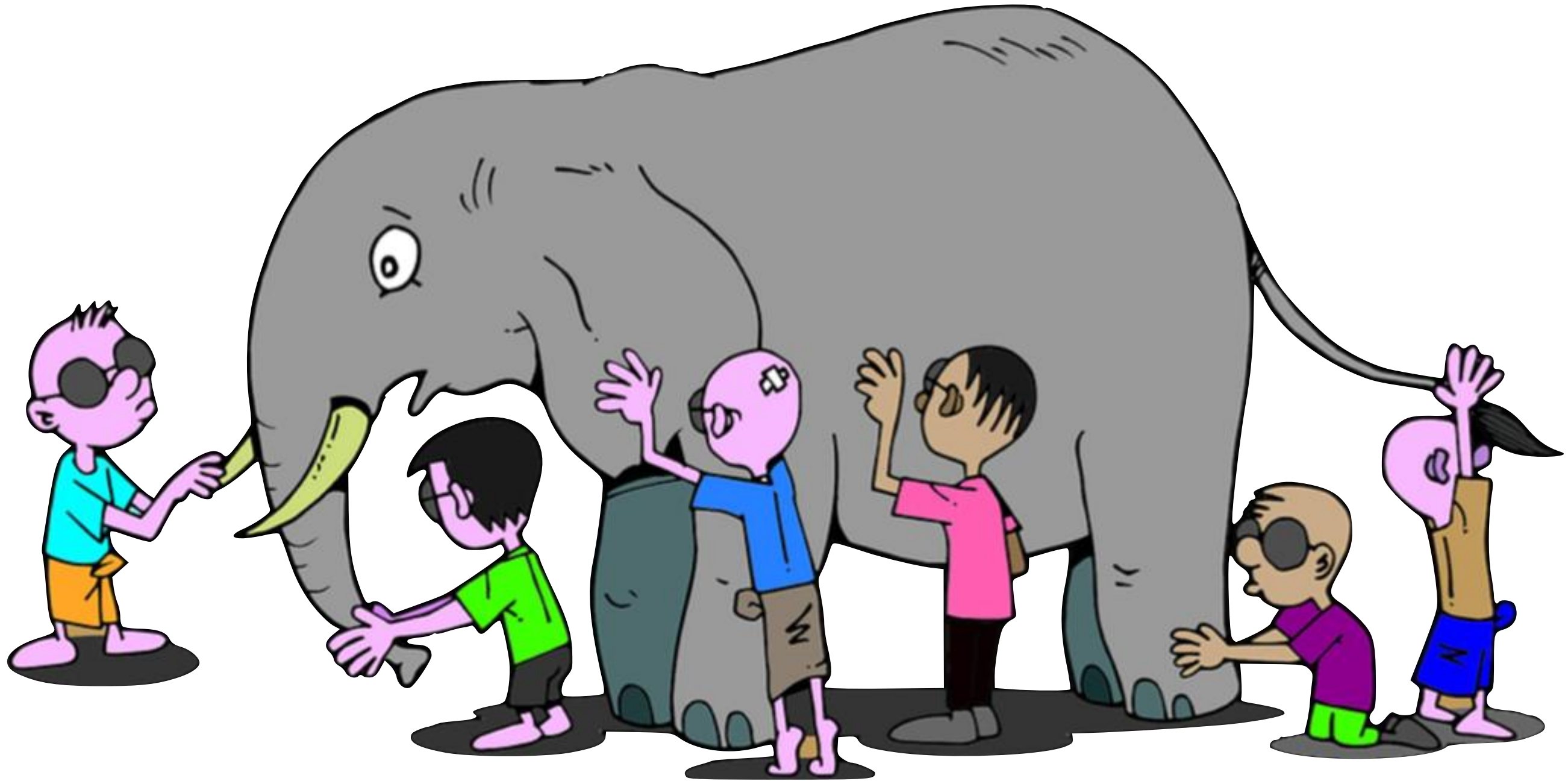# 微服务架构开发及平台演进

讲师：薛军

—— *"无论是微服务还是无服务器架构，逆向思考才能帮助我们拨开迷雾，理解本质"*

当我们谈微服务的时候，我们在谈论什么？

**达成统一的认知**

你眼中的 "微服务"

企业为什么需要微服务？

WHAT

Every organization on the planet knows WHAT they do. These are products they sell or the services

HOW

Some organizations know HOW they do it. These are the things that make them special or set them apart from their competition.

WHY

Very few organizations know WHY they do what they do. WHY is not about making money. That's a result. WHY is a purpose, cause or belief. It's the very reason your organization exists.
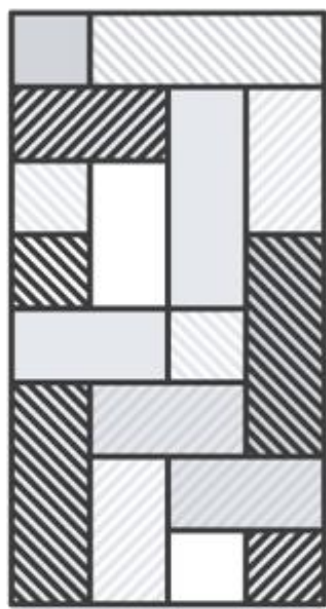
解决方案

VS

问题

aws
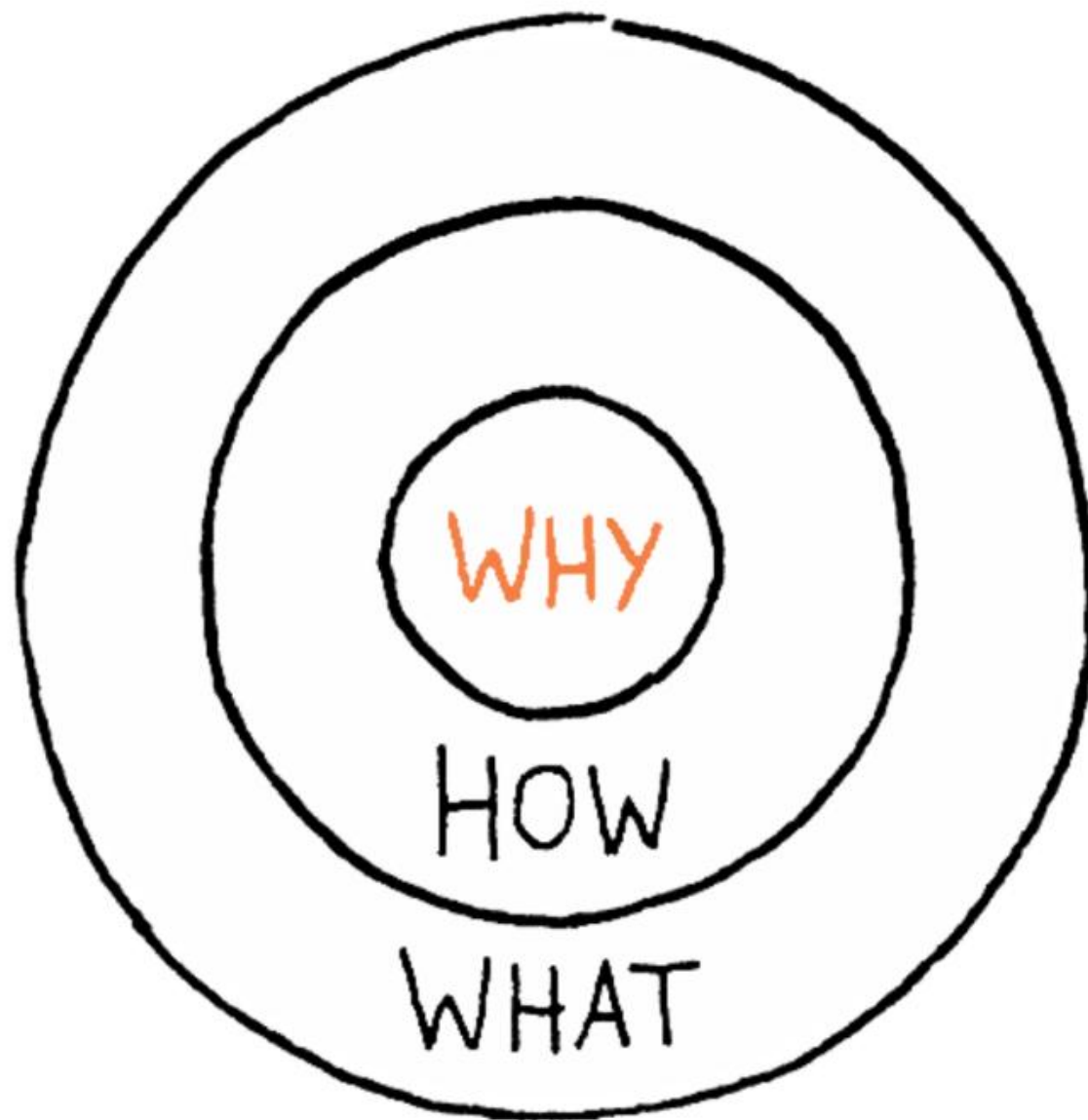
从单体到微服务转型的必然
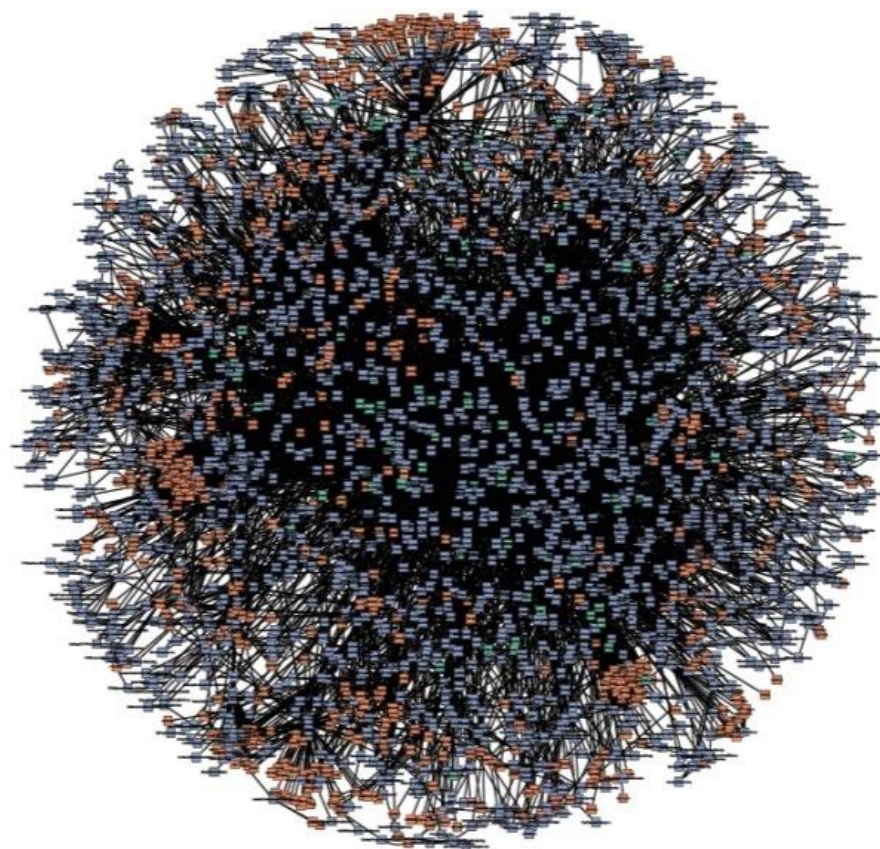
众多的微服务 + 小团队（**2 pizza teams**）

一个紧密耦合的庞大
应用+竖井式团队

"创新乏力"

重复建设

改不动

2001

慢，贵

2009

微服务成功背后的企业机制、文化和组织支撑

新闻稿

常见问题

图例

- **机制**
  - 逆向工作法流程
  - 安灯拉绳 (Andon cord)
  - 错误纠正

- **架构**
  - 微服务架构
  - 松耦合应用
  - 自服务平台 – 没有门卫

- **文化**
  - 人：我们雇佣建造者、创新者、创业者
  - 信念：亚马逊领导力准则

- **组织**
  - 试验
  - 双披萨团队

从三个维度进一步解读微服务

collaborate

accept

EMPATHY

iterate   synthesize

TEST   DEFINE

show   stoke

PROTOTYPE   IDEATE

select
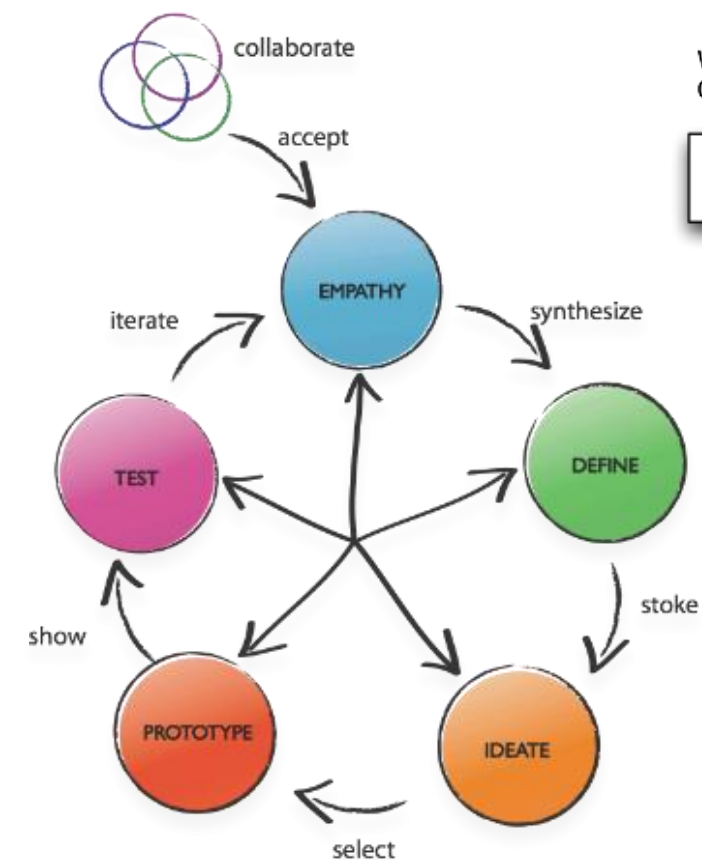
设计思维

逆向工作法

Which one is the best for my application?
Can I automatically figure this out and configure and deploy it.

100,000 Paths
through the stack

200 BigData
Environments

50 Operating
Environments

10 Hardware
Configurations
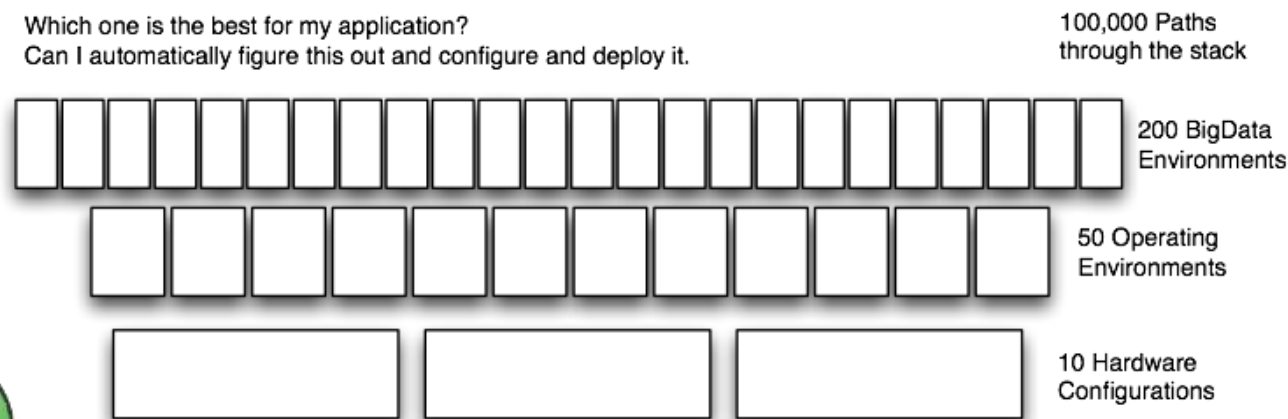
优良架构方法

演进式迭代
领域架构
数据架构
事件驱动

云交付和持续运营

技术平台
"零"运维
AI 运维

设计思维：从客户出发，逆向工作法

建造者，创业者，创新者或产品经理视角

Amazon 逆向工作法 - 客户驱动价值创新
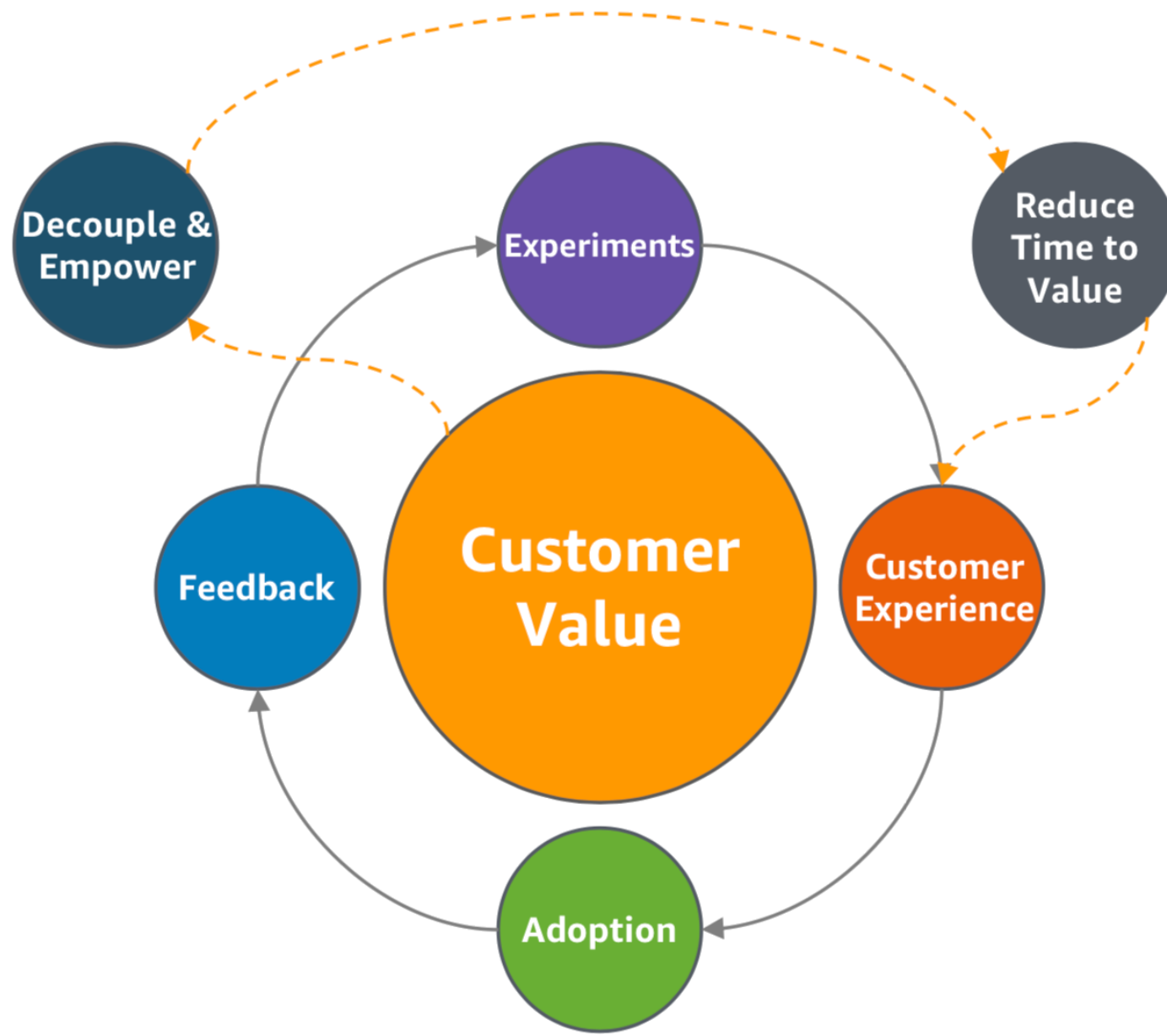
聚焦：

如何从客户角度分析、拆解问题域？

如何利用快速原型法从客户反馈驱动客户价值飞轮？

AWS 数字化创新工作坊

逆向工作的 5 个问题

- 谁是 客户?

- 客户的问题或机遇是什么?

- 客户最重要的效益是什么?

- 你怎么知道客户 需要 或 想要什么?

- 客户会有什么样的 体验?

从客户出发
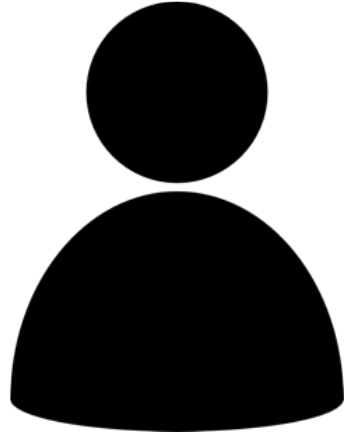
事实 *你的目标客户的真实信息*	**痛点** *你的目标客户的痛点是什么？*
行为 *客户当前解决痛点、达成目标的行为*	**目标** *客户在努力实现什么目标？*

问题域分析产出物：六页纸陈述

新闻稿

- 聚焦客户需求
- 引述客户发言是关键
- 穿越到未来: 远见卓识
- 避免拗口的专业术语
- 简单清晰地描述

常见问题

- 包括客户常见问题和利益相关者常见问题
- 包括难以回答的问题
- 提前分享新闻稿以收集问题

图例

- 粗略的想法 – 粗略的图画
- 精确性与想法的成熟度匹配
- 不要害怕有争议性
- 创造讨论

下一步：快速原型构建正向反馈环

针对某个特定的业务挑战，尝试通过一个双披萨团队，产生创新想法、快速试验和验证

亚马逊创新研讨会

新闻稿概述

新闻稿/常见问题原型

新闻稿/常见问题上线

发 散

汇 聚

问题 / 机遇

专注于客户

逆向工作

一个颠覆者会如何解决这个问题？
它是如何为客户提高了标准？

开发原型，根据反馈循环迭代直到上线

快速原型法

DAY 1   DAY 10   DAY 20   2+ MONTH
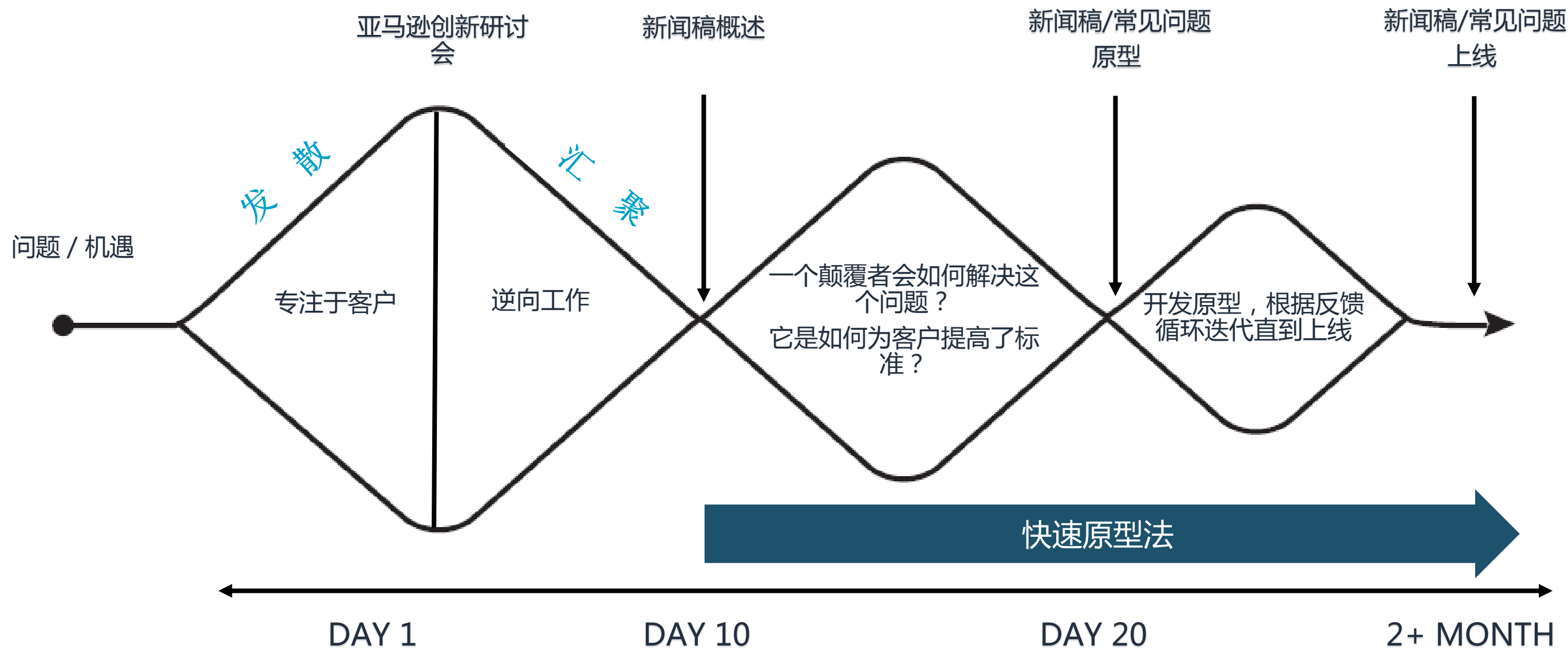
优良架构方法指导微服务架构演进

架构和技术落地视角

微服务架构兴起

微服务是小的、自治的服务一起工作.

A monolithic application puts all its functionality into a single process...

A microservices architecture puts each element of functionality into a separate service...

... and scales by replicating the monolith on multiple servers

... and scales by distributing these services across servers, replicating as needed.
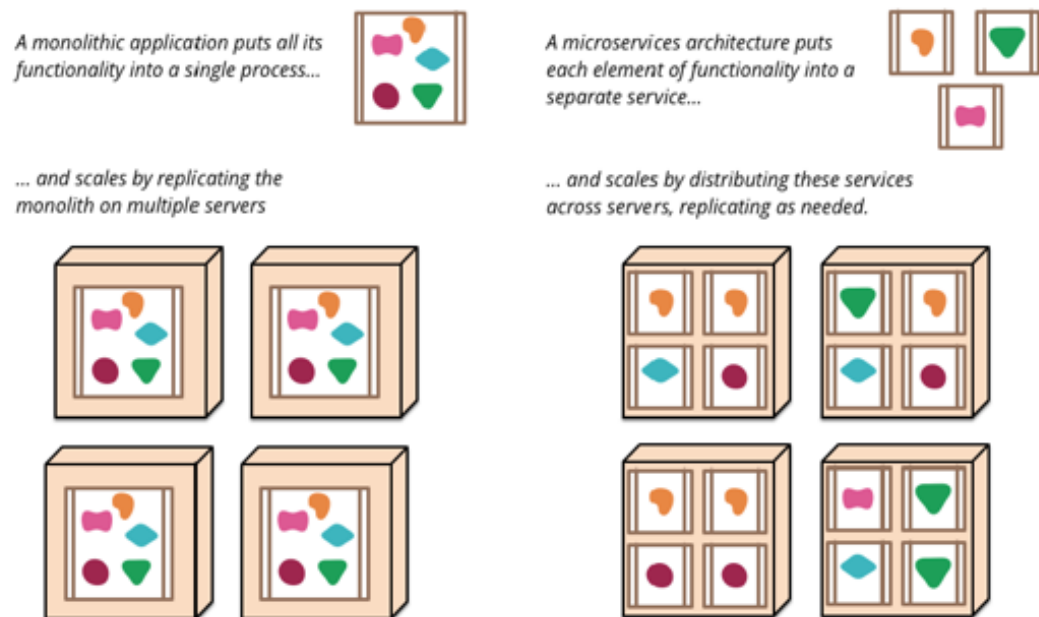
黄金法则: 你能在不改变任何情况的前提下对服务进行更改并且自动部署它吗？

将那些因相同原因而改变的事物聚集在一起，将那些因不同原因而改变的事物分开，*单一责任原则*

微服务架构的优势

- 技术异构性
- 恢复能力
- 按照服务扩展
- 易于部署/部分部署
- 与DevOps紧密整合
- 提高扩展性、可用性和容错性
- 与组织架构一致
- 与业务领域联系更紧密

微服务架构设计
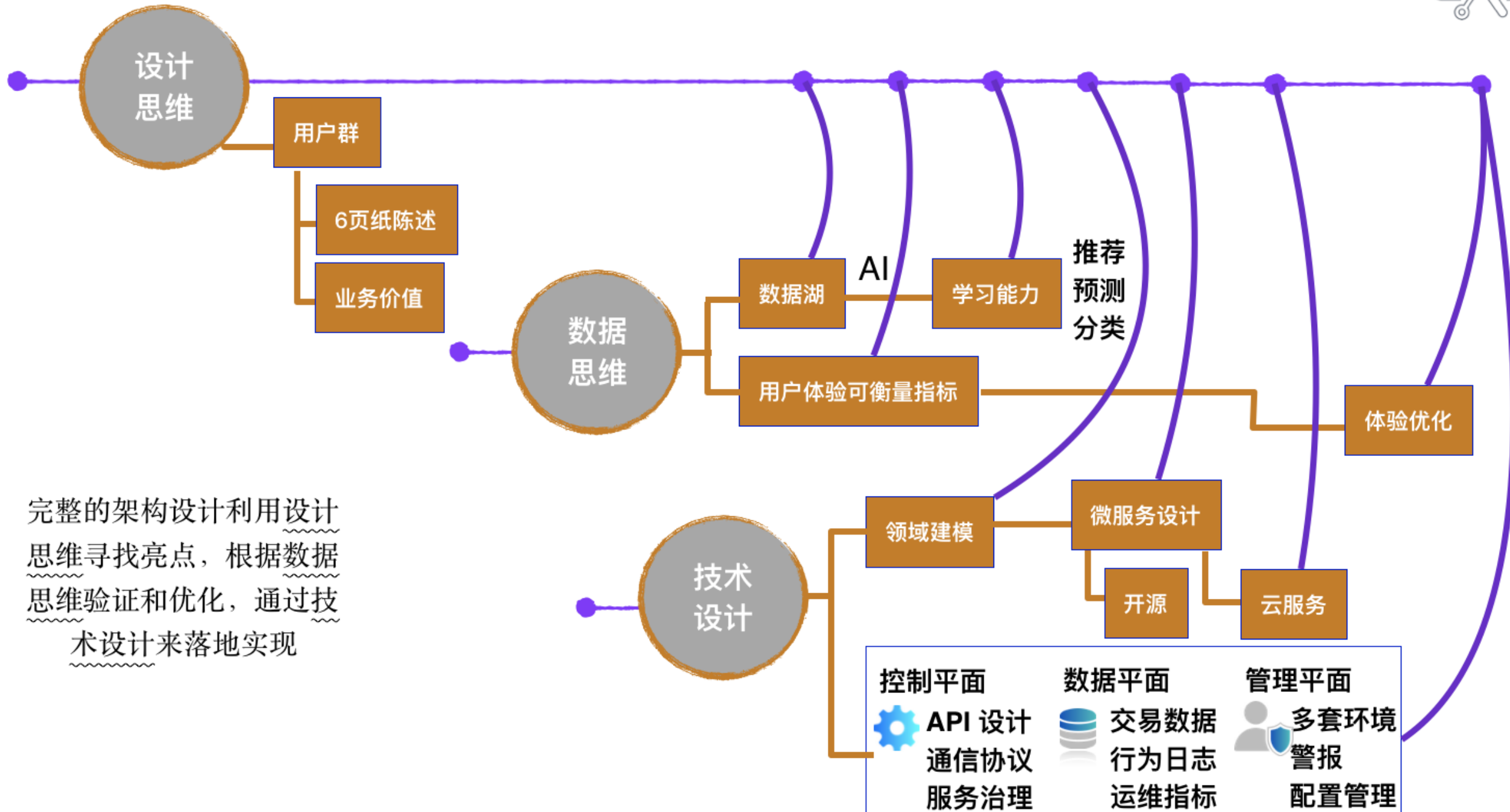
领域建模到服务拆分-定义服务边界

服务边界约束：
- 业务范围（高内聚）
- 团队资源（2个披萨团队）
- 交付时间（冲刺迭代，快速上线）
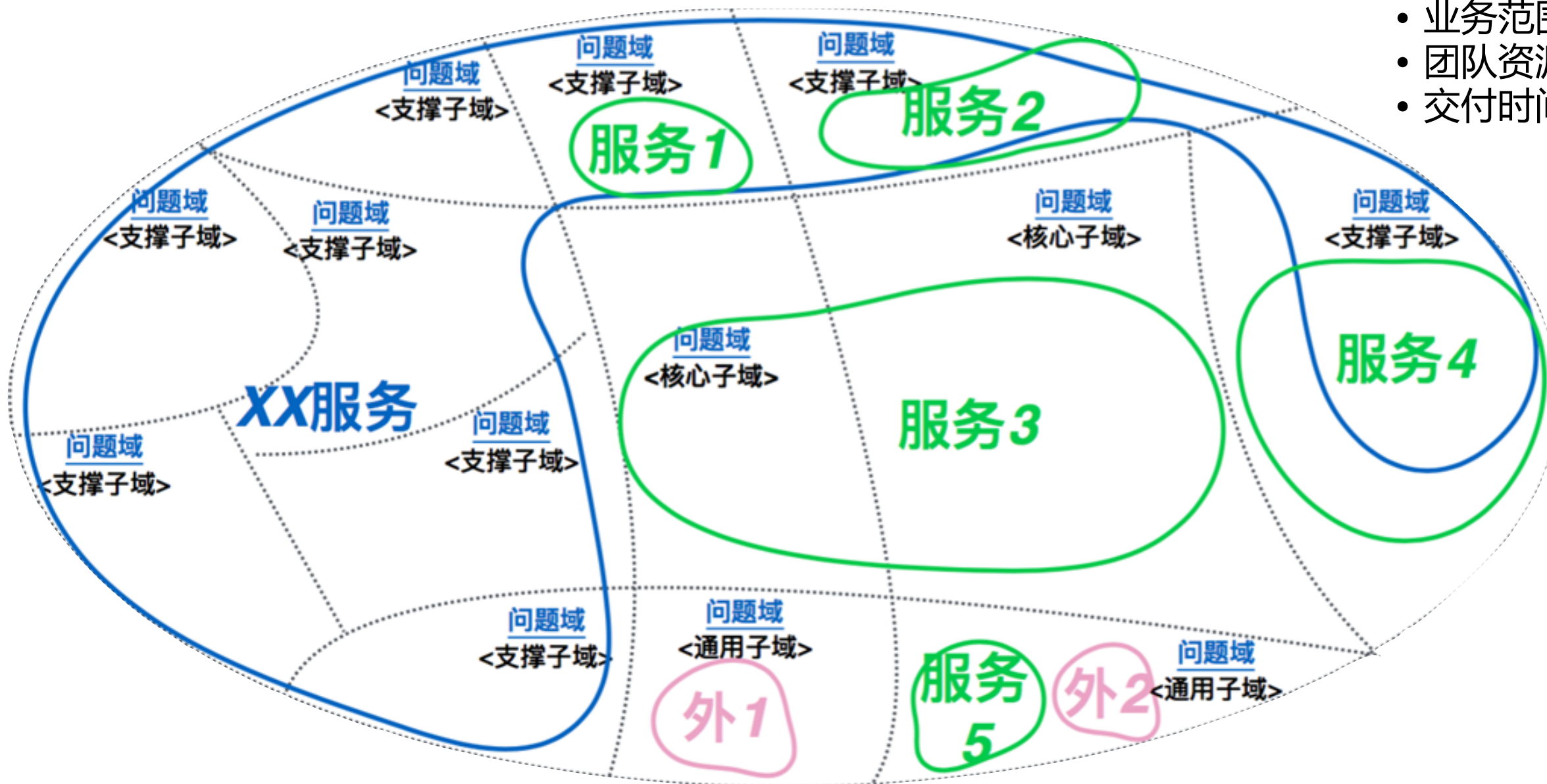
微服务和2个披萨团队

- 我们称之为 "Service Team"
- 决策权交给各个服务（而不是集中决策）
  - 产品开发计划（Roadmap）
  - 开发工作
  - 运维和客户支持工作
  - 以产品功能划分服务 (You build it, you run it.)
- 服务之间只能通过API交互（不可以直接访问或修改其他服务的数据）
- 面向失效的设计
- 基础架构自动化
- 不断演进 （Evolutionary Design）

康威定律：设计系统的组织，其产生的设计和架构等价于组织间的沟通结构

aws

单个微服务实现的参考架构：端口适配器
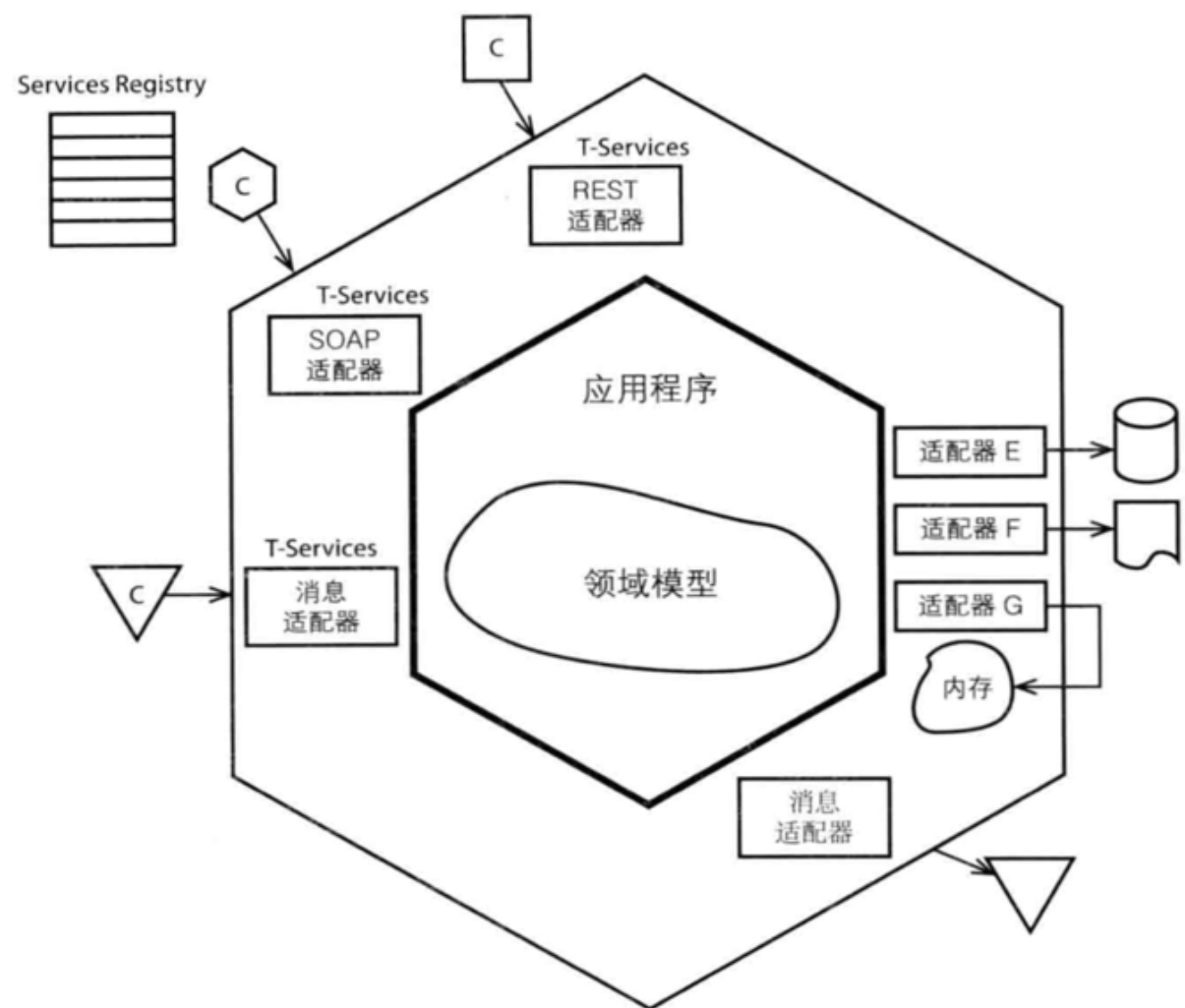
图4.5 一个支持SOA的六边形架构，其中包括REST、SOAP和消息服务。

微服务 ：自治、内聚、松耦合，分布式

微服务数据架构
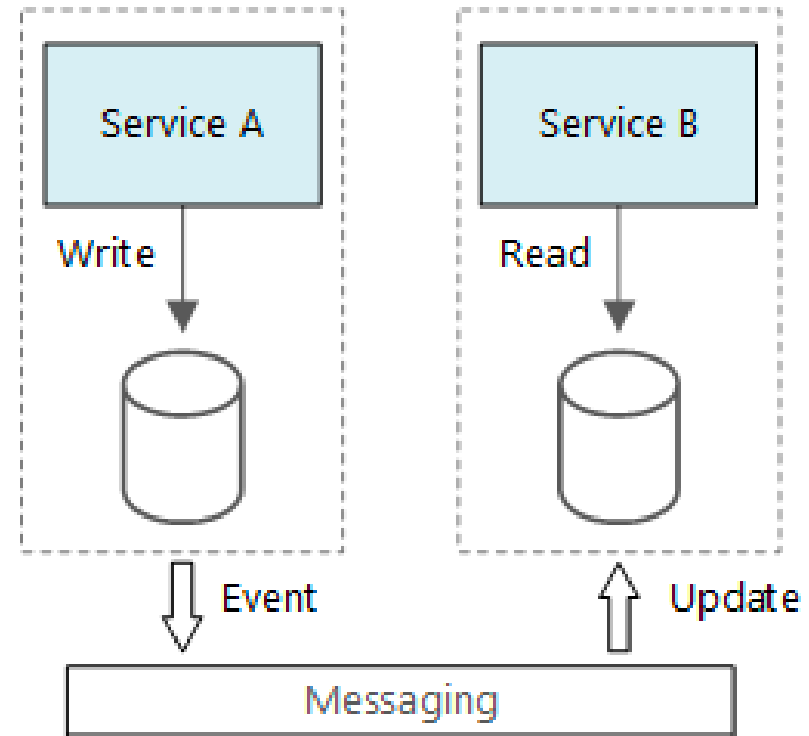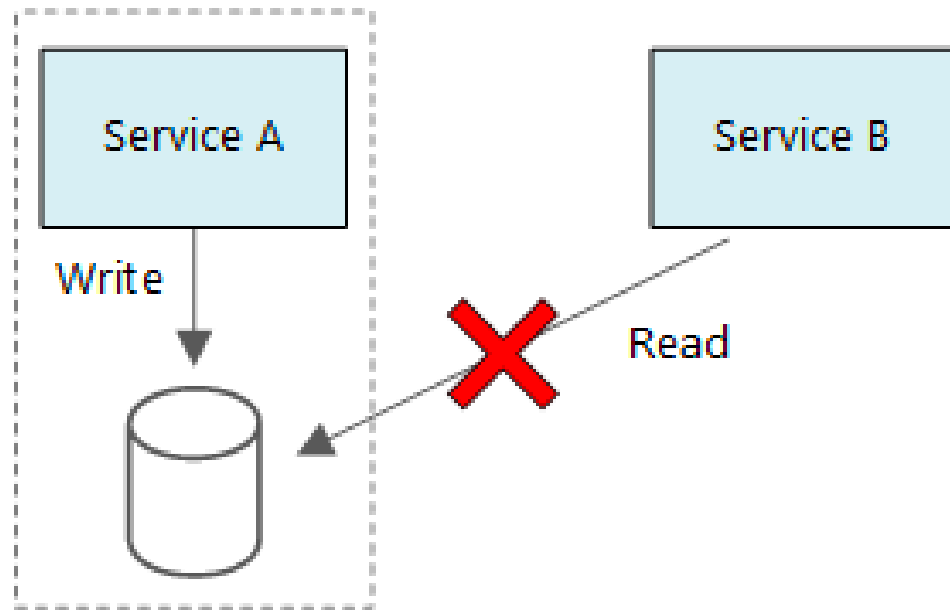
多个微服务协作实现一个业务需求

读写分离
事件驱动，最终一致
CQRS：命令和查询分离

微服务架构之服务通信 IPC

协议		交互模式	
		One-to-One	One-to-Many

同步
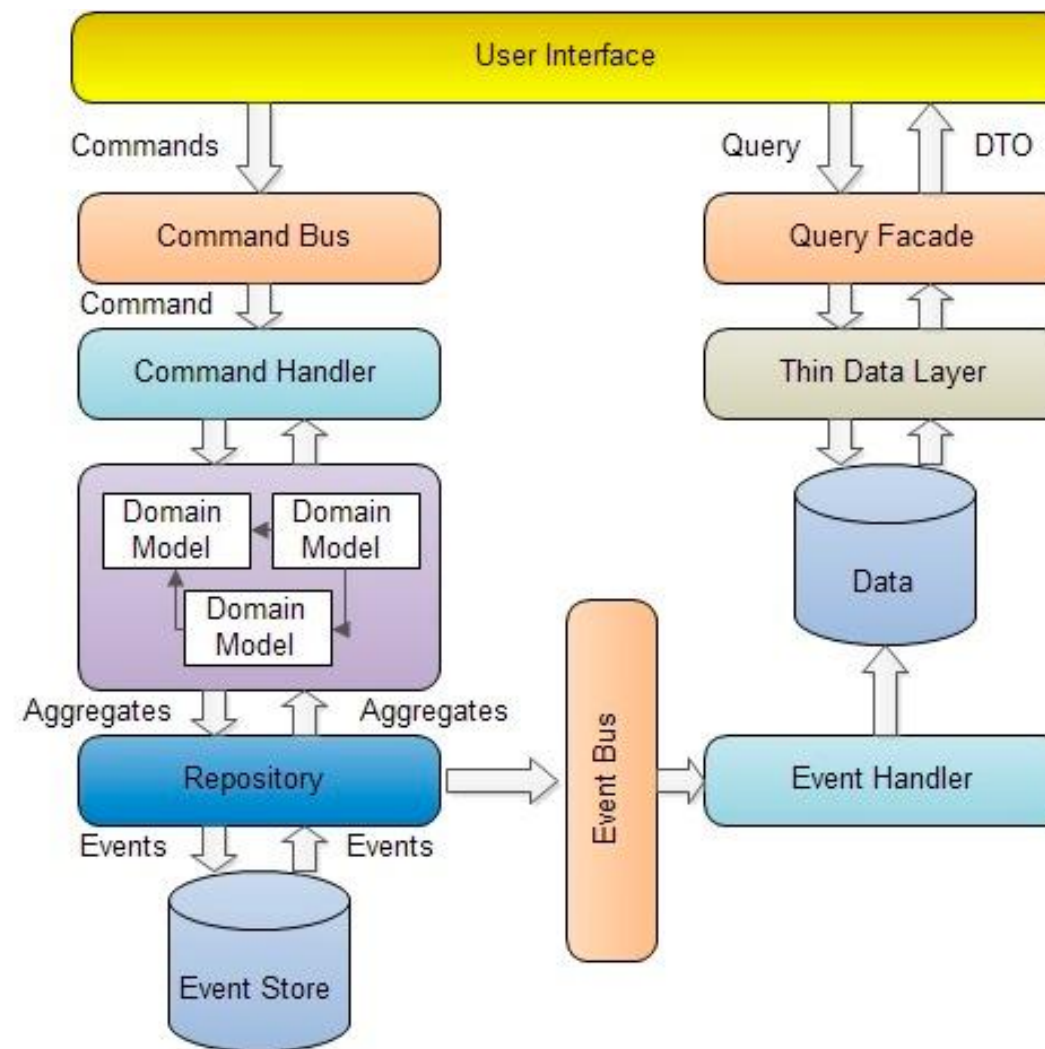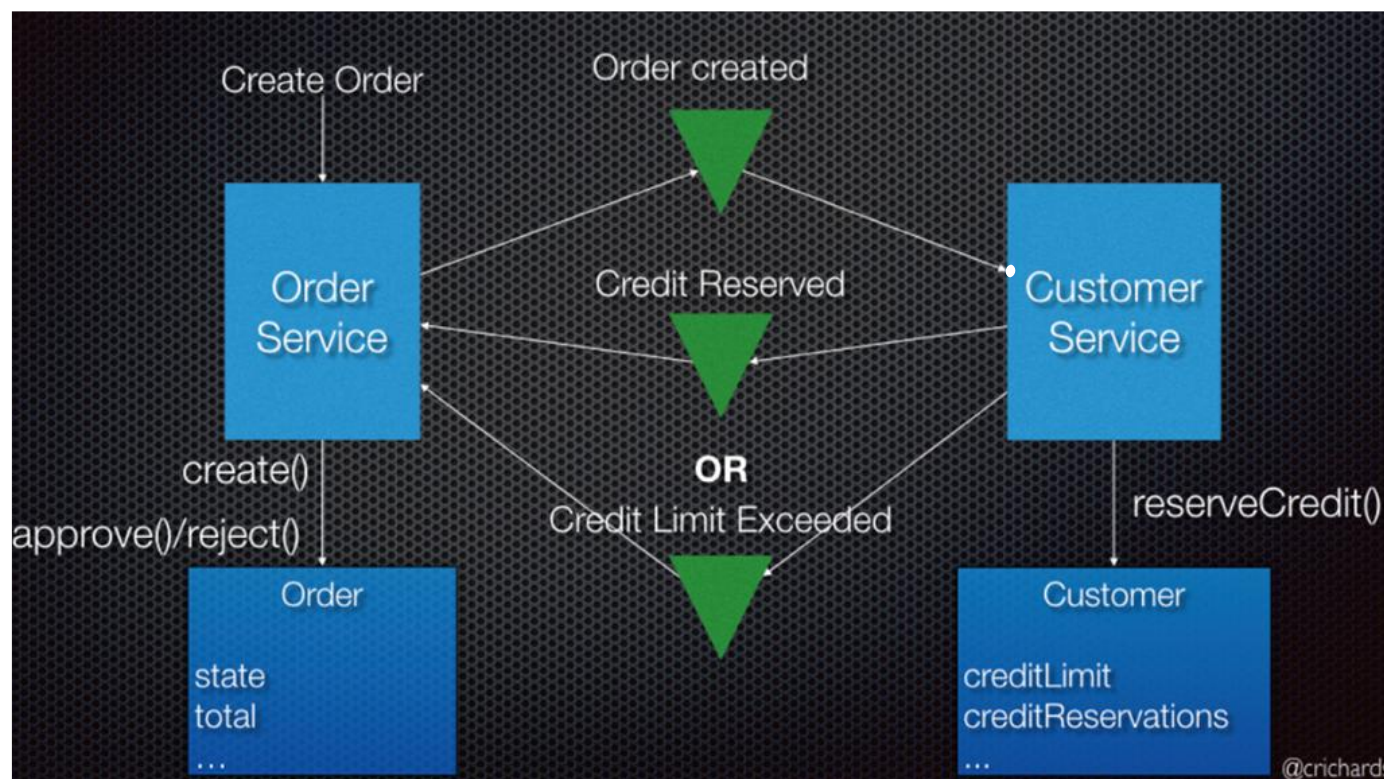
- **HTTP 1.1 or 2.0**
- **REST API**
- **JAVA RMI**
- **Hessian**
- **Thrift**
- **gRPC**

IDL

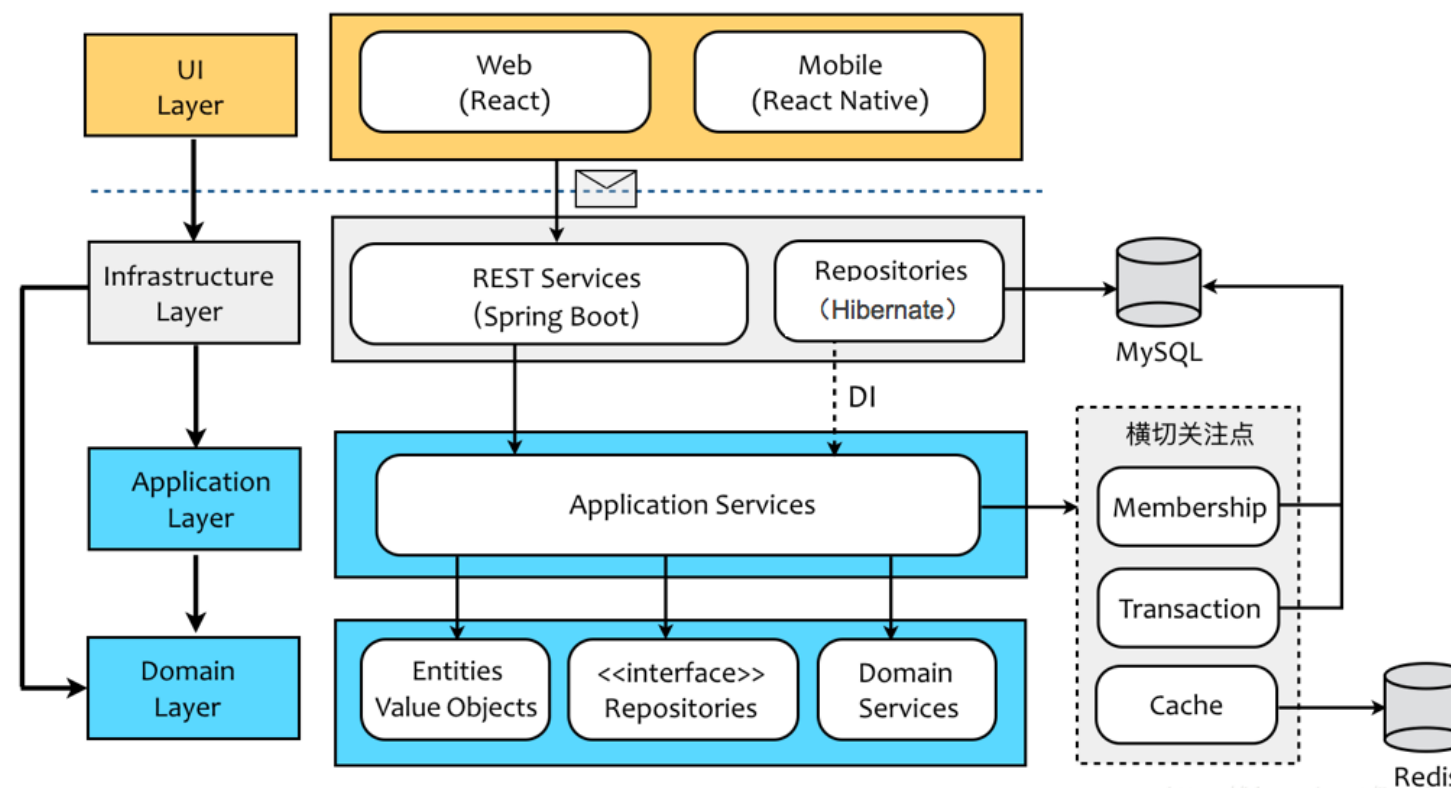**Text**: Json/XML
Binary:
- Avro,
- Protocol Buffer

请求/响应　　　　　　　　　——-

- 寻址　　　　- 序列化和反序列化　　　　- 网络传输

异步

- **MQTT**
- **JMS**
- **AMQP**
- **STOMP**

通知　　　　　　　　　发布/订阅
请求/异步响应　　　　发布/异步响应

aws re:Invent

aws

微服务架构之通信异常处理

- 快速失败
- 流量限制
- 指数回退方式重试
- 断路器模式（Circuit Breaker Pattern）
- 幂等
- **提供回滚**：当一个请求失败后可以进行回滚逻辑。例如，返回缓存数据或者一个系统默认值。
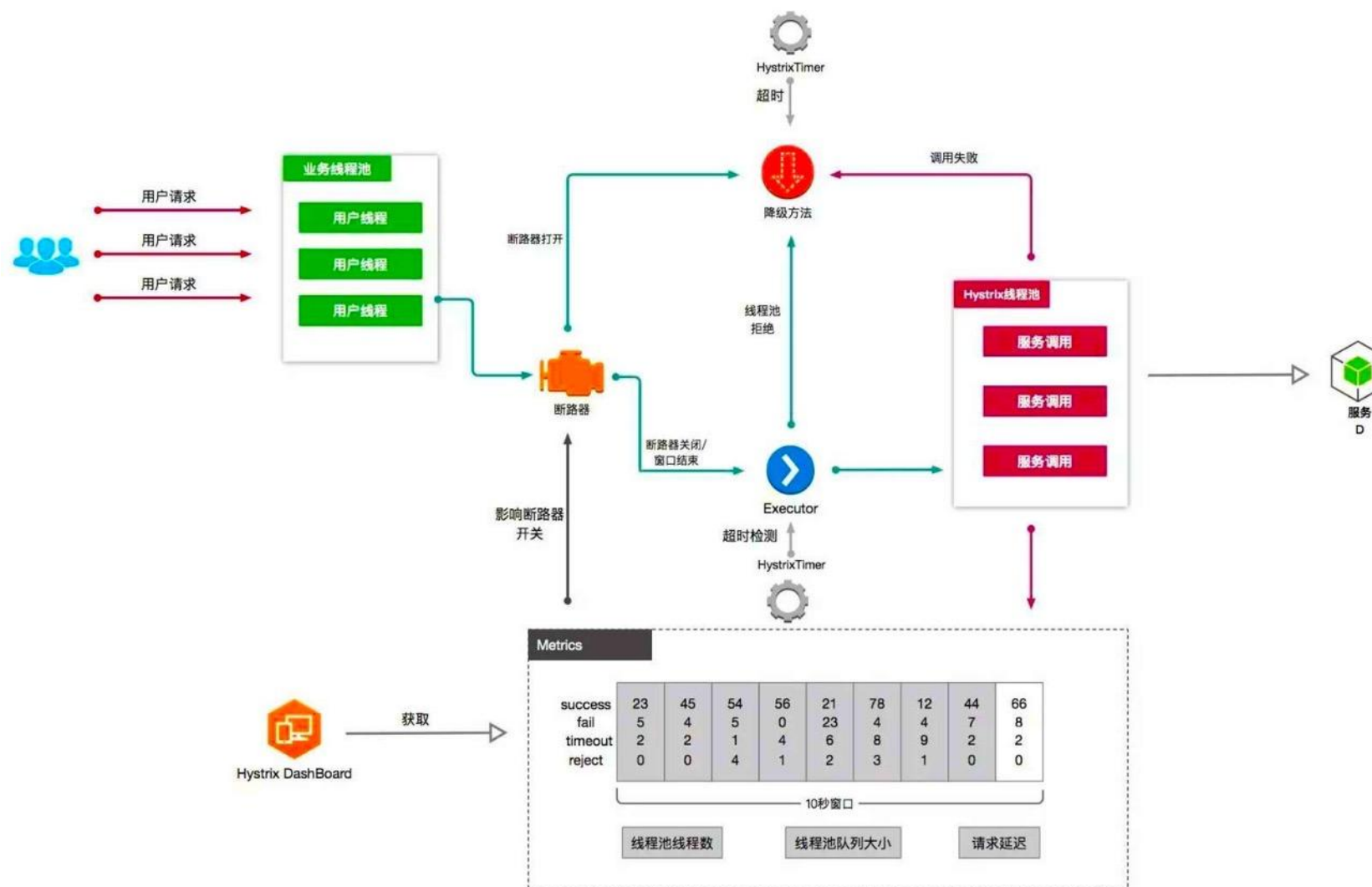
微服务架构之服务注册发现

- 注册表
- 自注册 vs 第三方注册
- 客户端服务发现 vs 服务器端
  服务发现
- **Service Mesh (**服务网格）
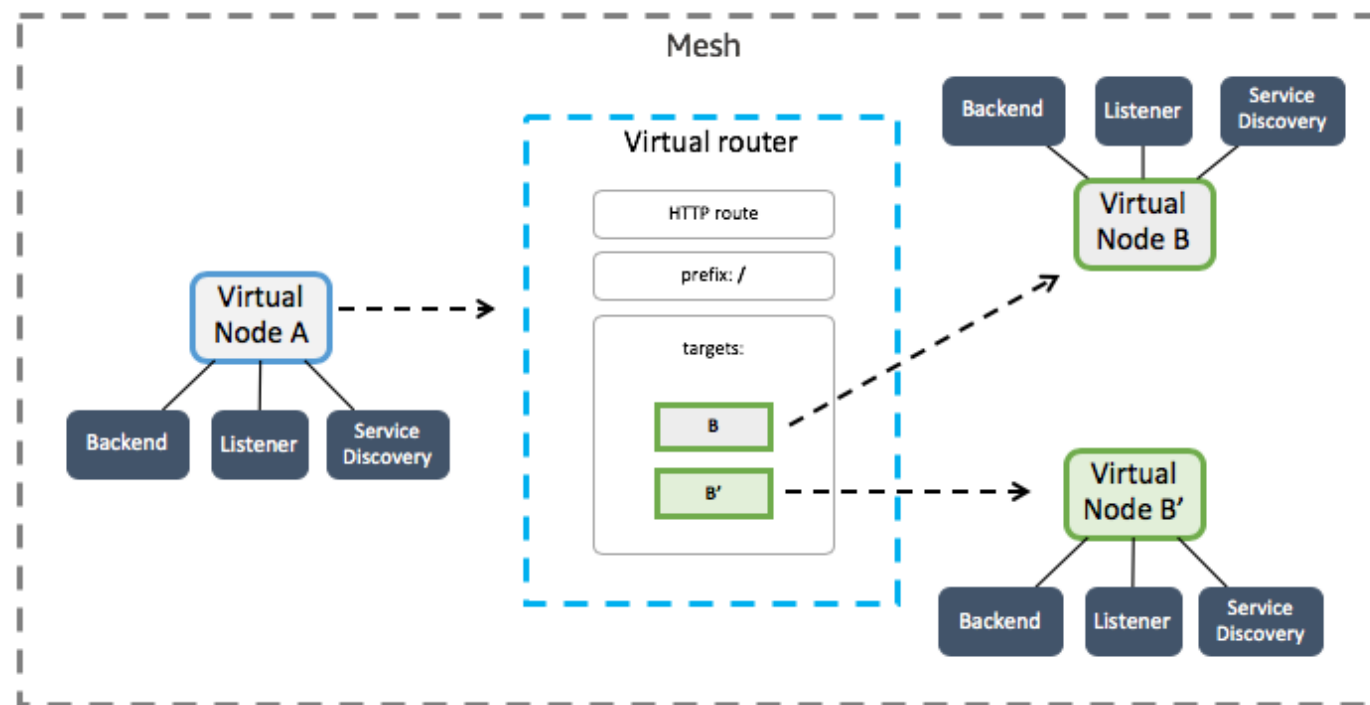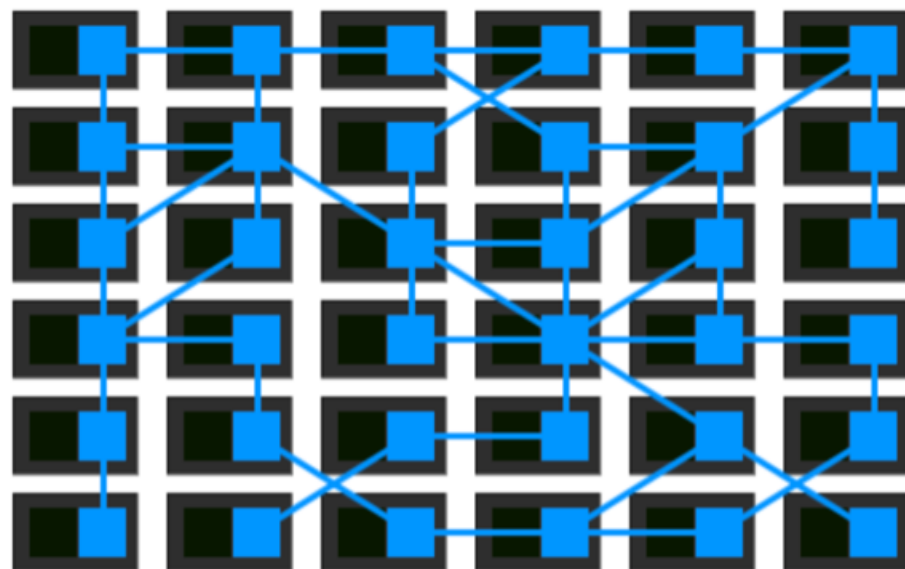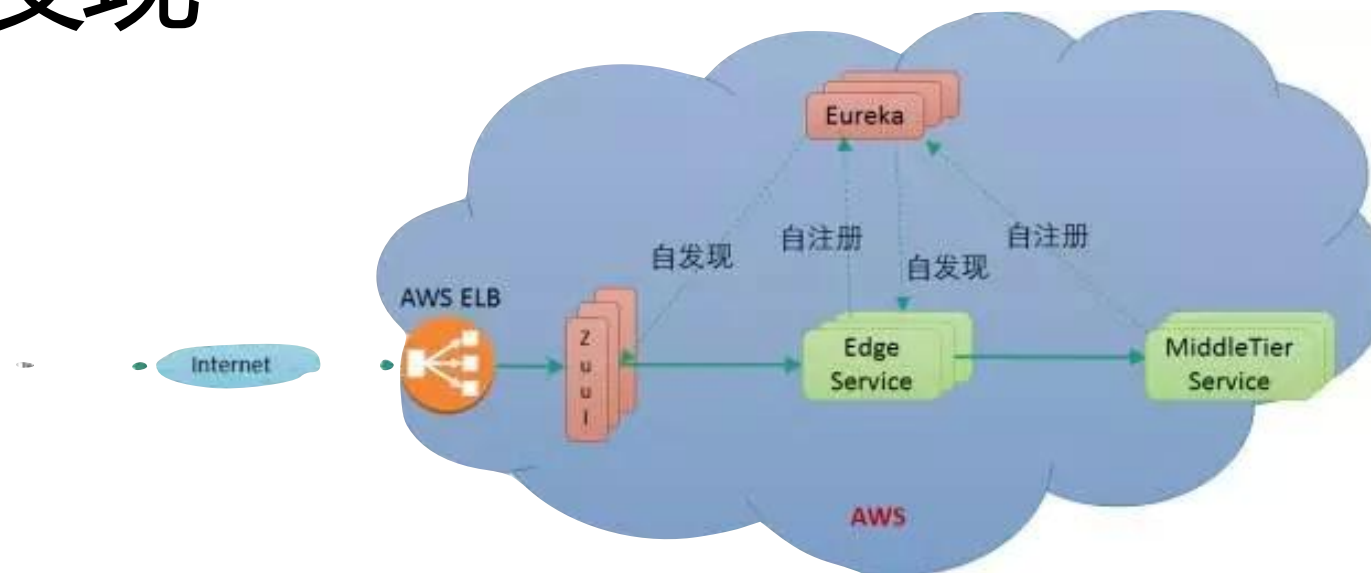
微服务架构之 API 分层
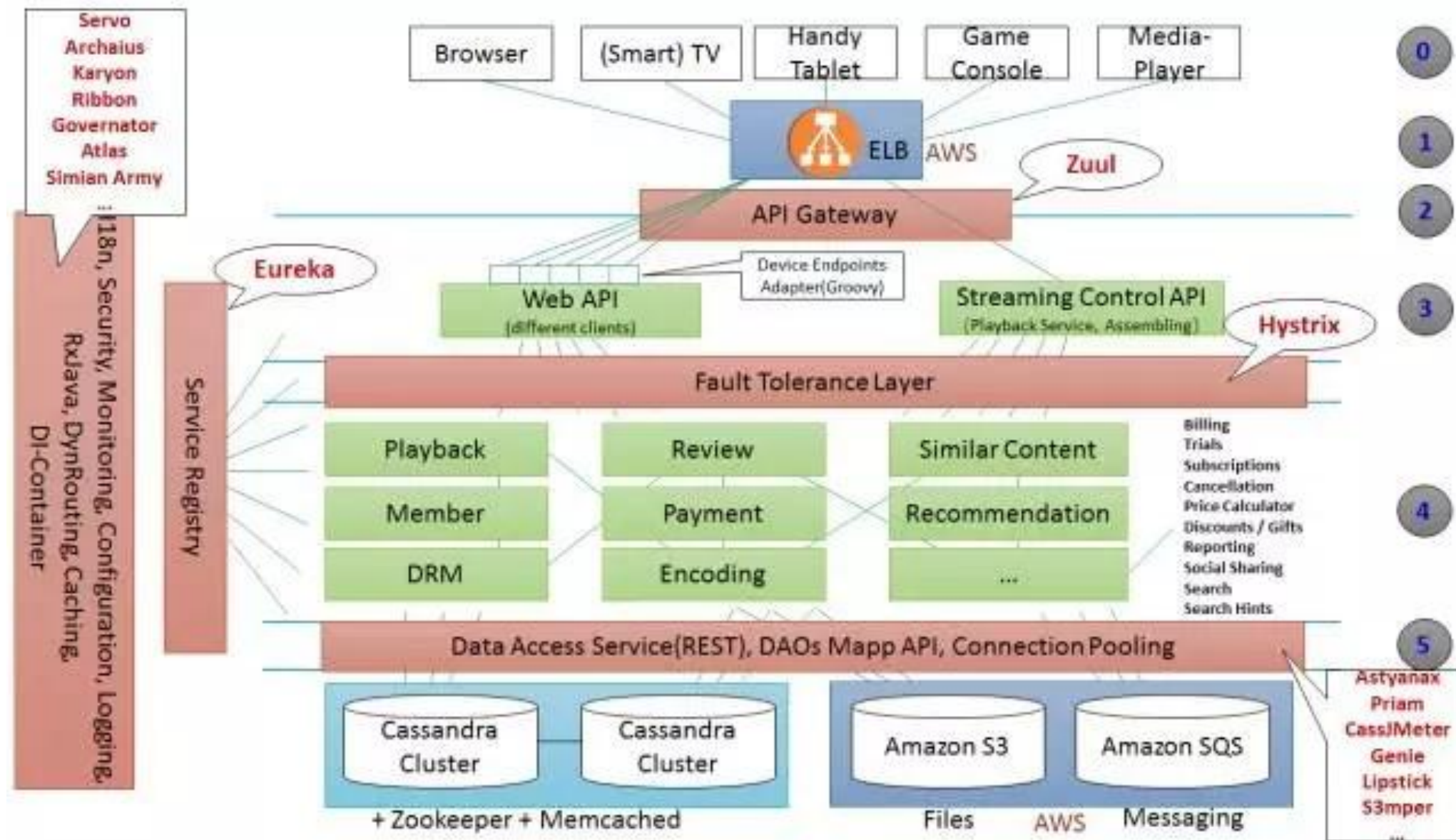
服务分层：
- API Gateway 模式
- 智能路由
- 服务注册发现机制
- 实例注册发现机制

微服务架构之 API 编排和聚合
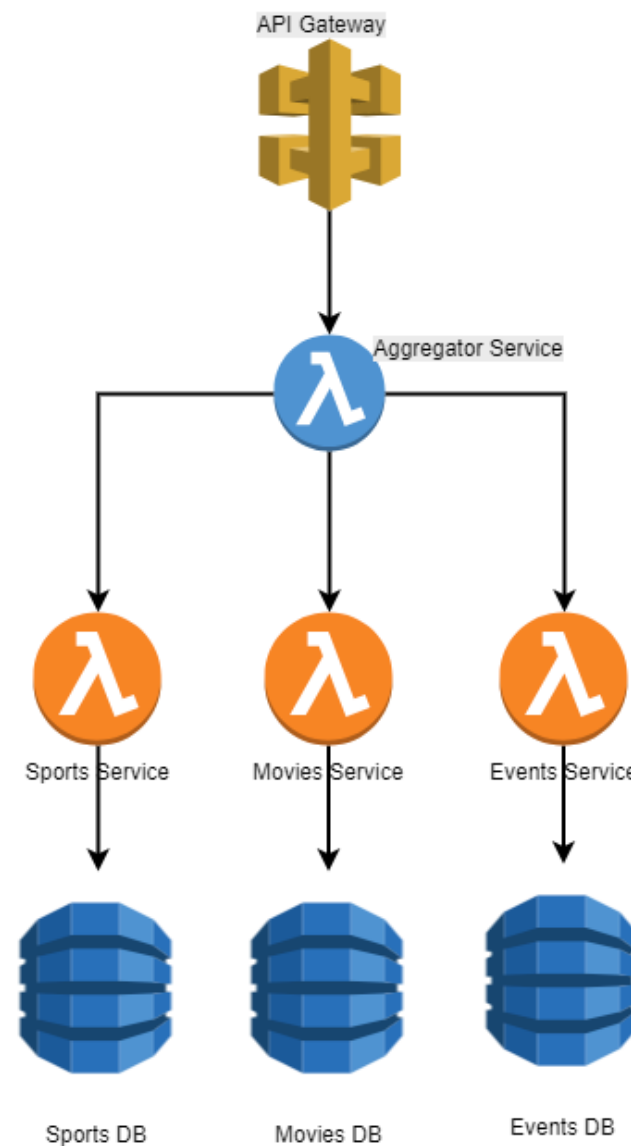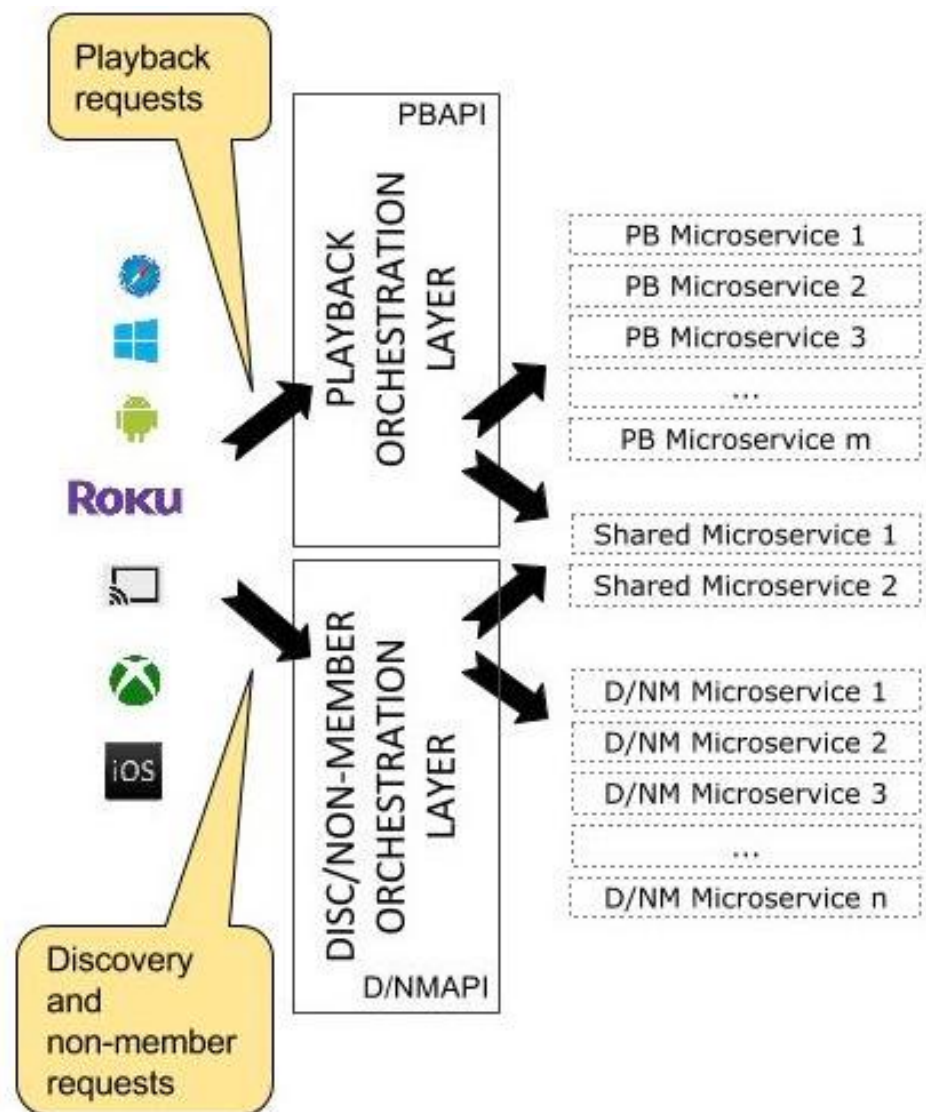
API 编排和聚合：
- 响应式编程 RxJava
- Lambda：Service Of Service
- CQRS：查询和命令关注点分离

Source: Netflix tech blog

微服务架构之 API 编排和聚合

API 编排和聚合：
- GraphQL 对于前端应用的优化

Source: https://samnewman.io/patterns/architectural/bff/

aws

微服务架构之可靠性测试

Failing Subscriber

Failure Scope
apply to targeted requests

Failure Metadata
aysnc update from control plane

Internet

Zuul

FIT Service

Request Context
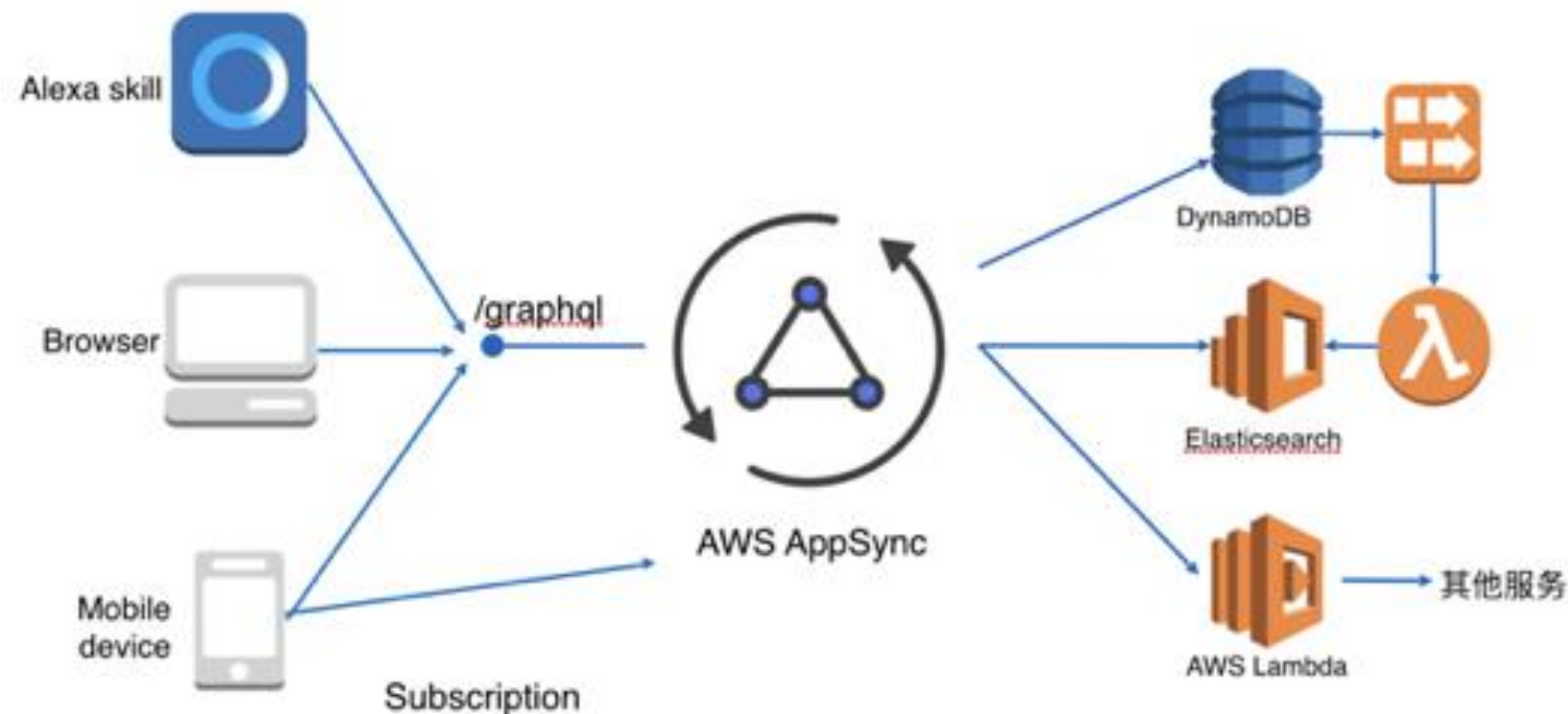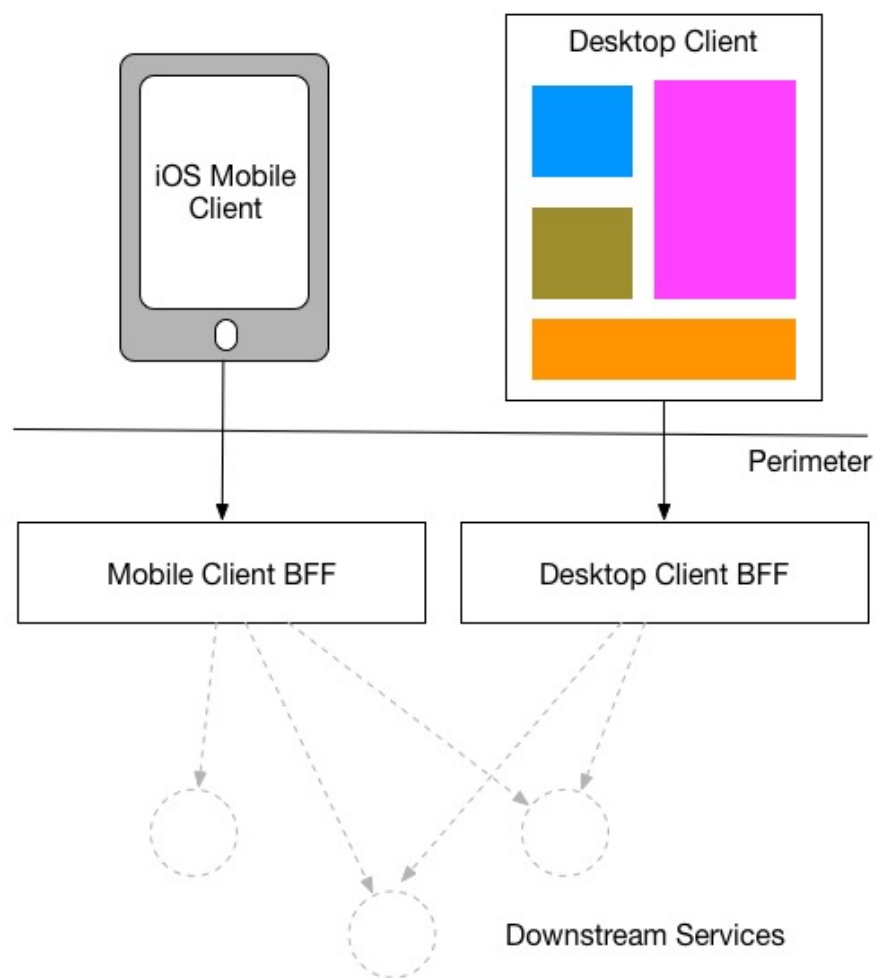failure details
travel with request

API

Hystrix

Ribbon

Network Calls
Simulated Failures
Injection Points

Subscriber

Recommendations

Gateway — 99% → API → Ratings

0.5% → API Control

0.5% → API Exp
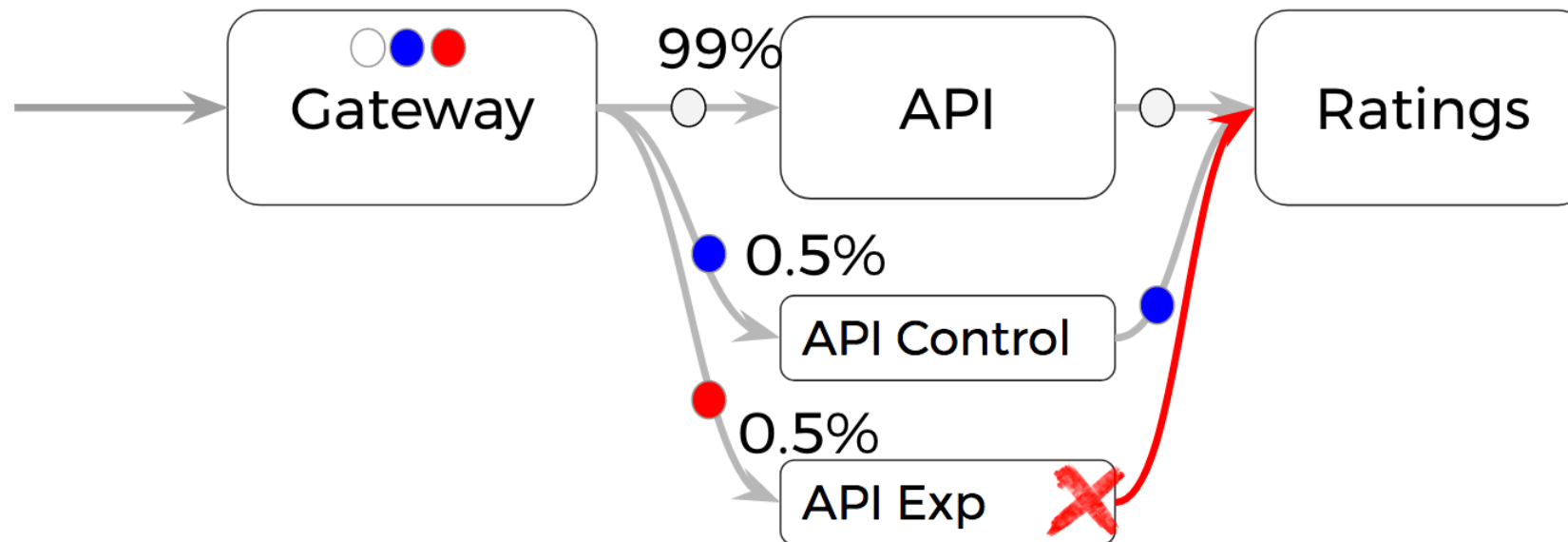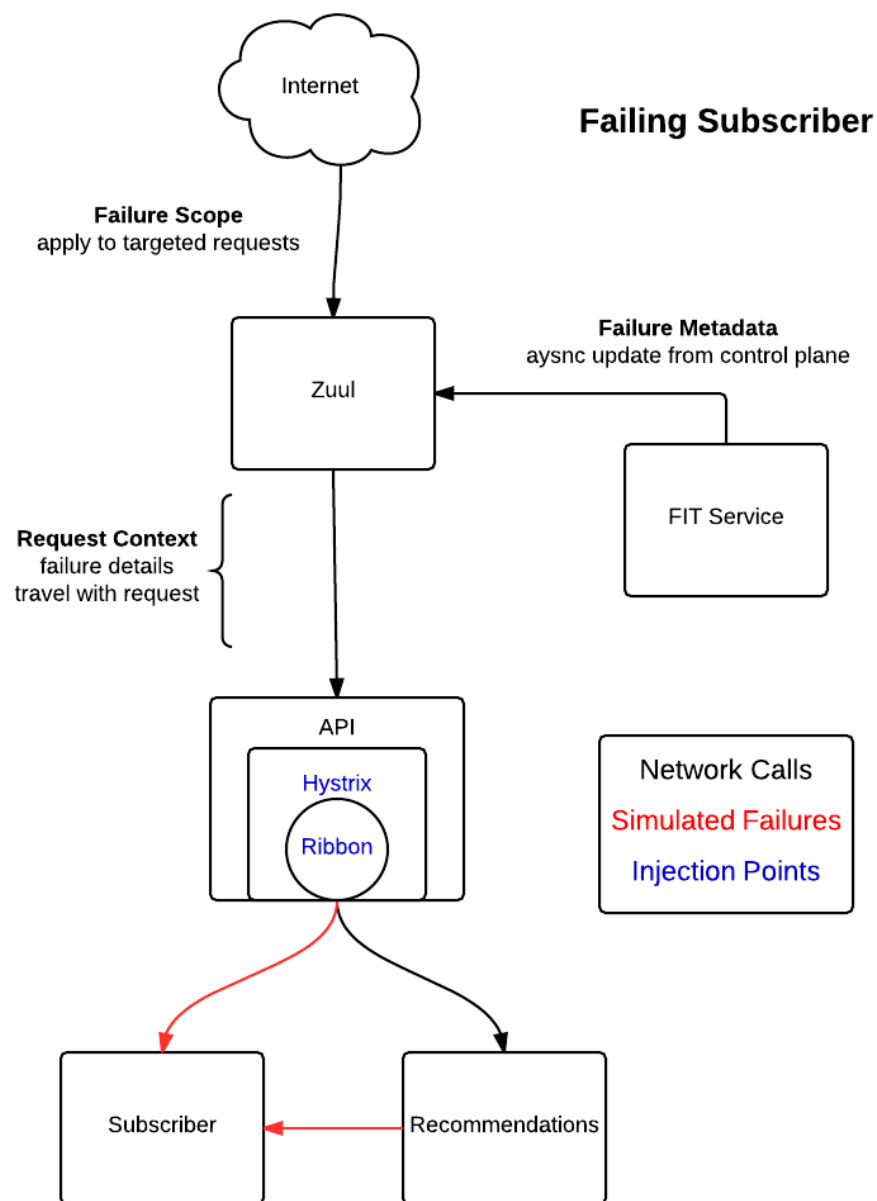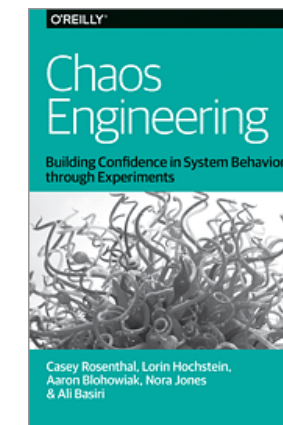
失效注入测试（FIT，Failure Injection Testing）

Source: Netflix tech blog

Chaos Engineering
Building Confidence in System Behavior through Experiments

Casey Rosenthal, Lorin Hochstein, Aaron Blohowiak, Nora Jones & Ali Basiri

微服务架构之规模化管理
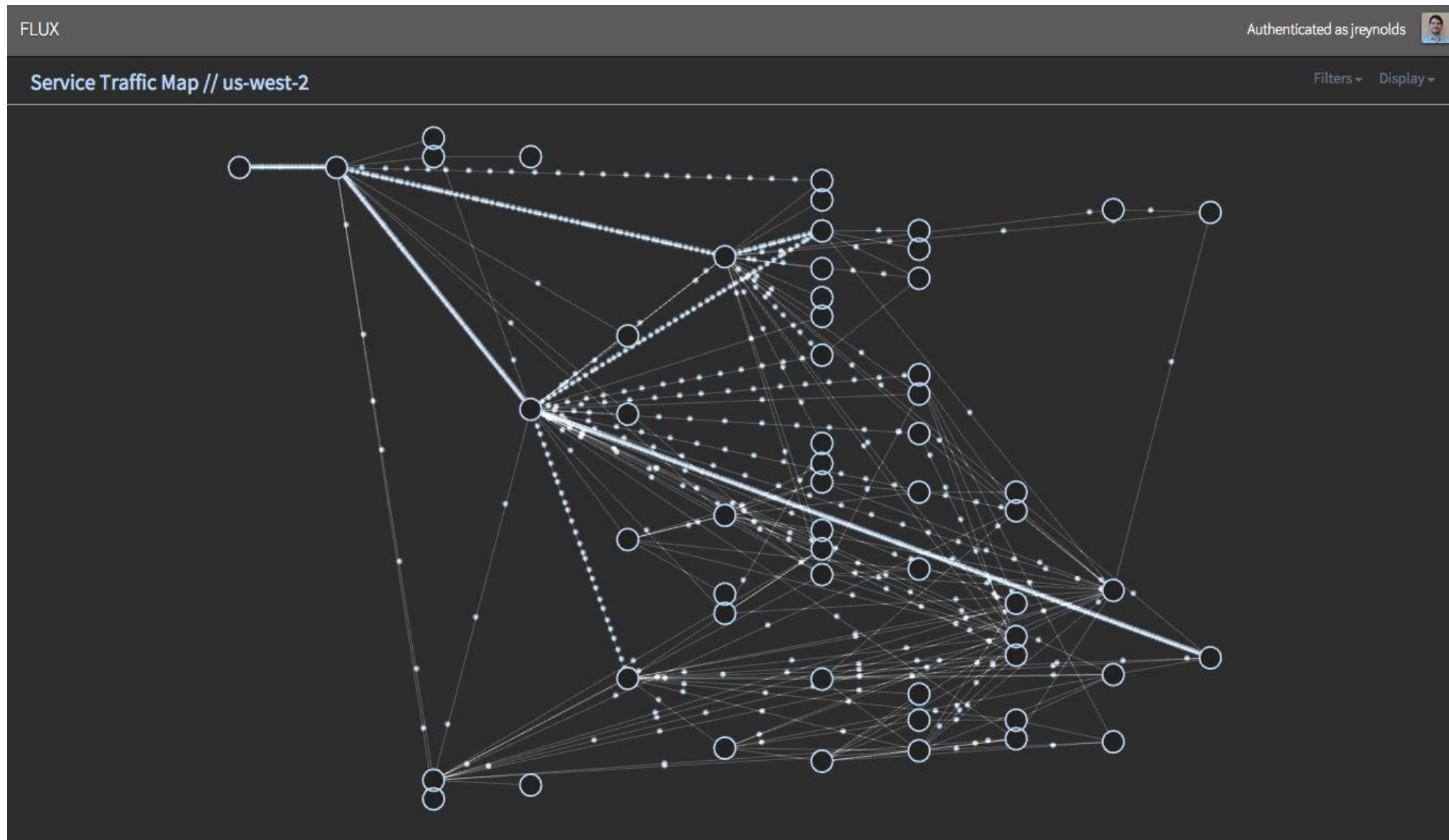
混沌工程及微服务系统洞察能力：
- 实时数据
- 指标：流量、延迟、服务健康度等
- Edge侧的请求情况
- 有能力深入服务间调用情况
- 服务依赖关系

微服务架构之跨区域数据同步

https://medium.com/netflix-techblog/global-cloud-active-active-and-beyond-a0fdfa2c3a45

高尔定律
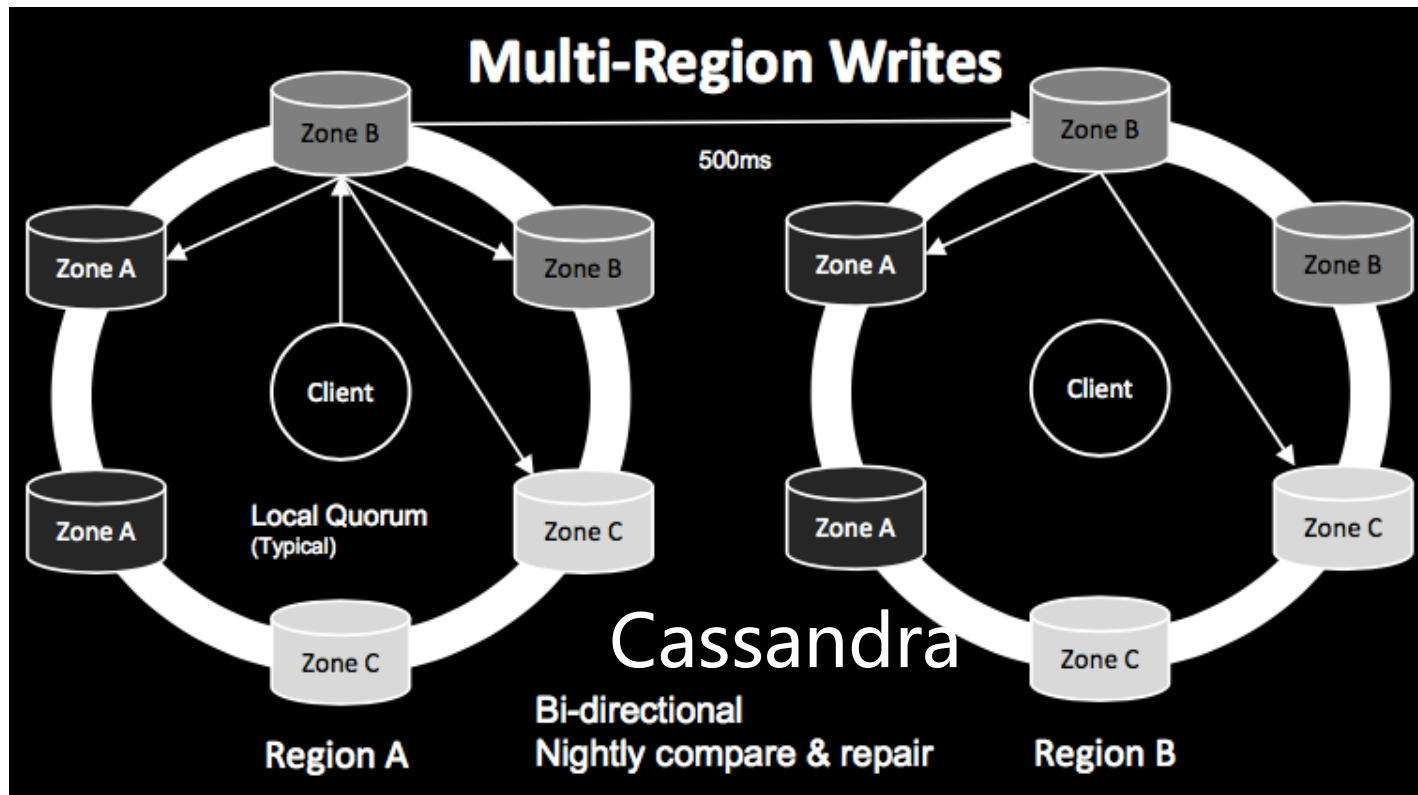
一个可以工作的复杂系统
总是被发现是从
一个可以工作的简单系统
演进而来

AWS 优良架构框架-不断优化和迭代

AWS Well-Architected
https://wa.aws.amazon.com

操作
最佳实践

安全

可靠性
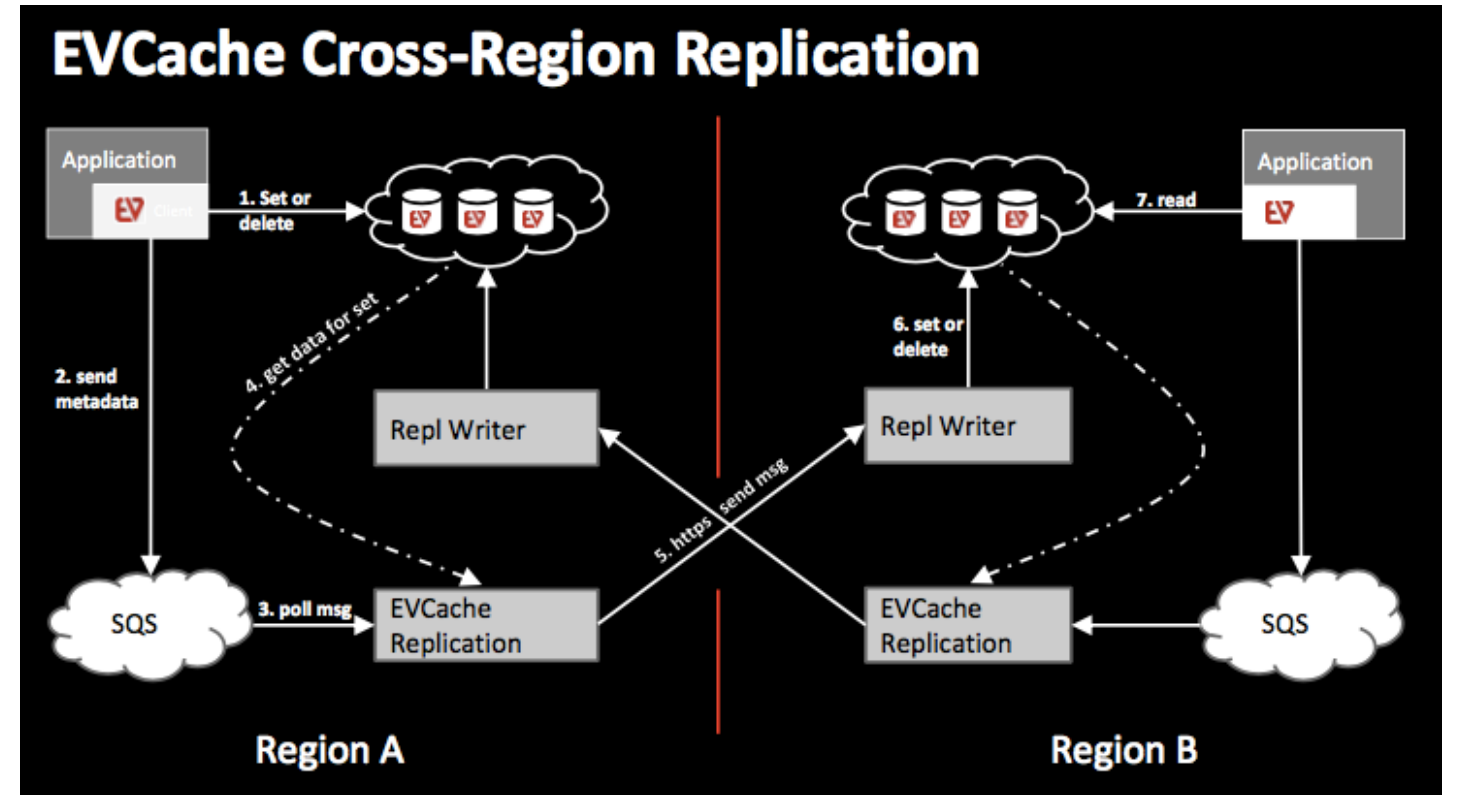
性能和效率

成本优化

微服务 "良好架构" 原则

不要猜测容量需求

以实际生产环境规模执行测试

使用自动化简化构建实验
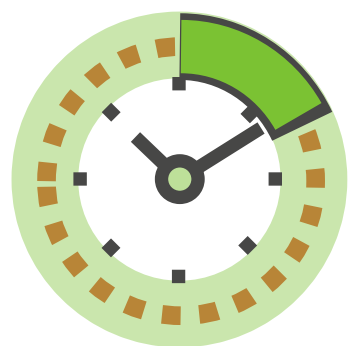
允许逐步完善架构

使用数据驱动你的架构

使用演练改进

云交付：从开源出发到云原生

持续交付和运营视角

时间都花到哪去了?

业务项目
39%

↑
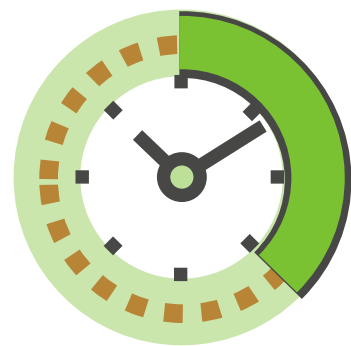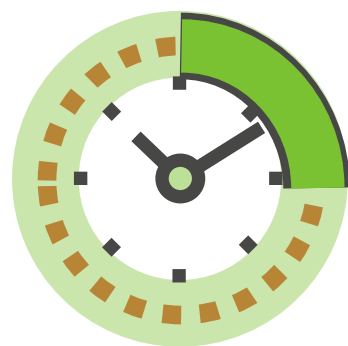
拯救世界的时间

非计划工作
24%

变更
20%

IT项目工作
19%

Let's Build on AWS & Reinvent the Architecture

不断创新发展的云服务
为企业用户带来后发优势

微服务复杂性功能需求下沉到平台

Amazon Kinesis

Amazon DynamoDB
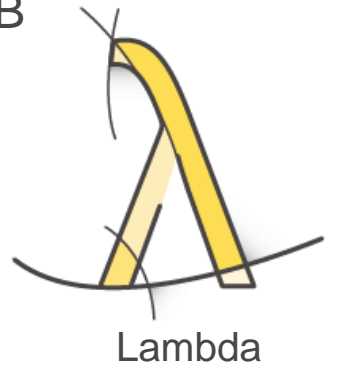
AWS App Mesh

API Gateway

Amazon SQS

AWS Cloud Map

Amazon SNS

SpringCloud
Netflix OSS

Cognito
User Pools

Lambda

AWS X-Ray

AWS Step Functions

Amazon CloudWatch

无服务器架构是企业微服务的未来

Amazon
Container Services

Amazon
ECS

Amazon
EKS

Amazon
Fargate

Amazon
ECR

从类库继续下层到平台服务

云原生微服务架构：
**你写的所有代码都是关于
业务逻辑的**

开源优先，微服务的容器化改造

aws

无服务器的微服务架构-FaaS+事件驱动

Web
Applications

Backends

Data
Processing

Chatbots

Amazon
Alexa

Autonomous
IT

Amazon API
Gateway

Amazon SQS

Amazon
DynamoDB

Amazon S3
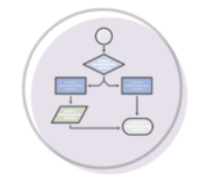
Amazon SNS
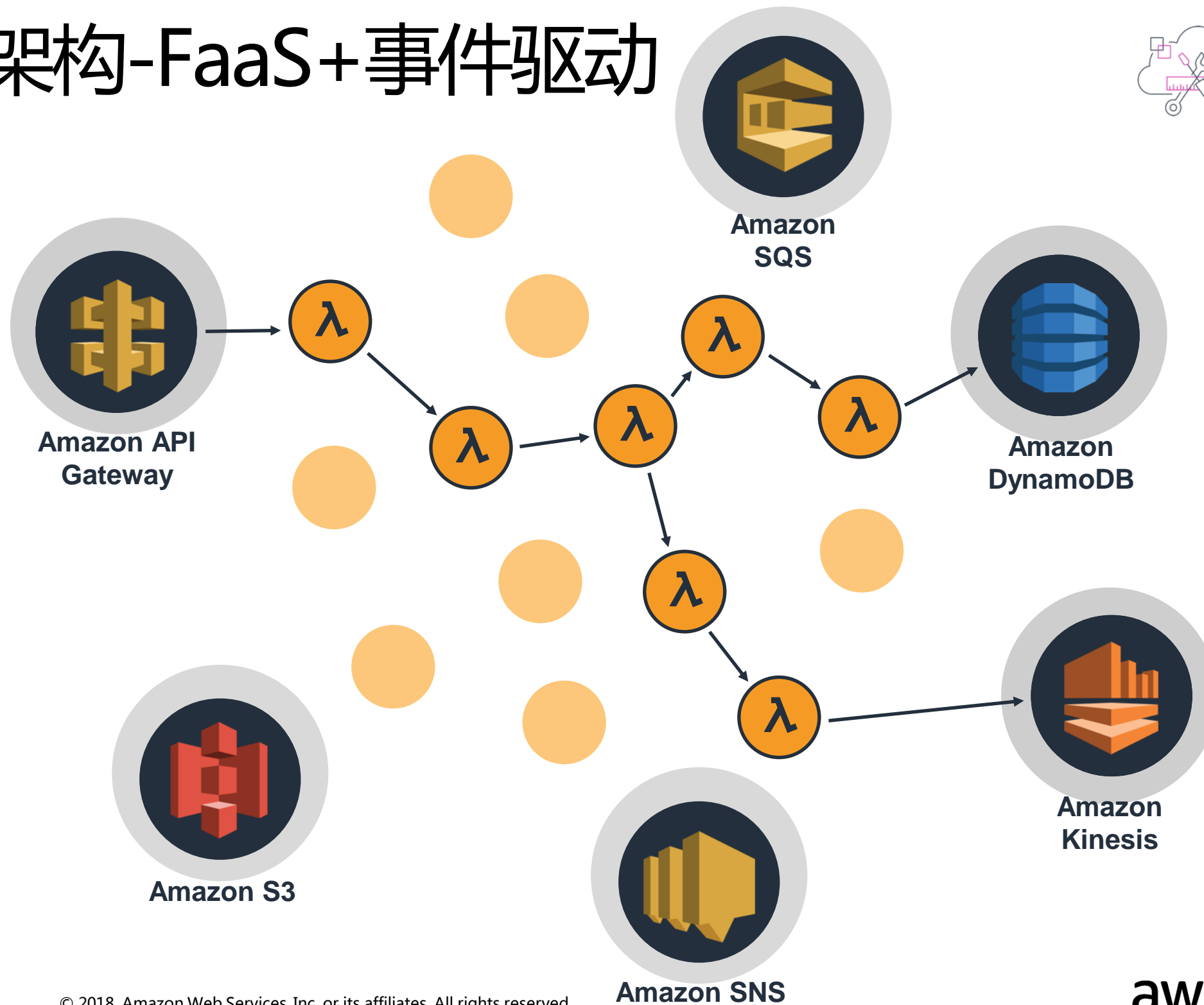
Amazon
Kinesis

Developing Mobile and Web Apps on AWS
简便快捷开发者体验
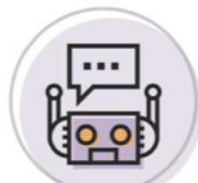
iOS Android Web React Native

Framework

CLI

Cloud Services Developer Tools

AWS re:Invent

aws
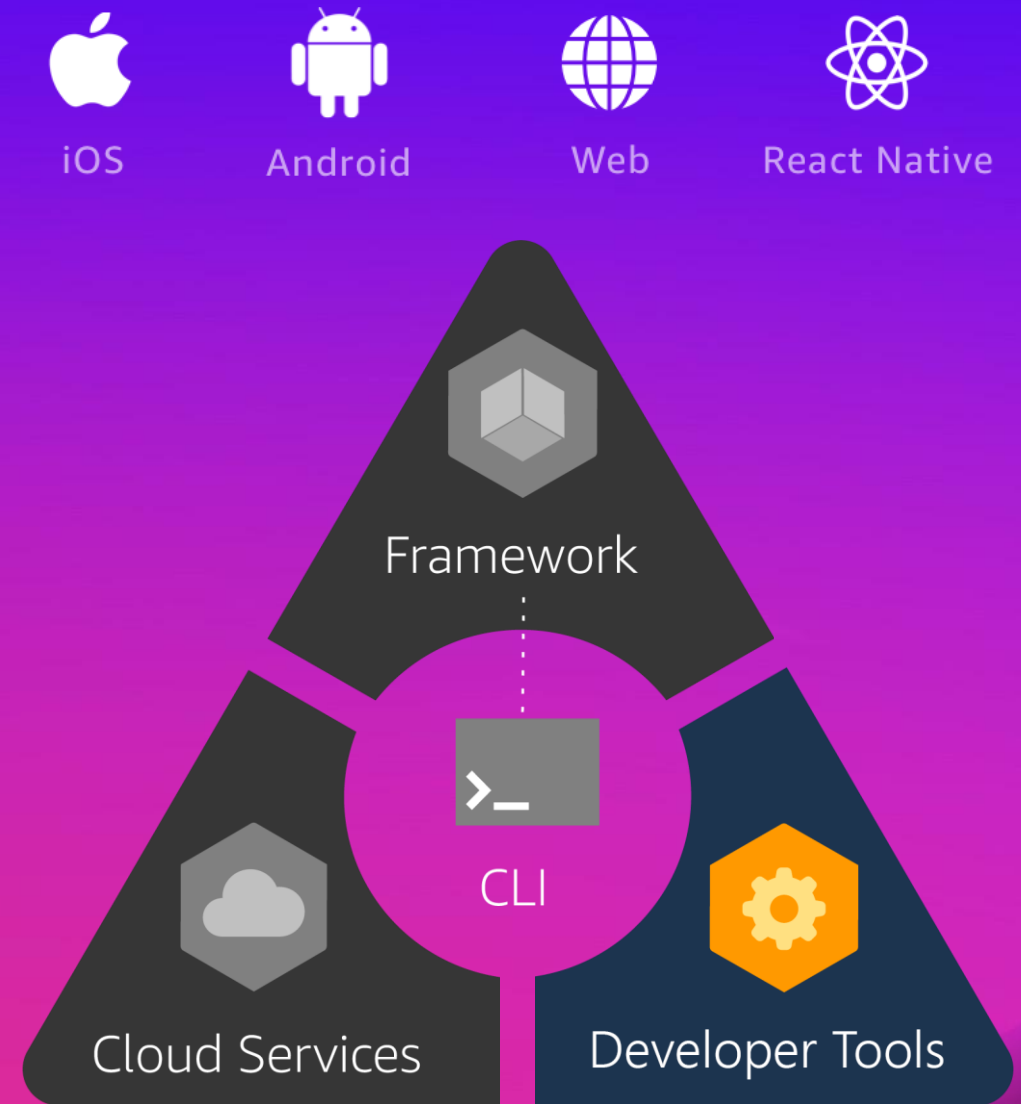
团队协作挑战

不同的微服务团队如何协作完成一个综合的业务需求？


```
{
    events(keyword:"shania", limit:10) {
        id
        name
        availability
        priceRanges {
            currency
            min
            max
        }
    }
}
```

aws
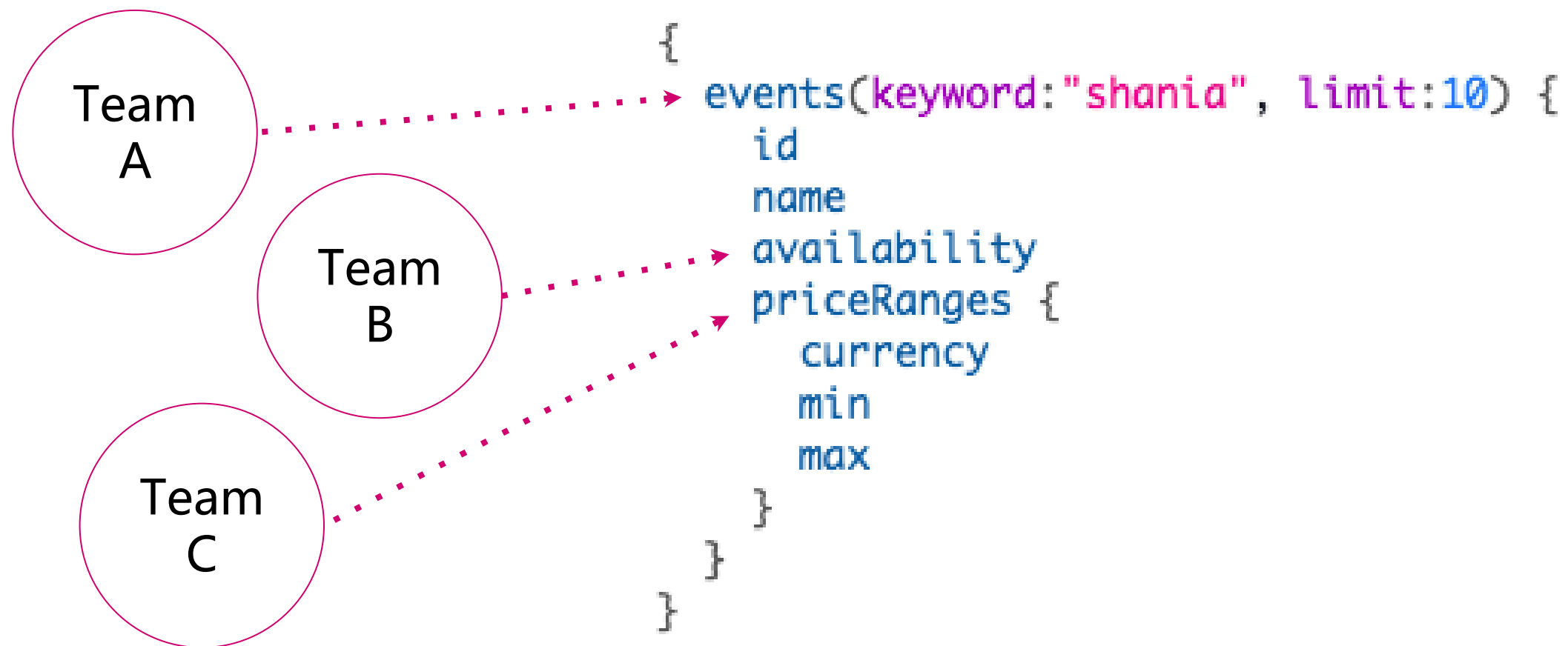
无服务器架构中的团队协作改进

GraphQL（AppSync）+无服务器架构可以非常棒的促进多开发团队协作

Monolithic API   vs.   AWS AppSync + serverless
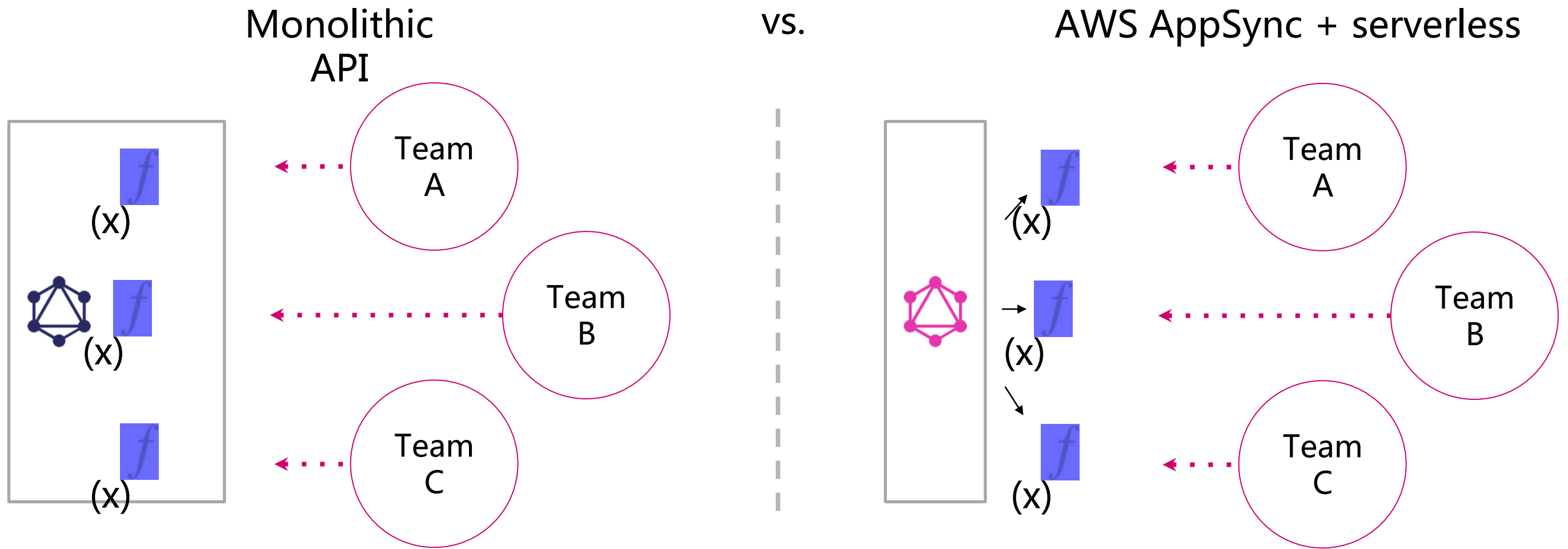
无服务器微服务架构 - 企业21 世纪现代应用架构

APPLICATION PRIMITIVES – COMPUTE AND DATASTORES

Amazon S3 | AWS Lambda | AWS Fargate | Amazon DynamoDB | Amazon Aurora Serverless | Amazon Kinesis

APPLICATION INTEGRATION
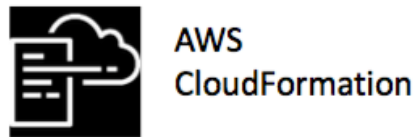
Amazon SNS | Amazon API Gateway | AWS Step Functions | Amazon MQ | Amazon SQS | AWS Appsync

DEVELOPER TOOLS
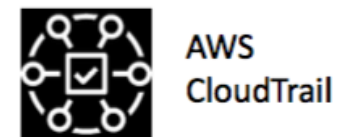
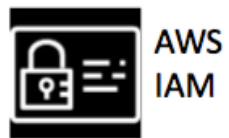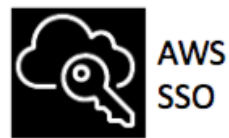AWS CloudFormation | AWS Cloud9 | AWS CodePipeline | AWS Config | AWS CloudTrail | Amazon CloudWatch | AWS X-Ray | AWS Serverless Application Repository
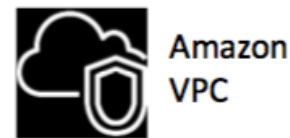
SECURITY AND ADMINISTRATION

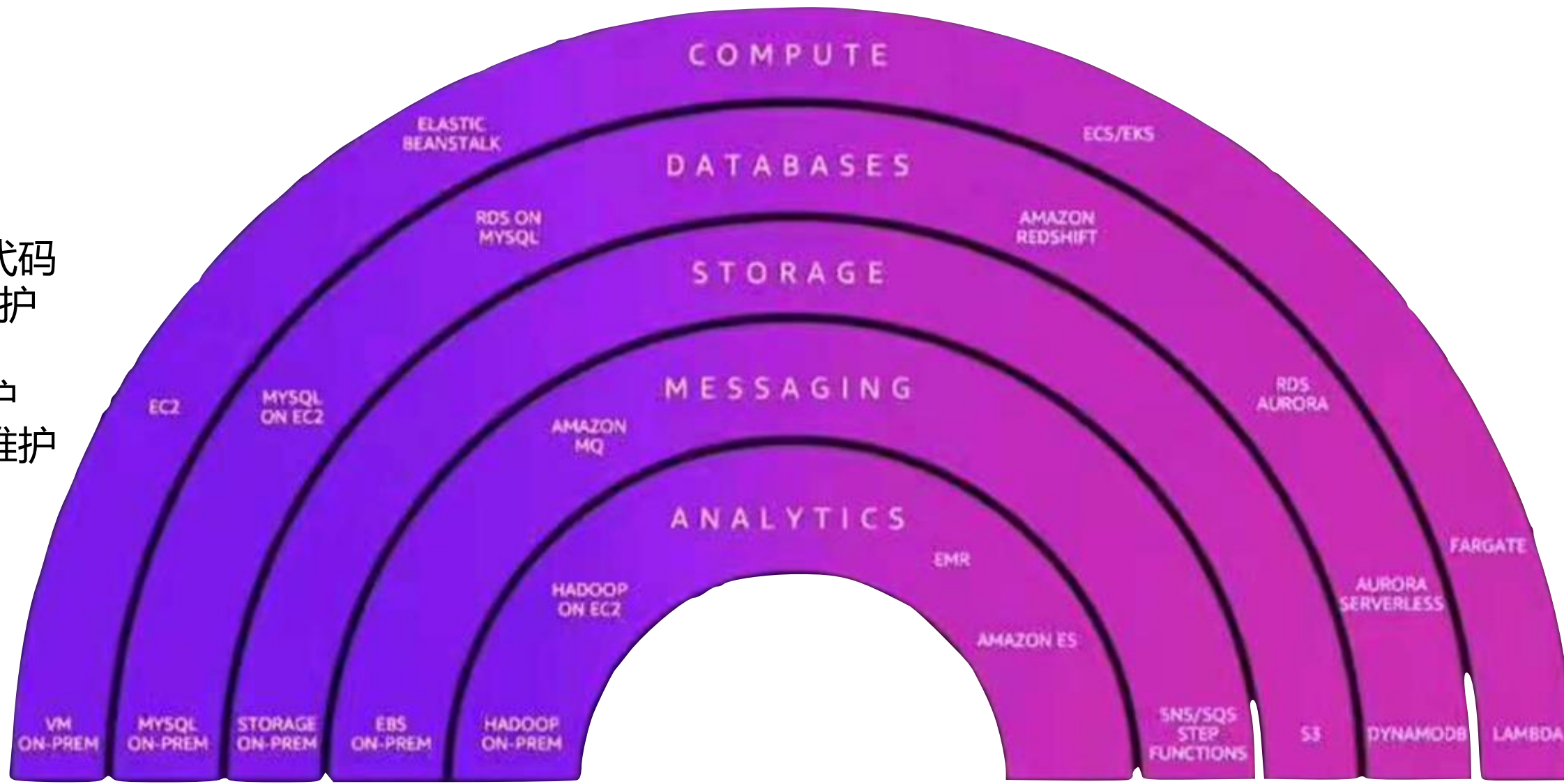AWS IAM | AWS SSO | Amazon GuardDuty | Amazon Inspector | Amazon VPC | AWS WAF | AWS Shield | Amazon Cognito

AWS re:Invent

aws

无服务器架构让你更聚焦业务

基础设施即代码
操作系统维护
网络
数据库维护
中间件安装维护
升级
部署
扩展

事件响应流程
混沌工程
和开发人员一样写代码
应用安全
成本优化

繁杂的底层运维工作

更有价值的Ops工作

总结

总结

- 当我们谈论微服务的时候，不仅仅是技术实践问题，要敢于问 "Why"
- 从客户出发，利用设计思维找创新点，数据思维来验证优化，技术思维进行落地
- 从简单架构开始，并对照优良架构方法不断迭代
- 从开源出发到云原生，无服务器微服务架构是企业架构未来

Thank you!

aws re:Invent

aws