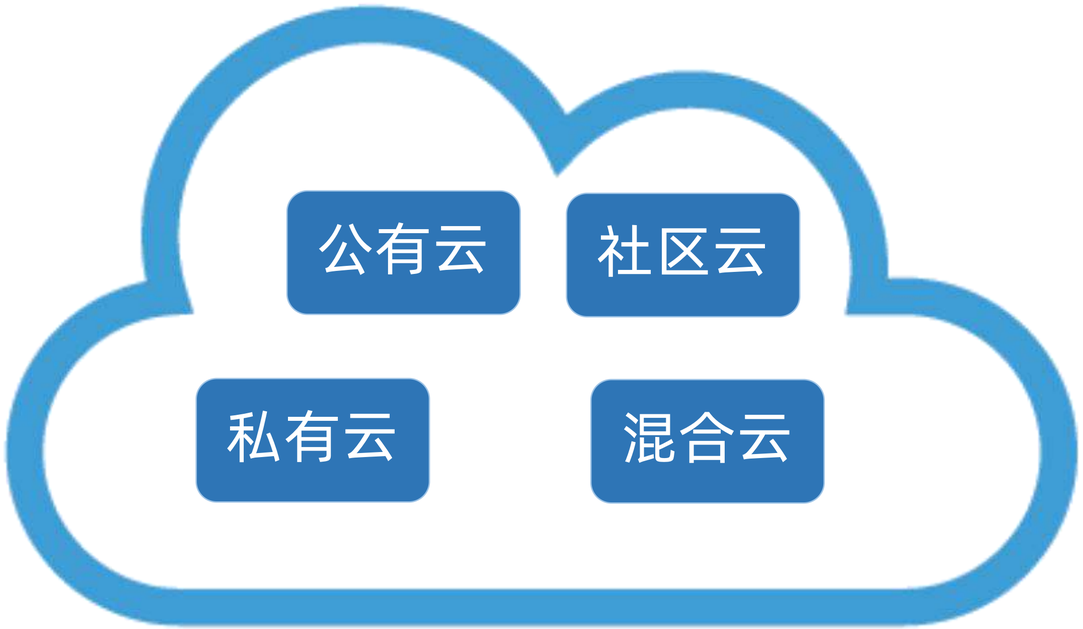
黄云坤

《云原生与Dubbo 3.0》

目录

什么是云原生

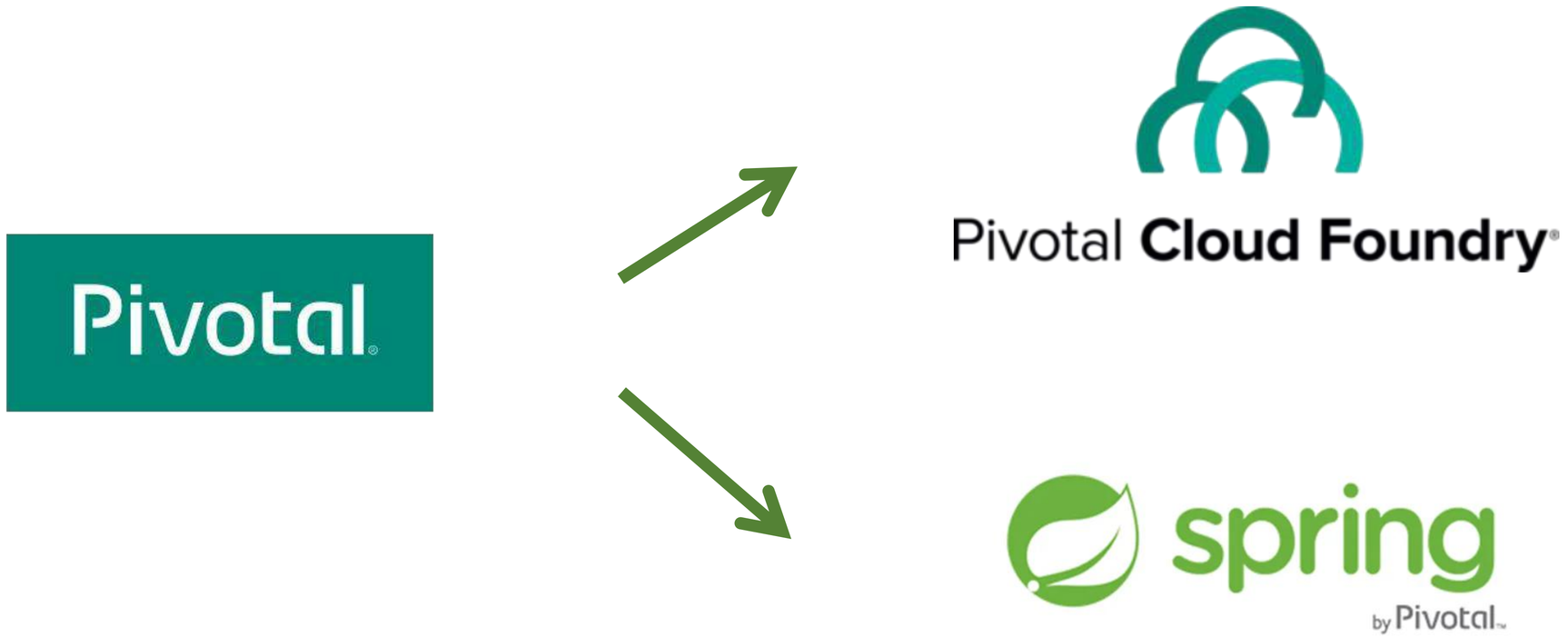
云原生



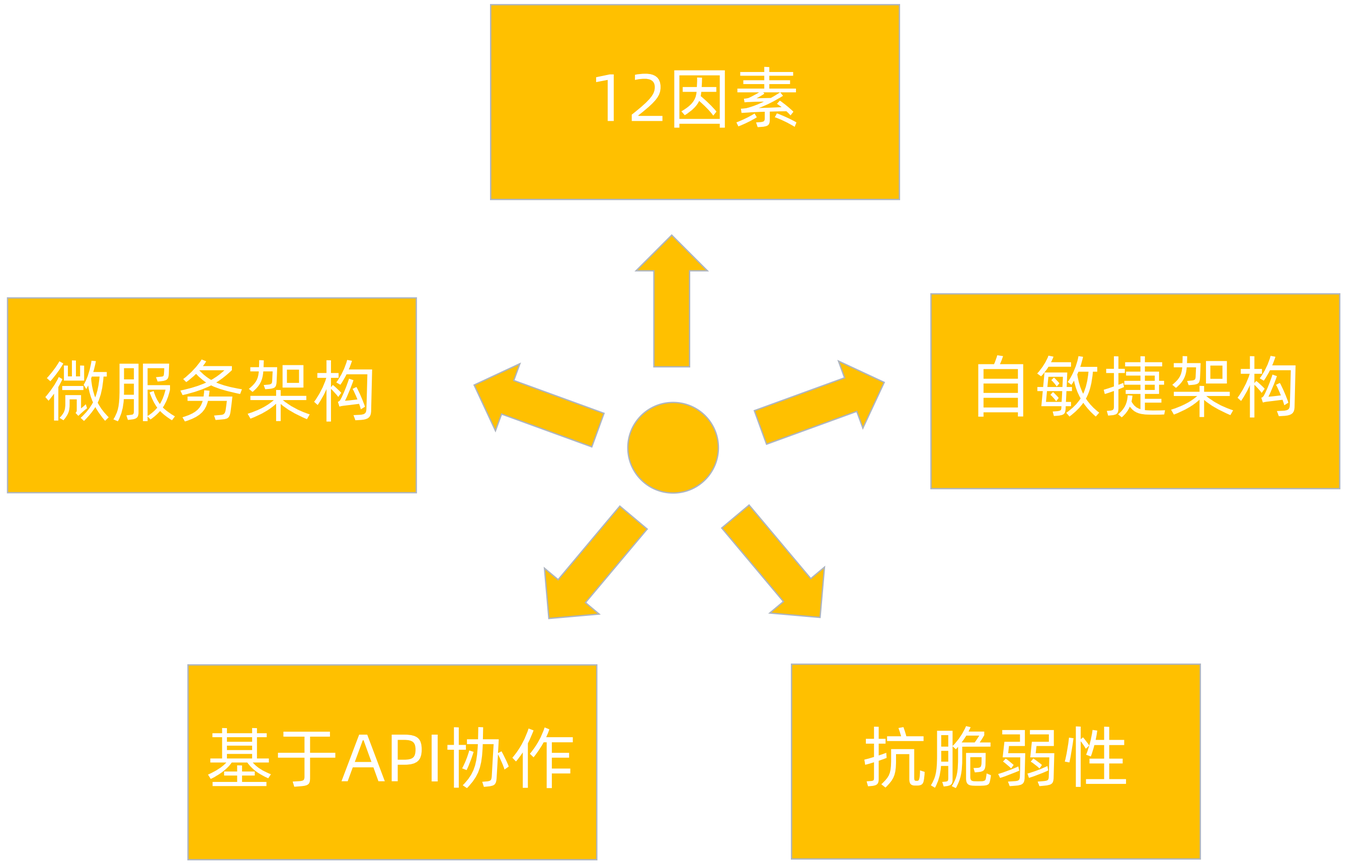
更好的适应云

什么是云原生

Pivotal是Cloud Native/云原生的提出者，是云原生的先驱者和探路者。

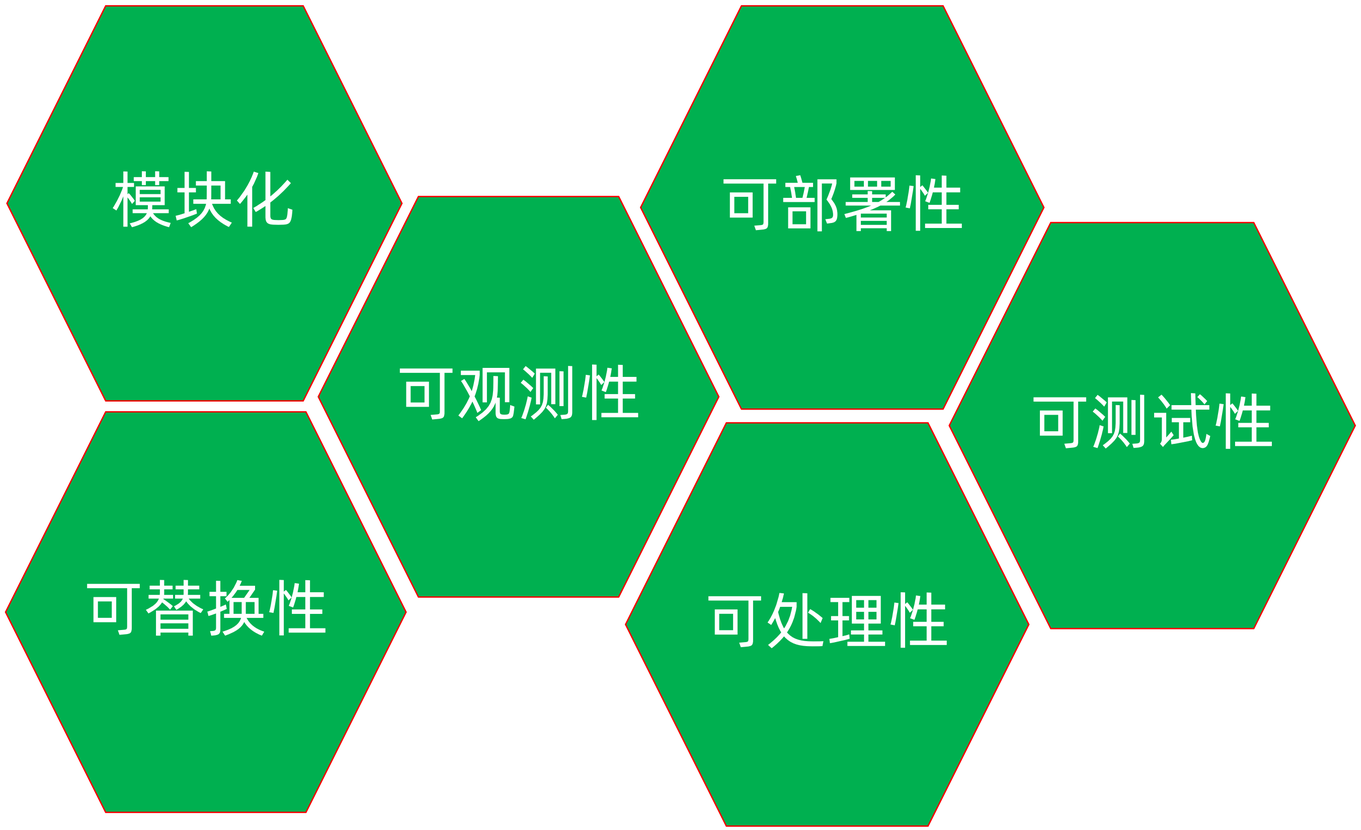


什么是云原生

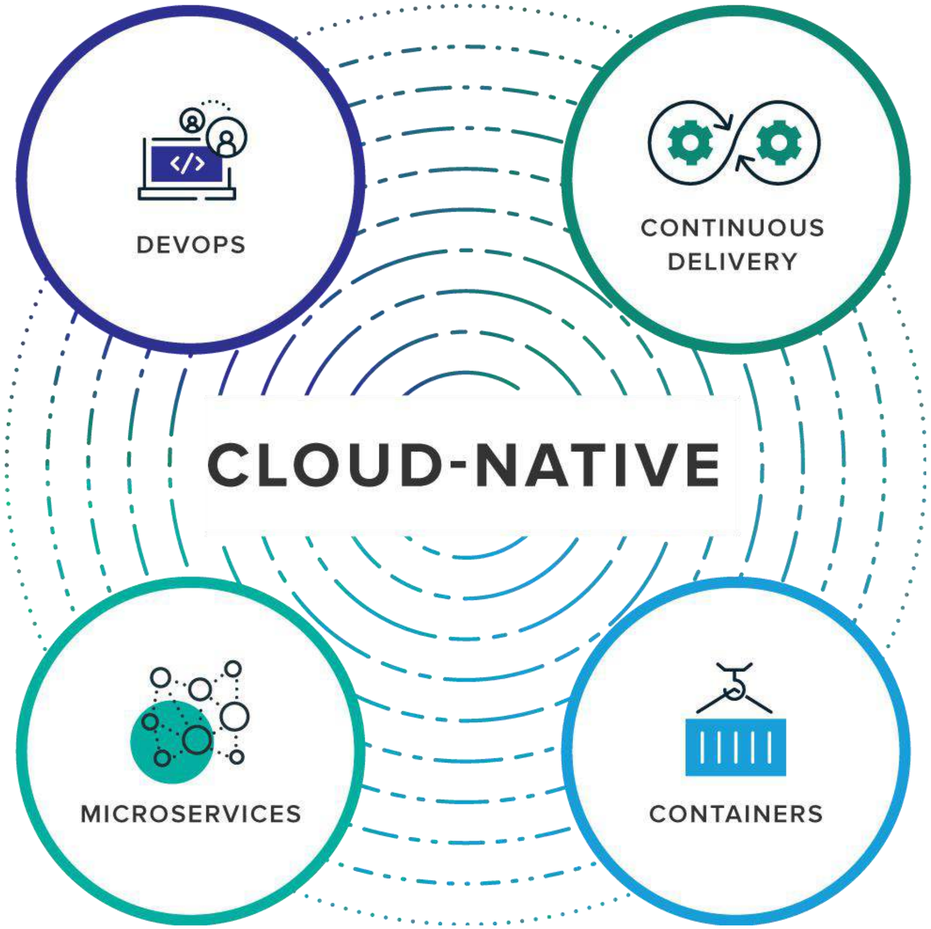


2015年《迁移到云原生应用架构》

什么是云原生



2017年 InfoQ访谈



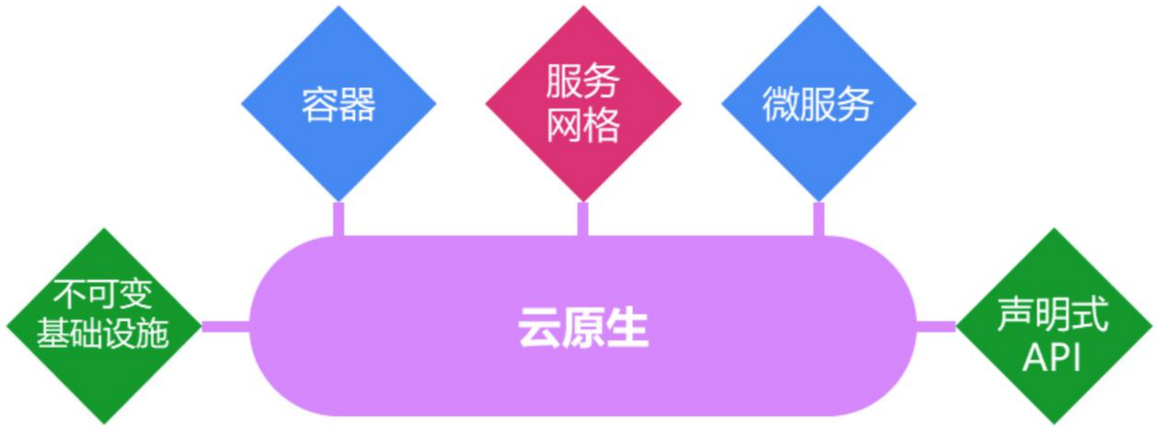
现在 Pivotal官网

什么是云原生

2015年CNCF建立，目标是围绕云原生的概念打造云原生生态体系

应用容器化
面向微服务架构
应用支持容器的编排调度

2018年



什么是云原生

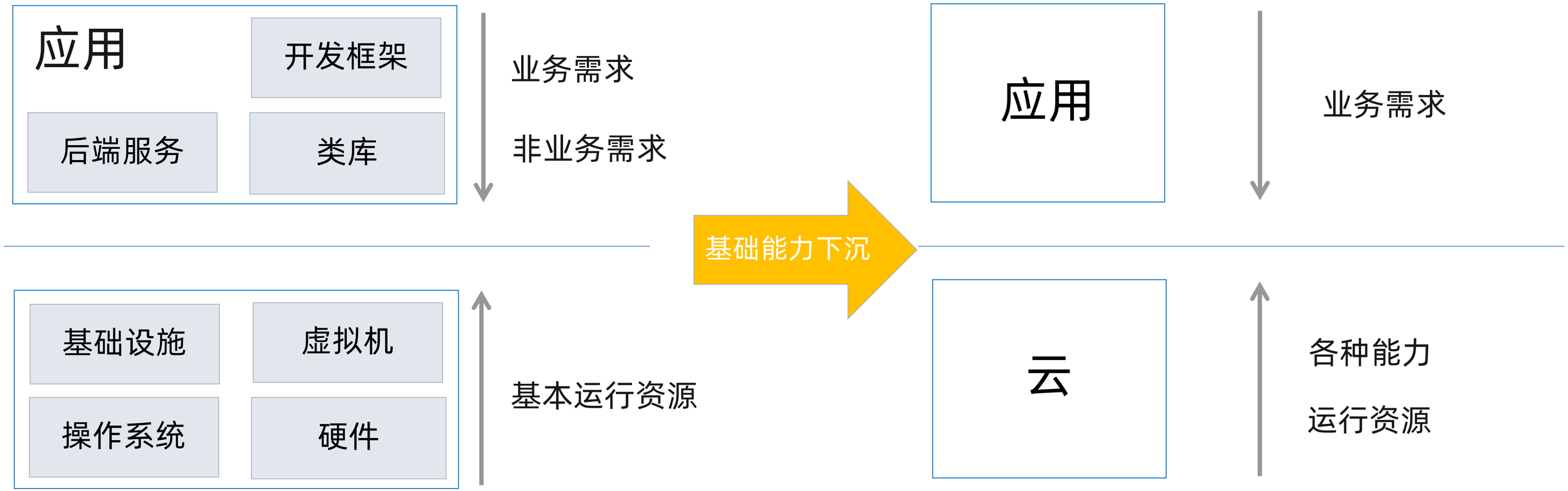
千人千面

不断变化的定义

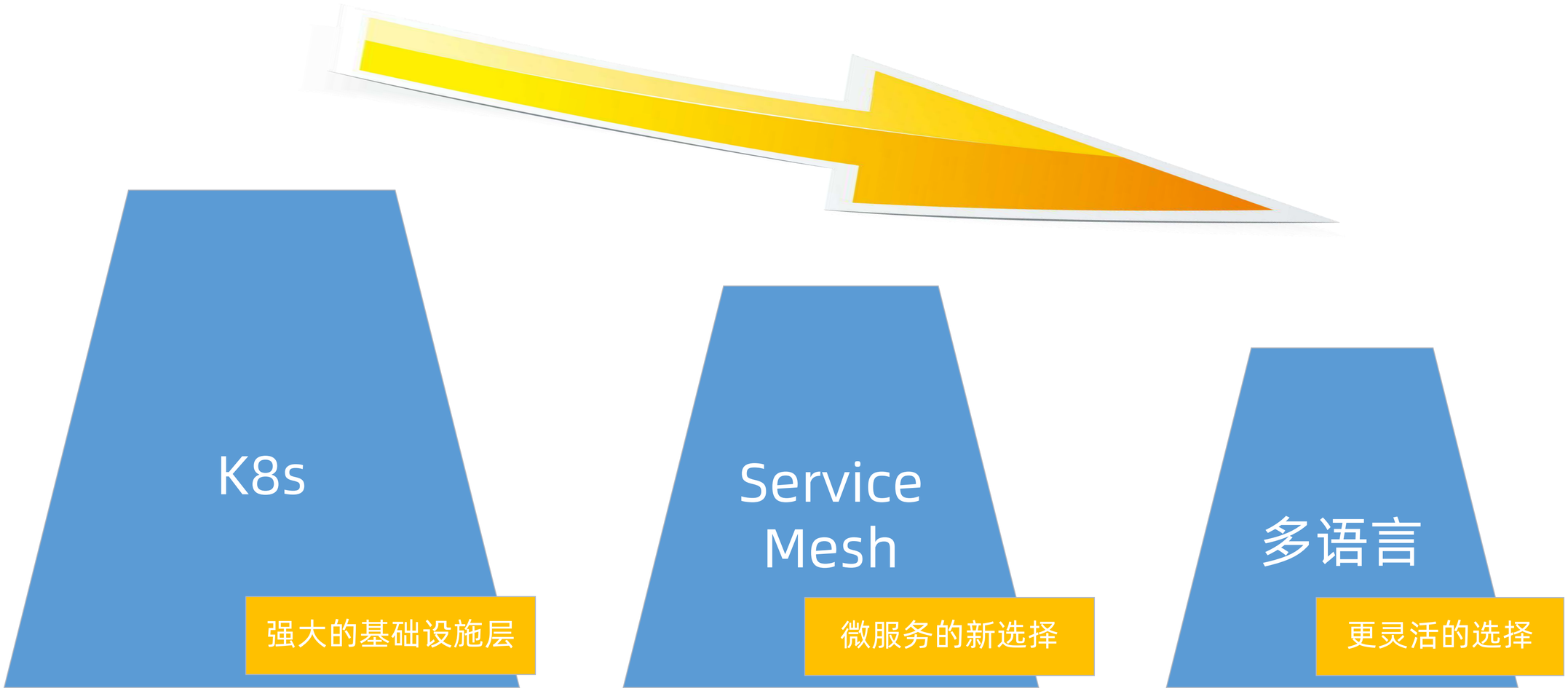
云原生带来的变革

关键目标

云原生带来的变革


应用
开发框架
后端服务
类库
业务需求
非业务需求
基础能力下沉
应用
业务需求
基础设施
虚拟机
操作系统
硬件
基本运行资源
云
各种能力
运行资源

云原生带来的变革



目录

Dubbo的云原生变革

用户为什么使用Dubbo?

Dubbo的云原生变革

功能特性

服务发现
负载均衡
流量治理
链接管理

面向接口快速易用

强类型约束
高度可扩展性
高性能私有RPC协议
生产验证

发力早

2011年项目开源
积累大量用户

规模大

生产环境规模大
迁移/切换不易

技术上：满足诉求
业务上：持续演进

Dubbo的云原生变革

下沉基础设施 → 服务注册下沉
服务生命周期接口
Mesh对接
Mesh共存

通讯协议与数据传输 → 通用性、网关友好性
多语言友好
Stream语义

服务治理能力 → 统一的路由规则
流量调度
可观测性

Dubbo的云原生变革

当前发布核心功能

- 全新的服务发现模型
- 下一代基于 HTTP/2 的 RPC 协议：Triple
- 服务治理重构：全新路由规则

即将交付的功能

- Kubernetes Service 模型适配

调研或开发阶段的功能

- Proxyless Mesh
- 可观测性
- 基于的反压的智能流量调度策略
- Native Image

全新的服务发现模型

consumer

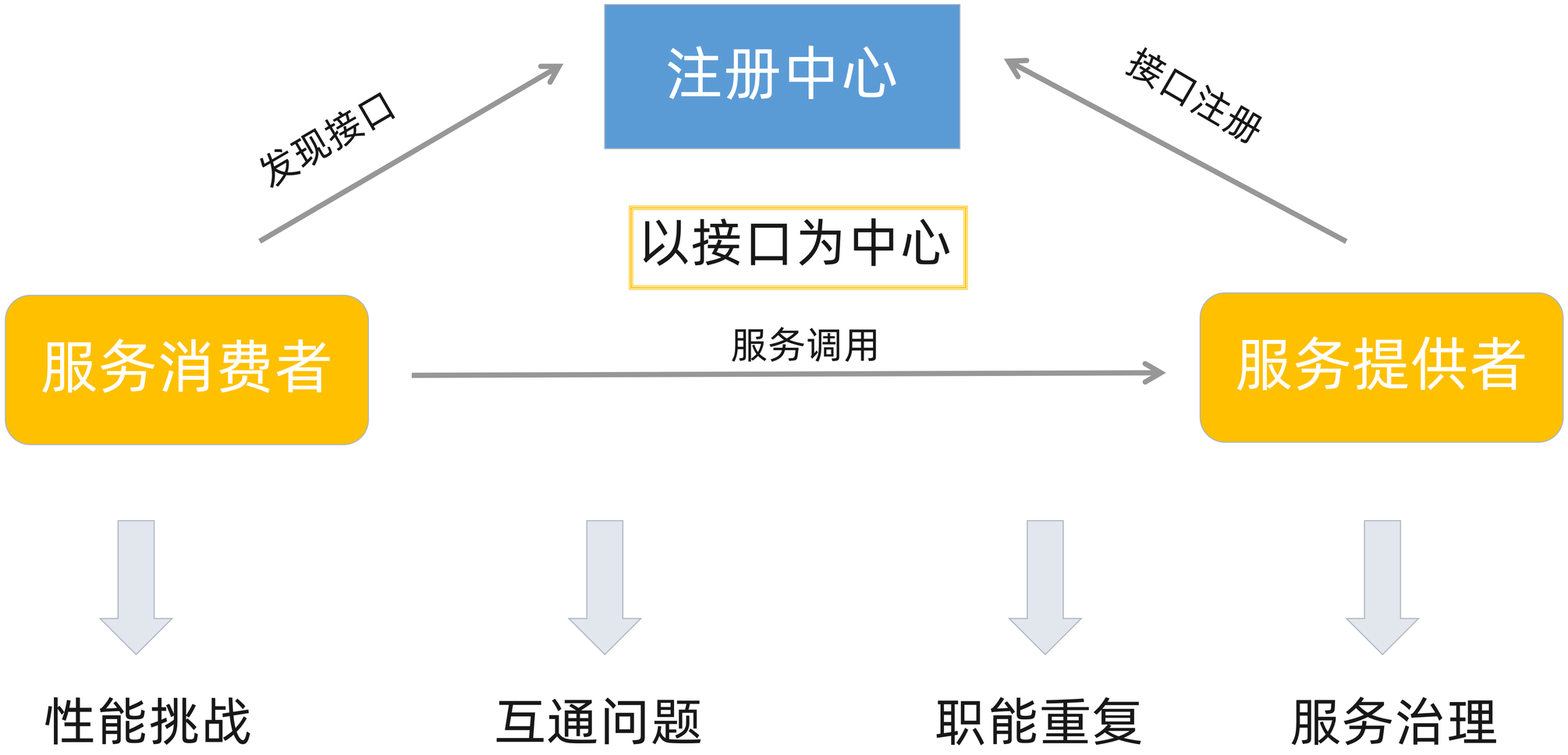
provider

provider

provider

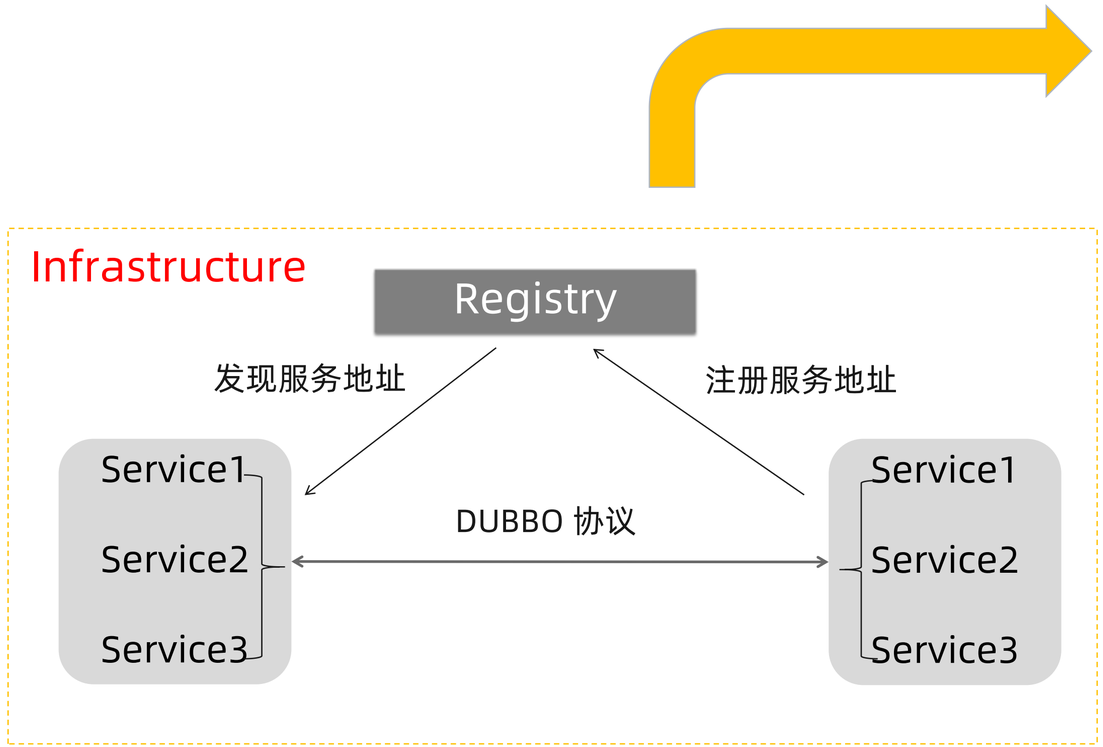
1. service name
2. method
3. args
4. protocol
5. version
6. ……

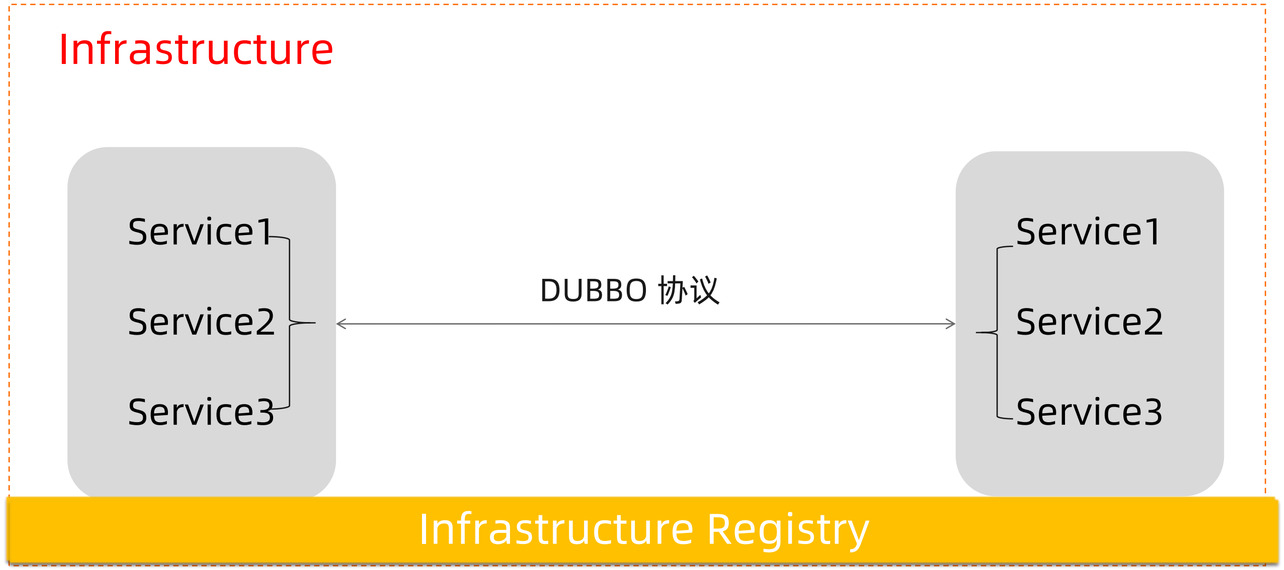
全新的服务发现模型



全新的服务发现模型

基础设施融入了微服务概念的抽象。
容器化微服务被编排、调度的过程，即完成了在基础设施层面的注册。
Dubbo 关注业务 RPC 服务的感知与通信





以应用为中心

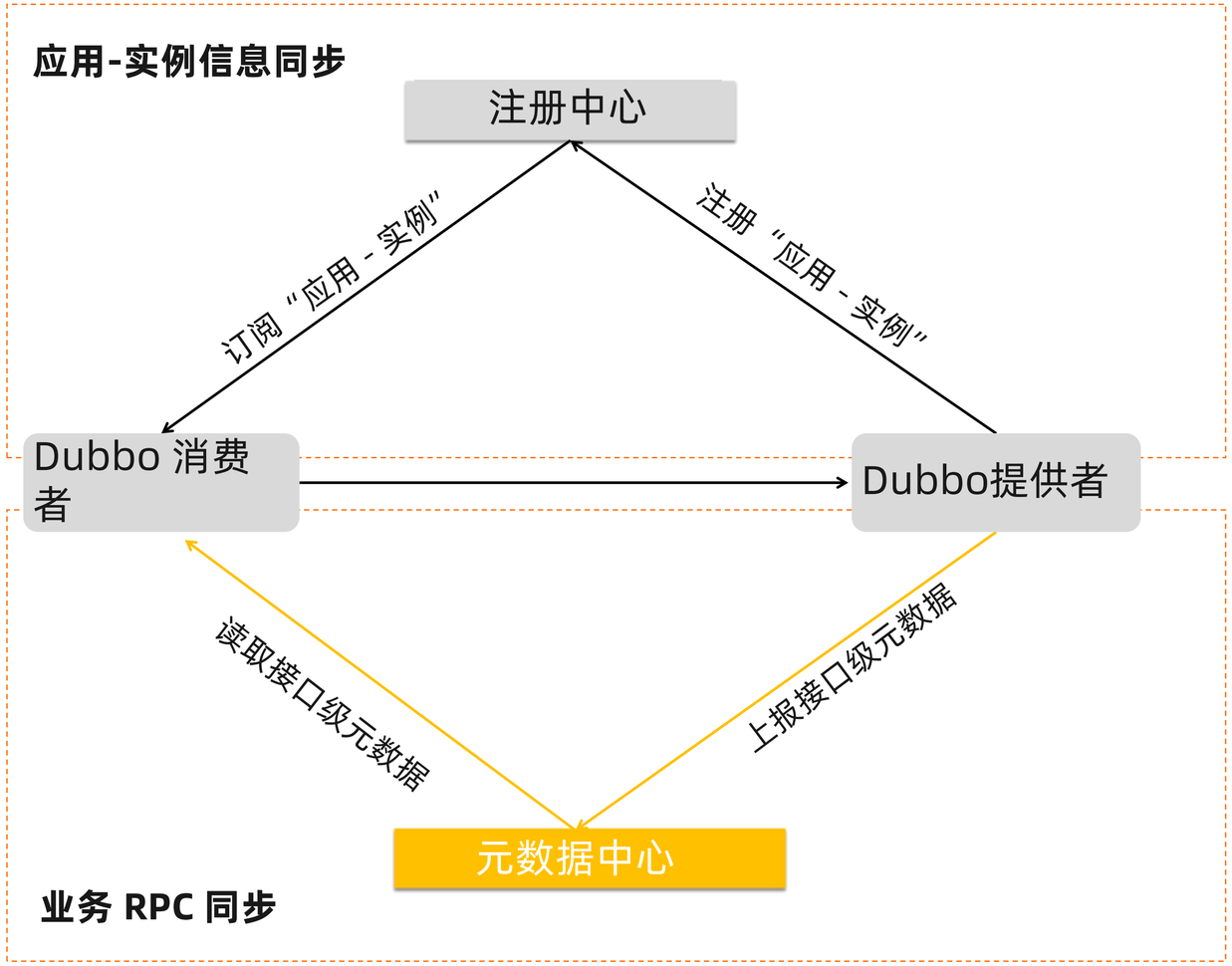
全新的服务发现模型

服务发现只关注应用和实例

- 服务注册

业务域 RPC 元数据同步

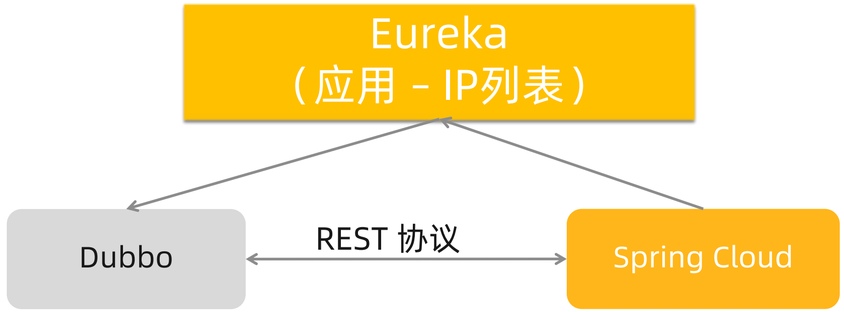
- Dubbo 配置、规则都是接口级的，不能假定所有实例对等
- 通过元数据中心实现配置同步
- 注册中心、元数据中心、配置中心可以共用



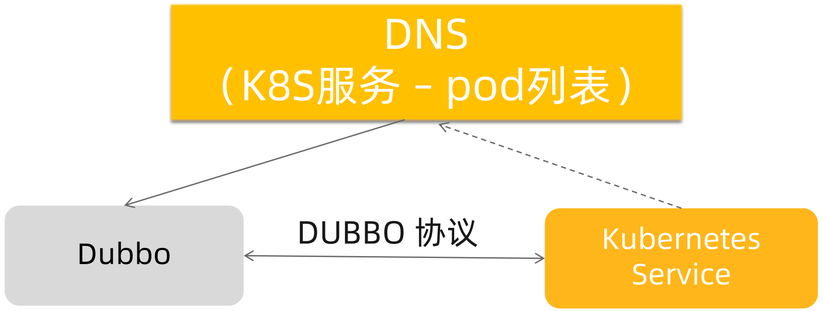
全新的服务发现模型

一、对齐基础设施和主流方案

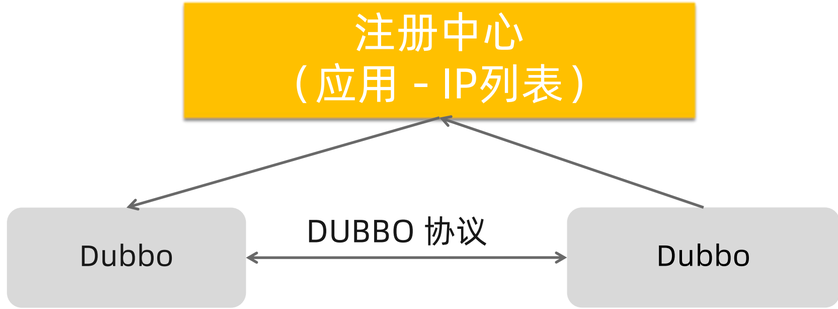
Spring Cloud



K8S Service



二、性能 - 可伸缩服务体系



平均场景

50 接口 * N 实例 ⟶ 20 应用 * N 实例 ↓60%

极限场景

10k+ 接口 * N 实例 ⟶ 2K 应用 * N 实例 ↓90%

Triple协议

协议是两个网络实体进行通信的基础 ⟺ 字节流和数据结构的转换

Dubbo Protocol

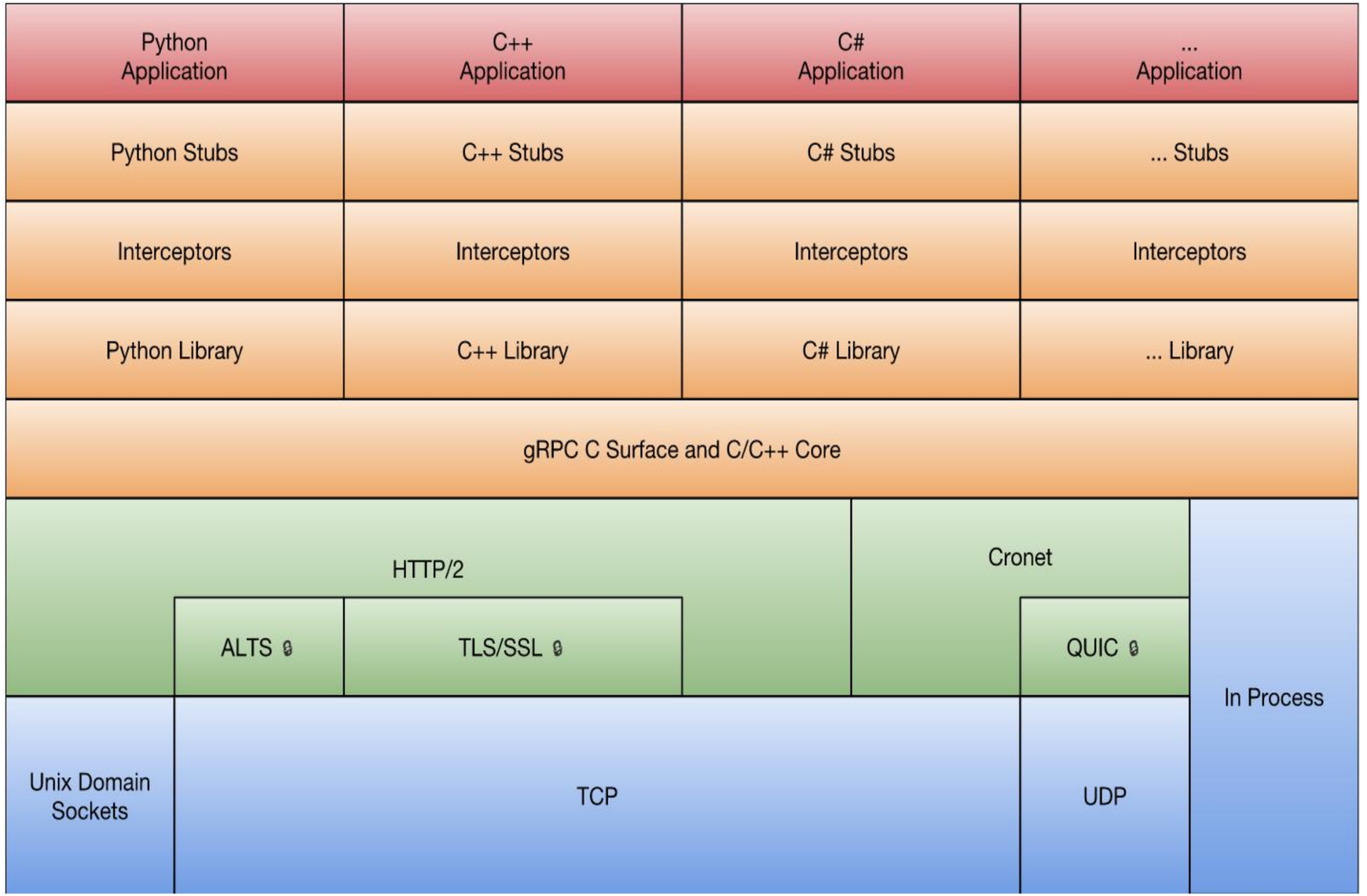
Offsets Octet		0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Magic High								Magic Low								Req/Res	2 Way	Event	Serialization ID					Status							
4	32	RPC Request ID																															
8	64																																
12	96	Data Length																															
16 ...	128 ...	Variable length part, in turn, is: dubbo version, service name, service version, method name, parameter types, arguments, attachments																															

基于'裸'TCP

1. 通用性不足
2. 协议穿透性不强
3. 对多语言实现不友好

Triple协议

1. 基于HTTP/2
2. Protobuf序列化
3. 多路复用
4. 生态认可度高



Triple协议

TRI:// 协议 - 主力协议

01 网关友好

基于 HTTP/2，协议头承载服务元信息

02 多语言友好

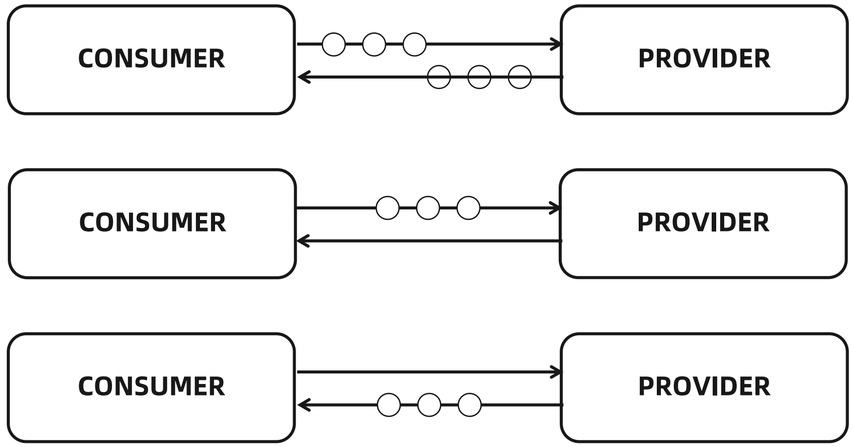
协议体基于 PB，多语言支持开始变得友好

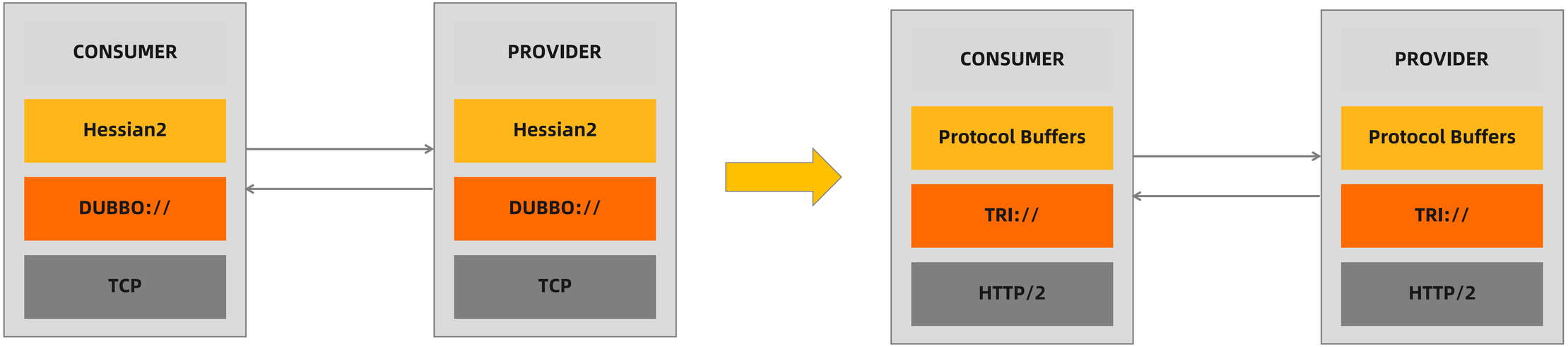
03 支持流式

基于流式包装出多种会话模型，支持更多业务场景

04 反压支持

协议头支持服务端负载反馈，Rx 语义的基础





统一的路由规则

没有什么问题不是一层 Proxy 解决不了的

Service Mesh

Istio

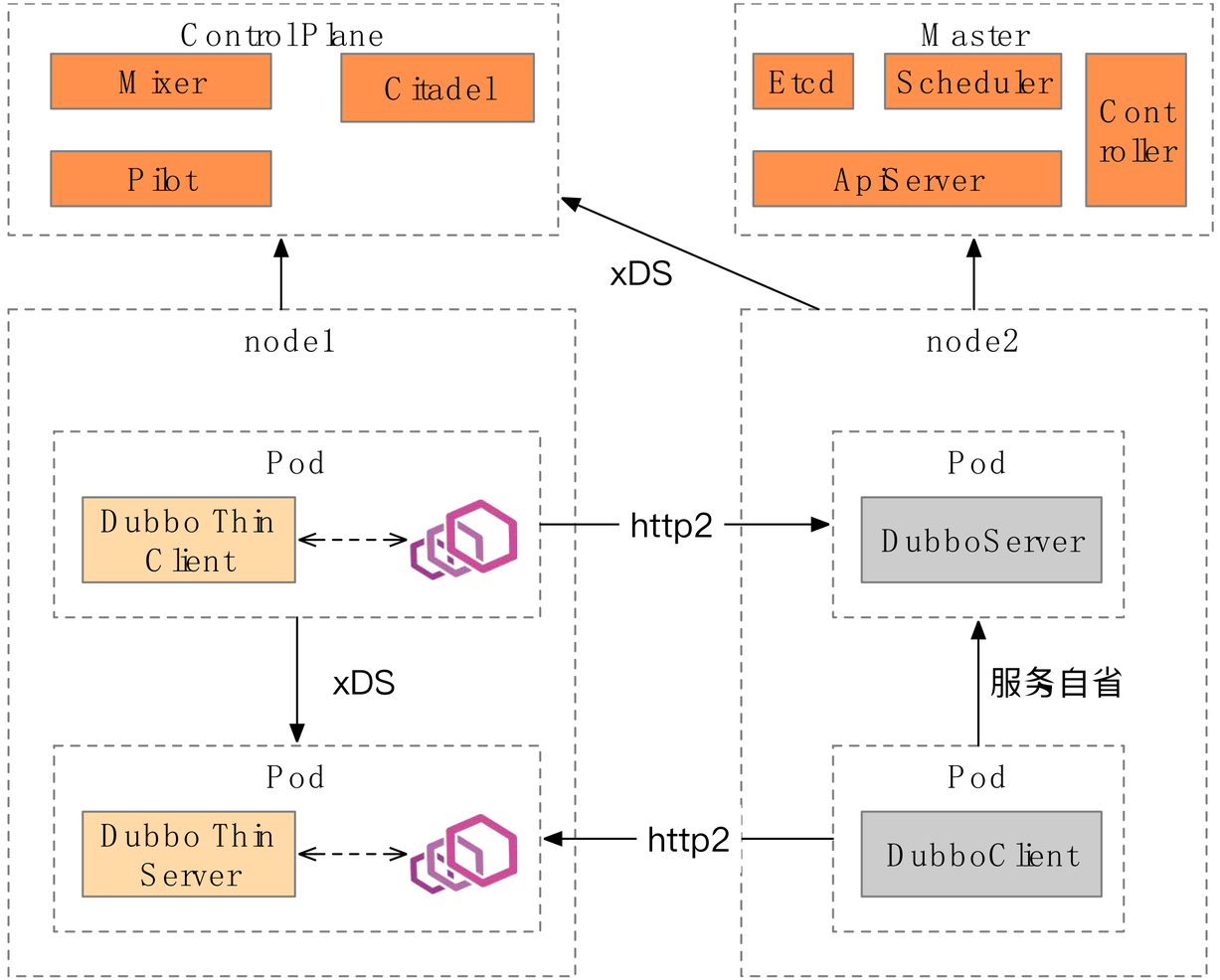
Proxyless Mesh

统一的路由规则

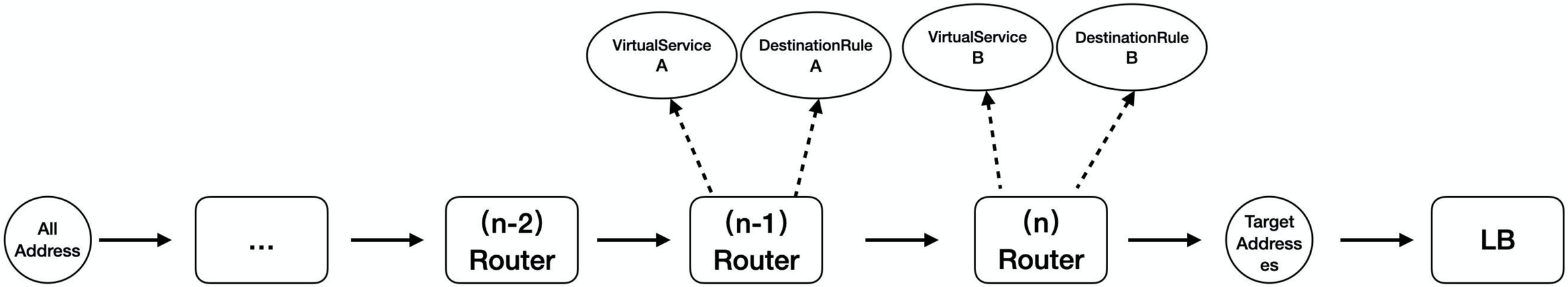
路由规则（Proxyless Mesh）

1. 富 SDK 模式的微服务不可替代，不是所有业务都需要Service Mesh
2. Service Mesh 架构优势明显，但现阶段也有缺陷，整体较重，普遍缺乏大规模落地经验

1. Dubbo3 进行协议升级、轻量化、可插拔
2. 探索 Proxyless Mesh、统一的路由规则。



统一的路由规则



基于路由链，采用 Pipeline 的处理方式

统一的路由规则

VirtualService -> DestinationRule

```yaml
apiVersion: service.dubbo.apache.org/v1alpha1
kind: DestinationRule
metadata:
  name: demo-route
spec:
  host: demo
  subsets:
    - name: v1
      labels:
        sigma.ali/mg: v1-host

    - name: v2
      labels:
        sigma.ali/mg: v2-host

    - name: v3
      labels:
        sigma.ali/mg: v3-host
```

```yaml
apiVersion: service.dubbo.apache.org/v1alpha1
kind: VirtualService
metadata:
  name: demo-route
spec:
  hosts:
    - demo
  dubbo:
    - service:
        - exact: com.taobao.hsf.demoService:1.0.0
      routedetail:
        - name: sayHello-String-method-route
          match:
            - method:
                name_match:
                  exact: "sayHello"
                argp:
                  - string
          route:
            - destination:
                host: demo
                subset: v1
              fallback:
                destination:
                  host: demo
                  subset: v2
                fallback:
                  destination:
                    host: demo
                    subset: v3

        - name: sayHello-method-route
          match:
            - method:
                name_match:
                  exact: "s-method"
          route:
            - destination:
                host: demo
                subset: v2
              fallback:
                destination:
                  host: demo
                  subset: v3

        - name: interface-route
          route:
            - destination:
              host: demo
              subset: v3
```

目录

使用Dubbo3.0

新应用

```xml
<dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo</artifactId>
    <version>3.0.0.preview</version>
</dependency>
<dependency>
    <groupId>com.google.protobuf</groupId>
    <artifactId>protobuf-java</artifactId>
    <version>3.11.0</version>
</dependency>
<dependency>
    <groupId>org.apache.curator</groupId>
    <artifactId>curator-recipes</artifactId>
    <version>2.13.0</version>
</dependency>
```

1 必要依赖引入

```xml
<beans>
    <dubbo:application name="newstarter-provider"/>
    <dubbo:protocol id="tri"/>
    <dubbo:config-center address="zookeeper://127.0.0.1:2181"/>
    <bean id="newStarterProvider"
          class="com.huangyunkun.cnd3.NewStarterProvider"/>
    <dubbo:service interface="com.huangyunkun.cnd3.IService"
                   ref="newStarterProvider" protocol="tri"/>
</beans>
```

2 指定协议为tri

平滑迁移到Dubbo3.0

```xml
<dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo</artifactId>
    <version>2.7.10</version>
</dependency>
```

```xml
<dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo</artifactId>
    <version>3.0.0.preview</version>
</dependency>
```

1 升级版本依赖

```xml
<dubbo:protocol id="tri"/>
<dubbo:protocol id="dubbo"/>
```

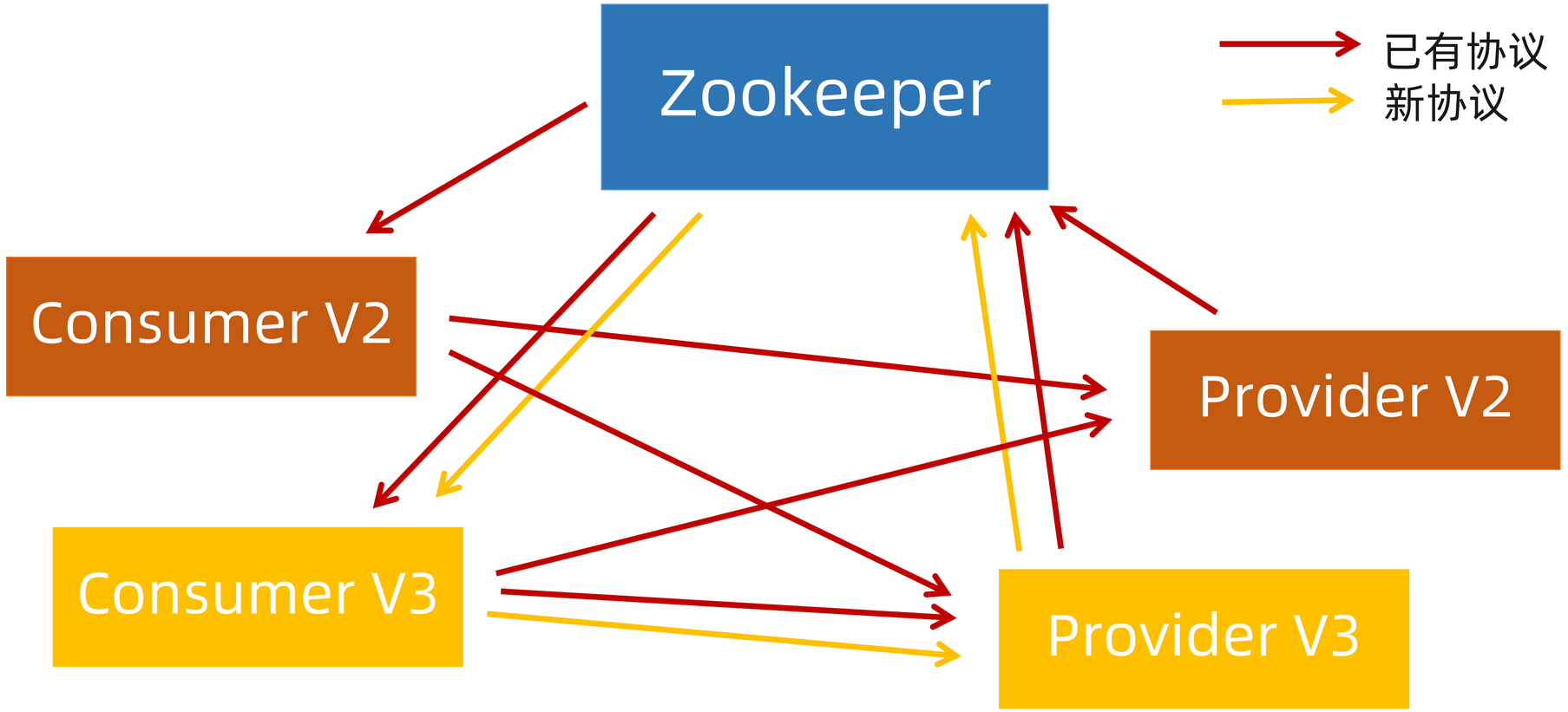
2 添加新协议

平滑迁移到Dubbo3.0

1.双协议注册

2.兼容Dubbo2

3.协议自动选择



目录

DUBBO 3.0 路线图

3.0 preview

Dubbo3 云原生架构升级

应用级服务发现、

Triple 、新路由规则

2021/03

3.0 正式版本

阿里、社区用户标杆案例

社区生产可用，文档、用例完备；

2021/06

3.1 preview1

云原生：官方 Kubernetes 部署方案

Kubernetes Service

2021/7

3.1 preview2

云原生：官方 Mesh 解决方案

Proxyless、ThinSDK

2021/9

3.1 正式版本

Dubbo3 全面完成云原生支持

社区生产可用，寻求更多社区案例

2021/11

3.2 正式版本

智能流量调度，提高系统稳定性与资源利用率

服务柔性

2022/3

Thanks

谢谢观看！