31.Headless服务

服务可以用于提供一个稳定的IP地址，它允许客户端连接到服务后面的pod。服务会将收到的每个连接转发给一个随机选择的pod。但是如果客户端需要连接到服务后的所有pod呢？如果pod本身也需要连接到其他所有pod呢？通过服务来连接很明显无法达到这个目的。
如果一个客户端希望连接到所有pod，它需要知道每个pod的IP地址。一个方式是让客户端通过一个API调用Kubernetes API服务器获取pod列表以及它们的IP地址。但是由于需要遵循应用程序与Kubernetes解耦原则，使用API服务显然是不合适的。
幸运的是Kubernetes允许客户端通过DNS查找发现pod的IP地址。通常，当执行某个服务的DNS查找时，DNS服务器会返回该服务的Cluster IP地址。我们可以在服务的spec中指定clusterIP字段为None，告诉Kubernetes不需要cluster IP，那么DNS服务器就会返回所有Pod的IP，而不是该服务的IP。
DNS服务器不会返回单个DNS A记录（A记录就是指定域名对应的IP地址），而是返回该服务的多条A记录，每条记录都指向单个pod的IP地址。因此，客户端就能进行一个简单的DNS A记录查找获取该服务所有pod的IP地址，从而可以连接到单个或多个pod。

创建headless服务

Headless服务也是一种Service，但不同的是它会定义spec:clusterIP: None，也就是不需要Cluster IP的服务。Kubernetes不会为headless服务分配集群IP地址。
下面我们创建一个名为test-headless的服务：
vim test-headless-svc.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: test-headless-svc
spec:
  clusterIP: None
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: tomcat
```

apiVersion: v1
kind: Service
metadata:
  name: test-headless-svc
spec:

```
  clusterIP: None
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: tomcat
```

kubectl create –f test–headless–svc.yaml

该服务对应的pod包含app=tomcat的标签

```
[root@k8s-master test]# kubectl create -f test-headless-svc.yaml
service/test-headless-svc created
[root@k8s-master test]# kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP        19d
test-headless-svc   ClusterIP   None            <none>        80/TCP         7s
test-nodeport       NodePort    10.97.42.53     <none>        80:30123/TCP   19d
test-svc            ClusterIP   10.110.132.48   <none>        80/TCP         19d
test-tomcat-svc     ClusterIP   10.105.75.249   <none>        8080/TCP       10d
[root@k8s-master test]#
```

可以看到该服务CLUSTER–IP列值为None。

还可以通过kubectl describe的方式查看该服务的信息：

kubectl describe svc test–headless–svc

```
[root@k8s-master test]# kubectl describe svc test-headless-svc
Name:              test-headless-svc
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=tomcat
Type:              ClusterIP
IP:                None
Port:              <unset>  80/TCP
TargetPort:        8080/TCP
Endpoints:
Session Affinity:  None
Events:            <none>
[root@k8s-master test]#
```

可以发现IP显示为None。另外，这里的Endpoints之所以为空，是因为在上一节中我们针对该服务的3个pod都设置了就绪探针并且就绪探针检查失败，所以这3个pod都未准备就绪：

```
[root@k8s-master test]# kubectl get pod
NAME                                 READY   STATUS    RESTARTS   AGE
test-tomcat-deploy-7898789b6f-2nwct   0/1     Running   108        8d
test-tomcat-deploy-7898789b6f-khn9g   0/1     Running   114        8d
test-tomcat-deploy-7898789b6f-lkrp6   0/1     Running   82         8d
```

执行如下命令，使就绪探针检查成功，这样三个pod就能加入到该服务的endpoints中：

kubectl exec –it test–tomcat–deploy–7898789b6f–2nwct –– touch /var/ready

kubectl exec –it test–tomcat–deploy–7898789b6f–khn9g –– touch /var/ready

kubectl exec –it test–tomcat–deploy–7898789b6f–lkrp6 –– touch /var/ready

```
[root@k8s-master test]# kubectl get pod -o wide
NAME                                 READY   STATUS    RESTARTS   AGE   IP            NODE        NOMINATED NODE   READINESS GATES
test-tomcat-deploy-7898789b6f-2nwct   1/1     Running   108        8d    10.244.1.27   k8s-node1   <none>           <none>
test-tomcat-deploy-7898789b6f-khn9g   1/1     Running   114        8d    10.244.1.26   k8s-node1   <none>           <none>
test-tomcat-deploy-7898789b6f-lkrp6   1/1     Running   82         8d    10.244.1.29   k8s-node1   <none>           <none>
[root@k8s-master test]#
```

```
[root@k8s-master test]# kubectl describe svc test-headless-svc
Name:              test-headless-svc
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=tomcat
Type:              ClusterIP
IP:                None
Port:              <unset>  80/TCP
TargetPort:        8080/TCP
Endpoints:         10.244.1.26:8080,10.244.1.27:8080,10.244.1.29:8080
Session Affinity:  None
Events:            <none>
[root@k8s-master test]#
```

通过DNS发现pod

现在我们来演示如何通过DNS获取pod的IP地址。

我们可以在上面的某个pod中执行查询，前提是pod需要包含nslookup或者dig程序。幸运的是我们演示使用的镜像tomcat:8.5.34-jre8-alpine刚好包含nslookup程序：

```
[root@k8s-master test]# kubectl exec -it test-tomcat-deploy-7898789b6f-2nwct -- bash
bash-4.4# nslookup
BusyBox v1.28.4 (2018-07-17 15:21:40 UTC) multi-call binary.

Usage: nslookup [HOST] [SERVER]

Query the nameserver for the IP address of the given HOST
optionally using a specified DNS server
bash-4.4#
```

当然，我们也可以使用一个含nslookup和dig程序的镜像tutum/dnsutils，并基于该镜像运行一个独立的pod：

kubectl run dnsutils --image=tutum/dnsutils --command -- sleep infinity

```
[root@k8s-master test]# kubectl run dnsutils --image=tutum/dnsutils --command -- sleep infinity
pod/dnsutils created
[root@k8s-master test]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
dnsutils                            1/1     Running   0          3s
test-tomcat-deploy-7898789b6f-2nwct   1/1     Running   108        8d
test-tomcat-deploy-7898789b6f-khn9g   1/1     Running   114        8d
test-tomcat-deploy-7898789b6f-lkrp6   1/1     Running   82         8d
[root@k8s-master test]#
```

工具准备好了后就可以开始执行DNS查找了：

kubectl exec dnsutils -- nslookup test-headless-svc

```
[root@k8s-master test]# kubectl exec dnsutils -- nslookup test-headless-svc
Server:         10.96.0.10
Address:        10.96.0.10#53

Name:   test-headless-svc.default.svc.cluster.local
Address: 10.244.1.26
Name:   test-headless-svc.default.svc.cluster.local
Address: 10.244.1.29
Name:   test-headless-svc.default.svc.cluster.local
Address: 10.244.1.27

[root@k8s-master test]#
```

可以看到DNS服务器(10.96.0.10)针对FQDN test-headless-svc.default.svc.cluster.local返回了3个不同的IP地址，它们都是处于Ready状态的pod的地址。

而针对常规的服务，DNS服务器返回的是服务的集群IP地址：

kubectl exec dnsutils -- nslookup test-svc

```
[root@k8s-master test]# kubectl exec dnsutils -- nslookup test-svc
Server:         10.96.0.10
Address:        10.96.0.10#53

Name:   test-svc.default.svc.cluster.local
Address: 10.110.132.48
```

```
[root@k8s-master test]# kubectl describe svc test-svc
Name:              test-svc
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=test1
Type:              ClusterIP
IP:                10.110.132.48
Port:              <unset>  80/TCP
TargetPort:        8080/TCP
Endpoints:         <none>
Session Affinity:  None
Events:            <none>
[root@k8s-master test]#
```

虽然headless服务看起来与常规的服务有所不同，但是对于客户端来说并无不同，都是通过服务的DNS名称来连接到服务的各个pod。但是当使用headless服务时，由于DNS服务器返回的是pod的IP地址，因此客户端可以直接连接到pod，而不是通过服务代理。不过需要注意的是，headless服务虽然没有使用服务代理，但是仍然可以通过DNS round-robin机制实现pod之间的负载均衡。