

# PPmoney 微服务之路

作者：PPmoney 基础架构 敖小剑



# Why



- 现状，存在的问题
- 出路在哪里？
- 为什么选择微服务？



PPmoney万惠是万惠金科（股票代码：430705）旗下的全资子公司，也是互联网金融行业中专注于服务及安全的领导企业。

当前，平台已安全运营近四年时间，累计成交额超600亿元，为用户带来近10亿元收益。

## 技术栈多样化

- 黑猫白猫，能上线就是好猫
- 怎么快怎么来，包括买
- 手头什么人，就用什么技术

## 野蛮成长

- 初创型企业，商务推动
- 产品需求多，时间短
- 快速开发，立即上线

## 规划不足实施艰难

- 需求太多，来不及规划
- 变更太快，规划跟不上
- 能力和精力有限

## 人员变动频繁

- 知识传承不足
- 技术积累不足
- 问题长期积累，难于解决

## 推高成本

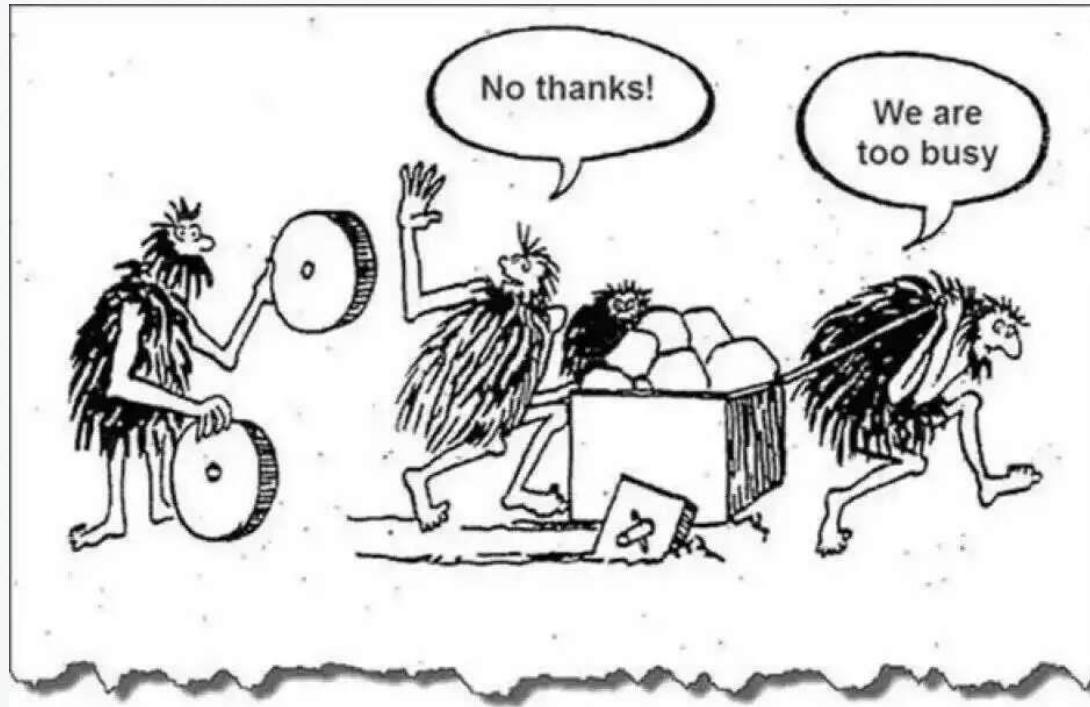
- 维护越来越难
- Bug越堆越多
- 新功能开发速度不理想
- 恶性循环.....





这一步，不是快与慢

而是生与死



- 方法不得当，开发效率低下
- 问题积累，不堪重负
- 没有时间改进，咬牙硬抗

### 无序变有序

- 统一并简化技术栈
- 规范开发流程



### 起点更高

- 基础类库
- 基础设施
- 基础组件



### 速度质量平衡

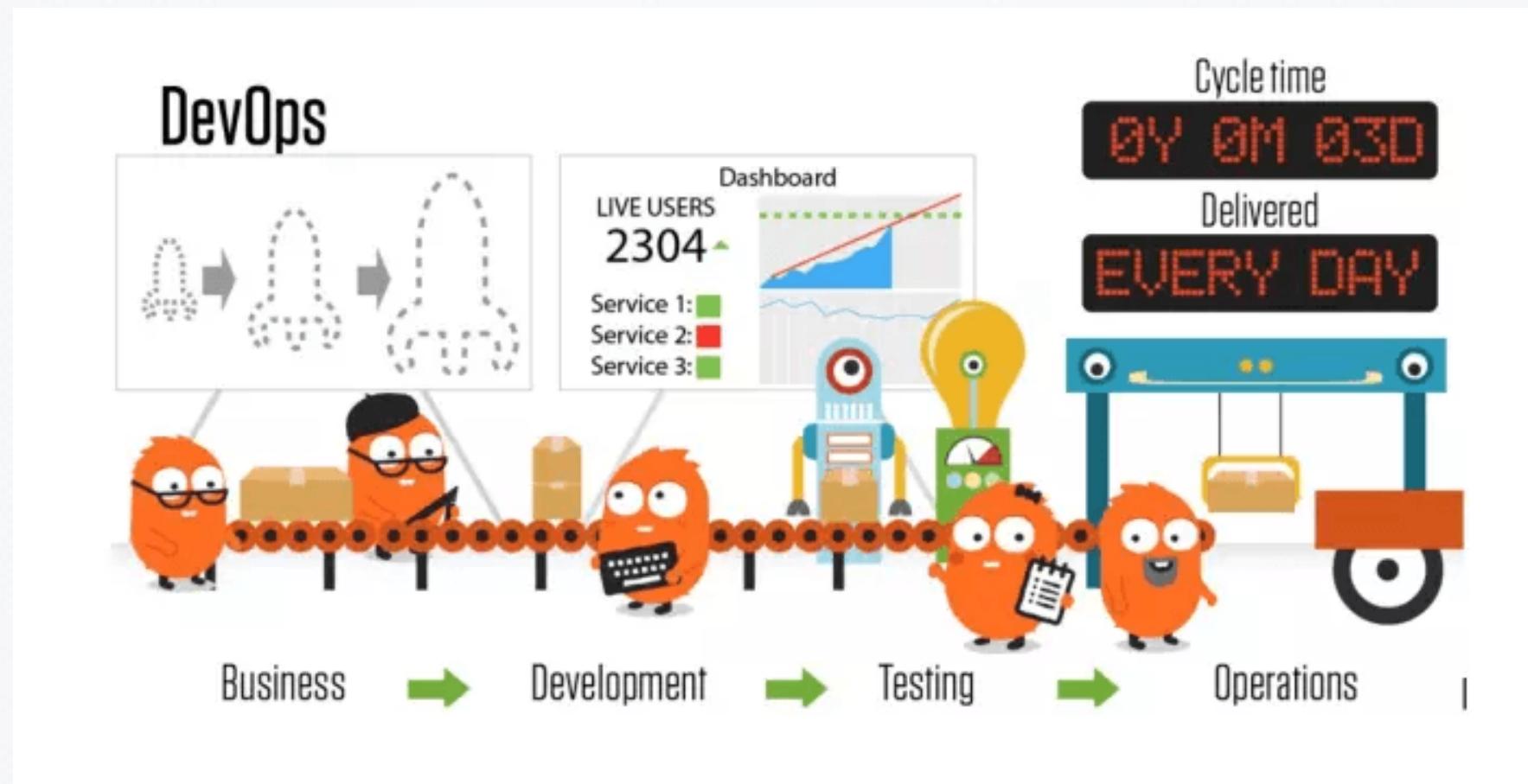
- 敏捷开发
- 持续集成/持续发布
- DevOps

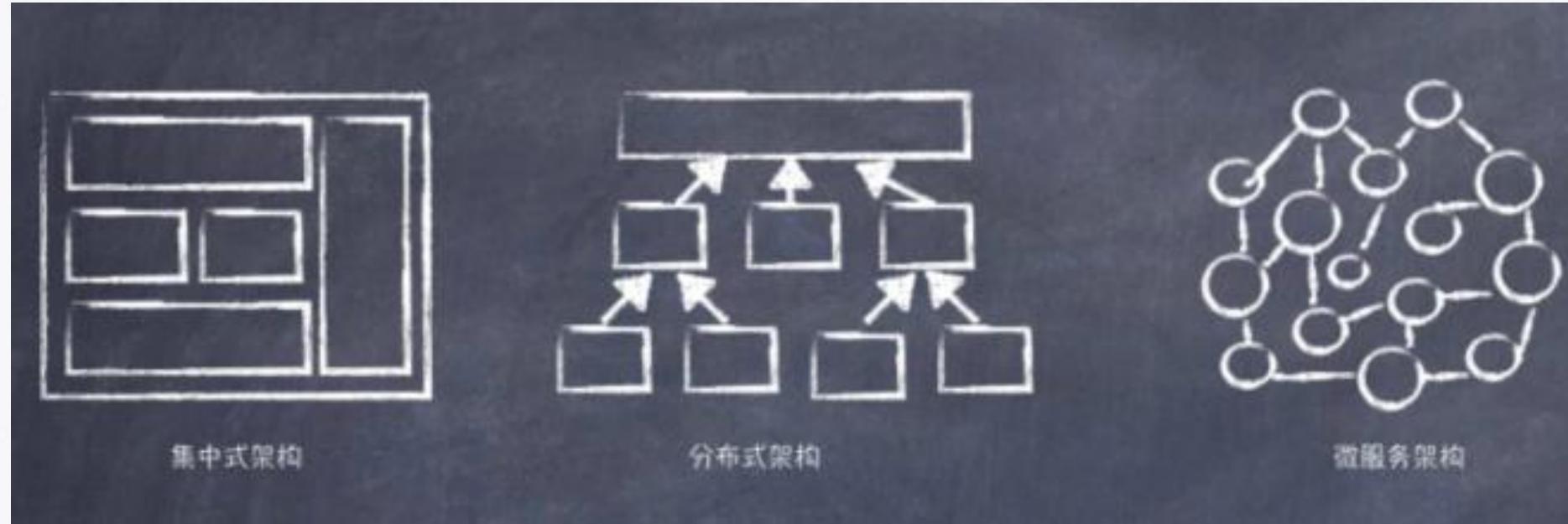


### 摆脱低级重复

- 自动化测试
- 自动化部署
- 云技术/容器化







# How

- Dolphin微服务框架
- 技术选型





请记住这只可爱的小海豚

我们的 Dolphin，我们的微服务框架





**gRPC**

Google开源的RPC  
框架，支持多平台  
多语言，特型丰富，  
**支持手机APP**

01

gRPC

02

SPRING



**SpringBoot**

过去两年中Spring团队集大成  
之作，业界好评如潮的微服务  
开发框架，**好用易上手**



**Etcd3**

**新鲜出炉**的分布式高可  
用一致性解决方案，用  
于服务发现和配置

03

distribution

### Protocol Buffer 3.0

在proto2上改进，以便支持Android  
Java, Objective C和Go，从而可以用于  
手机App端



2

编解码

1

协议

### HTTP/2



- 多路复用：一个socket跑8w QPS，不再需要连接池
- Server Push：服务器可以将响应主动“推送”到客户端
- 报头压缩，HTTP/2降低了开销

3

平台

4

类库



### Netty 4.1

- 主要提供对HTTP/2的支持



### Android/iOS

- App可以受益
- 有机会打通全套流程

### 有效的网络利用

- 单连接多路复用，无需多条链接或者多次请求排队
- HTTP/2无需像HTTP/1需要多次TCP握手和销毁

### 高效的编解码机制

- Proto3编解码速度和效率，非常接近thrift，远超JSON
- 对于App端，更小的CPU使用，节能省电；网络带宽消耗少，省流量



### 支持Stream

- 支持服务器端stream和客户端stream，包括同时stream
- 强调：特别适合用来做服务器端主动消息推送，如[变更通知](#)

### 灵活的编程API

- 支持异步编程模型，包括服务器端和客户端
- 从.proto契约文件自动生成的代码，已经同时包含同步和异步两种API



## 1 手机还是PC?

我们平时用哪个多？未来呢？



1

## 2 公司的用户

我们的用户，用app的比例有多少？投资的比例有多少？

2



未来

有什么理由将app排除在开发框架之外？！

4

## 3 现状

我们现在app和服务器通讯的机制是什么？有改进的空间吗？



## 内建starter



- 内建多种聚合模块/starter
- 是一些类库的组合，开箱即用
- 充分体现约定胜于配置

2 内涵

1 效率

4 受众

3 优化

## 生产率高



- 设计出发点就是**快速开发**(参考RoR等)
- 内置构件封装底层框架的复杂性
- 提供许多通用的功能
- 和spring高度集成

## 面向微服务

- 特别针对微服务架构开发
- 为开发者带来免费的关键功能



## 受众广泛

- 基于久经考验的Spring
- 使用者众多，拥有最广泛的群众基础
- 门槛低，易入门
- **好招人，好培养，好交流**



### 特性丰富

- 使用Raft 算法
- 支持多数据中心



### 外部集成

- 动态生成 nginx 和 haproxy 配置文件
- 提供DNS，方便接入

等等，我们刚刚发现了更适合Dolphin的：

## ETCD 3



### 开箱即用

- 提供成品，非原材料，开发量少，不易出错
- zookeeper只提供原始的K/V值存储
- 内置服务发现
- 内置失败检测



### 使用简单

- 支持 http 和 dns 协议接口
- 官方提供web管理界面



3月 

- 开始开发

9月 

- 大量细节更新和改进
- Consul换etcd3
- \* 配置中心
- \* 鉴权中心
- 完善API gateway



5月 

- 服务注册/服务发现
- 负载均衡
- 多版本共存：小版本兼容，大版本隔离
- 平滑上线下线/主动下线
- 加密
- \* 集成pinpoint
- 文档，demo

11/12月 

- \* 服务中心
- \* 资源中心
- 限流
- 熔断
- 服务路由

\* 配套的基础设施



More

- 微服务需要体系做支撑
- 基础设施
- 实施的难点和挑战



指望 **微服务** 单枪匹马的解决问题

下场只能是 Game Over!

## 服务中心

- 查看服务实时状态
- 管理服务上下线/灰度
- 设置服务路由，限流

## 配置中心

- 统一配置
- 实时下发

## 应用性能监控

- 标准APM
- 使用开源Pinpoint

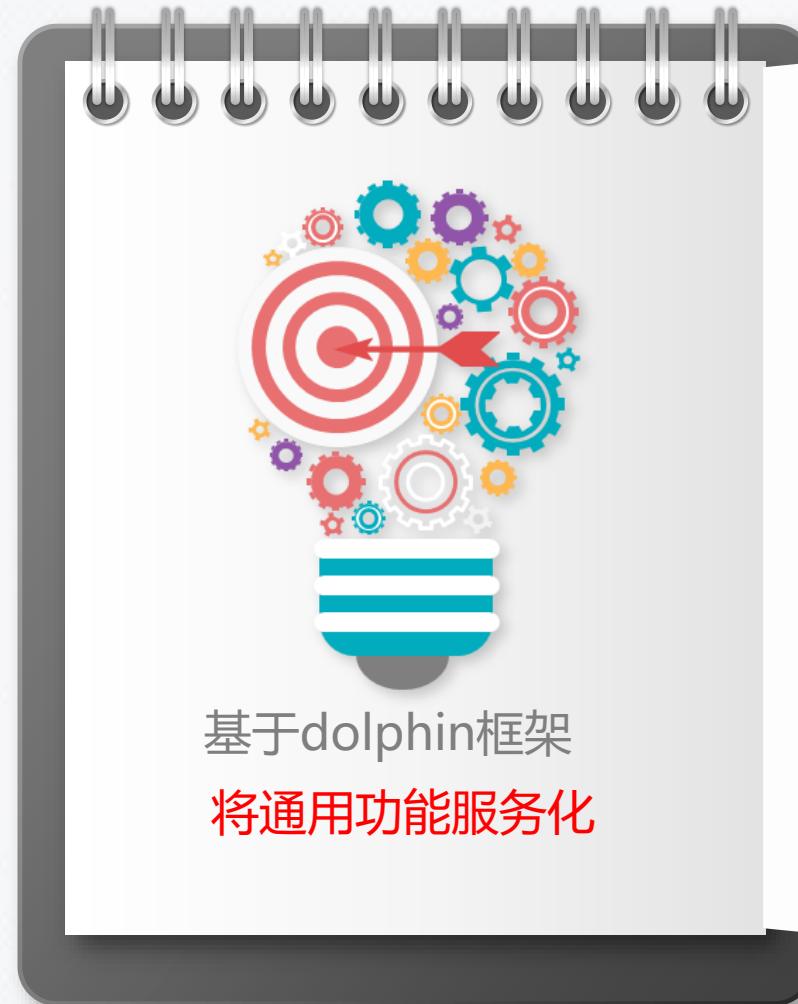
## 资源中心

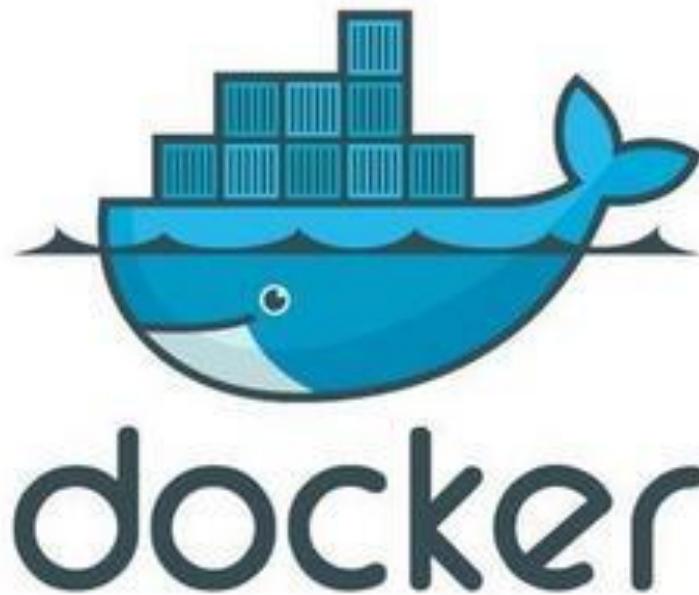
- 统一配置系统资源
- 监控资源使用

## 日志管理

- ELK
- Graylog做错误管理







还处于探索阶段.....

## 开发测试流程

- 真正的敏捷起来
- CI/CD/自动化
- TDD/BDD 有更好



## 组织决定架构

- 组建跨功能的团队
- 减少部门墙带来的沟通成本



## 产品而非项目

- 团队对整个产品的生命周期负责
- 吃自己的狗粮



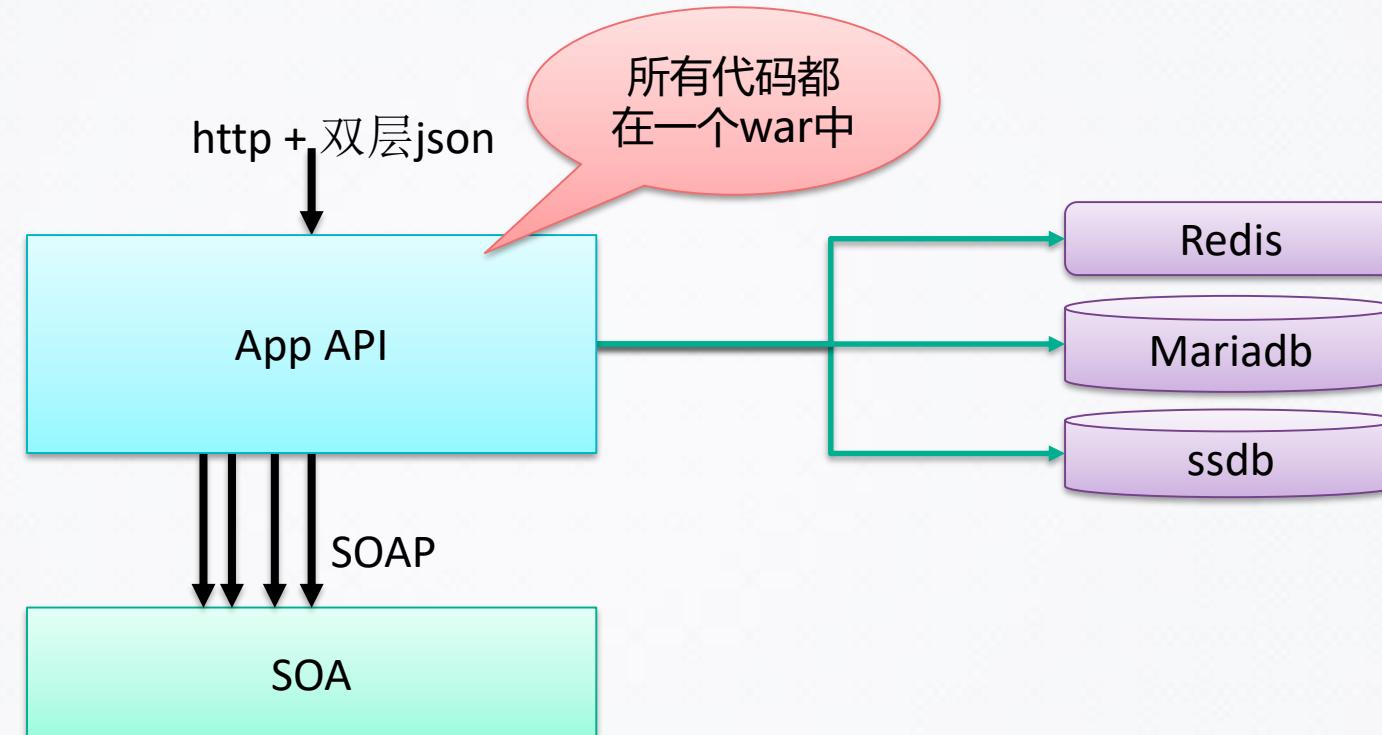
## 对团队高要求

- 对设计有高要求
- 对开发有高要求
- 对测试有高要求



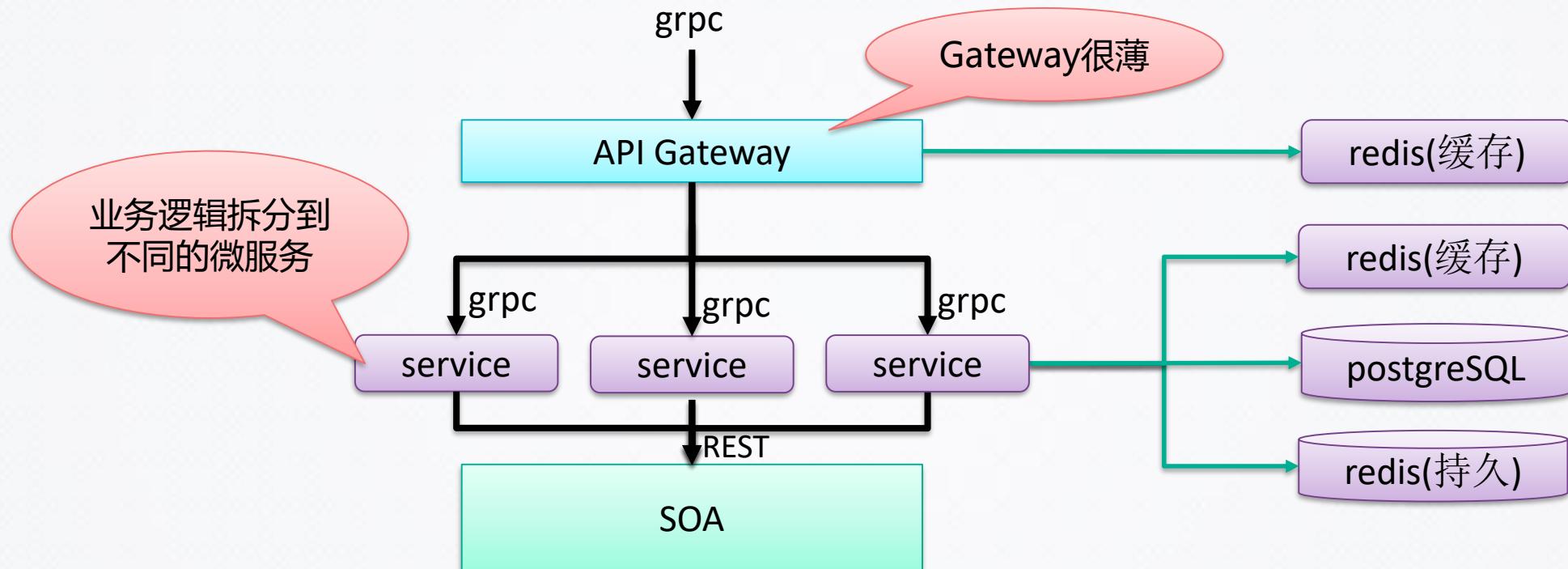


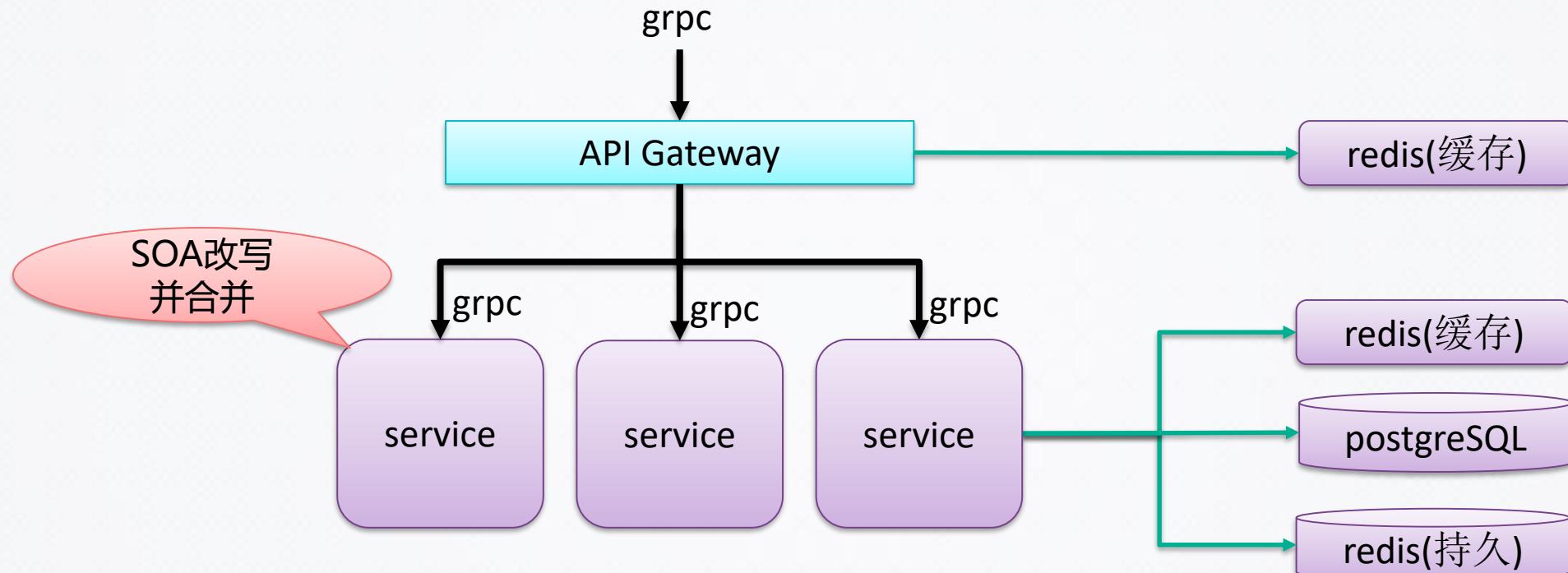
## 实践1：旧有系统改造



真实案例： 理财App后台的原有架构

## 改造：第一阶段





## 实践2：开发团队建设

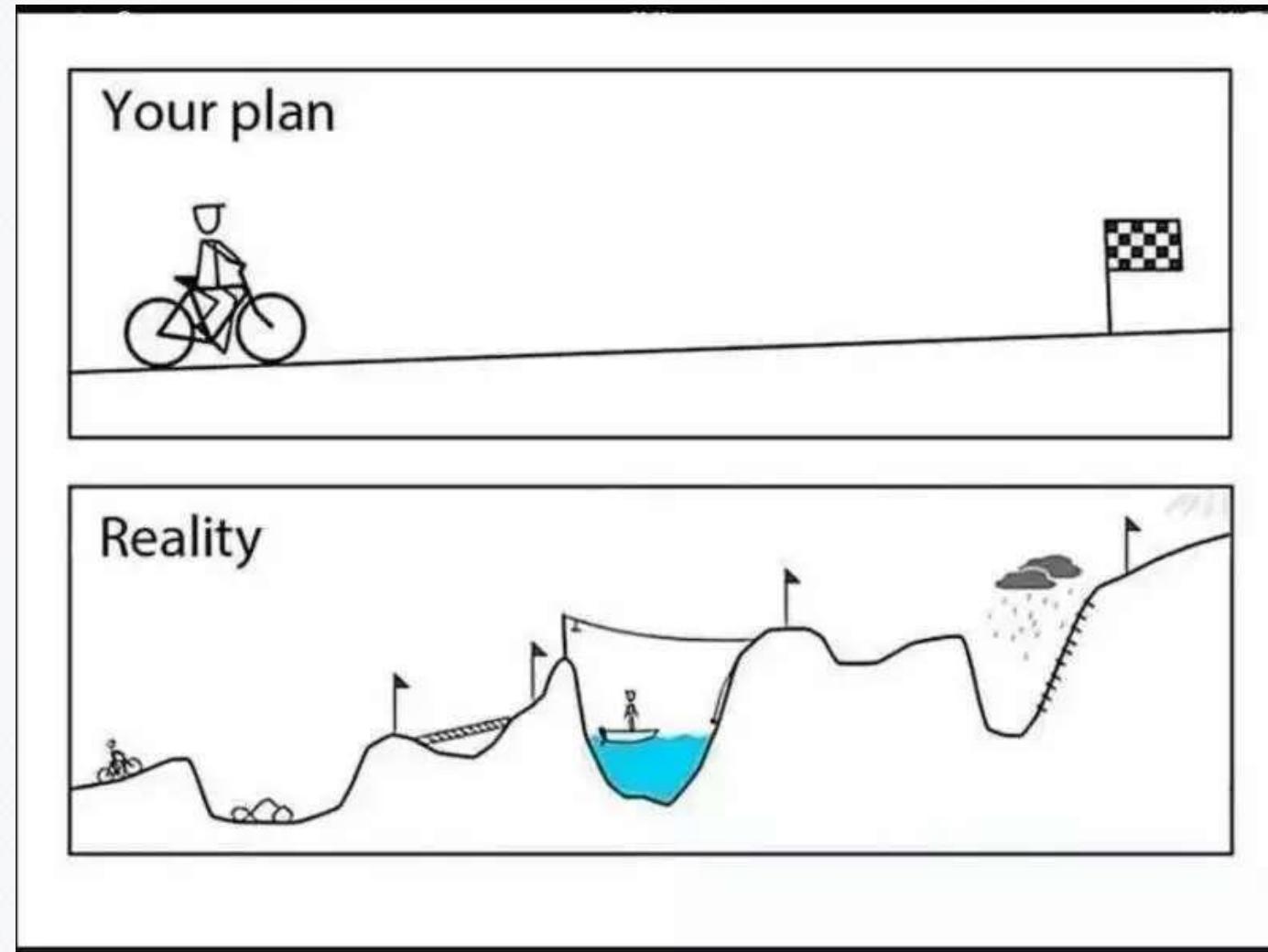


先将架构部门打造为高素质的敏捷团队



再孵化满足要求的新型业务开发团队

提醒：道路艰险





## 难点1：要投入，有门槛



面对英国记者“英企能否反过来在中国投资建设核电站”的疑问，中国驻英大使刘晓明反问BBC：

- 你们有**资金**吗？
- 你们有**技术**吗？
- 你们有**专家**吗？

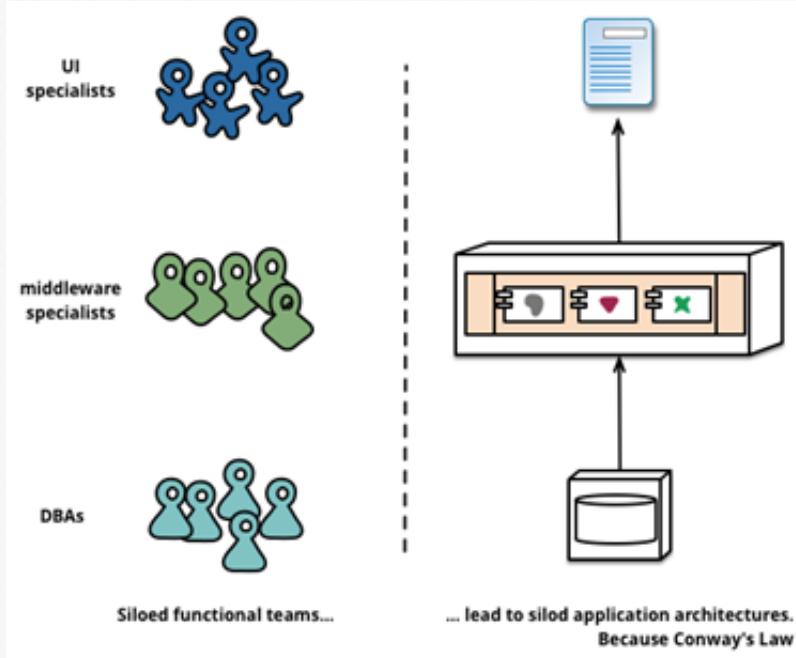


对于小作坊式的原始开发团队，变革是颠覆性的：

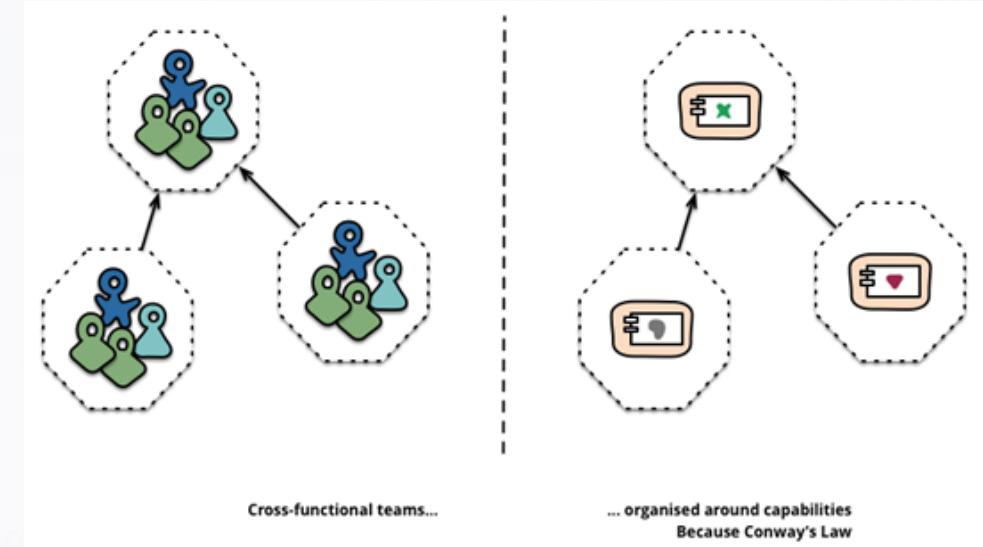
- 敏捷，CI/CD
- 各种自动化
- 服务化概念
- 新的开发流程
- 新的测试流程
- 新的部署方式
- 新的问题排查方式
- .....

## 难点3：对组织架构动力

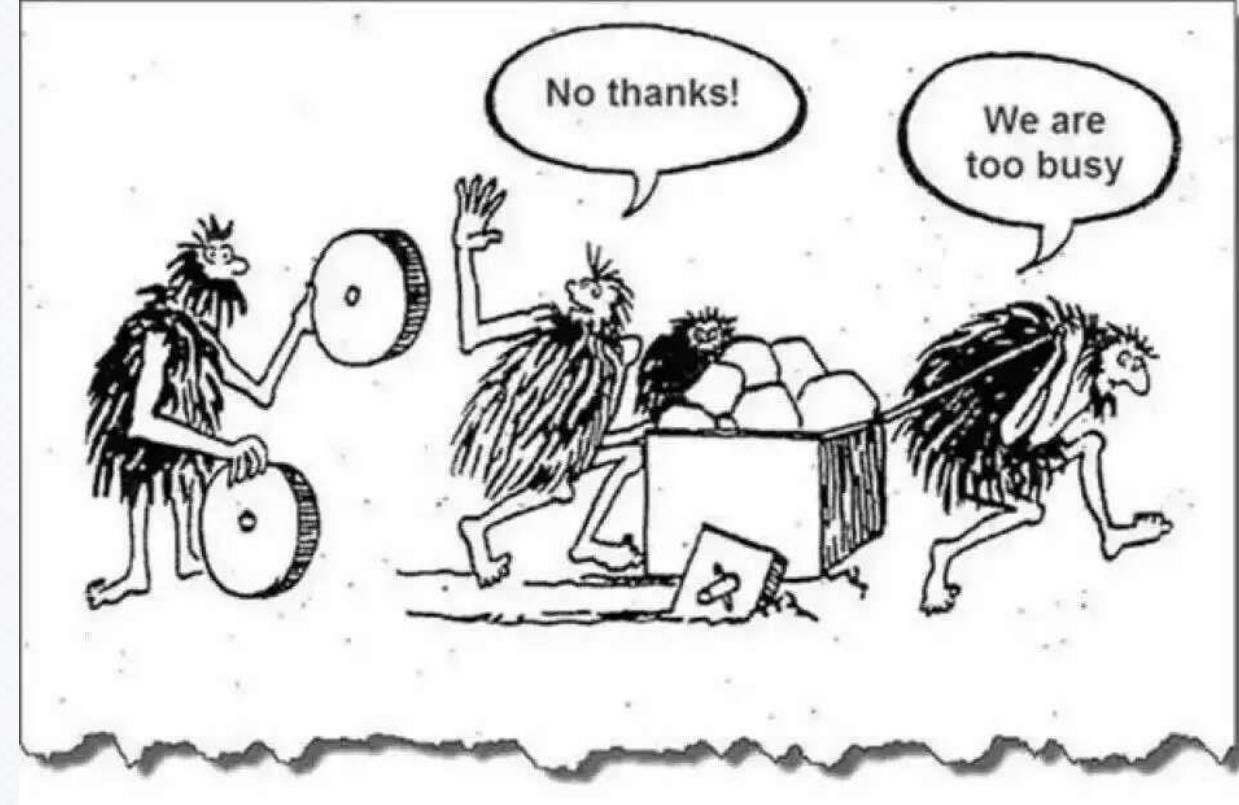
按照功能划分团队



按照业务边界划分团队



- 微服务架构真正的成功之处在于拥抱小型，跨职能团队，并且鼓励扁平、自我管理的组织
- 按照业务而不是技术来划分组织，在一个团队内全栈，让团队自治



这个悲伤的故事依然还在上演.....

3月 

- 开始开发

5月 

在 Dolphin 完成主要功能并通过实际项目检验之后开源，预计时间为：

# 2016年底

9月 

- 大量细节更新和改进
- Consul换etcd3
- \* 配置中心
- \* 鉴权中心
- 完善API gateway

11/12月 

- \* 服务中心
- \* 资源中心
- 限流
- 熔断
- 服务路由



PPmoney , 诚邀各路英才!

请把简历砸过来: [aoxiaojian@ppmoney.com](mailto:aoxiaojian@ppmoney.com)



年底 GitHub 见 .....