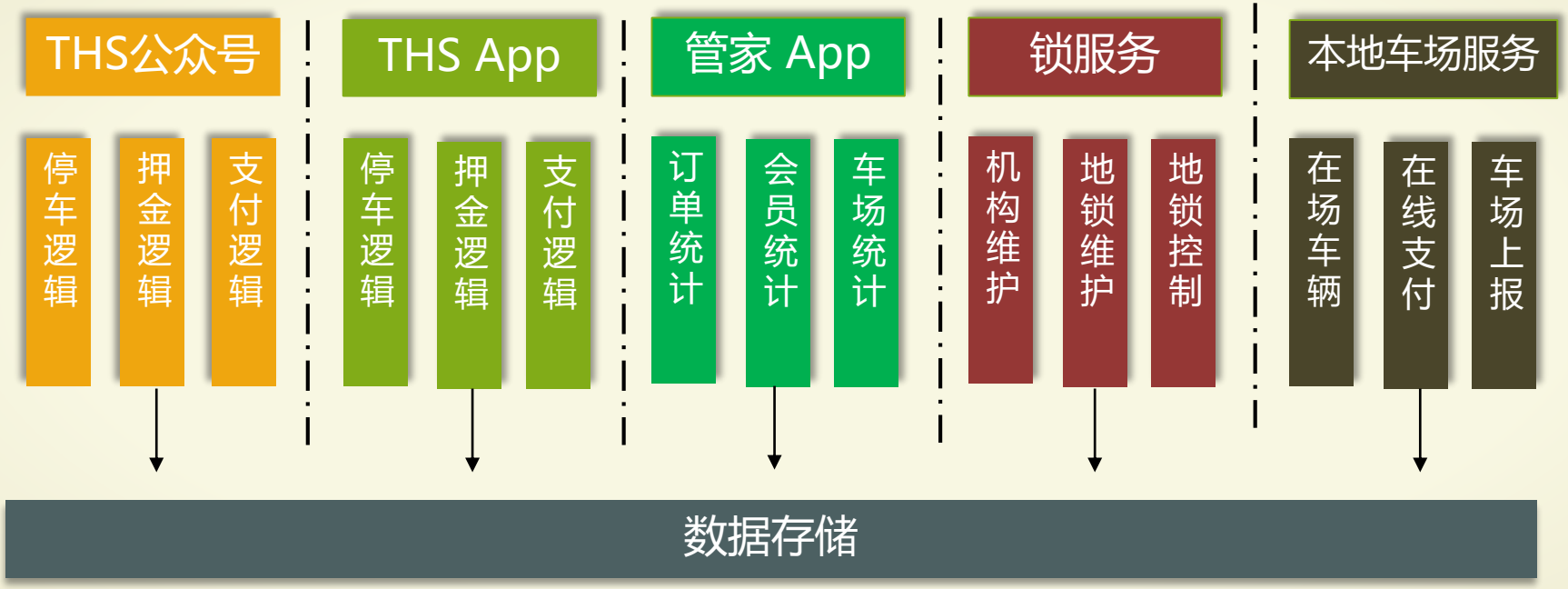
技术中心服务架构规划

技术中心-夏集龙
2018-02-02

目录

我们的应用



模式简单，各司其职

在初期可以很好完成业务迭代

但为满足业务会累加重复功能

当需求变更时，维护难度大，测试难度大

若还要加小程序呢?

是否要合呢？怎么合?

面临的问题

当服务架构是单体应用时，随着业务的发展，会变成一个一个巨无霸式的应用

系统迭代速度慢，开发、测试和运维效率低

整体架构耦合度高，开发难度大，而且容易出错

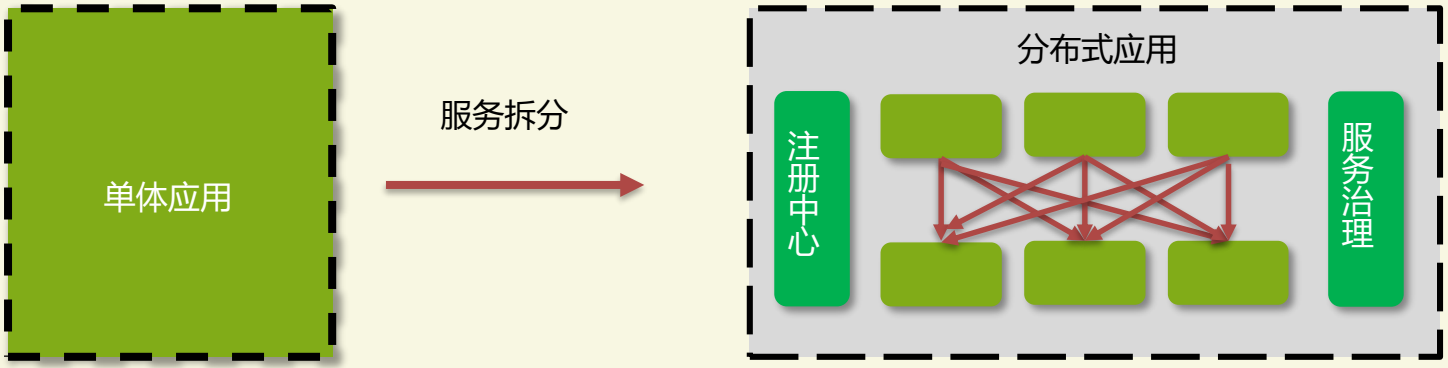
无法满足用户需求变化，无法支撑突发的爆发式访问，扩展能力低

避免误区


大公司方案
应该是借鉴，而不是一味照搬大公司的架构方案，否则会迷路
过度设计
需要低成本试错。当今技术发展日新月异，减少为很长时间内不存在的问题进行设计
为了技术而技术
只在合适的地方用合适的方法不能脱离业务的实际发展需要
想用技术解决所有问题
技术是辅助，通过调整业务模式来解决问题

目录

应用架构改造



引入分布式服务

负载均衡

从多个提供相同服务的提供方中选择一个进行调用

高可用无单点

减少宕机时间，保持服务的高度可用性，4个9服务

整合异构系统

采用RESTFUL API，不同系统可以快速集成

独立开发部署

每一个服务独立开发和部署，不影响其他功能的稳定

容错-重试-恢复

高可用组件要容忍其依赖组件的失败。采用熔断机制，快速失败

版本发布

蓝绿发布、滚动发布、灰度发布

服务发现

负责服务的注册与发现。业务服务快速集成到平台中

公共契约

跟业务相关，约定大于配置，自动生成API文档

服务监控

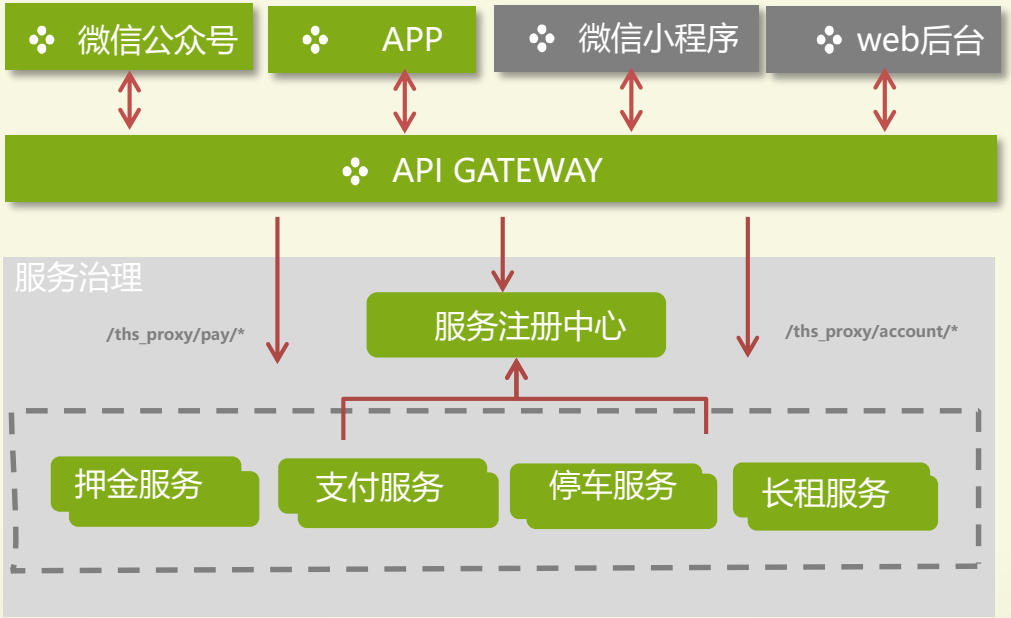
日志审计、服务调用链、调用失败统计。运维快速介入

分布式服务平台


外网用户
第三方server
API GATEWAY
全链跟踪
生命周期
版本控制
基础监控
告警管理
服务治理
/ths_proxy/pay/*
服务注册中心
/ths_proxy/account/*
服务提供方
服务消费方
容错保护
负载均衡
服务路由
服务调度
容量管理
服务鉴权
服务降级
流量控制
分布式配置
消息总线
消息队列
日志管理
分布式调度
分布式事务
正在开发中
未开始

如何把现有服务加入分布式平台?

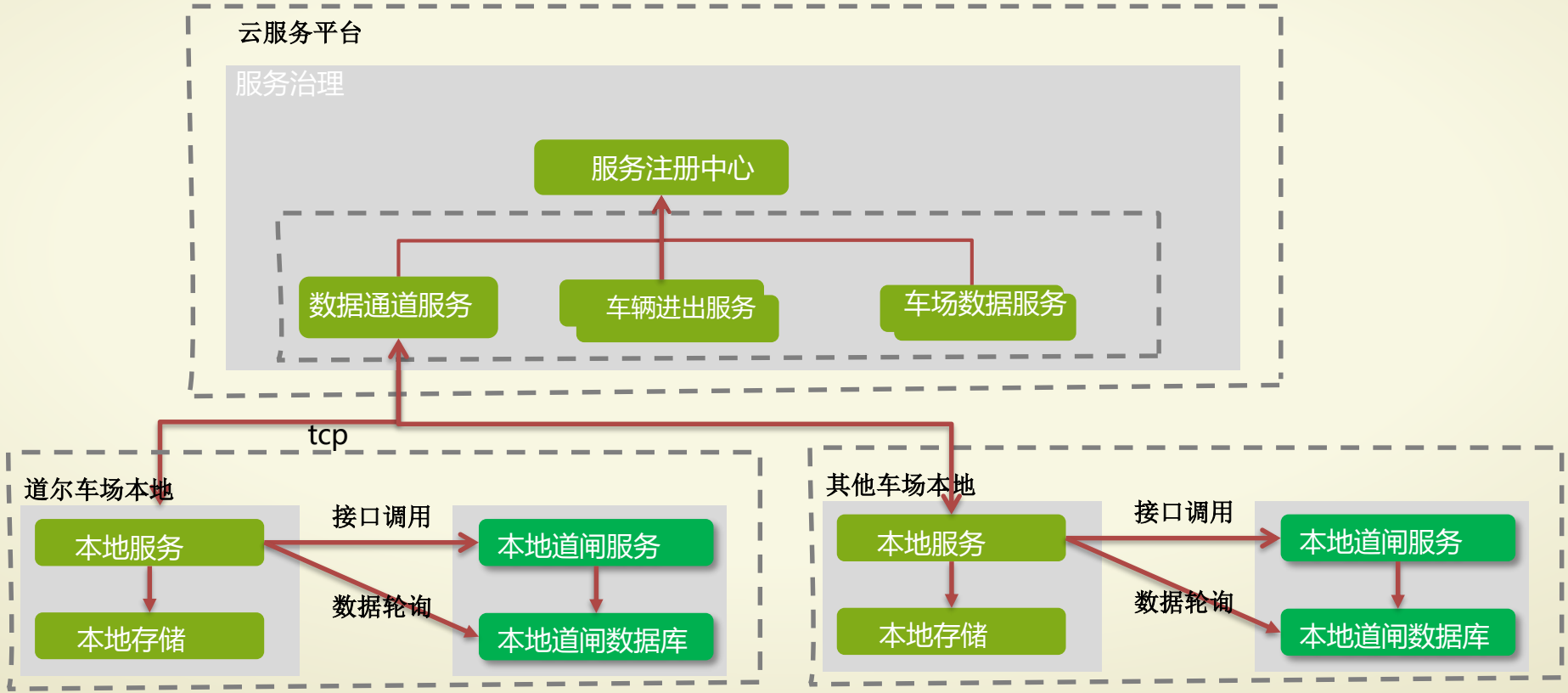
停划算共享停车服务



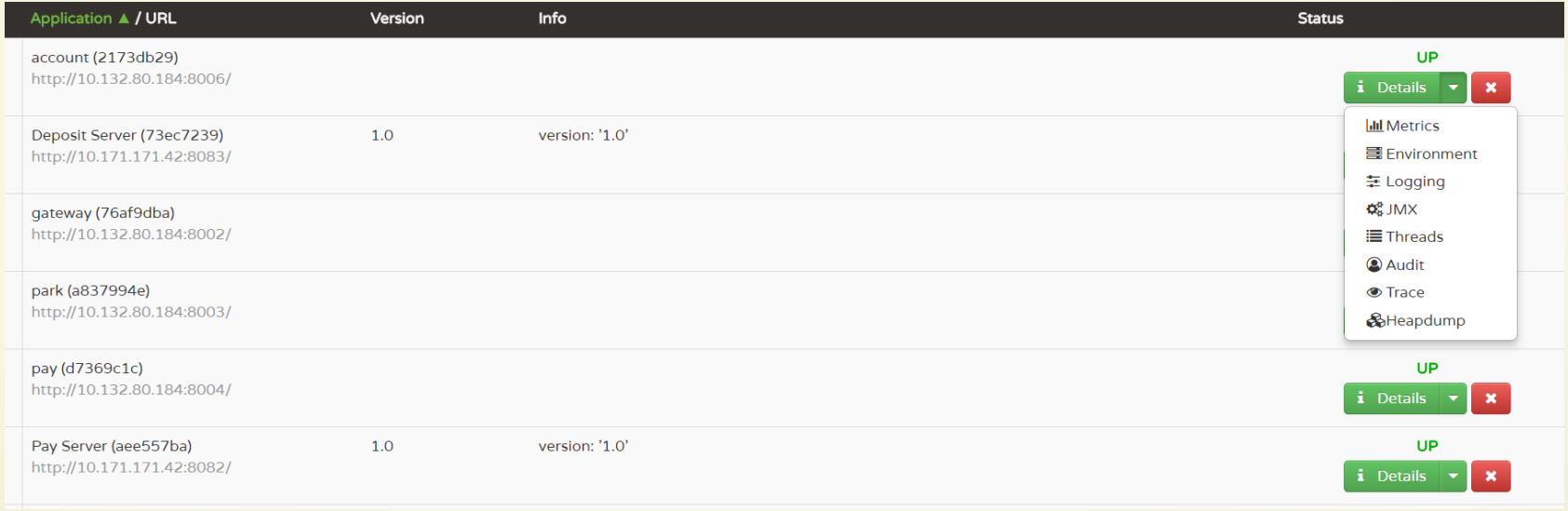
正在开发中

未开始

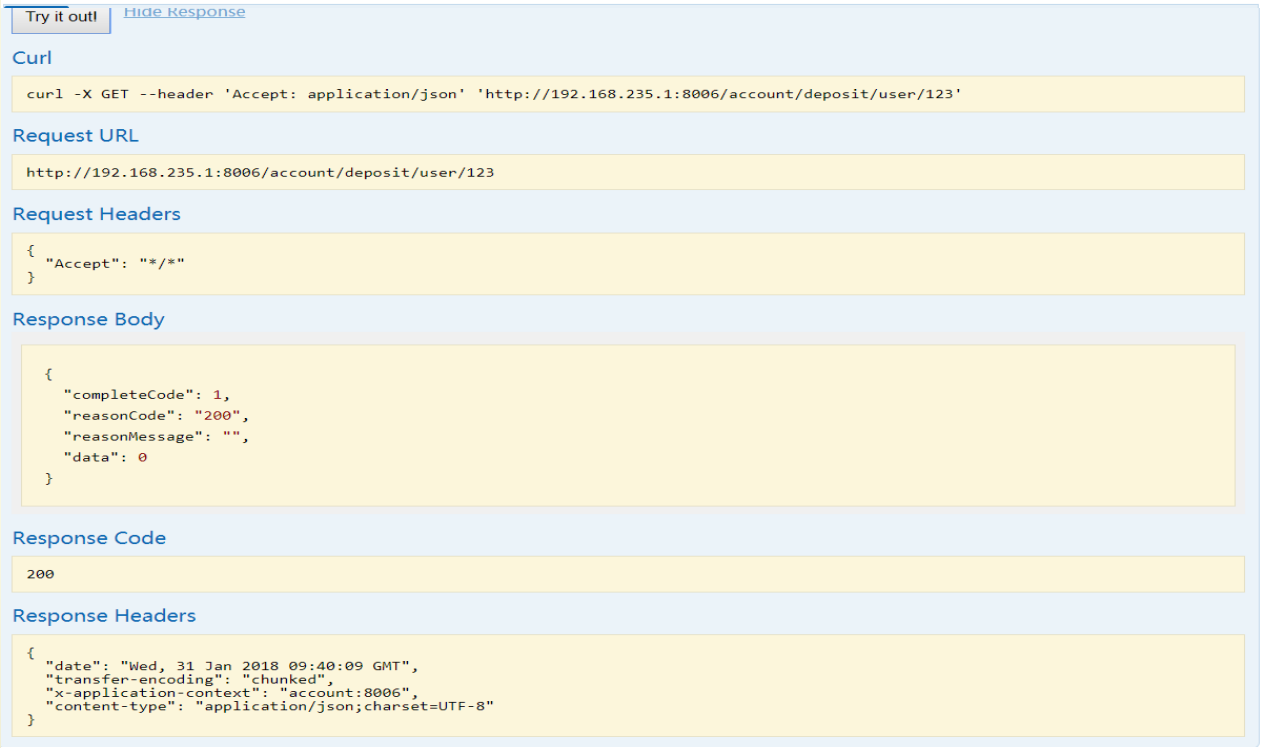
本地车场智慧停车服务


云服务平台
服务治理
服务注册中心
数据通道服务
车辆进出服务
车场数据服务
tcp
道尔车场本地
本地服务
本地存储
接口调用
数据轮询
本地道闸服务
本地道闸数据库
其他车场本地
本地服务
本地存储
接口调用
数据轮询
本地道闸服务
本地道闸数据库

服务列表

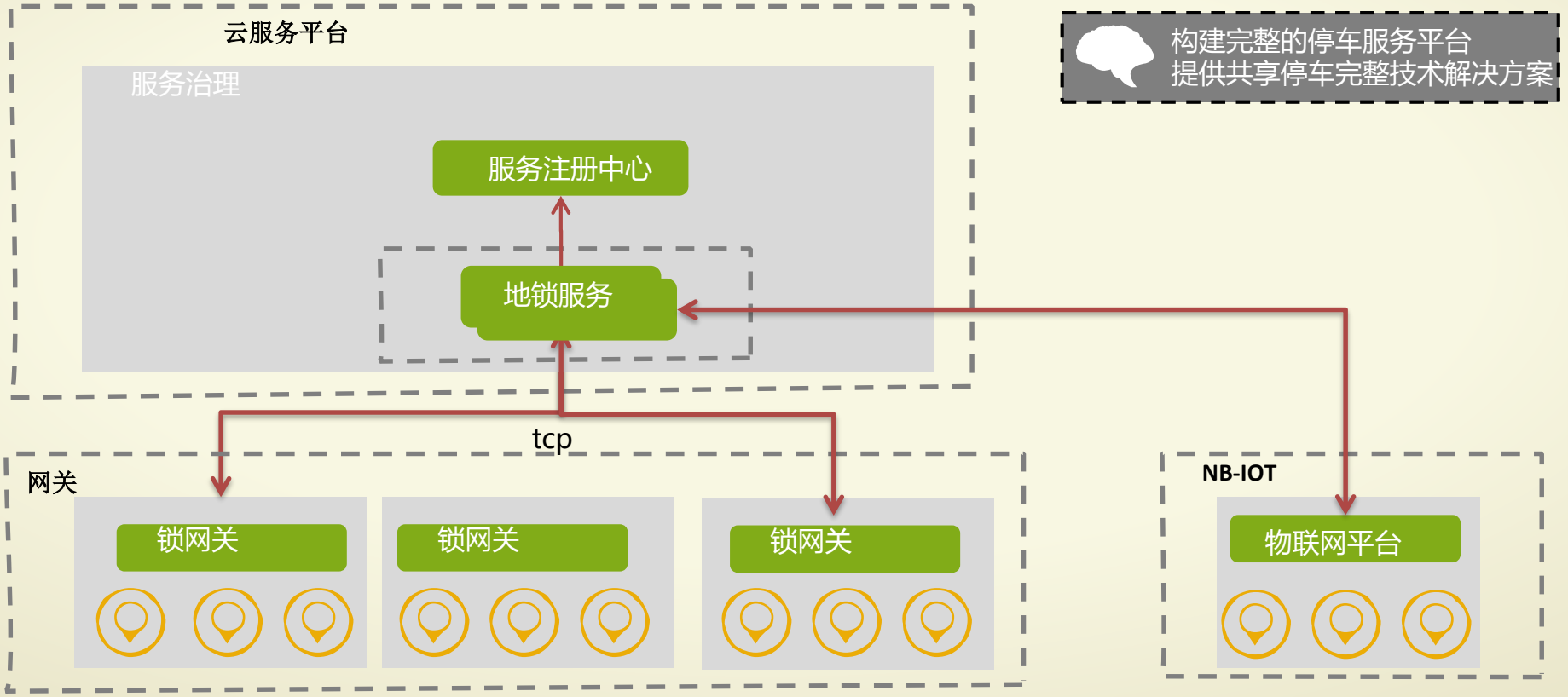

Application ▲ / URL
Version
Info
Status
account (2173db29)
http://10.132.80.184:8006/
UP
Details
Deposit Server (73ec7239)
http://10.171.171.42:8083/
1.0
version: '1.0'
Metrics
Environment
Logging
JMX
Threads
Audit
Trace
Heapdump
gateway (76af9dba)
http://10.132.80.184:8002/
park (a837994e)
http://10.132.80.184:8003/
pay (d7369c1c)
http://10.132.80.184:8004/
UP
Details
Pay Server (aee557ba)
http://10.171.171.42:8082/
1.0
version: '1.0'
UP
Details

OPEN-API

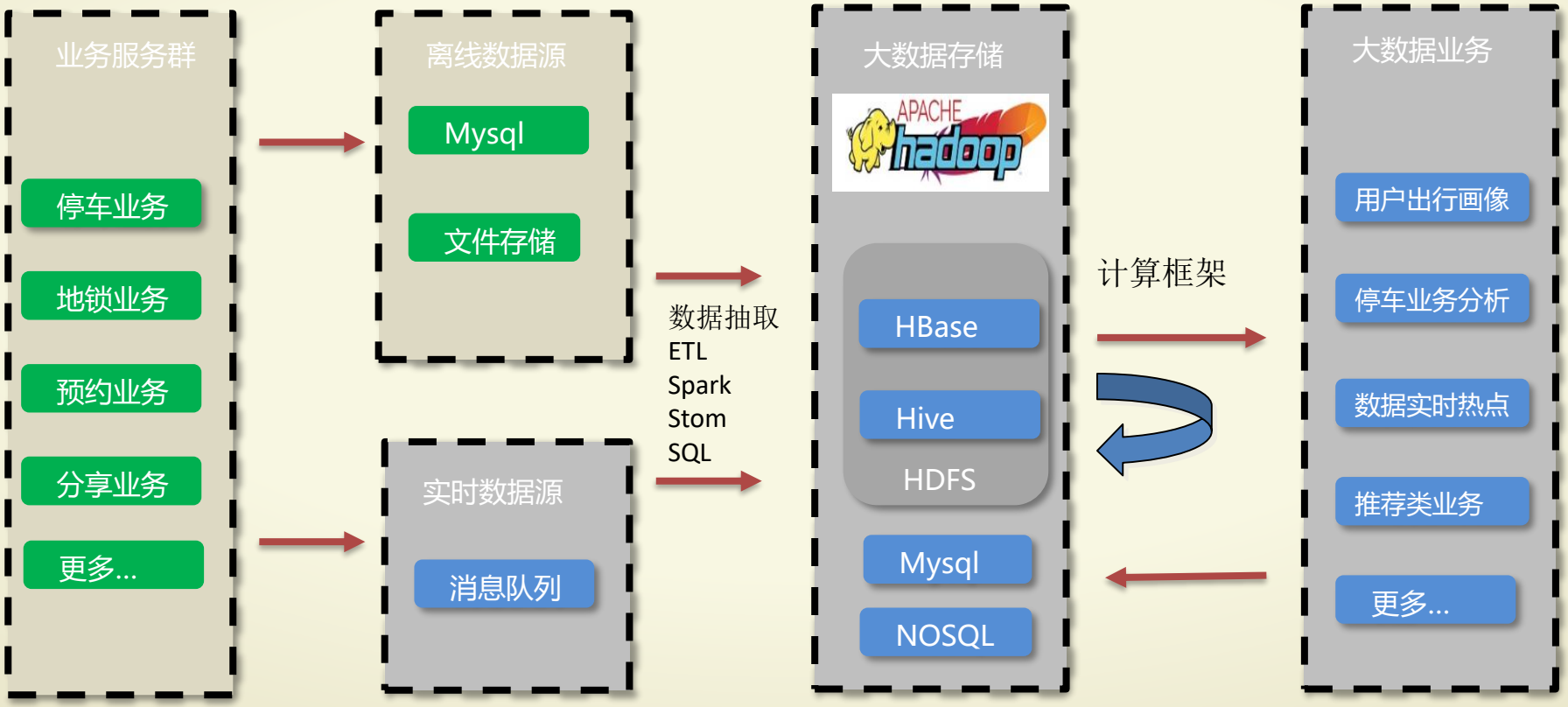

Try it out!
Hide Response
Curl
curl -X GET --header 'Accept: application/json' 'http://192.168.235.1:8006/account/deposit/user/123'
Request URL
http://192.168.235.1:8006/account/deposit/user/123
Request Headers
{
"Accept": "*/*"
}
Response Body
{
"completeCode": 1,
"reasonCode": "200",
"reasonMessage": "",
"data": 0
}
Response Code
200
Response Headers
{
"date": "Wed, 31 Jan 2018 09:40:09 GMT",
"transfer-encoding": "chunked",
"x-application-context": "account:8006",
"content-type": "application/json;charset=UTF-8"
}

目录

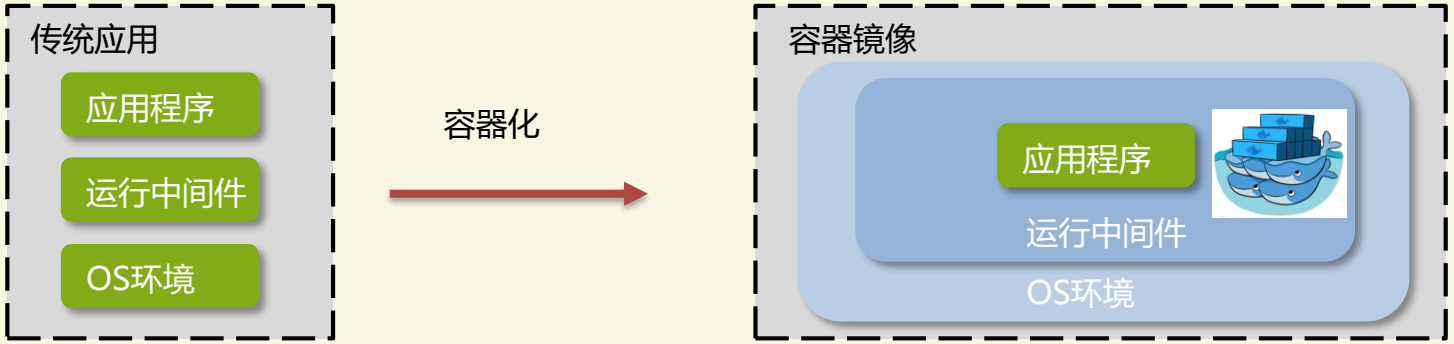
地锁管控服务


云服务平台
服务治理
服务注册中心
地锁服务
构建完整的停车服务平台
提供共享停车完整技术解决方案
tcp
网关
锁网关
锁网关
锁网关
NB-IOT
物联网平台

服务规划


业务服务群
停车业务
地锁业务
预约业务
分享业务
更多...
离线数据源
Mysql
文件存储
实时数据源
消息队列
数据抽取
ETL
Spark
Stom
SQL
大数据存储
APACHE hadoop
HBase
Hive
HDFS
Mysql
NOSQL
计算框架
大数据业务
用户出行画像
停车业务分析
数据实时热点
推荐类业务
更多...

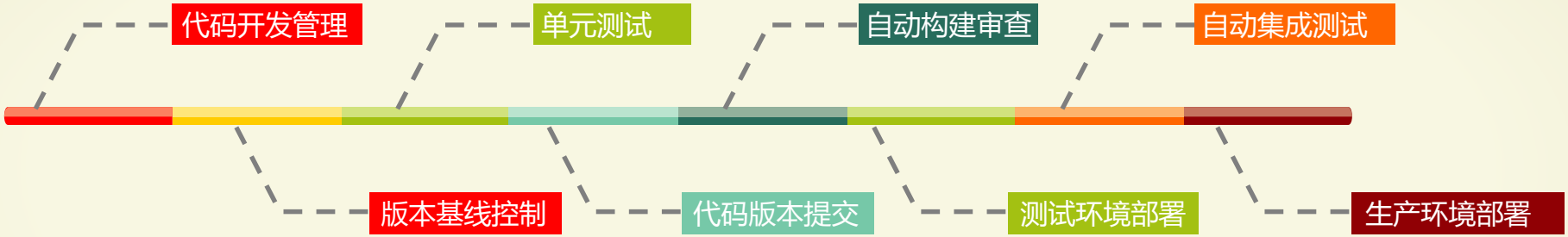
容器化部署



隔离应用依赖，创建应用镜像并进行复制，创建容易分发的即启即用的应用，允许实例简单、快速地扩展，测试应用并随后销毁它们

但需要很多技术的支撑，比如说，容器管理、编排、应用打包、容器间的网络、数据快照等等

持续交付



平台优化差不多之后，还要解决的问题是我们给业务交付怎么能够更快一些。包括代码管理，基线控制，自动构建审查，自动化测试还有环境部署。

利用一些工具做集成测试，例如，findbugs、junit、jenkins做自动化测试。通过自动化测试可以节省一些人力成本，但最终还是需要人工的测试用例来进行测试的。

目录

Q&A

谢谢！