

Services

Problem Statement:

If load increases, we can create one more Pod. Who will create it ?

There are two ways: manual or replication controller (it will set the desire state for you automatically)

So, we cannot give all IP address of servers to the client, -> not user friendly.

We are going to create an intermediate program between client and pods. Say it has the IP (100) , now client comes to this ip 100 and the request is recreated and connect to respective port of backend servers.


This intermediate program is frontend of the backend servers and is also known as LOAD BALANCER.

Now challenge is, how LB will register as the new pod launches.

Since ip changes of system on each restart, so we have to tag the OS of label it.

LB will look for this Label and as soon as it finds it, that particular OS will get register under LB

Yml file to enable service.

 service - Notepad
File Edit Format View Help

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

```
metadata:
  name: lb
```

```
spec:
  selector:
    app: shreya
```

```
ports:
- targetPort: 80
  port: 8080
```

After writing the yml file, I have created the service using the above written file.

kubectl create -f service.yml

```
C:\Users\lenovo\Desktop\K8s>kubectl create -f service.yml
service/lb1 created
```

Using the command **kubectl get svc**, we get all the available services.

```
C:\Users\lenovo\Desktop\K8s>kubectl get svc
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes          ClusterIP     10.96.0.1     <none>         443/TCP          6d23h
lb                  ClusterIP     10.104.124.42 <none>         8080/TCP         18m
rc1                 NodePort      10.106.25.214 <none>         80:31381/TCP     4h52m
```

kubectl describe svc lb command is used to get details of the service named lb. Here we can see that there are not any end points available.

```
C:\Users\lenovo\Desktop\K8s>kubectl describe svc lb
Name: lb
Namespace: default
Labels: <none>
Annotations: <none>
Selector: app=shreya
Type: ClusterIP
IP Families: <none>
IP: 10.104.124.42
IPs: 10.104.124.42
Port: <unset> 8080/TCP
TargetPort: 80/TCP
Endpoints: <none>
Session Affinity: None
Events: <none>
```

Now, using the yml file to create pods, I have created a pod here. The pod named skpod2 has been created.

Command **kubectl create -f lbpod.yml**

```
C:\Users\lenovo\Desktop\K8s>kubectl create -f lbpod.yml
pod/skpod2 created
```

After launching the pod, check for available pods using **kubectl get pods**.

```
C:\Users\lenovo\Desktop\K8s>kubectl get pods -L app
NAME          READY   STATUS    RESTARTS   AGE   APP
rc1-jld2v     1/1     Running   2           5h    web
rc1-jwbvs     1/1     Running   2           5h    web
rc1-m985v     1/1     Running   2           5h    web
rc1-q8t2f     1/1     Running   2           5h    web
skpod1        1/1     Running   2           5h15m web
skpod2        1/1     Running   0           2m10s shreya
```

After the pod is launched, we describe it using `kubectl describe svc lb`. One new IP has been added to the end point.

```
C:\Users\lenovo\Desktop\K8s>kubectl describe svc lb
Name: lb
Namespace: default
Labels: <none>
Annotations: <none>
Selector: app=shreya
Type: ClusterIP
IP Families: <none>
IP: 10.104.124.42
IPs: 10.104.124.42
Port: <unset> 8080/TCP
TargetPort: 80/TCP
Endpoints: 172.17.0.10:80
Session Affinity: None
Events: <none>
```

Now we launch one more pod named skpod3 and launch it using the yml file.

```
lbpod - Notepad
File Edit Format View Help
apiVersion: v1
kind: Pod
metadata:
  name: "skpod3"
  labels:
    app: shreya

spec:
  containers:
  - name: "container1"
    image: "vimal13/apache-webserver-php"
```

On describing the service, we see that one more ip has been added in the end point.

```
C:\Users\lenovo\Desktop\K8s>kubectl create -f lbpod.yml
pod/skpod3 created

C:\Users\lenovo\Desktop\K8s>kubectl describe svc lb
Name: lb
Namespace: default
Labels: <none>
Annotations: <none>
Selector: app=shreya
Type: ClusterIP
IP Families: <none>
IP: 10.104.124.42
IPs: 10.104.124.42
Port: <unset> 8080/TCP
TargetPort: 80/TCP
Endpoints: 172.17.0.10:80,172.17.0.11:80
Session Affinity: None
Events: <none>
```

Using the command **kubectl get svc**, we find the ip and port number of the process.

```
C:\Users\lenovo\Desktop\K8s>kubectl get svc
NAME         TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes   ClusterIP     10.96.0.1     <none>         443/TCP          6d23h
lb           ClusterIP     10.104.124.42 <none>         8080/TCP         42m
rc1          NodePort      10.106.25.214 <none>         80:31381/TCP     5h15m
```

If we use curl command on windows command prompt, we won't be able to access it. However, on VM, it will work.

```
$ curl 10.104.124.42:8080
<body bgcolor='aqua'>
<pre>

welcome to vimal web server for testingeth0: flags=4163<UP,BROADCAST,RUNNING,MUL
TICAST>  mtu 1500
    inet 172.17.0.11  netmask 255.255.0.0  broadcast 172.17.255.255
    ether 02:42:ac:11:00:0b  txqueuelen 0  (Ethernet)
    RX packets 73  bytes 5802 (5.6 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 47  bytes 14053 (13.7 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    loop txqueuelen 1000  (Local Loopback)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

</pre>
$
```

For making it work on windows command prompt, we make the following changes in the service yml file



service2 - Notepad

File Edit Format View Help

apiVersion: v1

kind: Service

metadata:

name: lb

spec:

type: NodePort

selector:

app: shreya

ports:

- targetPort: 80

port: 8080

nodePort: 30000

Now, using the command `kubectl apply -f service2.yml`, we can see the port number that we have just assigned in yml file using nodePort.

```
C:\Users\lenovo\Desktop\K8s>kubectl apply -f service2.yml
Warning: resource services/lb is missing the kubect1.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
service/lb configured

C:\Users\lenovo\Desktop\K8s>kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP          7d
lb           NodePort    10.104.124.42 <none>        8080:30000/TCP   73m
rc1          NodePort    10.106.25.214 <none>        80:31381/TCP     5h47m

C:\Users\lenovo\Desktop\K8s>
```

Command `kubectl describe svc lb` shows that Type has been changed from clusterIP to NodePort.

```
C:\Users\lenovo\Desktop\K8s>kubectl describe svc lb
Name:         lb
Namespace:    default
Labels:       <none>
Annotations:  <none>
Selector:     app=shreya
Type:         NodePort
IP Families:  <none>
IP:           10.104.124.42
IPs:          10.104.124.42
Port:         <unset> 8080/TCP
TargetPort:   80/TCP
NodePort:     <unset> 30000/TCP
Endpoints:    172.17.0.10:80,172.17.0.11:80
Session Affinity: None
External Traffic Policy: Cluster
Events:       <none>
```

It means that, now, we can access the pods using curl command on windows command prompt too.

curl ipOfK8s:provided_port_number

```
Command Prompt

C:\Users\lenovo\Desktop\K8s>curl 192.168.99.101:30000
<body bgcolor='aqua'>
<pre>

welcome to vimal web server for testingeth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.11 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:ac:11:00:0b txqueuelen 0 (Ethernet)
    RX packets 79 bytes 6258 (6.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 51 bytes 15296 (14.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

</pre>

C:\Users\lenovo\Desktop\K8s>curl 192.168.99.101:30000
<body bgcolor='aqua'>
<pre>

welcome to vimal web server for testingeth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.10 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:ac:11:00:0a txqueuelen 0 (Ethernet)
    RX packets 68 bytes 5352 (5.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 43 bytes 12770 (12.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Each time I type the command, I get different IP which is any one of the two provided IP.

Even after the IP is changing, we can still access the page.