Node.js 微应用实践

关于我

- 郑新林 (剪巽)
- 阿里云飞天一部：数据产品
- 关注：数据产品、前端、Node.js、可视化
- 目前主要产品：DTBoost、 DataV、城市大脑

Github

今天的话题

围绕前端、Node.js，聊聊前端的微应用开发体系的实践

- 什么是微应用
- 为什么微应用
- 怎么构建微应用

什么是微应用

微应用定义：

把产品分割为多个小模块，切细了开发

一个微应用完成一个业务功能

几个微应用组合，得到新产品

为什么微应用

想造轮子是不是?

一个产品一座冰山

- 一个系统：用户不可见部分多的多
- 除了用户业务，还有各种管控、数据报表
- 前台业务耗费大量精力，后台系统力不从心

数据场景的特点

- 众多的计算引擎：离线、在线、近线、实时等等，需要串联起来
- 服务众多：东查西查，这监控那监控
- 服务的访问量不高，但很重要，用户都是boss，数据就是$
- 需求众多：流程控制、数据转换、用户自定义、数据安全....

前端的通常状况

Using Node.js as dev tools

模块化

数据mock

打包编译

集成后端服务

- 发布到CDN
- 集成到Java Web Server

开发、联调、集成，一样一样的，全套流程繁琐

前端的通常状况

bug

fixed

....

trigger new bug

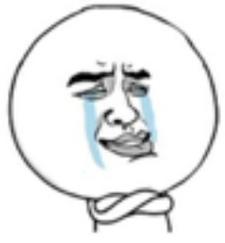
shit code

try refactoring

try ...

try ...

give up



我们都需要勇气 来面对重构危机

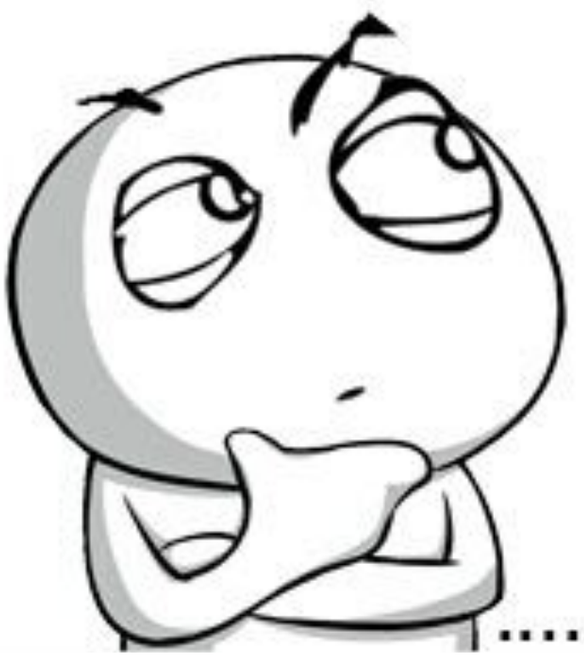
前端的通常状况

搞个可视化小工具?

搞个小饭桌管理?

做个ab测试?

抓点小数据?



支起文档服务?

.....

机器、访问、运维... 算了好麻烦?

.....

“创新”阻力大

前端紧缺

这事情我是知道的

提高前端的效率，从侧面解决人手短缺

如何改变

构建node服务开发体系（前后端分离在架构上）

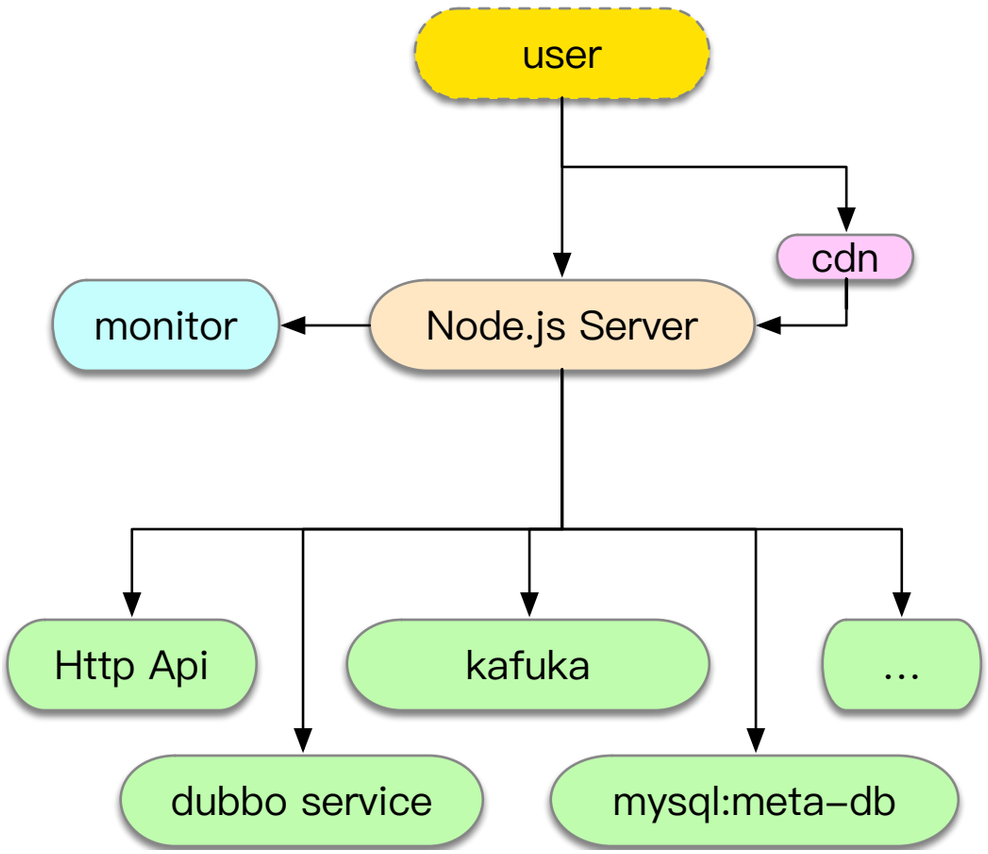
分割产品（控制不住屎代码，但可以控制屎的大小）

构建微应用运维体系（承载node服务，解放前端开发）

管理好产品经理

构建Node层开发体系

Using Node.js as service



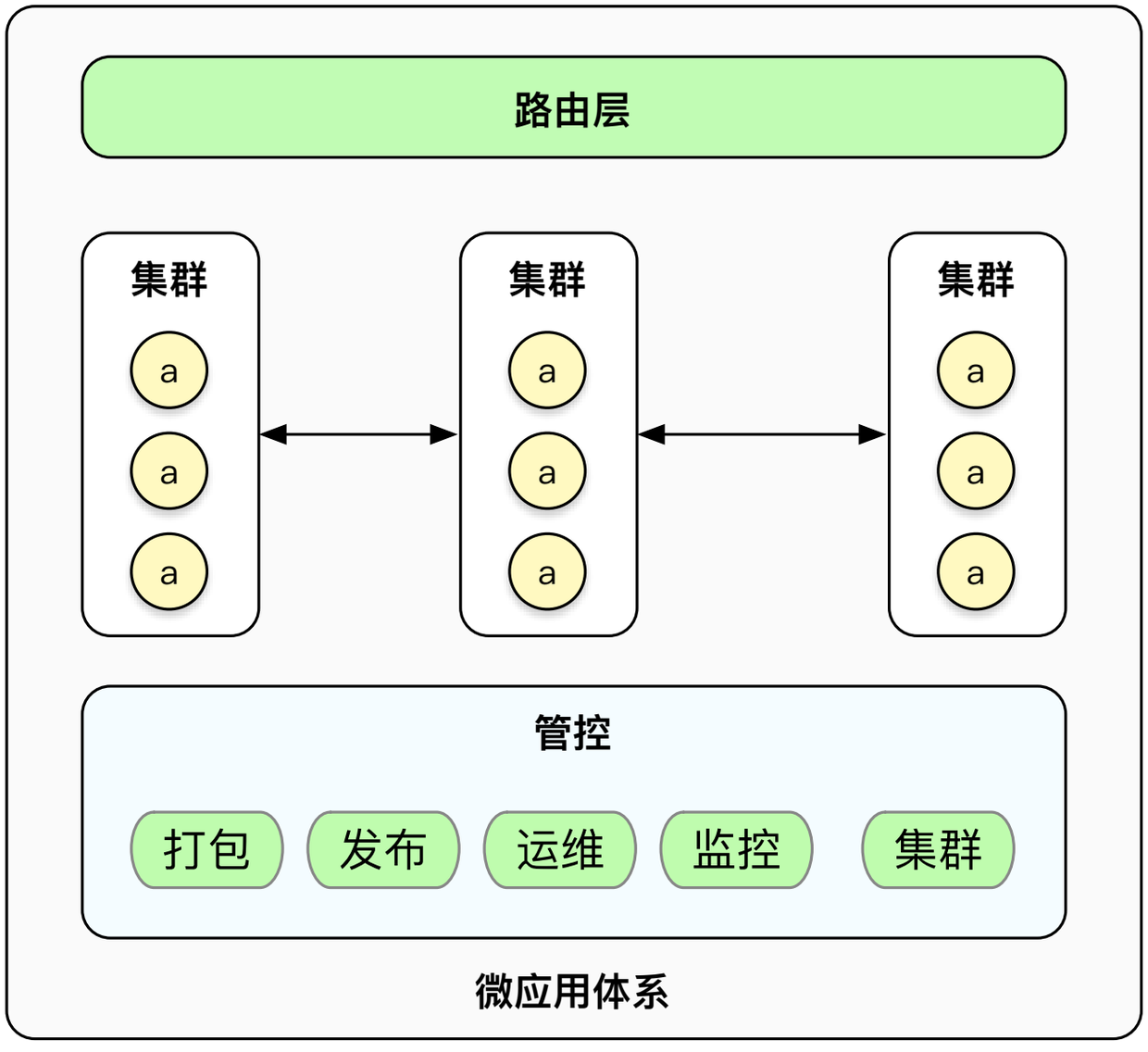
- web层框架一体
- 抽取前端的组件包
- 抽取Node.js层包

make install / make test / make release -> app.tgz

分割产品

信息架构 -> 业务架构 -> 多应用

构建微应用运维体系



构建
发布
运维
监控
扩容

我们的实践体系

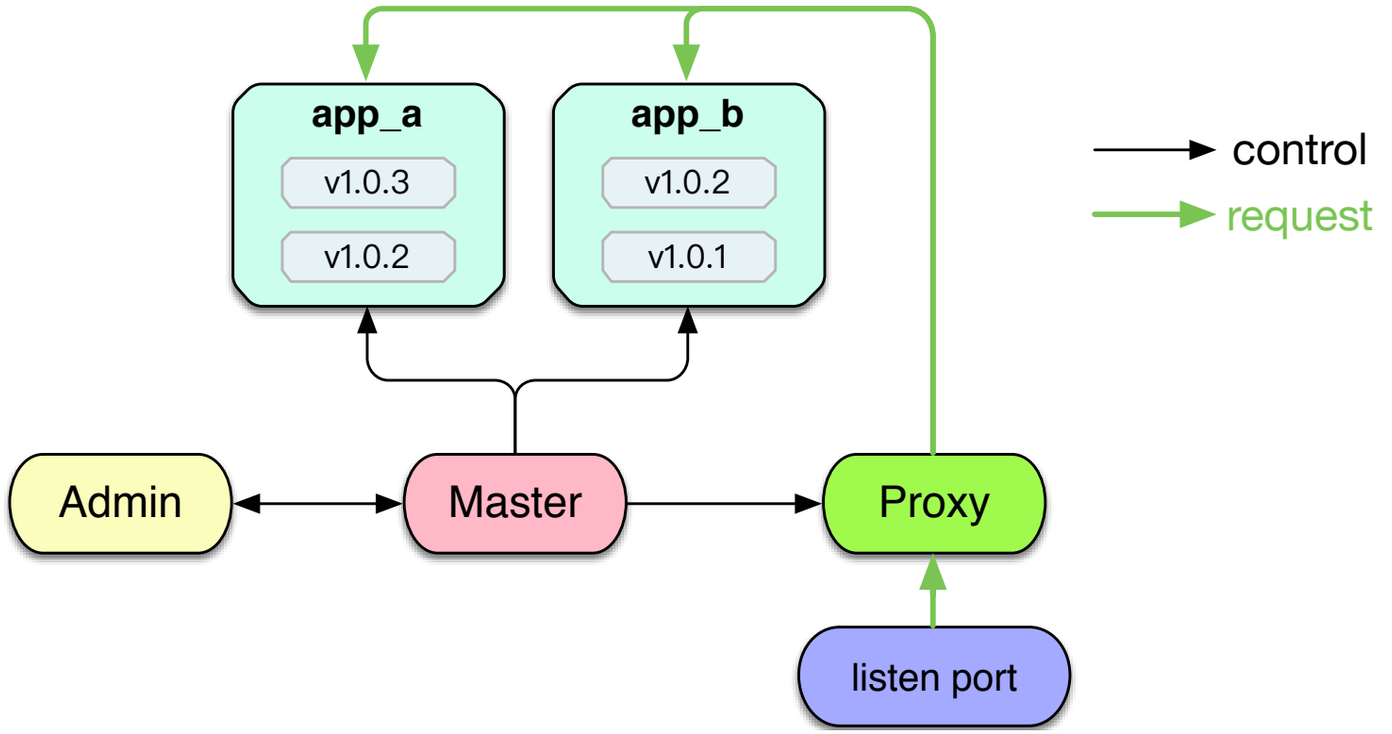
微应用体系

- server：app 运行托管
- framework：app开发框架
- ui-console：运维
- cli：开发工具

进程管理之外

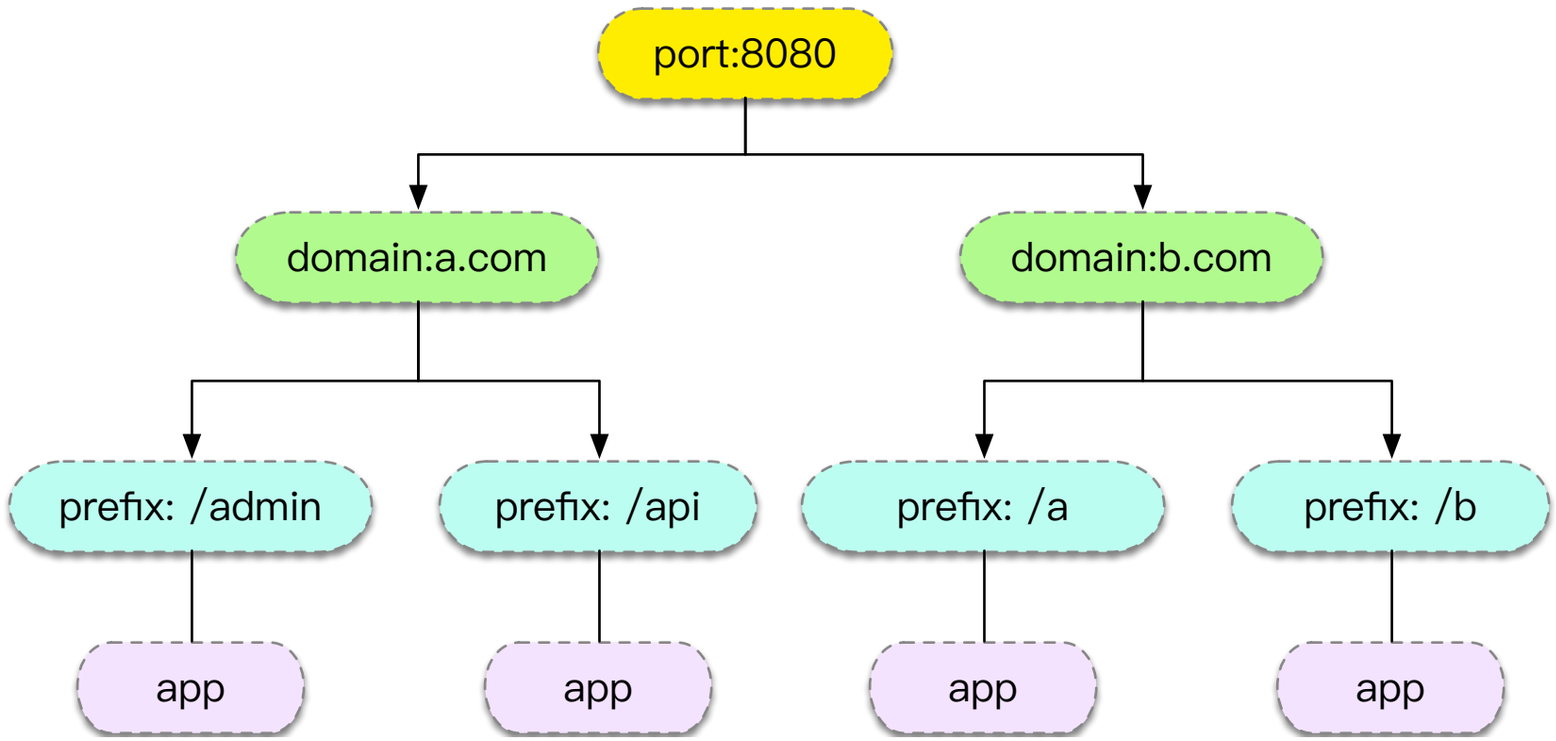
- 路由托管
- 高可用、版本无缝切换
- 便捷配置、运维管理
- 面向进程 vs 面向应用

微应用管理服务的架构

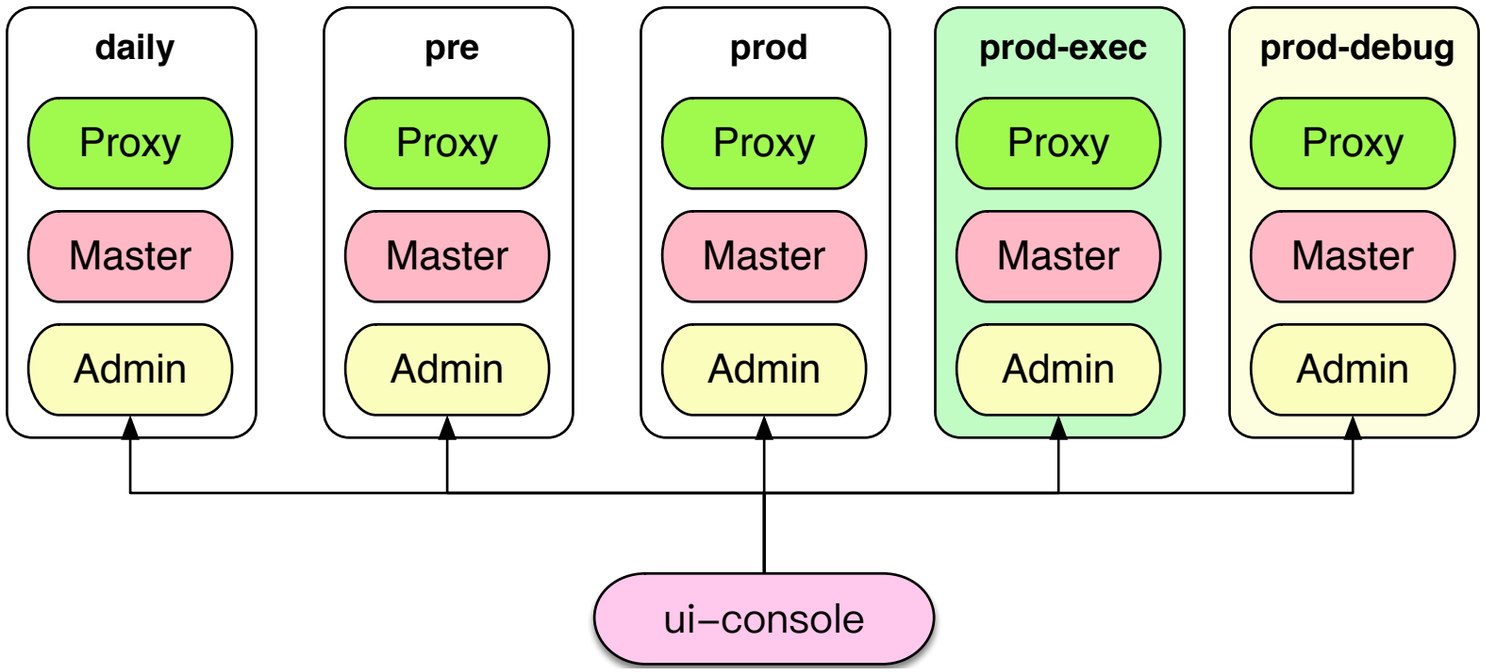


- 发布应用
- 管理路由
- 运维、监控

微应用的路由管理

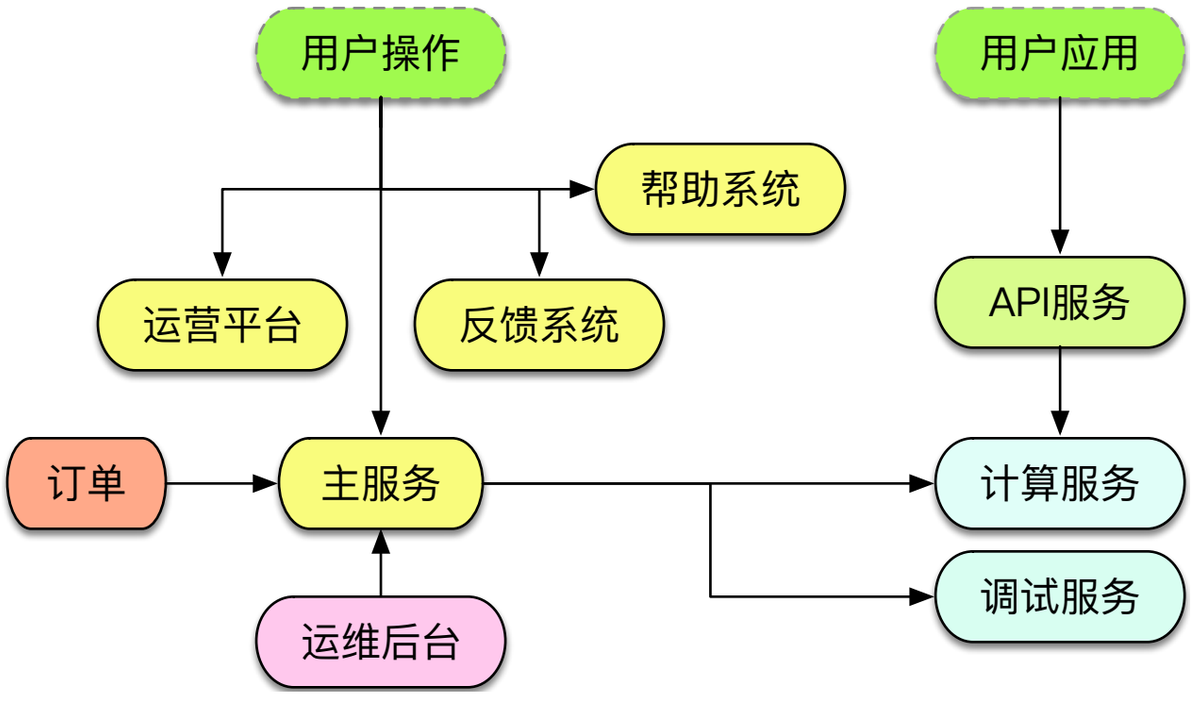

port:8080
domain:a.com
domain:b.com
prefix: /admin
prefix: /api
prefix: /a
prefix: /b
app
app
app
app

微应用 - 集群


daily
Proxy
Master
Admin
pre
Proxy
Master
Admin
prod
Proxy
Master
Admin
prod-exec
Proxy
Master
Admin
prod-debug
Proxy
Master
Admin
ui-console

微应用分解 - 推荐系统

分解应用：



规划路径：

$domain:8888/　订单系统
$domain/main　运营平台
$domain/help 帮助系统
$domain/feedback 反馈系统
$domain/console 主服务 控制台
$doamin/service_debug 调试服务
$domain/service 在线服务
$domain/api 接口服务

场景和产品的迫切需要

动态语言
网络能力
前后端无缝
社区丰富的模块

微应用体系
连接数据：网络
动态特性：编译、安全
执行：谁说密集计算不行

开发、部署、运维

效率、门槛、成本

一个前端干n个App，招全栈工程师的这点小心思

套路

微应用的优势

1. 效率：开发、测试、重构、复用
2. 解放：前端想象力
3. 资源：节约成本

另一面

需要权衡拆到什么力度，管理的成本控制

大规模使用，需要专业的后端工程师、DevOps角色

和微服务的对比

微应用 <> 微服务

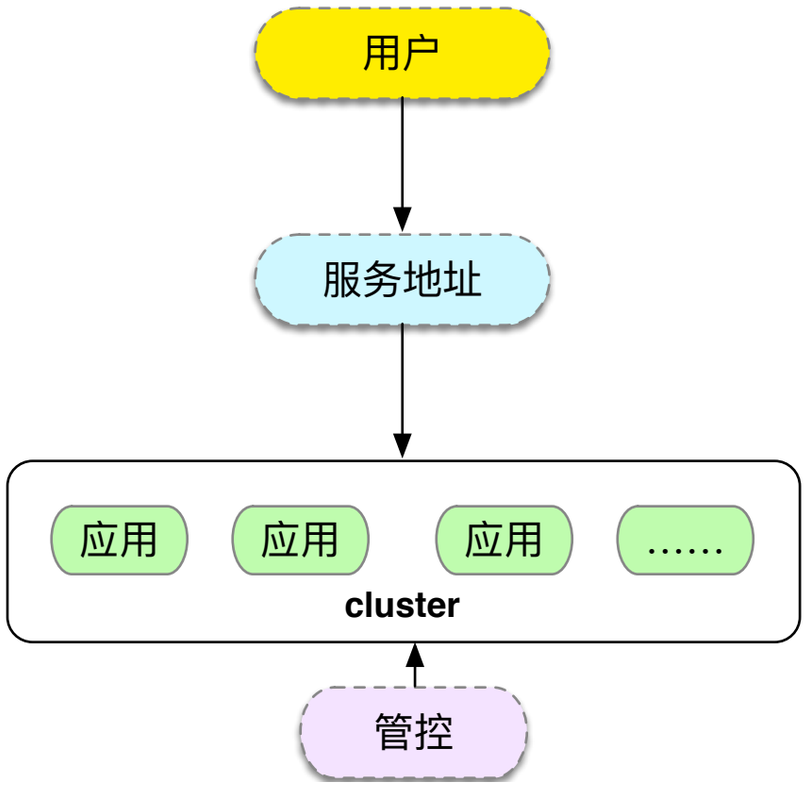
java <> javascript ?

雷锋 <> 雷峰塔 ?

NETFLIX

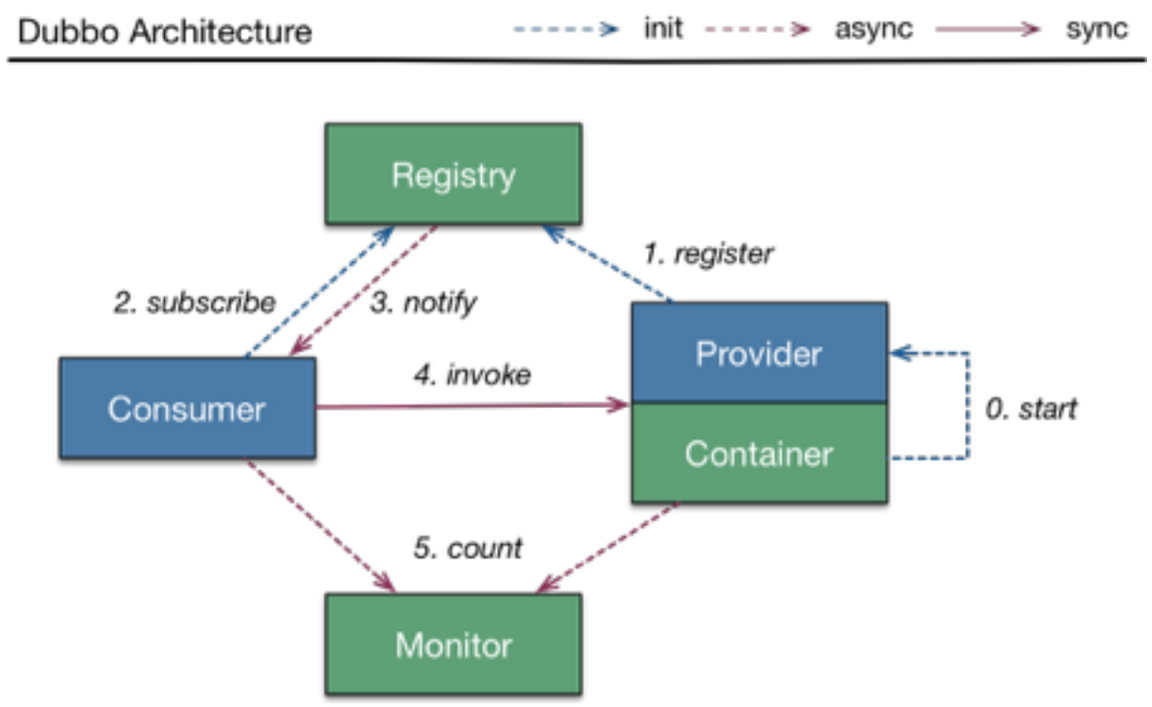
和微服务的对比

微应用



关注应用接入、访问

微服务



关注服务接入、消费

微应用趋势分析

- 产品数据化研发，水下冰山在壮大，需求增加
- 前后端分离是架构分离，而非代码分离
- 产品 = 应用A + 应用B + 应用C
- AI = 智能 + 数据 + App（外设：接收端、执行端）

微应用体系，可以试试

Honeycomb

a micro-app container

谢 谢

Q&A