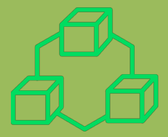
Dubbo 2.7.3/4 新特性及发版计划

刘军（chickenlj@github）

Agenda

2.7版本

概括性的介绍2.7版本的新特性，包括升级注意事项、异步调用、注册中心、配置中心及治理规则等



Microservices

探索Dubbo如何更好的支持微服务体系建设，对比Dubbo与微服务体系有哪些不同，各自的优劣是什么，我们最近在这方面做了哪些探索与改造。

Cloud Native

Dubbo应用能在k8s体系下部署，提供更完善的Observability、Streaming能力，探索ServiceMesh场景下Dubbo的工作模式

2.7版本

特性一览

Repackage

Apache的项目
org.apache.dubbo

JDK8

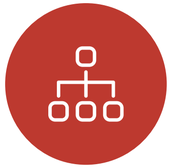
Default method
CompletableFuture

异步支持

支持更友好的使用方式
支持Provider端异步

元数据

简化注册中心配置项
丰富元数据信息



动态配置

远程配置
服务治理参数的配置

路由规则

与注册中心分离，使用配置中心存储
支持更多路由规则

2.7版本

升级与兼容性

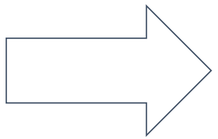
低版本兼容性

- 服务发现
- 通信协议
- 服务治理规则

Apache Dubbo

- Maven坐标: org.apache.dubbo
- Package：org.apache.dubbo
- JDK 1.8
- API，建议上线前直接替换
- SPI，仅限深度定制用户

```
<dependency>
   <groupId>com.alibaba</groupId>
   <artifactId>dubbo</artifactId>
   <version>2.6.x or lower</version>
</dependency>
```



```
<dependency>
   <groupId>org.apache.dubbo</groupId>
   <artifactId>dubbo</artifactId>
   <version>2.7.x</version>
</dependency>
```

2.7版本

异步支持


```
//方式一：接口中直接定义CompletableFuture
public interface HelloService {
  // Synchronous style
  String sayHello(String name);

  // Asynchronous style
  default CompletableFuture<String> sayHelloAsync(String name)
  {
     return CompletableFuture
            .completedFuture(sayHello(name));
  }

}
```

- Jdk8新特性：CompletableFuture方式
- 同时定义同步和异步签名方法
- 直接定义异步签名方法

2.7版本

三个中心

注册中心集群

配置中心集群

元数据中心
集群

```
<dubbo:configcenter address=”nacos://127.0.0.1:8848"/>

<dubbo:service interface=”DemoService” />
<dubbo:service interface=”DemoService” />
```

```
<dubbo:registry address="zookeeper://127.0.0.1:2181"/>

<dubbo:service interface=”DemoService” />
<dubbo:service interface=”DemoService” />
```

```
dubbo.registry.address=zookeeper://127.0.0.1:2181
dubbo.metadata-report.address=zookeeper://127.0.0.1:2181
dubbo.provider.timeout=2000
... ...
```

2.7版本

配置中心

动态配置

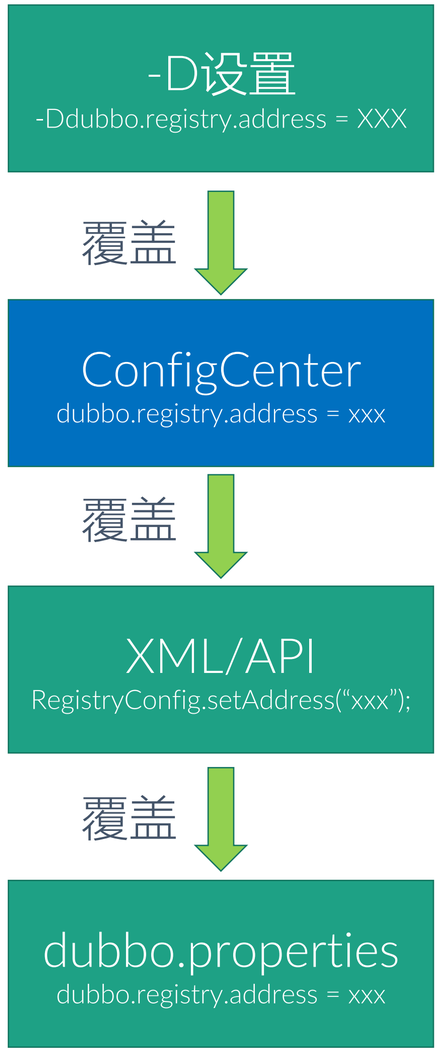
支持类似于Spring Cloud Config的远程配置方式托管
支持新的覆盖关系

1. 编码方式
<dubbo:provider tag="grey"/>

2. 环境变量
DUBBO_PROVIDER_TAG = grey

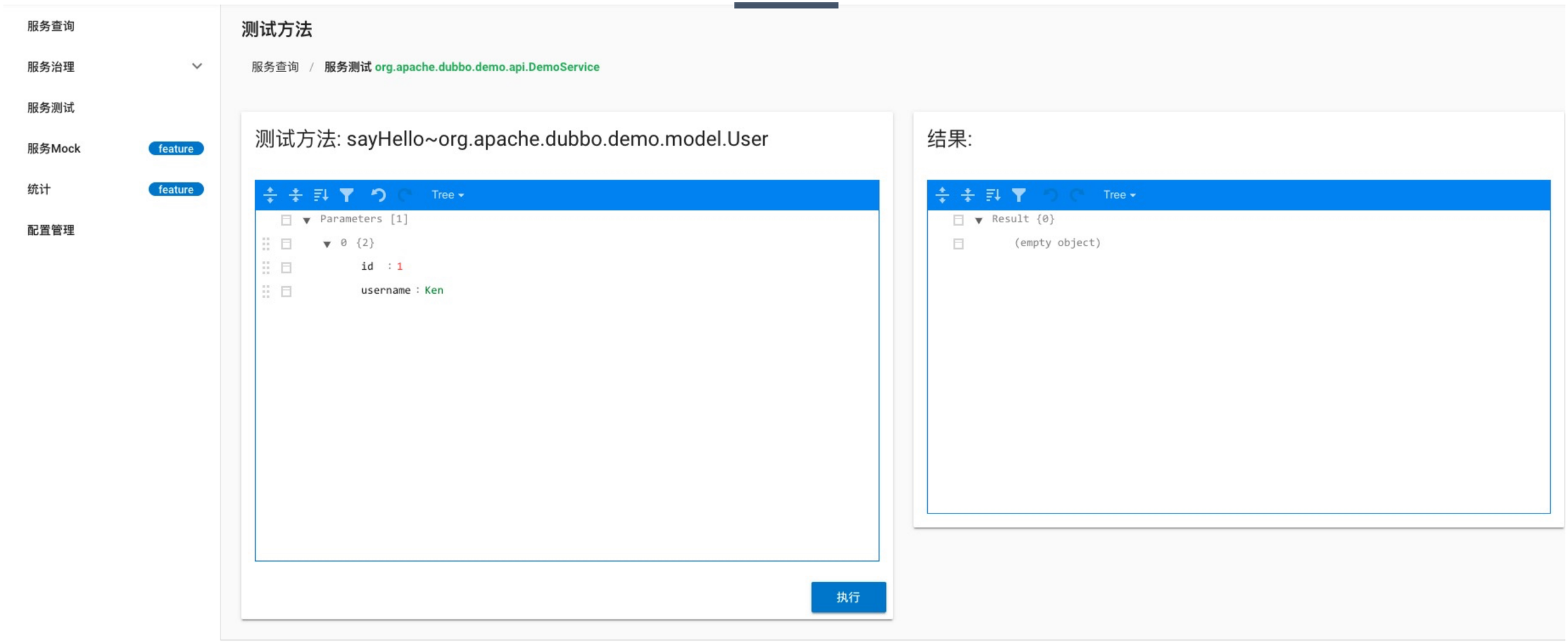
3. JVM 参数
-Ddubbo.provider.tag = grey

4. dubbo.properties
dubbo.provider.tag = grey



2.7版本

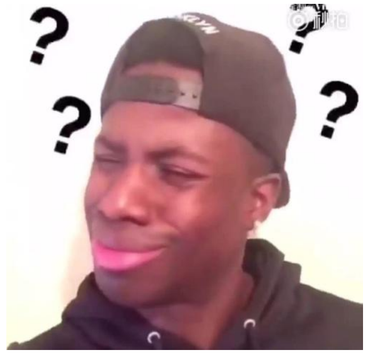
元数据与服务测试

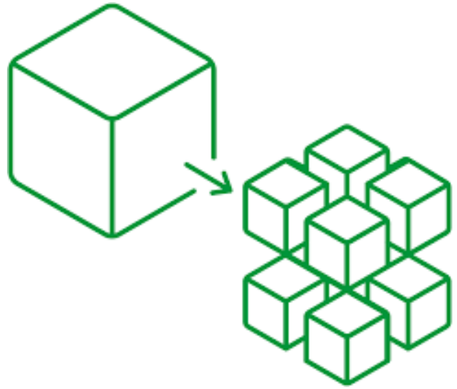


接下来的发布计划.....

微服务

RPC vs Microservices



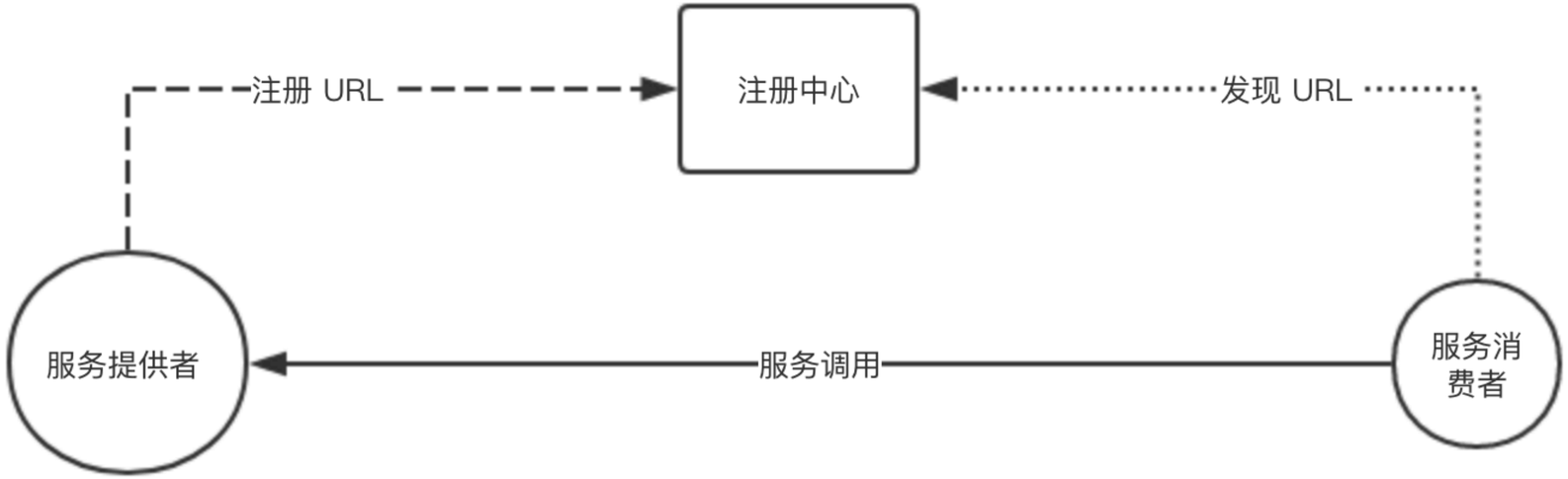


VS

Spring Cloud + Microservices

微服务

现有服务发现模型



性能挑战

- 注册中心数据条目和接口数成正比；
- 大量配置信息和地址混合；
- 应用上下线造成地址推送、计算挑战

微服务体系互通

- 和主流微服务体系概念难以对齐
- 无法复用K8S服务基础设施
- 很难对接一些专业的微服务产品，如Consul等

服务治理

- 缺乏应用/实例粒度的服务治理，包括查询和规则下发等；

微服务

微服务服务发现模型

以应用粒度注册、注册中心只关注地址变更

- IP + PORT；
- 少量实例环境相关配置，如协议、实例所属区域、部署环境等；

元数据服务提供额外信息

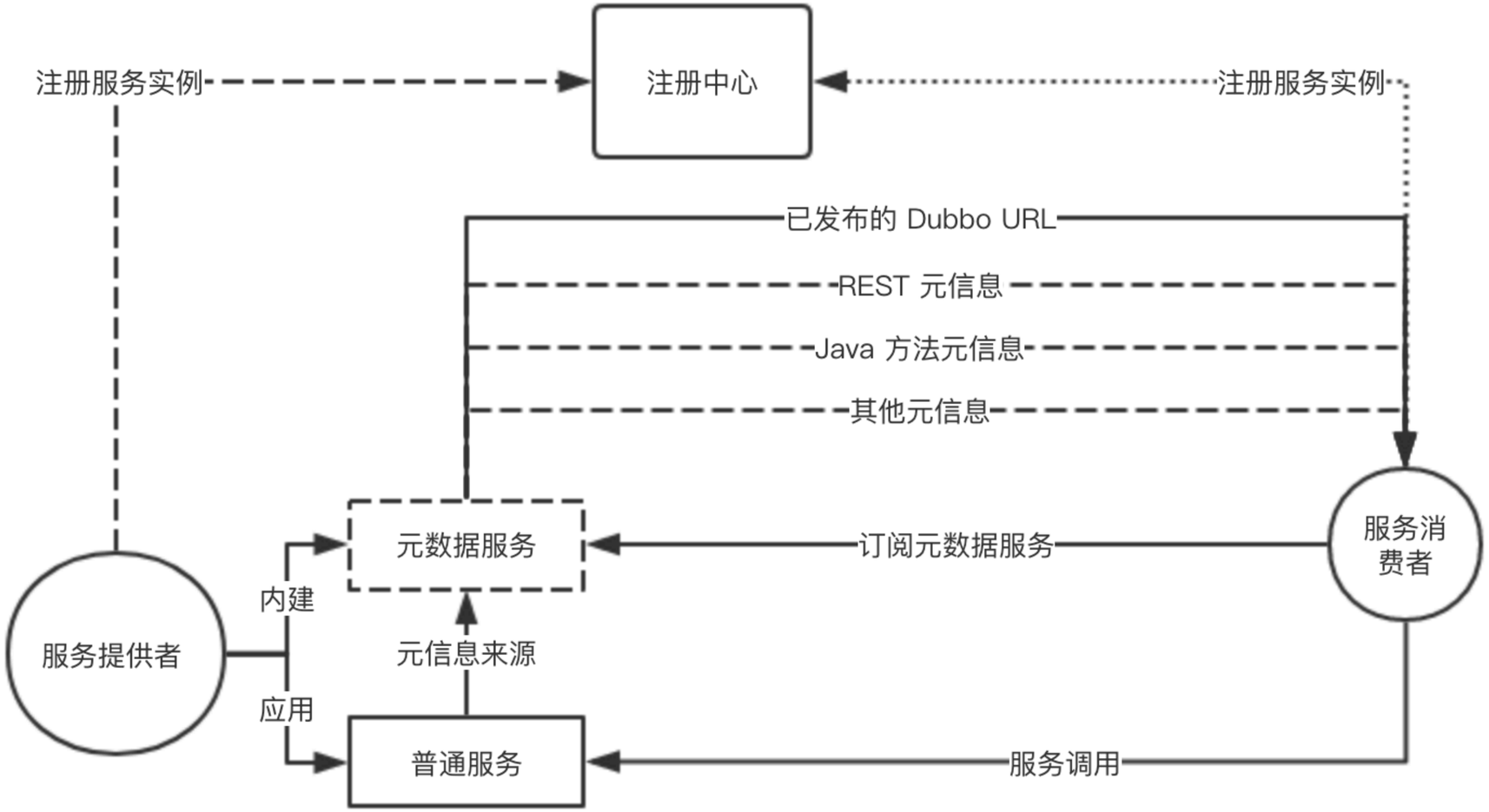
- 接口列表、方法列表、方法签名等；
- 实例特有配置；

保持RPC编程风格不变

- 继续面向接口编程，无需额外改造
- 仅通过Registry配置，区别新老服务发现模型；

微服务

微服务服务发现模型

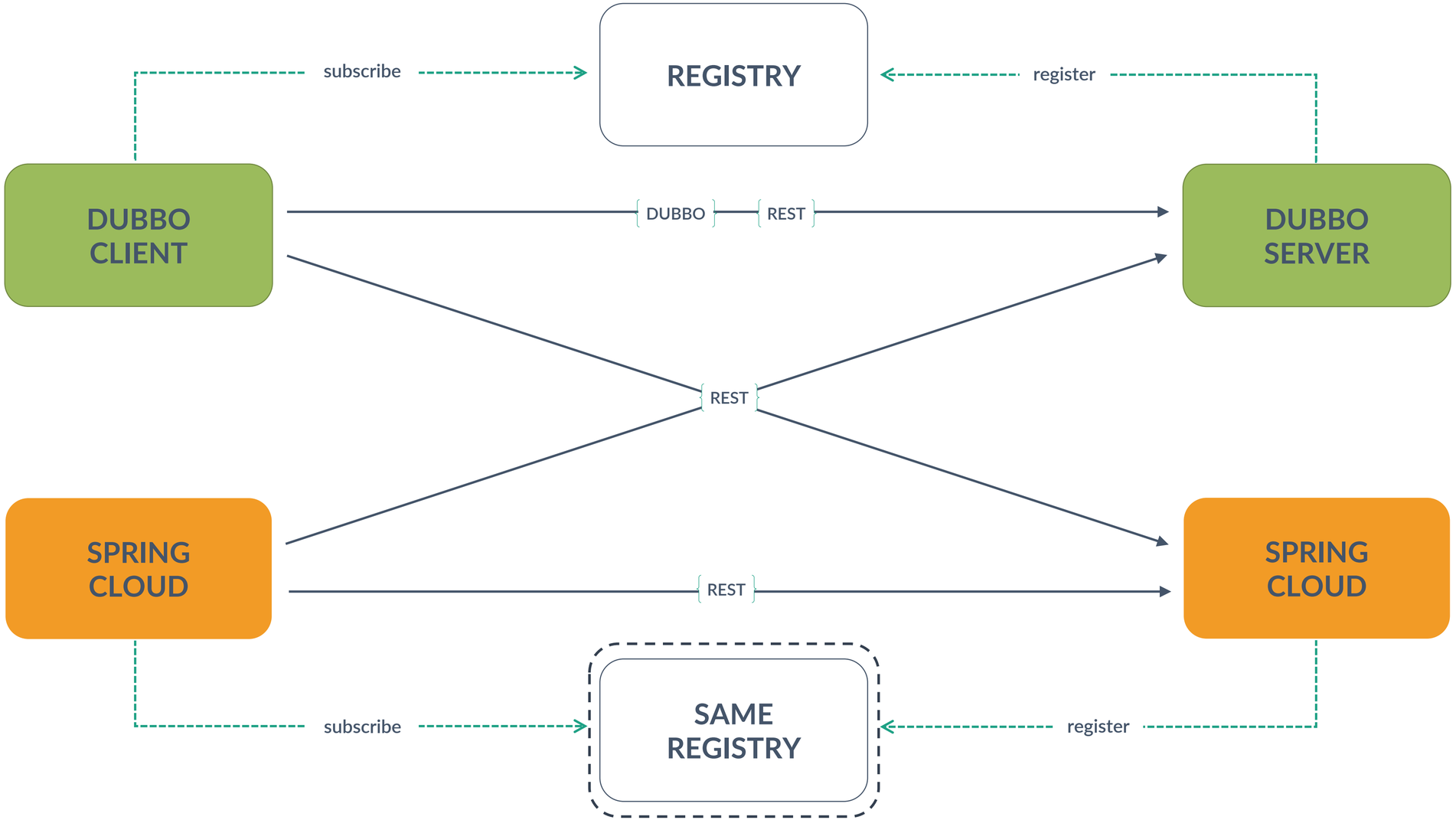


新老模式、不同框架共存

- Dubbo 体系内，新老模式迁移；
- Dubbo 与其他微服务体系
    - Spring Cloud
    - Kubernetes、 ServiceMesh

微服务

与Spring Cloud体系互通



云原生

Dubbo云原生总体目标

Docker容器

容器环境部署Dubbo应用

Kubernetes

支持k8s平台下的两种部署模式：

- 仅将k8s作为Dubbo进程编排调度引擎
- 复用k8s服务管控能力

Service Mesh

Date Plane 和 Control Plane是mesh的两个核心概念。

从Dubbo角度，考虑支持两种模式：

- 有Sidecar
- 无Sidecar

Observability

- Prometheus
- OpenTracing

Stream & Messaging

- gRPC协议支持，框架集成
- 内置HTTP/2协议支持

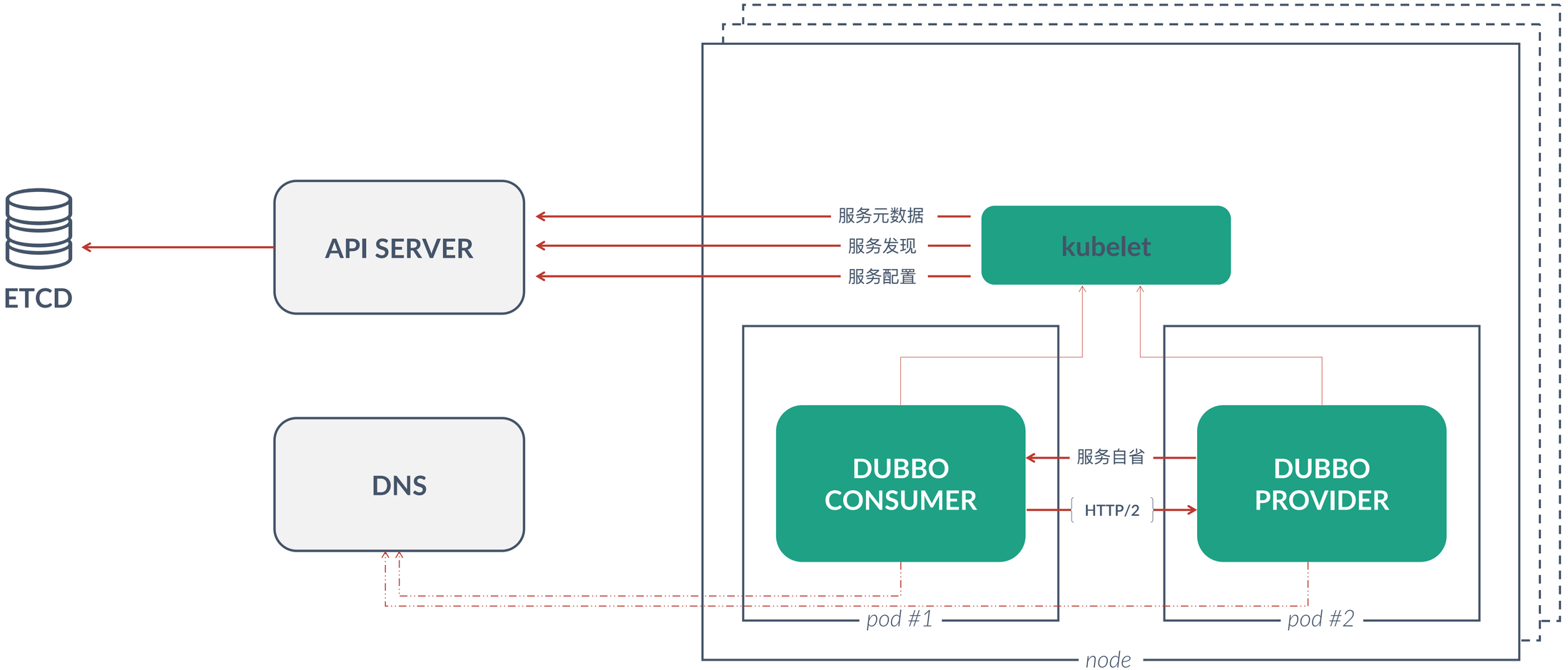
云原生

K8S原生服务发现

	Kubernetes API	DNS Lookup
Description	• 使用 REST API 与 Kubernetes Master 交互，获取 IPs 和 PODs 。 • 使用 Java SDK 与 Kubernetes Maste 交互，获取 IPs 和 PODS 。	对接DNS，获取指定服务下的 IPs 和 PODs
Pros	灵活。 • 任意类型K8S服务 • 单个服务 • 组合多个服务	通用、方便，没有权限等限制。
Cons	访问 Kubernetes Master API 需要认证和权限配置	• Headless Service • 限定单个服务 • 解决 DNS 缓存等问题

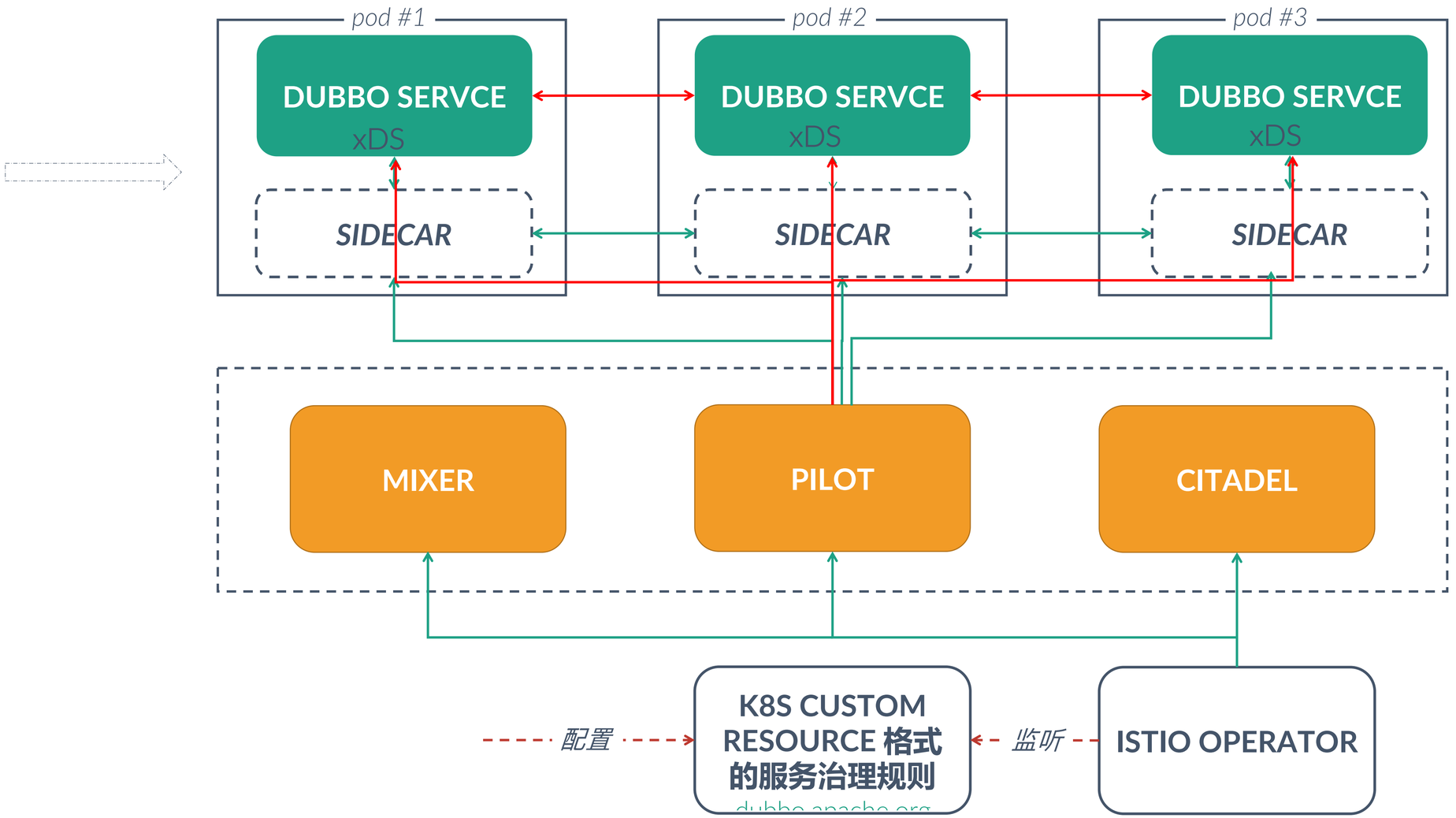
云原生

K8S原生服务发现



云原生

Service Mesh



Thank You

dubbo.apache.org

Home

dubbo.apache.org

GitHub

github.com/apache/dubbo

Mailing List

dev@dubbo.apache.org

IM

钉钉群：21973601