QINIU

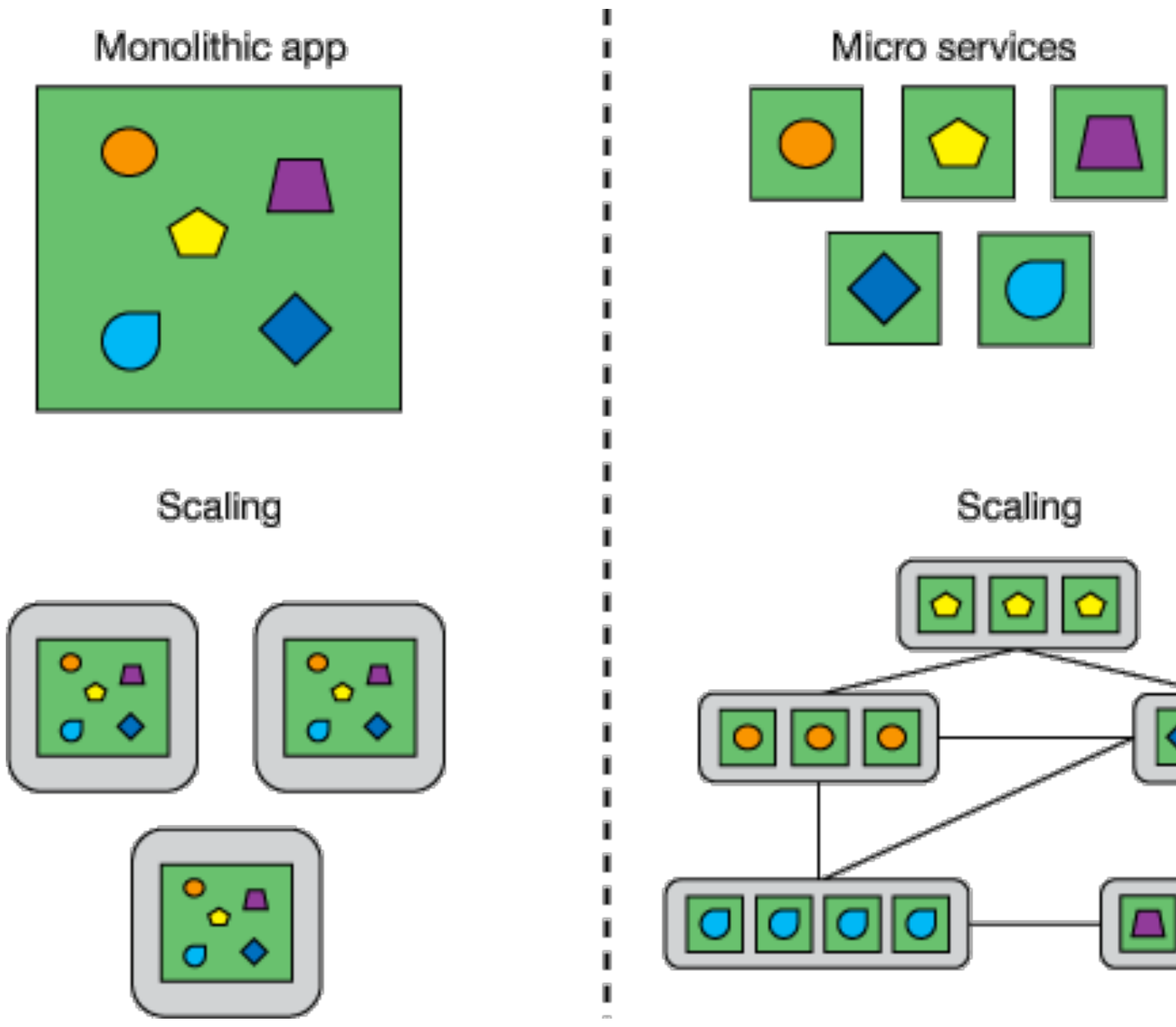
基于容器的微服务架构实践

陈爱珍

- 单体应用 VS 微服务
- 基于容器构建微服务架构
- 七牛的微服务架构实践
- 踩过的那些坑

单体应用 VS 微服务

单体应用 VS 微服务



单体应用 VS 微服务

单体应用	微服务
整体部署	拆分部署
紧耦合	松耦合
基于整个系统的扩展	基于独立服务，按需扩展
集中式管理	分布式管理
应用无依赖关系管理	微服务间较强的依赖关系管理
局部修改，整体更新	局部修改，局部更新
故障全局性	故障隔离，非全局
代码不易理解难维护	代码易于理解维护
开发效率低	开发效率高
资源利用率低	资源利用率高
重，慢	轻，快
部署简单	部署复杂

单体应用 VS 微服务

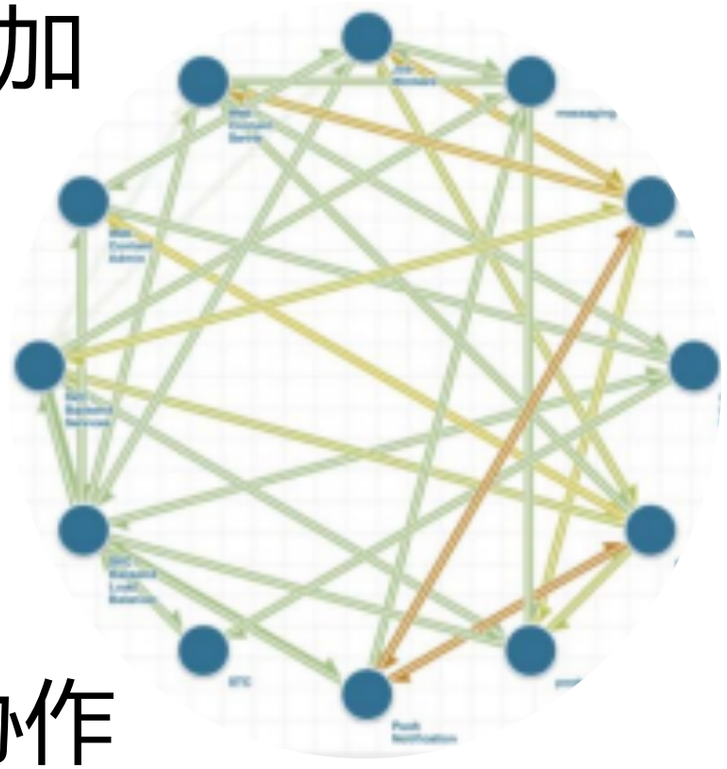
- 微服务

- 单体应用

微服务运维的复杂度

依赖关系

数量增加



运行环境

技术栈多

资源分配

调用协作

监控管理

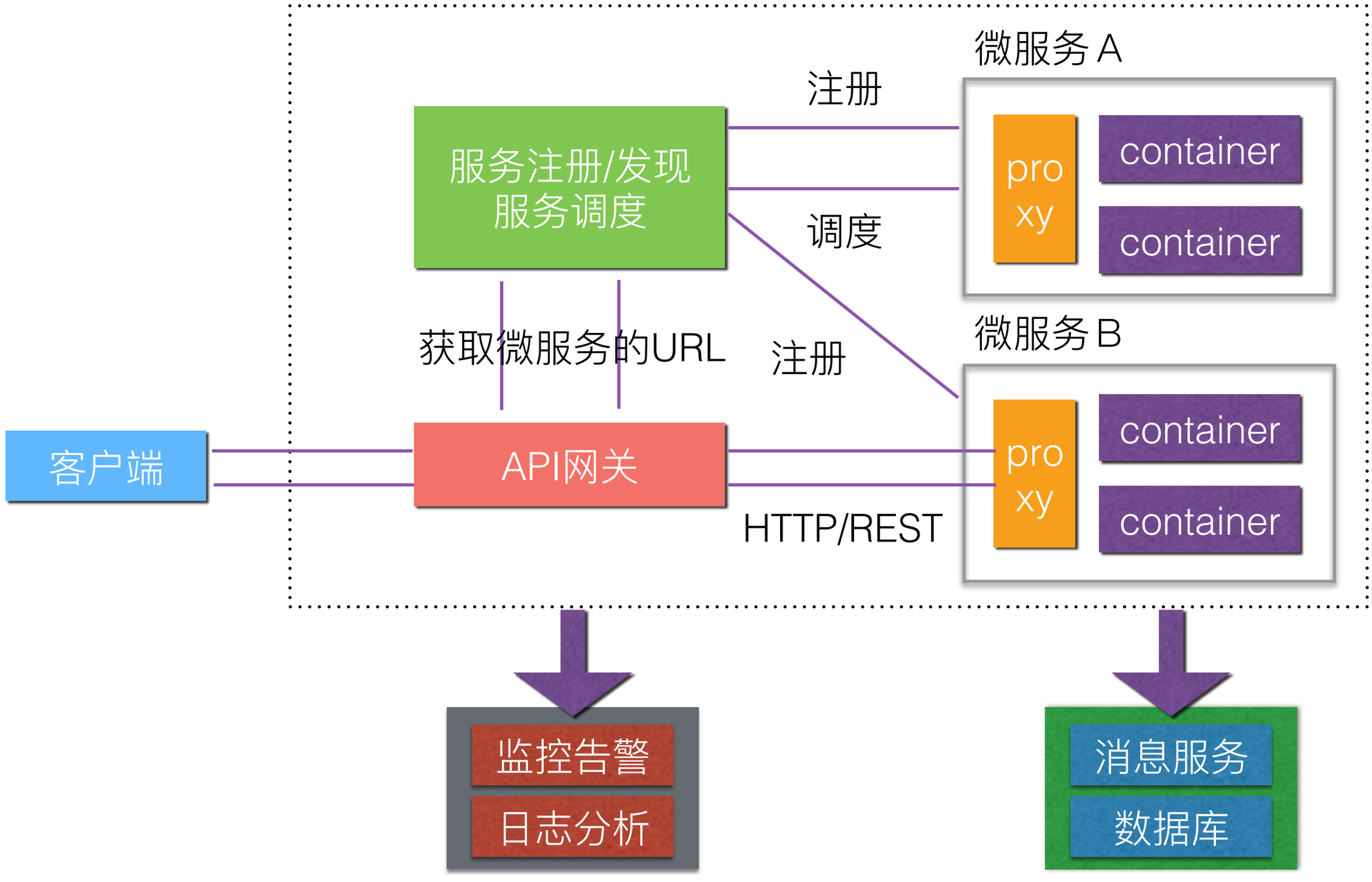
基于容器平台简化运维复杂度

- 微服务的标准化构建及交付
  - 基于容器封装环境
  - 持续集成/持续交付
- 微服务的自动化管理
  - 服务发现与注册
  - 依赖关系管理
- 微服务的自动化运维
  - 基于服务状态的弹性伸缩
  - 服务的监控及故障自动处理

基于容器构建微服务架构

QINIU

典型的微服务架构


微服务 A
注册
服务注册/发现
服务调度
pro
xy
container
container
调度
微服务 B
获取微服务的URL
注册
客户端
API网关
pro
xy
container
container
HTTP/REST
监控告警
日志分析
消息服务
数据库

构建一体化的DevOps平台

开发

版本控制和协作
构建和测试自动化
持续集成和交付

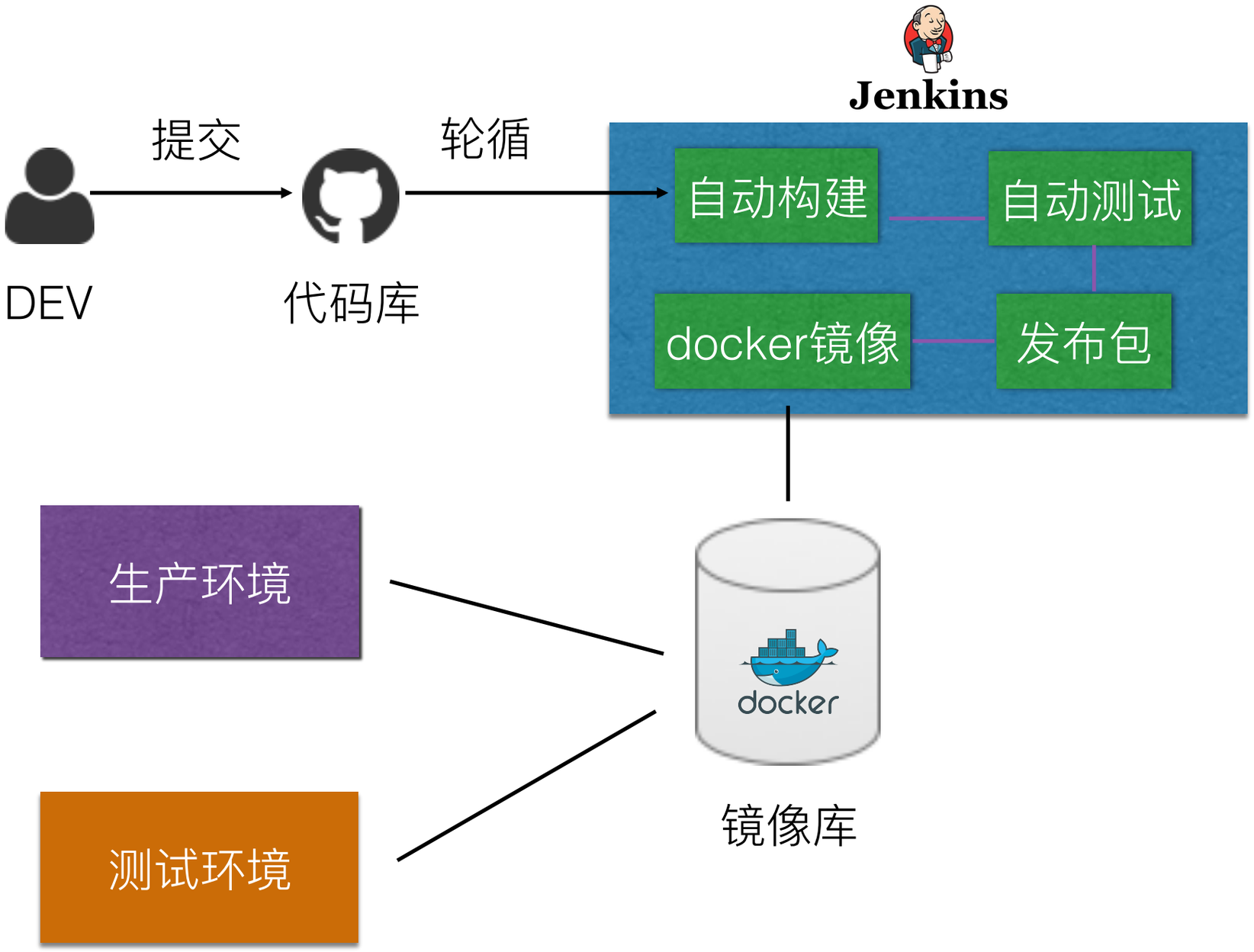
部署

配置管理
微服务平台
服务开通

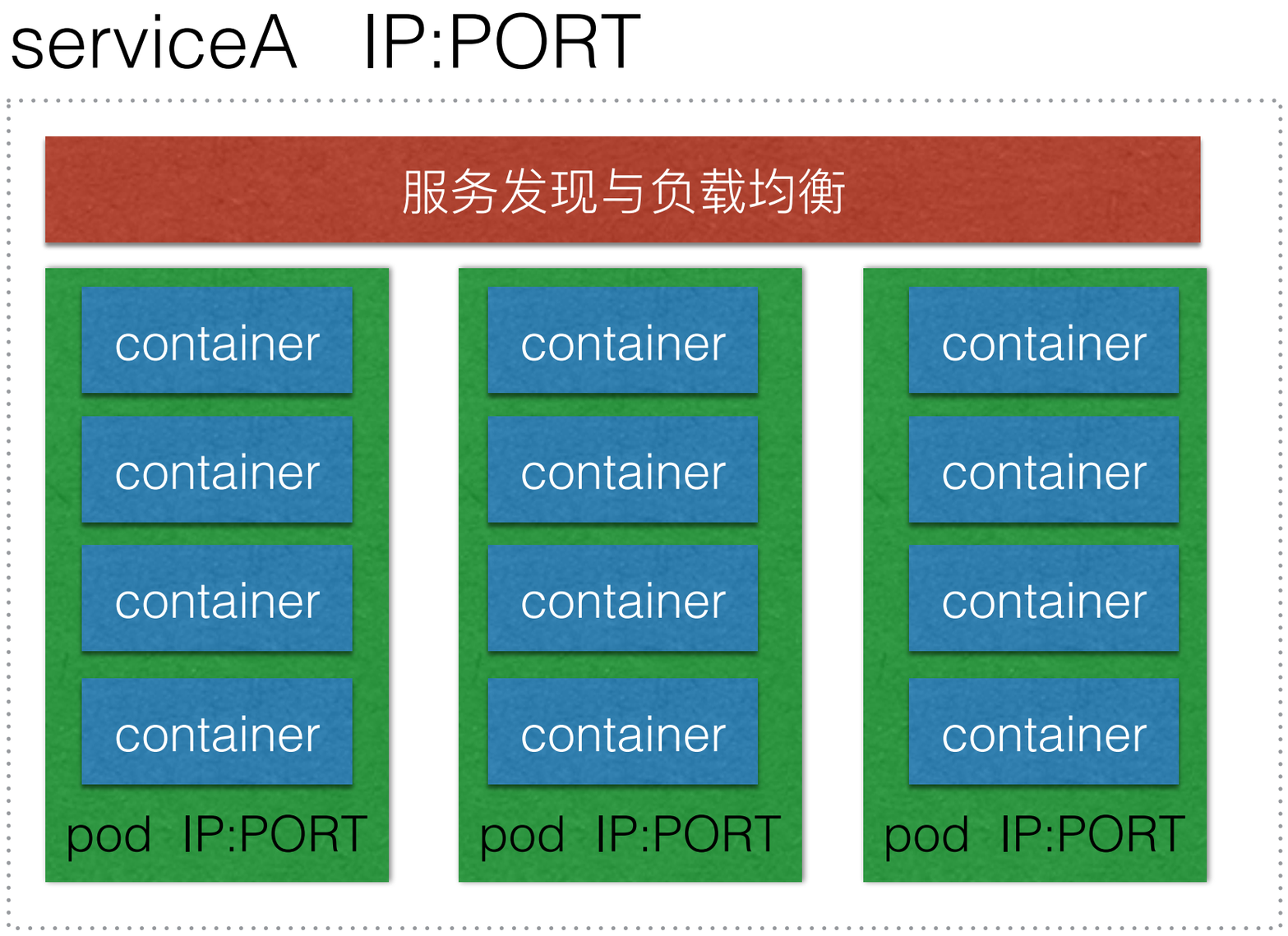
维护

日志管理
监控告警和分析

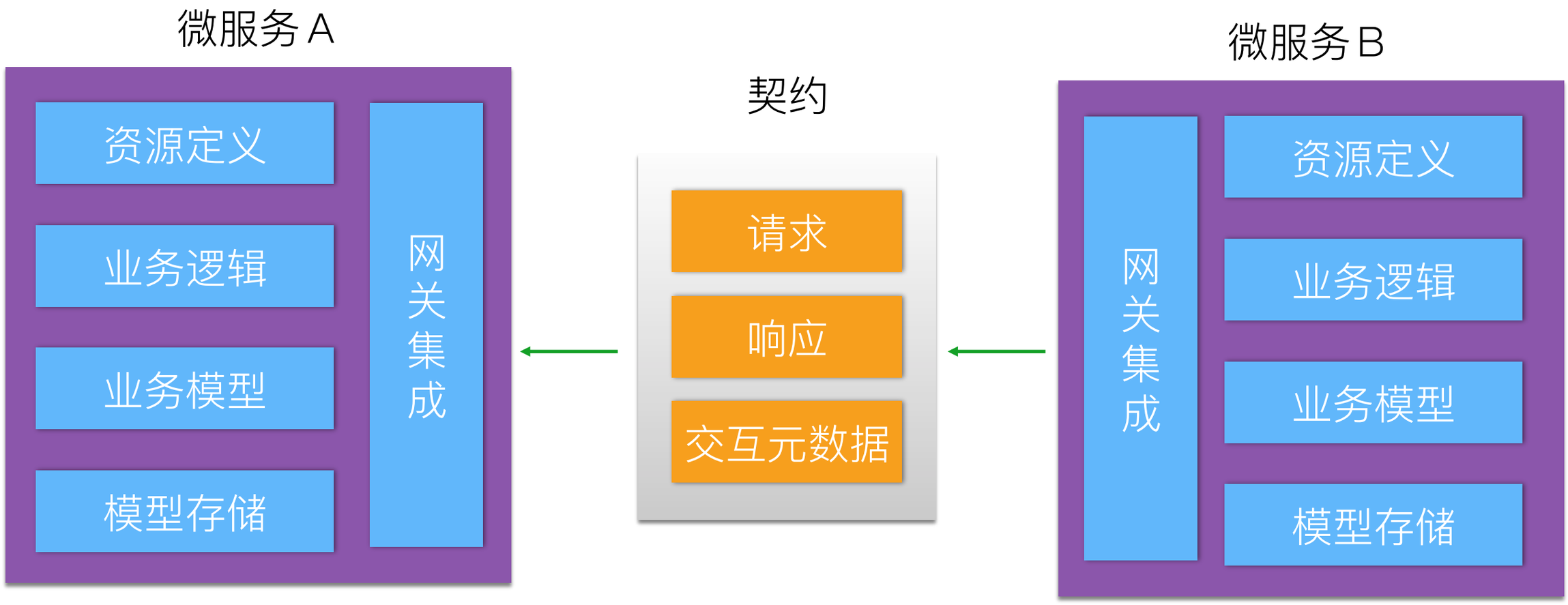
持续集成与持续发布


Jenkins
提交
轮循
自动构建
自动测试
docker镜像
发布包
DEV
代码库
生产环境
测试环境
docker
镜像库

服务发现与负载均衡

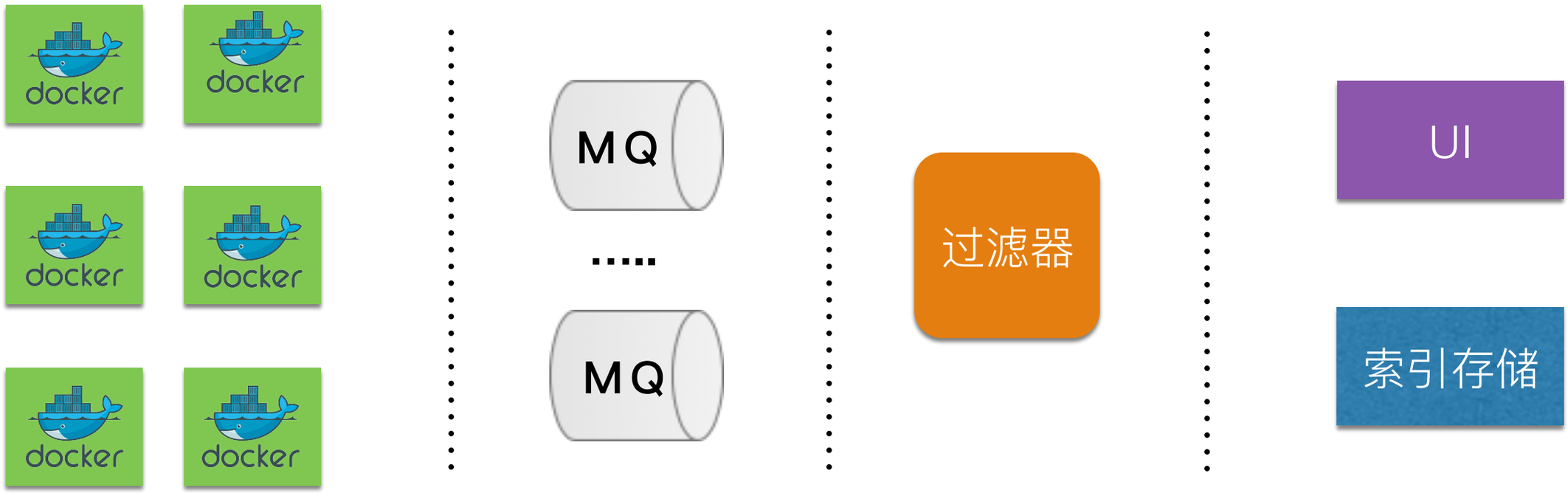

serviceA IP:PORT
服务发现与负载均衡
container
container
container
container
pod IP:PORT
container
container
container
container
pod IP:PORT
container
container
container
container
pod IP:PORT

依赖关系管理



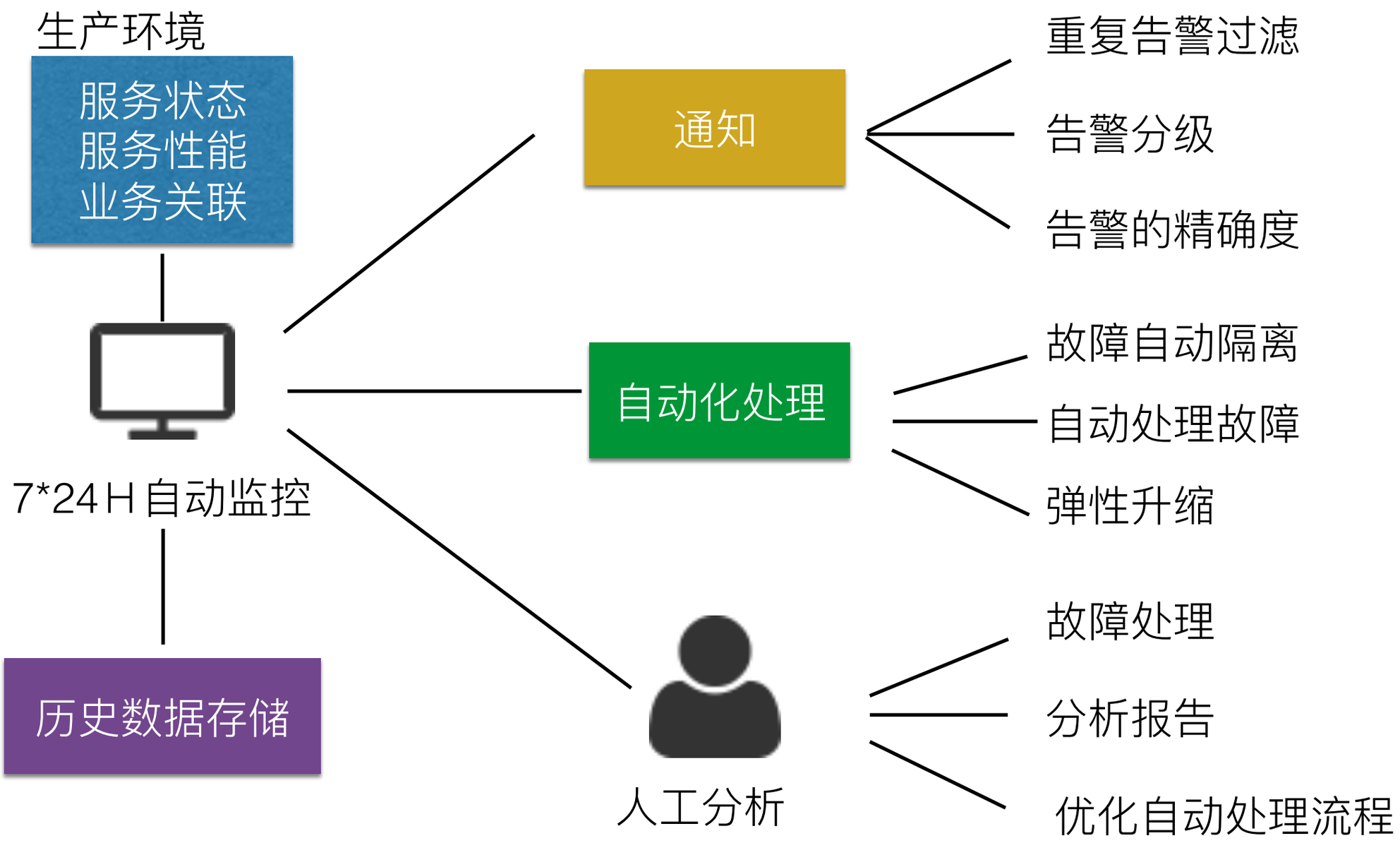
基于消费者驱动的契约管理

日志集中式管理

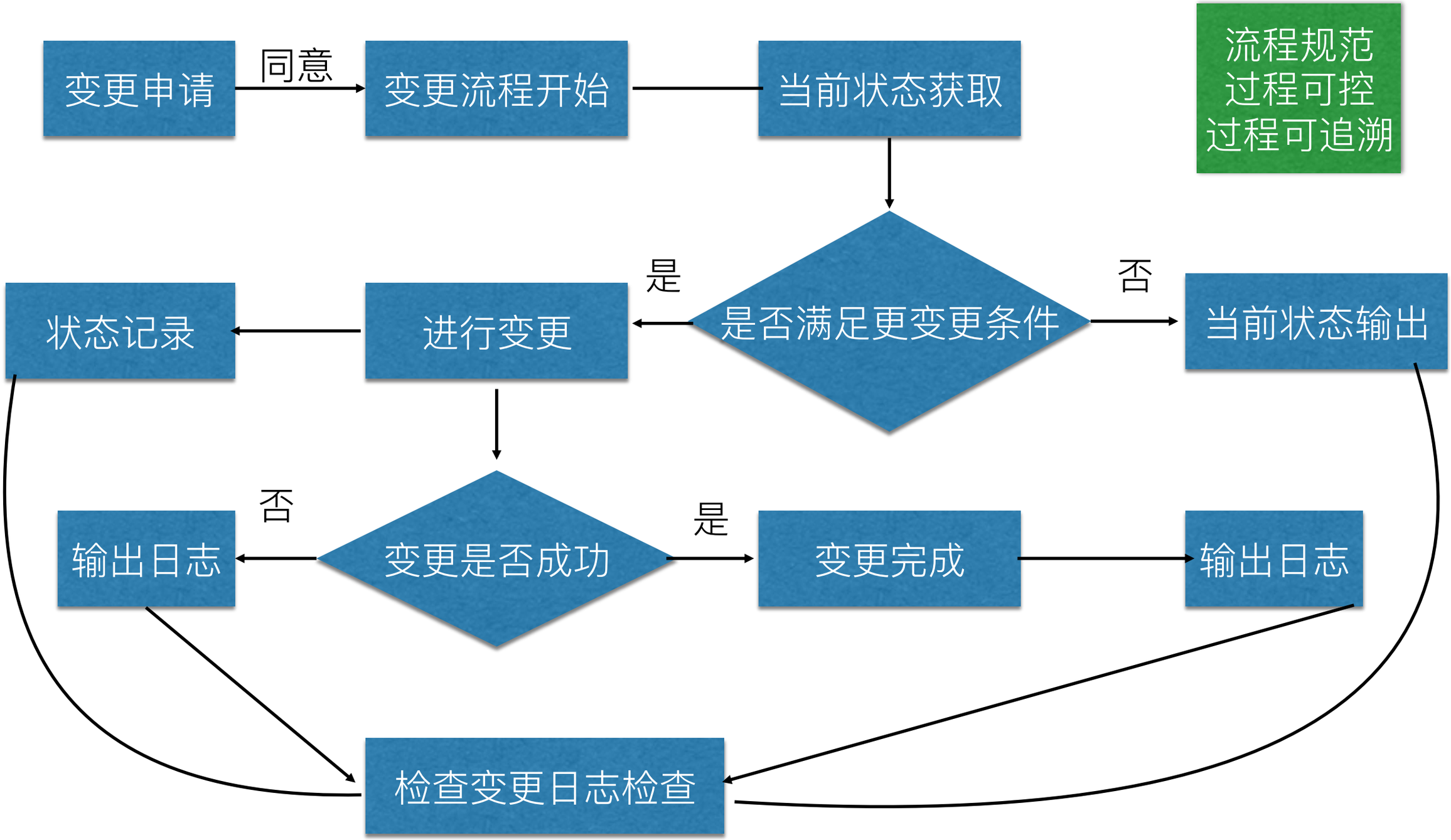


分布式节点 → 消息队列 → 过滤器 → 索引,查询

监控管理


生产环境
服务状态
服务性能
业务关联
7*24H自动监控
历史数据存储
通知
重复告警过滤
告警分级
告警的精确度
自动化处理
故障自动隔离
自动处理故障
弹性升缩
人工分析
故障处理
分析报告
优化自动处理流程

变更管理

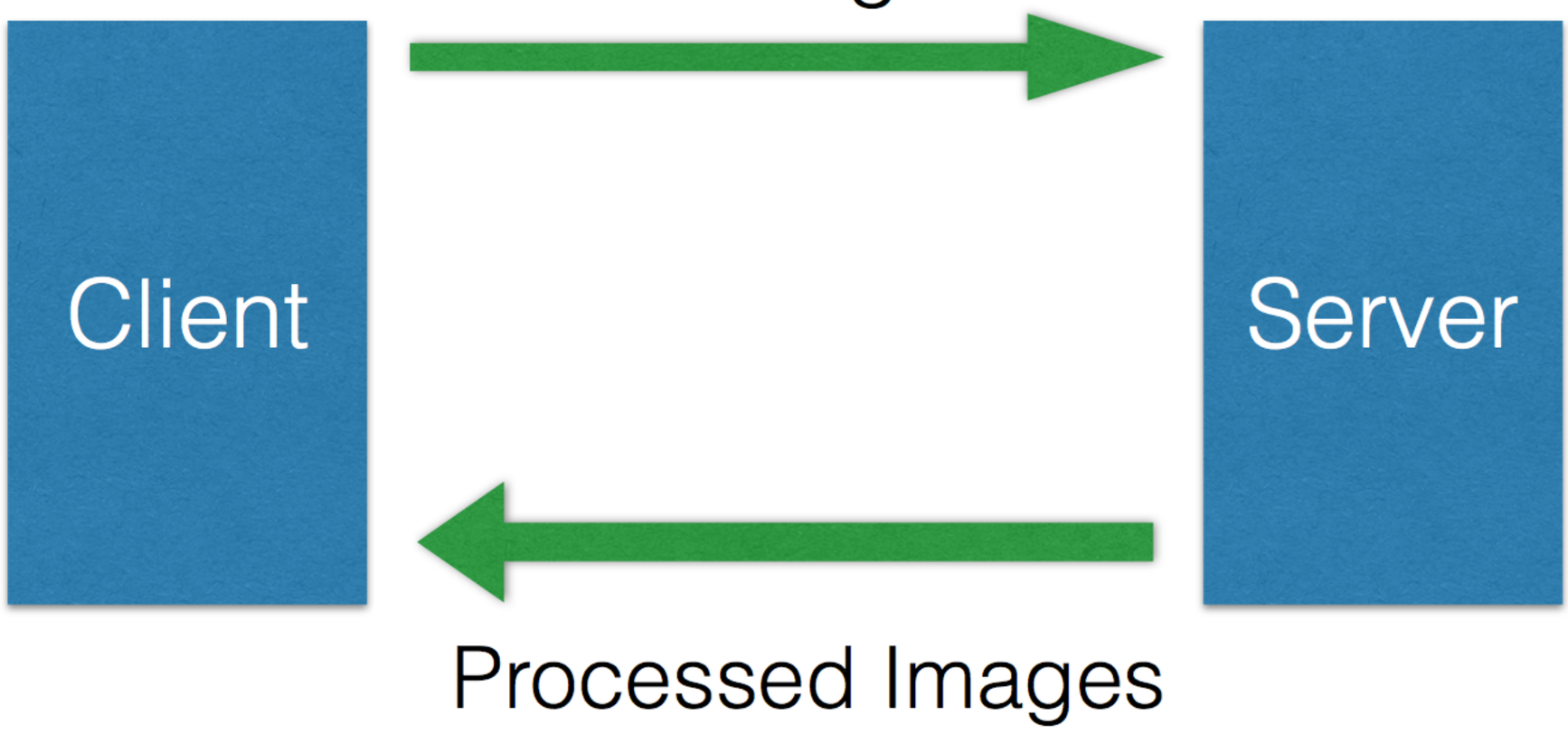

变更申请
同意
变更流程开始
当前状态获取
流程规范
过程可控
过程可追溯
是否满足更变更条件
是
否
当前状态输出
进行变更
状态记录
变更是否成功
否
是
输出日志
变更完成
输出日志
检查变更日志检查

七牛的微服务架构实践

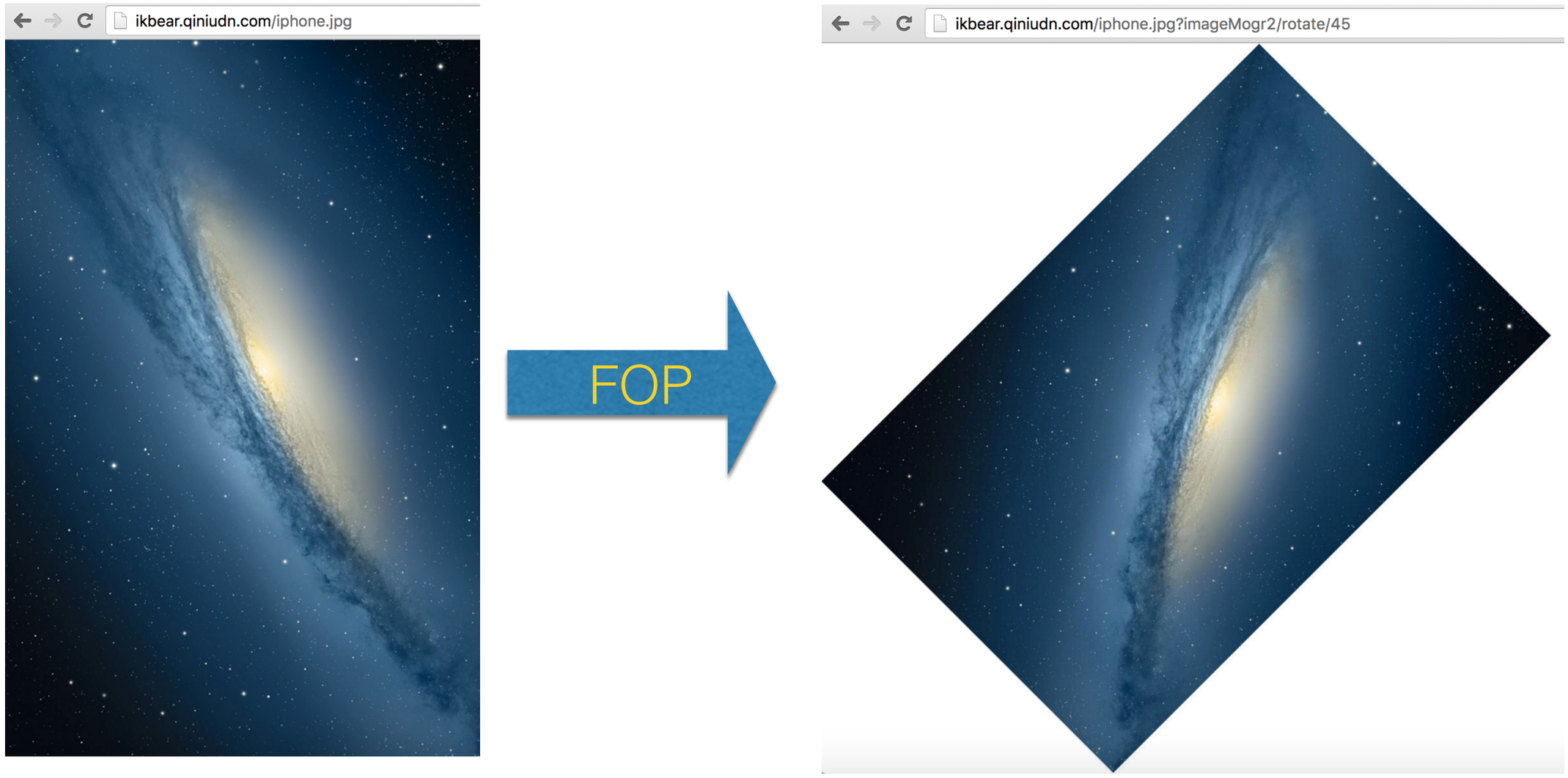
QINIU

文件处理服务

FOP: File Operation

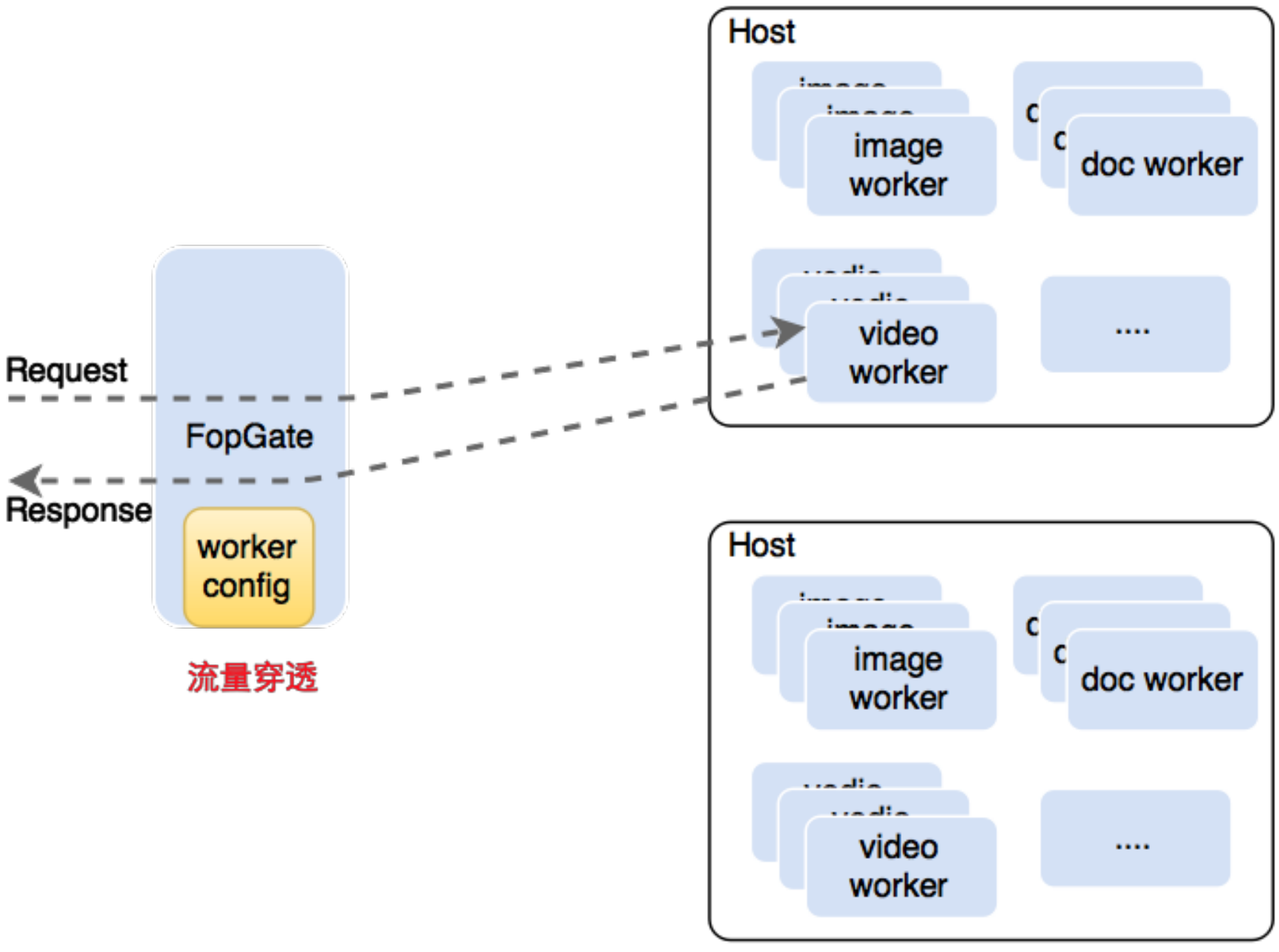


文件处理服务演示



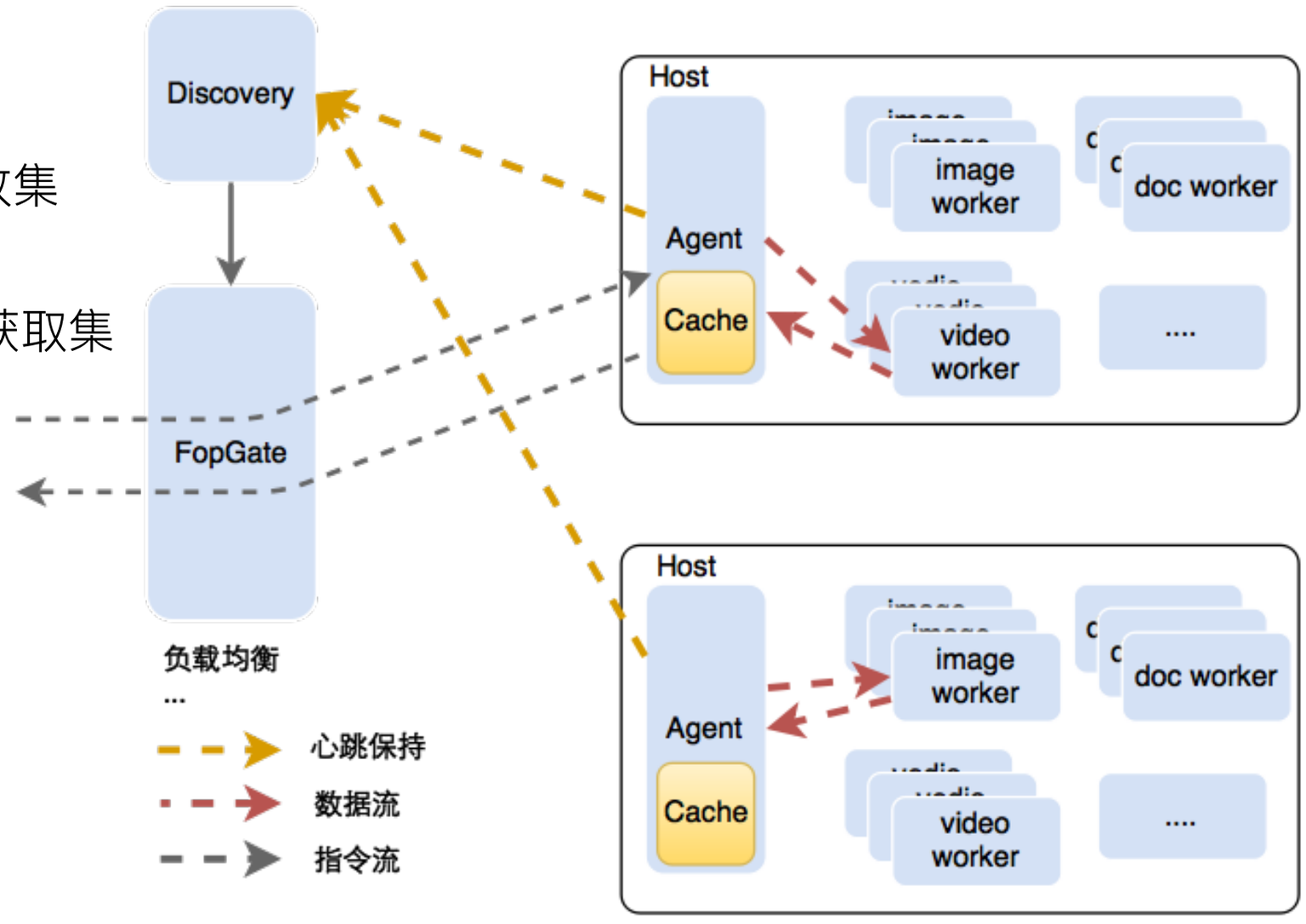
http://ikbear.qiniudn.com/iphone.jpg?imageMogr2/rotate/45

文件处理架构演变stage1



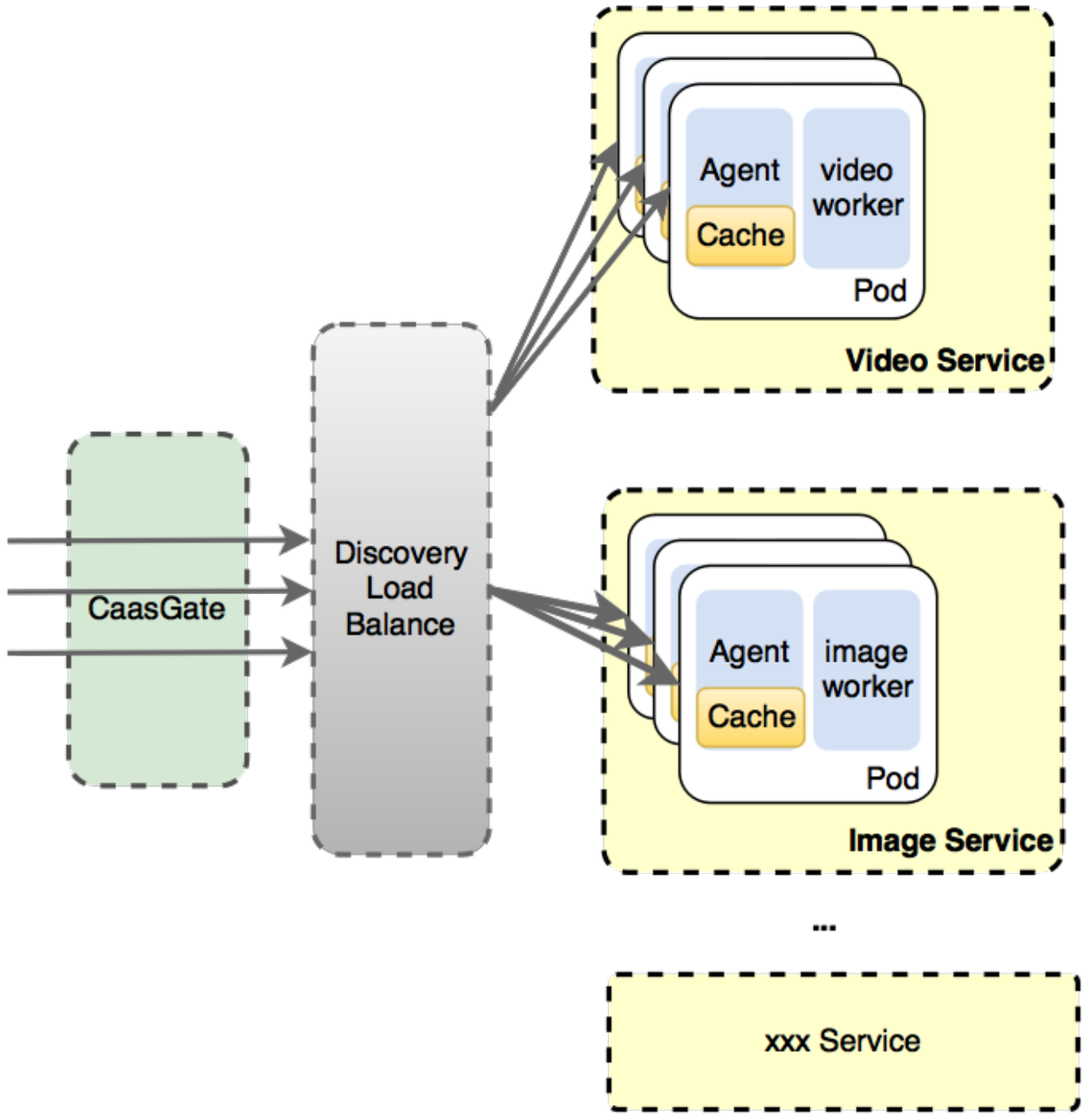
文件处理架构演变stage2

- 增加 Discovery 组件，收集 Agent 上报信息
- FopGate 从 Discovery 获取集群信息，做 LB
- 增加业务 Agent
  - 上报后端信息
  - 上报保活信息
  - 单机内 worker LB



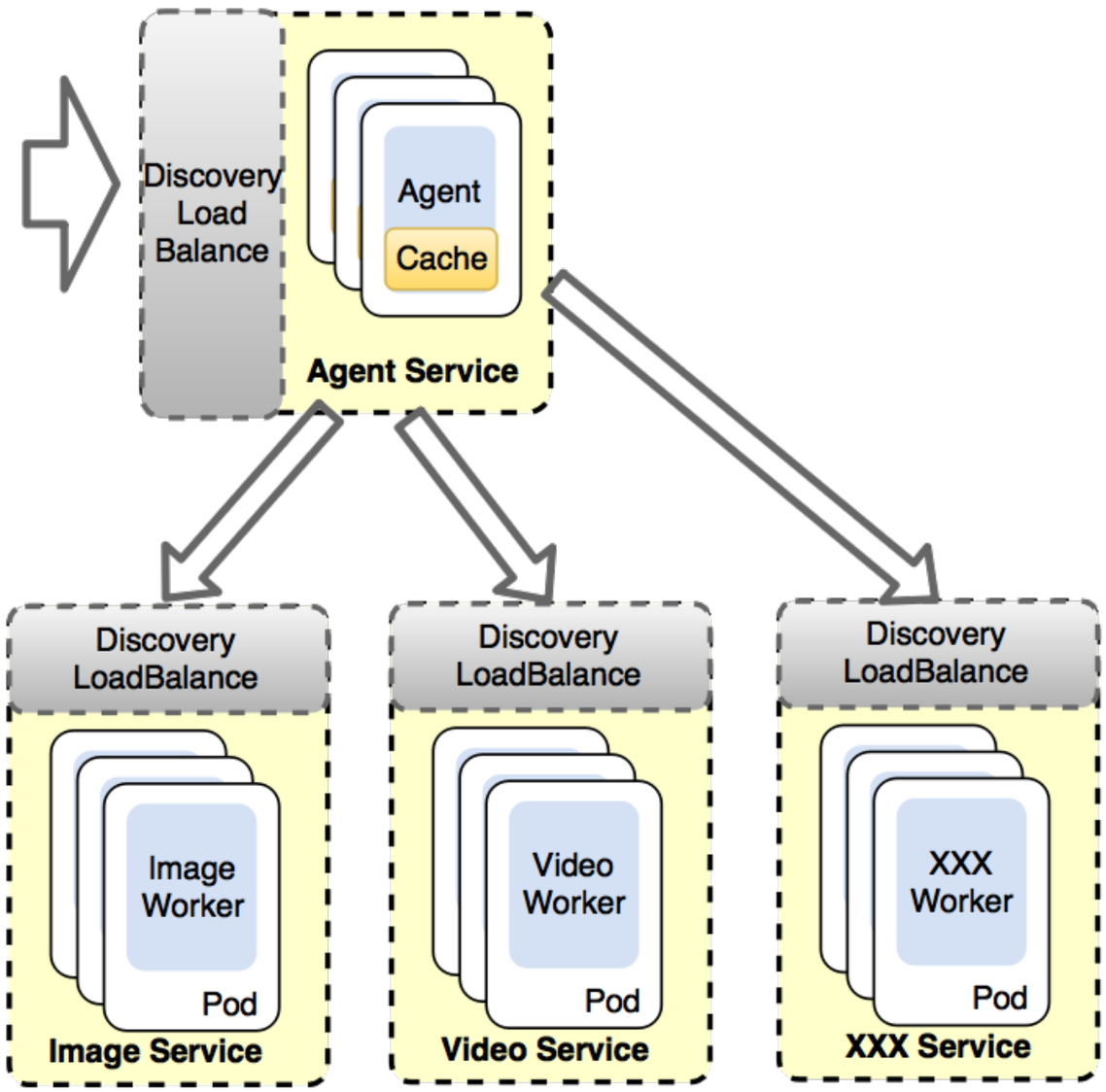
文件处理架构演变stage3

- 取消业务层 Discovery 服务
- 业务 Agent 功能退化
  - 无需与 Discovery 心跳保持
  - 简化 Agent 后端，Agent 无需 LB
- FopGate 取消 LB 逻辑



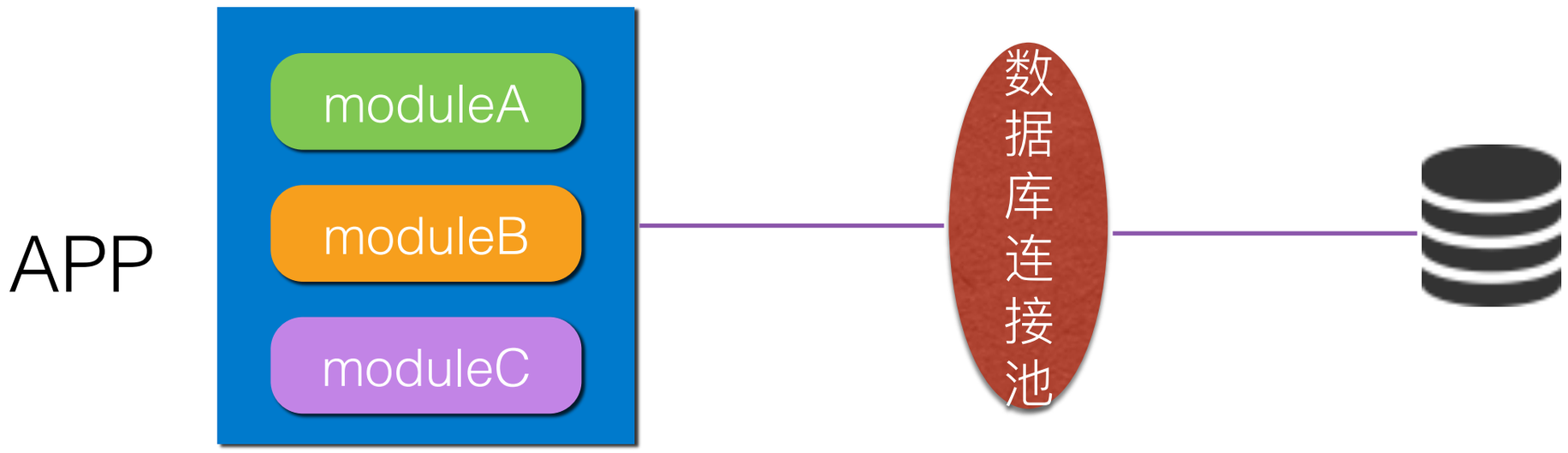
文件处理架构演变stage4

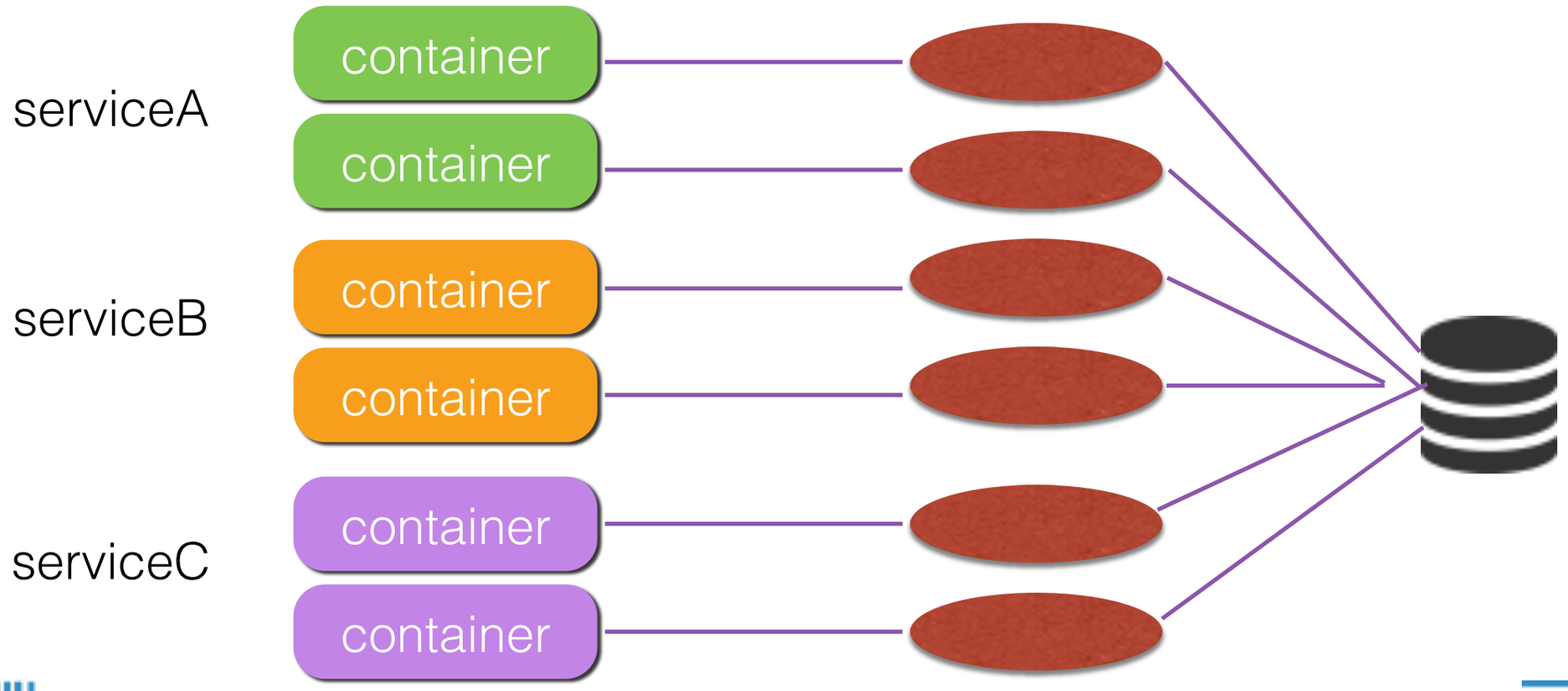
- 进一步分离
  - 有状态服务 Agent
  - 无状态服务 worker
- 通过调度调配 worker 数量



踩过的那些坑

数据库连接风暴

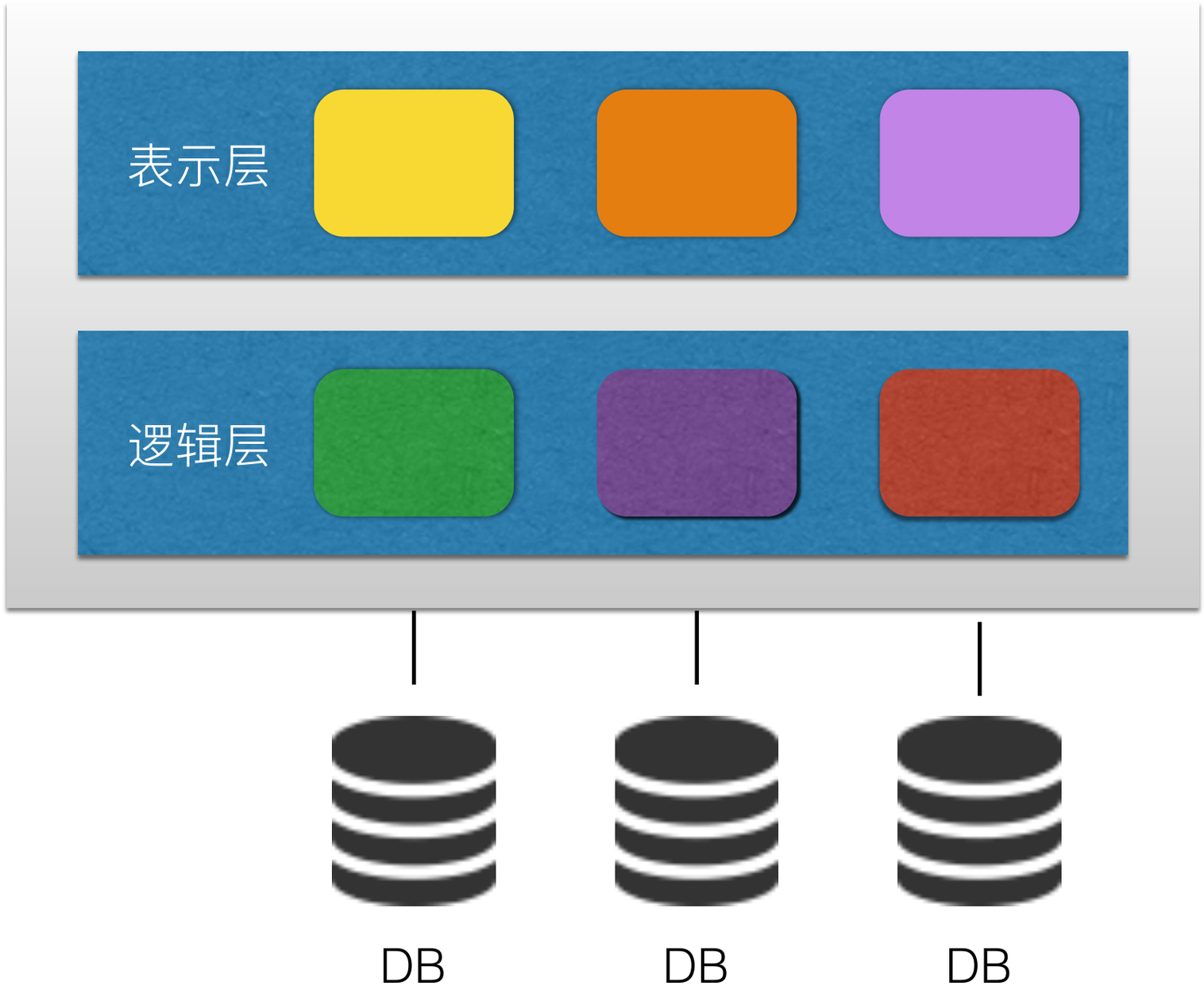




QINIU

数据库连接风暴暂解办法

如何将单体应用改造成微服务



微服务设计的准则

- ✓ 更多、更小的最小化功能微服务，不能再划分
- ✓ 每个微服务可以单独部署，有自己一整套的完整的运行机制，有和外部通讯的标准化接口。
- ✓ 每个微服务都有自己的数据存储
- ✓ 基于REST的调用，消息或者二进制数据，异步通信
- ✓ 状态信息是在分布式数据网格– 实例是无状态的
- ✓ 每个微服务都可以选择不同的技术框架
- ✓ 每个微服务异步扔出的消息可以被其他微服务所消费

Thank
you

Q&A