25.服务发现

创建服务之后，我们就拥有了一个单一的、固定的IP地址和端口号，通过它就可以访问到pod。这个地址在服务的整个生命周期内都不会改变，而服务后面的pod可能变来变去。它们的IP地址可能变化，数量也会增减，但是始终可以通过服务的单一不变的IP地址访问到这些pod。

但是客户端pod如何知道服务的IP和端口？是否需要先创建服务，然后手动查找它的IP地址并将IP地址传递给客户端pod的配置选项？当然不是。Kubernetes为客户端pod发现服务的IP地址和端口提供了几种方法。

通过环境变量发现服务

当启动pod时，Kubernetes会初始化一组环境变量，指向当前存在的每个服务。如果在创建客户端pod之前就创建了服务，pod中的进程就可以通过检查环境变量而获取到服务的IP地址和端口。

当前环境中有三个pod：

```
[david@dhr-demo root]$ kubectl get pod
NAME            READY   STATUS    RESTARTS   AGE
test-rc-8n2b8   1/1     Running   0          2d11h
test-rc-bxlvx   1/1     Running   0          2d11h
test-rc-xmll8   1/1     Running   0          2d11h
```

执行如下命令可以查看pod的环境变量：

kubectl exec test-rc-8n2b8 -- env

```
[david@dhr-demo root]$ kubectl exec test-rc-8n2b8 -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=test-rc-8n2b8
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
NPM_CONFIG_LOGLEVEL=info
NODE_VERSION=7.10.1
YARN_VERSION=0.24.4
HOME=/root
[david@dhr-demo root]$
```

由于我们是在创建pod之后再创建的service，所以当前我们无法看到在上一节创建的service的环境变量。所以我们需要先删除之前由ReplicationController创建的所有pod，然后由ReplicationController再次重新创建新的pod：

kubectl delete po --all

```
[david@dhr-demo root]$ kubectl delete po --all
pod "test-rc-8n2b8" deleted
pod "test-rc-bxlvx" deleted
pod "test-rc-xmll8" deleted
```

查看自动创建的pod：

```
[david@dhr-demo root]$ kubectl get pod
NAME            READY   STATUS    RESTARTS   AGE
test-rc-j9j4w   1/1     Running   0          49s
test-rc-qhpkl   1/1     Running   0          49s
test-rc-qpnd5   1/1     Running   0          49s
[david@dhr-demo root]$
```

可以在容器中运行env命令查看pod的环境变量：

kubectl exec test-rc-j9j4w -- env

```
[david@dhr-demo root]$ kubectl exec test-rc-j9j4w -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=test-rc-j9j4w
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
TEST_SVC_PORT_80_TCP=tcp://10.106.143.212:80
TEST_SVC1_PORT_80_TCP=tcp://10.107.168.13:80
TEST_SVC1_PORT_80_TCP_ADDR=10.107.168.13
KUBERNETES_SERVICE_PORT_HTTPS=443
TEST_SVC_SERVICE_HOST=10.106.143.212
TEST_SVC_PORT=tcp://10.106.143.212:80
TEST_SVC1_SERVICE_PORT=80
TEST_SVC_PORT_80_TCP_ADDR=10.106.143.212
TEST_SVC1_SERVICE_HOST=10.107.168.13
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
TEST_SVC_SERVICE_PORT=80
TEST_SVC_PORT_80_TCP_PORT=80
TEST_SVC1_PORT_80_TCP_PORT=80
KUBERNETES_PORT_443_TCP_PORT=443
TEST_SVC_PORT_80_TCP_PROTO=tcp
TEST_SVC1_PORT=tcp://10.107.168.13:80
TEST_SVC1_PORT_80_TCP_PROTO=tcp
NPM_CONFIG_LOGLEVEL=info
NODE_VERSION=7.10.1
YARN_VERSION=0.24.4
HOME=/root
[david@dhr-demo root]$
```

TEST_SVC_SERVICE_HOST就是服务的集群IP地址，TEST_SVC_PORT是服务的端口。

在上一节我们提到了前端web服务器pod和后端数据库服务器pod的例子，如果前端pod需要使用后端数据库服务器pod，可以通过一个比如叫做backend-database的服务将后端pod暴露出去，然后让前端pod通过环境变量BACKEND_DATABASE_SERVICE_HOST和BACKEND_DATABASE_SERVICE_PORT查找服务的IP地址和端口。

> 服务名中的横杠会被转换成下划线，当服务名被用作环境变量名前缀的时候所有字母转成大写

环境变量是查找服务IP地址和端口的一种方式。Kubernetes还包含了一个DNS服务器，可以通过DNS来查找服务IP地址。

通过DNS发现服务

在kube-system命名空间下有一个叫做coredns的pod。从 K8S 1.11 开始，K8S 使用 CoreDNS替换KubeDNS作为DNS服务器：

kubectl get po --namespace kube-system

```
[david@dhr-demo root]$ kubectl get po --namespace kube-system
NAME                               READY   STATUS    RESTARTS   AGE
coredns-6c76c8bb89-m9jft           1/1     Running   27         47d
etcd-minikube                      1/1     Running   15         47d
kube-apiserver-minikube            1/1     Running   136        47d
kube-controller-manager-minikube   1/1     Running   0          47d
kube-proxy-bsdlz                   1/1     Running   0          47d
kube-scheduler-minikube            1/1     Running   13         47d
storage-provisioner                1/1     Running   140        47d
```

在该命名空间下还存在一个名为kube-dns的service：

kubectl get svc --namespace kube-system

```
[david@dhr-demo root]$ kubectl get svc --namespace kube-system
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AGE
kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   47d
[david@dhr-demo root]$
```

通过名称可以看出该pod中运行了一个DNS服务器，集群中运行的所有其他pod会自动地配置使用该DNS服务器（Kubernetes通过修改每个容器的/etc/resolv.conf文件来实现）。Pod中运行进程执行的任何DNS查询都会被Kubernetes自身的DNS服务器处理，它知道系统中运行的所有服务。

> 可以通过在每个pod的spec区段中配置dnsPolicy属性来决定其是否使用内部的DNS服务器

每个服务从内部的DNS服务器中获得一个DNS条目，客户端pod在知道服务名的情况下可以通过它的全限定域名（FQDN=Fully Qualified Domain Name）来访问服务，而不是使用环境变量。

通过FQDN连接服务

一个前端pod可以通过打开一个到如下FQDN的连接来与后端数据库服务建立连接：

backend-database.default.svc.cluster.local

backend-database对应于服务名称，default表示服务所在的命名空间，svc.cluster.local是一个在所有集群本地服务名中使用的可配置的集群domain后缀。

> 客户端仍然必须知道服务的端口号。如果服务正在使用一个标准的端口（HTTP的80端口或者MySQL的3306端口），这样是不会有问题的。如果使用的是非标准端口，客户端可以从环境变量获得端口号。

我们还可以去掉svc.cluster.local后缀，甚至命名空间（如果前端pod和数据库pod位于同一个命名空间下），因此可以简单地只通过backend-database引用服务。

现在我们试着通过FQDN来访问之前创建的service，这需要在一个已有的pod中来完成。

可以使用kubectl exec命令在一个已有的pod中运行bash命令进入pod容器：

kubectl exec -it test-rc-j9j4w -- bash

```
[david@dhr-demo root]$ kubectl exec -it test-rc-j9j4w -- bash
root@test-rc-j9j4w:/#
```

当前我们处在容器中，因此可以通过如下几种方式使用curl命令访问test-svc服务：

curl http://test-svc.default.svc.cluster.local

```
root@test-rc-j9j4w:/# curl http://test-svc.default.svc.cluster.local
已发送消息至： test-rc-qpnd5
root@test-rc-j9j4w:/#
```

curl http://test-svc.default

```
root@test-rc-j9j4w:/# curl http://test-svc.default
已发送消息至： test-rc-qhpkl
root@test-rc-j9j4w:/#
```

curl http://test-svc

```
root@test-rc-j9j4w:/# curl http://test-svc
已发送消息至： test-rc-qhpkl
root@test-rc-j9j4w:/#
```

在请求URL中，可以将服务名作为主机名来访问服务，因为根据每个pod容器DNS解析器的配置方式，可以将命名空间和svc.cluster.local后缀移除。
在容器中查看/etc/resolv.conf文件：

```
root@test-rc-j9j4w:/# cat /etc/resolv.conf
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
root@test-rc-j9j4w:/#
```

如果我们试着直接ping服务，会发现无法ping通：
ping test-svc

```
root@test-rc-j9j4w:/# ping test-svc
PING test-svc.default.svc.cluster.local (10.106.143.212): 56 data bytes
^C--- test-svc.default.svc.cluster.local ping statistics ---
477 packets transmitted, 0 packets received, 100% packet loss
root@test-rc-j9j4w:/#
```

为什么curl可以，而ping不可以呢？这是由于服务的集群IP是一个虚拟的IP，而且只有和服务端口号结合使用的时候才有意义。这点我们在后面的学习中再慢慢了解具体原因以及服务的工作方式。