# Service ClusterIP

1. Familiar with the deployment services/dep-mypython.yaml.

Run and investigate the output:

$ docker run --rm -p 8000:8000 -d docker.io/burrsutter/mypython:1.0.0  
24f8067968c8d49b3cce47715738071f54423b2e6cb6416ecb217f6e37d9e7eb  
  
$ curl localhost:8000  
Python Hello on 24f8067968c8  
  
$ docker stop $(docker ps -lq)  
24f8067968c8

1. Create the deployment services/dep-mypython.yaml.

Get IP addresses of the Pod using selector app=mypython:

$ kubectl get pods -o wide -l app=mypython

1. Create a temporary pod:

$ kubectl run -it --rm --image=yauritux/busybox-curl busybox

Ping the deployment’s pod using IP address from step 2:

$ ping 172.17.0.3

Get content from deployment pod’s 8000 port:

$ curl 172.17.0.3:8000

Exit from the temporary pod (press Ctrl+D or execute the exit command).

1. Create a Service services/service-clusterip-single-port.yaml.

Check the service and the endpoint:

$ kubectl get services -o wide  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR  
myservice ClusterIP 10.96.53.88 <none> 80/TCP 31m inservice=myservice  
  
$ kubectl get endpoints  
NAME ENDPOINTS AGE  
myservice 172.17.0.3:8000 22m

Check detailed information:

* for the service by kubectl describe service myservice
* for the endpoint by kubectl describe endpoints myservice

1. Start curling the service:

$ kubectl run -it busybox --image=yauritux/busybox-curl --rm  
If you don't see a command prompt, try pressing enter.  
  
/home # while true; do curl myservice; sleep 1; done  
Python Hello on mypython-deployment-69c66864fc-8hsnc  
...

You will see the output of the only pod.

1. In the second terminal, scale the deployment to 2 replicas:

$ kubectl scale deploy mypython-deployment --replicas=2

1. Check IPs in the endpoint again (you should be two IPs):

$ kubectl get endpoints  
NAME ENDPOINTS AGE  
myservice 172.17.0.3:8000,172.17.0.5:8000 29m

1. Check the output of curl in the first terminal (it should contain the output of two pods):

...  
Python Hello on mypython-deployment-69c66864fc-8hsnc  
Python Hello on mypython-deployment-69c66864fc-ncqf9  
...

1. Create two new deployments services/dep-mygo.yaml and services/dep-mynode.yaml:

$ kubectl create -f services/dep-mygo.yaml  
$ kubectl create -f services/dep-mynode.yaml

1. Check the endpoints again. There should be new pods IPs:

$ kubectl get endpoints  
NAME ENDPOINTS AGE  
myservice 172.17.0.3:8000,172.17.0.5:8000,172.17.0.6:8000 + 1 more... 33m

1. Output of curl should contain output of new pods.
2. Don’t clean up.

## Solution

1. Familiar with the deployment services/dep-mypython.yaml.

Run and investigate the output:

$ docker run --rm -p 8000:8000 -d docker.io/burrsutter/mypython:1.0.0  
85eca0d2b43c5a983daafb0a95ea711df88df8536fb513afd667d2217e6d0bc6  
  
$ curl localhost:8000  
Python Hello on 85eca0d2b43c  
  
$ docker stop $(docker ps -lq)  
85eca0d2b43c

1. Create the deployment services/dep-mypython.yaml.

$ kubectl create -f services/dep-mypython.yaml  
deployment.apps/mypython-deployment created

Get IP addresses of the Pod using selector app=mypython:

$ kubectl get pods -o wide -l app=mypython  
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES  
mypython-deployment-69c66864fc-8hsnc 1/1 Running 0 6m23s 172.17.0.3 minikube <none> <none>

1. Create a temporary pod to ping the deployment’s pod using IP address from step 2:

$ kubectl run -it --rm --image=yauritux/busybox-curl busybox  
If you don't see a command prompt, try pressing enter.  
  
/home # ping 172.17.0.3  
PING 172.17.0.3 (172.17.0.3): 56 data bytes  
64 bytes from 172.17.0.3: seq=0 ttl=64 time=0.107 ms  
64 bytes from 172.17.0.3: seq=1 ttl=64 time=0.160 ms  
64 bytes from 172.17.0.3: seq=2 ttl=64 time=0.137 ms  
^C  
--- 172.17.0.3 ping statistics ---  
3 packets transmitted, 3 packets received, 0% packet loss  
round-trip min/avg/max = 0.107/0.134/0.160 ms  
  
/home # curl 172.17.0.3:8000  
Python Hello on mypython-deployment-69c66864fc-8hsnc  
  
/home # exit  
pod "busybox" deleted

1. Create a Service services/service-clusterip-single-port.yaml.

$ kubectl create -f services/service-clusterip-single-port.yaml  
service/myservice created

Check the service and the endpoint:

$ kubectl get services -o wide  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR  
myservice ClusterIP 10.96.53.88 <none> 80/TCP 55s inservice=myservice  
  
$ kubectl get endpoints  
NAME ENDPOINTS AGE  
myservice 172.17.0.3:8000 82s

Check detailed information:

* for the service by kubectl describe service myservice
* for the endpoint by kubectl describe endpoints myservice

$ kubectl describe service myservice  
Name: myservice  
Namespace: default  
Labels: <none>  
Annotations: <none>  
Selector: inservice=myservice  
Type: ClusterIP  
IP Families: <none>  
IP: 10.96.53.88  
IPs: 10.96.53.88  
Port: http 80/TCP  
TargetPort: 8000/TCP  
Endpoints: 172.17.0.3:8000  
Session Affinity: None  
Events: <none>  
  
$ kubectl describe endpoints myservice  
Name: myservice  
Namespace: default  
Labels: <none>  
Annotations: endpoints.kubernetes.io/last-change-trigger-time: 2021-06-13T04:34:50Z  
Subsets:  
 Addresses: 172.17.0.3  
 NotReadyAddresses: <none>  
 Ports:  
 Name Port Protocol  
 ---- ---- --------  
 http 8000 TCP  
  
Events: <none>

There is only pod’s IP here.

1. Start curling the service:

$ kubectl run -it busybox --image=yauritux/busybox-curl --rm  
If you don't see a command prompt, try pressing enter.  
  
/home # while true; do curl myservice; sleep 1; done  
Python Hello on mypython-deployment-69c66864fc-8hsnc  
...

You will see the output of the only pod.

1. In the second terminal, scale the deployment to 2 replicas:

$ kubectl scale deploy mypython-deployment --replicas=2  
deployment.apps/mypython-deployment scaled

1. Check IPs in the endpoint again (you should be two IPs):

$ kubectl get endpoints  
NAME ENDPOINTS AGE  
myservice 172.17.0.3:8000,172.17.0.5:8000 29m

1. Check the output of curl in the first terminal (it should contain the output of two pods):

...  
Python Hello on mypython-deployment-69c66864fc-8hsnc  
Python Hello on mypython-deployment-69c66864fc-8hsnc  
Python Hello on mypython-deployment-69c66864fc-ncqf9  
Python Hello on mypython-deployment-69c66864fc-ncqf9  
Python Hello on mypython-deployment-69c66864fc-8hsnc  
...

1. Create two new deployments services/dep-mygo.yaml and services/dep-mynode.yaml.

$ kubectl create -f services/dep-mygo.yaml  
deployment.apps/mygo-deployment created  
  
$ kubectl create -f services/dep-mynode.yaml  
deployment.apps/mynode-deployment created

1. Check the endpoints again. There should be new pods IPs:

$ kubectl get endpoints  
NAME ENDPOINTS AGE  
myservice 172.17.0.3:8000,172.17.0.5:8000,172.17.0.6:8000 + 1 more... 33m

1. Output of curl should contain output of new pods:

...  
Go Hello on mygo-deployment-55894d8449-jvcc2  
Python Hello on mypython-deployment-69c66864fc-8hsnc  
Node Hello on mynode-deployment-5d58f6459-c7r5g 17  
Python Hello on mypython-deployment-69c66864fc-8hsnc  
Python Hello on mypython-deployment-69c66864fc-ncqf9  
...

1. Don’t clean up.