Dubbo Go Registry For Kubernetes

Dubbo-Go Registry

注册中心作为dubbo-go中的底层设施提供了dubbo进程元数据（IP, Port, Interface, Group,Method等）存储，被Watch的功能，每个Dubbo进程均需接入同一组持久化的K/V介质集群（比如: zookeeper, etcdv3等)。各进程均需将本进程的元数据存储于注册中心，并且能够Watch到其他dubbo进程的元数据变化（包括创建，更新等）。

Kubernetes

Kubernetes作为容器集群化管理方案管理资源的维度可主观的分为服务进程管理和服务接入管理。

- 服务进程管理，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（Kubernetes-Label）的Pod保持在恒定的数量（多删，少补）。
- 服务接入管理，主要为Kubernetes-Service，该Service默认为具有特定标签（Kubernetes-Label）的Pod统一提供一个VIP（Kubernetes-ClusterIP）所有需要请求该组Pod的请求都会按照round-robin的负载策略转发到真正提供服务的Pod。并且CoreDNS为该Kubernetes-Service提供集群内唯一的域名。

Service与Dubbo-Go存在的冲突点

- Kubernetes-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Kubernetes-Service进行服务注册与发现。
- dubbo-do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。
- Kubernetes-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。

抛弃Service对象，选择Pod对象进行注册

Kubernetes-Service与dubbo-go现有架构的冲突导致Dubbo-Go在选择服务注册与发现的时候只能选择放弃该资源对象。dubbo-go既然选择了每个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Kubernetes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Kubernetes-Pod 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:
W3siayI6Ii9kdWJibyIsInYiOiIifSx7ImsiOiIvZHViYm8vY29tLmlrdXJlbnRvLnVzZXIuVXNlcl
Byb3ZpZGVyIiwidiI6IiJ9LHsiayI6Ii9kdWJiby9jb20uaWt1cmVudG8udXNlci5Vc2VyUHJvdmlk
ZXIvY29uc3VtZXJzIiwidiI6IiJ9LHsiayI6Ii9kdWJibyIsInYiOiIifSx7ImsiOiIvZHViYm8vY2
9tLmlrdXJlbnRvLnVzZXIuVXNlclByb3ZpZGVyIiwidiI6IiJ9LHsiayI6Ii9kdWJiby9jb20uaWt1
cmVudG8udXNlci5Vc2VyUHJvdmlkZXIvcHJvdmlkZXJzIiwidiI6IiJ9LHsiayI6Ii9kdWJiby9jb2
0uaWt1cmVudG8udXNlci5Vc2VyUHJvdmlkZXIvY29uc3VtZXJzL2NvbnN1bWVyJTNBJTJGJTJGMTcy
LjE3LjAuOCUyRlVzZXJQcm92aWRlciUzRmNhdGVnb3J5JTNEY29uc3VtZXJzJTI2ZHViYm8lM0RkdW
Jib2dvLWNvbnN1bWVyLTIuNi4wJTI2cHJvdG9jb2wlM0RkdWJibyIsInYiOiIifV0=
```

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。

Dubbo-Go 服务发现

解决掉了服务注册问题，接下来需要解决的是服务发现的问题。

Kubernetes Api-Server 提供了Watch的功能，可以观察特定namespace甚至整个集群内各类资源的变化。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 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通知建立在注册中心之上的服务集群管理，或者其他功能。

工作流程

- 启动dubbo-go的Deployment或其他类型控制器使用Kubernetes Downward-Api将本Pod所在namespace通过环境变量的形式注入dubbo-go进程。
- dubbo-go进程的Pod启动后通过环境变量获得当前的namespace以及该Pod名称， 调用Kubernetes-Apiserver PATCH 功能为本Pod添加Key为**dubbo.io/label** Value为 **dubbo.io-value** 的label。
- dubbo-go进程调用Kubernetes-Apiserver 将本进程的元数据通过PATCH接口写入当前Pod的Annotations字段。
- dubbo-go进程 LIST 当前namespace下其他具有同样标签的Pod，并解码对应的Annotations字段获取其他Pod的信息。
- dubbo-go进程 WATCH 当前namespace下其他具有同样标签的Pod的Annotations的字段变化。

总结

Kubernetes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选择保持自有的服务集群管理系，放弃Kubernetes-Service功能，将元数据直接写入到Kubernetes Pod内。依赖Kubernetes提供的Watch功能提供维护服务集群状态。