QINIU

架构探险

从支持一个 App 的后端到 BaaS

卜赫 | 2016.6.25

PHPCON
www.phpconchina.com

这是关于一家有理想的创业公司的故事

QINIU

每个创业公司都相信他们在
走一条少有人走的路
并且相信自己可以改变世界

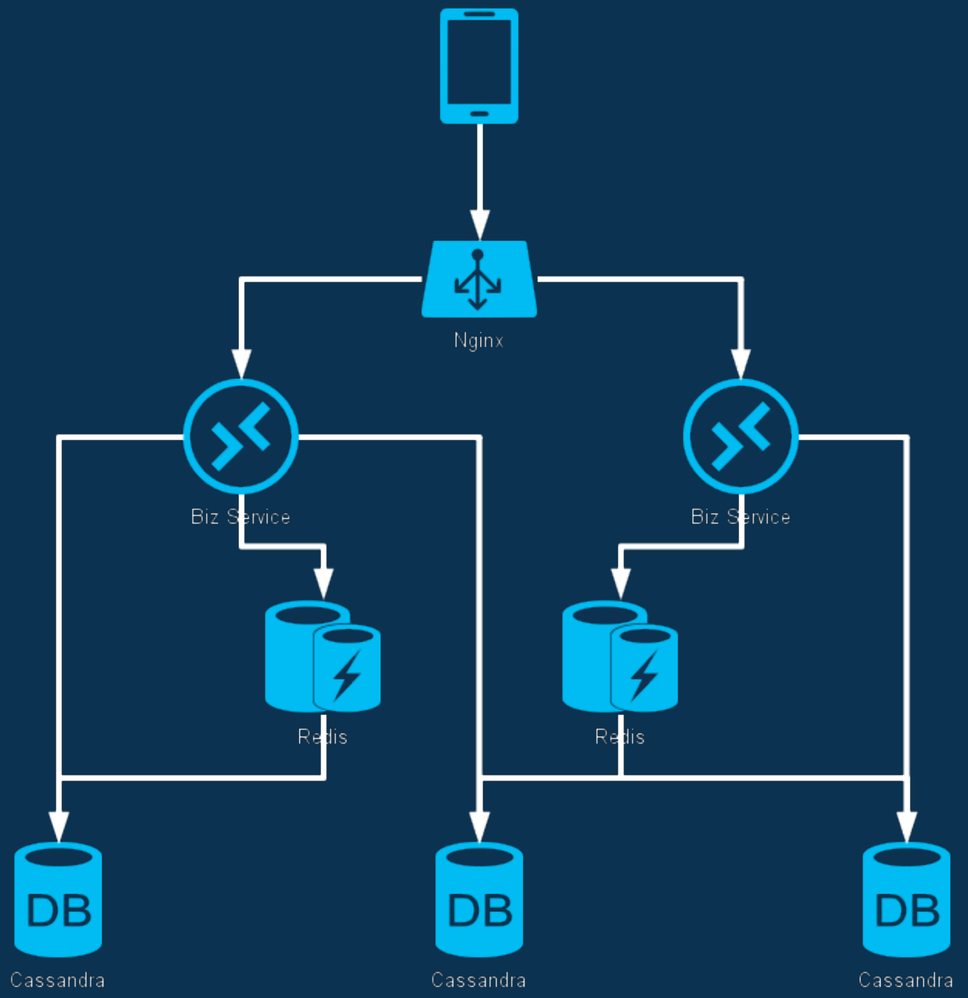
QINIU

App

Backend

支持一个 App 就够了！

QINIU


Nginx
Biz Service
Biz Service
Redis
Redis
DB
DB
DB
Cassandra
Cassandra
Cassandra

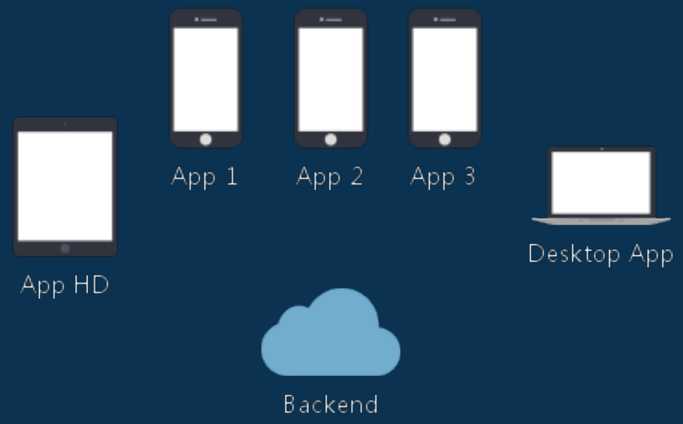
够用就好

- 公司内部有一些盈利的App
- 快速迭代
- NoSQL
- Layer 7 Load Balancing

问题

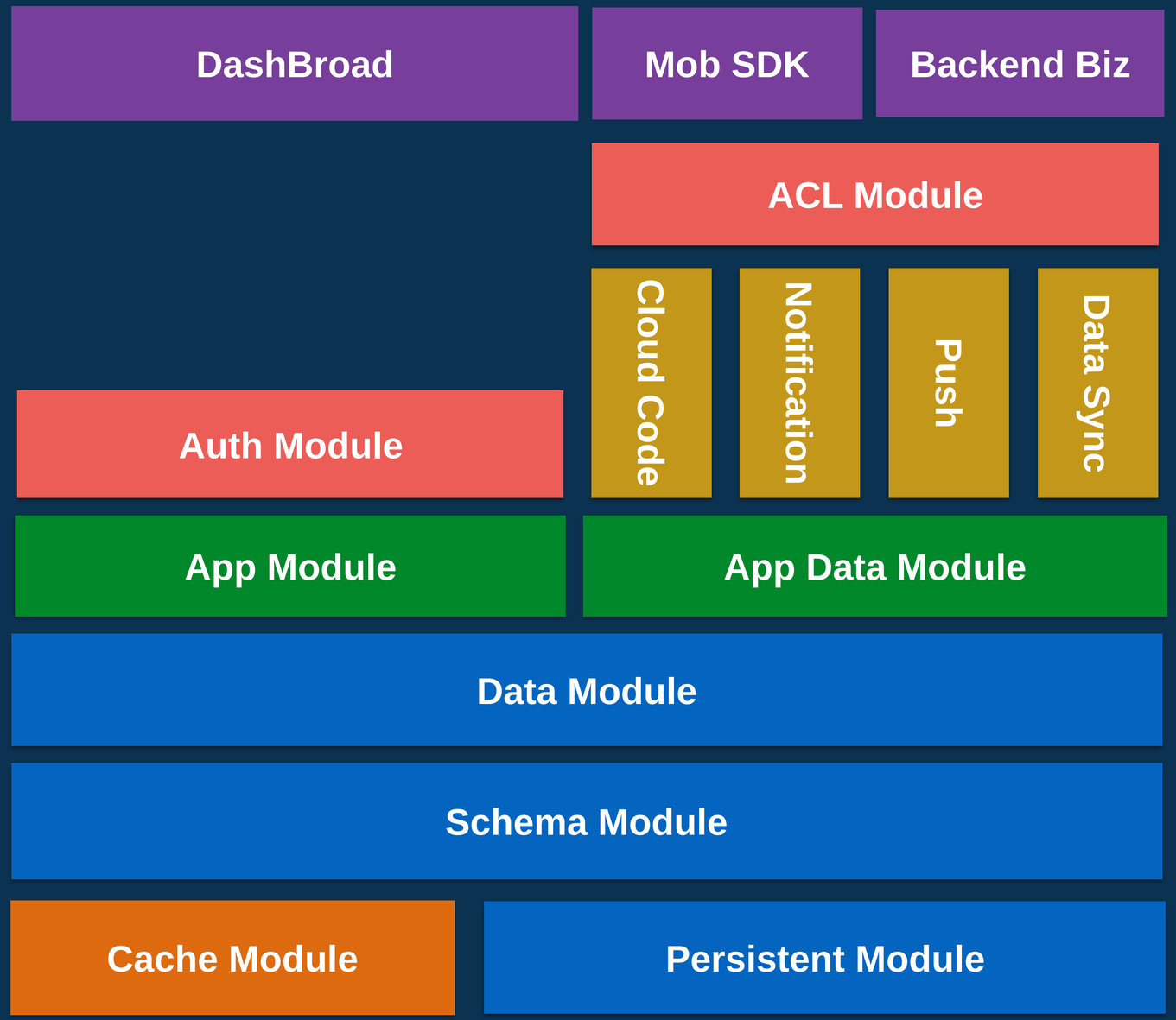
- 支持的 App 逐渐增多，业务代码和基础代码在同一个repo中
- 每次更新都要更新整个项目
- 业务方的需求不断改变，需要经常修改表结构和API
- 有些业务耦合代码需要经常更新（两天一次，修改算法）

面对快速搭建的后端，CTO 决定必须要做出改变



公司内的 App 都接进来！

QINIU


DashBroad
Mob SDK
Backend Biz
ACL Module
Cloud Code
Notification
Push
Data Sync
Auth Module
App Module
App Data Module
Data Module
Schema Module
Cache Module
Persistent Module

内部BaaS

- 模块化，明确业务代码和基础代码的边界
- 支持动态修改表结构
- 支持原始版本的 Cloud Code
- 提供 iOS 和 Android 推送服务
- 提供 IM 和 通知功能
- 提供 Data Sync 服务
- DashBroad

问题

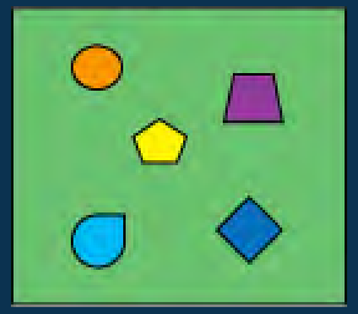
- 依然是单体应用
- Cassandra 的动态性不够好
- Backend 多语言通信需求
- 线上事故，各业务系统互相影响

CEO 觉得自己内部用的不错，也许别人也用的到呢

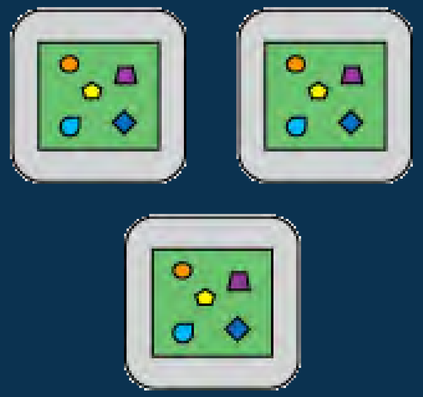
重构为 BaaS

单体应用 VS 微服务

单体应用

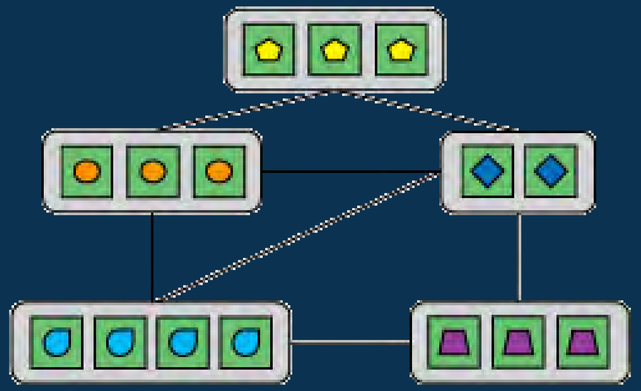


伸缩



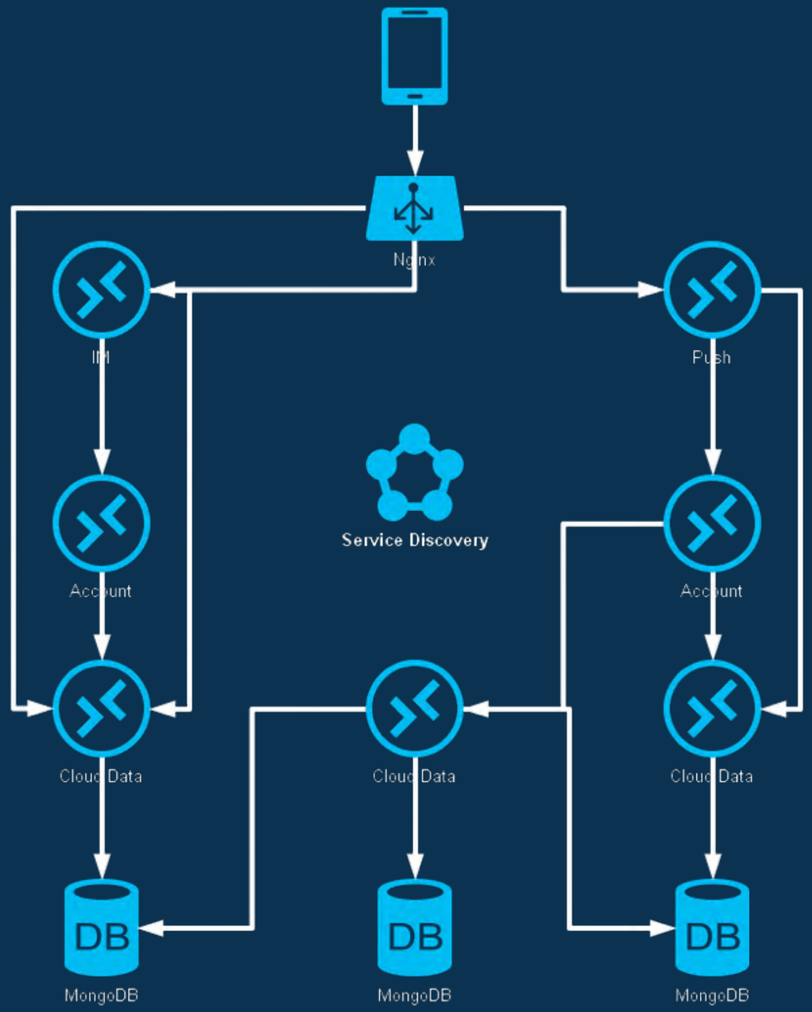
微服务

伸缩

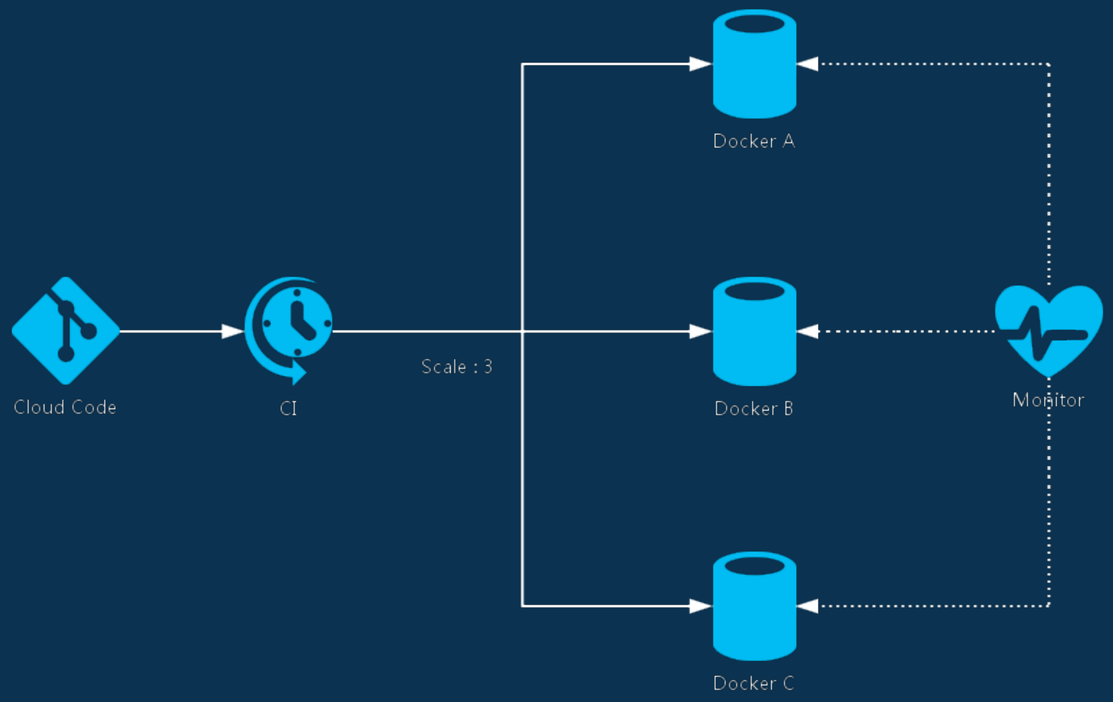


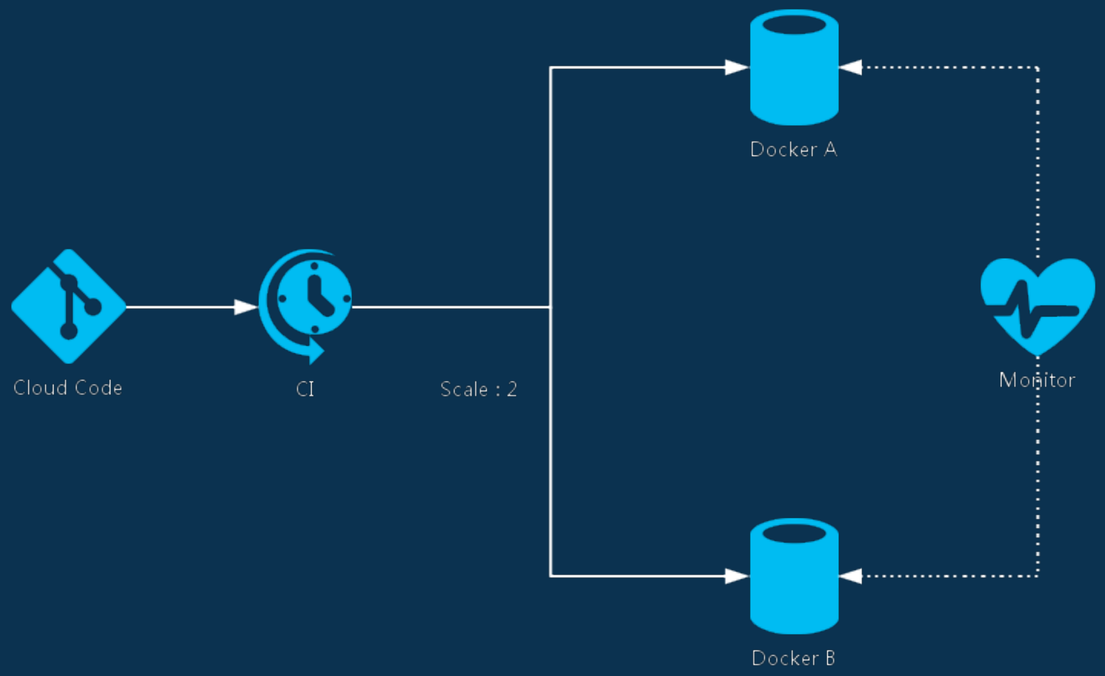
单体应用 VS 微服务

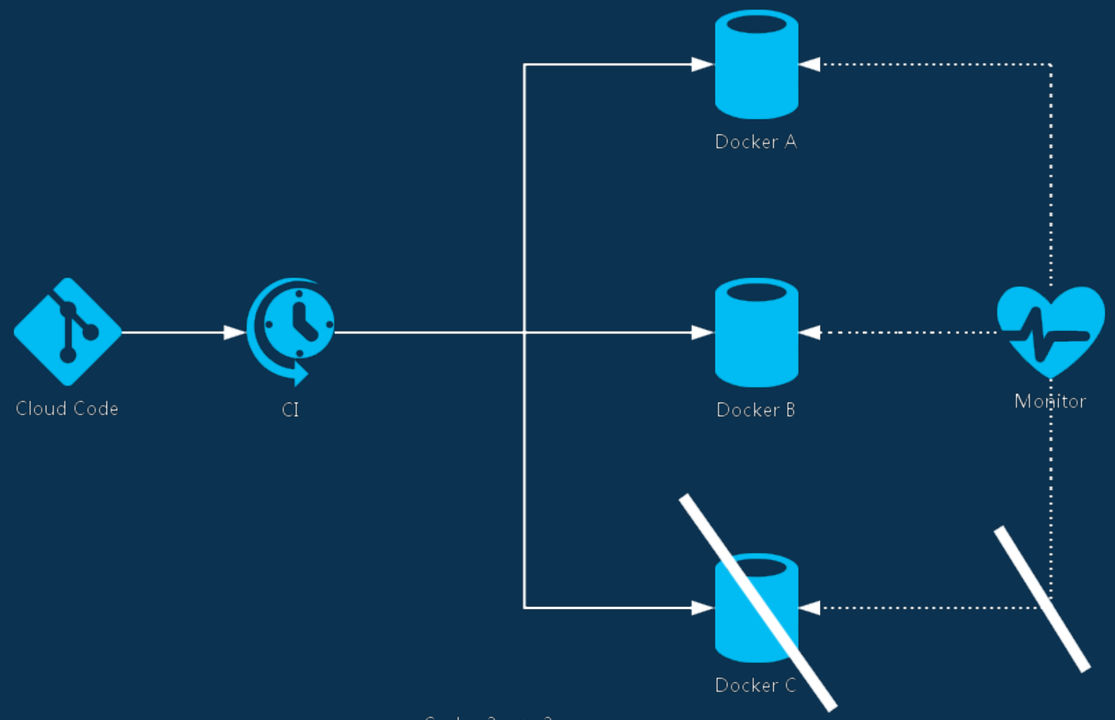
单体应用	微服务
整体部署	拆分部署
紧耦合	松耦合
基于整个系统的扩展	基于独立服务，按需扩展
集中式管理	分布式管理
应用无依赖关系管理	微服务间较强的依赖关系管理
局部修改，整体更新	局部修改，局部更新
故障全局性	故障隔离，非全局
代码不易理解难维护	代码易于理解维护
开发效率低	开发效率高
资源利用率低	资源利用率高
重，慢	轻，快
部署简单	部署复杂


Nginx
IM
Push
Service Discovery
Account
Account
Cloud Data
Cloud Data
Cloud Data
DB
DB
DB
MongoDB
MongoDB
MongoDB

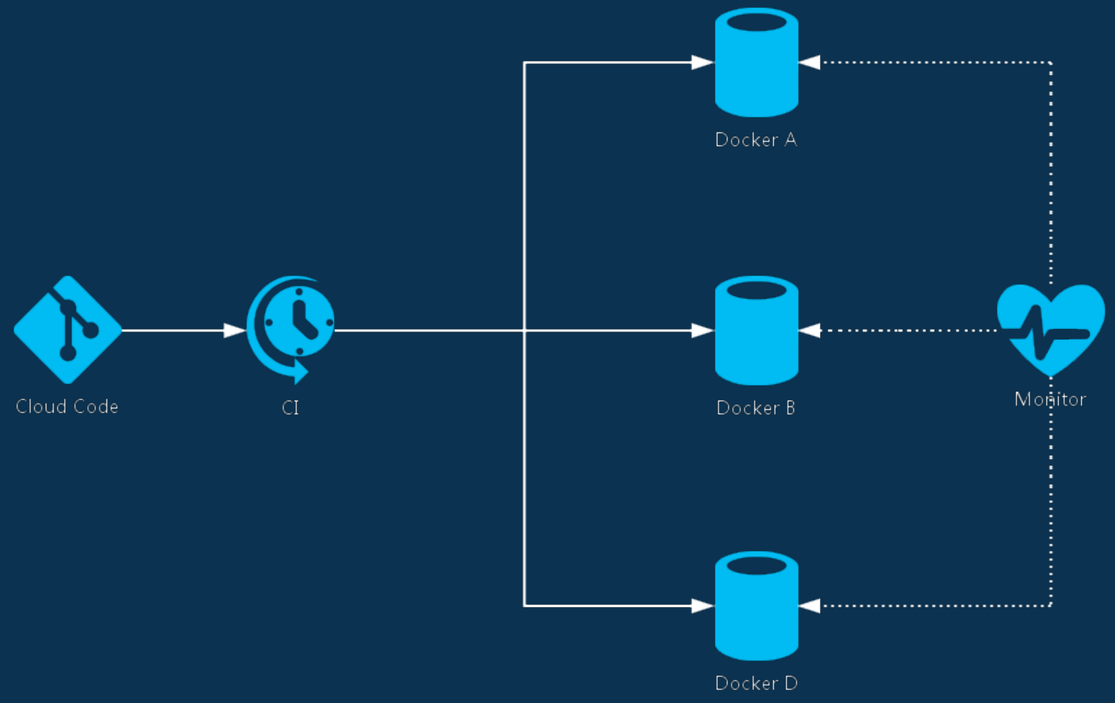
基于容器的 Cloud Code


Docker A
Cloud Code
CI
Scale : 3
Docker B
Monitor
Docker C


Docker A
Cloud Code
CI
Scale : 2
Monitor
Docker B



Scale : 3 --> 2



Scale : 2 --> 3

服务化，容器化

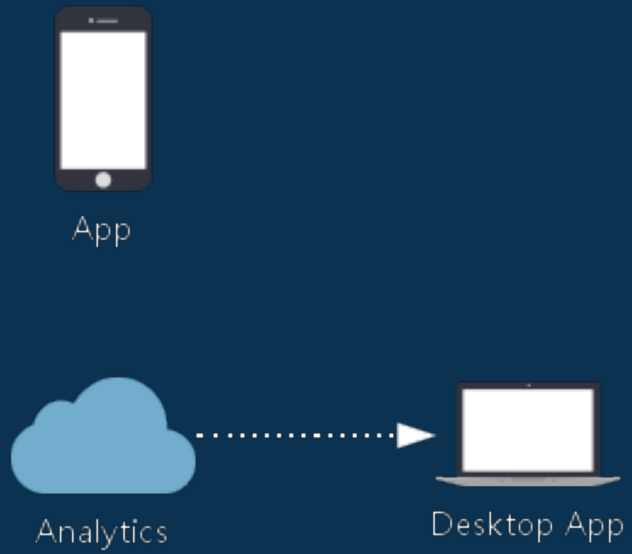
- 拆分成细粒度服务
- 通过虚拟 EventBus 通信
- Cassandra -> MongoDB
- 基于容器的 Cloud Code

问题

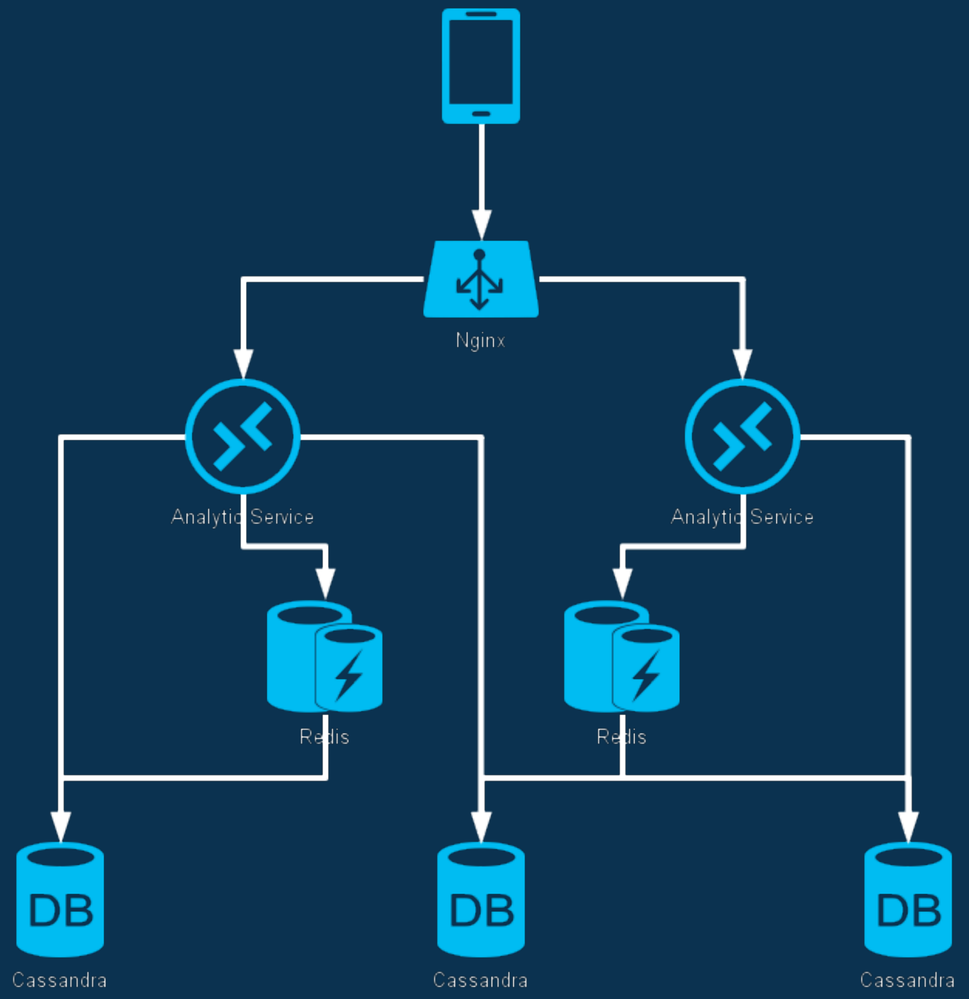
- 整个应用没办法做到 auto-scaling
- 自动化部署还不够完善，人工介入过多

CEO 说，我要知道我们到底多受欢迎

QINIU



我们先分析这个 APP 的数据！


Nginx
Analytic Service
Analytic Service
Redis
Redis
DB
DB
DB
Cassandra
Cassandra
Cassandra

统计一些自己关心的信息

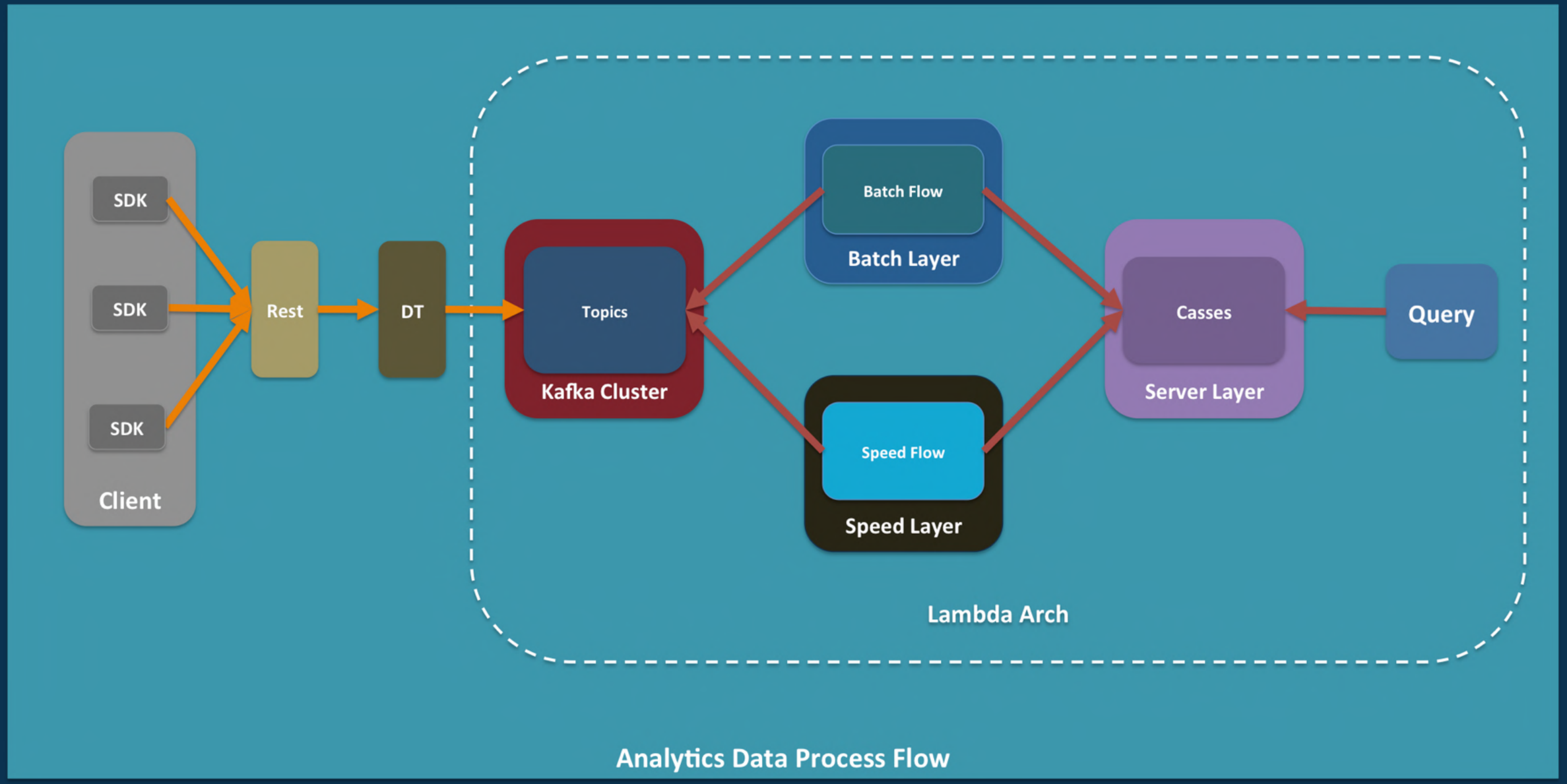
- 日活
- 日新增
- Session 时长
- IAP 数据统计

问题

- 强依赖 Redis
- 难于水平扩展
- 原始数据没有落地，一旦算法错误没有办法重新计算
- 基于如此薄弱的技术基础难于实现复杂算法
- 无法应对大数据量

CTO 说，我们的数据量太大了

重构数据分析平台



Analytics Data Process Flow

数据分析 Lambda

- 采用 Lambda 架构
- 原始数据快速落地，不用担心算法错误，或计算引擎出问题造成不可挽回的损失
- 数据只 CR 不 UD ，保证数据的可重复计算 (Lambda)
- 基于改造的 Cassandra 存储海量结构化数据

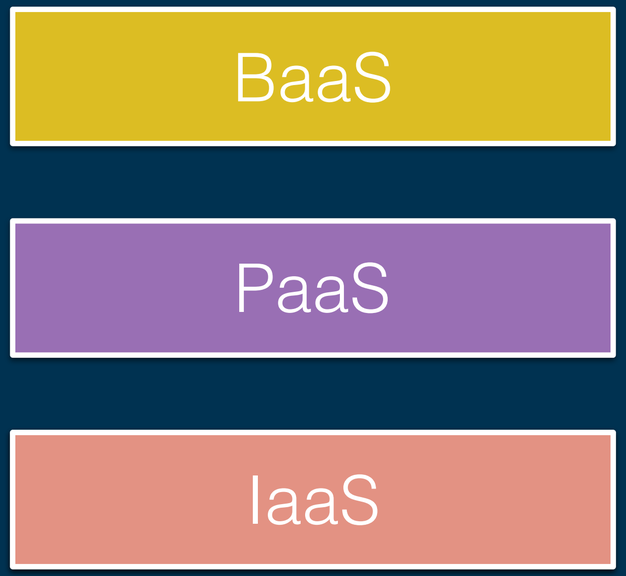
问题

- 不能 auto-scaling
- 离线数据分析时延长
- 闲时资源利用率低

微服务 + Lambda = 终点？

其实我们才上路...

微服务 + DCOS + SDN



provision backend services

deploy and manage app/services

provision and manage machine resources

Author

buhe

github.com/buhe

wechat: 81128054

email: buhe@qiniu.com

focus: DCOS,Data,full-stack