前言

前几章，我们已经实现了对一个单体应用的部署。可是，我们的项目中还经常遇到以下几种情况：

> 我要用Nginx做负载均衡，如何才能转发到别的服务上面？ 我的后端需要MySQL数据库，我怎样才能连接到同级服务的数据库呢？ ……

这些场景都有个共性问题： `A服务` 依赖另一个 `B服务` ，而我们常常不知道 `B服务` 的端口和IP，且端口和IP也相对不固定有可能经常更改。

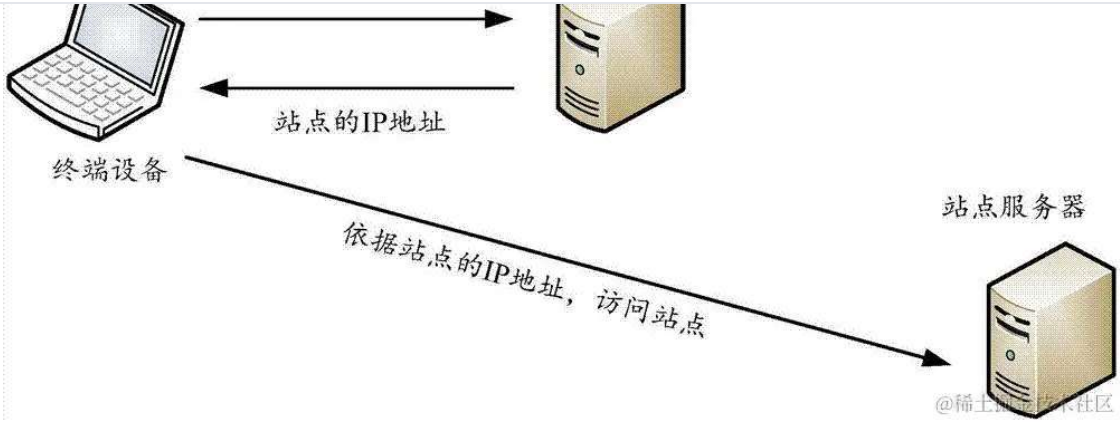
这时候，我们就需要一个神器 —— **服务发现**

什么是服务发现

我们先看下百度百科的解释：

> 服务发现是指使用一个注册中心来记录分布式系统中的全部服务的信息，以便其他服务能够快速的找到这些已注册的服务。

是不是有点懵？我们思考一下：当我们通过域名访问一个网站时，浏览器不会直接访问域名。而是先将域名发送至 `DNS` 服务器，获取到域名对应的IP后，再通过IP去访问真实服务器（如下图）。



其实**我们日常上网，DNS服务器 将域名映射为真实IP的过程，就是一个服务发现的过程**。而我们再也不需要记住每个网站的IP，只需要记住永远不会更改的**域名**即可。

那么在 Kubernetes 中，如何做服务发现呢？我们前面写到过， Pod 的 IP 常常是漂移且不固定的，所以我们要使用 Service 这个神器来将它的访问入口固定住。

但是，我们在部署 Service 时，也不知道部署后的ip和端口如何。那么在 Kubernetes 中，我们可以利用 DNS 的机制给每个 Service 加一个内部的**域名**，指向其真实的IP。

Kubernetes CoreDNS

在 Kubernetes 中，对 Service 的服务发现，是通过一种叫做 CoreDNS 的组件去实现的。

CoreDNS 是使用 Go 语言实现的一个DNS服务器。当然，它也不只是可以用在 Kubernetes 上。也可以用作日常 DNS 服务器使用。在 Kubernetes 1.11版本后，CoreDNS 已经被默认安装进了 Kubernetes 内。

我们也通过下面的命令验证下 CoreDNS 是否已经安装成功：

shell 复制代码

```
kubectl -n kube-system get all  -l k8s-app=kube-dns -o wide
```

```
[root@master ~]# kubectl -n kube-system get all  -l k8s-app=kube-dns -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
pod/coredns-6c76c8bb89-9x494  1/1     Running   5          41d   10.244.0.13  master   <none>           <none>
pod/coredns-6c76c8bb89-lvr4j  1/1     Running   5          41d   10.244.0.12  master   <none>           <none>

NAME               TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AGE   SELECTOR
service/kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   41d   k8s-app=kube-dns

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES                                                  SELECTOR
deployment.apps/coredns   2/2     2            2           41d   coredns      registry.cn-hangzhou.aliyuncs.com/google_containers/coredns:1.7.0   k8s-app=kube-dns

NAME                                 DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES                                                  SELECTOR
replicaset.apps/coredns-6c76c8bb89   2         2         2       41d   coredns      registry.cn-hangzhou.aliyuncs.com/google_containers/coredns:1.7.0   k8s-app=kube-dns,pod-template-hash=6c76c8bb89
```

在了解了 CoreDNS 的背景后，我们开始来验证下服务发现的规则。

首先，我们先进入一个 Pod 进行测试。我们先使用 kubectl get pods 命令来看下当前运行了哪些 Pod：

```
[root@master ~]# kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
front-v1-787bf5c86d-t78x5     1/1     Running   0          75m
front-v2-b65d5fd66-22g7d      1/1     Running   0          75m
front-v2-b65d5fd66-kjgqm      1/1     Running   0          75m
front-v2-b65d5fd66-zm9vl      1/1     Running   0          75m
[root@master ~]#
```


接着使用 kubectl get svc 看下运行了哪些 Service ：

```
[root@master ~]# kubectl get svc
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
front-service-v1   NodePort    10.106.251.47   <none>        80:31048/TCP   41d
front-service-v2   NodePort    10.99.38.70     <none>        80:30889/TCP   39d
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP        41d
[root@master ~]#
```


可以看到，我们自己创建的 Service 有2个： front-service-v1 和 front-service-v2 。稍后我们就使用这两个 Service 来测试。

这里我们选择其中一个 Pod 进入看下。在这里，我们可以使用 kubectl exec 命令进入 Pod 内的容器。

shell 复制代码

```
kubectl exec -it front-v1-787bf5c86d-t78x5 -- /bin/sh
```

> kubectl exec 的作用是可以直接在容器内执行Shell脚本。 命令格式：kubectl exec -it [PodName] -- [Command] -i：即使没有连接，也要保持标准输入保持打开状态。一般与 -t 连用。 -t：分配一个伪TTY（终端设备终端窗口），一般与 -i 连用。可以分配给我们一个Shell终端

执行后，我们就进入了容器内部环境。此时，我们可以验证下服务规则。

在 Kubernetes DNS 里，服务发现规则有2种：跨 namespace 和同 namespace 的规则。

> kubernetes namespace（命名空间）是 kubernetes 里比较重要的一个概念。在启动集群后，kubernetes 会分配一个默认命名空间，叫default。不同的命名空间可以实现资源隔离，服务隔离，甚至权限隔离。

在同 namespace 下的规则，我们只需要直接访问 **http://ServiceName:Port** 就可以访问到相应的 Service 。这里使用 wget -q -O- 即可将访问内容输出到控制台上：

shell 复制代码

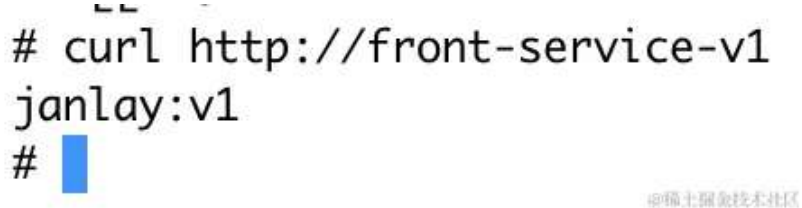
```
wget -q -O- http://front-service-v1
```

> wget 是 Linux 平台中的一个下载文件的工具

如果你没有 wget 命令，使用 curl 命令替代它也可以：

shell 复制代码

```
curl http://front-service-v1
```



这里我们可以看到，是可以访问到同级服务的。当然我们上面也写到，还有一种是**跨 namespace 的发现规则。**不过即使是同 namespace ，也可以使用**跨namespace的发现规则。**

在 Kubernetes DNS 中，跨 namespace 的规则略为复杂。格式如下：

css 复制代码

```
[ServiceName].[NameSpace].svc.cluster.local
```

这里的 ServiceName 就是我们创建的 Service 名称； NameSpace 则是命名空间。如果你没有命名空间，则这个值为 default 。

我们按照这个规则，再来尝试访问下：

shell 复制代码

```
curl http://front-service-v1.default.svc.cluster.local
```

经过验证，可以访问到。

结束语

在本章，我们通过学习 Kubernetes DNS 的规则，了解到了如何访问到同级服务。如何访问到跨命名空间的服务。在下一章，我们将学习如何更好地利用 ConfigMap 抽离环境变量，更好地解耦配置。

如果你有疑问的话，欢迎在评论区指出 👏

上一章 下一章

留言

输入评论（Enter换行，Ctrl + Enter发送）

发表评论

全部评论（2）

叶_同学 LV.3 JY.4 前端切图仔 1年前

exit 命令退出容器交互模式

1 回复 ···

赧然一笑 JY.3 1年前

通俗易懂，真的牛。

1 回复 ···