



# What's new in Dubbo 2.7.4

定义 Dubbo Cloud Native 架构

小马哥 (@mercyblitz)

# 自我介绍

- 小马哥 (@mercyblitz)
- 父亲, Java 劝退师, Apache Dubbo PMC、Spring Cloud Alibaba 架构师, 《Spring Boot 编程思想》作者。
- 个人主页: <https://mercyblitz.github.io/>



# 主要议程

- 2.7.4 新特性概览
- 生命周期管理 - 事件机制
- 拥抱云原生 - Cloud Native 注册机制
- 去中心化实现 - “服务自省” 机制
- 下一个里程

## 2.7.4 新特性概览

- 生命周期管理 - 事件机制
- 拥抱云原生 - Cloud Native 注册机制
- 去中心化实现 - “服务自省” 机制





# 生命周期管理 - 事件机制

## Dubbo 事件机制

核心理念：Event/Listener、观察者模式（Observer Pattern）扩展

灵感来源：Java 事件监听机制、Spring Event 抽象、Guava Event BUS

### 接口抽象

事件对象 - `org.apache.dubbo.event.Event`

事件监听器 - `org.apache.dubbo.event.EventListener`

服务发现 - `org.apache.dubbo.event.EventDispatcher`

# 拥抱云原生 - Cloud Native 注册机制

## Dubbo 服务注册与发现基本特点

注册/发现对象 - Dubbo 服务接口

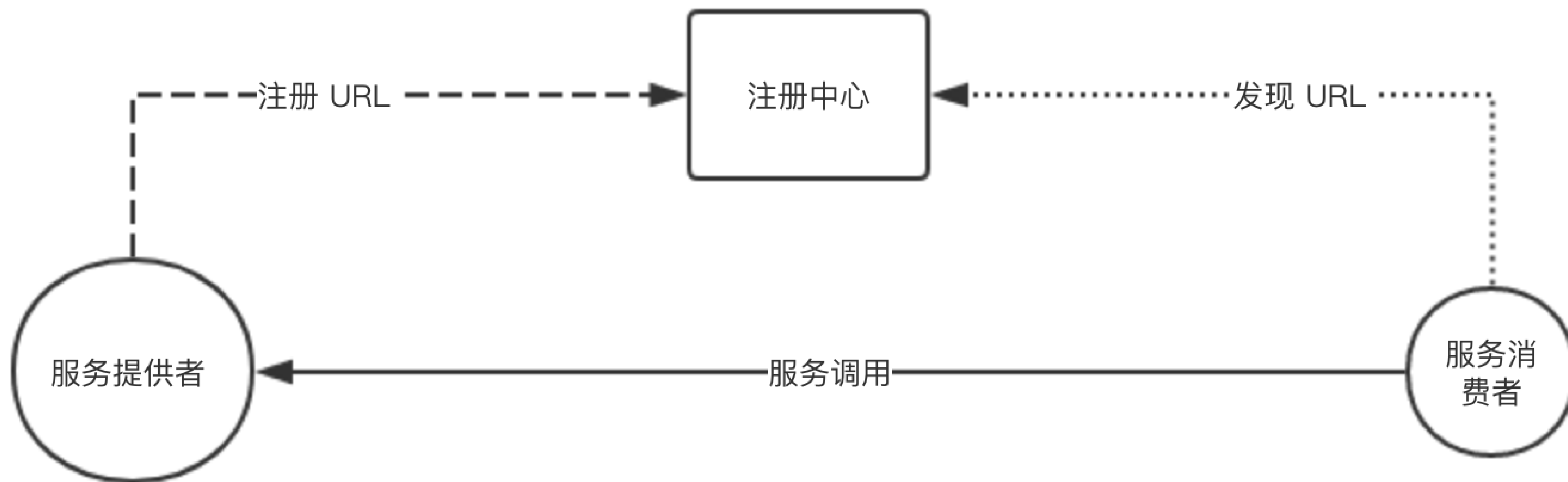
注册/发现载体 - Dubbo URL （元信息：服务接口、版本、分组等）

注册中心



# 拥抱云原生 - Cloud Native 注册机制

## Dubbo 服务注册与发现设计



# 拥抱云原生 - Cloud Native 注册机制

## Dubbo 服务注册与发现面临的挑战

**描述：**在微服务架构中，注册中心管理的对象是**应用（服务）**，而非对外的服务接口，不过目前 Dubbo 的注册中心管理的对象是 **Dubbo 服务接口，与 Spring Cloud 或 Cloud Native 注册方式背道而驰**

延伸问题1:为什么以应用（服务）维度来注册？

延伸问题2：如何适配和兼容 Spring Cloud 以及 K8S 注册中心？

延伸问题3：如果将 Dubbo 服务接口作为 Dubbo 服务注册，是否可以兼顾以上需求



# 拥抱云原生 - Cloud Native 注册机制

## 服务注册与发现改造

核心理念：Cloud Native

灵感来源：Spring Cloud Commons 抽象以及 Curator X Discovery

### 接口抽象

服务实例 - `org.apache.dubbo.registry.client.ServiceInstance`

服务注册 - `org.apache.dubbo.registry.client.ServiceRegistry`

服务发现 - `org.apache.dubbo.registry.client.ServiceDiscovery`

# 去中心化实现 - “服务自省” 机制

## 起源

JavaBeans 自省 - At runtime and in the builder environment we need to be able to figure out which properties, events, and methods a Java Bean supports. We call this process introspection.

## 定义

Dubbo 应用在运行时处理和分析 Dubbo 服务元信息的过程，如当前应用暴露的Dubbo 服务以及各自的通讯协议等。期间会伴随着事件的广播和处理，如服务暴露事件。



# 去中心化实现 - “服务自省” 机制

## Dubbo 服务注册与发现面临的挑战

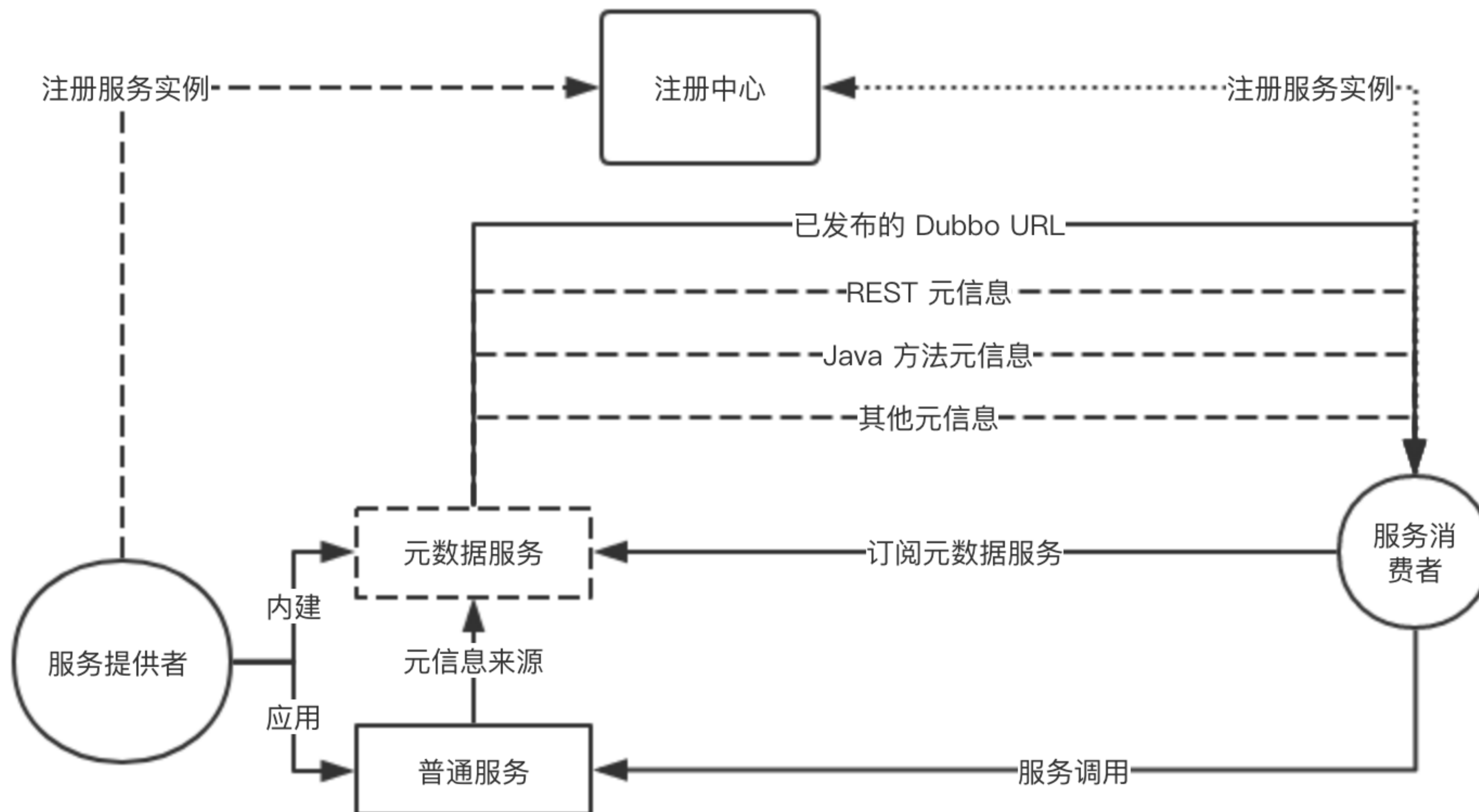
**描述：** 一个Dubbo 应用（服务）允许注册  $N$  个 Dubbo 服务接口，**当  $N$  越大时，注册中心的负载越重**，根据不完全统计，每个微服务应用通常会暴露 20 ~ 50 个服务接口。注册中心是中心化的基础设施，其稳定性**面临严峻考验**。

反之，如果使用应用（服务）粒度注册，注册中心的负载只有过去的  $1/N$  甚至更少（云信息保存更少）



# 去中心化实现 - “服务自省” 机制

## Apache Dubbo 服务自省架构



# 去中心化实现 - “服务自省” 机制

## Apache Dubbo 服务自省实现

- 服务注册与发现改造

- 事件驱动

- 元信息管理





# 去中心化实现 - “服务自省” 机制

## 元数据管理

核心理念：元数据服务是为服务提供元数据的服务

灵感来源：JavaBeans 内省、JMX

## 接口抽象

元数据服务 - `org.apache.dubbo.metadata.MetadataService`

本地元数据 - `org.apache.dubbo.metadata.LocalMetadataService`

服务输出器 - `org.apache.dubbo.metadata.MetadataServiceExporter`

# 去中心化实现 - “服务自省” 机制

## 元信息服务 Facade 接口 - MetadataService

```
public interface MetadataService {

    /** The value of all service names ...*/
    String ALL_SERVICE_NAMES = "*";

    /** The value of All service instances ...*/
    String ALL_SERVICE_INTERFACES = "*";

    /** The contract version of {@link MetadataService}, the future update must make sure compatible. ...*/
    String VERSION = "1.0.0";

    /** Gets the current Dubbo Service name ...*/
    String serviceName();

    /** Gets the version of {@link MetadataService} that always equals {@link #VERSION} ...*/
    default String version() { return VERSION; }

    /** the list of String that presents all Dubbo subscribed {@link URL urls} ...*/
    List<String> getSubscribedURLs();

    /** Get the list of String that presents all Dubbo exported {@link URL urls} ...*/
    default List<String> getExportedURLs() { return getExportedURLs(ALL_SERVICE_INTERFACES); }

    /** Get the list of String that presents the specified Dubbo exported {@link URL urls} by the <code>serviceInterface</code> ...*/
    default List<String> getExportedURLs(String serviceInterface) { return getExportedURLs(serviceInterface, group: null); }

    /** Get the list of String that presents the specified Dubbo exported {@link URL urls} by the ...*/
    default List<String> getExportedURLs(String serviceInterface, String group) {...}

    /** Get the list of String that presents the specified Dubbo exported {@link URL urls} by the ...*/
    default List<String> getExportedURLs(String serviceInterface, String group, String version) {...}

    /** Get the list of String that presents the specified Dubbo exported {@link URL urls} by the ...*/
    List<String> getExportedURLs(String serviceInterface, String group, String version, String protocol);

    /** Interface definition. ...*/
    String getServiceDefinition(String interfaceName, String version, String group);

    /** Interface definition. ...*/
    String getServiceDefinition(String serviceKey);

    /** Convert the multiple {@link URL urls} to a {@link List list} of {@link URL urls} ...*/
    static List<URL> toURLs(Iterable<String> urls) {...}
}
```

# 如何实现“服务自省”

## 元信息服务 Facade 接口 - MetadataService

### 核心方法

`getExportedURLs()` - 获取当前 Dubbo 应用发布的 Dubbo 服务 URL (多重载方法)

`getSubscribedURLs()` - 获取当前 Dubbo 应用订阅的 Dubbo 服务 URL

`version()` - 当前 MetadataService 接口的版本, 用于兼容性测试

`serviceName()` - 当前 Dubbo 应用 (服务) 名称

# 如何实现“服务自省”

## 暴露 MetadataService

元信息

服务接口 - `org.apache.dubbo.metadata.MetadataService`

服务版本 (version) - `version()` 方法

服务分组 (group) - `serviceName()` 方法

通讯协议 (protocol) - `dubbo` (默认)、自定义



# 下一个里程

## 基于 Apache Dubbo Cloud Native 特性

Kubernetes

gGRC

分布式服务网关

服务文档化

Dubbo 服务治理平台 - Dubbo Admin







I WANT YOU



小马哥



扫一扫上面的二维码图案，加我微信





Thank you !