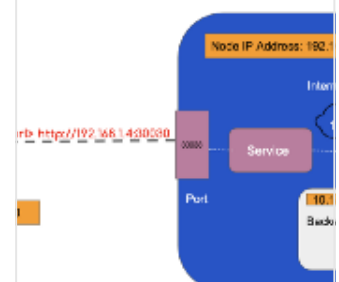
In continuation to my last piece on working with services :

Working With Services In Kubernetes

What is nodeport service in kubernetes? How NodePort Works ?

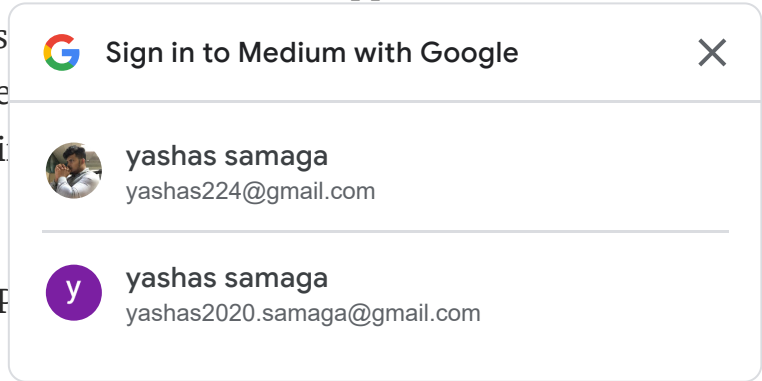
medium.com

Working With Services In Kubernetes



where we discussed how we can enable the external world application to talk to our internal pods deployed in the k8s cluster. Today I am excited to discuss Kubernetes role of **ClusterIP as one of the** promising multiple pods within the cluster.

But, before we jump into the ClusterIP Services in K8s briefly



What Are Services in K8s?

In Kubernetes, a Service is an abstraction that defines a logical set of Pods and a policy by which to access them, this kind of patterns is also sometimes called a micro-service.

If you are from a programming background and have been an API developer you must be familiar with the term REST object, services in k8s is quite similar to this REST object.

- One can **POST** a Service definition to the API server to create a new instance.
- The name of a Service object must be a valid **DNS label name**.

Each pod in the Kubernetes cluster has got the cluster IP and Network IP, but these pods cannot be directly accessed externally as those IPs are not exposed outside the cluster without a **Service**.

Kubernetes services allow our applications to be exposed to receive external world traffic, and also helps these pods lying in the node cluster to communicate with each other internally.

Types of Kubernetes Service:

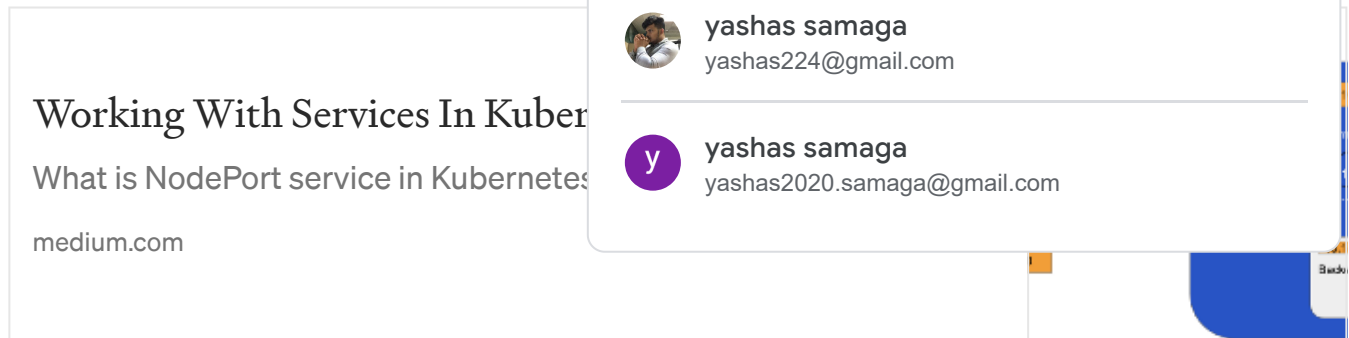
Kubernetes Services allows external connection to the internal cluster Pods and also manages internal communication among the Pods within the cluster via different Services **type** in defined in the **ServiceSpec**.

These service types are

- **ClusterIP**
- **NodePort**

- **LoadBalancer**

NodePort we have already covered in



Let me today help you all, understand **ClusterIP** type and its implementation using YAML file definition and Kubectl commands.

What Is ClusterIP?

A Society Analogy :

If you have lived in the gated society in the metro cities, you would have experienced that this society comprises multiple high-rise towers with multiple flats belonging to individual towers. Multiple families reside within each flat of the given tower. Imagine each tower having its own common address which in turn has a gateway to multiple flats with their own unique address, so if one has to reach out to a particular flat in tower B, it has to first locate the tower B gate address and then will be getting the access of local flat address lying within the tower B.

The address of tower A & Tower B can be considered similar to **ClusterIP**, this cluster IP's of each tower acts like a single endpoint/ gateway (Service) to connect to each pod (Family) living in the given cluster having their own local IP addresses.

Let's Understand ClusterIP Further With One More Example:

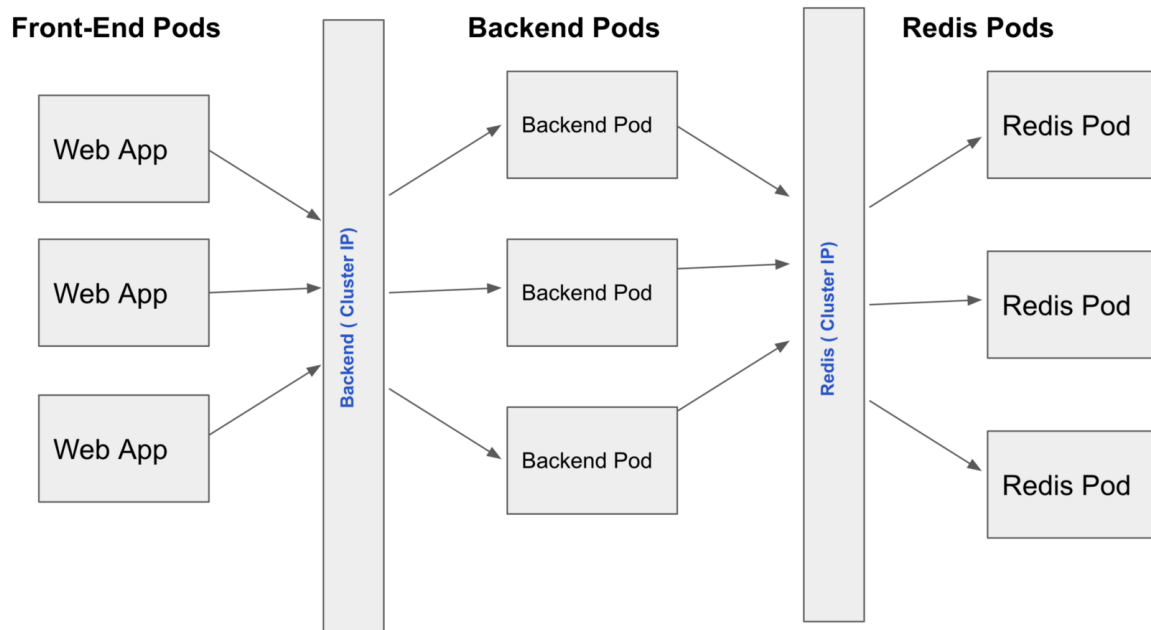
In any given cluster node there can be multiple types of Pods viz

- **Front-end pod**
- **Backend pods**
- **Redis pods**
- **Database MySQL pods**

etc...

Each of these types of **Pods** lying within an internal cluster will have a different internal network IP, which is liable to example, the front-end pod may be talking to Redis pods, the efficient mechanism, **ClusterIP** is our

Refer to the image below:



As can be clearly seen that whenever a front-end pod wants to communicate with backend pods, it has to simply communicate to a ClusterIP service named **backend**, which is a single endpoint service managing the communication to all backend pods. Similarly, if backend pods need to access the Redis cache service it has to make a service call to cluster IP service named **Redis**, which will allow the backend pods to communicate to respective Redis pods.

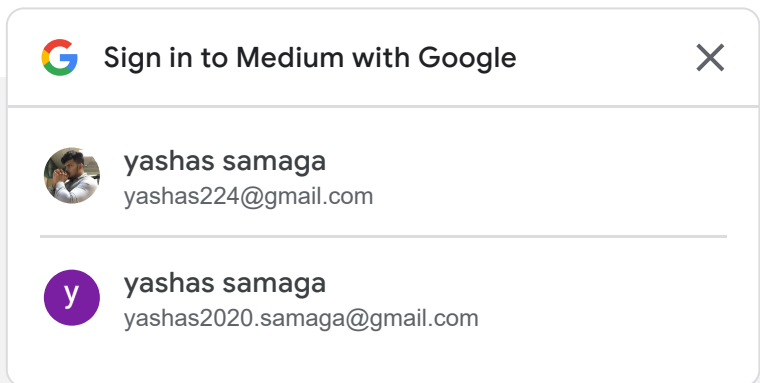
To summarize: These common backend / Redis services are the **virtual IP's** also known as **ClusterIP type service**, which effectively allows one pod type to communicate to another Pod type in the given cluster Node.

Defining ClusterIP Service Type: (YAML)

Let's put **ClusterIP** to work, by defining the sample ClusterIP service YAML file:

my-cluster-ip-demo.yaml file:

```
apiVersion: v1
kind: Service
metadata:
  name: Backend
spec:
  type: ClusterIP
  ports:
    - targetPort: 80
      port: 80
```



Let's Define A Sample Front-End Pod:

And bind it to our ClusterIP service defined in **my-cluster-ip-demo.yaml** file

my-demo-pod.yaml file :

```
apiVersion: v1
kind: Pod
metadata:
  name: my-demo-pod
  labels:
    app: my-test-pod
    type: mobile-front-end-app
spec:
  containers:
    - name: nginx-container
      image: nginx
```

Now we have a pod, with the label: my-demo-pod, let's map this label to our service definition file to create our ClusterIP services, in order to do so we will have to define a new parameter called **selector**, in the **spec section**, and pull the label parameter and place it under **selector** field as shown below

Backend Service: (ClusterIP type)

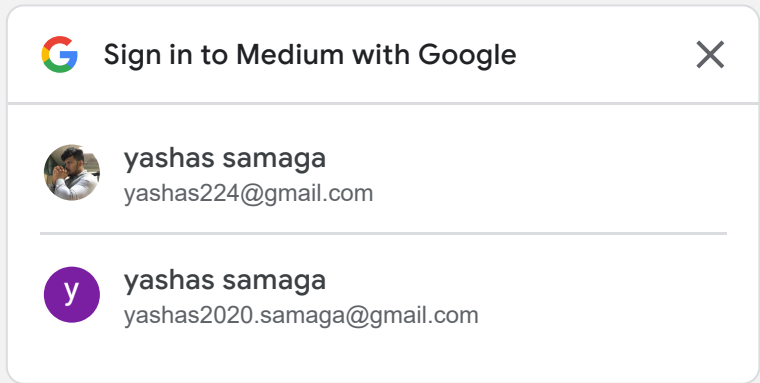
my-cluster-ip-demo.yaml file:

```
apiVersion: v1
kind: Service
```

```
metadata:
  name: Backend
spec:
  type: ClusterIP
  ports:
    - targetPort: 80
      port: 80

  selector:

    name: my-demo-pod
    type: front-end-app
```



Now that we have defined all the required YAML files it is time to create and compile those YAML file using **kubectl** command

So let's first create a pod file as defined in our, **my-demo-pod.yaml** file above

Open your Kubernetes cluster terminal : (minikube cluster in case you are on the local machine)

- Create **my-demo-pod.yaml** using **vim** command
- Copy-paste the YAML code defined **my-demo-pod.yaml** file
- create the pod using the command below

```
$ kubectl apply -f test-pod.yaml
```

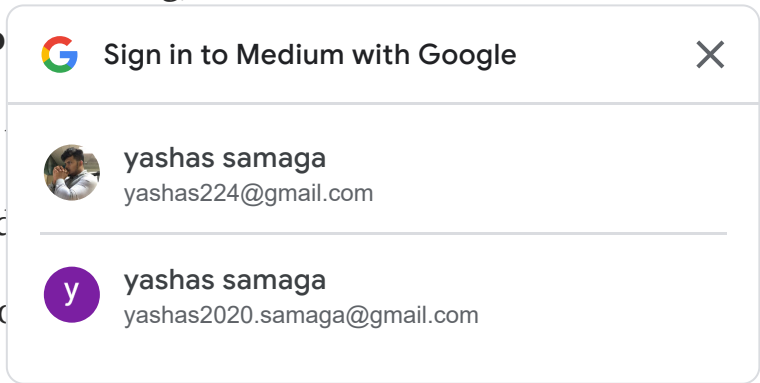
Output:

Our pod file my-pod-demo has been created and running



Now that we have our pod file up and running, it's time to create service as defined in our "my-cluster-ip-demo"

- create **my-cluster-ip-demo.yaml**
- copy paste the YAML code defined
- create the service pod using the c



```
$ kubectl create -f my-cluster-ip-demo.yaml
```

Output:

```
root@controlplane:~# vim my-cluster-ip-demo.yaml
root@controlplane:~# kubectl apply -f my-cluster-ip-demo.yaml
service/backend created
root@controlplane:~# kubectl get svc
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP  PORT(S)    AGE
backend         ClusterIP      10.110.68.224   <none>       80/TCP     21s
kubernetes      ClusterIP      10.96.0.1       <none>       443/TCP    17m
root@controlplane:~#
```

Our service of type ClusterIP has been successfully created and has been bound to our demo using pod using selector as shown below :

```
$ kubectl describe svc backend
```

```
root@controlplane:~# kubectl describe svc backend
```

```
Name: backend
Namespace: default
Labels: <none>
Annotations: <none>
Selector: name=my-demo-pod
Type: ClusterIP
IP Families: <none>
IP: 10.110.68.224
IPs: 10.110.68.224
Port: <unset> 80/TCP
TargetPort: 80/TCP
Endpoints: <none>
Session Affinity: None
Events: <none>
root@controlplane:~# █
```



Sign in to Medium with Google



yashas samaga

yashas224@gmail.com



yashas samaga

yashas2020.samaga@gmail.com

What's Next?

We have covered NodePort & ClusterIP services type in detail so far and understood how one can define and create such services using YAML file definition and Kubectl CLI. But still we have not covered one more important K8s service type called “LoadBalancer”

We will look into this in our next piece of working with services in K8s , till then its time to sign-off with this

Food for Thought: #ChandrayanLogy

It will always be your desire to learn every single day something new from this life which will make your life worthy and fulfilling, so keep learning and keep growing and eventually you will find your true purpose to remain blissful no matter what

Thank you for reading and supporting my passion to write and share ...

Kubernetes

DevOps

Software Development

Programming

Microservices


[About](#) [Write](#) [Help](#) [Legal](#)

Get the Medium app

Download on the App Store

GET IT ON Google Play

Sign in to Medium with Google



yashas samaga
yashas224@gmail.com

y

yashas samaga
yashas2020.samaga@gmail.com