Huobi

GitChat

（火币集团研发中心&GitChat技术沙龙）

微服务架构深度解析与最佳实践

秦金卫KimmKing

2019年04月

Huobi

Financial Hub of Digital Assets

个人介绍

① 火币集团高级技术总监/Apache Dubbo PMC，前阿里架构师/某商业银行北京研发中心负责人，阿里云MVP、TGO鲲鹏会会员

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Dubbo/Fastjson/ActiveMQ等多个开源社区

④ 《高可用可伸缩微服务架构：基于Dubbo、Spring Cloud和ServiceMesh》合著作者

⑤ 个人博客：http://kimmking.github.io

目录

Part 1 什么是微服务架构

Part 2 微服务架构深度解析

Part 3 微服务架构最佳实践

Part 1 什么微服务架构

微服务架构概述

2011年

现在

1
- 2011年5月威尼斯会议提出“微服务”

2
- 2012年3月James Lewis关于微服务原则的演讲

3
- 2012年11月Fred George提出“微服务架构”概念

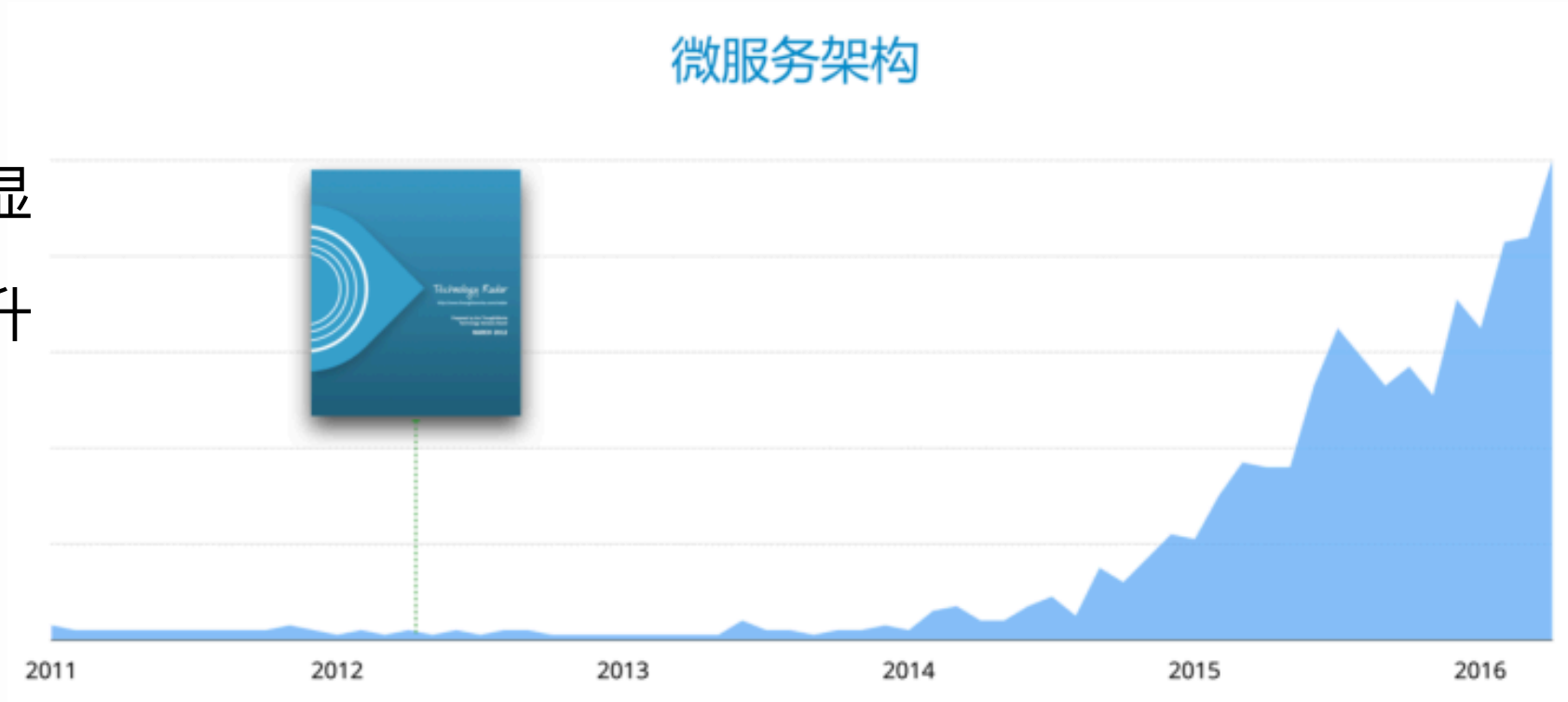
4
- 2014年3月Martin Flowler全面介绍了微服务架构

5
- 2016年4月JonasBonér提出“响应式微服务架构”

微服务架构技术热度

发展趋势明显

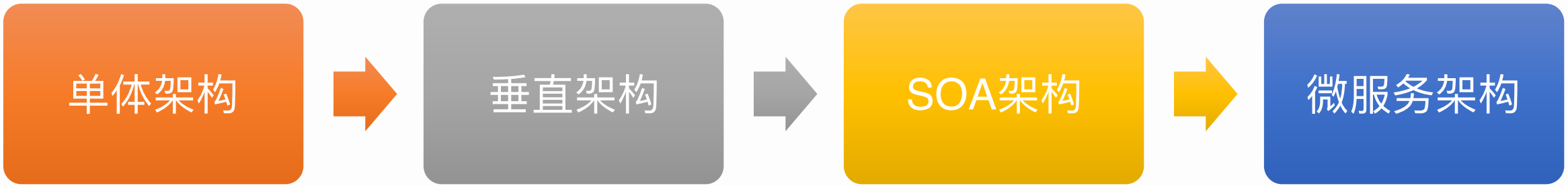
热度不断提升



Huobi

Financial Hub of Digital Assets

微服务架构发展历程



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。

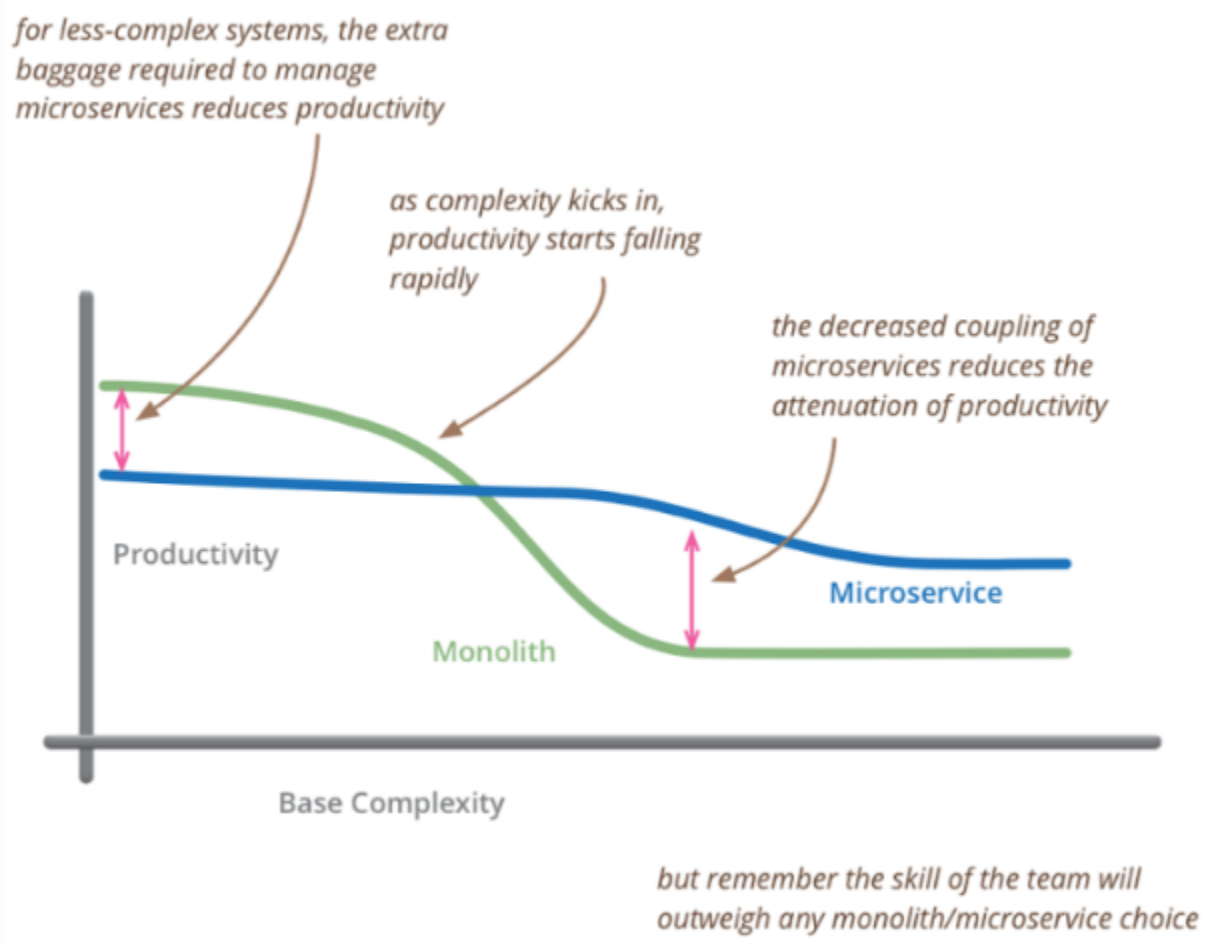
微服务架构

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。

微服务架构的特点

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

微服务架构的特点



- ✓ 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？

微服务架构的常见特性

服务发现

负载均衡

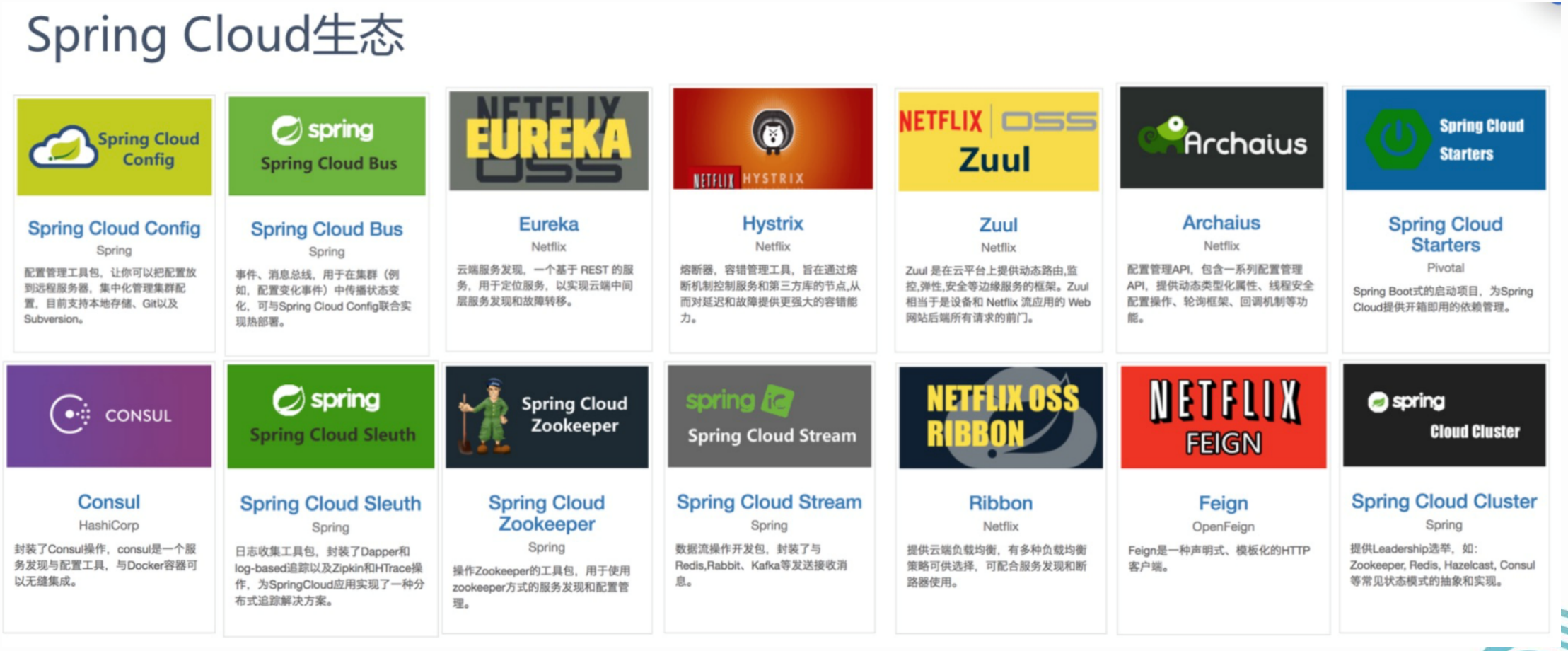
服务网关

分布式配置

服务熔断

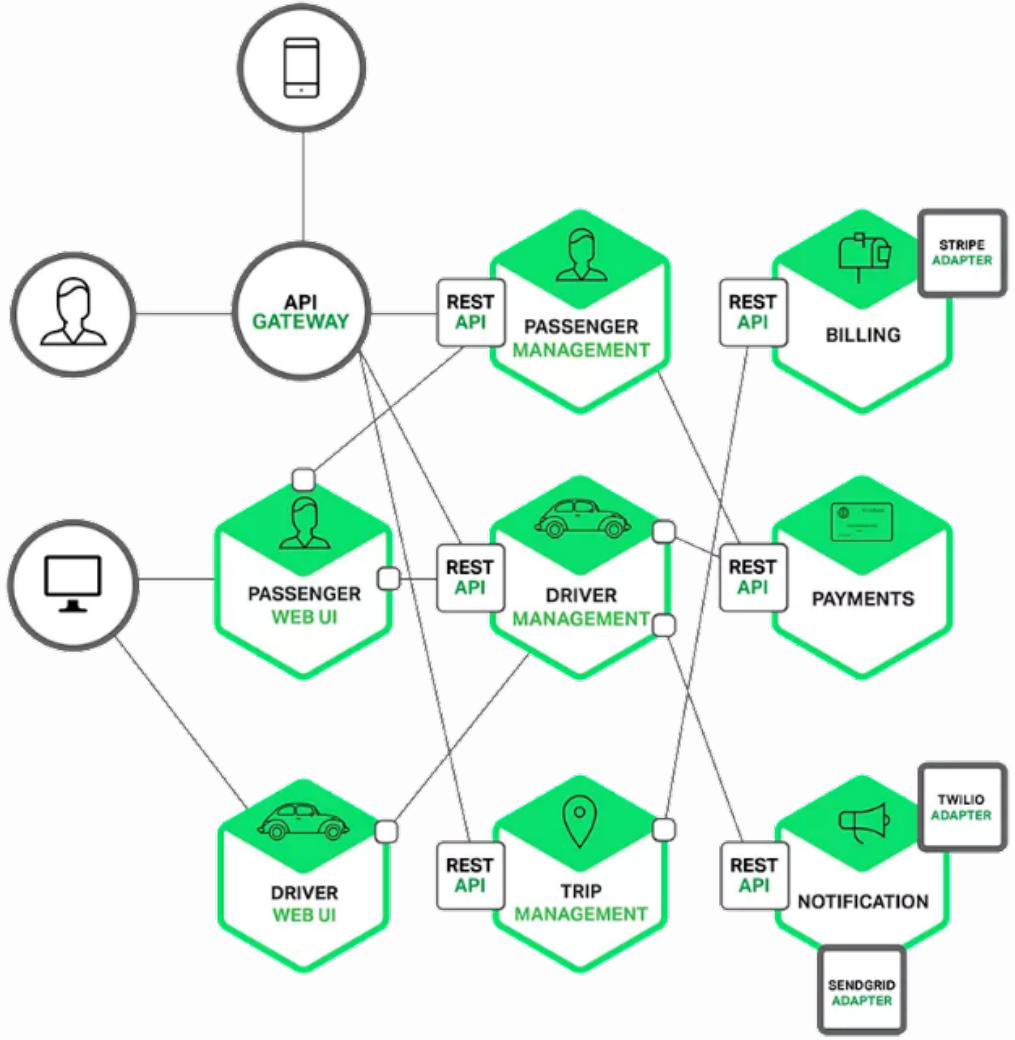
APM服务监控

微服务架构的常见技术



此外，还有Apache Dubbo, Vert.x, Logam 等技术

微服务架构的示例1

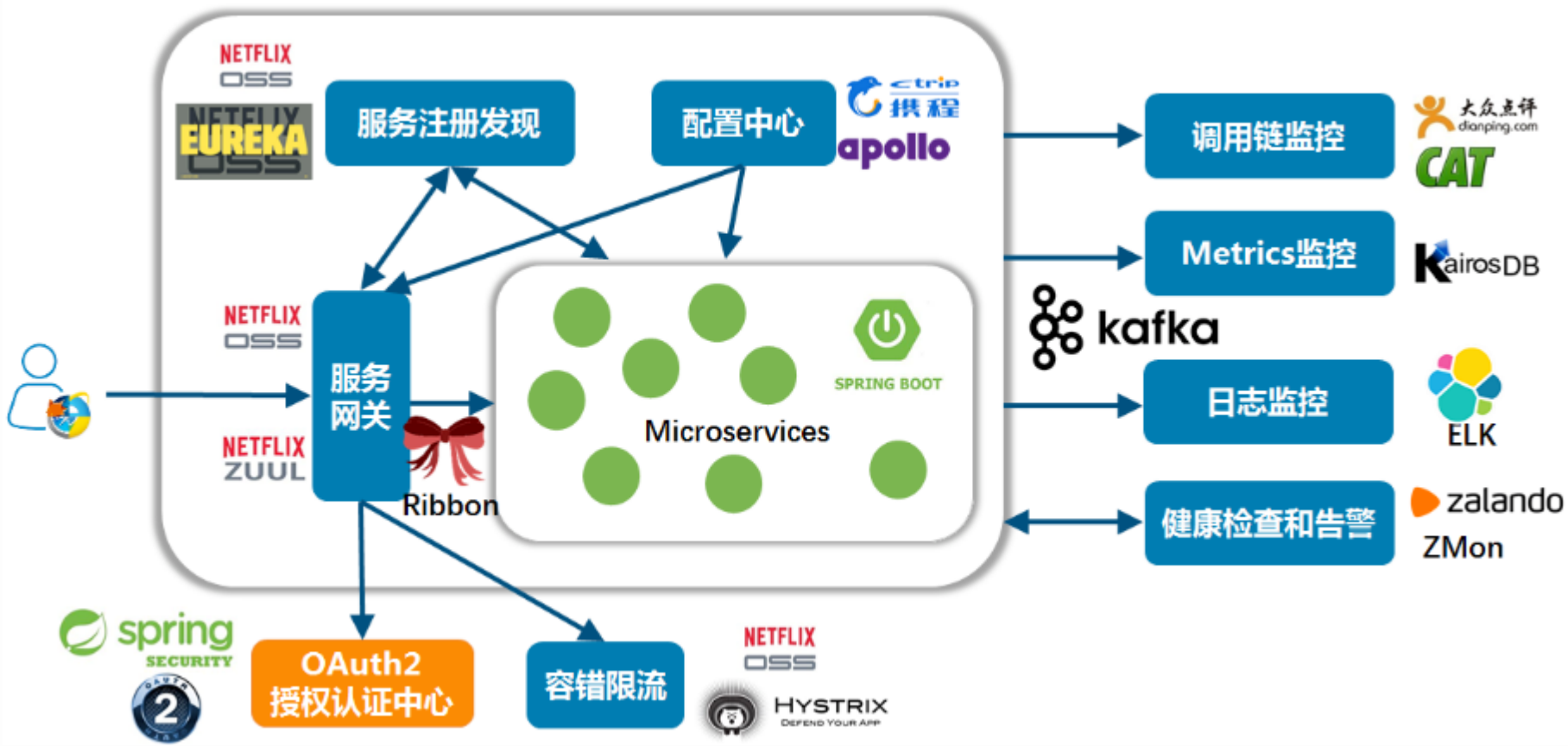


Gateway

REST API

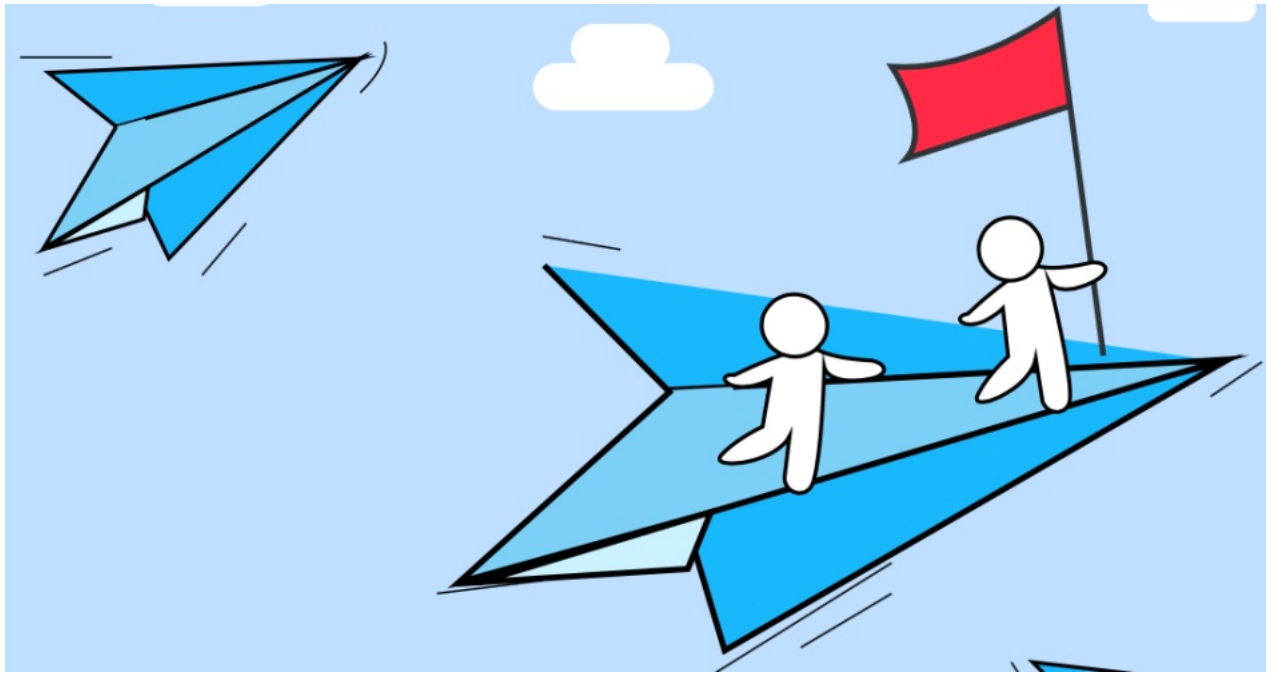
Service

微服务架构的示例2


NETFLIX OSS
NETFLIX EUREKA OSS
服务注册发现
配置中心
ctrip 携程
apollo
调用链监控
大众点评 dianping.com
CAT
NETFLIX OSS
服务网关
NETFLIX ZUUL
Ribbon
SPRING BOOT
Microservices
Metrics监控
KairosDB
kafka
日志监控
ELK
健康检查和告警
zalando
ZMon
spring SECURITY
OAuth2 授权认证中心
容错限流
NETFLIX OSS
HYSTRIX
DEFEND YOUR APP

Part 2 微服务架构深度解析

微服务的优势



- 服务简单
- 灵活扩展
- 便于维护
- 独立演进
- 混合开发
- 持续交付

微服务的几个问题

- 使用与维护过于复杂?
- 服务与服务之间的关系?
- 提升单个服务的性能?

Vert.x响应式编程

响应式编程

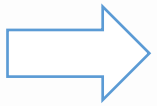
- 响应式编程是一个专注于数据流和变化传递的**异步编程范式。 -- 维基百科**

响应式宣言

即时响应性	可恢复性	弹性	消息驱动

为什么需要响应式

```
int a=1;
int b=a+1;
System.out.print("b="+b)     //   b=2
a=10;
System.out.print("b="+b)     //   b=2
```



```
int a=1;
int b <= a+1;    // <= 符号只是表示a和b之间关系的操作符
System.out.print("b="+b)     //   b=2
a=10;
System.out.print("b="+b)     //   b=11
```

响应式的优势

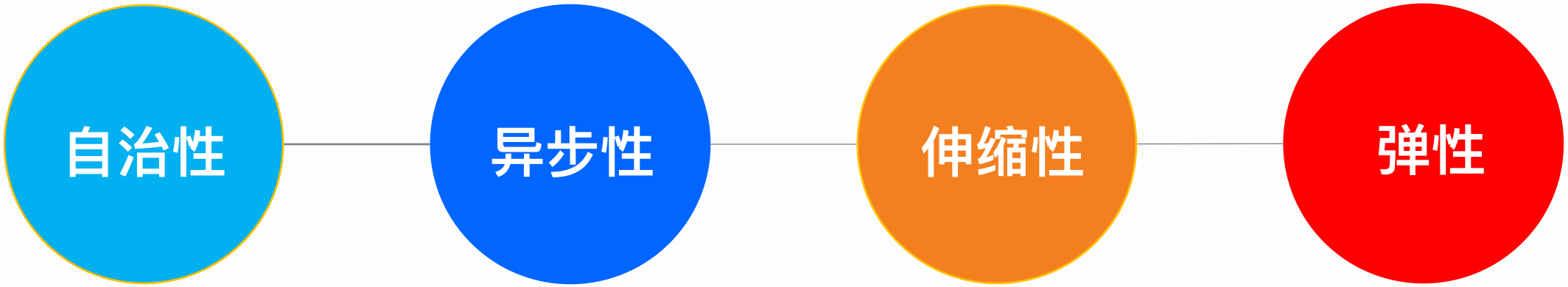
- 在业务层面实现代码逻辑分离，方便后期维护和拓展
- 极大提高程序响应速度，充分发掘CPU的能力
- 帮助开发者提高代码的抽象能力和充分理解业务逻辑
- 丰富的操作符会帮助我们极大的简化代码逻辑

数据流+异步编程

基于数据流，做异步编程，
有什么好处?

什么是响应式微服务

一个**响应式的微服务系统**须由**响应式微服务**组成。
这些响应式的微服务需要具有以下四个特性：



响应式微服务框架Vert.x的总结

高可扩展性

Vert.x是事件驱动和非阻塞的。这意味着应用程序可以使用少量内核线程处理大量并发。Vert.x可让应用程序以最少的硬件进行灵活扩展。

通用性技术

Vert.x非常灵活，无论是简单的网络应用程序，复杂的现代Web应用程序，HTTP/REST微服务，海量事件处理还是完整的后端消息总线，Vert.x都适用。

多语言支持

Vert.x可以与多种语言一起使用，包括Java，JavaScript，Groovy，Ruby，Ceylon，Scala和Kotlin等。Vert.x支持以上各种语言提供调用 API。

非限定框架

Vert.x不是限制性框架或容器，它并不限制什么才是编写应用程序的正确方法。相反，它会提供很多有用的积木，让开发者使用自己想要的方式创建应用程序。

Part 3 微服务架构最佳实践

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

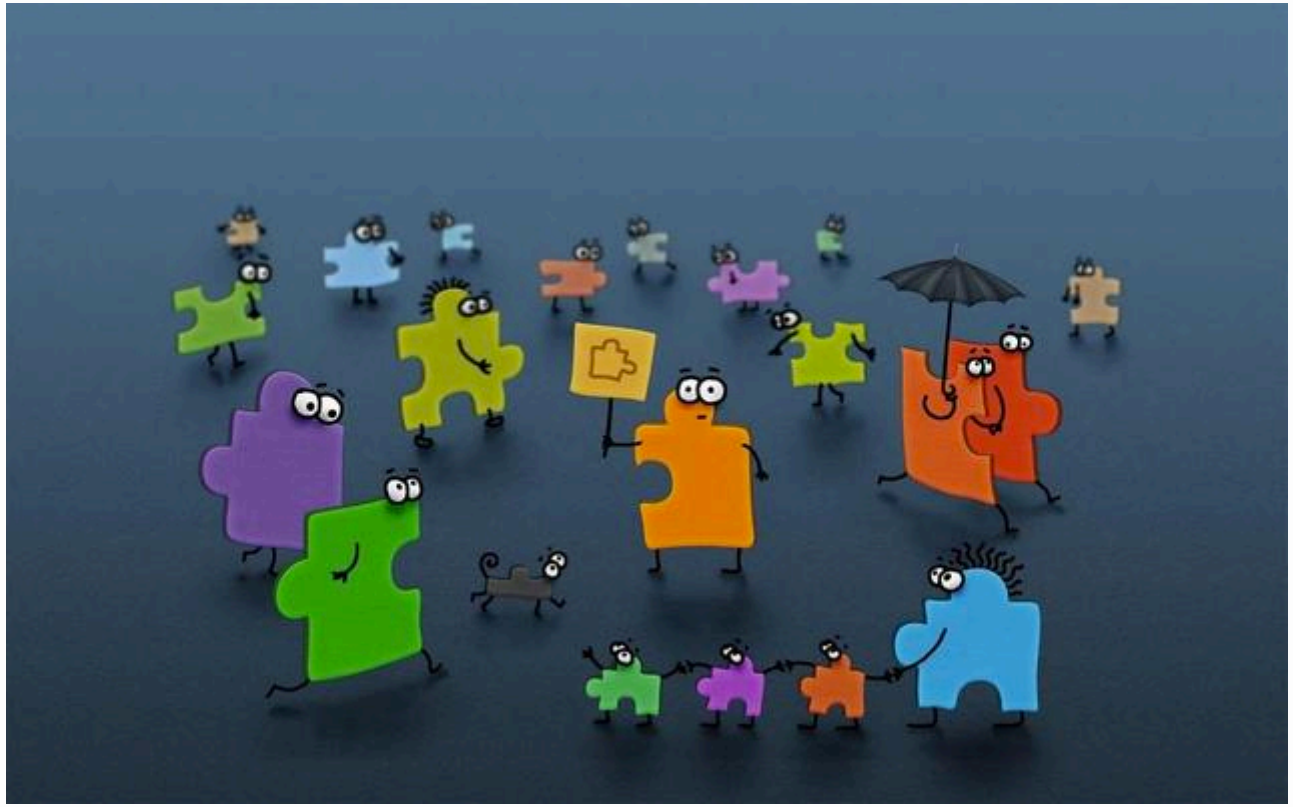
② 自然演进、逐步拆分

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

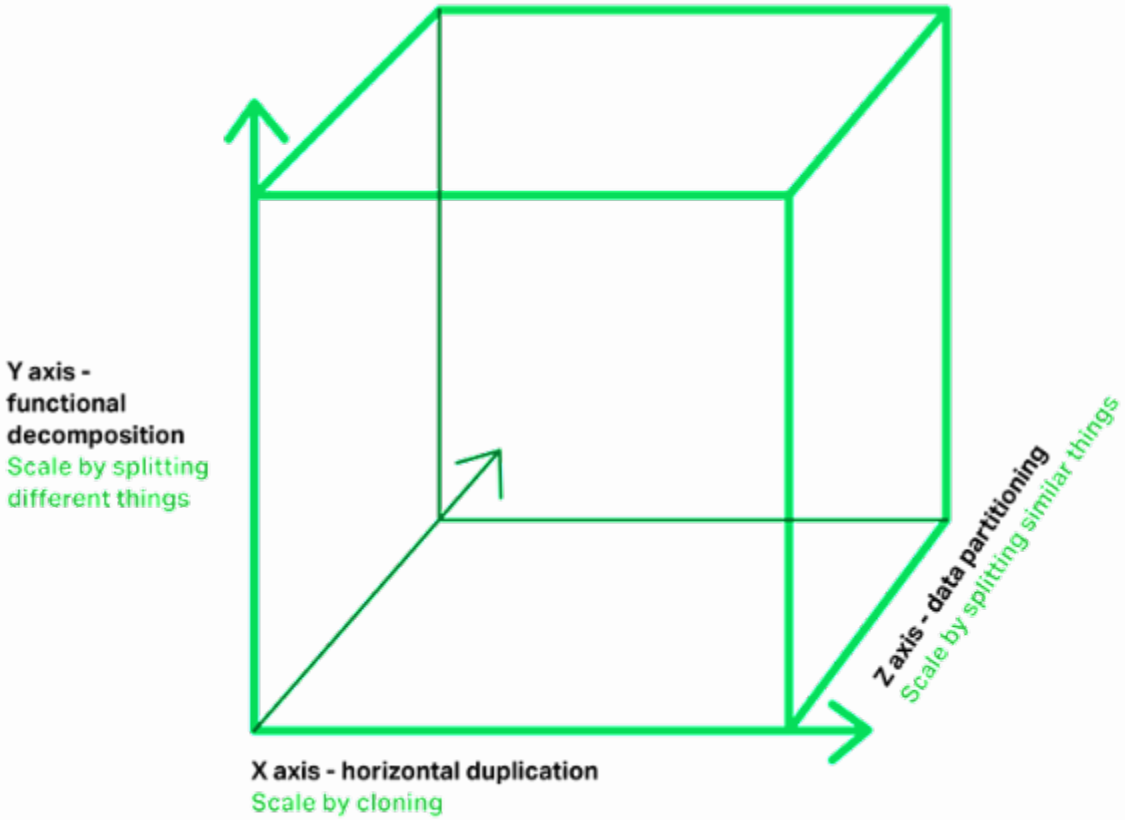
最佳实践之二：如何做恰当粒度的拆分



拆分原则：

1. 高内聚低耦合
2. 不同阶段拆分要点不同

最佳实践之三：如何扩展微服务系统

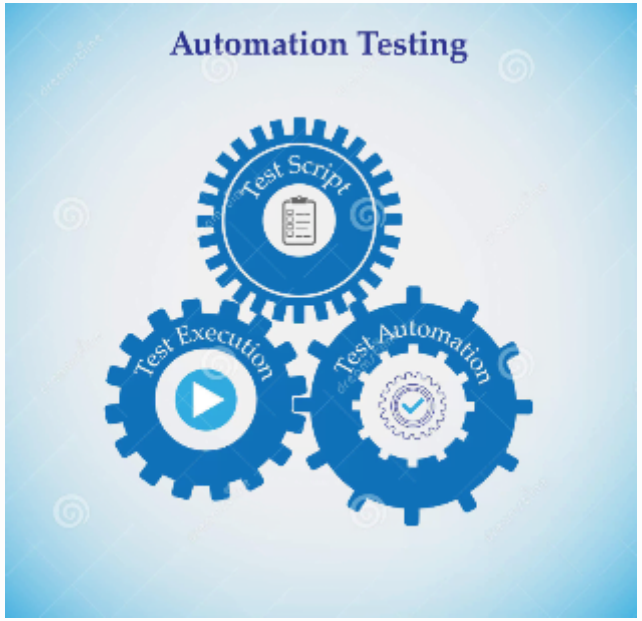


扩展立方体：

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

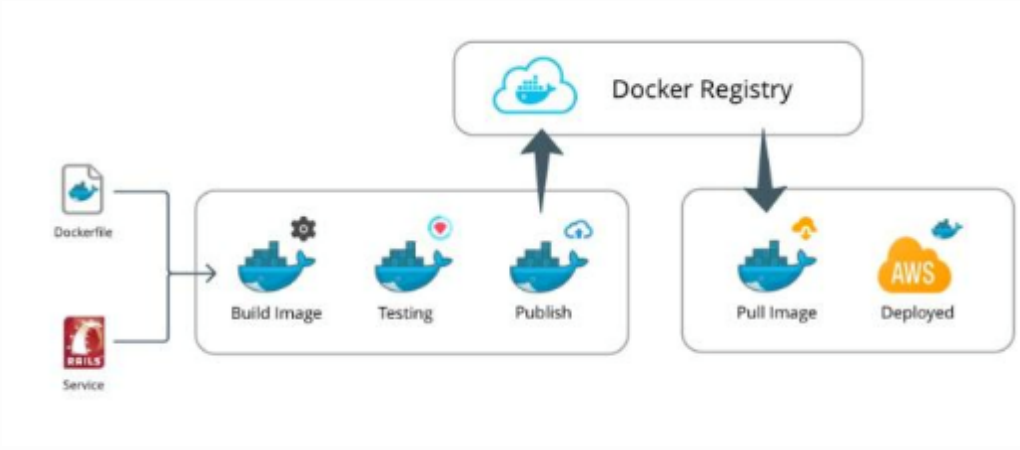
+特性开关 +容错设计

最佳实践之四：如何提升研发效率

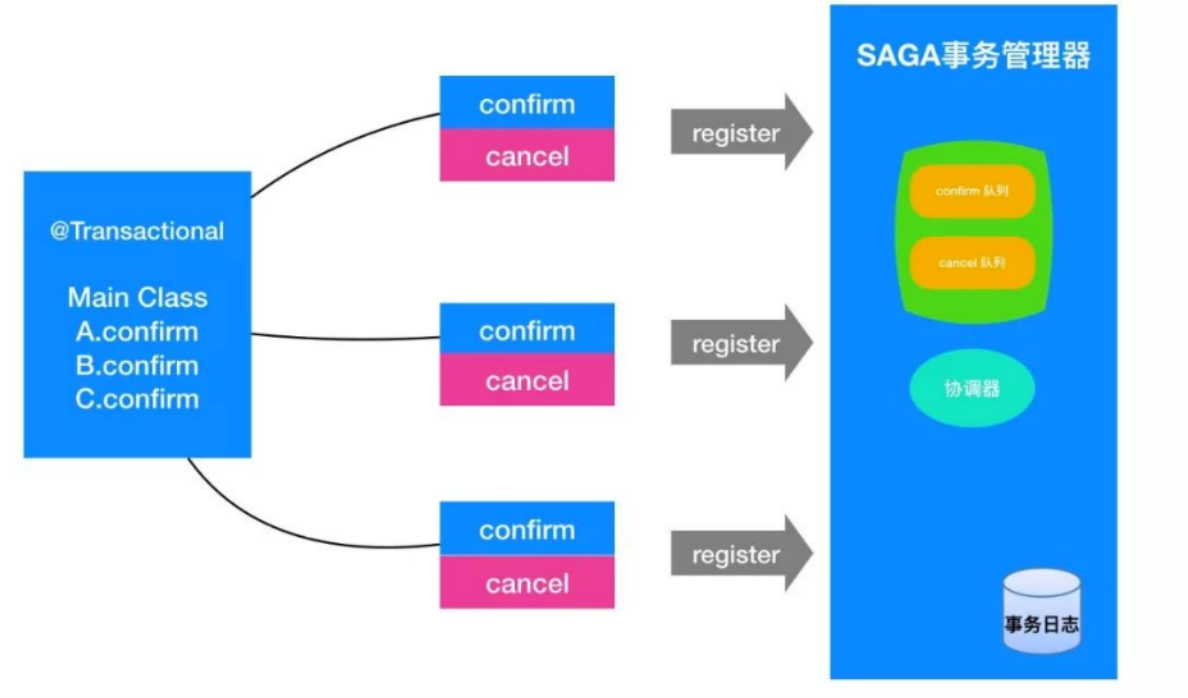


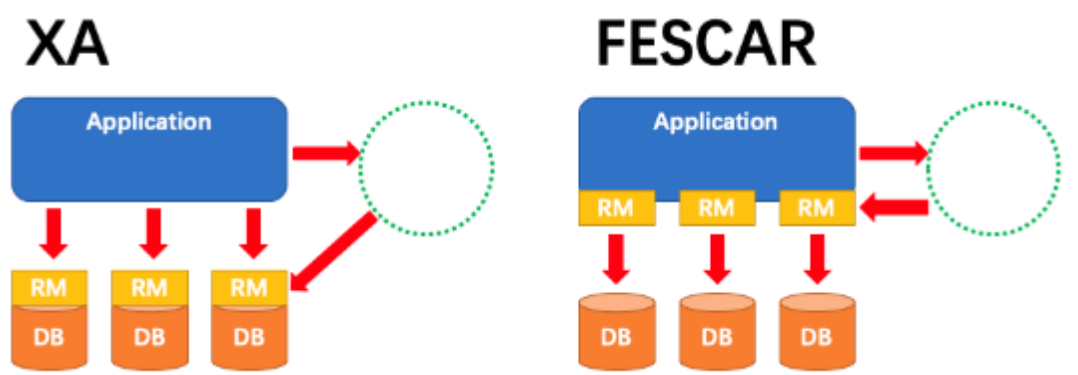
- 自动化测试
- 自动化部署
- 自动化运维

降低服务拆分带来的复杂性
提升测试、部署、运维效率



最佳实践之五：如何取舍分布式事务





幂等/去重/补偿

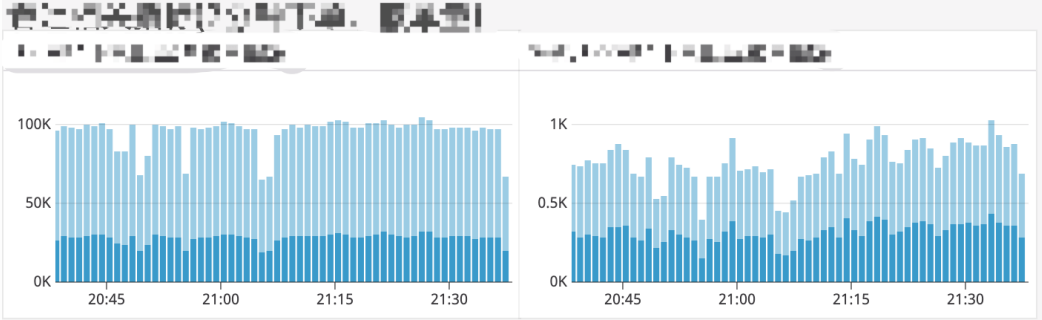
最好的办法就是不用分布式事务！

Huobi

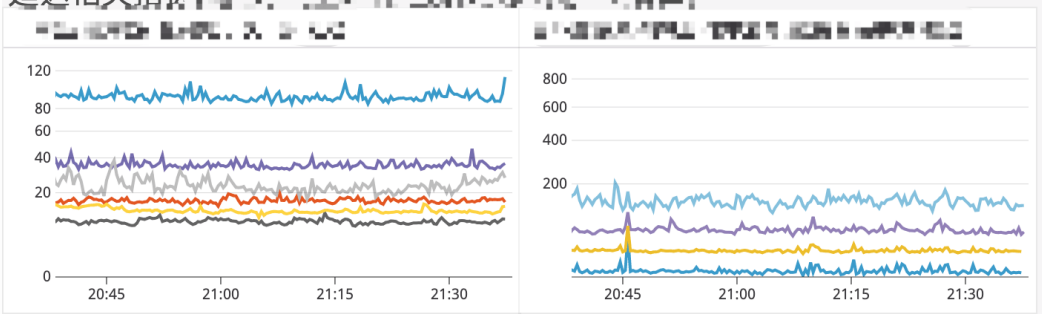
Financial Hub of Digital Assets

最佳实践之六：如何监控系统与排查问题

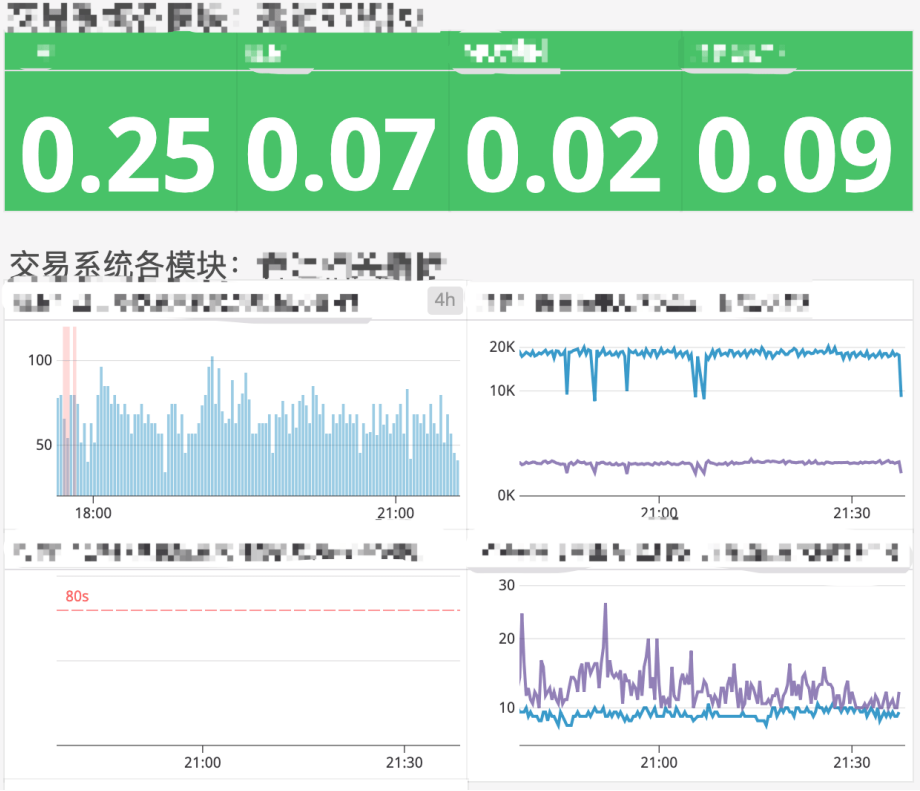
系统业务指标



延迟相关指标



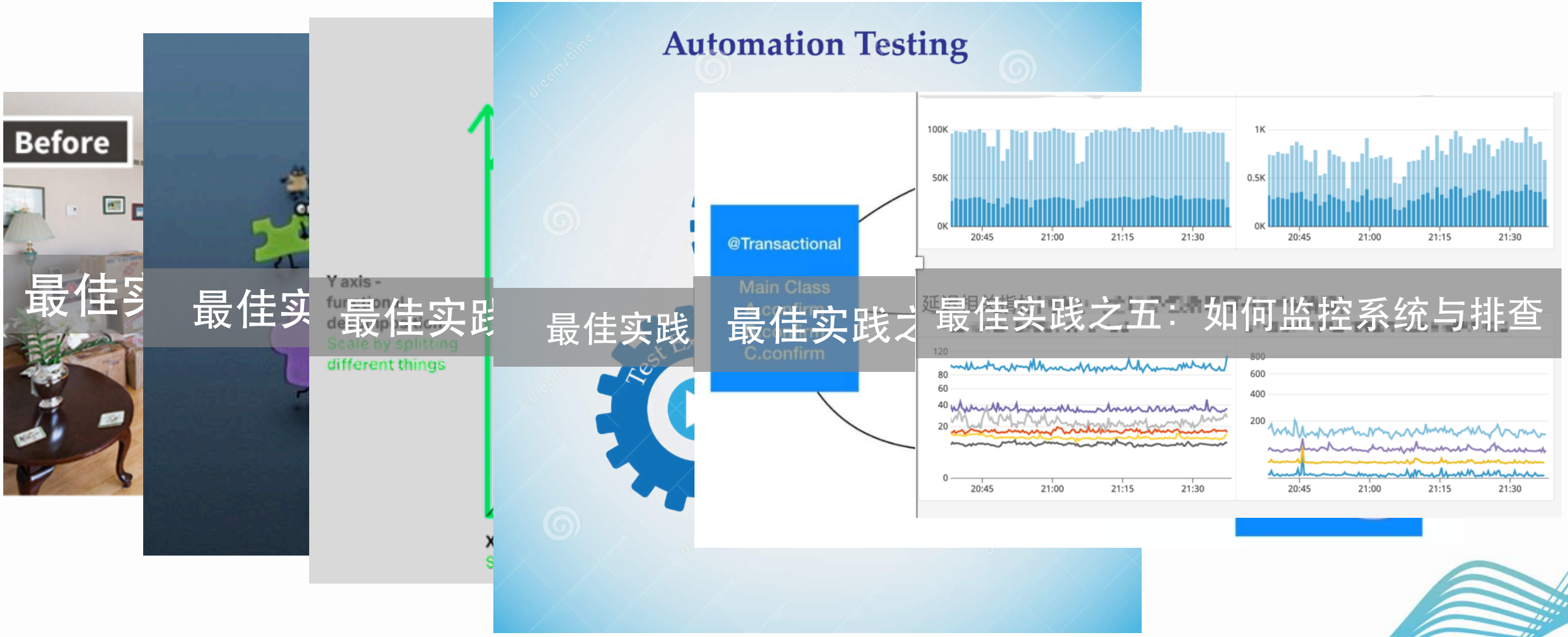
系统内部模块指标



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

最佳实践的总结



最佳实践之五：如何监控系统与排查

Huobi

Financial Hub of Digital Assets

火币集团研发中心诚聘英才

成为火伴

此生无憾

Huobi

GitChat

Thank you