

# 瓜子后端技术架构的变迁

瓜子二手车高级技术总监 纪鹏程

## 自我介绍

纪鹏程

- 2000.9-2004.7 南开大学
- 2004.9-2007.7 中科院软件所
- 2007.7-2011.5 百度-贴吧
- 2011.5-2015.9 赶集-交友、社区、二手物品交易、二手车交易
- 2015.9-now 瓜子二手车





# 演讲大纲

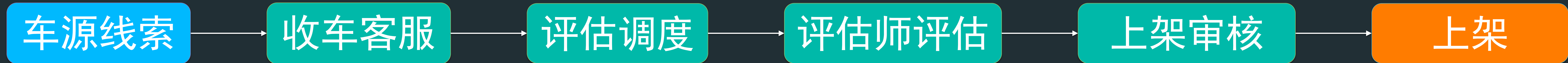
- 瓜子及二手车业务流程介绍
- 架构变迁过程的经验与教训
- EP的实践工作
- 目标

# 关于瓜子

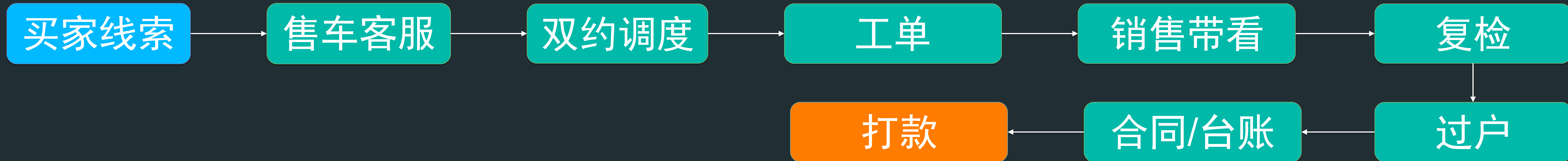


# 二手车C2C交易流程

- 收车流程



- 售车流程



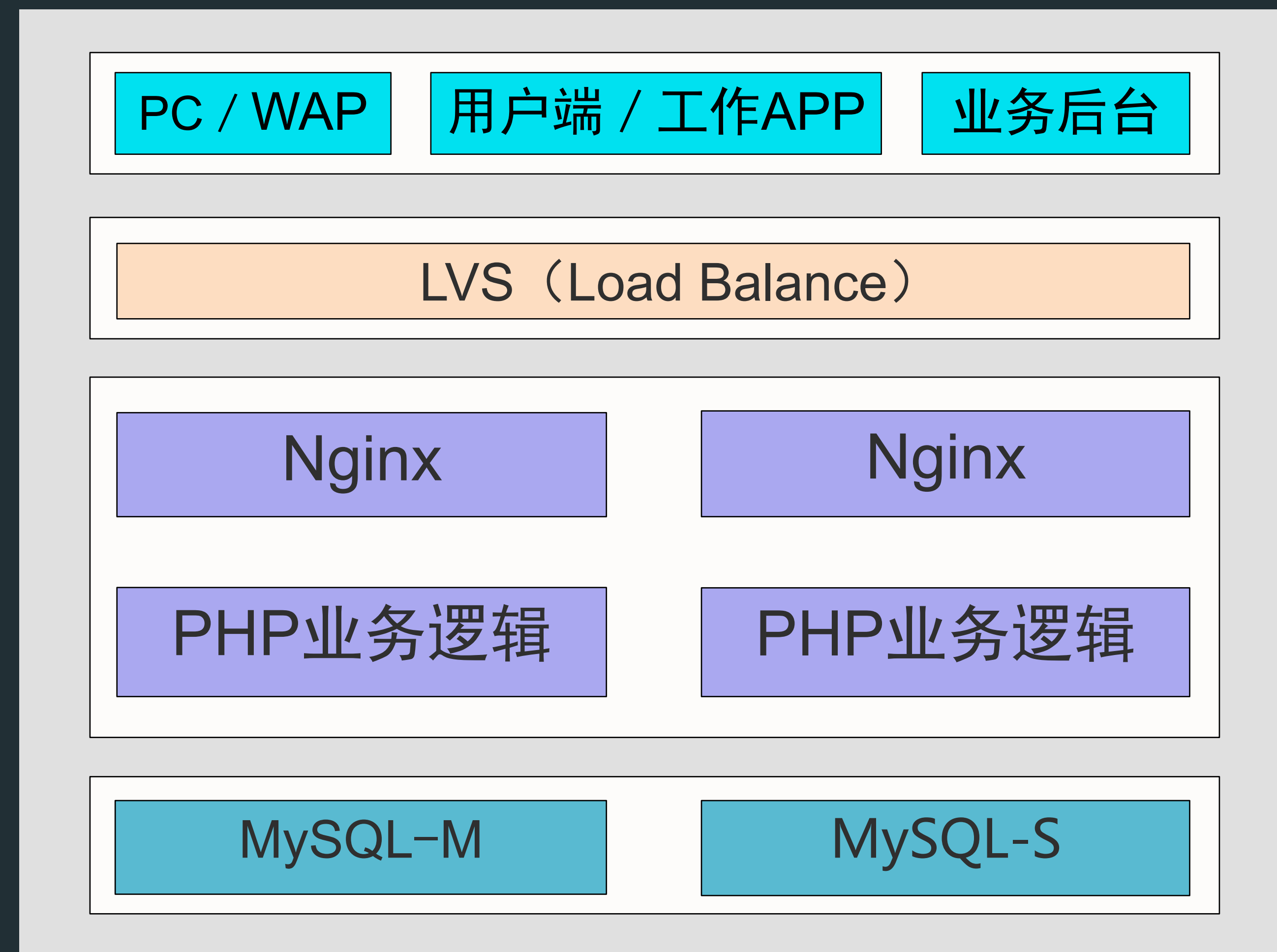




# 石器时代



# 架构V0.1



- “Quick but dirty work”, “过早优化是一切罪恶的根源”
- 主要精力集中在业务上, 快速迭代, 实时响应业务需求
- 周边服务尽量通过外部购买
- 缓存的使用可以先缓缓
- 从开始就要重视“单点”问题
- 尽早确定数据库规范、数据字典
- 敏感数据的加密和脱敏处理

# 问题



- 代码可维护性差，新人介入成本太高
- 耦合太紧密，很小的错误也会造成整体的crash
- 全部业务都在一个DB实例上，性能扩展性极低
- 日志缺乏和规范不统一，定位问题困难
- 系统、应用、业务级别的监控缺乏
- 从赶集体系剥离，一些第三方服务需要重新搭建

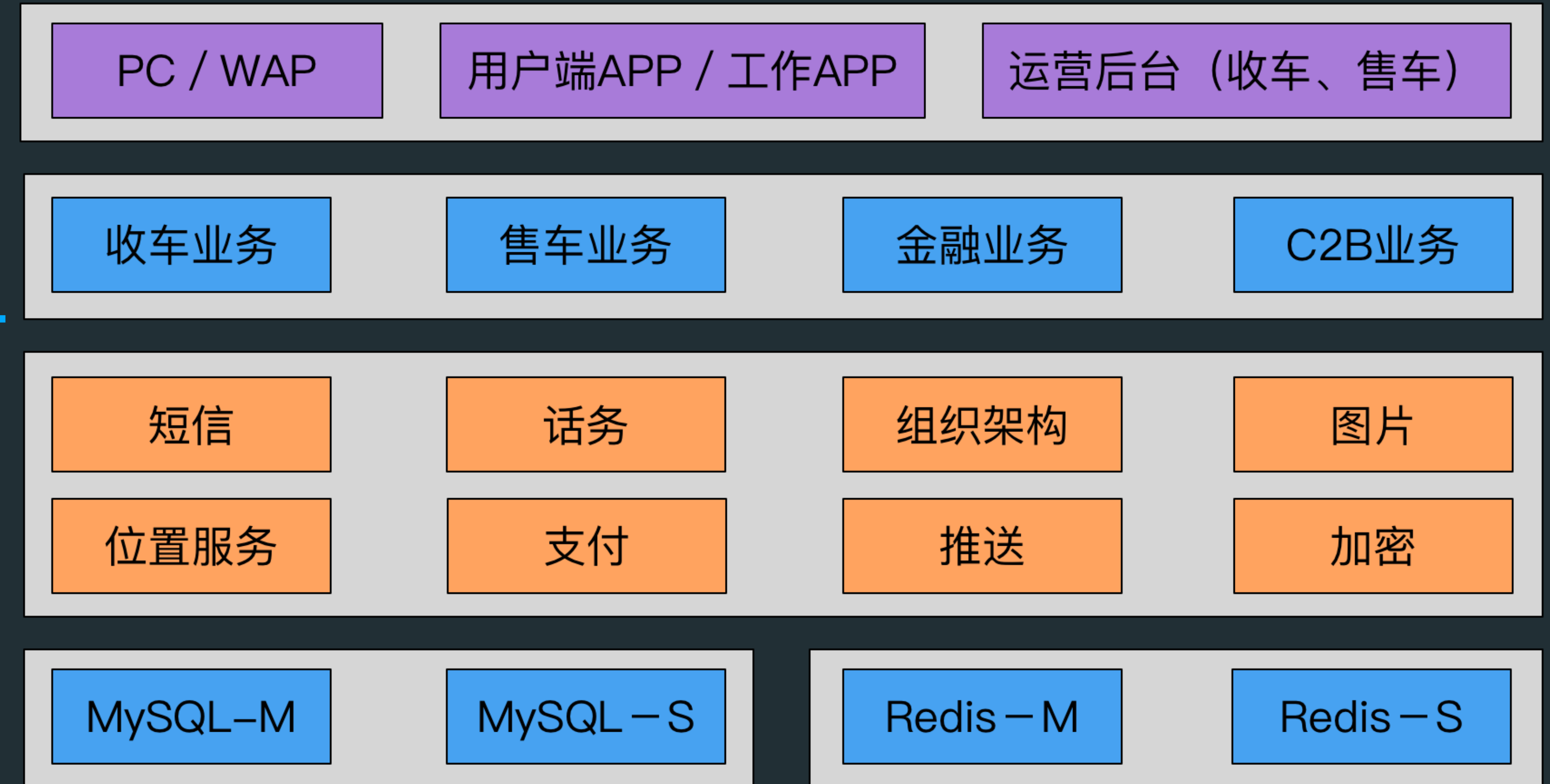




# 铁器时代

# 架构V0.5

- 代码按照业务线分拆，业务之间通过HTTP接口调用
- 数据库按照业务线分实例，实例内分库分表
- 搭建监控平台，增加 XHProf 和应用日志，提供订阅功能
- 制定统一应用日志规范
- 增加Redis集群，基于Twemproxy来做中间代理





# PHP的性能发现

- 所有业务统一接入XHPProf性能采样
- 业务不同，采样率不同
- 数据导入ES中，方便根据业务线、时间进行检索
- 增加订阅机制

# 问题



- 业务内部拆分不够细，耦合依然比较重
- 代码分层混乱，越级、跨级调用多
- 业务间接口调用方式千差万别
- 接口的可用性和性能监控不够
- 上线工具不完善，不支持回滚
- 测试以黑盒为主
- 慢查询问题层出不穷

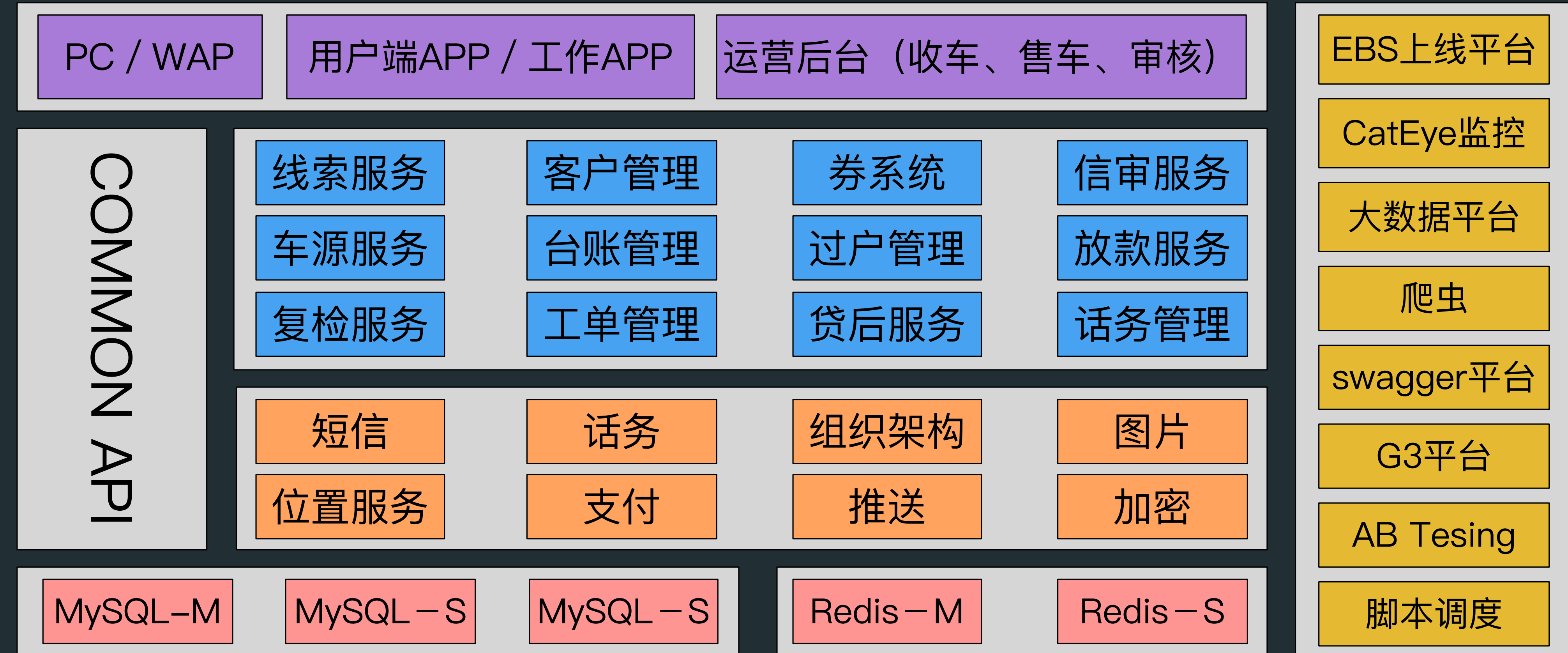




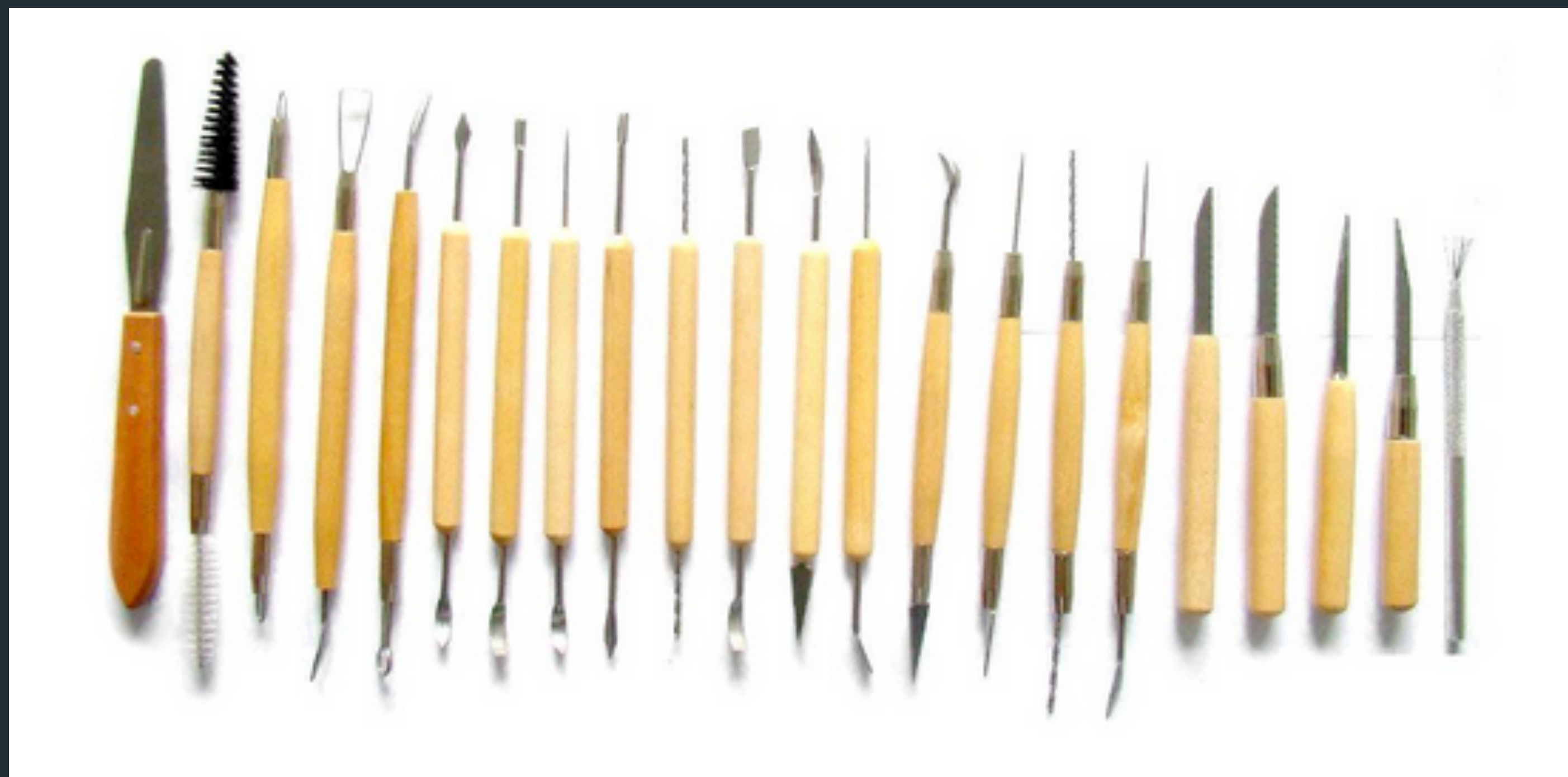
# 蒸汽时代

# 架构V1.0

- 业务内部拆分出细粒度的服务
- 业务之间通过内部COMMON API接口调用
- EBS上线平台，实现自动上线和回滚
- 数据库平台增加对SQL上线的审核
- 模块内代码分层，禁止跨层、跃层调用
- 第三方服务要使用多家，保证可以无缝切换
- 引入自有研发的A/B测试框架







# EP的实践工作

# 代码的艺术

- 代码的分层
- 代码规范 – PSR1、PSR2
- 使用git hook，代码提交前code sniffer检查准入
- 内部库，使用Composer封装和集成
- 代码遵循PSR4自动加载机制



# 单测

- 单测重要性一直被低估
- 持续集成的关键
- 核心逻辑代码一定要做，覆盖率要达标
- 用好的方法让单测更简单

## PHPUnit的Mock

- ✓ Public
- ✓ Protected
- × Static
- × Final
- × Private

## PHPUnit的Mock

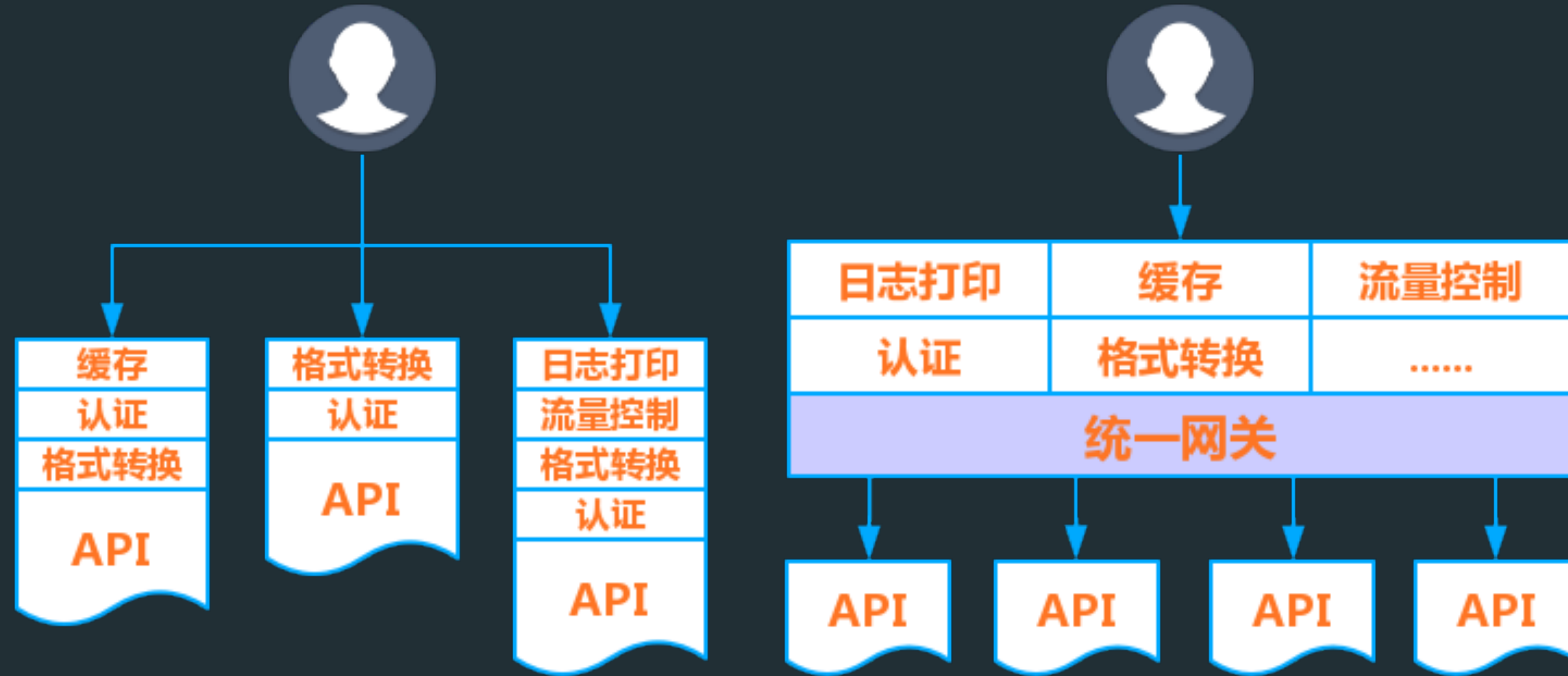
- ✓ Public
- ✓ Protected
- ✓ Static
- ✓ Final
- ✓ Private

# 接口的管理

- 接口的重要性与日俱增
- 用swagger集中统一管理
- 接口测试的case管理
- 可用性及性能监控



# 接口网关



- API的统一入口，路由分发
- 协议转换、身份认证、权限控制、流量控制、日志记录等
- KONG + Openresty

# RPC的实践

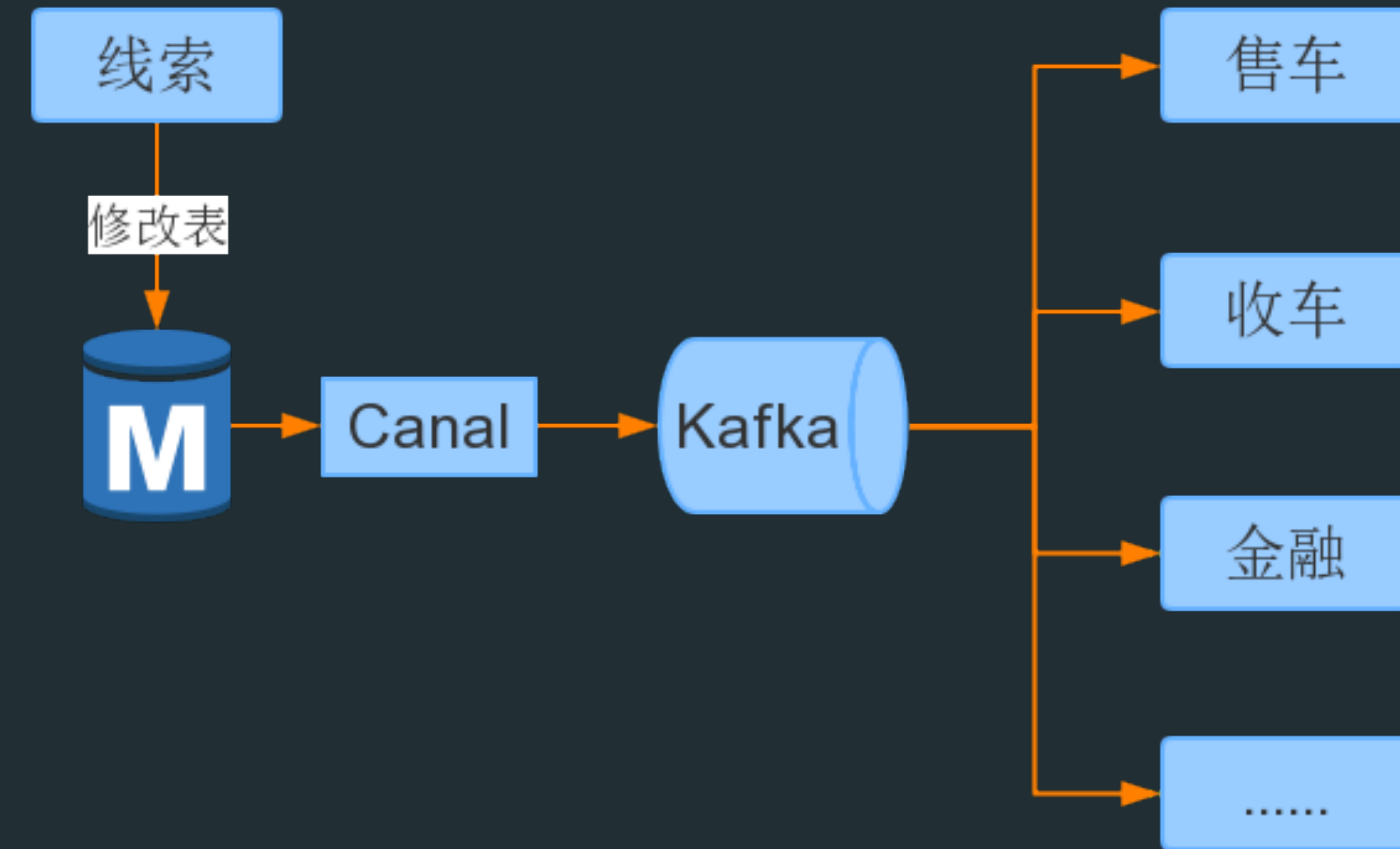
- 规范、统一接口的调用方式
- 基于HTTP接口的简化版
- 在Guzzle的基础上进行封装，支持签名、并发请求、超时设置等
- 要跟内部接口区分开，加特殊前缀，防止滥用！！！！



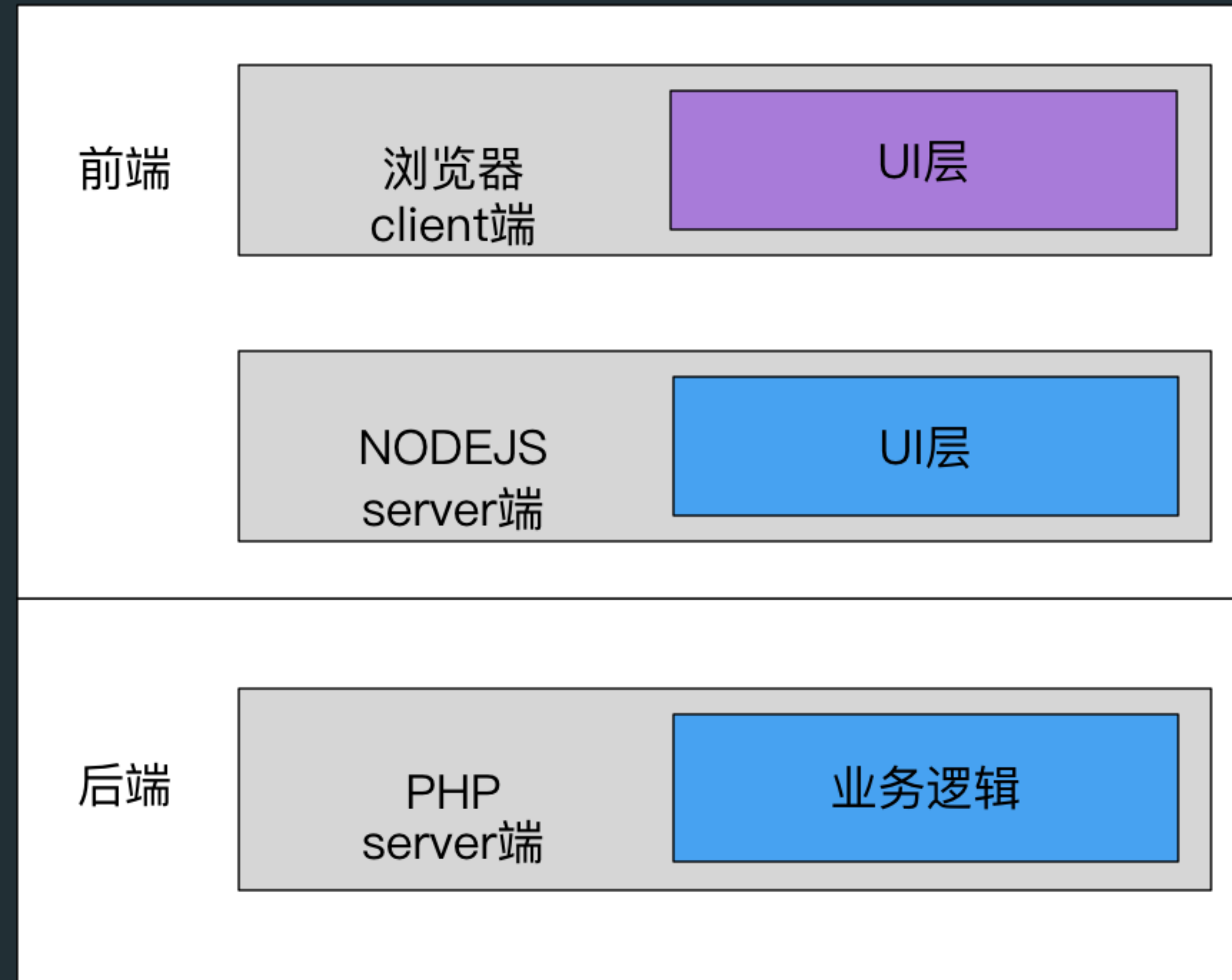
# 服务的解耦

服务级别, Rabbitmq & Kafka

数据表级别, Canal+Kafka

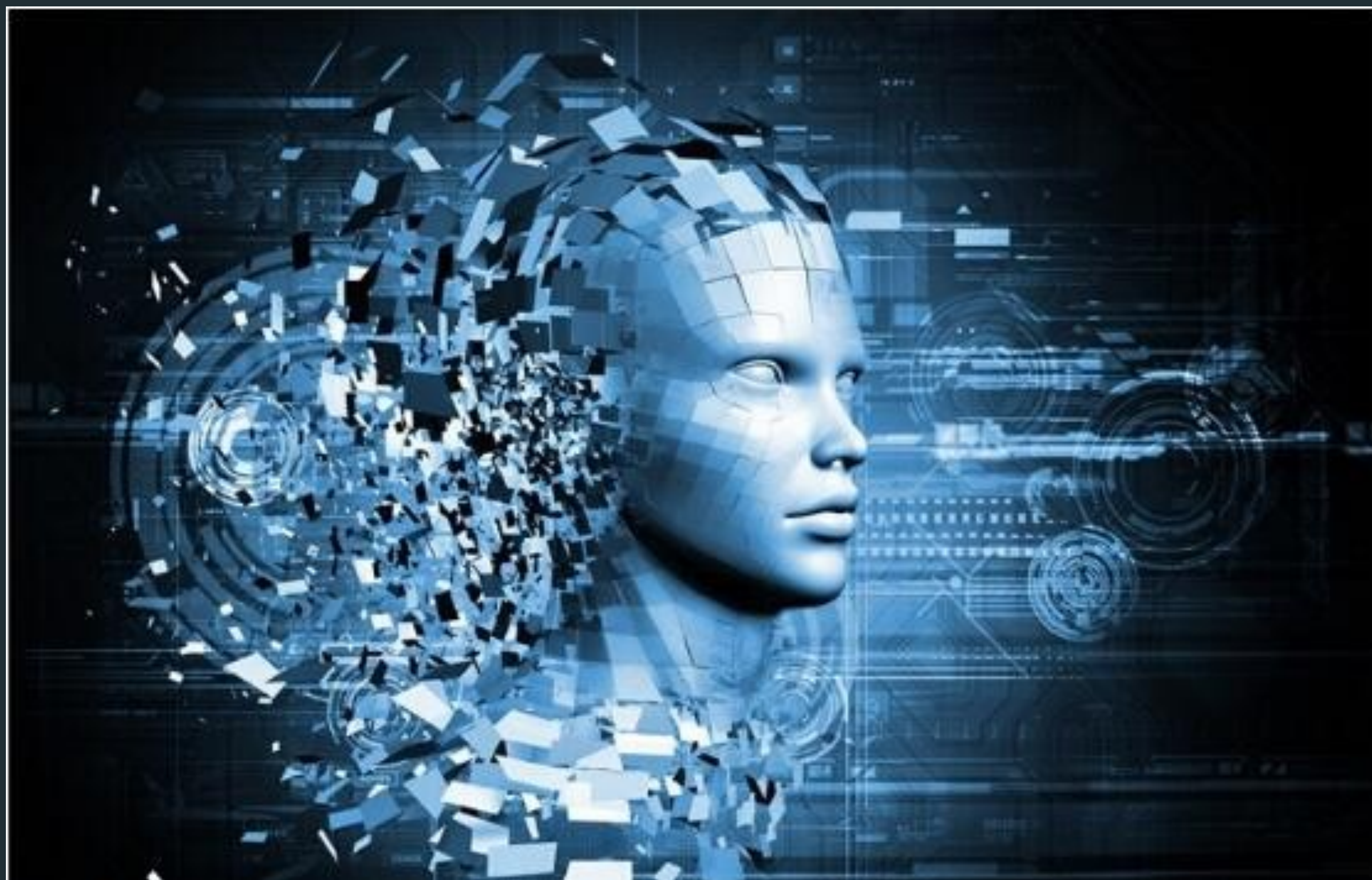


# 前后端分离实践



- 满足SEO, SSR的必要性
- 大前端思想
- 前后端界限更加清晰
- 后端开发更加聚焦在架构和逻辑上
- NODEJS让并发能力进一步提升





# 目标



目标

平台化



服务化



智能化





# 平台化

- 基于docker的PAAS平台让资源更有效的配置
- 容器的部署、管理，服务的部署
- 持续集成（CI）和持续交付（CD）的能力

# 服务化

- 微服务化不是银弹
- 服务粒度的权衡
- 处理好分布式系统带来的复杂性和时间消耗
- 需要中间件配合，服务注册、服务发现、服务部署、监控等

# 智能化

- “算法+数据+应用场景”的模式  
将变得前所未有的重要
- 瓜子大脑



## 商业决策

- 1、业务报表
- 2、多维分析&可视化
- 3、数据魔方

## 主场景应用

- 1、销售/评估/客服分单调度
- 2、车源成交/车价预测
- 3、匹配&个性化
- 4、投放优化

## 大数据金融

- 1、反欺诈模型
- 2、信用评估模型
- 3、风险定价

## 基础设施

- 1、大数据平台
- 2、Tracking系统
- 3、瓜子基因图谱库

## 人工智能算法引擎

- 1、分类模型、聚类模型、回归模型
- 2、NLP, CV, Speech Recognition
- 3、深度学习



# Q&A

Email: [ji.pengcheng@foxmail.com](mailto:ji.pengcheng@foxmail.com)

# PHP 2017·北京

## 全球开发者大会

