

# 数字转型 架构演进

# SACC

## 2019 中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2019



2019年10月31-11月2日



北京海淀永泰福朋喜来登酒店



全新IT技术私域交流平台

# 瓜子Service Mesh的工具链和实践

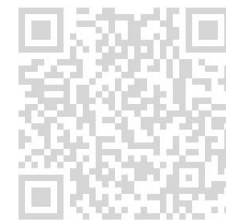
如何高效地构建微服务架构



全新IT技术私域交流平台

## Agenda

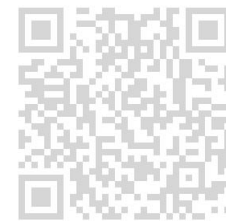
- Service Mesh 的发展历程
- RPC 协议选型
- 瓜子的gRPC + Mesh 平台介绍
- Kong 网关应用
- 开发和部署工具



全新IT技术私域交流平台

# 背景：单体架构向微服务架构转变

- 单体程序的构建，发布和变更成本不断增加
- 单体服务不适应小团队开发模式
- 业务的迅速变化和迭代

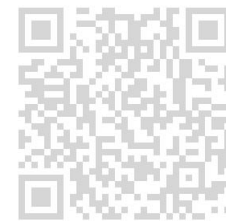


全新IT技术私域交流平台



# 微服务引入更多问题

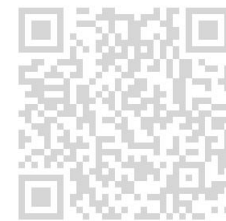
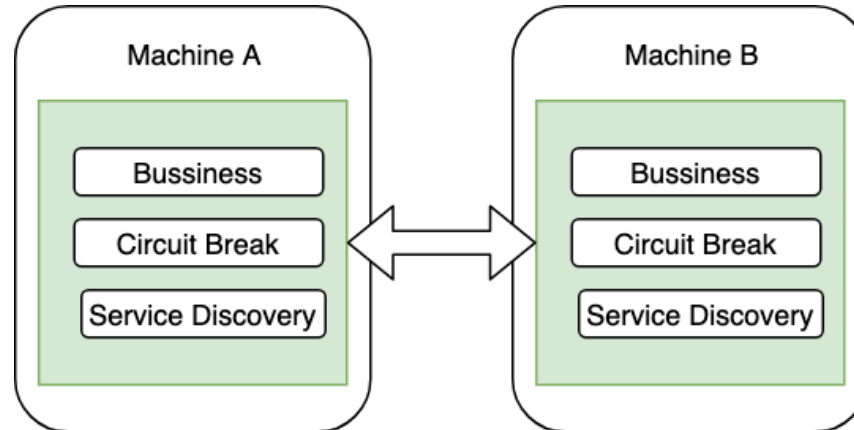
- 服务注册发现：自动化地址管理和平滑上线
- 接口规范：接口文档集中化，API规范统一
- 熔断限流：单个服务的故障影响降到最低
- 负载均衡：提升资源使用率，水平扩展能力
- 动态路由：单个服务灰度发布，A/B Test，多分支测试
- 测试环境治理：测试环境的增加让容器资源不可控



全新IT技术私域交流平台

# Service Mesh 的发展历程-手工时代

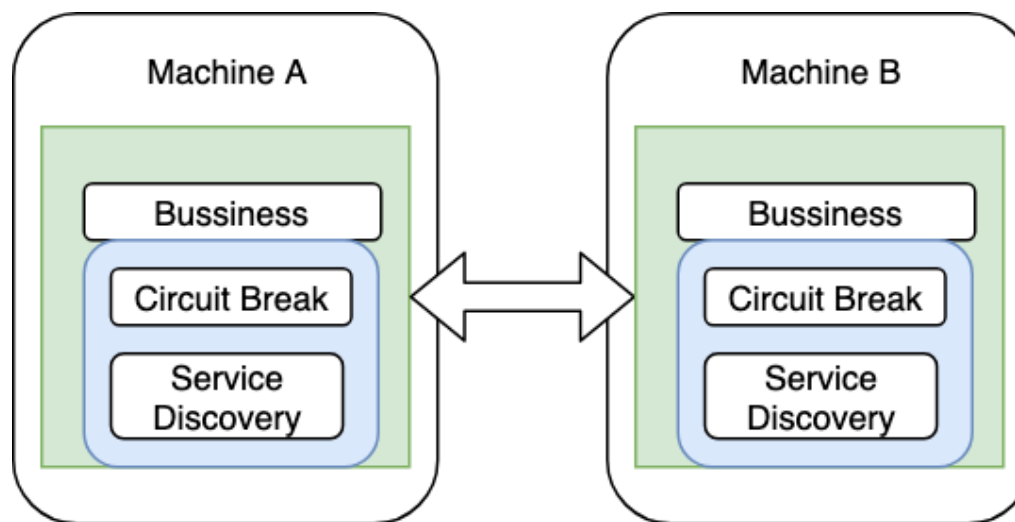
- 业务研发自研微服务语义
- 和业务代码紧耦合，难以维护



全新IT技术私域交流平台

# SDK时代

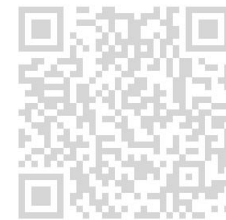
以 Spring Cloud, Dubbo 为代表的特定语言框架



全新IT技术私域交流平台

# SDK模式的问题

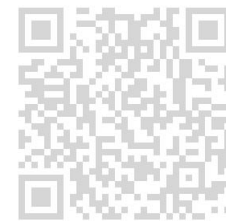
- SDK的接入和掌握的成本较高
- 框架往往依赖大量的基础库，和项目本身的依赖冲突经常发生
- 框架只支持少数特定语言，和语言特性强绑定，多语言支持困难
- 框架被打包到项目内部，升级框架极难推动，导致多版本同时维护的窘境



全新IT技术私域交流平台



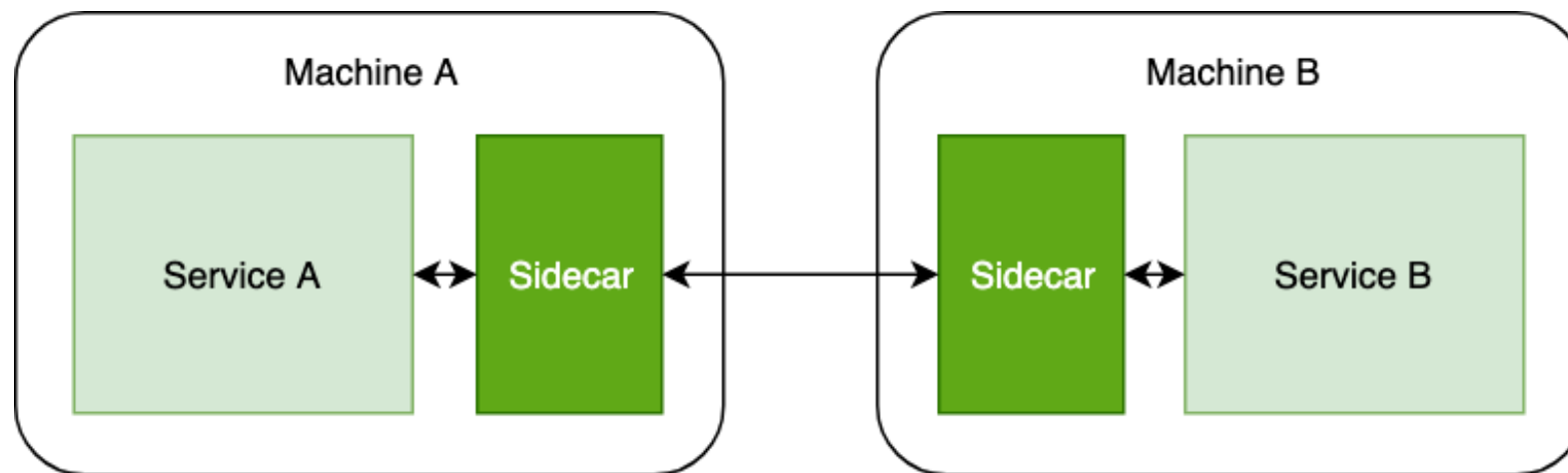
# 微服务治理=管理服务间通讯



全新IT技术私域交流平台

# Service Mesh 登场

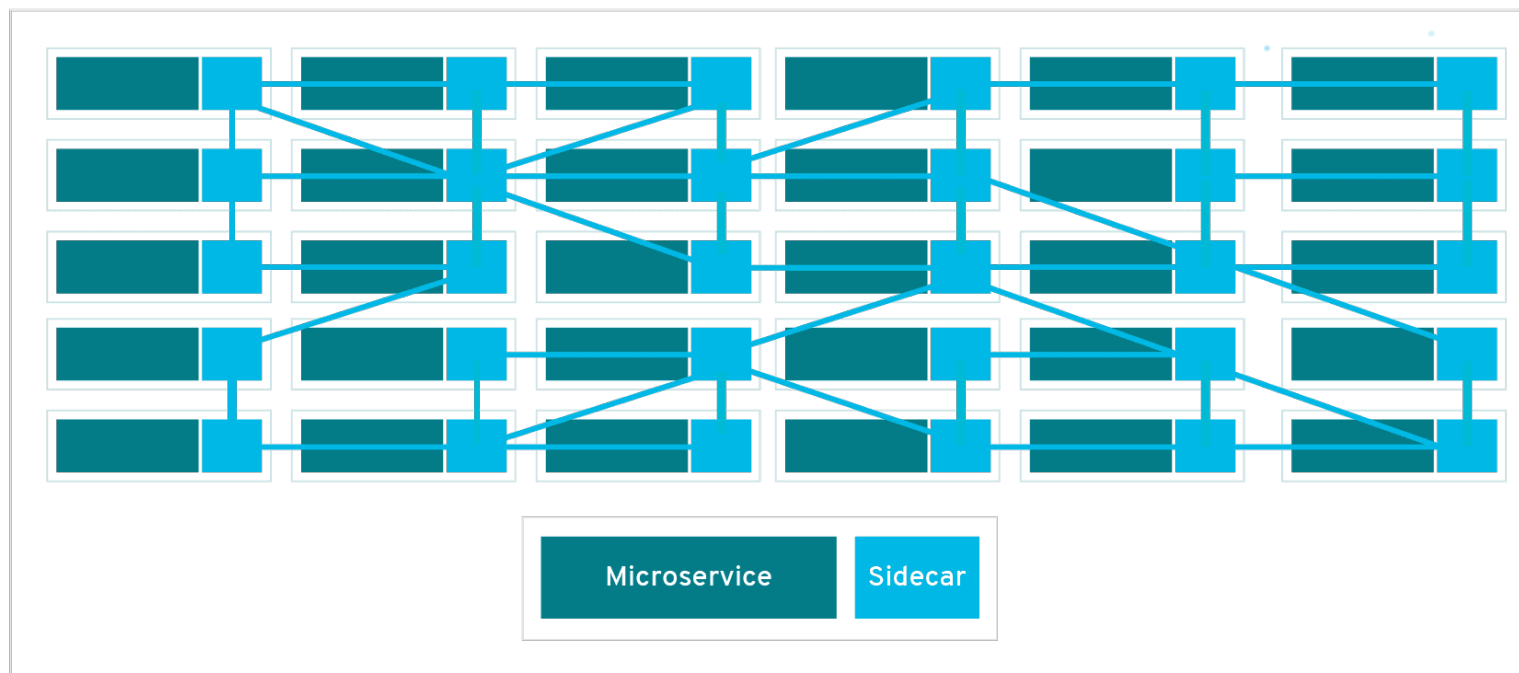
- 拦截流量：解析协议，转发流量
- 管理流量：注册发现，路由策略



全新IT技术私域交流平台

# 全局视角

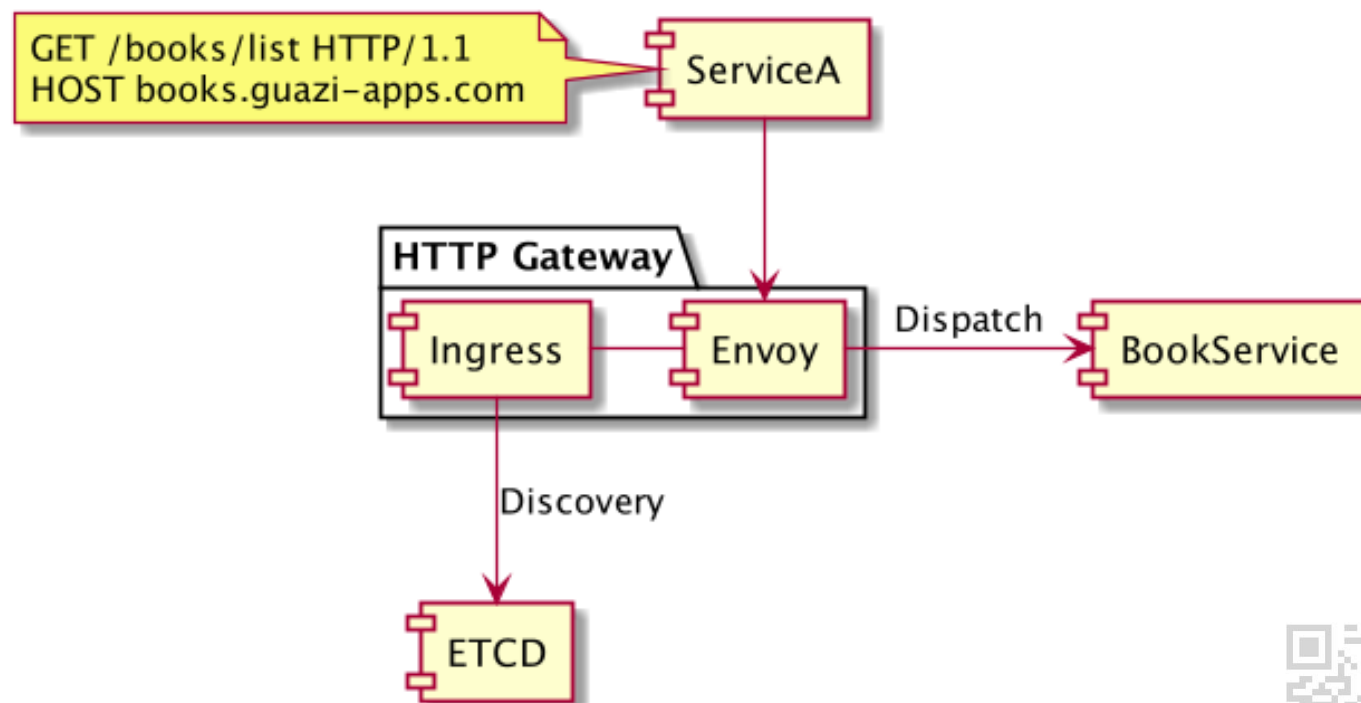
- 从全局看，Sidecar代理了微服务之间的所有流量，接管了中间件能力，让服务只需关注业务



全新IT技术私域交流平台

# Mesh vs API Gateway

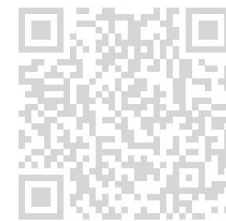
- API Gateway 主要用于外网流量，也可用于内网微服务网关
- 增加两跳网络通信，延迟较大
- 网关流量压力大，故障影响面大
- 只支持HTTP API



全新IT技术私域交流平台

# 控制面是必须的么

- Mesh的应用需要循序渐进，控制面接入有门槛
- 部分控制面功能可以嵌入到Sidecar，比如路由规则，配置管理
- 配置随部署分发能够满足绝大部分场景
- Prometheus已经成为内部的标准
- 安全，AB等策略控制利用成熟系统



全新IT技术私域交流平台



# SDK和Sidecar的比较

|         | 多语言支持 | 接入成本 | 学习门槛 | 性能        | 升级成本  |
|---------|-------|------|------|-----------|-------|
| SDK     | 成本高   | 高    | 高    | 高         | 高     |
| Sidecar | 成本低   | 低    | 低    | 增加 1ms 延时 | 无(透明) |



全新IT技术私域交流平台

# 通讯协议选择：RPC vs HTTP API

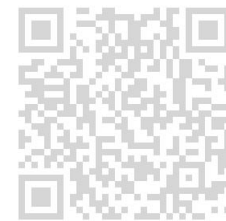
- 内网服务间调用适用RPC，外网客户端调用适用HTTP API
- 服务开发者需要开发和维护两套技术栈

|          | 生态丰富 | IDL | 代码生成 | 性能 | 类型安全 | 跨语言  | 开发成本 |
|----------|------|-----|------|----|------|------|------|
| HTTP API |      | X   | X    | 低  | X    | 支持   | 高    |
| RPC      | X    |     |      | 高  |      | 部分支持 | 低    |



全新IT技术私域交流平台

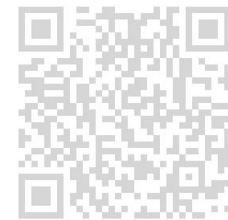
# 鱼肉熊掌兼得：RPC + HTTP



全新IT技术私域交流平台

# RPC 协议栈

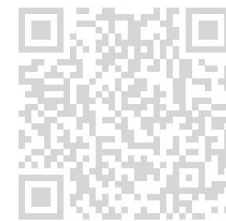
- **传输层**定义如何传输数据，例如头格式，长度，是否压缩，多路复用等，常见协议包括：HTTP, HTTP2, Dubbo, Thrift
- **协议层**定义消息体如何被编解码，分为两类：
  - 文本：JSON，XML
  - 二进制：Hession，Protobuf，Thrift
- **应用层**是开发者的界面，包括IDL，Stub，SDK，例如：
  - Protobuf
  - Thrift
  - Swagger



全新IT技术私域交流平台

# RPC 选型原则

- 可以转换成HTTP协议和Json编码
  - HTTP+Json 生态丰富
  - 通过 API 网关的安全和流控，RPC可以直接为客户端提供服务
- 支持跨语言
  - 没有哪种语言适应所有领域
  - 未来技术架构的不确定性
- 支持IDL契约
  - 使接口定义与语言平台无关，可以生成代码和文档
  - 实现协议前后兼容



全新IT技术私域交流平台



# 主流RPC比较

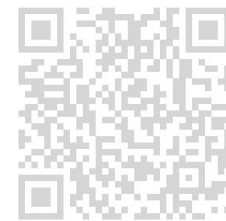
|        | 跨语言 | IDL            | 社区支持/活跃 | 序列化      | 传输协议  | JSON序列化 |
|--------|-----|----------------|---------|----------|-------|---------|
| Dubbo  | X   | Java Interface | 4       | Hession  | 私有    | X       |
| gRPC   |     | Protobuf       | 5       | Protobuf | HTTP2 |         |
| Thrift |     | Thrift         | 3       | Thrift   | 私有    |         |



全新IT技术私域交流平台

# 为什么选择gRPC

- 可以转换成 HTTP+JSON API
- HTTP2 被网关广泛支持，包括 Nginx，Envoy 等
- HTTP2 的基础库丰富，便于自研 Sidecar
- Protobuf支持所有主流编程语言，跨语言容易
- 支持双向Stream和异步编程，适用于高并发高吞吐



全新IT技术私域交流平台

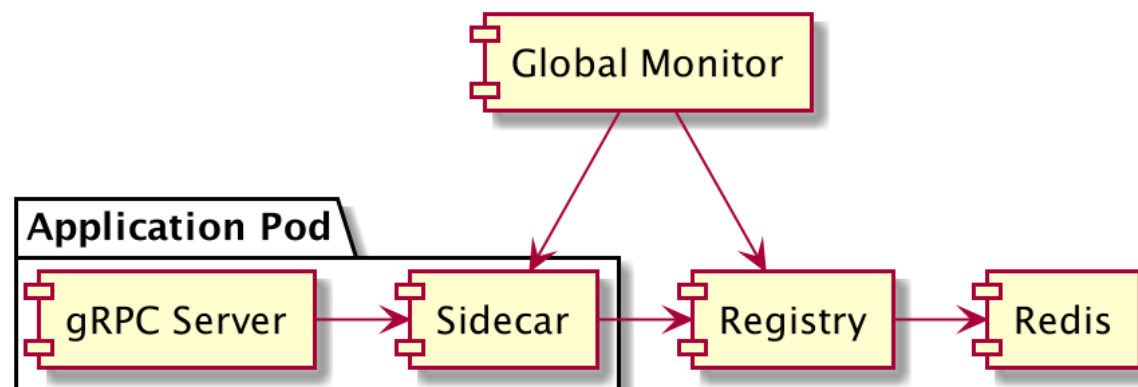
# 瓜子的gRPC + Mesh 平台



全新IT技术私域交流平台

# 服务注册设计

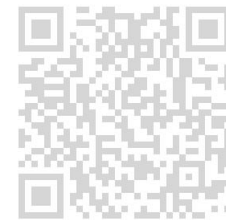
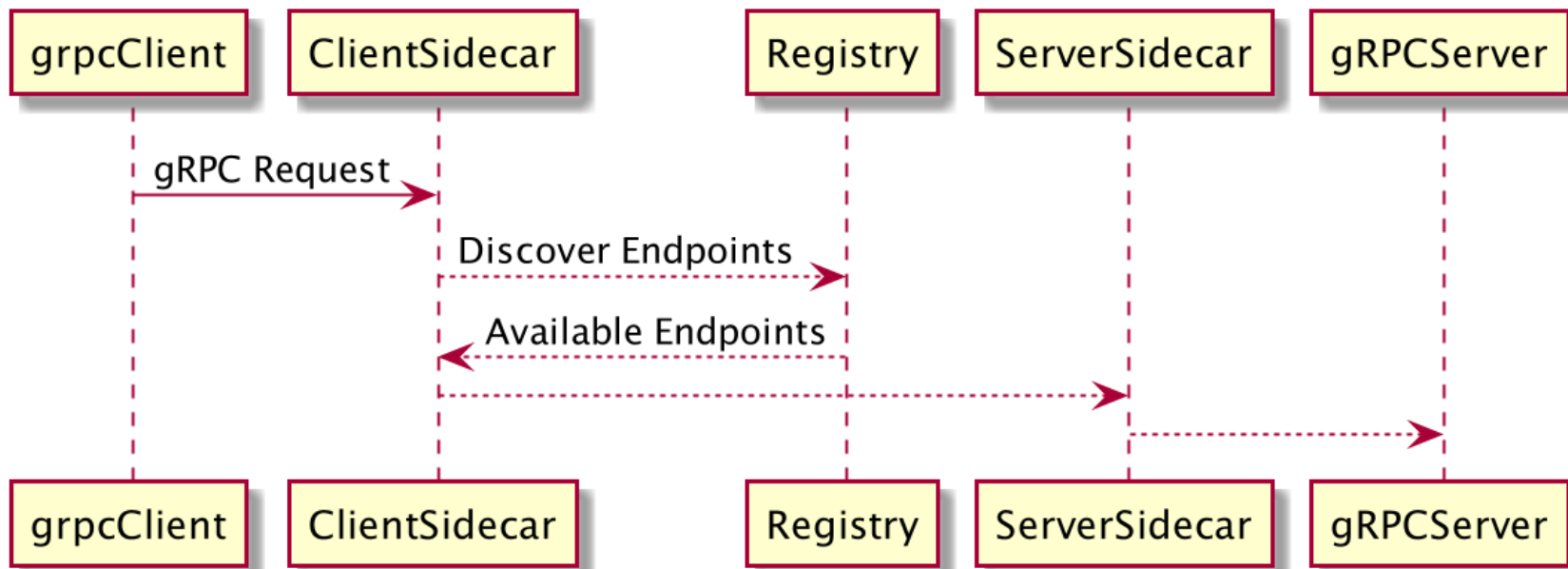
- Key : `package.Service/Method@Branch`
- Registry是无状态分布式服务，使用Redis存储
- Sidecar定时主动拉取 Service Metadata 和 Health Check，发送到Registry
- 服务关闭或者不健康，Sidecar会从Registry摘除注册
- 全局的 gRPC Monitor 服务会定时检查所有gRPC接口的可用性



全新IT技术私域交流平台

# 服务发现

- gRPC Client 只需 Hardcode 127.0.0.1:<Sidecar Port>
- Sidecar 获取Registry中对应的服务地址，然后使用负载均衡策略把gRPC请求转发到目标gRPC server上

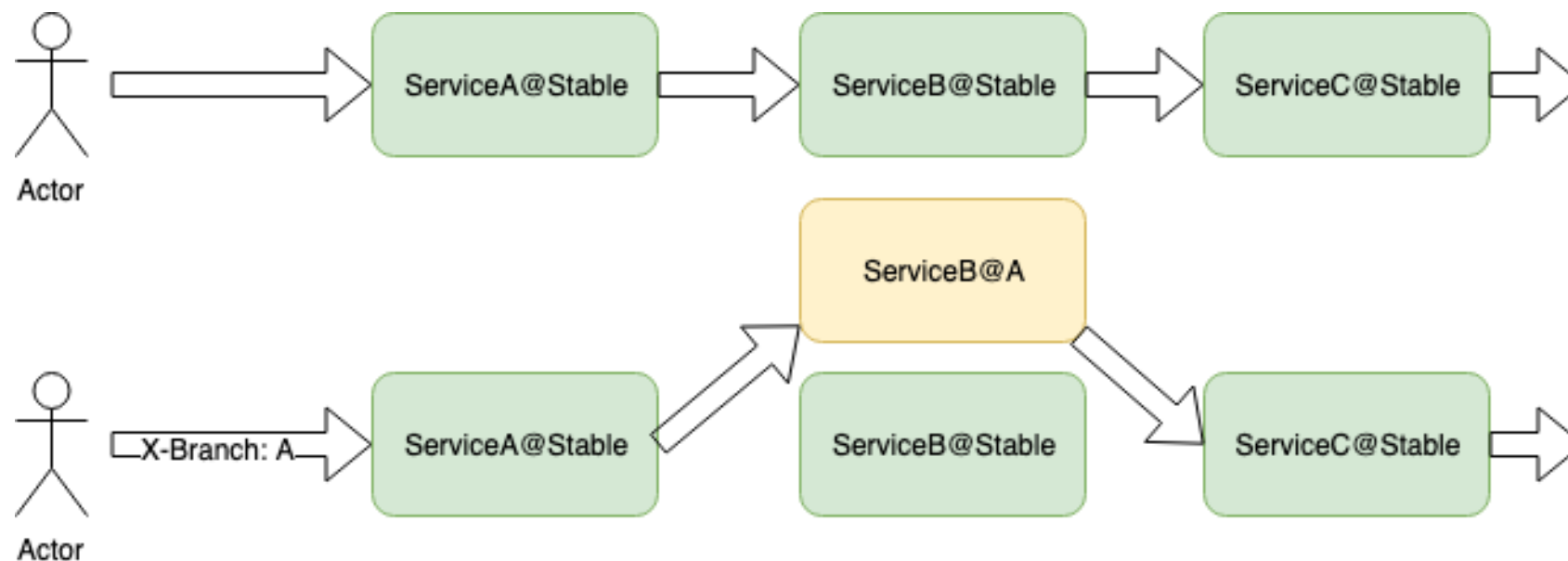


全新IT技术私域交流平台



# 根据分支服务发现—高级路由

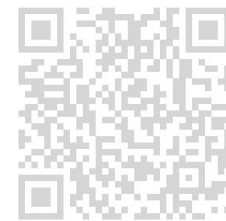
- 对某个服务的分支测试，只需部署必要的分支版本



全新IT技术私域交流平台

# Kong网关的应用

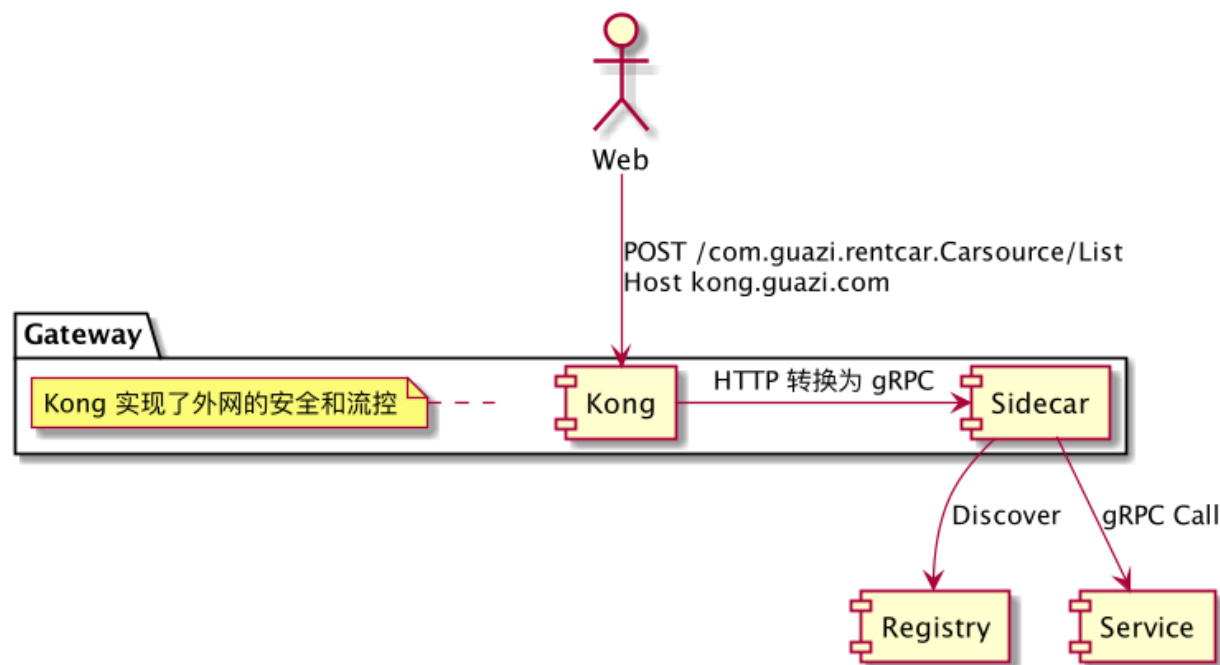
统一的HTTP API中间件



全新IT技术私域交流平台

# gRPC 服务于HTTP客户端

- gRPC经过Kong代理鉴权，安全和流控，转变为可靠的Web服务

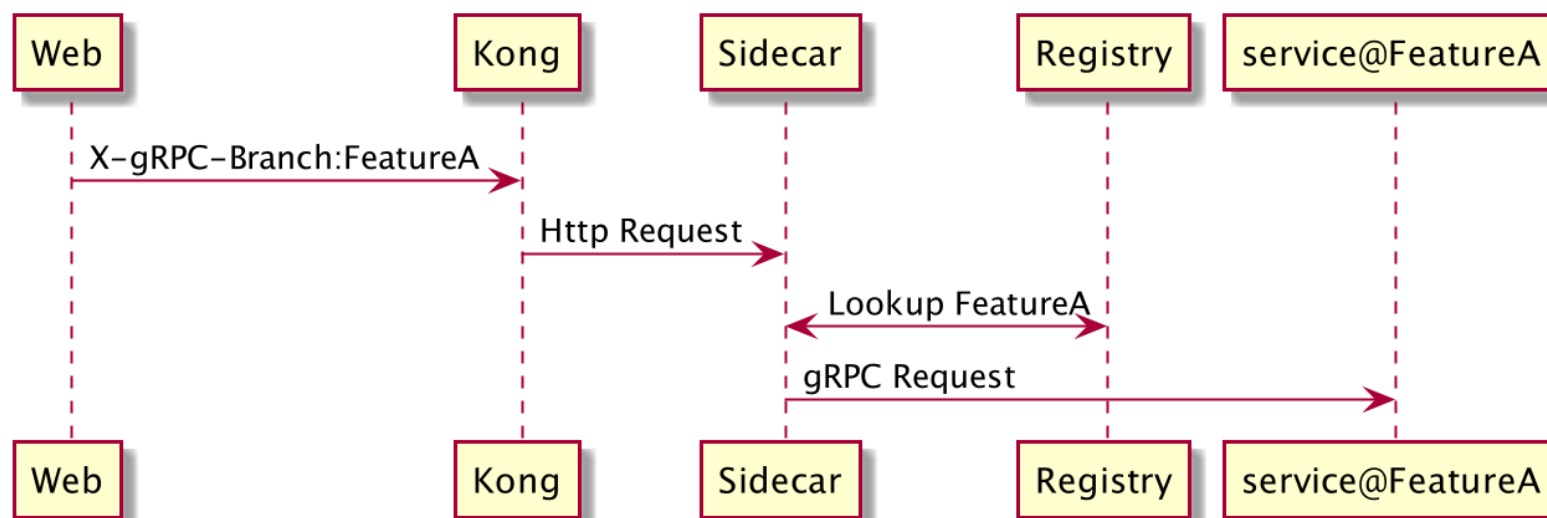


全新IT技术私域交流平台

# Kong上高级路由

场景：

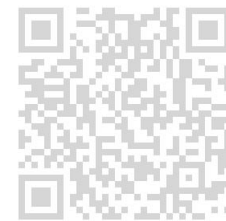
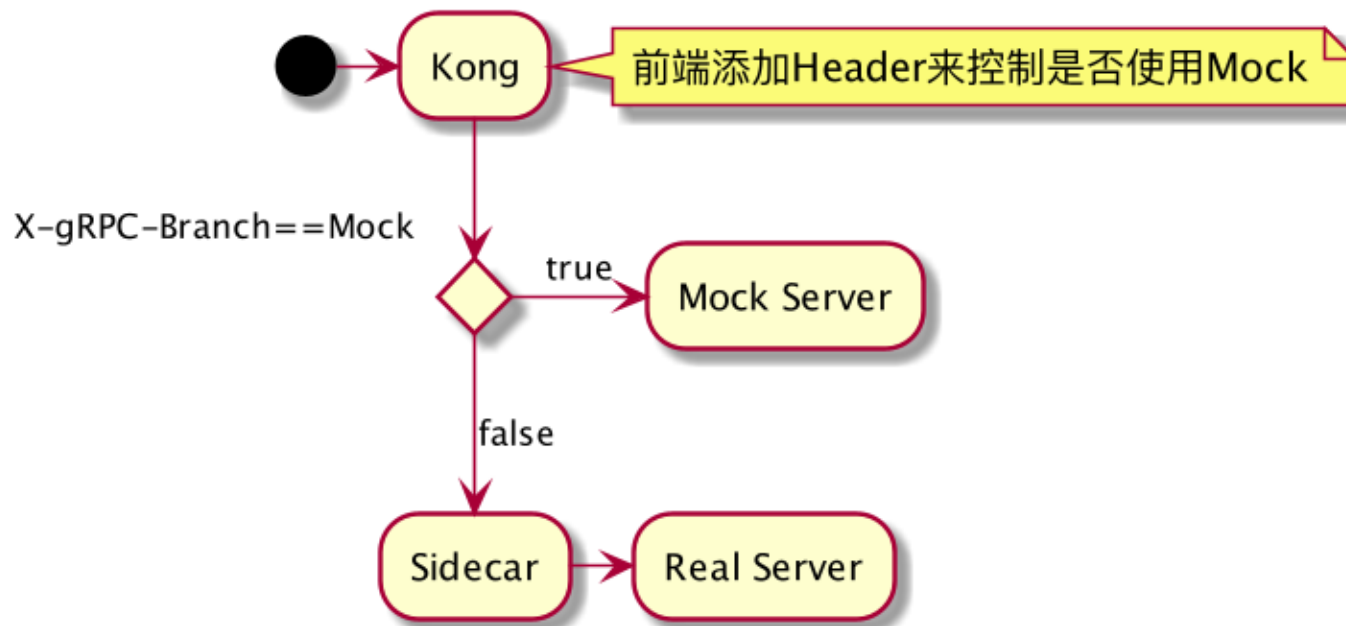
- 多Feature分支并行部署和前后端联调
- 本地机器前后端联调，无需修改代码和配置



全新IT技术私域交流平台

# 如何做 HTTP API Mock

- 在后端还没有开发完的阶段，前端添加Header来把请求分发到 Mock Server

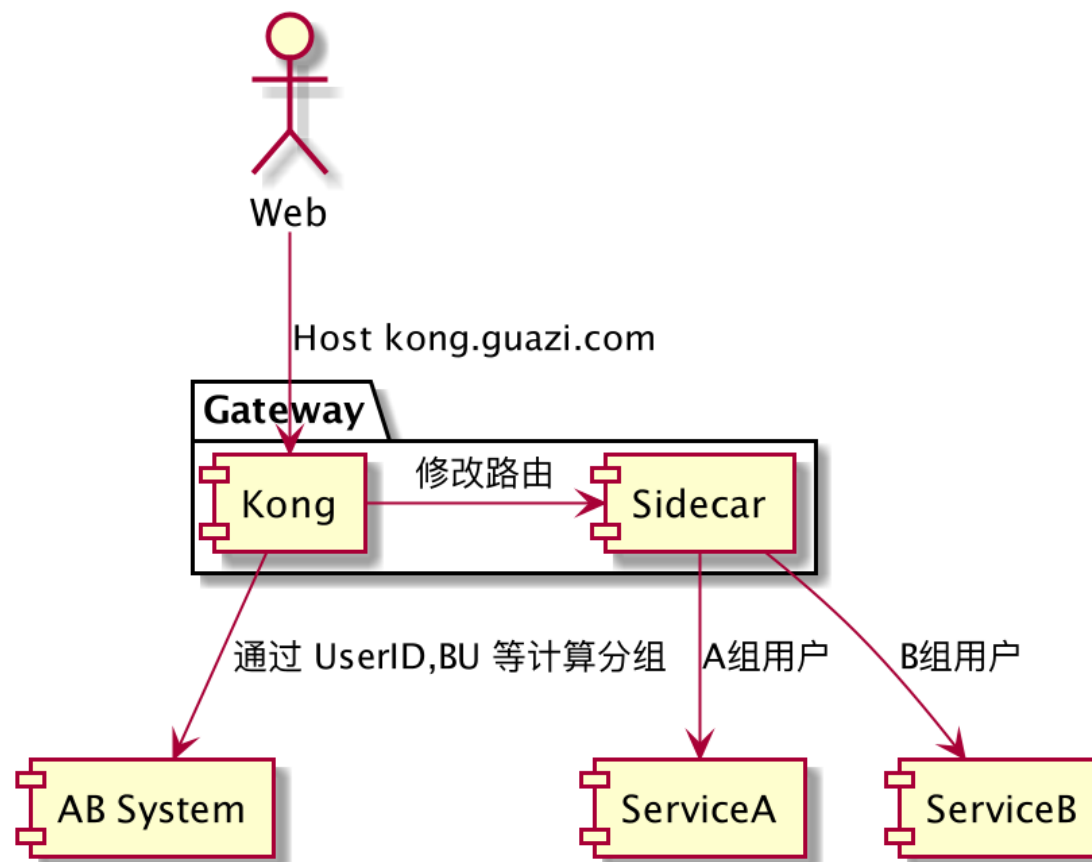


全新IT技术私域交流平台



# Kong + A/B 做分流

- 通过A/B规则引擎，基于业务分流
- 策略和系统分离



全新IT技术私域交流平台

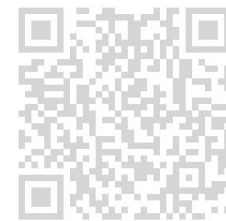
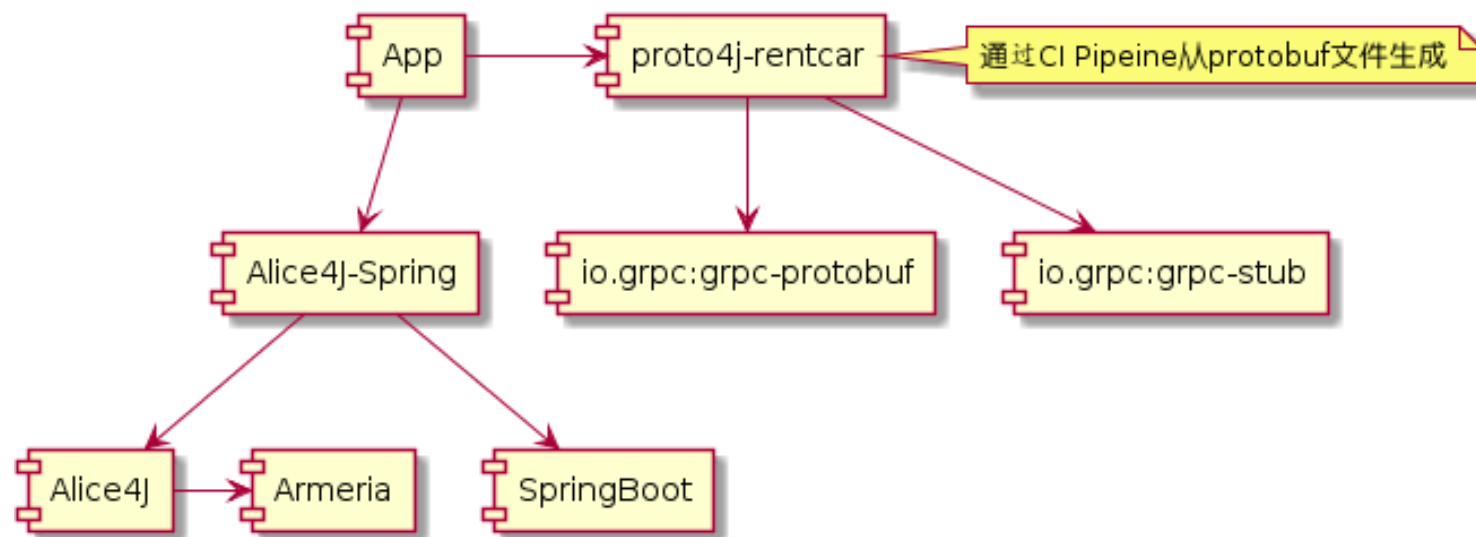
# 开发和部署工具



全新IT技术私域交流平台

# Java gRPC 框架

- 基于 Line 的 gRPC 框架 Armeria
- 单端口双协议：HTTP和 gRPC(HTTP2)
- 集成 SpringBoot ，自动注入 Prometheus ， Sentry等必要组件 ，零配置
- gRPC Client 自动传播调用链信息等上下文

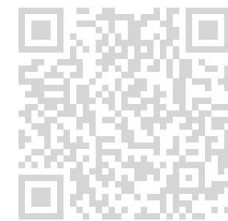


全新IT技术私域交流平台

# SpringBoot Server Example

```
@ComponentScan("com.guazi.tools.alice4j.spring")
// 扫描所有 gRPC Service
@ComponentScan("com.guazi.tools.alice4j.sample.service")
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        ApplicationContext ctx = SpringApplication.run(DemoApplication.class, args);
    }
}

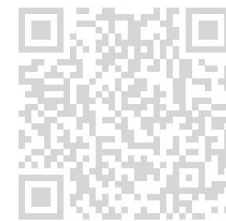
@Named
@Singleton
public final class ExperienceService extends ExperienceGrpc.ExperienceImplBase {
    @Override
    public void sayHello(HelloRequest request, io.grpc.stub.StreamObserver<HelloReply> responseObserver) {
        ServiceRequestContext ctx = RequestContext.current();
        try {
            responseObserver.onNext(doSayHello(request));
            responseObserver.onCompleted();
        } catch (CustomException e) {
            responseObserver.onError(e);
        }
    }
}
```



全新IT技术私域交流平台

# 本地开发联调流程

- 开发者首先提交 Proto，经 CI 生成 Stub 和 Swagger API
- 前端通过 Swagger 地址开发调试，Kong 返回 Mock Response
- 后端开发完成后，通过本地 Sidecar 把服务注册为 package.Service/Method@laptop
- 前端代码注入 Header：X-gRPC-Branch: laptop，请求会被转发到后端的本地开发机



全新IT技术私域交流平台

# Client Example

- 同时支持gRPC 和 HTTP 调用

```
helloService = GrpcxClient.newBuilder("127.0.0.1:8888").build(ExperienceBlockingStub.class);
HelloRequest request = HelloRequest.newBuilder().setName("Avril Lavigne").setGender(Gender.FEMALE).setUserId(800).build();
HelloReply rsp = GrpcxClient.newContext().
    importContext(ServiceRequestContext.current()).
    addHeader("x-request-id", reqID).
    addHeader("x-trace-id", reqID).
    exec(() -> helloService.sayHello(request));
```

```
curl -X POST http://127.0.0.1/api/Experience/SayHello -d '{
  "user_id":1000,
  "name":"Bruce Lee",
  "gender":0
}'
```



全新IT技术私域交流平台




# 瓜子云平台部署

- 构建阶段添加Mesh Image

应用镜像构建 (Build) | + Add

#1 grpc-mesh | 

#2 grpc-server | 

- 部署Mesh为Sidecar

模块1 ×

模块2 ×

\* 模块名称 ?

grpc-mesh

\* CPU核数

1

\* 内存 (G)

0.5



全新IT技术私域交流平台

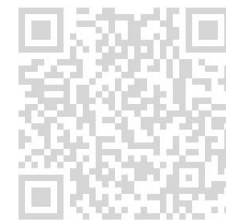
# 统一生成监控大盘



全新IT技术私域交流平台

# 总结

- Service Mesh 技术是微服务架构推动下逐步演变而来，把中间件从程序中剥离出来
- 选择gRPC因为其支持HTTP+JSON，跨语言，以及HTTP2
- 瓜子的Mesh体系实现了服务治理，高级路由，监控等能力
- Java的SDK通过整合Spring，Armeria等基础库实现了快速上手，零配置的开发框架。



全新IT技术私域交流平台

# Q&A



全新IT技术私域交流平台



# *Thanks*

