# 微服务编排与FaaS探索

吕红利

巍信: kingkongslove

2017年4月

# 工匠精神

不能讨好自己，就毁掉作品

为了完美作品，可以毁掉自己

# 主要内容

- 服务治理简史
- 微服务架构要素
- App + BaaS之困惑
- PaaS + 服务编排
- 弹性服务编排 (FaaS/Serverless)

# 服务治理简史

- EAI & DI：　　先秦（Hub） - 共主，进贡
- 　　|
- SOA：　　秦汉（ESB） - 郡县制，强制统一度量衡
- 　　|
- Microservices：邦联（Registry） - 欧盟/英联邦，普世，去中心

# EAI

- Sys1 —— Adapter1 —— **Hub** —— Adapter2 —— Sys2

# DI

- Sys1 DB —— **ETL** —— Sys2 DB

# SOA

SOAP SOAP

Sys1 —— **ESB** —— Sys2

# Microservices

**Your Protocol**

Sys1 —— Sys2

# 关于“第三方”的思考

- 何进　—> 董卓
- 汉献帝 —> 曹操

- 你　—> 两个老婆
- 公司　—> 技术资产

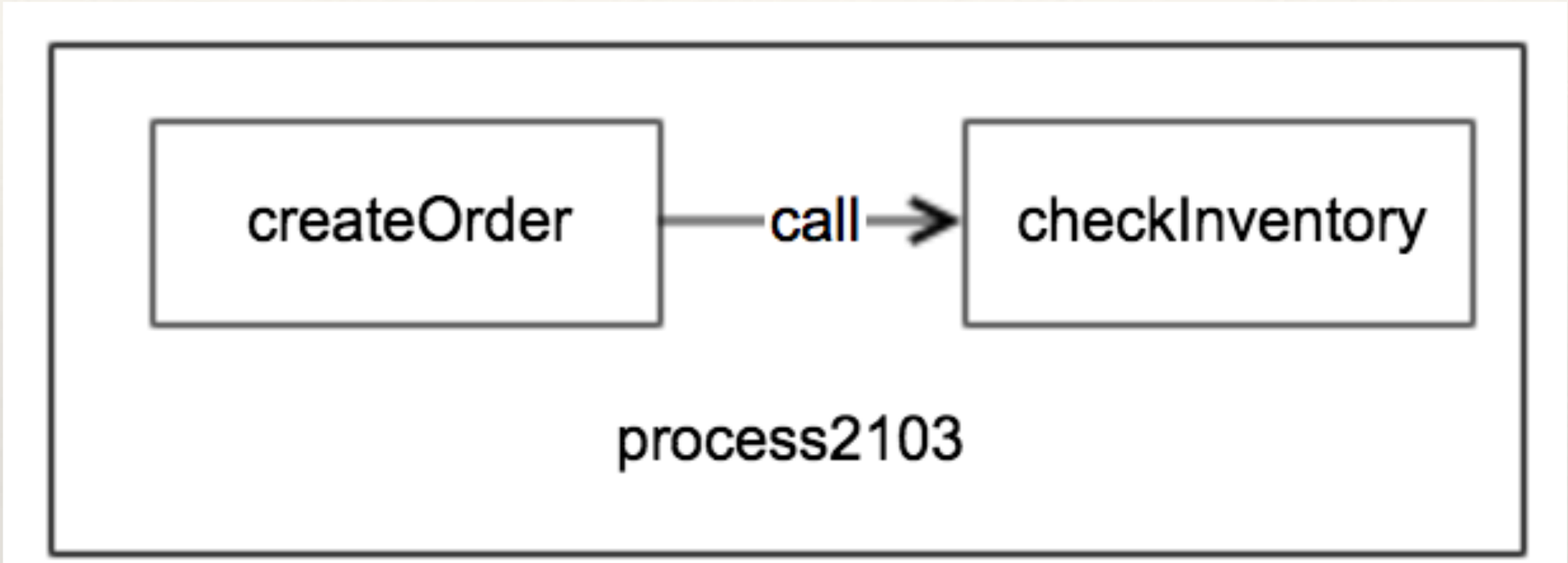
# 微服务的引入

- 远程调用 —> 网络介质 + 信息传递 + 目的地
- 网络介质 —> 传输协议
- 信息传递 —> 格式协议
- 目的地 —> 路由表
- 高可用 —> 负载均衡 + 失效转移
- 负载均衡 —> 分发策略
- 失效转移 —> 健康检查

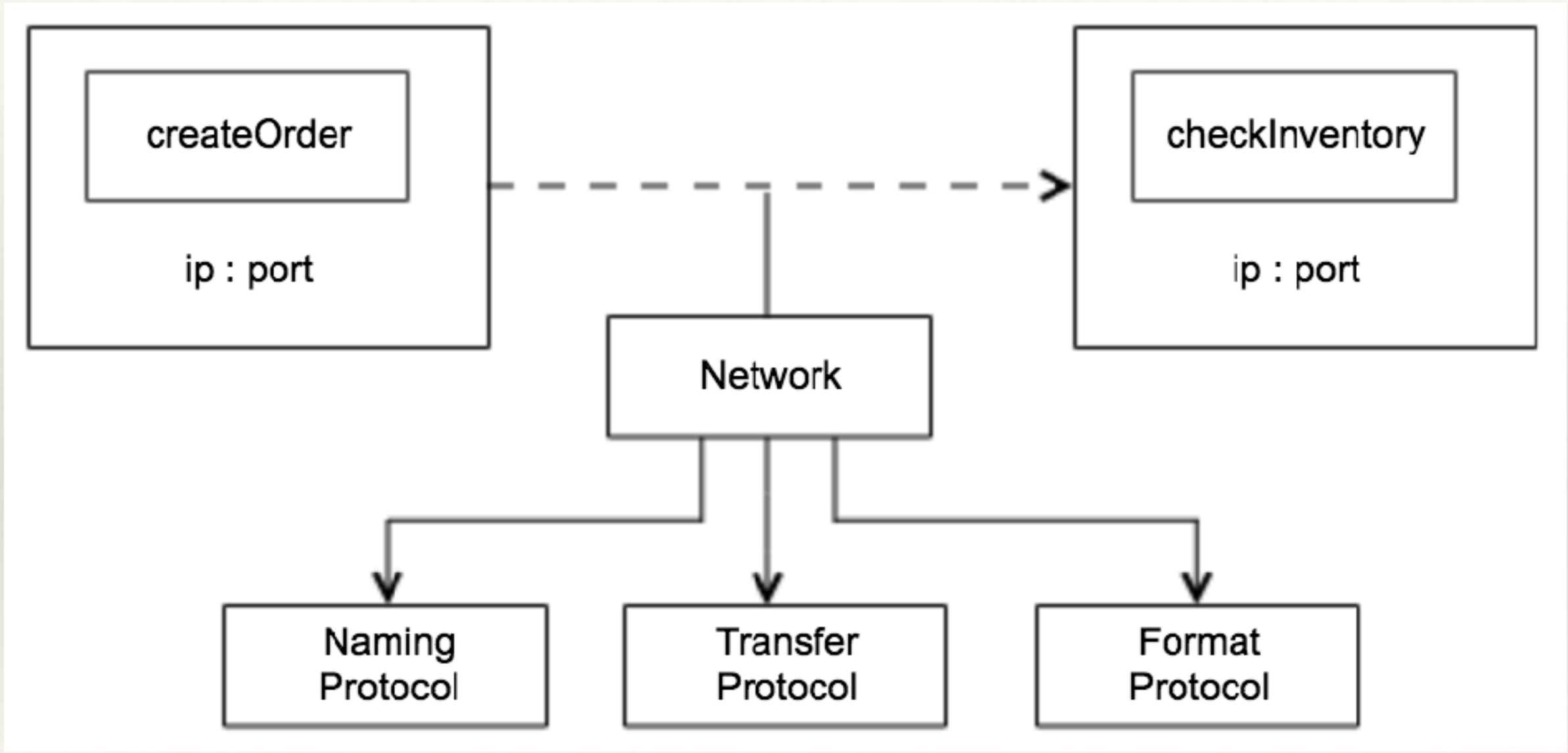
# 微服务架构要素

- 服务注册与发现
- 负载均衡与失效转移
- 服务协议：传输控制与消息格式
- 服务编排

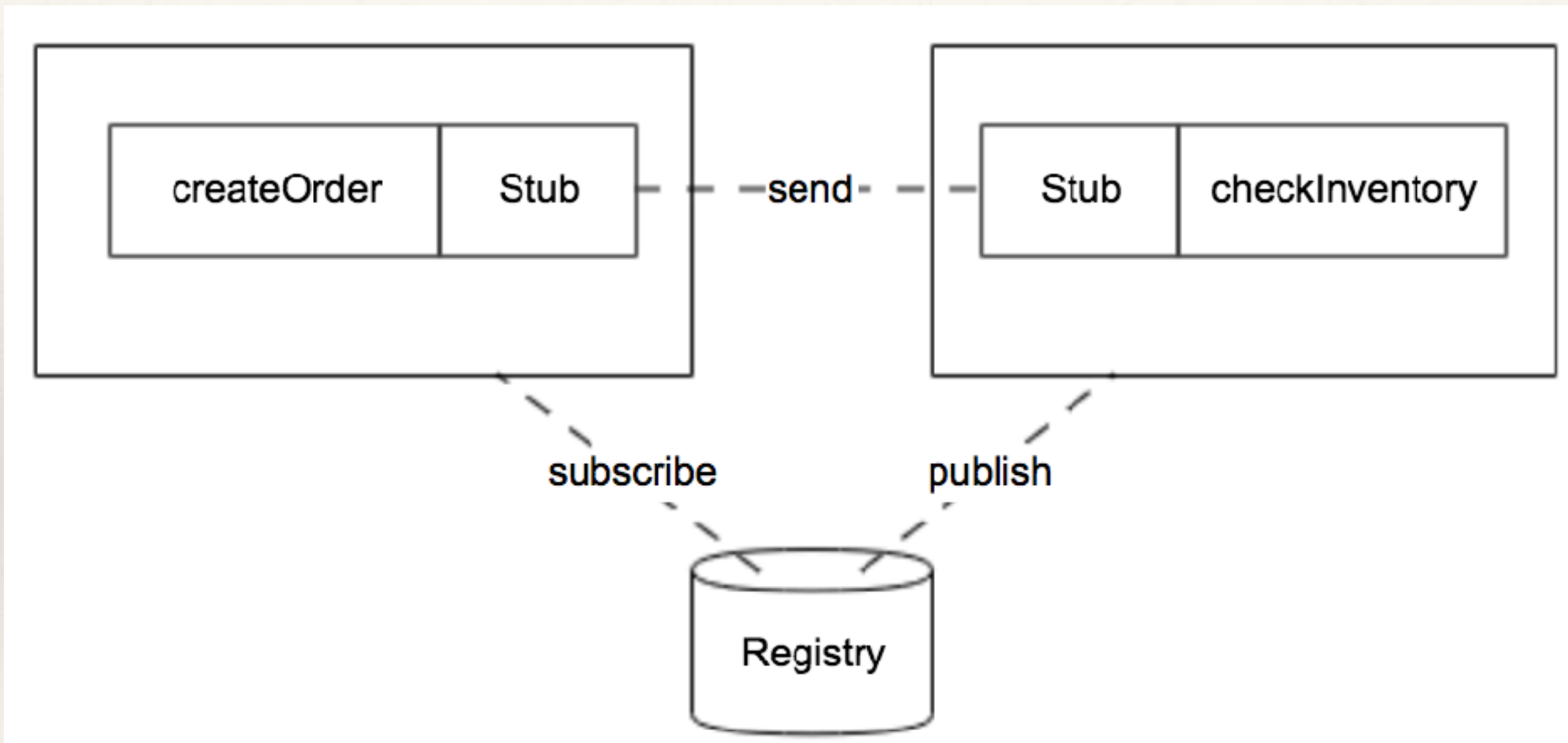
# Monolith



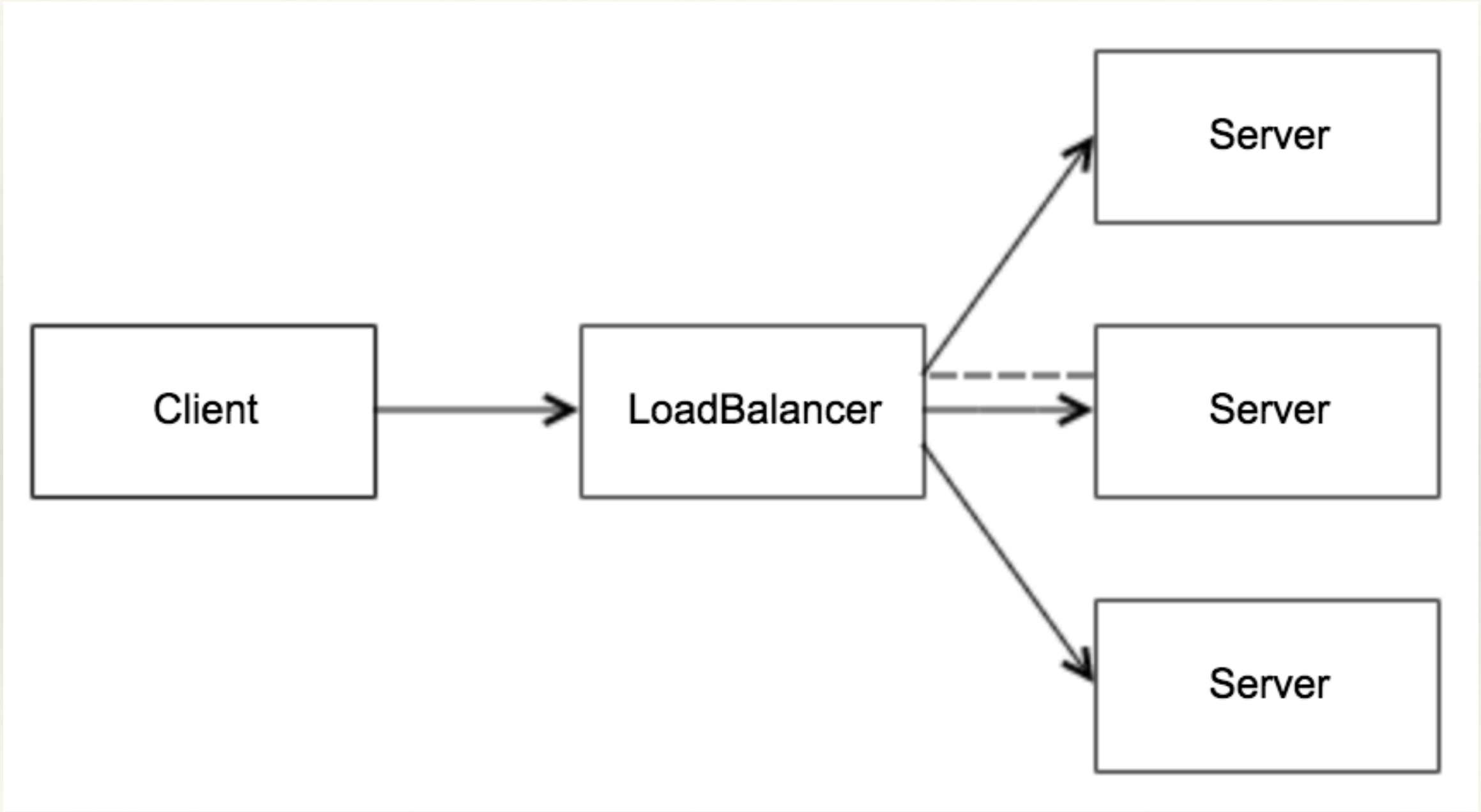
# 跨进程跨网引入的问题


createOrder
ip : port
checkInventory
ip : port
Network
Naming Protocol
Transfer Protocol
Format Protocol

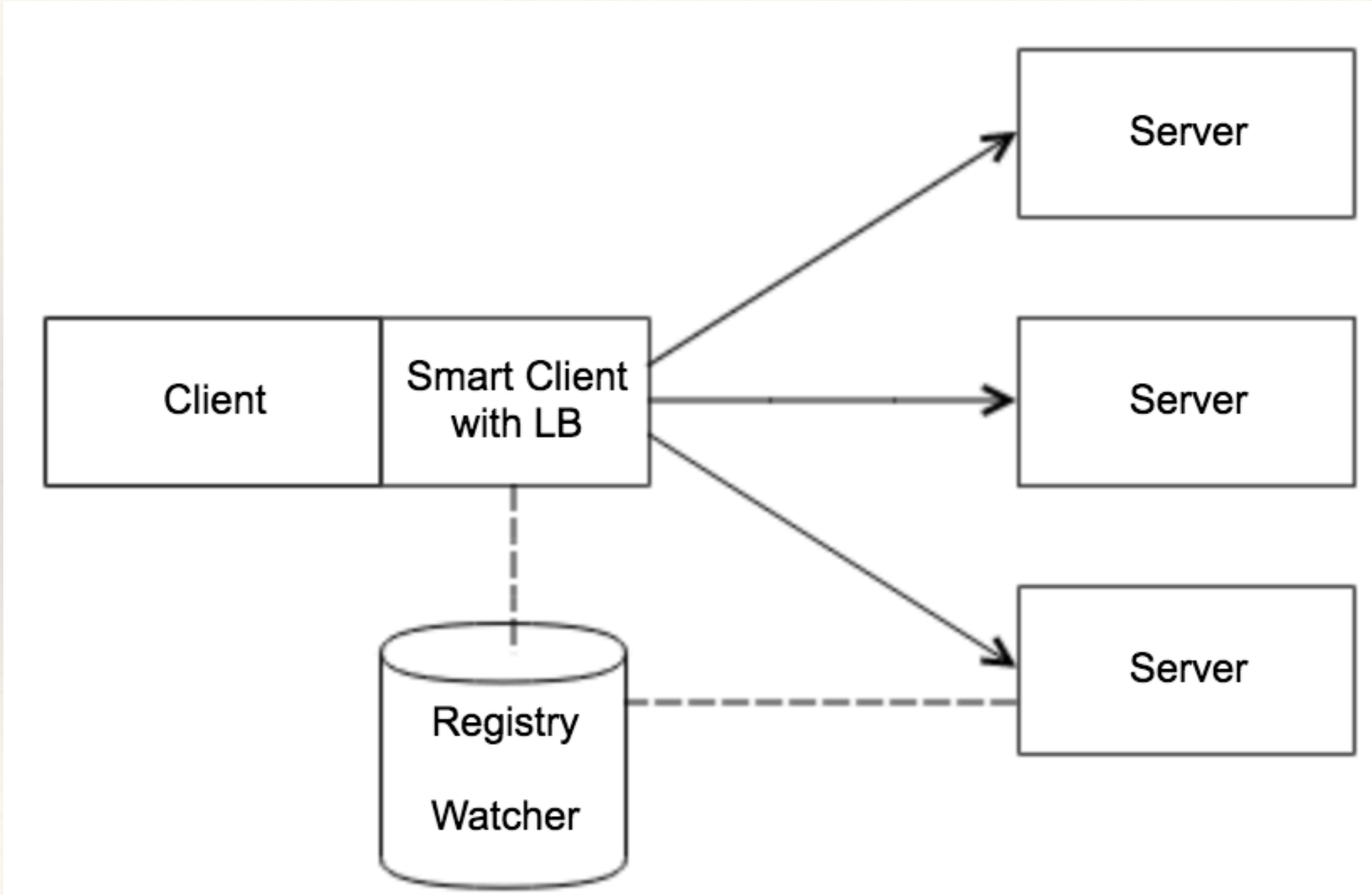
# RPC - Stubbing



# 路由模式：中间件



# 路由模式：客户端


Server
Client
Smart Client
with LB
Server
Server
Registry
Watcher

# 微服务治理选型

- consul
- etcd
- zookeeper
- eureka + ribbon

严复是一位伟大的翻译家

他不懂英文

# 服务协议

- 传输协议（Transfer Protocol）
  - HTTP，HTTP2
  - Custom TCP
- 消息格式（Format Protocol）
  - Protobuffer - IDL
  - Thrift ** - IDL
  - JSON/XML
  - Avro
- 综合性协议
  - AMQP
  - MQTT
  - Thrift
  - Avro

# 服务协议的一点心得

- IDL和Nodejs不是一国的
- 紧凑格式对Nodejs不友好：Protobuffer，Thrift **
- 业务实体对象和传递对象的推演 - 没人做
- 业务实体对象和传递对象的元数据管理 - 没人做
- 格式转换应该是可插拔的 - 没人做
- Google gRPC是一种新的生物：RPC和API的混血

# 服务协议选择

- 面向协议的架构
  - thrift
  - gRPC
- 面向技术的架构
  - dubbo
  - senecajs
  - eureca.io

# App + BaaS之困惑

```
 7    const startup = async (ctx)=>{
 8        //获取用户信息并验证用户登录
 9        const user = await ctx.getCurrentUser();
10        if(!user || !user.isValid()){
11            return handleLoginFailure();
12        }
13
14        //获取登录用户所有最新“好友”
15        const friendIds = ctx.getAllFriendIds(user.id);
16        const friends = [];
17        for(let friendId of friendIds){
18            friends.push(await ctx.getUserDetail(friendId));
19        }
20
21        //获取登录用户所有最新“邀请”
22        const invitations = [];
23        const invitationIds = ctx.getAllInvitationIds(user.id);
24        const invitations = [];
25        for(let id of friendIds){
26            invitations.push(await ctx.getInvitation(id));
27        }
28
29        return {user, friends, invitations};
```

# App + BaaS之困惑

- 微服务对应原子API，
- APP端SDK对应页面渲染和导航逻辑
- 前重后轻，动不动就得升级APP
- 需求一变，改三端
- APP已启动5M流量就没了，发出几十个Rest API请求

# API的两种分类

- 面向“业务”逻辑的API：
  - 原子性，不可再分
  - 与微服务一一对应
  - 很少改变
- 面向“应用”逻辑的API：
  - 复合性，由原子API+处理逻辑封装而来
  - 易变化
  - 每个App和版本都不同

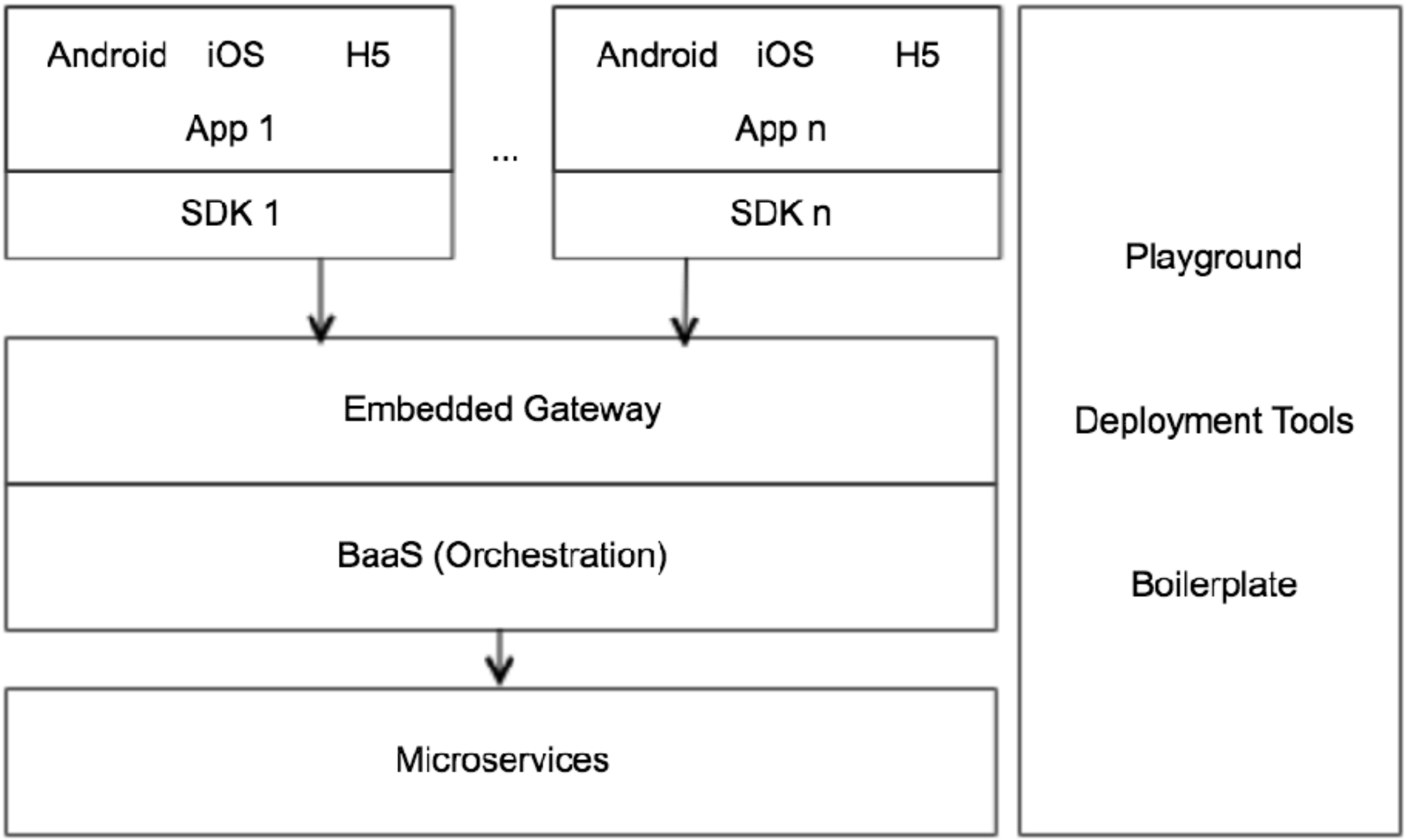
# 几种解决方案

- 前端封装SDK
- 后端封装复合API
- 中间件服务编排与转换
- 翻译层：GraphQL
- 中间层 + 自动化 => PaaS + 服务编排 => BaaS

# 服务编排 - Orchestration BaaS

- APIs (RPC functions) Management
- Atomic APIs (RPC functions) connecting
- Composite functions runtime
- A framework for function wrapping
- Playground
- Deployment Tools + Boilerplate project

# 服务编排 - Orchestration BaaS



# 弹性服务编排 (FaaS/Serverless)

- 增加弹性：自动创建和销毁实例（百毫秒级）
- 按量付费：Hosting —> Using（调用级）
- 自服务化
- 多语言多技术

# 一点分析

- 操作系统 — 硬件：成就了微软，Windows和个人电脑
- 虚拟机 — 操作系统：成就了VMware、*aaS和无数云计算公司
- 容器 — 虚拟机：成就了运维、FaaS和？

- 要不向下走，去练气，做专家；要不向上走，去练剑，去杀人