

dubbo-go K8s 注册中心的设计方案与实现

王翔 阿里云云栖号 3天前

Dubbo-go k8s注册中心设计方案与实现

随着云原生的推广，越来越多的公司或组织将服务容器化，并将容器化后的服务部署在k8s集群中。

k8s管理资源的哲学

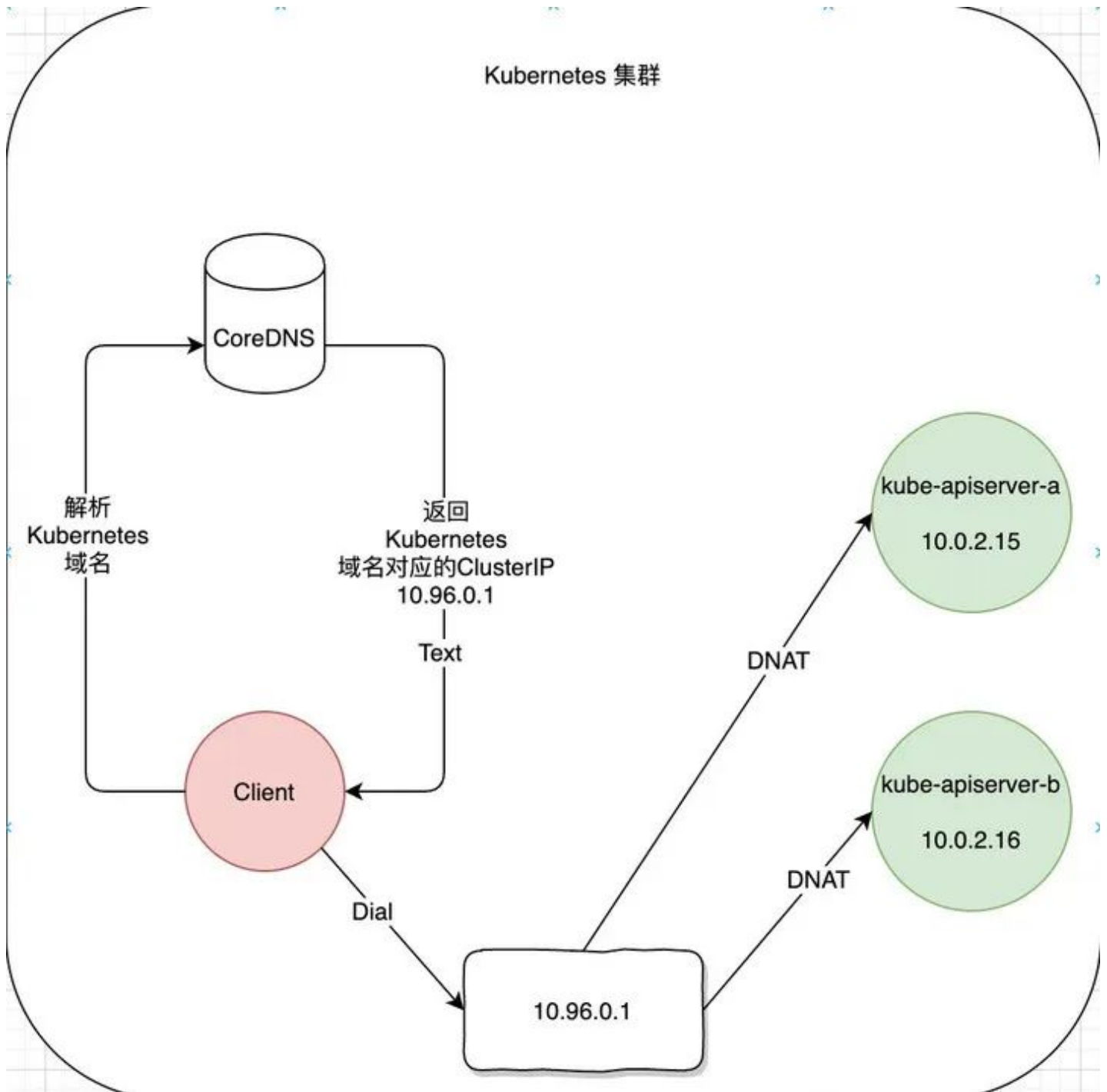
k8s作为容器集群化管理方案可以将管理资源的维度可主观的分为服务实例管理和服务接入管理。

- 服务实例管理，主要体现方式为Pod设计模式加控制器模式。控制器保证具有特定标签（Label）的Pod保持在恒定的数量（多删，少补）。
- 服务接入管理，主要为Service，该Service默认为具有特定标签（Label）的一批Pod提供一个VIP（ClusterIP）作为服务的接入点，默认会按照round-robin的负载均衡策略将请求转发到真正提供服务的Pod。并且CoreDNS为该Service提供集群内唯一的域名。

k8s服务发现模型

为了明确k8s在服务接入管理提供的解决方案，我们以kube-apiserver提供的API(HTTPS)服务为例。k8s集群为该服务分配了一个集群内有效的ClusterIP，并通过CoreDNS为其分配了唯一的域名 `kubernetes`。

Kubernetes 集群



具体流程如上图所示(红色为客户端，绿色为kube-apiserver)：

1. 首先客户端通过CoreDNS解析域名为kubernetes的服务获得对应的ClusterIP为

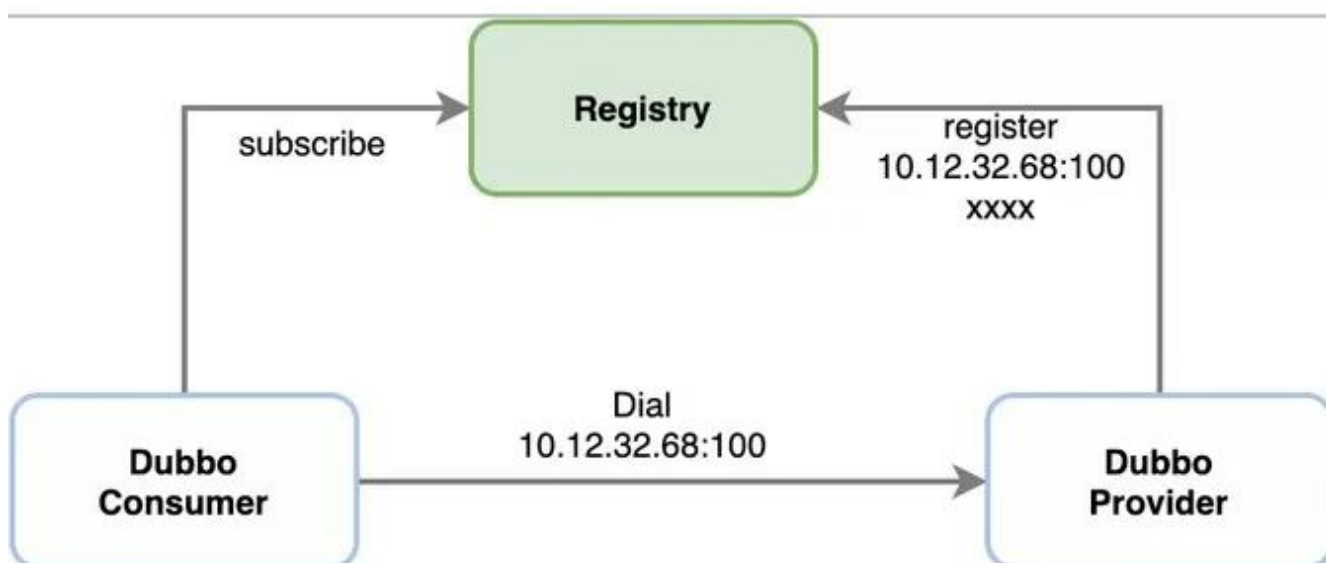
10.96.0.1。

2. 客户端向10.96.0.1发起HTTPS请求。
3. HTTPS之下的TCP连接被kube-proxy创建的iptables的PREROUTING链拦截并DNAT 为 10.0.2.16或10.0.2.15。
4. Client与最终提供服务的Pod建立连接并交互。

由此可见，k8s提供的服务发现为域名解析级别。

Dubbo服务发现模型

同样为了明确Dubbo服务发现的模型，以一个简单的Dubbo-Consumer发现并访问Provider的具体流程为例。



具体流程如上图所示：

1. Provider将本进程的元数据注册到Registry中，包括IP，Port，以及服务名称等。
2. Consumer通过Registry获取Provider的接入信息，直接发起请求

由此可见，Dubbo当前的服务发现模型是针对Endpoint级别的，并且注册的信息不只IP

和端口还包括其他的一些元数据。

K8s service vs dubbo-go 服务

通过上述两个小节，答案基本已经比较清晰了。总结一下，无法直接使用k8s的服务发现模型的原因主要为以下几点：

1. k8s的Service标准的资源对象具有的服务描述字段 中并未提供完整的Dubbo进程元数据字段因此，无法直接使用该标准对象进行服务注册与发现。
2. dubbo-go的服务注册是基于每个进程的，每个Dubbo进程均需进行独立的注册。
3. k8s的Service默认为服务创建VIP，提供round-robin的负载策略也与Dubbo-go自有的Cluster模块的负载策略形成了冲突。

Dubbo-go 当前的方案

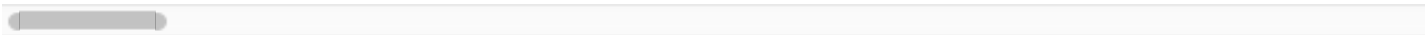
服务注册

K8s基于Service对象实现服务注册/发现。可是dubbo现有方案为每个dubbo-go进程独立注册，因此dubbo-go选择将该进程具有的独有的元数据写入运行该dubbo-go进程的Pod在k8s中的Pod资源对象的描述信息中。每个运行dubbo进程的Pod将本进程的元数据写入Pod的Annotations字段。为了避免与其他使用Annotations字段的Operator或者其他类型的控制器（istio）的字段冲突。dubbo-go使用Key为 dubbo.io/annotation value为具体存储的K/V对的数组的json编码后的base64编码。

样例为：

```
apiVersion: v1
kind: Pod
metadata:
```

```
annotations:  
  dubbo.io/annotation: W3siaYI6Ii9kdWJibyIsInYiOiIifSx7ImsiOiIvZHVhYm8\
```

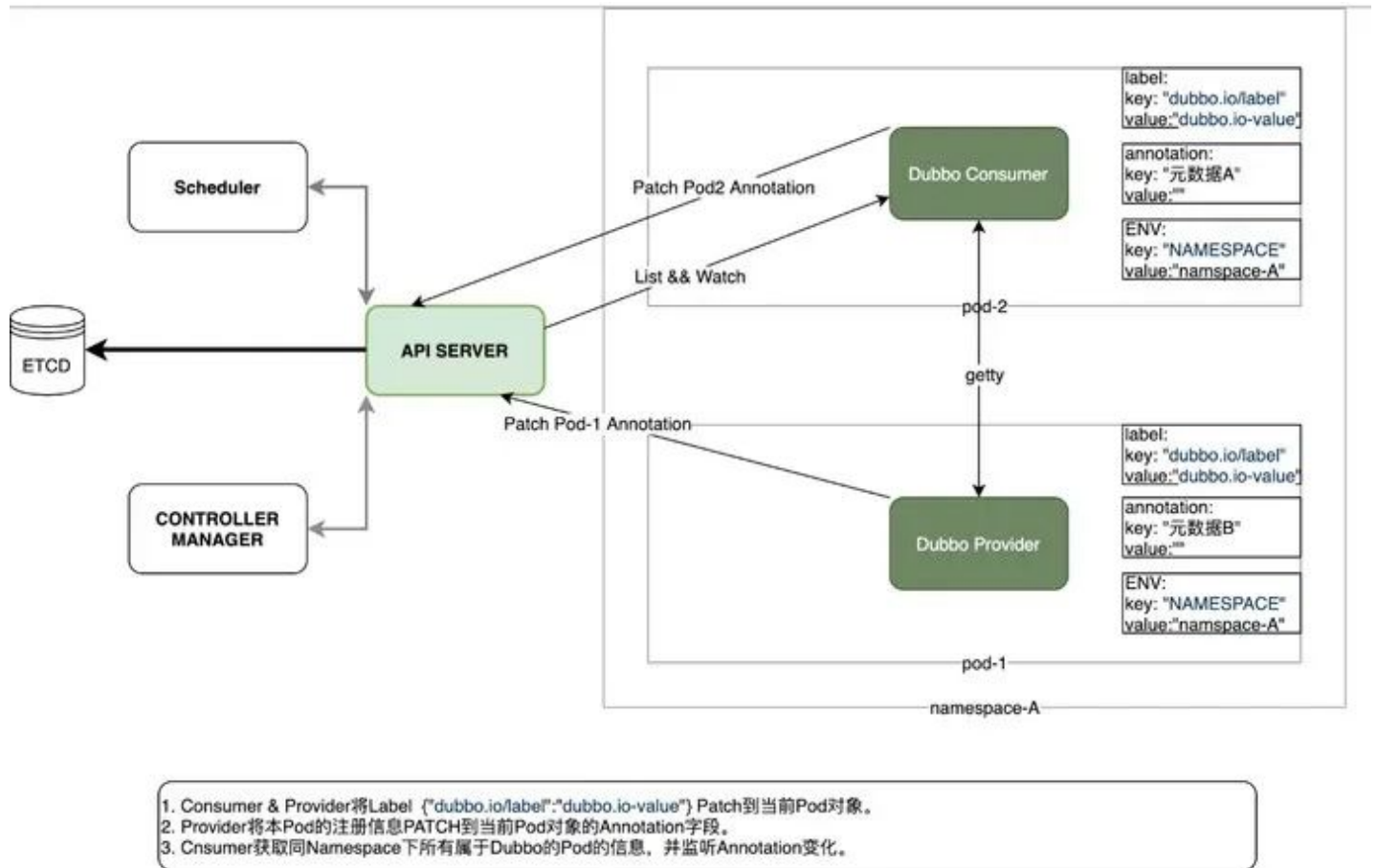


由于每个dubbo-go的Pod均只负责注册本进程的元数据，因此Annotations字段长度也不会因为运行dubbo-go进程的Pod数量增加而增加。

服务发现

依赖kube-apiserver 提供了WATCH的功能。可以观察特定namespace内各Pod对象的变化。dubbo-go为了避免dubbo-go进程WATCH到与dubbo-go进程无关的Pod的变化，dubbo-go将WATCH的条件限制在当前Pod所在的namespace，以及仅WATCH具有Key为 dubbo.io/label Value为 dubbo.io-value 的Pod。在WATCH到对应Pod的变化后实时更新本地Cache，并通过Registry提供的Subscribe接口通知建立在注册中心之上的服务集群管理其他模块。

总体设计图



具体流程如上图所示：

1. 启动dubbo-go的Deployment或其他类型控制器使用k8s Downward-Api将本Pod所在namespace通过环境变量的形式注入dubbo-go进程。
2. Consumer/Provider进程所在的Pod启动后通过环境变量获得当前的namespace以及该Pod名称，调用 kube-apiserver PATCH 功能为本Pod添加Key为 dubbo.io/label Value为 dubbo.io-value 的label。
3. Consumer/Provider进程所在的Pod启动后调用kube-apiserver将本进程的元数据通过PATCH接口写入当前Pod的Annotations字段。
4. Consumer进程通过kube-apiserver LIST 当前namespace下其他具有同样标签的Pod，并解码对应的Annotations字段获取Provider的信息。
5. Consumer进程通过 kube-apiserver WATCH 当前namespace下其他具有同样label的Pod的Annotations的字段变化，动态更新本地Cache。

总结

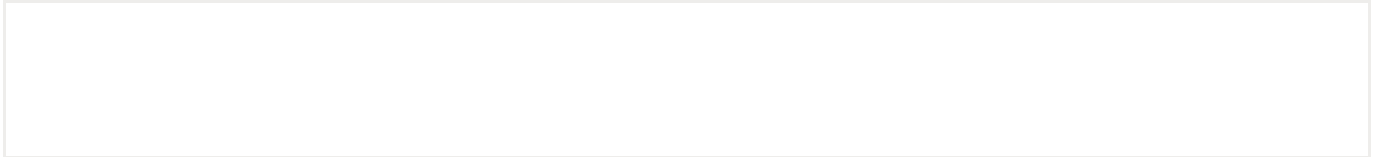
k8s已经为其承载的服务提供了一套服务发现，服务注册，以及服务集群管理机制。而dubbo-go的同时也拥有自成体系的服务集群管理。这两个功能点形成了冲突，在无法调谐两者的情况，dubbo-go团队决定保持dubbo自有的服务集群管理系，而选择性的放弃了Service功能，将元数据直接写入到Pod对象的Annotations中。

当然这只是dubbo-go在将k8s作为服务注册中心的方案之一，后续社区会以更加“云原生”的形式对接k8s，让我们拭目以待吧。



长按二维码识别关注云栖号

动动小手指 了解更多详情！



[阅读原文](#)