

# 从微服务 到服务网格

一起感受未来架构之旅

20

- 07/15

秦金卫 (kimmking)

Apache Dubbo/ShardingSphere PMC



# 个人介绍



- ① Apache Dubbo/ShardingSphere PMC, 前某集团高级技术总监/阿里架构师/某银行北京研发中心负责人, 阿里云MVP、腾讯TVP、TGO会员
- ② 10多年研发管理和架构经验, 关注于互联网, 电商, 金融, 支付, 区块链等领域, 熟悉海量并发低延迟交易系统的设计实现
- ③ 对于中间件、SOA、微服务, 以及各种开源技术非常热衷, 活跃于多个开源社区
- ④ 《高可用可伸缩微服务架构: 基于Dubbo、Spring Cloud和ServiceMesh》、《JVM核心技术32讲》合著作者
- ⑤ 个人博客: <http://kimmking.github.io>
- ⑥ 联系方式: [kimmking@apache.org](mailto:kimmking@apache.org)

# 目录

## CONTENTS

1

### 微服务概述

MICROSERVICE SUMMARY

2

### 最佳实践

BEST PRACTICE

3

### 服务网格

SERVICE MESH

4

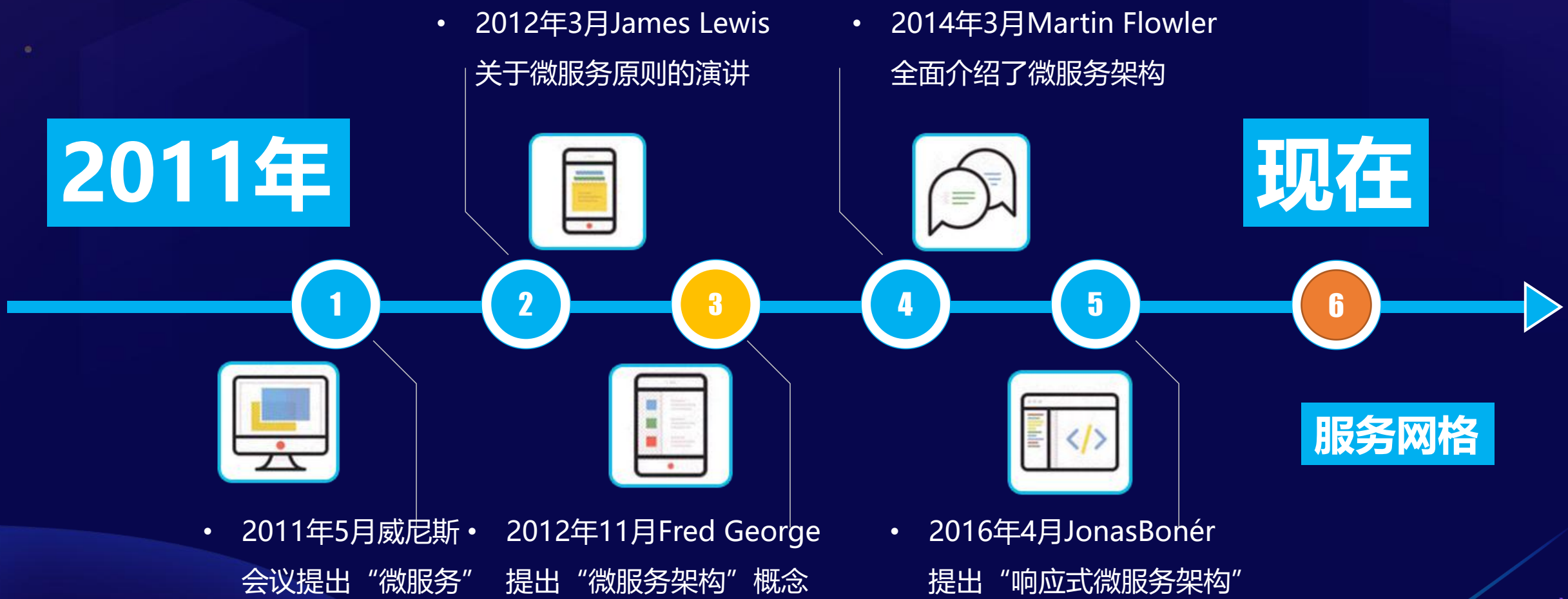
### 走向未来

GO TO FUTURE



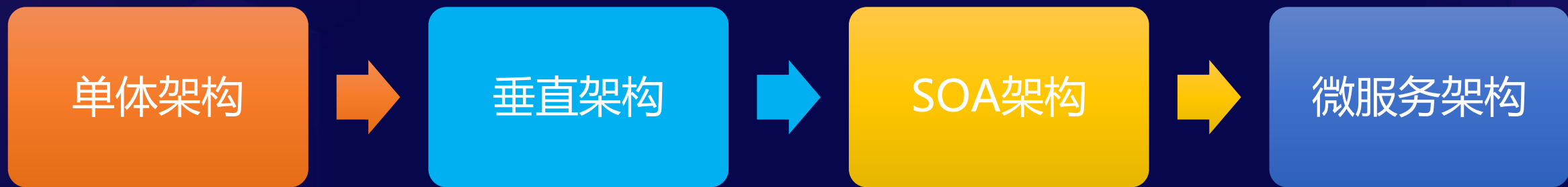


# 一、微服务概述--微服务发展历程





## 一、微服务概述--架构发展阶段



### 单体架构

简单单体模式是最简单的架构风格，所有的代码全都在一个项目中。这样研发团队的任何一个人都可以随时修改任意的一段代码，或者增加一些新的代码。



### 垂直架构

分层是一个典型的对复杂系统（不仅仅是软件）进行结构化思考和抽象聚合的通用性办法，也符合金字塔原理。MVC是一个非常常见的3层（3-Tier）结构架构模式。



### 面向服务架构

面向服务架构（SOA）是一种建设企业IT生态系统的架构指导思想。SOA的关注点是服务。服务最基本的业务功能单元，由平台中立性的接口契约来定义。



### 微服务架构

微服务架构风格，以实现一组微服务的方式来开发一个独立的应用系统的方法。其中每个小微服务都运行在自己的进程中，一般采用HTTP资源API这样轻量的机制相互通信。



## 一、微服务概述--微服务特点

### 恰当拆分

按照业务和一定的粒度进行拆分服务，DDD

### 独立部署

每个服务独立部署，拥有自己的数据和状态

### 自动化管理

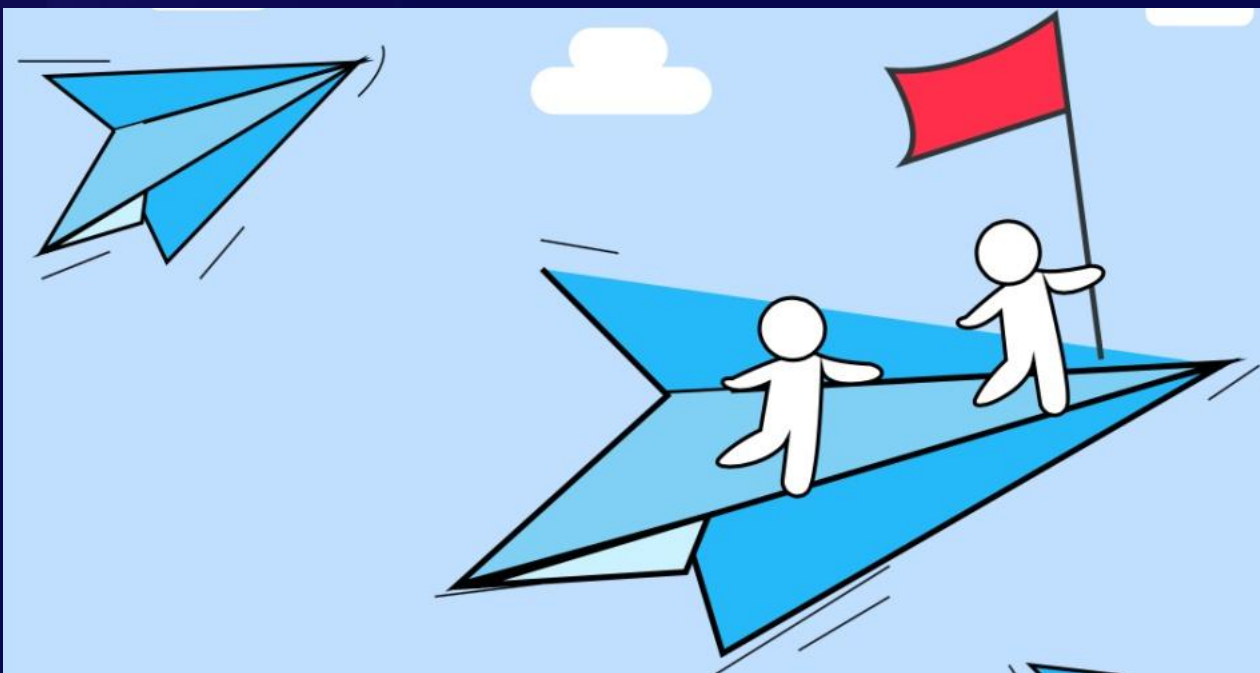
使用自动化的方式测试、运维，做好监控

### 统一治理

进行统一的注册、发现、路由、降级等



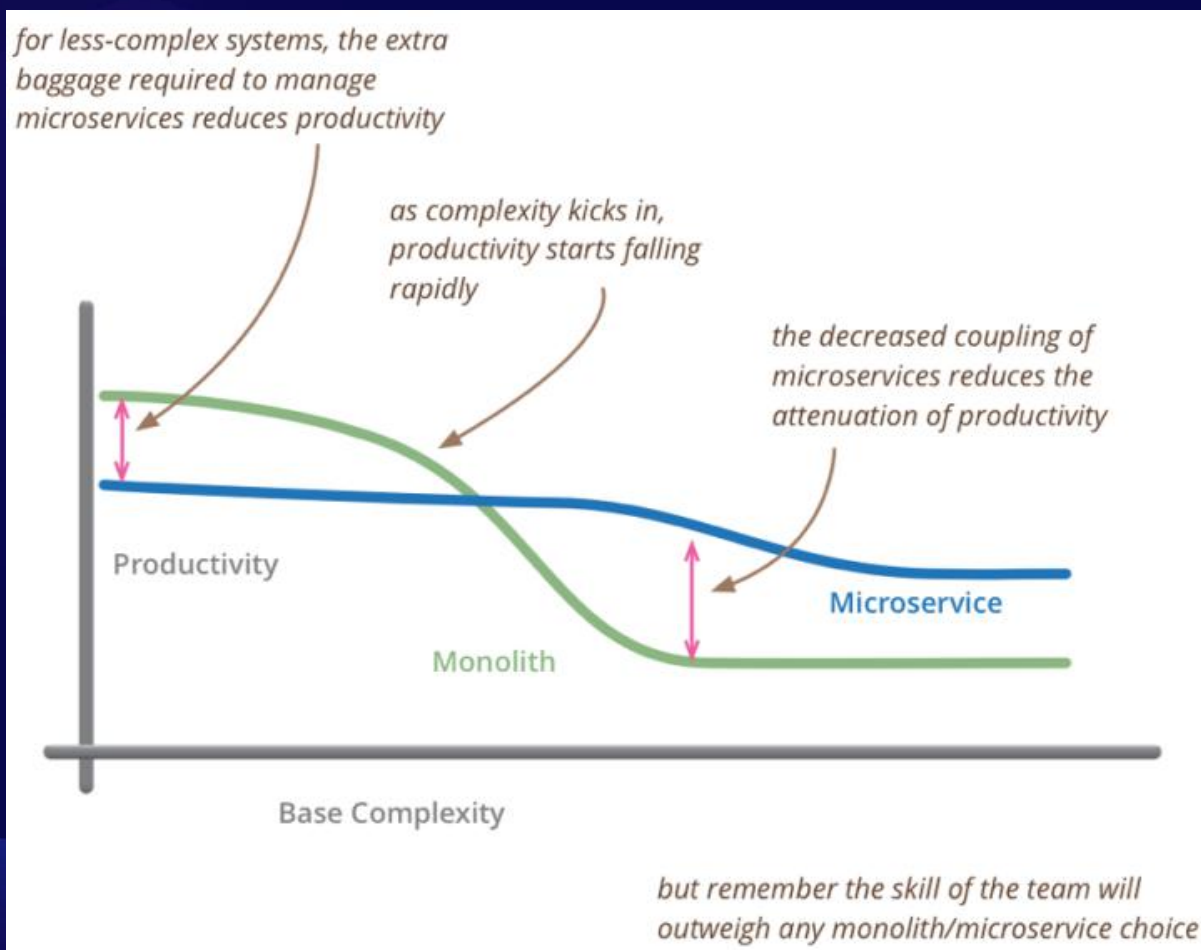
## 一、微服务概述--微服务优势



- 一. 服务简单
- 二. 灵活扩展
- 三. 便于维护
- 四. 独立演进
- 五. 混合开发
- 六. 持续交付



## 一、微服务概述--业务复杂度与研发效率

















- ✓ 微服务应用在复杂度低的情况下，生产力反而比单体架构低
- ✓ 在复杂度高的地方，情况恰恰相反
- ✓ 随着复杂度升高，单体架构的生产力快速下降，而微服务相对平稳，为什么？

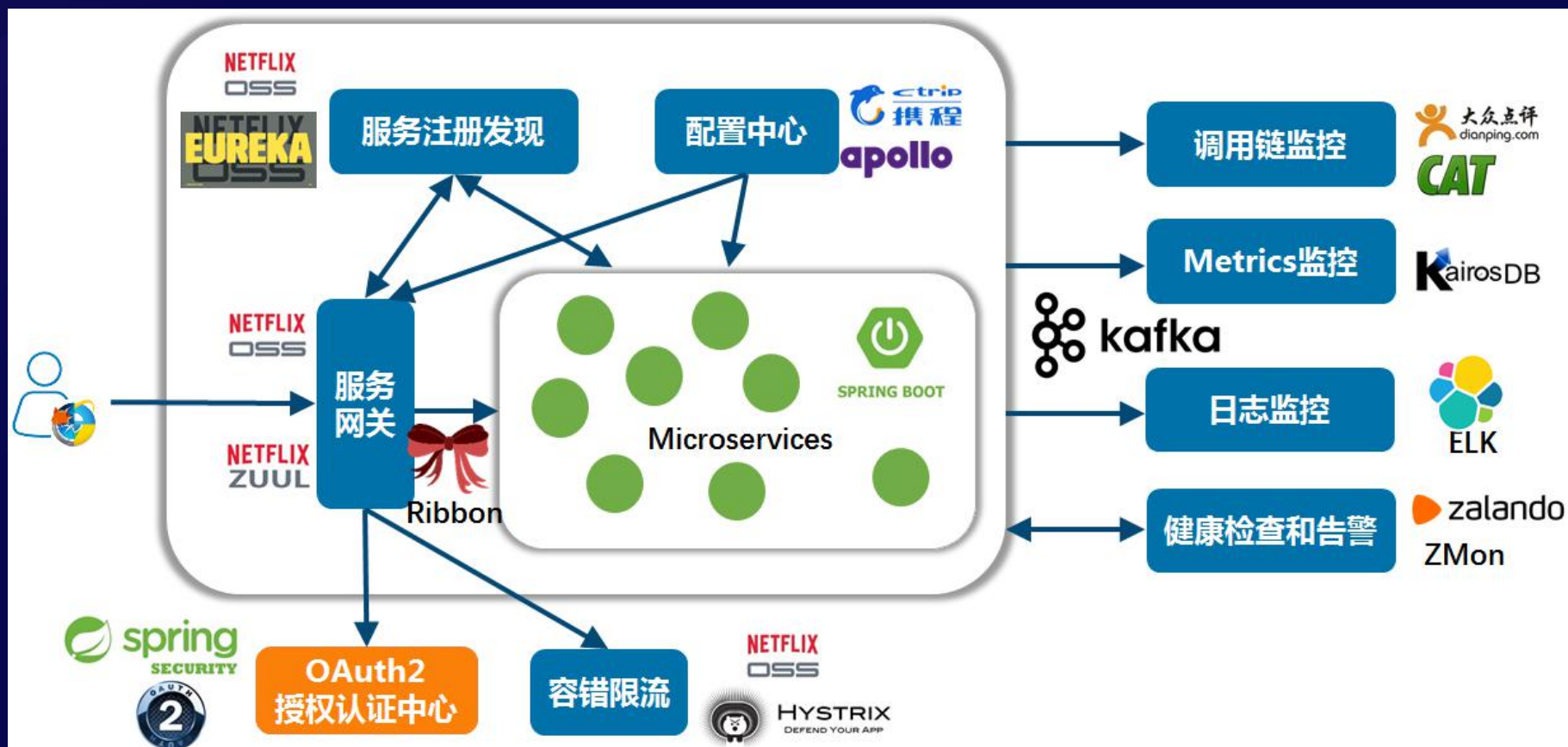


# 一、微服务概述--微服务相关框架

## Spring Cloud生态

 <b>Spring Cloud Config</b> Spring 配置管理工具包, 让你可以把配置放到远程服务器, 集中化管理集群配置。目前支持本地存储、Git以及Subversion。	 <b>Spring Cloud Bus</b> Spring 事件、消息总线, 用于在集群(例如, 配置变化事件)中传播状态变化。可与Spring Cloud Config联合实现热部署。	 <b>Eureka</b> Netflix 云端服务发现, 一个基于 REST 的服务。用于定位服务, 以实现云端中间层服务发现和故障转移。	 <b>Hystrix</b> Netflix 熔断器, 容错管理工具, 旨在通过熔断机制控制服务和第三方库的节点, 从而对延迟和故障提供更强大的容错能力。	 <b>Zuul</b> Netflix Zuul 是在云平台上提供动态路由, 监控, 弹性, 安全等边缘服务的框架。Zuul 相当于是设备和 Netflix 流应用的 Web 网站后端所有请求的前门。	 <b>Archaius</b> Netflix 配置管理API, 包含一系列配置管理API, 提供动态类型化属性、线程安全配置操作、轮询框架、回调机制等功能。	 <b>Spring Cloud Starters</b> Pivotal Spring Boot式的启动项目, 为Spring Cloud提供开箱即用的依赖管理。
 <b>Consul</b> HashiCorp 封装了Consul操作, consul是一个服务发现与配置工具, 与Docker容器可以无缝集成。	 <b>Spring Cloud Sleuth</b> Spring 日志收集工具包, 封装了Dapper和log-based追踪以及Zipkin和HTrace操作, 为SpringCloud应用实现了一种分布式追踪解决方案。	 <b>Spring Cloud Zookeeper</b> Spring 操作Zookeeper的工具包, 用于使用zookeeper方式的服务发现和配置管理。	 <b>Spring Cloud Stream</b> Spring 数据流操作开发包, 封装了与Redis, Rabbit, Kafka等发送接收消息。	 <b>Ribbon</b> Netflix 提供云端负载均衡, 有多种负载均衡策略可供选择, 可配合服务发现和断路器使用。	 <b>Feign</b> OpenFeign Feign是一种声明式、模板化的HTTP客户端。	 <b>Spring Cloud Cluster</b> Spring 提供Leadership选举, 如: Zookeeper, Redis, Hazelcast, Consul等常见状态模式的抽象和实现。

## 一、微服务概述--微服务相关技术



## 二、最佳实践

### 最佳实践之一：遗留系统的微服务改造



- ① 功能剥离、数据解耦
- ② 自然演进、逐步拆分
- ③ 小步快跑、快速迭代
- ④ 灰度发布、谨慎试错
- ⑤ 提质量线、还技术债



## 二、最佳实践

### 最佳实践之二：如何做恰当粒度的拆分

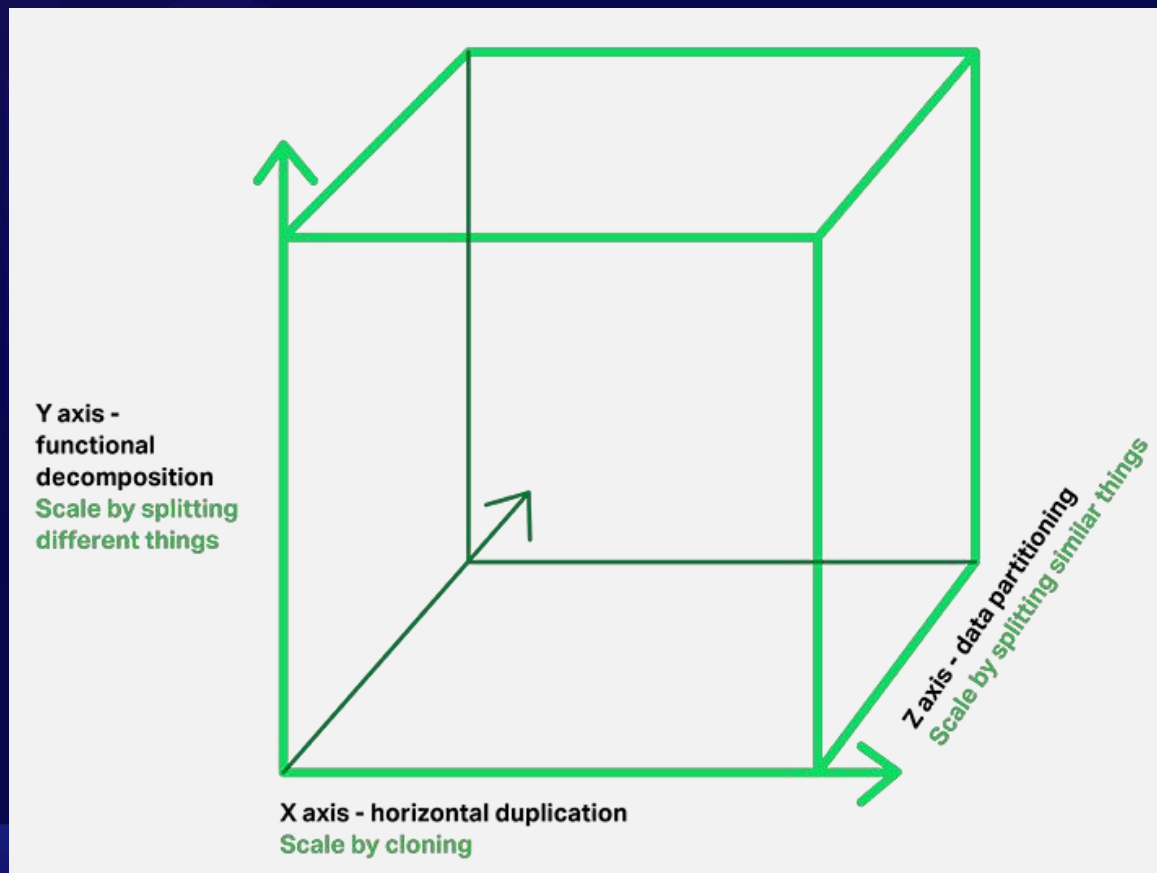


拆分原则：

1. 高内聚低耦合
2. 不同阶段拆分要点不同

## 二、最佳实践

### 最佳实践之三：如何扩展微服务系统



扩展立方体：

1. 水平复制：复制系统
2. 功能解耦：拆分业务
3. 数据分区：切分数据

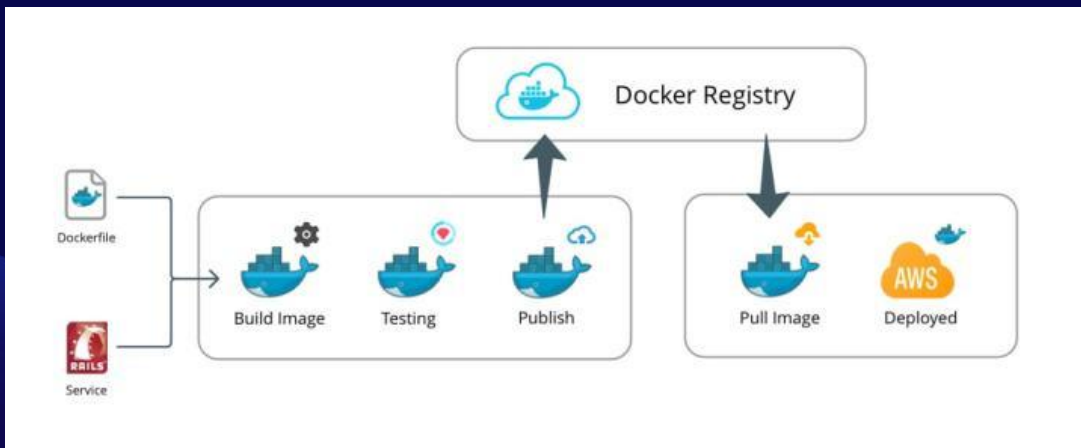
## 二、最佳实践

### 最佳实践之四：如何提升研发效率



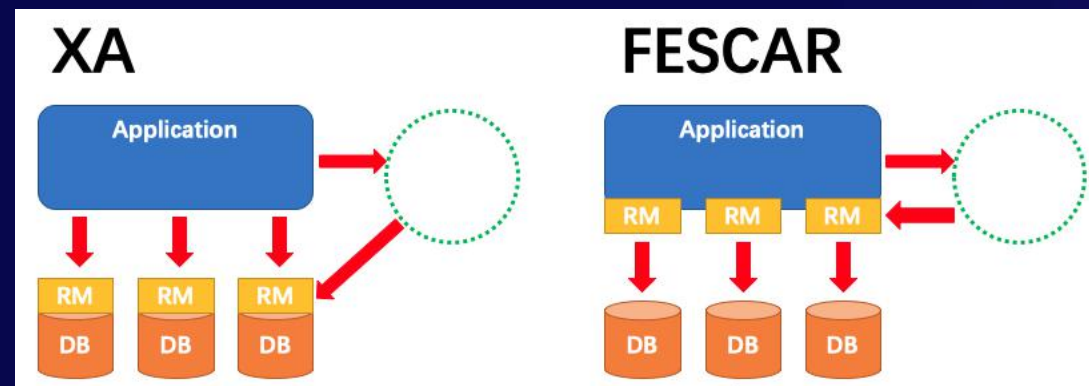
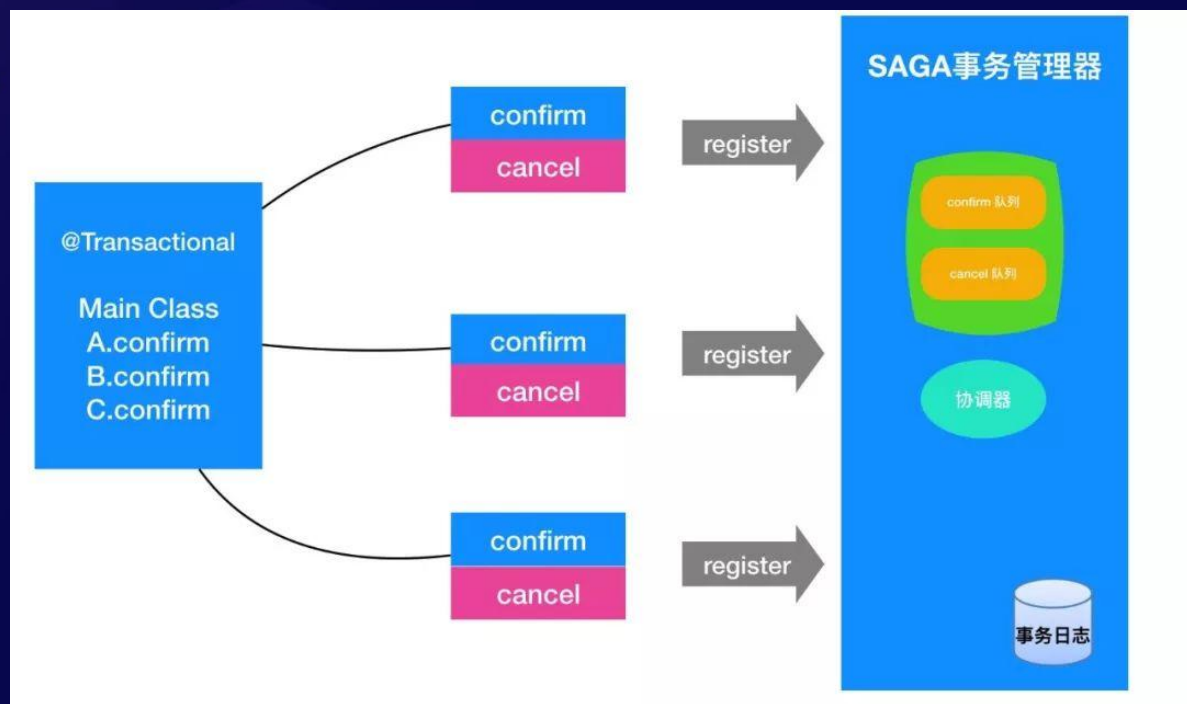
- 自动化测试
- 自动化部署
- 自动化运维

降低服务拆分带来的复杂性  
提升测试、部署、运维效率



## 二、最佳实践

### 最佳实践之五：如何取舍分布式事务



幂等/去重/补偿

最好的办法就是不用分布式事务！





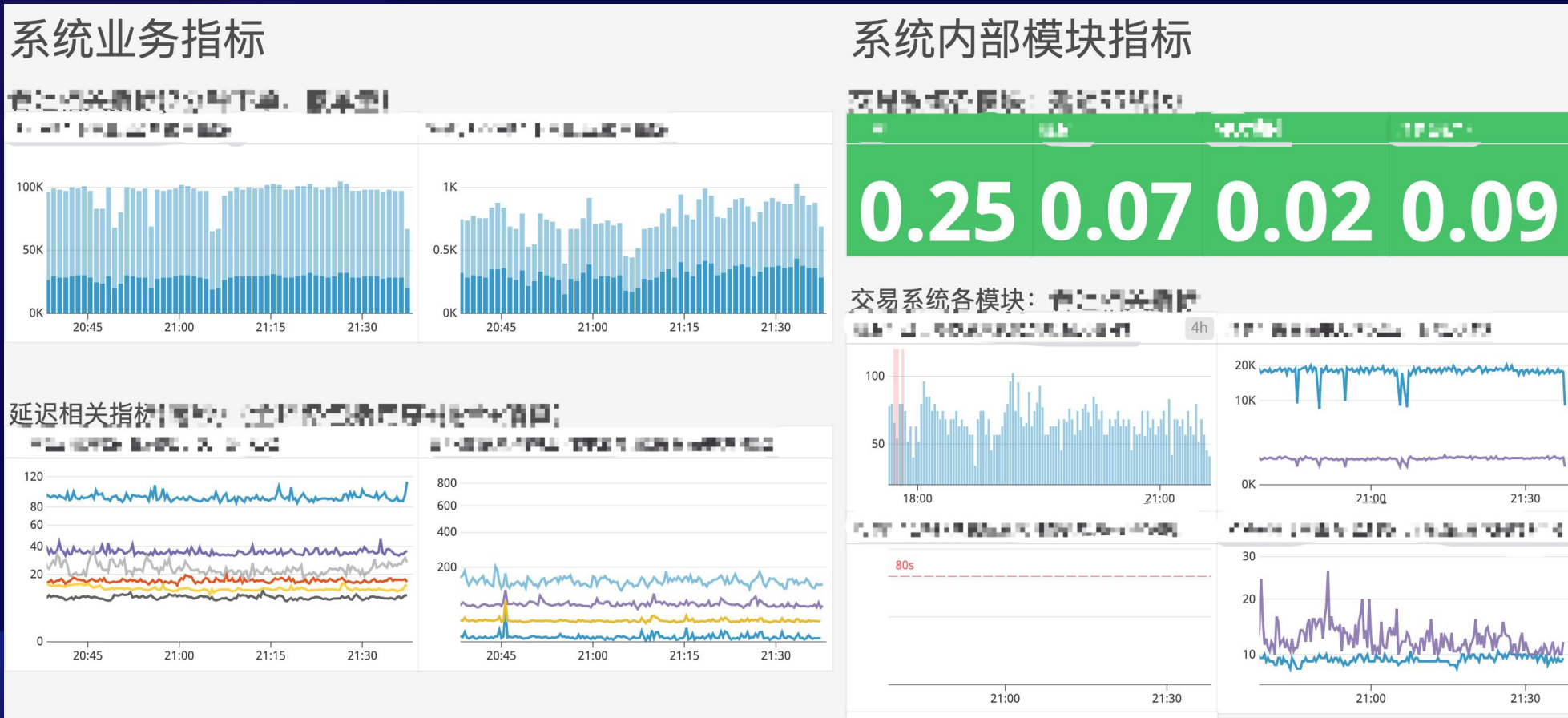
## 二、最佳实践

# 最佳实践之六：如何监控系统与排查问题



### 监控与运维：

1. 业务监控
2. 系统监控
3. 容量规划
4. 报警预警
5. 运维流程
6. 故障处理





## 二、最佳实践--总结

遗留系统改造

自动化管理

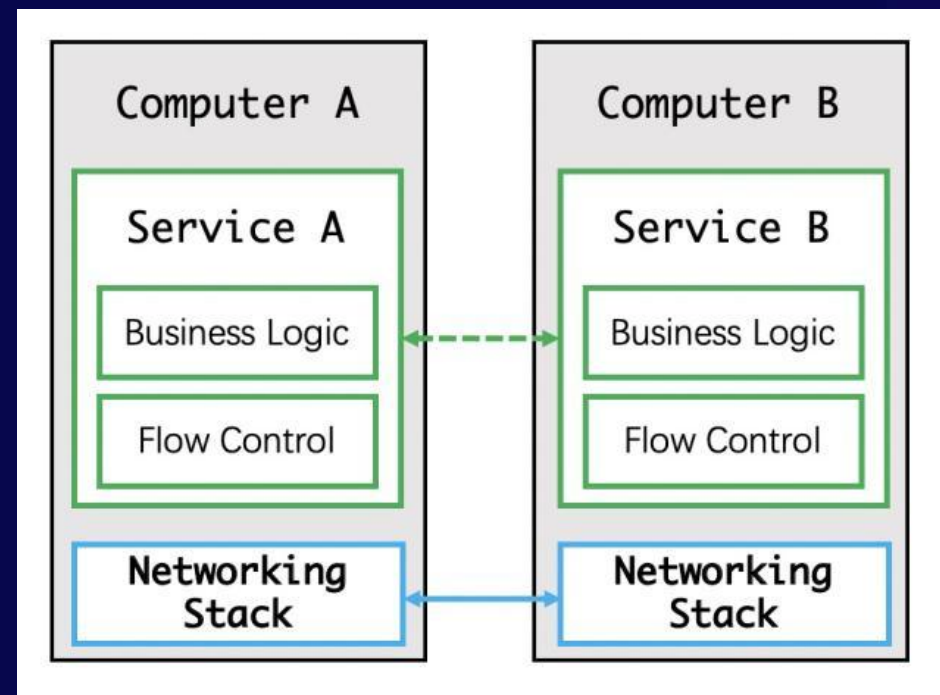
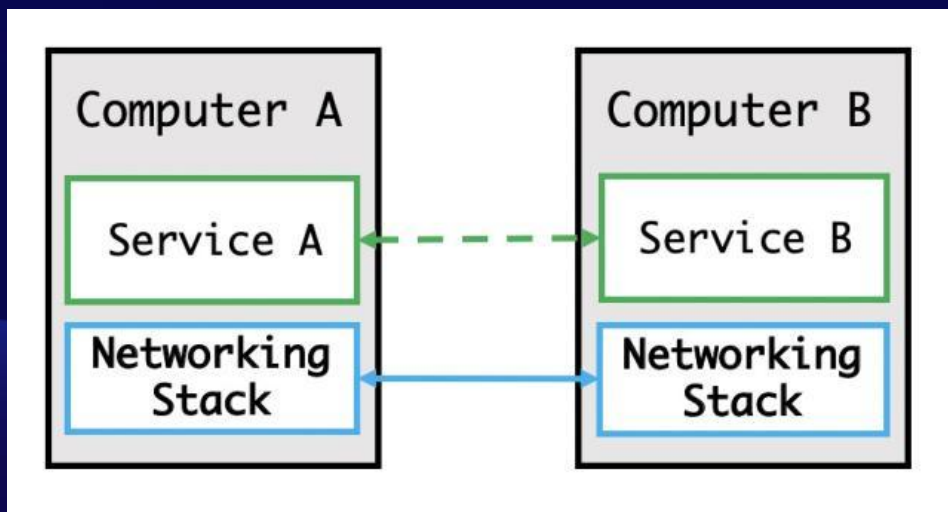
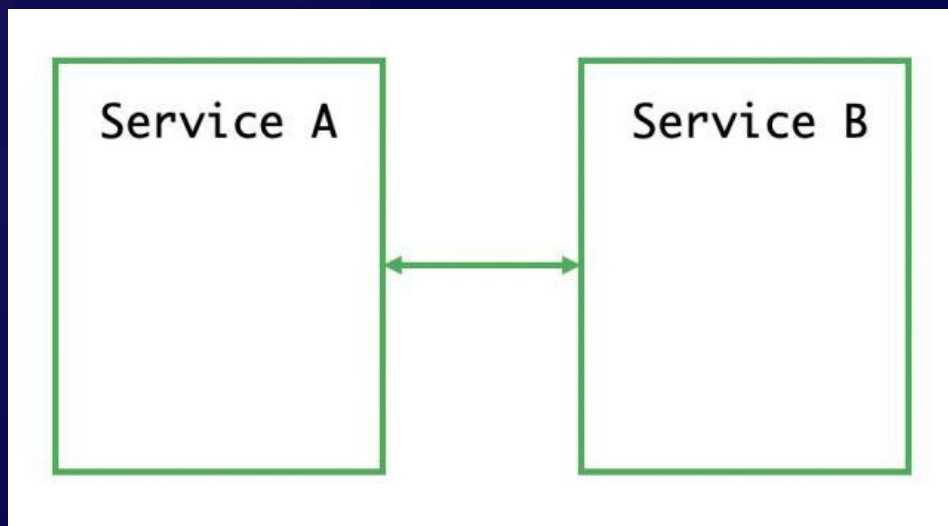
恰当粒度拆分

分布式事务

扩展立方体

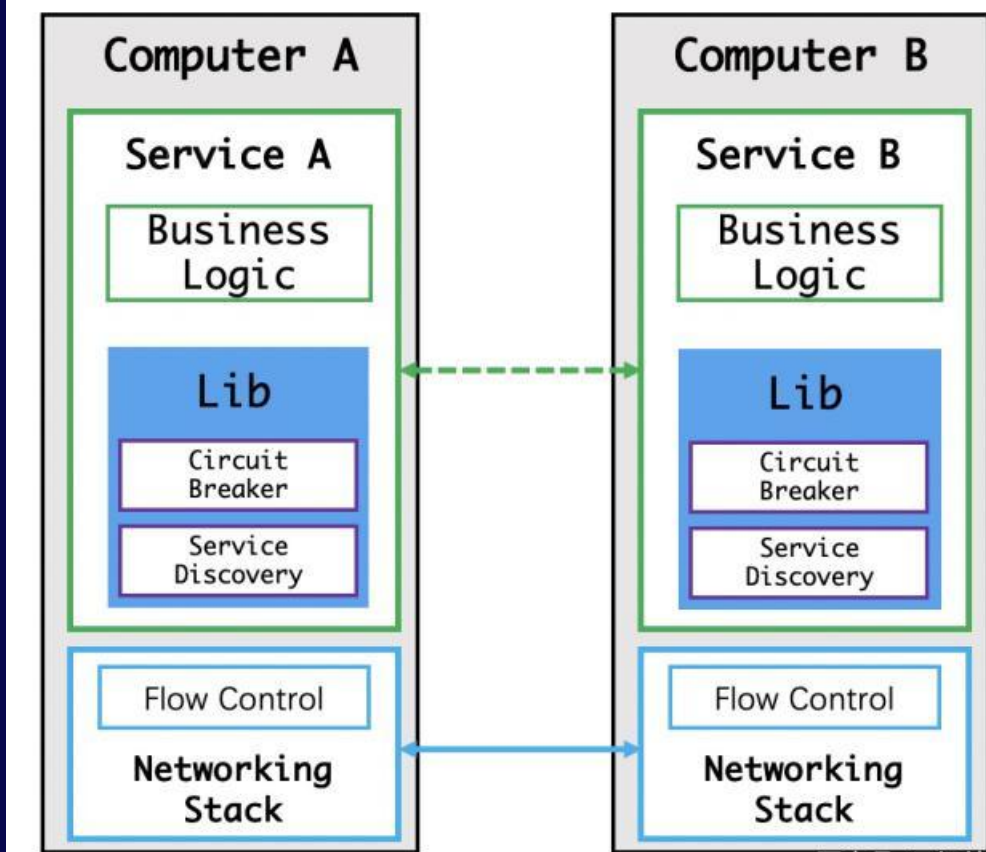
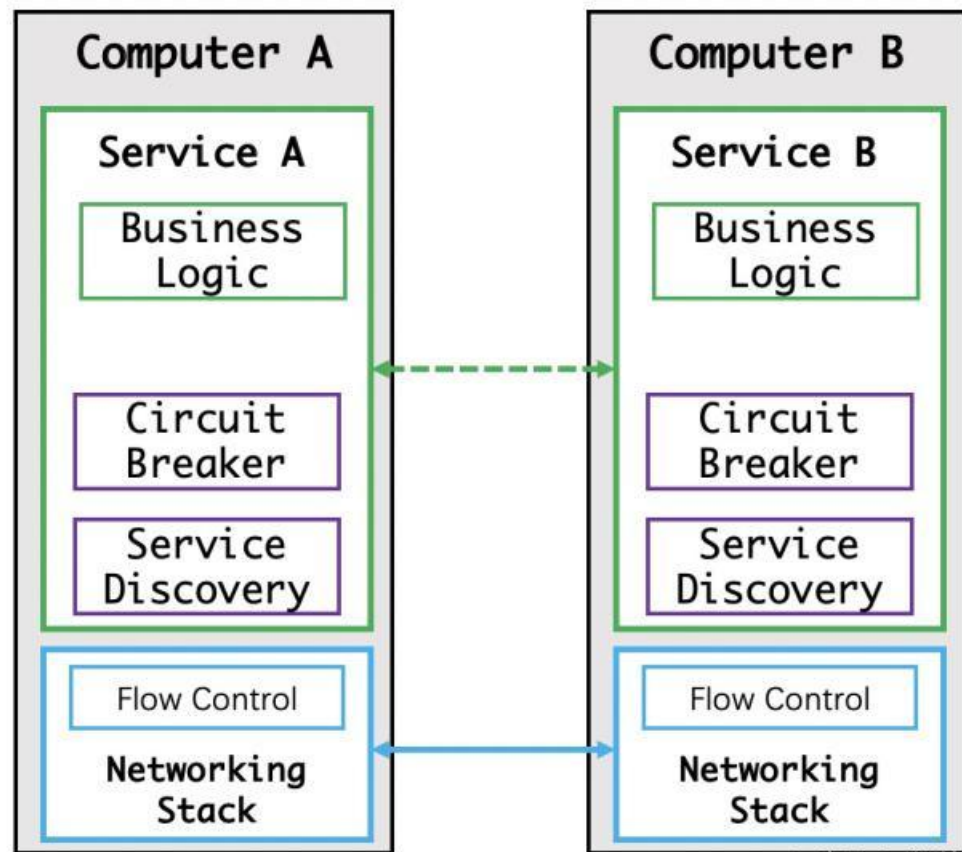
完善监控体系

### 三、服务网格--Service Mesh演化过程1

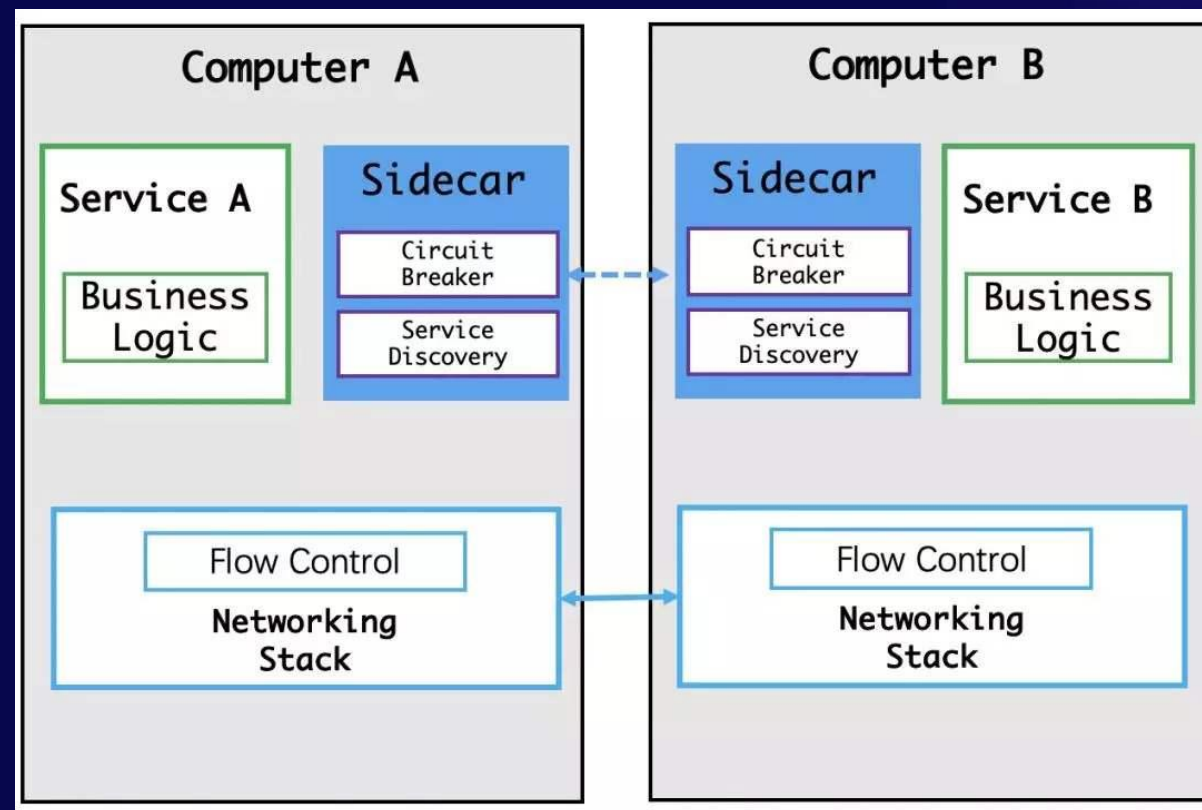
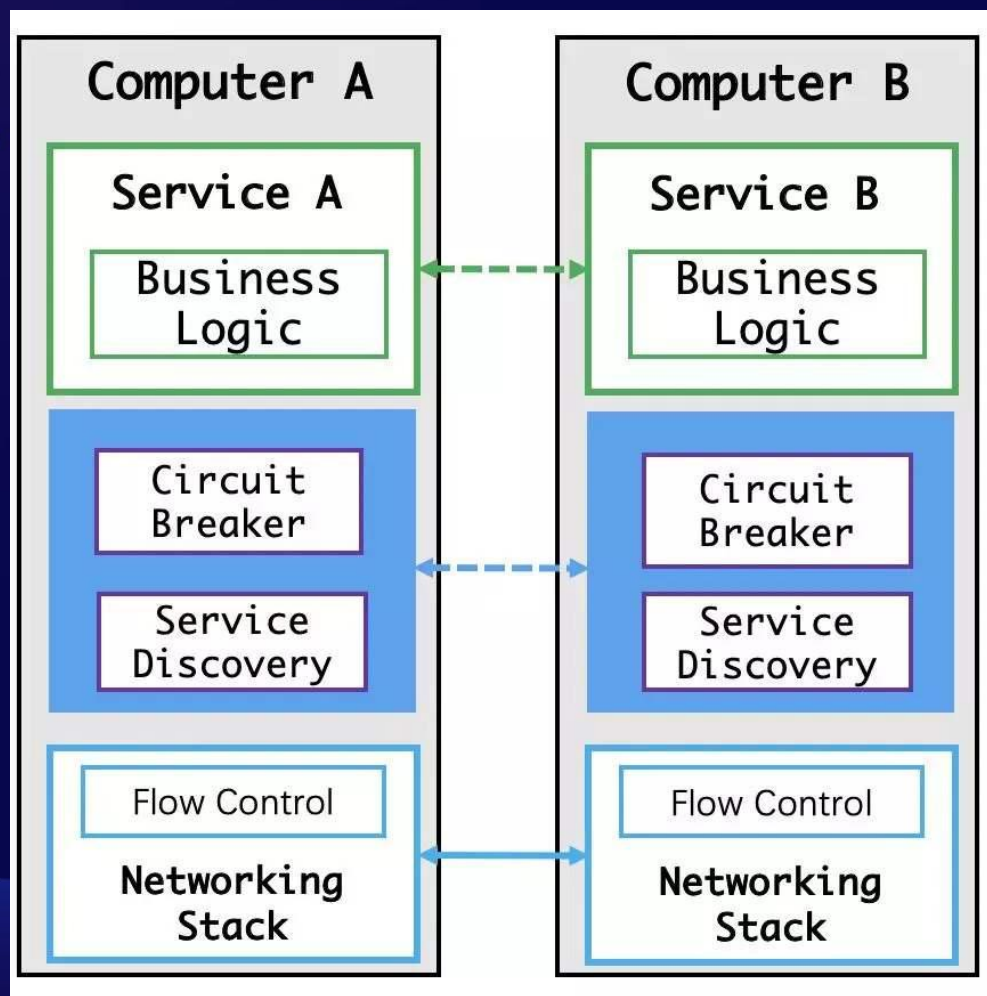


[https://philcalcado.com/2017/08/03/pattern\\_service\\_mesh.html](https://philcalcado.com/2017/08/03/pattern_service_mesh.html)

### 三、服务网格--Service Mesh演化过程2



### 三、服务网格--Service Mesh演化过程3



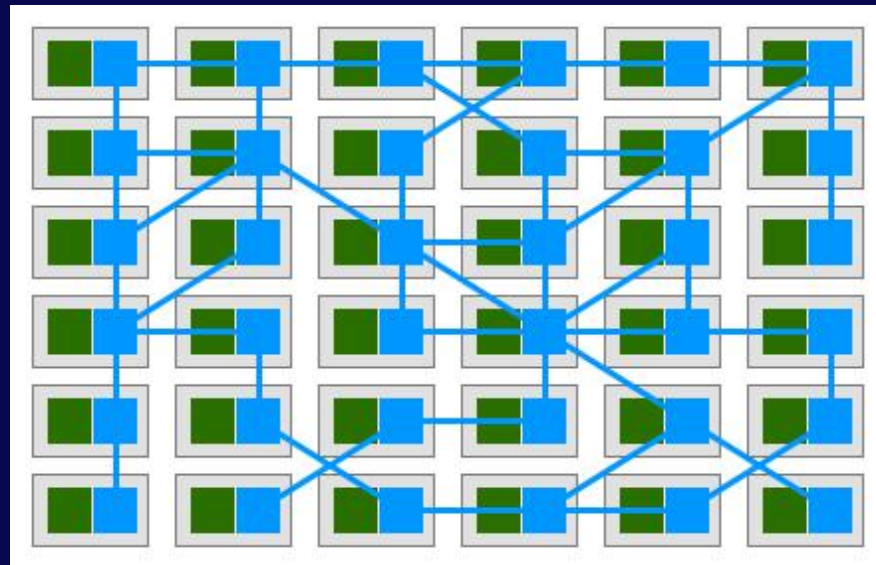
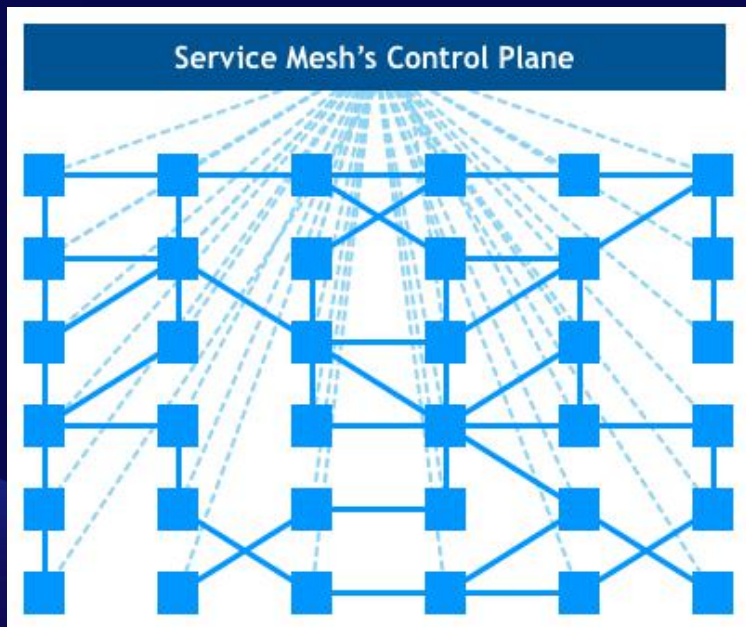
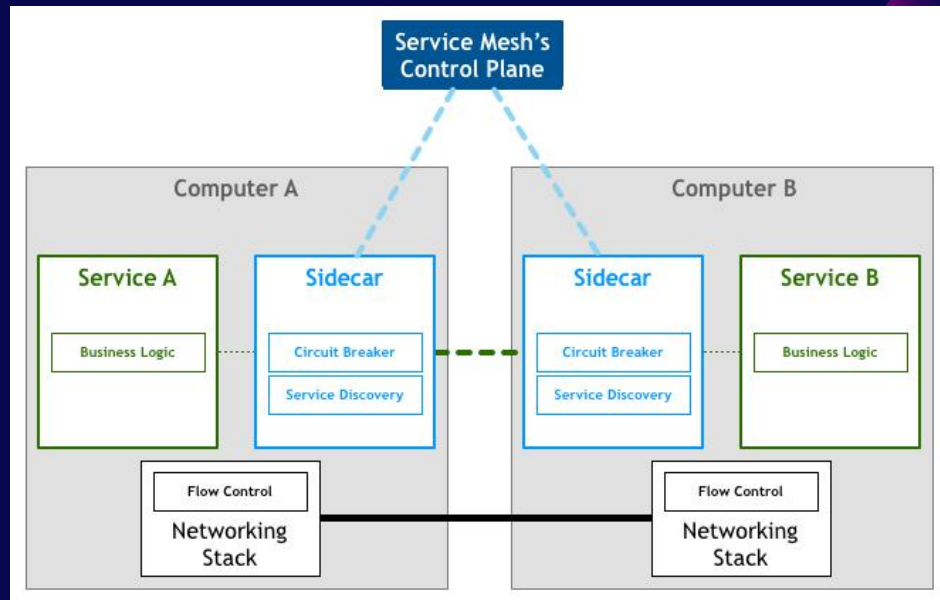




### 三、服务网格--定义与特点

Service Mesh 是一个基础设施层，用于处理服务间通信。云原生应用有着复杂的服务拓扑，Service Mesh 保证请求可以在这些拓扑中可靠地穿梭。在实际应用当中，Service Mesh 通常是由一系列轻量级的网络代理组成的，它们与应用程序部署在一起，但应用程序不需要知道它们的存在。

-- Willian Morgan / Linkerd CEO

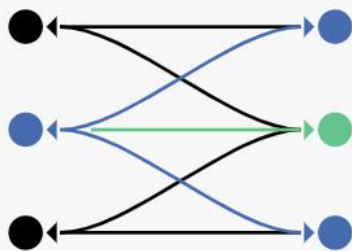


### 三、服务网格--Istio



# Istio

连接、保护、控制和观测服务。



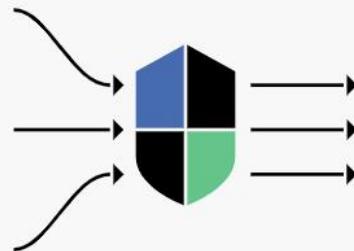
连接

智能控制服务之间的流量和 API 调用，进行一系列测试，并通过红/黑部署逐步升级。



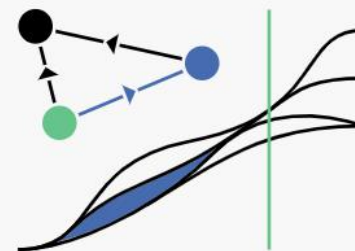
保护

通过托管身份验证、授权和服务之间通信加密自动保护您的服务。



控制

应用策略并确保其执行，使得资源在消费者之间公平分配。

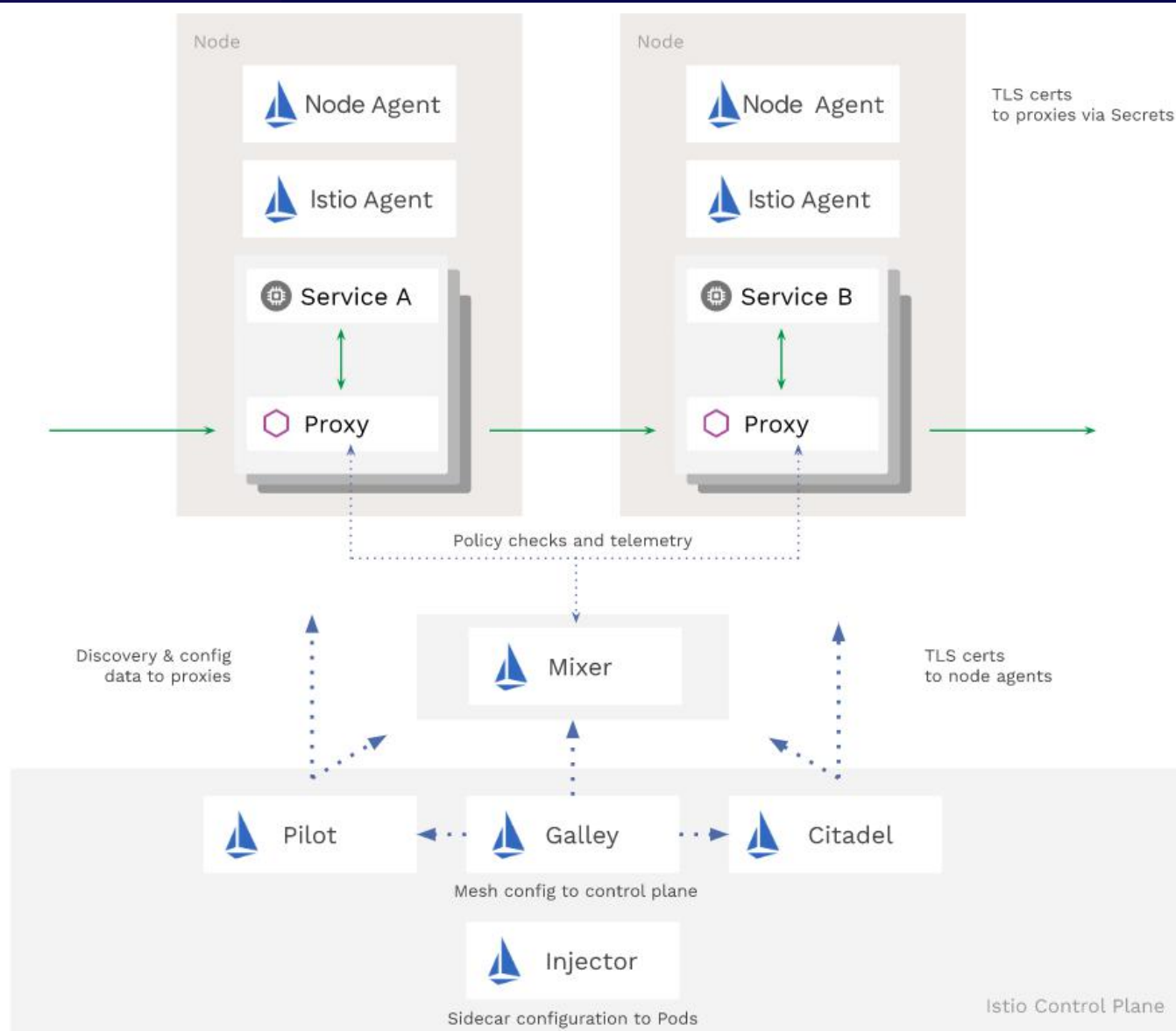


观测

对您的一切服务进行多样化、自动化的追踪、监控以及记录日志，以便实时了解正在发生的事情。



### 三、服务网格--Istio结构



逻辑上分为数据平面和控制平面。

数据平面由一组以 Sidecar 方式部署的智能代理 (Envoy) 组成。这些代理可以调节和控制微服务及 Mixer 之间所有的网络通信。

控制平面负责管理和配置代理来路由流量。此外控制平面配置 Mixer 以实施策略和收集遥测数据。

Pilot是整个架构中的抽象层，将对接K8S等资源调度器的步骤进行抽象并以适配器的形式展现，并和用户以及Sidecar交互。

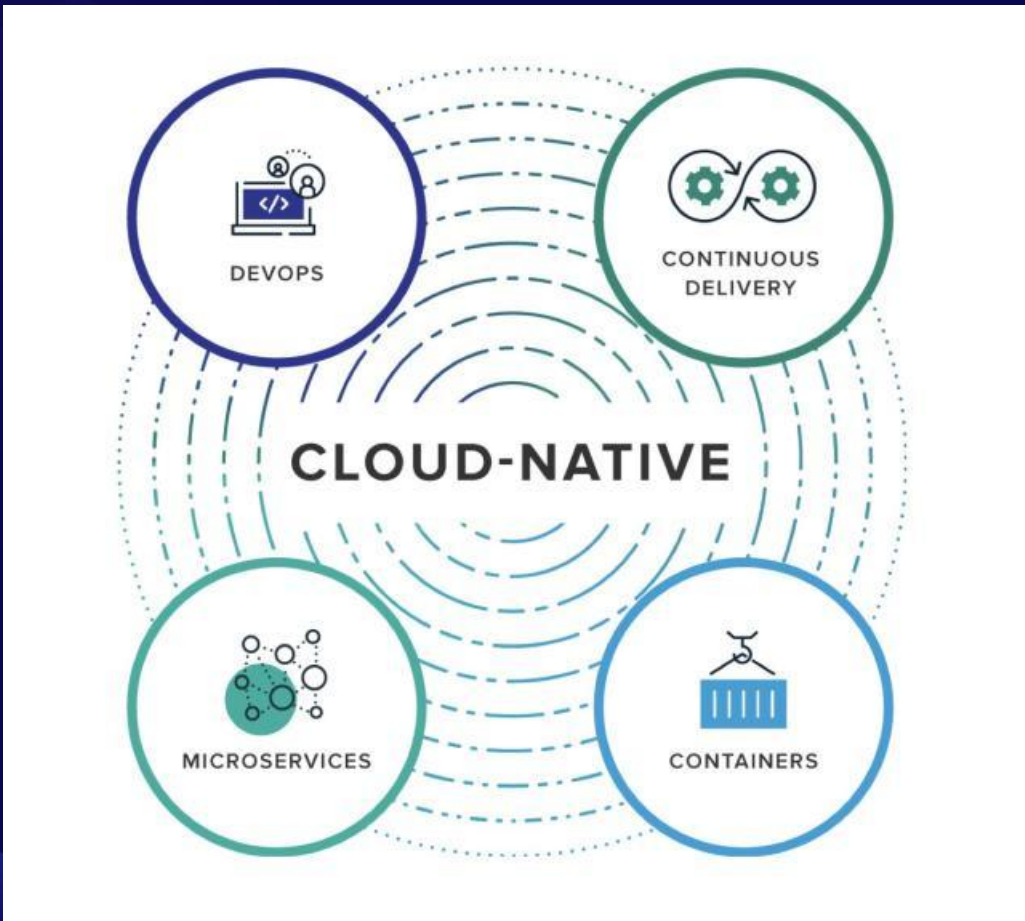
Galley负责资源配置为验证。

Citadel用于生成身份，及密钥和证书管理

核心功能包括流量控制、安全控制、可观测性、多平台支持、可定制化。



## 四、走向未来--云原生



云原生 Cloud-Native

1. 微服务
2. 容器
3. 持续交付
4. DevOps



# 四、走向未来--云原生全景图

App Definition and Development

Database

Streaming & Messaging

Application Definition & Image Build

Continuous Integration & Delivery

Orchestration & Management

Cloud Native Storage

Container Runtime

Automation & Configuration

Platform

Certified Kubernetes - Hosted

Certified Kubernetes - Installer

PaaS/Container Service

Observability and Analysis

Monitoring

Logging

Tracing

Chaos Engineering

Serverless

Kubernetes Certified Service Provider

Kubernetes Training Partner

Members

App Definition and Development

Database

Streaming & Messaging

Application Definition & Image Build

Continuous Integration & Delivery

Orchestration & Management

Cloud Native Storage

Container Runtime

Automation & Configuration

Platform

Certified Kubernetes - Hosted

Certified Kubernetes - Installer

PaaS/Container Service

Observability and Analysis

Monitoring

Logging

Tracing

Chaos Engineering

Serverless

Kubernetes Certified Service Provider

Kubernetes Training Partner

Members

App Definition and Development

Database

Streaming & Messaging

Application Definition & Image Build

Continuous Integration & Delivery

Orchestration & Management

Cloud Native Storage

Container Runtime

Automation & Configuration

Platform

Certified Kubernetes - Hosted

Certified Kubernetes - Installer

PaaS/Container Service

Observability and Analysis

Monitoring

Logging

Tracing

Chaos Engineering

Serverless

Kubernetes Certified Service Provider

Kubernetes Training Partner

Members

App Definition and Development

Database

Streaming & Messaging

Application Definition & Image Build

Continuous Integration & Delivery

Orchestration & Management

Cloud Native Storage

Container Runtime

Automation & Configuration

Platform

Certified Kubernetes - Hosted

Certified Kubernetes - Installer

PaaS/Container Service

Observability and Analysis

Monitoring

Logging

Tracing

Chaos Engineering

Serverless

Kubernetes Certified Service Provider

Kubernetes Training Partner

Members

App Definition and Development

Database

Streaming & Messaging

Application Definition & Image Build

Continuous Integration & Delivery

Orchestration & Management

Cloud Native Storage

Container Runtime

Automation & Configuration

Platform

Certified Kubernetes - Hosted

Certified Kubernetes - Installer

PaaS/Container Service

Observability and Analysis

Monitoring

Logging

Tracing

Chaos Engineering

Serverless

Kubernetes Certified Service Provider

Kubernetes Training Partner

Members

CLOUD NATIVE LANDSCAPE

CLOUD NATIVE COMPUTING FOUNDATION

Redpoint

Amplify

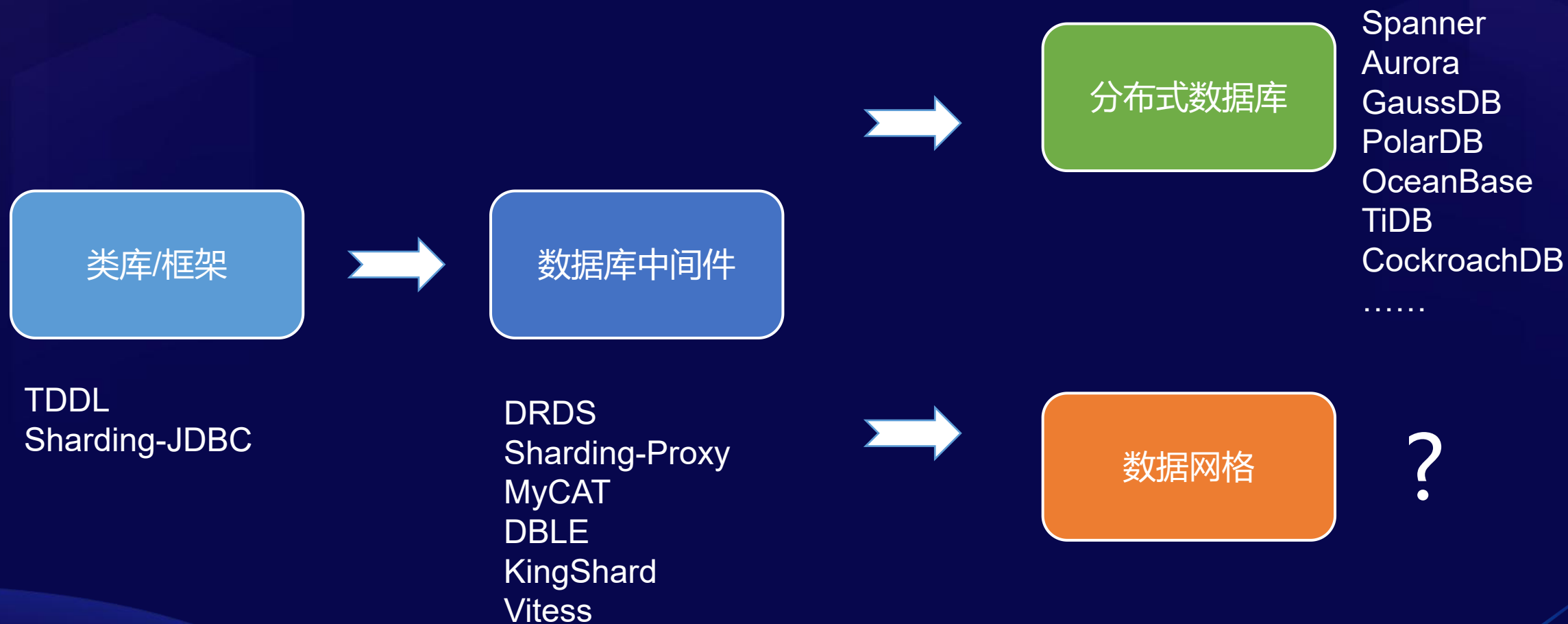
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

Special

Kubernetes Certified Service Provider

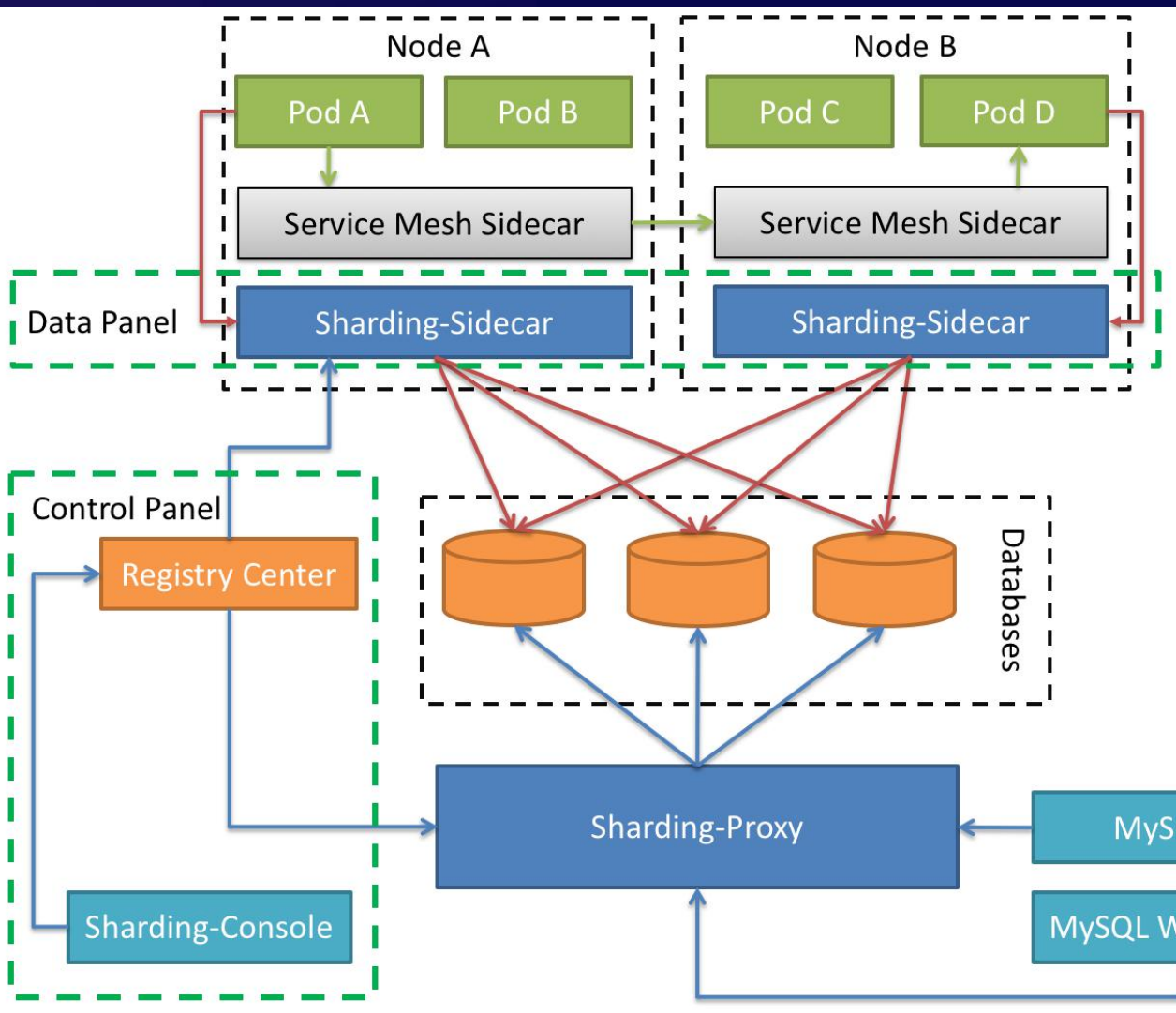
Kubernetes Training Partner

## 四、走向未来--数据库网格Database Mesh





## 四、走向未来--数据库网格Database Mesh



Level 6: Sharding-Engine (6.x+)

Level 5: Sharding-Sidecar (5.x+)

Level 4: Sharding-Scaling (4.x+)

We're  
Here  
Now

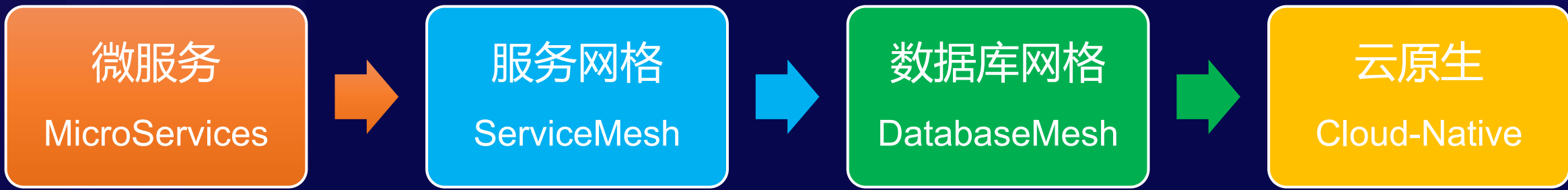
Level 3: Sharding-Proxy中间件 (3.x+)

Level 2: Sharding-JDBC框架 (1.x+)

Level 1: MySQL数据库提供的能力



## 四、走向未来--未来已来



拥抱开源  
拥抱新技术

[kimmking@apache.org](mailto:kimmking@apache.org)



# THANKS

