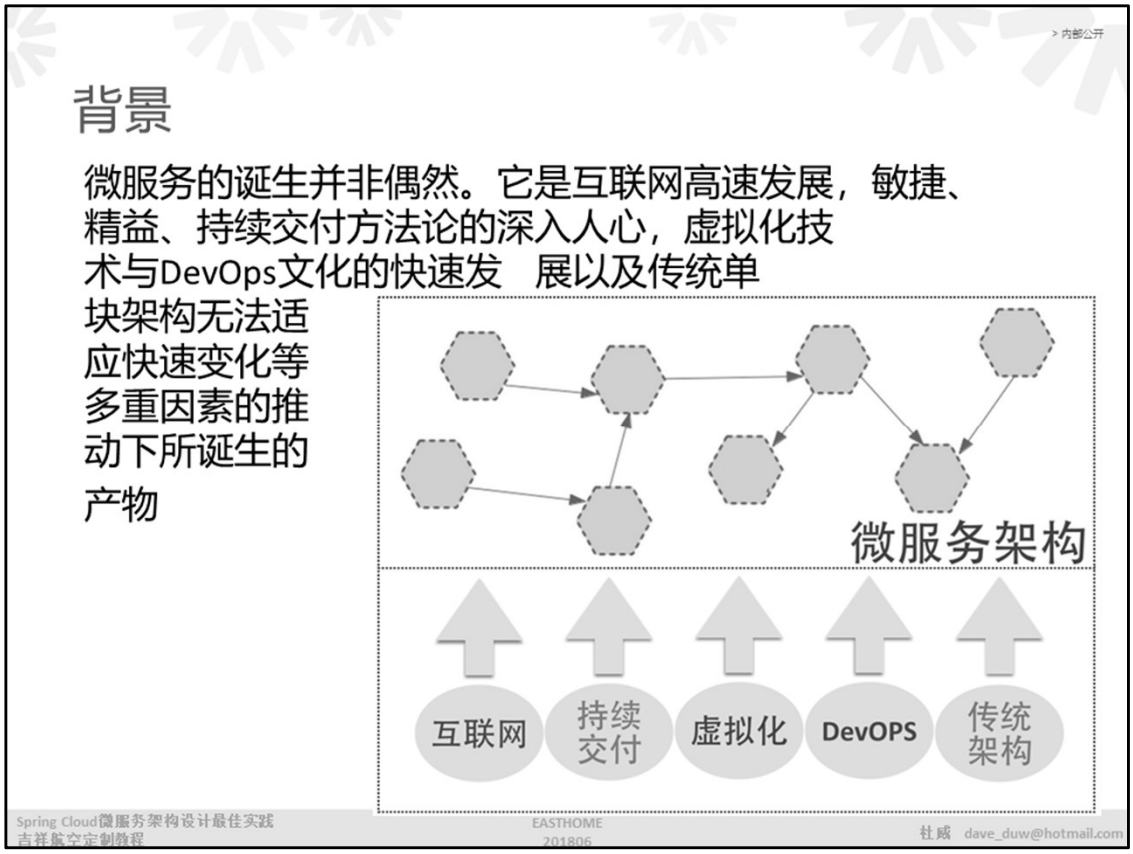
Notes:

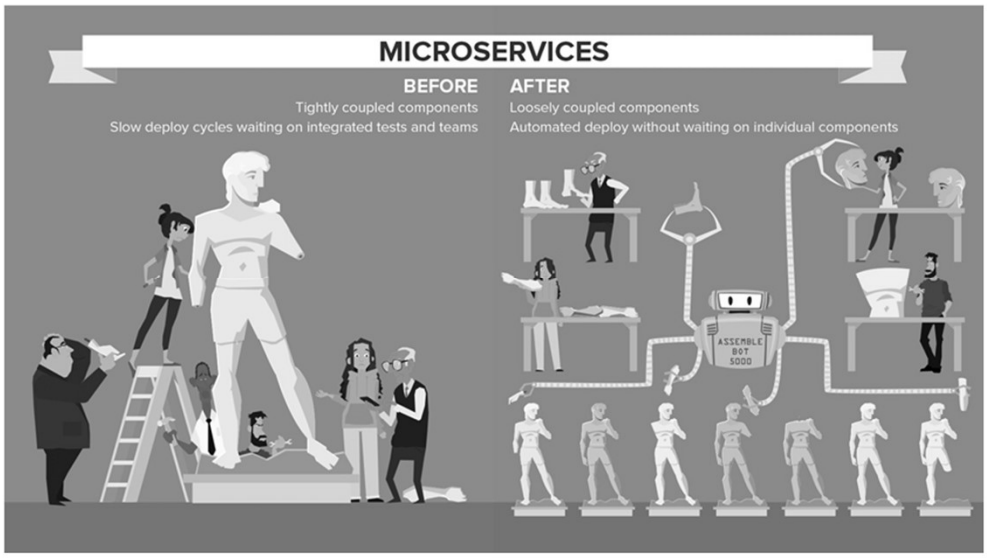

> 内部公开
1
微服务的诞生
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
背景
微服务的诞生并非偶然。它是互联网高速发展，敏捷、精益、持续交付方法论的深入人心，虚拟化技术与DevOps文化的快速发 展以及传统单块架构无法适应快速变化等多重因素的推动下所诞生的产物
微服务架构
互联网
持续交付
虚拟化
DevOPS
传统架构
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

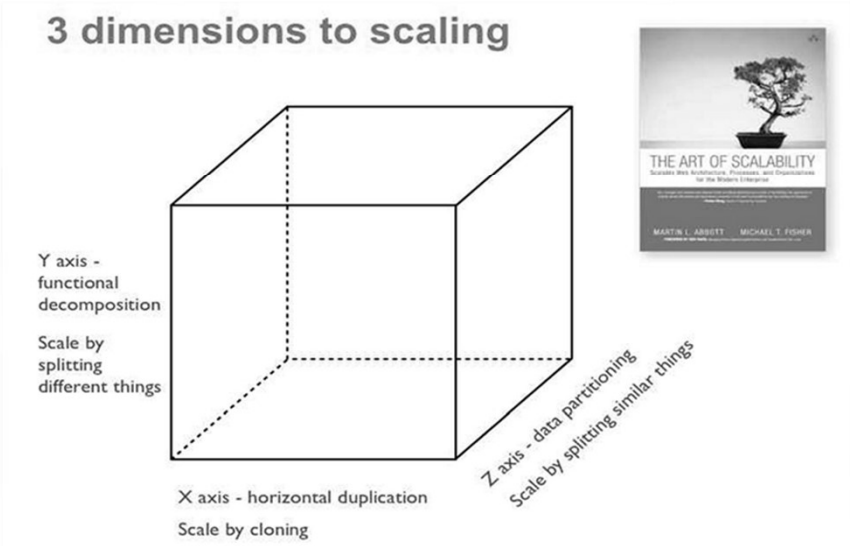
Notes:

与单体系统的区别



Notes:

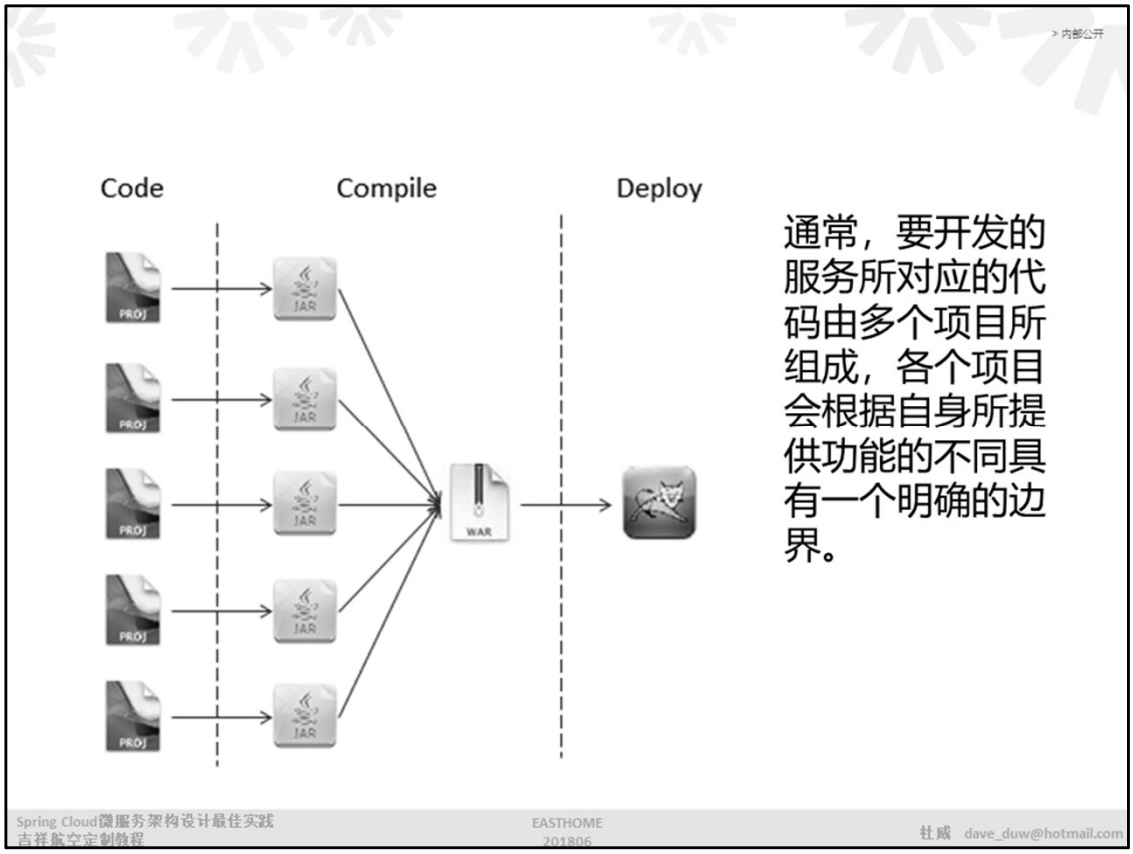
扩展立方


3 dimensions to scaling
Y axis - functional decomposition
Scale by splitting different things
X axis - horizontal duplication
Scale by cloning
Z axis - data partitioning
Scale by splitting similar things
THE ART OF SCALABILITY
MARTIN L. ABBOTT
MICHAEL T. FISHER

Notes:


> 内部公开
2
Monolith
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

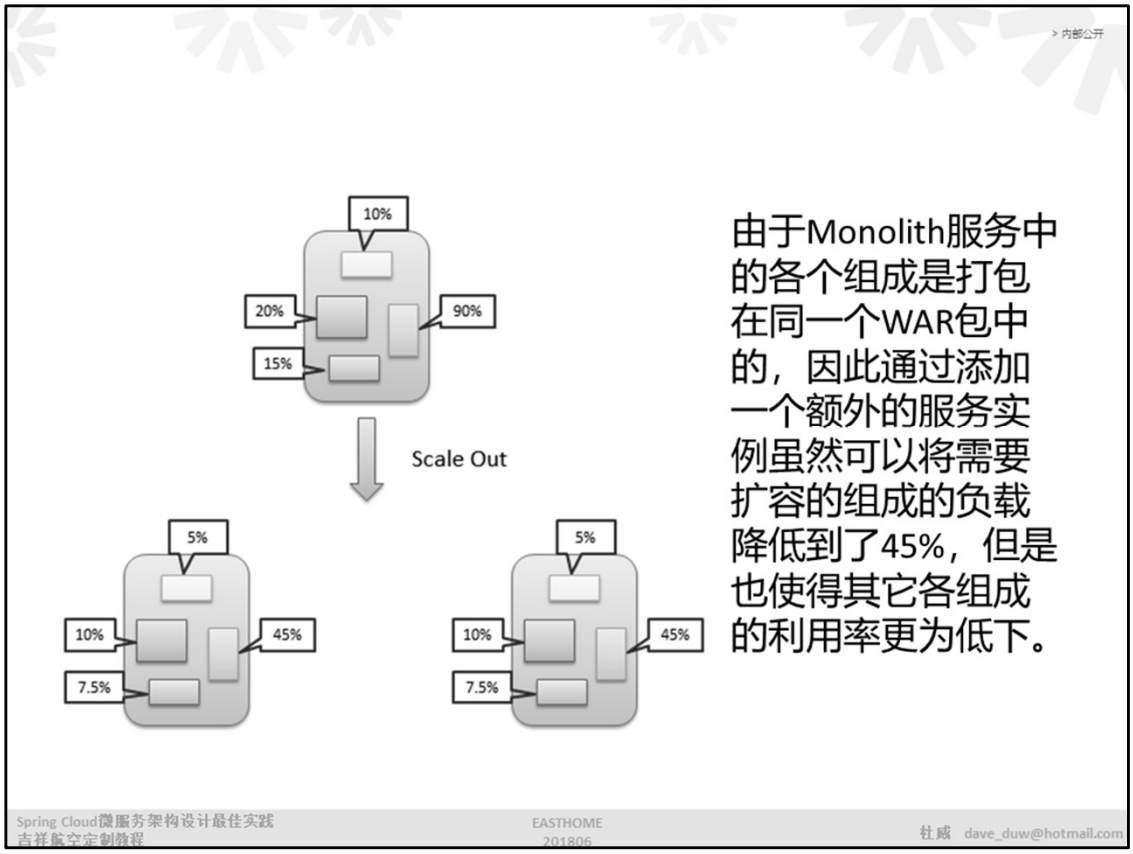

> 内部公开
Code
Compile
Deploy
PROJ
JAR
WAR
通常，要开发的服务所对应的代码由多个项目所组成，各个项目会根据自身所提供功能的不同具有一个明确的边界。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

如果应用的部署非常麻烦，那么为了对自己的更改进行测试，软件开发人员还需要在部署前进行大量的环境设置，进而使得软件开发人员 的工作变得繁杂而无趣

从上面的示意图中可以看到，在应用变大之后，软件开发人员花在编译及部署的时间明显增多，甚至超过了他对代码进行更改并测试的时间，效率已经变得十分低下。

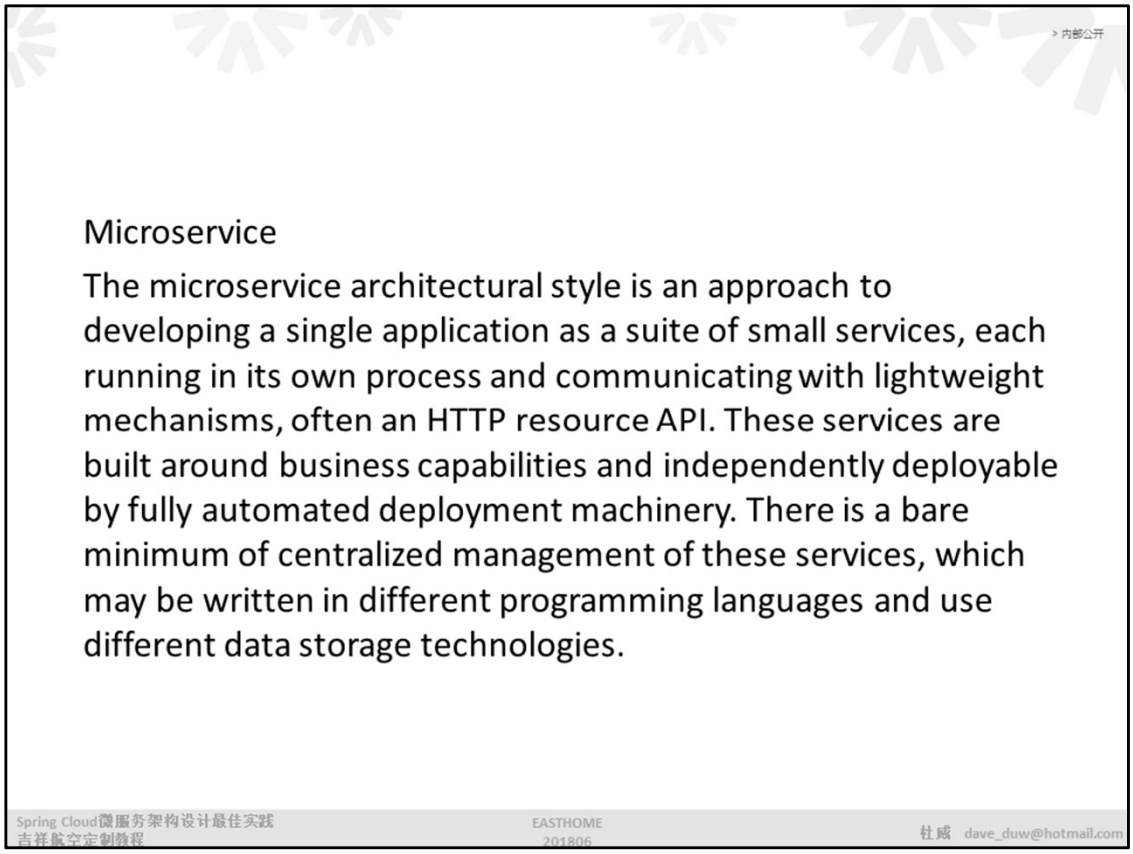
Notes:


> 内部公开
10%
20%
90%
15%
Scale Out
5%
10%
45%
7.5%
5%
10%
45%
7.5%
由于Monolith服务中的各个组成是打包在同一个WAR包中的，因此通过添加一个额外的服务实例虽然可以将需要扩容的组成的负载降低到了45%，但是也使得其它各组成的利用率更为低下。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

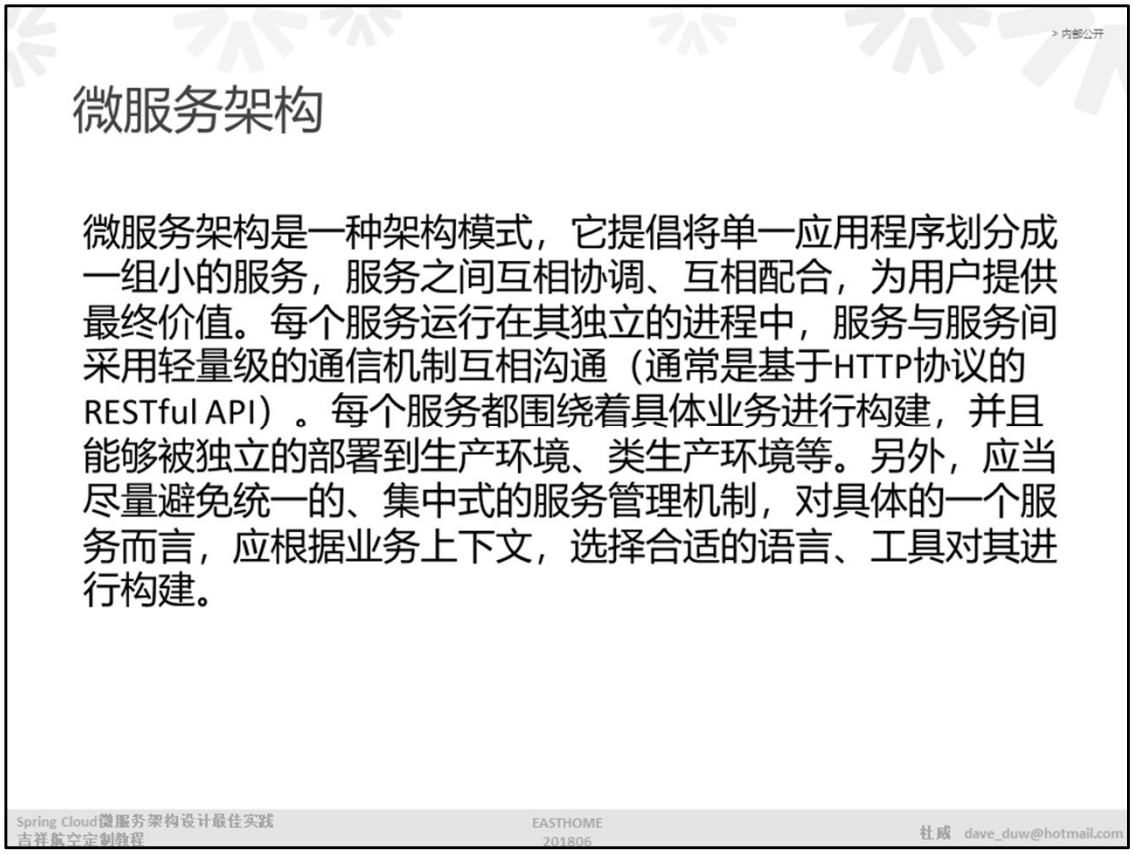
Notes:


> 内部公开
3
微服务的定义
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

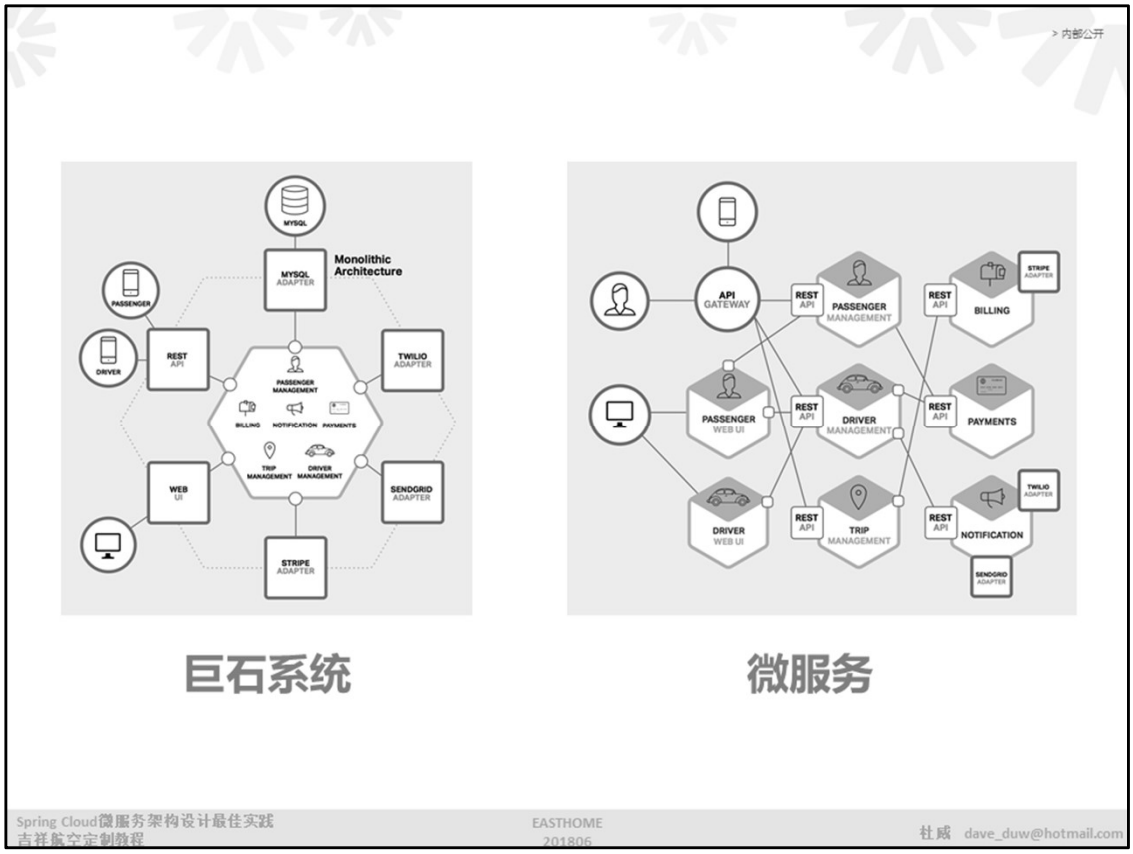
Notes:


> 内部公开
Microservice
The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

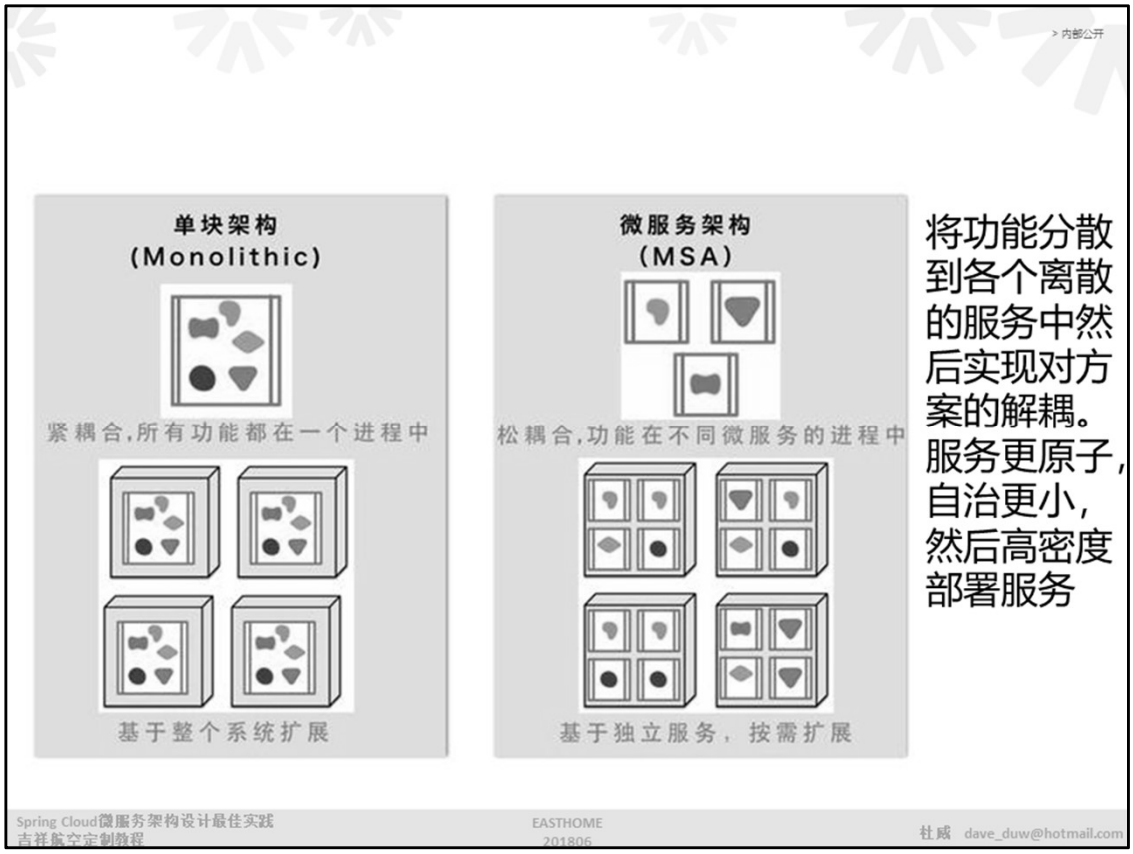
Notes:


> 内部公开
微服务架构
微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采用轻量级的通信机制互相沟通（通常是基于HTTP协议的RESTful API）。每个服务都围绕着具体业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。另外，应当尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

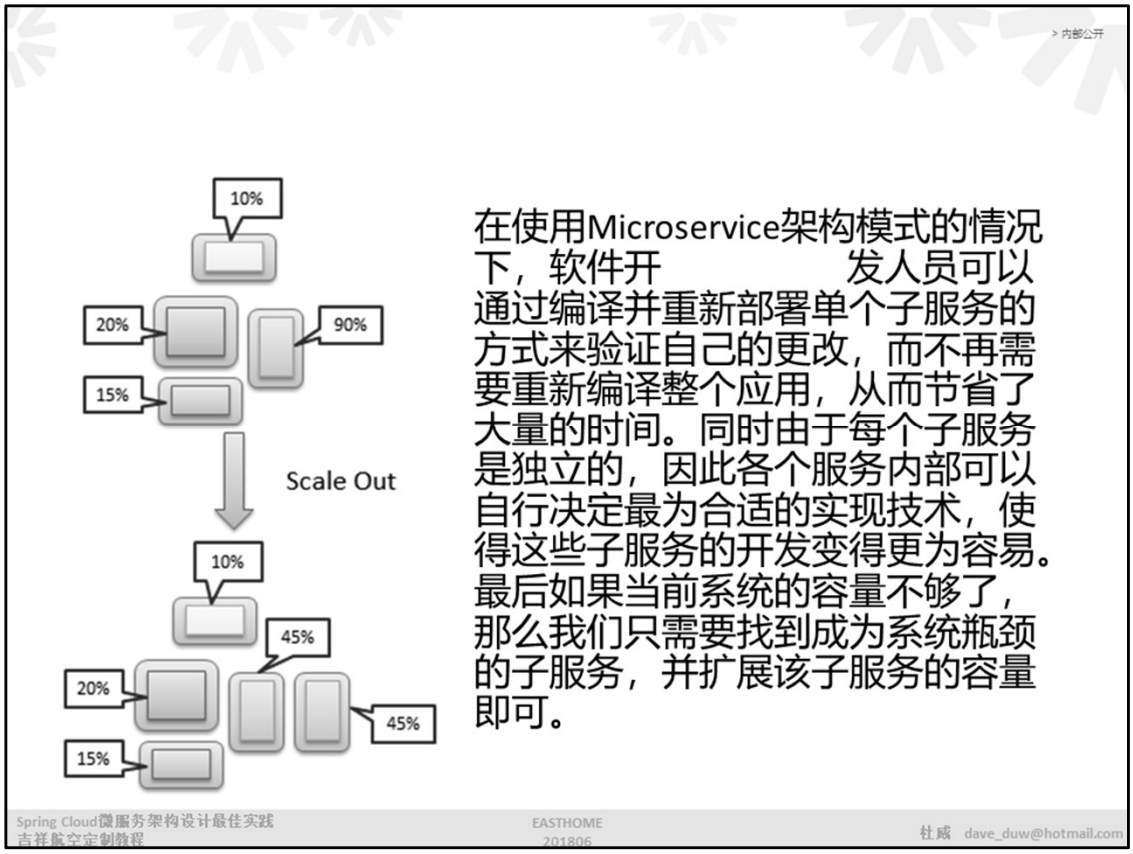
Notes:


> 内部公开
MYSQL
Monolithic Architecture
MYSQL ADAPTER
PASSENGER
DRIVER
REST API
TWILIO ADAPTER
PASSENGER MANAGEMENT
BILLING
NOTIFICATION
PAYMENTS
TRIP MANAGEMENT
DRIVER MANAGEMENT
WEB UI
SENDGRID ADAPTER
STRIPE ADAPTER
巨石系统
API GATEWAY
REST API
PASSENGER MANAGEMENT
STRIPE ADAPTER
BILLING
PASSENGER WEB UI
DRIVER MANAGEMENT
PAYMENTS
DRIVER WEB UI
TRIP MANAGEMENT
TWILIO ADAPTER
NOTIFICATION
SENDGRID ADAPTER
微服务
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
单块架构
(Monolithic)
紧耦合,所有功能都在一个进程中
基于整个系统扩展
微服务架构
(MSA)
松耦合,功能在不同微服务的进程中
基于独立服务，按需扩展
将功能分散到各个离散的服务中然后实现对方案的解耦。服务更原子，自治更小，然后高密度部署服务
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

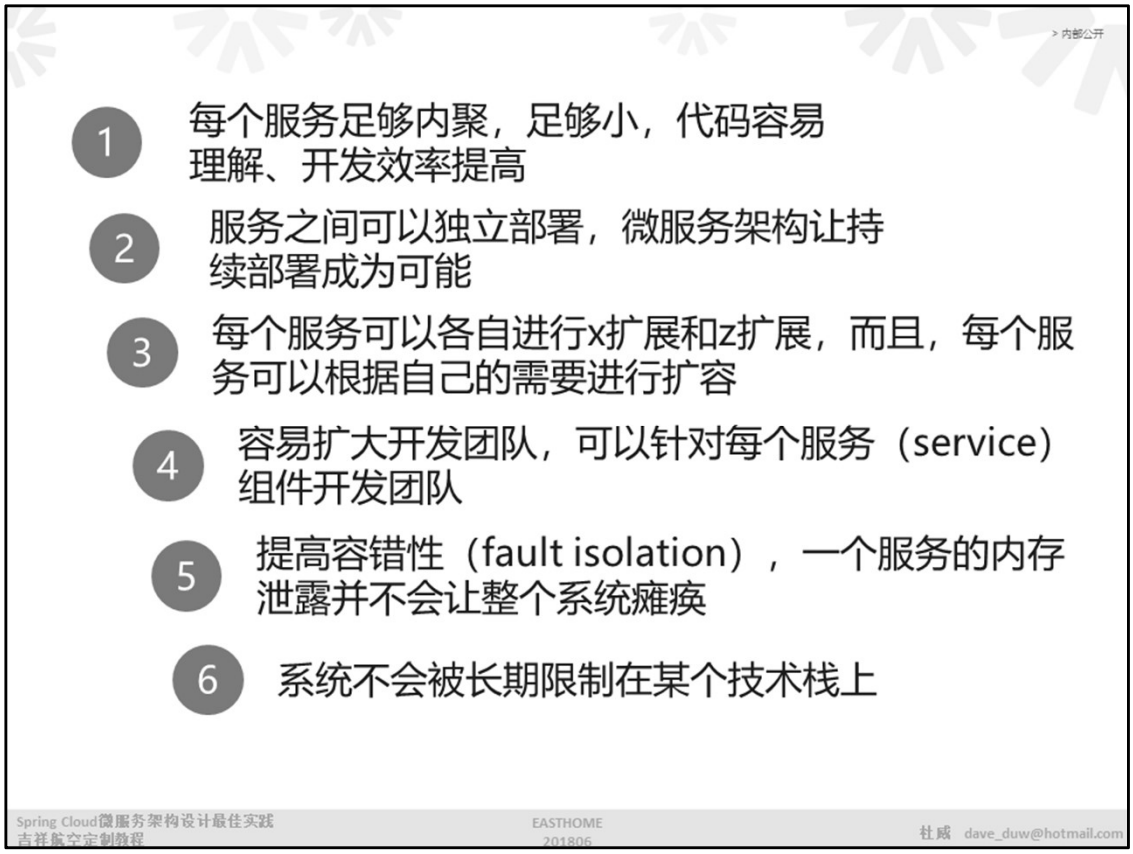
Notes:


> 内部公开
10%
20%
90%
15%
Scale Out
10%
45%
20%
45%
15%
在使用Microservice架构模式的情况下，软件开 发人员可以通过编译并重新部署单个子服务的方式来验证自己的更改，而不再需要重新编译整个应用，从而节省了大量的时间。同时由于每个子服务是独立的，因此各个服务内部可以自行决定最为合适的实现技术，使得这些子服务的开发变得更为容易。最后如果当前系统的容量不够了，那么我们只需要找到成为系统瓶颈的子服务，并扩展该子服务的容量即可。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

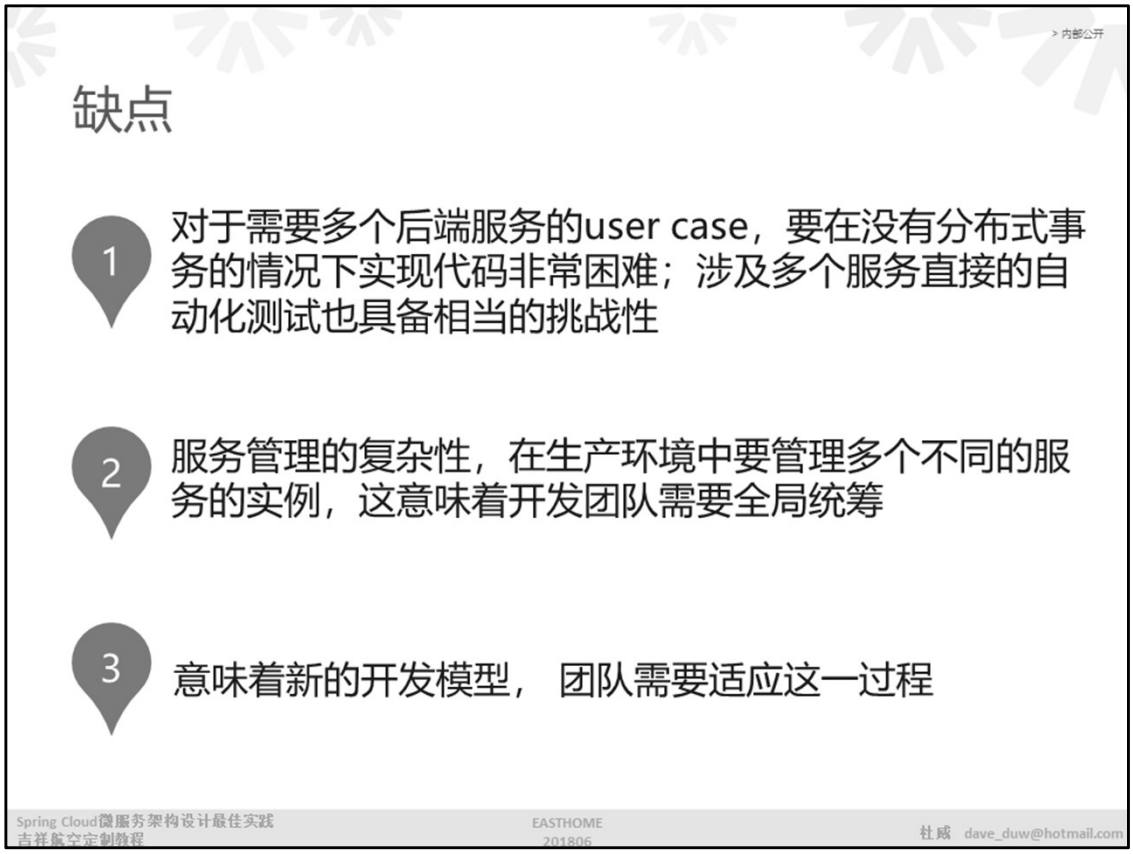
Notes:


> 内部公开
4
微服务架构的优点与缺点
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
1
每个服务足够内聚，足够小，代码容易理解、开发效率提高
2
服务之间可以独立部署，微服务架构让持续部署成为可能
3
每个服务可以各自进行x扩展和z扩展，而且，每个服务可以根据自己的需要进行扩容
4
容易扩大开发团队，可以针对每个服务（service）组件开发团队
5
提高容错性（fault isolation），一个服务的内存泄露并不会让整个系统瘫痪
6
系统不会被长期限制在某个技术栈上
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

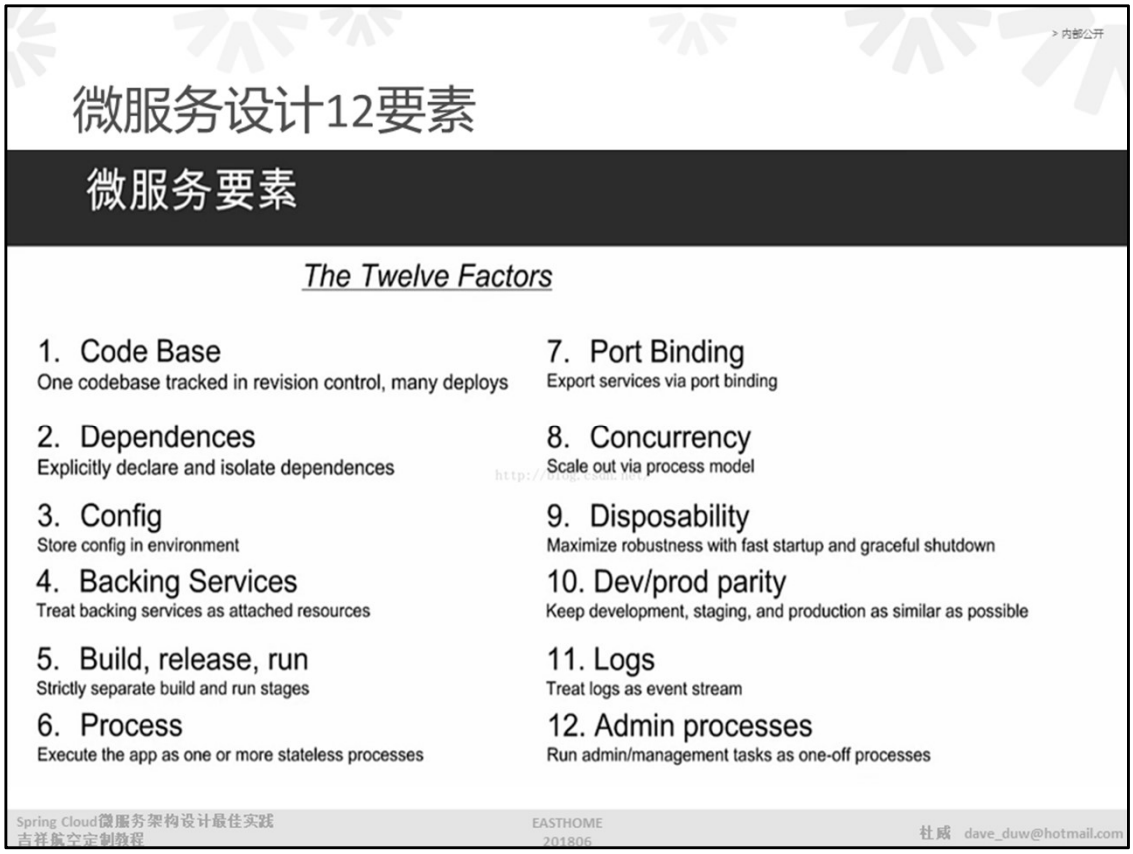
Notes:


> 内部公开
缺点
1
对于需要多个后端服务的user case，要在没有分布式事务的情况下实现代码非常困难；涉及多个服务直接的自动化测试也具备相当的挑战性
2
服务管理的复杂性，在生产环境中要管理多个不同的服务的实例，这意味着开发团队需要全局统筹
3
意味着新的开发模型， 团队需要适应这一过程
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
5
微服务架构12要素
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
微服务设计12要素
微服务要素
The Twelve Factors
1. Code Base
One codebase tracked in revision control, many deploys
2. Dependences
Explicitly declare and isolate dependences
3. Config
Store config in environment
4. Backing Services
Treat backing services as attached resources
5. Build, release, run
Strictly separate build and run stages
6. Process
Execute the app as one or more stateless processes
7. Port Binding
Export services via port binding
8. Concurrency
Scale out via process model
9. Disposability
Maximize robustness with fast startup and graceful shutdown
10. Dev/prod parity
Keep development, staging, and production as similar as possible
11. Logs
Treat logs as event stream
12. Admin processes
Run admin/management tasks as one-off processes
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

基准代码

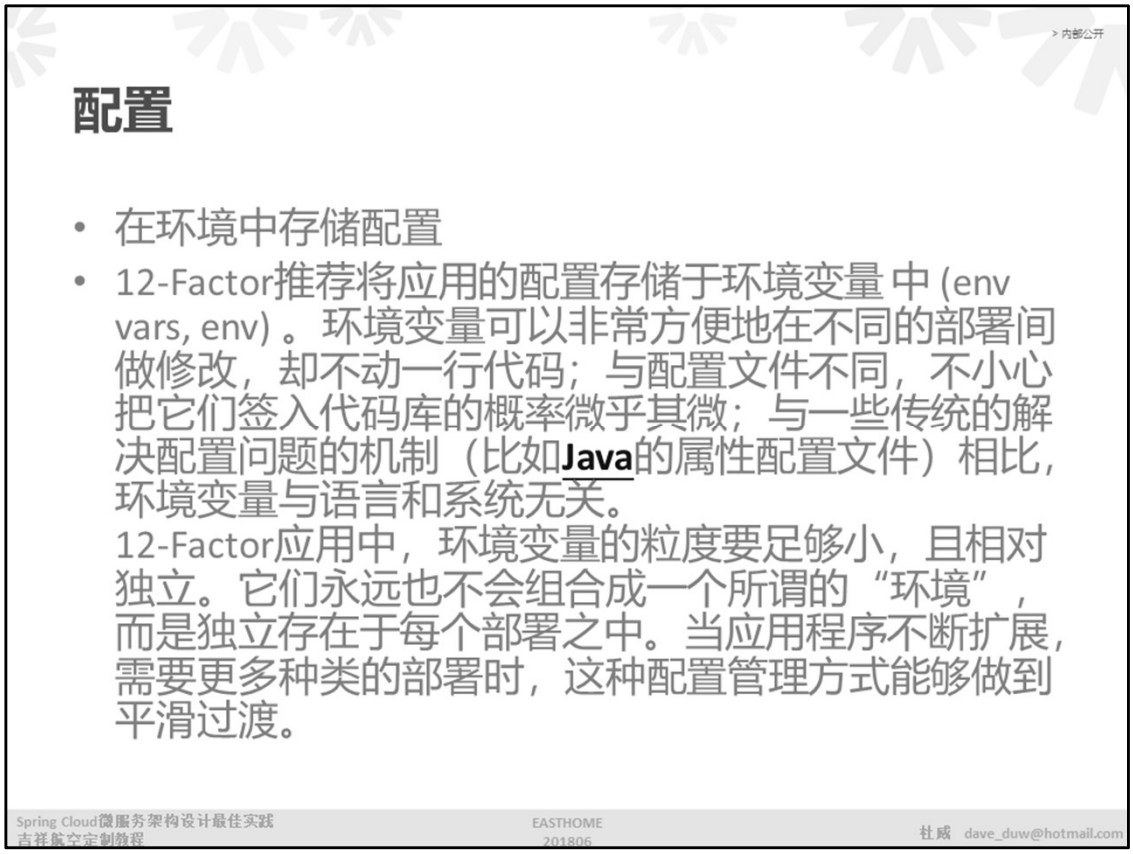
- 一份基准代码，多份部署
- 基准代码和应用之间总是保持一一对应的关系：
  一旦有多个基准代码，就不能称为一个应用，而是一个分布式系统。分布式系统中的每一个组件都是一个应用，每一个应用可以分别使用12-Factor进行开发。
  多个应用共享一份基准代码是有悖于12-Factor原则的。
  解决方案是将共享的代码拆分为独立的类库，然后使用依赖管理策略去加载它们。尽管每个应用只对应一份基准代码，但可以同时存在多份部署。所有部署的基准代码相同，但每份部署可以使用其不同的版本。

Notes:

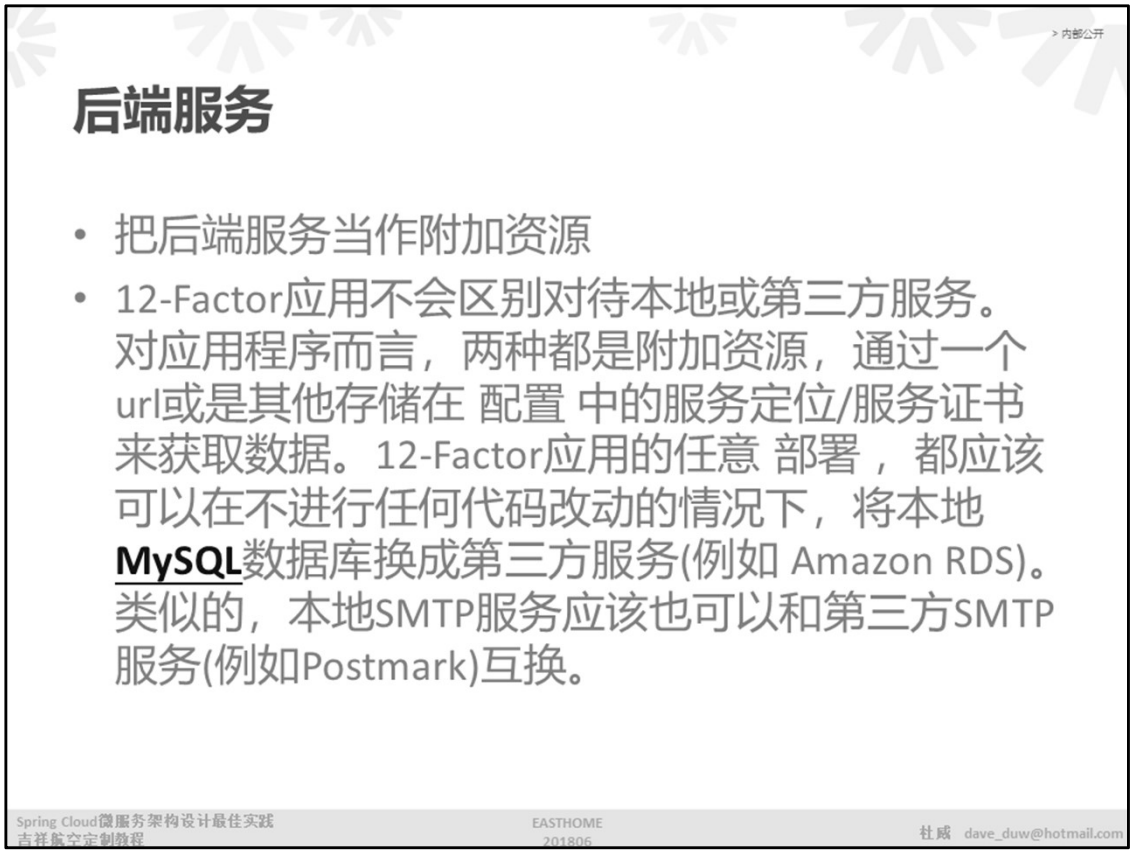
依赖

- 显式声明依赖关系
- 12-Factor规则下的应用程序不会隐式依赖系统级的类库。 它一定通过依赖清单， 确切地声明所有依赖项。此外，在运行过程中通过 依赖隔离 工具来确保程序不会调用系统中存在但清单中未声明的依赖项。这一做法会统一应用到生产和开发环境。

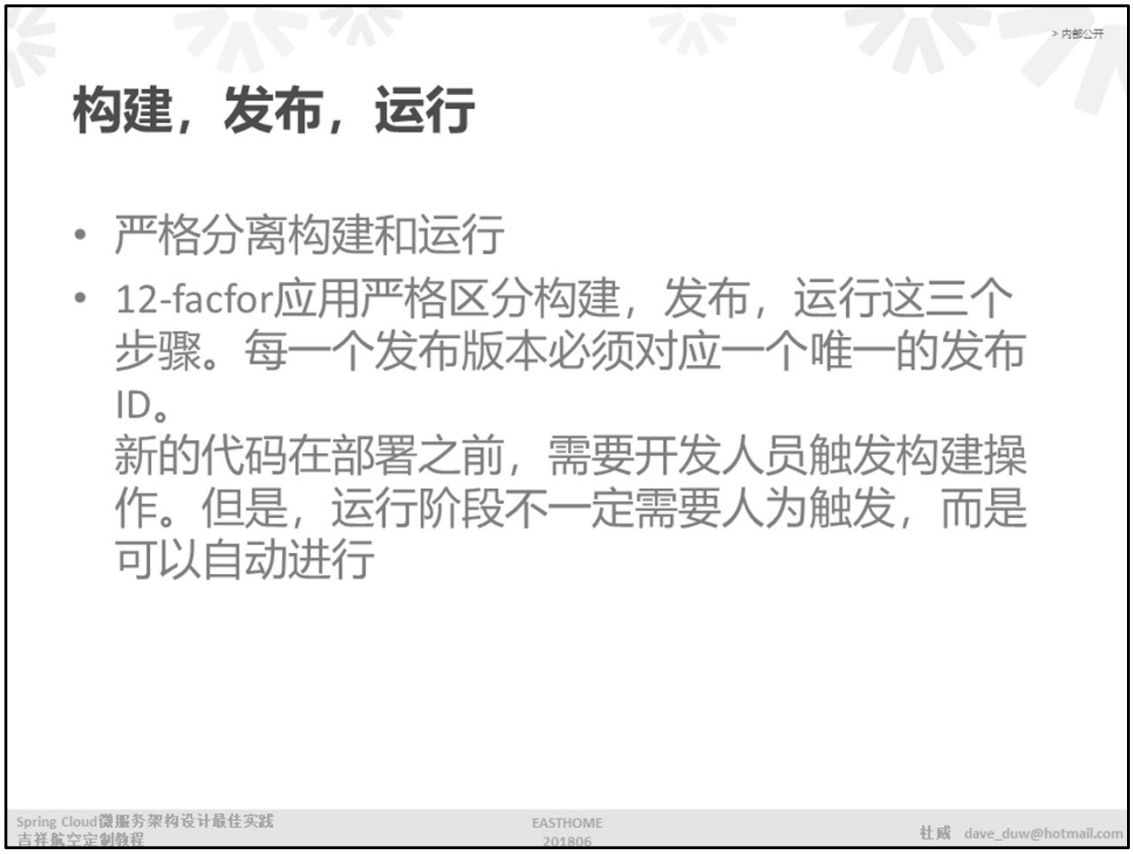
Notes:


> 内部公开
配置
• 在环境中存储配置
• 12-Factor推荐将应用的配置存储于环境变量中(env vars, env)。环境变量可以非常方便地在不同的部署间做修改，却不动一行代码；与配置文件不同，不小心把它们签入代码库的概率微乎其微；与一些传统的解决配置问题的机制（比如Java的属性配置文件）相比，环境变量与语言和系统无关。
12-Factor应用中，环境变量的粒度要足够小，且相对独立。它们永远也不会组合成一个所谓的“环境”，而是独立存在于每个部署之中。当应用程序不断扩展，需要更多种类的部署时，这种配置管理方式能够做到平滑过渡。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
后端服务
• 把后端服务当作附加资源
• 12-Factor应用不会区别对待本地或第三方服务。对应用程序而言，两种都是附加资源，通过一个url或是其他存储在 配置 中的服务定位/服务证书来获取数据。12-Factor应用的任意 部署 ， 都应该可以在不进行任何代码改动的情况下，将本地MySQL数据库换成第三方服务(例如 Amazon RDS)。类似的，本地SMTP服务应该也可以和第三方SMTP服务(例如Postmark)互换。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

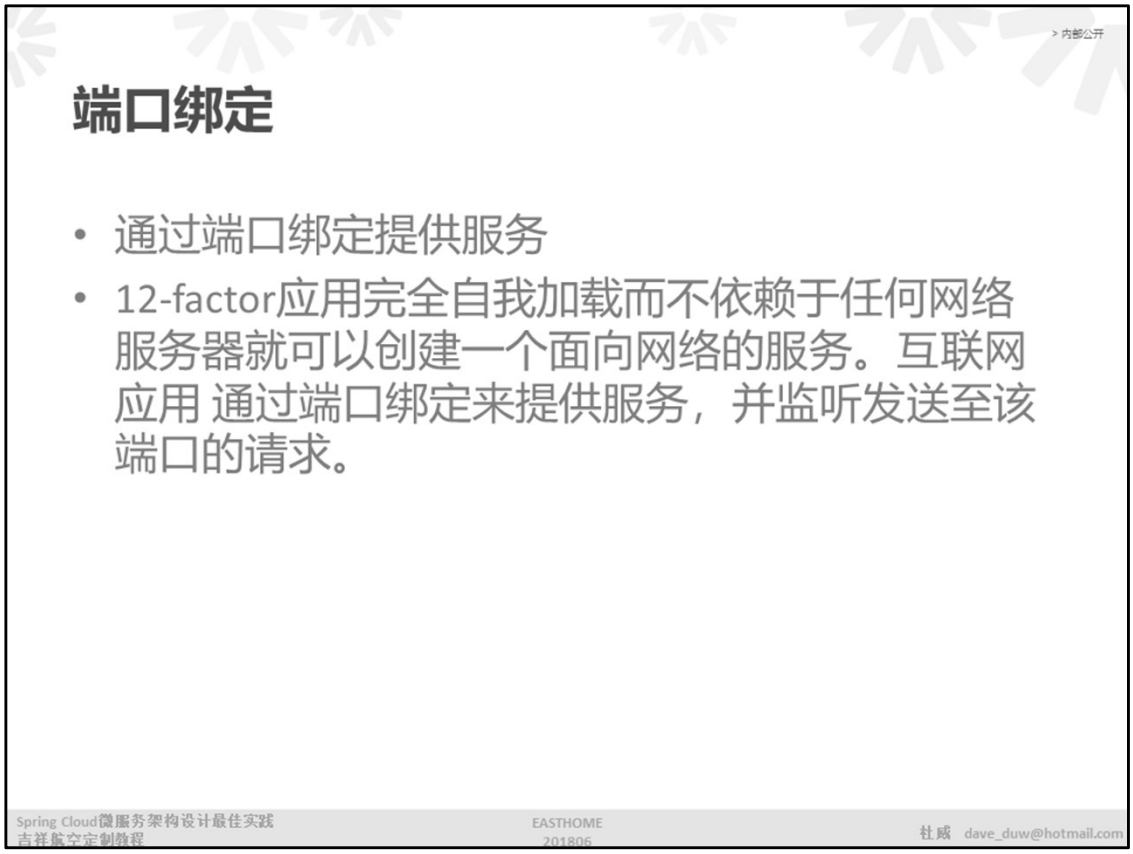
Notes:


> 内部公开
构建，发布，运行
• 严格分离构建和运行
• 12-facfor应用严格区分构建，发布，运行这三个步骤。每一个发布版本必须对应一个唯一的发布ID。
新的代码在部署之前，需要开发人员触发构建操作。但是，运行阶段不一定需要人为触发，而是可以自动进行
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
进程
• 以一个或多个无状态进程运行应用
• 12-factor应用的进程必须无状态且无共享 。任何需要持久化的数据都要存储在后端服务内，比如数据库。粘性Session是twelve-factor极力反对的。Session中的数据应该保存在诸如Memcached或 Redis 这样的带有过期时间的缓存中。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
端口绑定
• 通过端口绑定提供服务
• 12-factor应用完全自我加载而不依赖于任何网络服务器就可以创建一个面向网络的服务。互联网应用 通过端口绑定来提供服务，并监听发送至该端口的请求。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

并发

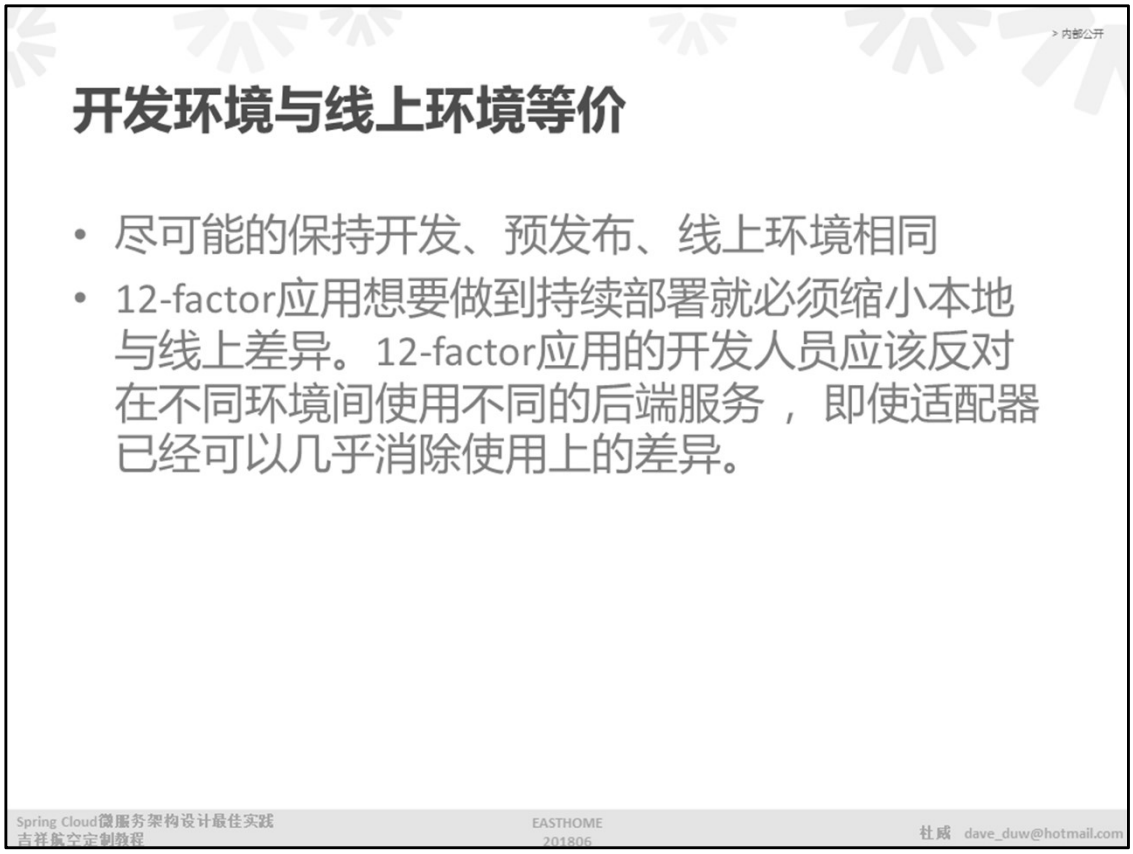
- 通过进程模型进行扩展
- 在12-factor应用中，进程是一等公民。 12-factor应用的进程主要借鉴于 unix守护进程模型 。开发人员可以运用这个模型去设计应用**架构**，将不同的工作分配给不同的进程类型 。

Notes:

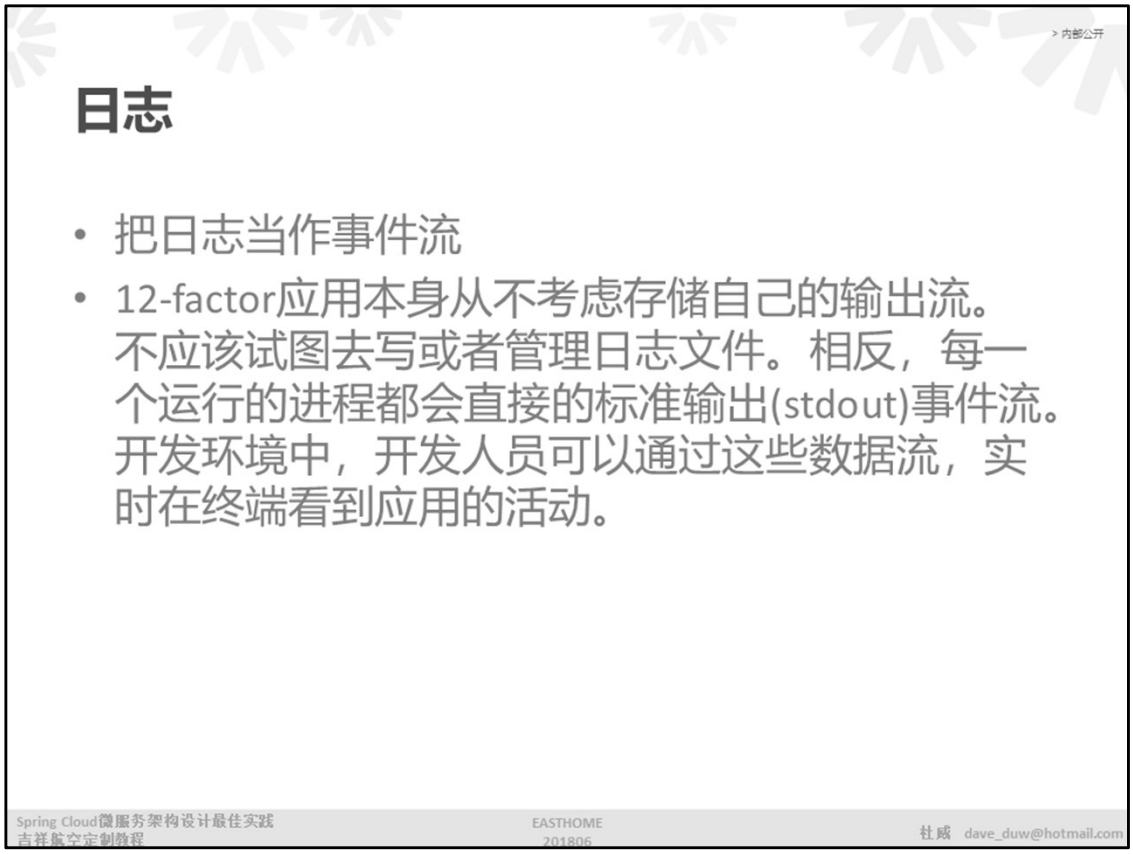
易处理

- 快速启动和优雅终止可最大化健壮性
- 12-factor应用的进程是可支配的，意思是说它们可以瞬间开启或停止。 这有利于快速、弹性的伸缩应用，迅速部署变化的代码或配置，稳健地部署应用。进程应当追求最小启动时间；进程一旦接收终止信号(SIGTERM) 就会优雅的终止 。进程还应当在面对突然死亡时保持健壮 。

Notes:


> 内部公开
开发环境与线上环境等价
• 尽可能的保持开发、预发布、线上环境相同
• 12-factor应用想要做到持续部署就必须缩小本地与线上差异。12-factor应用的开发人员应该反对在不同环境间使用不同的后端服务，即使适配器已经可以几乎消除使用上的差异。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
日志
• 把日志当作事件流
• 12-factor应用本身从不考虑存储自己的输出流。不应该试图去写或者管理日志文件。相反，每一个运行的进程都会直接的标准输出(stdout)事件流。开发环境中，开发人员可以通过这些数据流，实时在终端看到应用的活动。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

管理进程

- 后台管理任务当作一次性进程运行
- 一次性管理进程应该和正常的 常驻进程 使用同样的环境。这些管理进程和任何其他的进程一样使用相同的代码和配置，基于某个发布版本运行。后台管理代码应该随其他应用程序代码一起发布，从而避免同步问题。所有进程类型应该使用同样的依赖隔离技术。12-factor尤其青睐那些提供了REPL shell的语言，因为那会让运行一次性脚本变得简单。

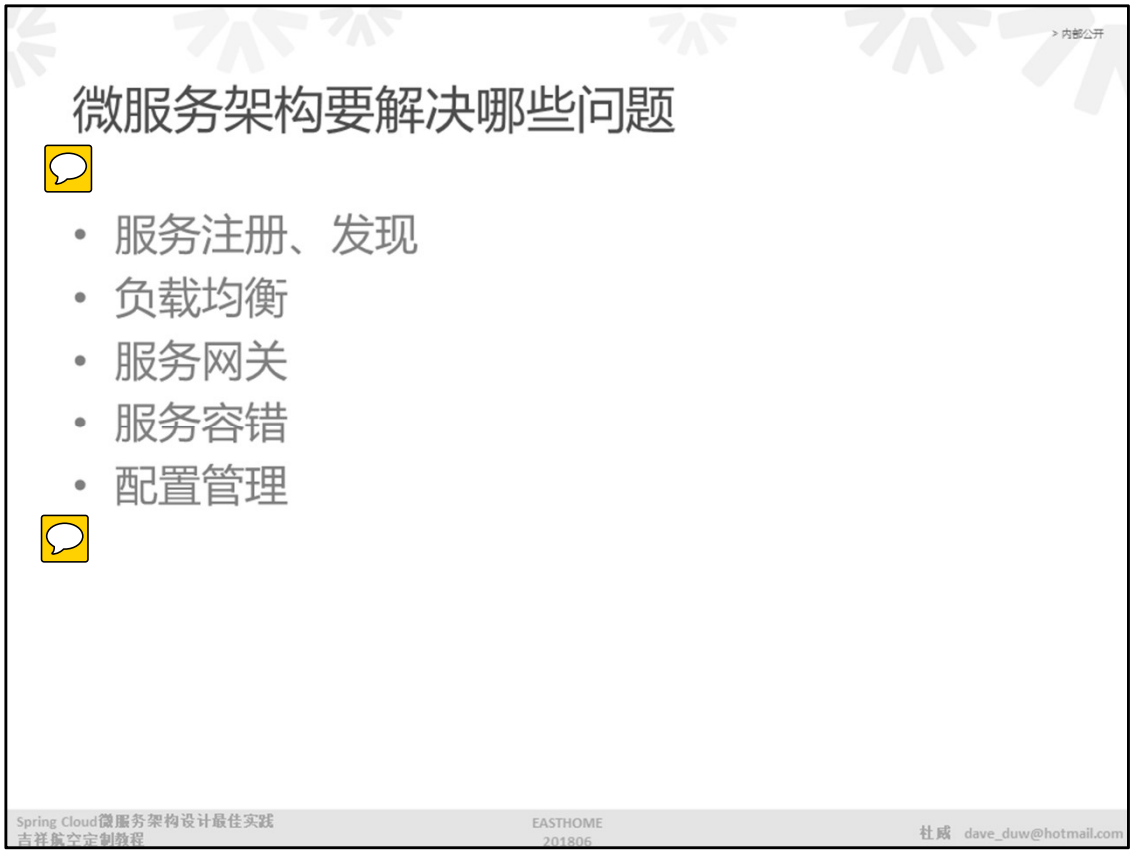
Notes:


> 内部公开
6
微服务架构实现
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
微服务设计实践
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
微服务架构要解决哪些问题
• 服务注册、发现
• 负载均衡
• 服务网关
• 服务容错
• 配置管理
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

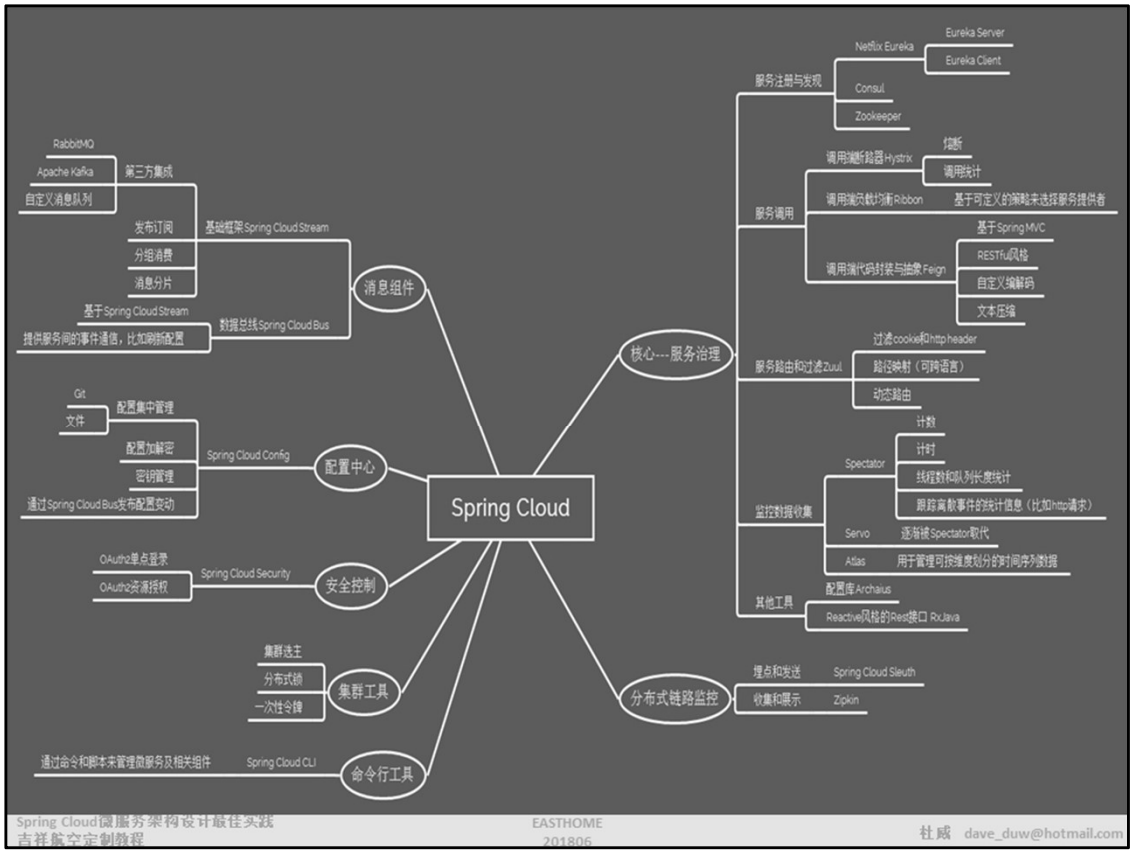
Notes:


> 内部公开
服务门户
配置查看／推送
存活监控
降级
调用分析图表
限流
服务性能数据收集
负载均衡
服务调用
服务寻址
服务发布
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
社威 dave_duw@hotmail.com

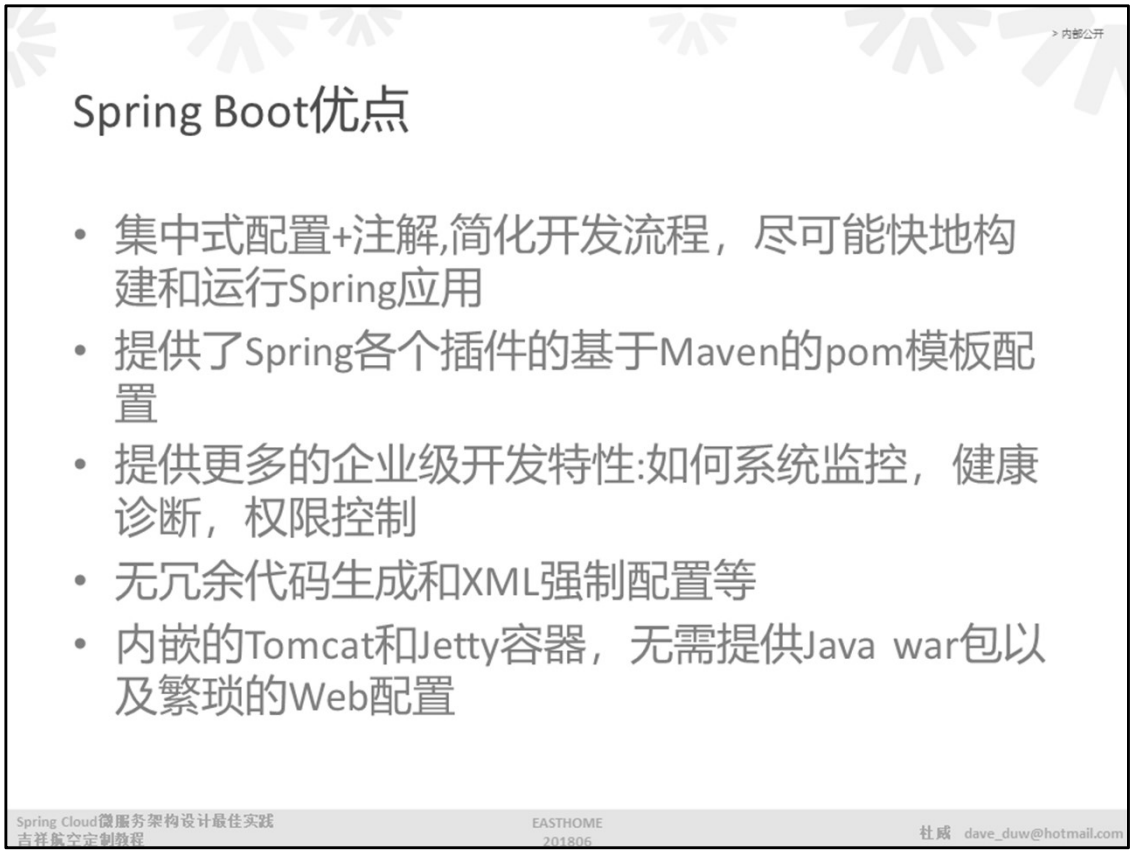
Notes:


> 内部公开
SPRING CLOUD中常见的微服务公共组件
负载均衡
服务注册与发现
监控
分布式配置管理
API网关
分布式追踪
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

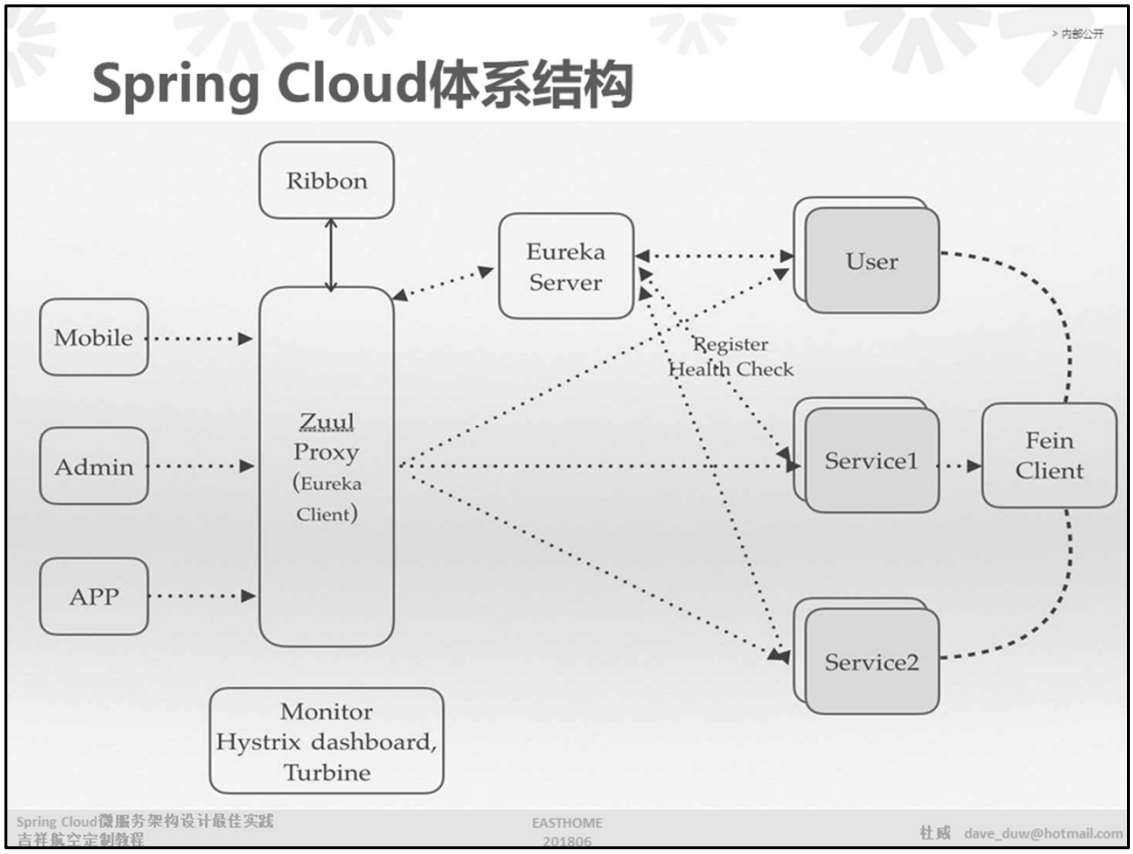
Notes:


Spring Cloud
核心---服务治理
服务注册与发现
Netflix Eureka
Eureka Server
Eureka Client
Consul
Zookeeper
服务调用
调用熔断器Hystrix
熔断
调用统计
调用端负载均衡Ribbon
基于可定义的策略来选择服务提供者
调用端代码封装与抽象Feign
基于Spring MVC
RESTful风格
自定义编解码
文本压缩
服务路由和过滤zuul
过滤cookie和http header
路径映射（可跨语言）
动态路由
监控数据收集
Spectator
计数
计时
线程数和队列长度统计
跟踪离散事件的统计信息（比如http请求）
Servo
逐渐被Spectator取代
Atlas
用于管理可按维度划分的时间序列数据
其他工具
配置库Archaius
Reactive风格的Rest接口 RxJava
分布式链路监控
埋点和发送
Spring Cloud Sleuth
收集和展示
Zipkin
消息组件
RabbitMQ
Apache Kafka
自定义消息队列
第三方集成
基础框架Spring Cloud Stream
发布订阅
分组消费
消息分片
数据总线Spring Cloud Bus
基于Spring Cloud Stream
提供服务间的事件通信，比如刷新配置
配置中心
Git
文件
配置集中管理
Spring Cloud Config
配置加解密
密钥管理
通过Spring Cloud Bus发布配置变动
安全控制
OAuth2单点登录
OAuth2资源授权
Spring Cloud Security
集群工具
集群选主
分布式锁
一次性令牌
命令行工具
通过命令和脚本来管理微服务及相关组件
Spring Cloud CLI
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

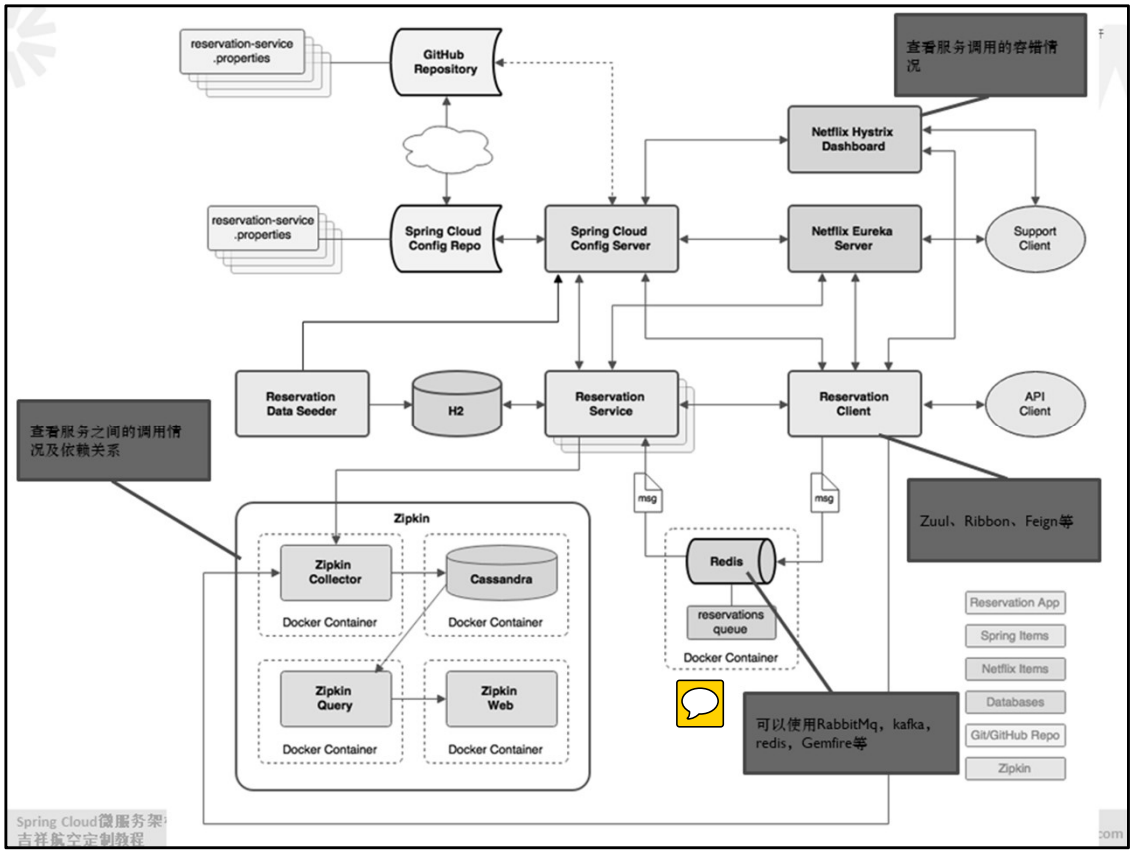
Notes:


> 内部公开
Spring Boot优点
• 集中式配置+注解,简化开发流程，尽可能快地构建和运行Spring应用
• 提供了Spring各个插件的基于Maven的pom模板配置
• 提供更多的企业级开发特性:如何系统监控，健康诊断，权限控制
• 无冗余代码生成和XML强制配置等
• 内嵌的Tomcat和Jetty容器，无需提供Java war包以及繁琐的Web配置
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

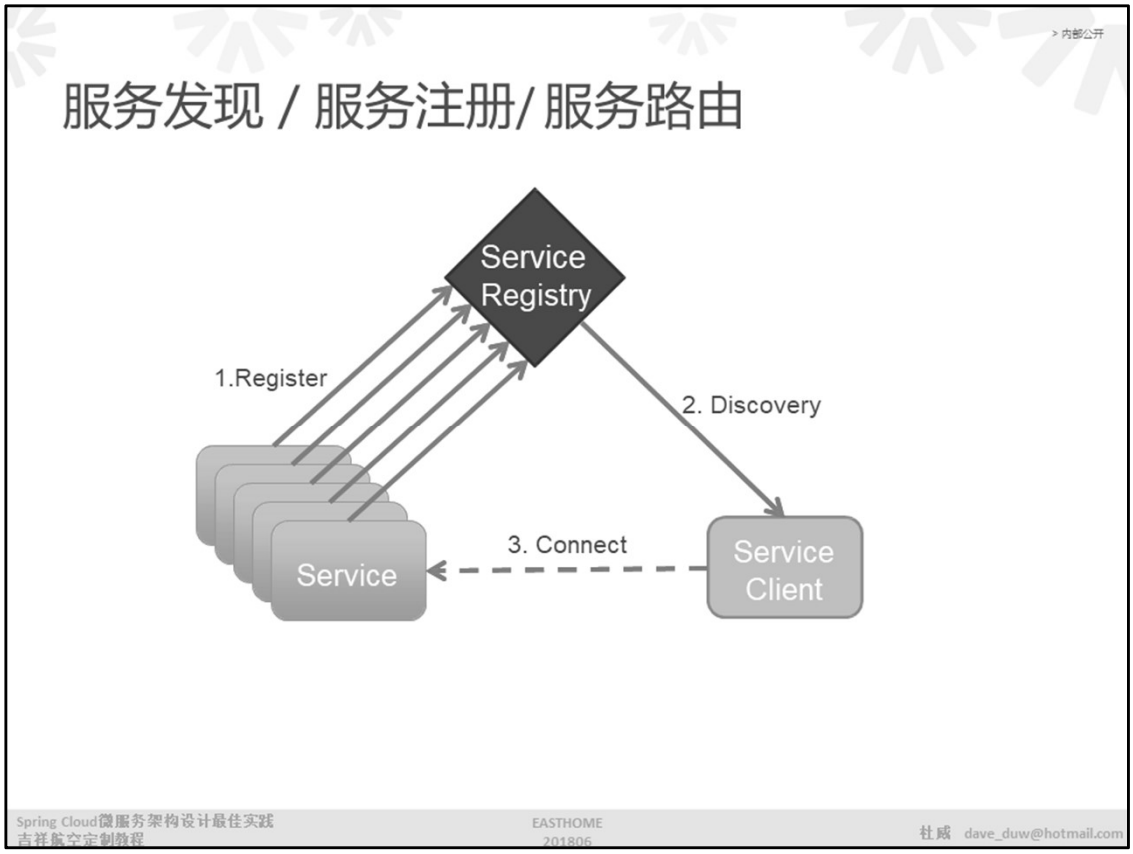
Notes:


> 内部公开
Spring Cloud体系结构
Ribbon
Eureka Server
User
Mobile
Admin
APP
Zuul Proxy (Eureka Client)
Register
Health Check
Service1
Fein Client
Service2
Monitor Hystrix dashboard, Turbine
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

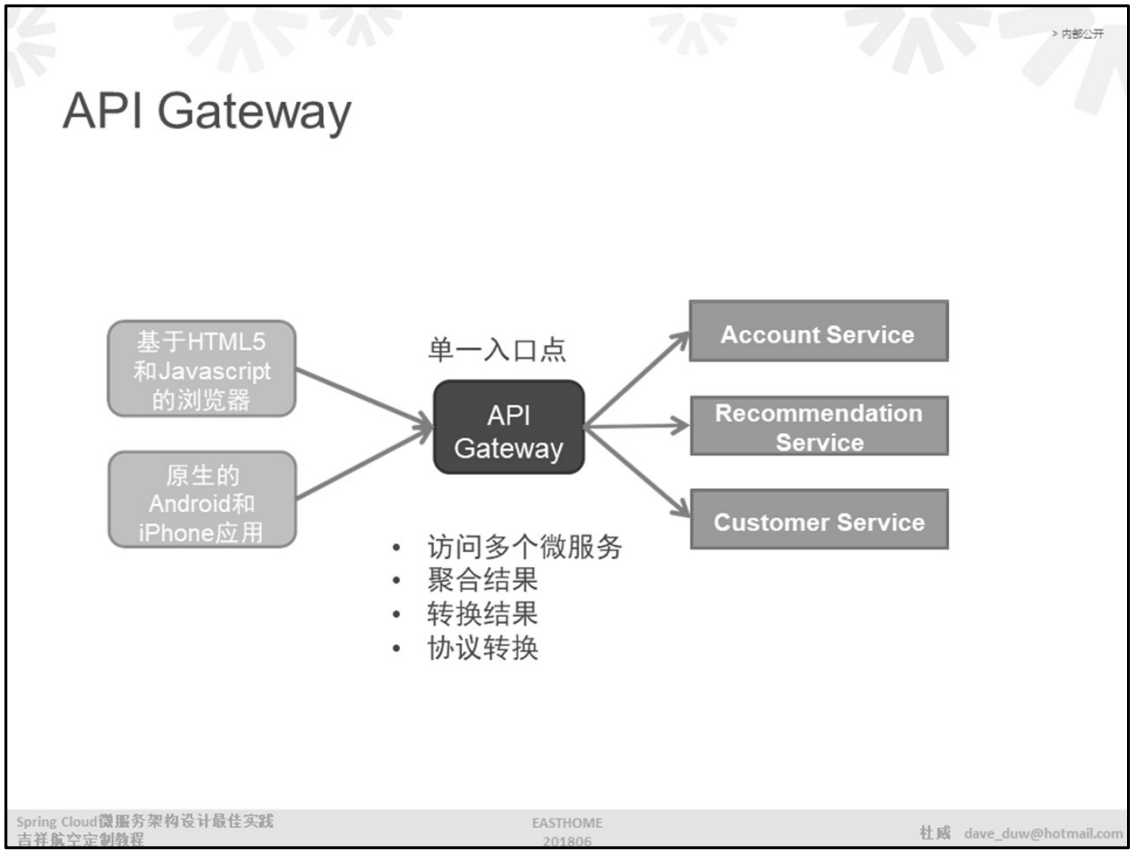
Notes:


reservation-service .properties
GitHub Repository
查看服务调用的容错情况
Netflix Hystrix Dashboard
reservation-service .properties
Spring Cloud Config Repo
Spring Cloud Config Server
Netflix Eureka Server
Support Client
Reservation Data Seeder
H2
Reservation Service
Reservation Client
API Client
查看服务之间的调用情况及依赖关系
msg
msg
Zuul、Ribbon、Feign等
Zipkin
Zipkin Collector
Cassandra
Docker Container
Docker Container
Redis
reservations queue
Docker Container
Reservation App
Spring Items
Netflix Items
Databases
Git/GitHub Repo
Zipkin
Zipkin Query
Zipkin Web
Docker Container
Docker Container
可以使用RabbitMq，kafka，redis，Gemfire等
Spring Cloud微服务架
吉祥航空定制教程

Notes:

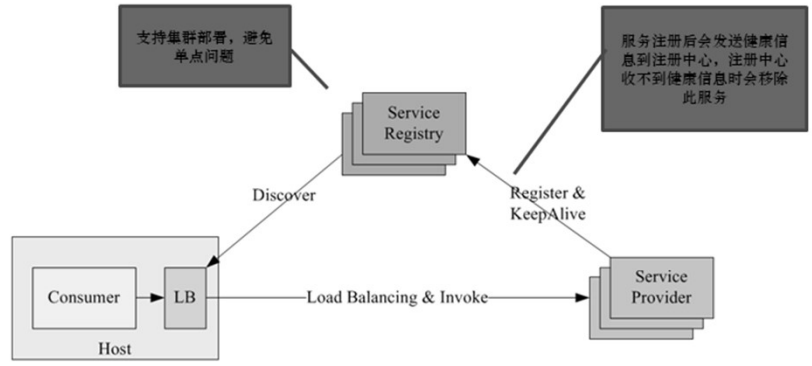

> 内部公开
服务发现 / 服务注册/ 服务路由
Service Registry
1.Register
2. Discovery
3. Connect
Service
Service Client
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
API Gateway
基于HTML5和Javascript的浏览器
原生的Android和iPhone应用
单一入口点
API Gateway
Account Service
Recommendation Service
Customer Service
• 访问多个微服务
• 聚合结果
• 转换结果
• 协议转换
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

服务注册、发现

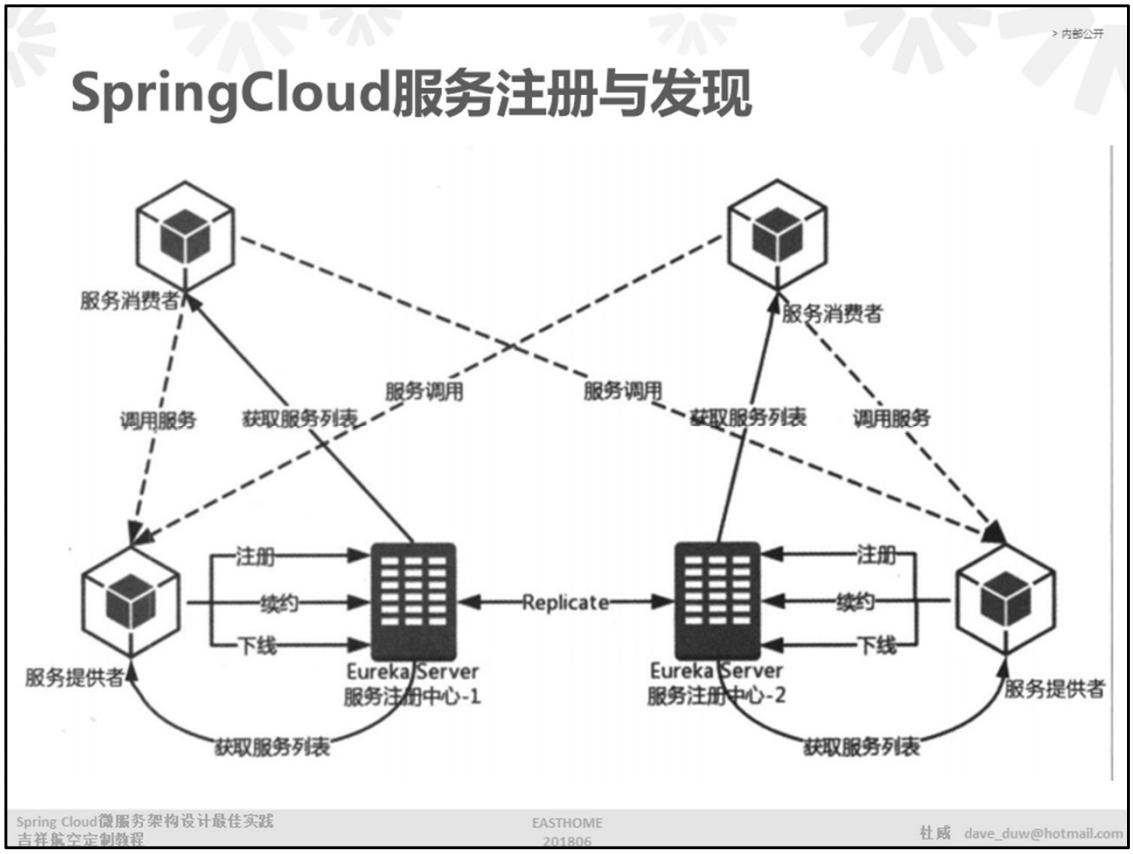


和单体(Monolithic)架构不同，微服务架构是由一系列职责单一的细粒度服务构成的分布式网状结构，服务之间通过轻量机制进行通信，这时候必然引入一个服务注册发现问题，也就是说服务提供方要注册通告服务地址，服务的调用方要能发现目标服务

Notes:


> 内部公开
• 高效服务调用
简单高效的RPC服务调用框架，通过IDL定义数据结构，在服务端、客户端之间通过二进制传输数据，体积更小，序列化和反序列化更高效；
Thrift
目前流行的服务调用方式有很多种，例如基于 SOAP 消息格式的 Web Service，基于 JSON 消息格式的 RESTful 服务等。其中所用到的数据传输方式包括 XML，JSON 等，然而 XML 相对体积太大，传输效率低，JSON 体积较小，新颖，但还不够完善。由 Facebook 开发的远程服务调用框架 Apache Thrift，采用接口描述语言定义并创建服务，支持可扩展的跨语言服务开发，所包含的代码生成引擎可以在多种语言中，如 C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk 等创建高效的、无缝的服务，其传输数据采用二进制格式，相对 XML 和 JSON 体积更小，对于高并发、大数据量和多语言的环境更有优势
Thrift 可以很好的与spring cloud集成，方便使用!
Protocol Buffers
Protocol Buffers是Google公司开发的一种数据描述语言可用于数据存储、通信协议等方面,它不依赖于语言和平台并且可扩展性极强。
gRPC是一个高性能、通用的开源RPC框架，其由Google主要面向移动应用开发并基于HTTP/2协议标准而设计，基于ProtoBuf(Protocol Buffers)序列化协议开发，且支持众多开发语言。
gRPC已经应用在Google的云服务和对外提供的API中，其主要应用场景如下：
低延迟、高扩展性、分布式的系统
同云服务器进行通信的移动应用客户端
设计语言独立、高效、精确的新协议
便于各方面扩展的分层设计，如认证、负载均衡、日志记录、监控等
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

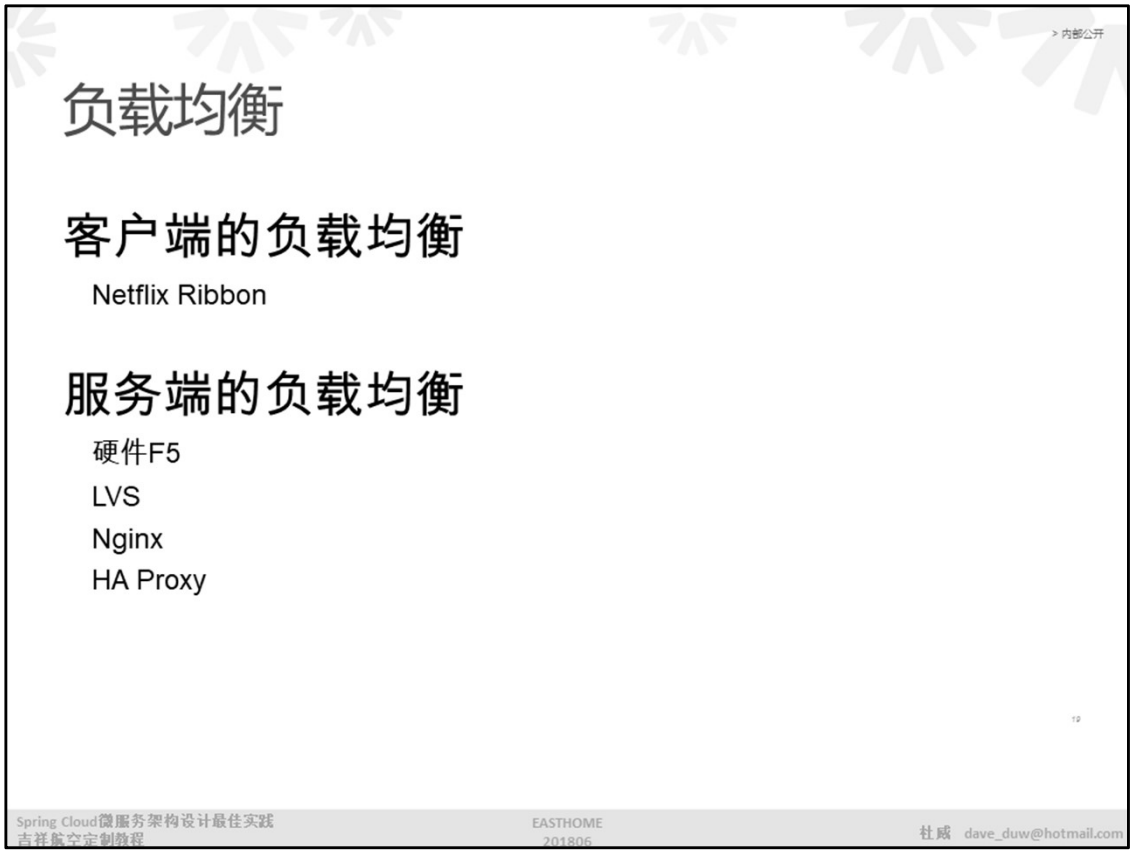

> 内部公开
SpringCloud服务注册与发现
服务消费者
服务消费者
调用服务
获取服务列表
服务调用
服务调用
获取服务列表
调用服务
注册
续约
下线
Replicate
注册
续约
下线
服务提供者
Eureka Server
服务注册中心-1
Eureka Server
服务注册中心-2
服务提供者
获取服务列表
获取服务列表
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
自我保护
心跳失败的比例在15分钟之内低于85%时，Eureka Server会将当前的实例注册信息保护起来，让这些实例不会过期。
服务注册中心
失效剔除
默认每隔一段时间（默认60秒）将当前清单中超时（默认为90秒）没有续约的服务剔除出去
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

Notes:

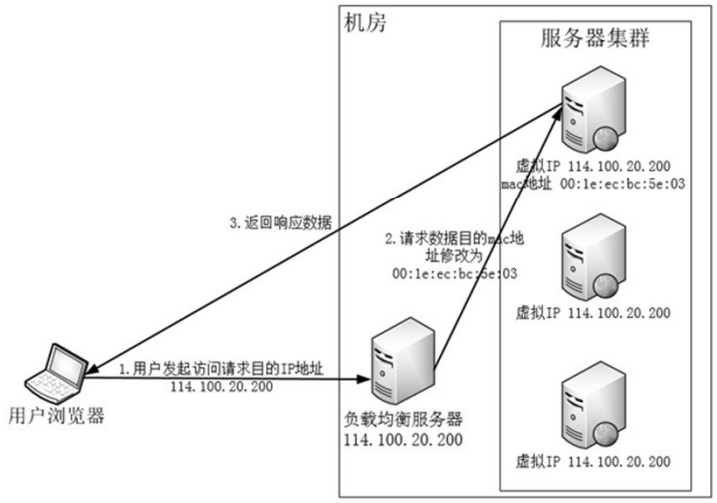

> 内部公开
负载均衡
客户端的负载均衡
Netflix Ribbon
服务端的负载均衡
硬件F5
LVS
Nginx
HA Proxy
10
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

负载均衡

数据链路层负载均衡优点

- 不需要修改数据包的源地址
- 响应数据不需要通过负载均衡器



Notes:

> 内部公开

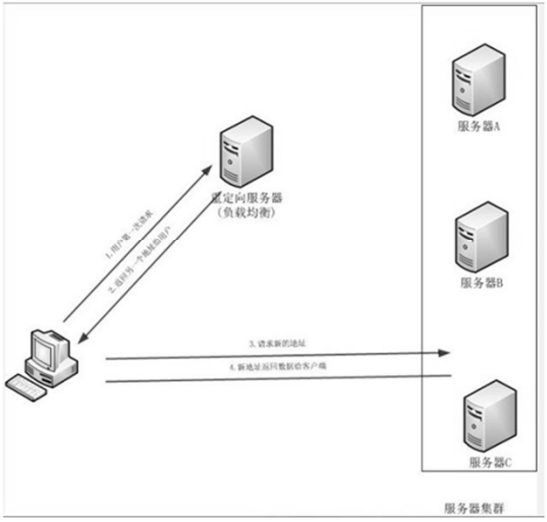
负载均衡-重定向负载均衡

HTTP重定向负载均衡优点

- 比较简单

HTTP重定向负载均衡缺点

- 浏览器需要两次请求服务器才能完成一次访问
- 性能较差
- 重定向服务器自身的处理能力有可能成为瓶颈

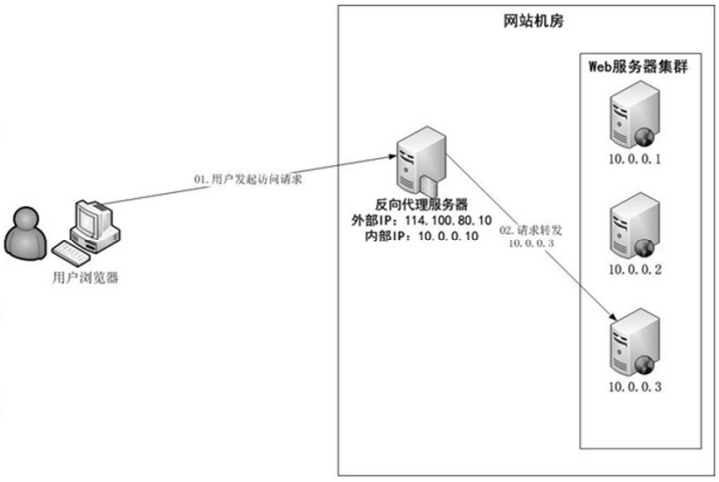


Notes:

负载均衡

反向代理负载均衡优点
- 比较简单
- 可以利用反向代理缓存资源
- 改善网站性能

反向代理负载均衡缺点
- 所有请求和响应的中转站
- 其性能可能会成为瓶颈

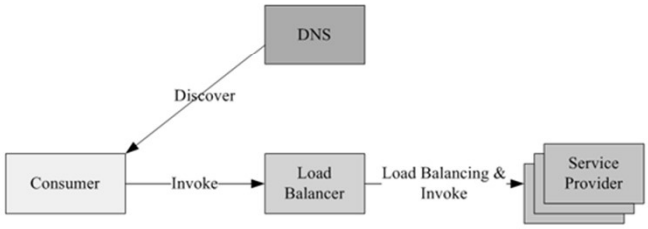


Notes:

负载均衡

- 集中式负载均衡

在服务消费者和服务提供者之间有一个独立的LB，LB通常是专门的硬件设备如F5，或者基于软件如LVS，HAproxy等实现

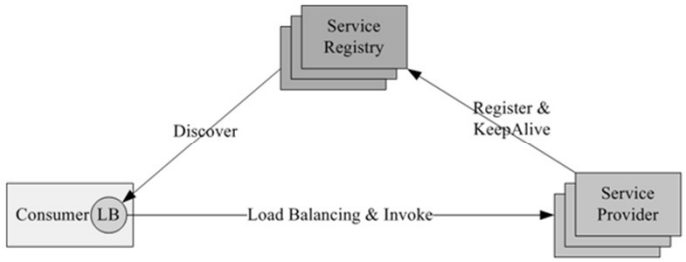


1. 单点问题
2. 所有服务调用流量都经过LB，当服务数量和调用量大的时候，LB容易成为瓶颈
3. LB在服务消费方和服务提供方之间增加了一跳(hop)，有一定性能开销。

Notes:

负载均衡

- 进程内LB

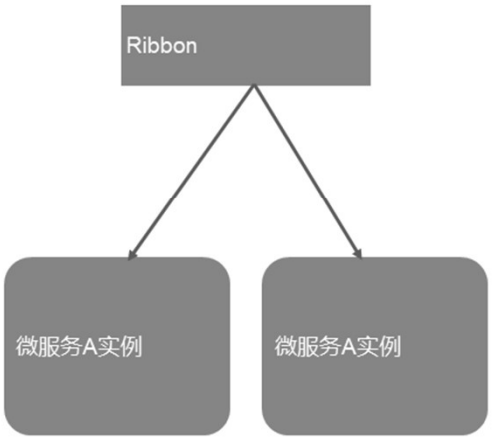


进程内LB方案是一种分布式方案，LB和服务发现能力被分散到每一个服务消费者的进程内部，同时服务消费方和服务提供方之间是直接调用，没有额外开销，性能比较好

Notes:

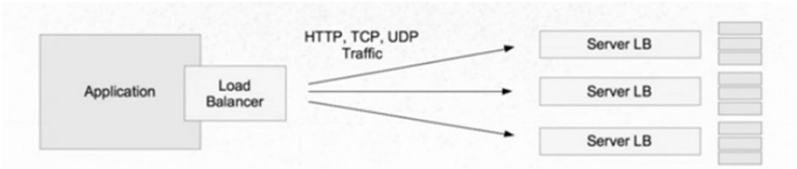
Ribbon

- **负载均衡** Zuul网关将一个请求发送给某一个服务的应用的时候，如果一个服务启动了多个实例，就会通过Ribbon来通过一定的负载均衡策略来发送给某一个服务实例。



Notes:

客户端的负载均衡 - NETFLIX RIBBON




```
2017-01-19 02:41:38.425  INFO 7 --- [nio-8080-exec-1] c.n.l.DynamicServerListLoadBalancer      :
DynamicServerListLoadBalancer for client sample-hystrix-aggregate initialized: DynamicServerListLoadBalancer:
{NFLoadBalancer:name=sample-hystrix-aggregate,current list of Servers=[sampleaggregate2:8081, sampleaggregate1:8081],Load
balancer stats=Zone stats: {defaultzone=[Zone:defaultzone;Instance count:2;Active connections count: 0;Circuit breaker
tripped count: 0;Active connections per server: 0.0;]
},Server stats: [[Server:sampleaggregate2:8081;Zone:defaultZone;Total Requests:0;Successive connection failure:0;Total
blackout seconds:0;Last connection made:Thu Jan 01 00:00:00 UTC 1970;First connection made: Thu Jan 01 00:00:00 UTC 1970;
Active Connections:0;total failure count in last (1000) msecs:0;average resp time:0.0;90 percentile resp time:0.0;95
percentile resp time:0.0;min resp time:0.0;max resp time:0.0;stddev resp time:0.0]
, [Server:sampleaggregate1:8081;Zone:defaultZone;Total Requests:0;Successive connection failure:0;Total blackout
seconds:0;Last connection made:Thu Jan 01 00:00:00 UTC 1970;First connection made: Thu Jan 01 00:00:00 UTC 1970;Active
Connections:0;total failure count in last (1000) msecs:0;average resp time:0.0;90 percentile resp time:0.0;95 percentile
resp time:0.0;min resp time:0.0;max resp time:0.0;stddev resp time:0.0]
]}ServerList:org.springframework.cloud.netflix.ribbon.eureka.DomainExtractingServerList@3e03fb88
```

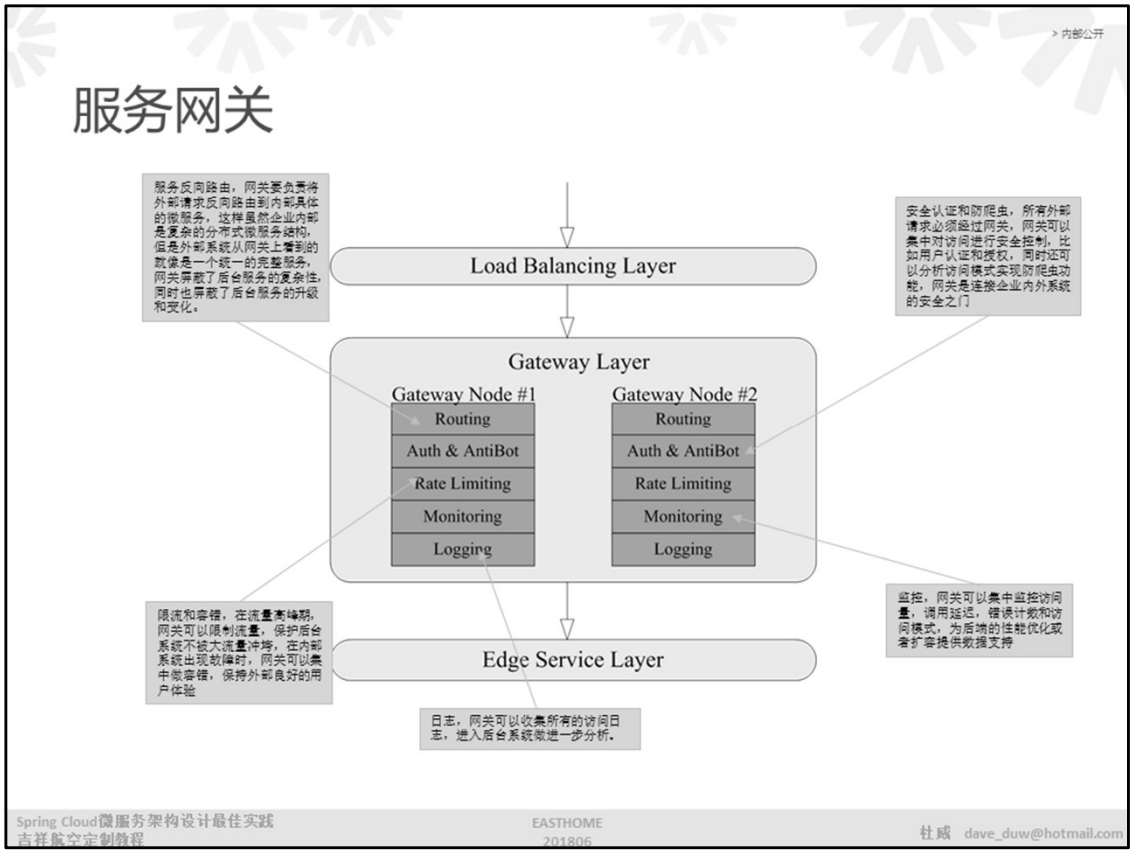
Notes:


> 内部公开
Feign
服务客户端 服务之间如果需要相互访问，可以使用RestTemplate，也可以使用Feign客户端访问。它默认会使用Ribbon来实现负载均衡
@HystrixCommand(fallbackMethod = "hiError")
public String sayHi(String name) {
return restTemplate.getForObject(url: "http://SERVICE-HI/hi?name=" + name, String.class);
}
@FeignClient(value = "service-hi", fallback = ScheduleServiceHiHystric.class)
public interface ScheduleServiceHi {
@RequestMapping(value = "/hi", method = RequestMethod.GET)
String sayHi(@RequestParam("name") String name);
}
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

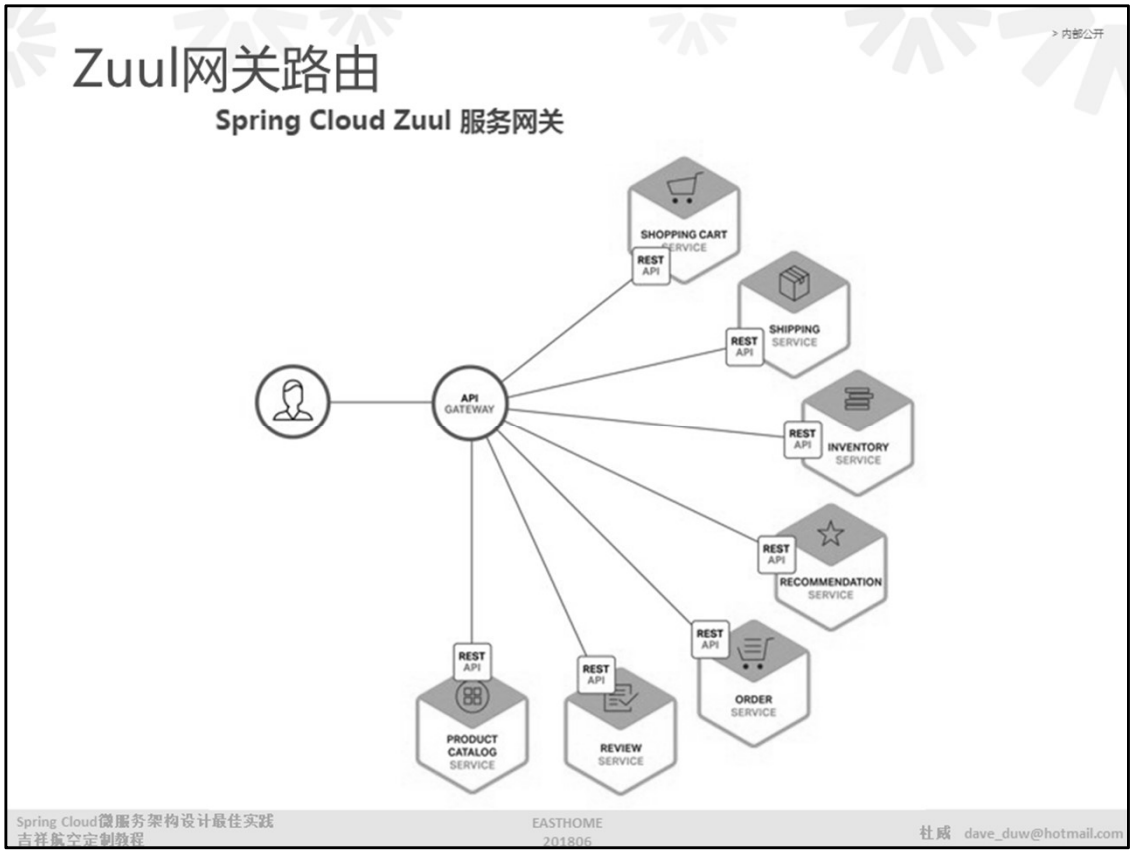
Notes:

为什么要引入API网关?

Notes:


> 内部公开
服务网关
服务反向路由，网关要负责将外部请求反向路由到内部具体的微服务，这样虽然企业内部是复杂的分布式微服务结构，但是外部系统从网关上看到的就像是一个统一的完整服务，网关屏蔽了后台服务的复杂性，同时也屏蔽了后台服务的升级和变化。
Load Balancing Layer
安全认证和防爬虫，所有外部请求必须经过网关，网关可以集中对访问进行安全控制，比如用户认证和授权，同时还可以分析访问模式实现防爬虫功能，网关是连接企业内外系统的安全之门
Gateway Layer
Gateway Node #1
Routing
Auth & AntiBot
Rate Limiting
Monitoring
Logging
Gateway Node #2
Routing
Auth & AntiBot
Rate Limiting
Monitoring
Logging
限流和容错，在流量高峰期，网关可以限制流量，保护后台系统不被大流量冲垮，在内部系统出现故障时，网关可以集中做容错，保持外部良好的用户体验
监控，网关可以集中监控访问量，调用延迟，错误计数和访问模式，为后端的性能优化或者扩容提供数据支持
Edge Service Layer
日志，网关可以收集所有的访问日志，进入后台系统做进一步分析。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

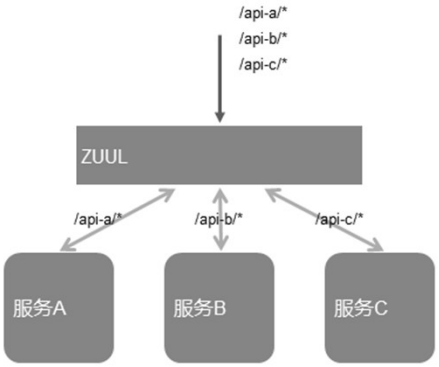

> 内部公开
Zuul网关路由
Spring Cloud Zuul 服务网关
SHOPPING CART SERVICE
REST API
SHIPPING SERVICE
API GATEWAY
INVENTORY SERVICE
RECOMMENDATION SERVICE
ORDER SERVICE
PRODUCT CATALOG SERVICE
REVIEW SERVICE
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

> 内部公开

ZUUL API Gateway

- **API网关** 所有的客户端请求通过这个网关访问后台的服务。他可以使用一定的路由配置来判断某一个URL由哪个服务来处理。并从Eureka获取注册的服务来转发请求。

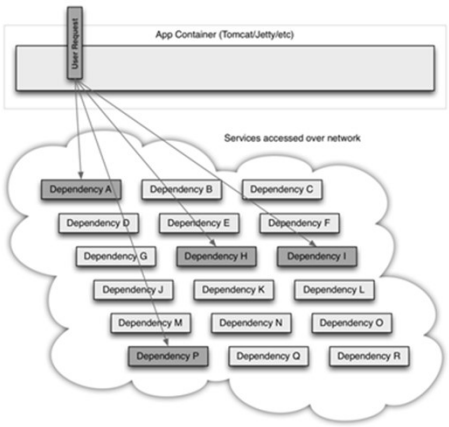


Notes:

服务容错

- 服务之间相互依赖

当企业微服务化以后，服务之间会有错综复杂的依赖关系，例如，一个前端请求一般会依赖于多个后端服务，技术上称为1 -> N扇出。

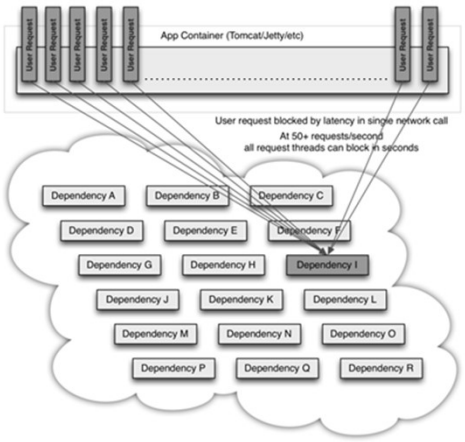


Notes:

服务容错

- 单服务异常导致雪崩

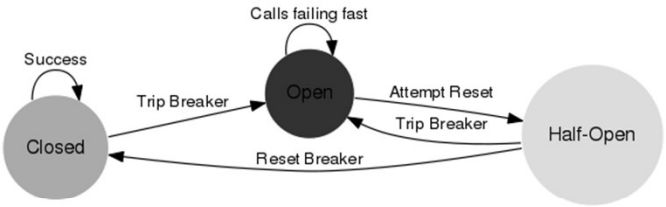
在实际生产环境中，服务往往不是百分百可靠，服务可能会出错或者产生延迟，如果一个应用不能对其依赖的故障进行容错和隔离，那么该应用本身就处在被拖垮的风险中。在一个高流量的网站中，某个单一后端一旦发生延迟，可能在数秒内导致所有应用资源(线程，队列等)被耗尽，造成所谓的雪崩效应(Cascading Failure)，严重时可致整个网站瘫痪。



Notes:

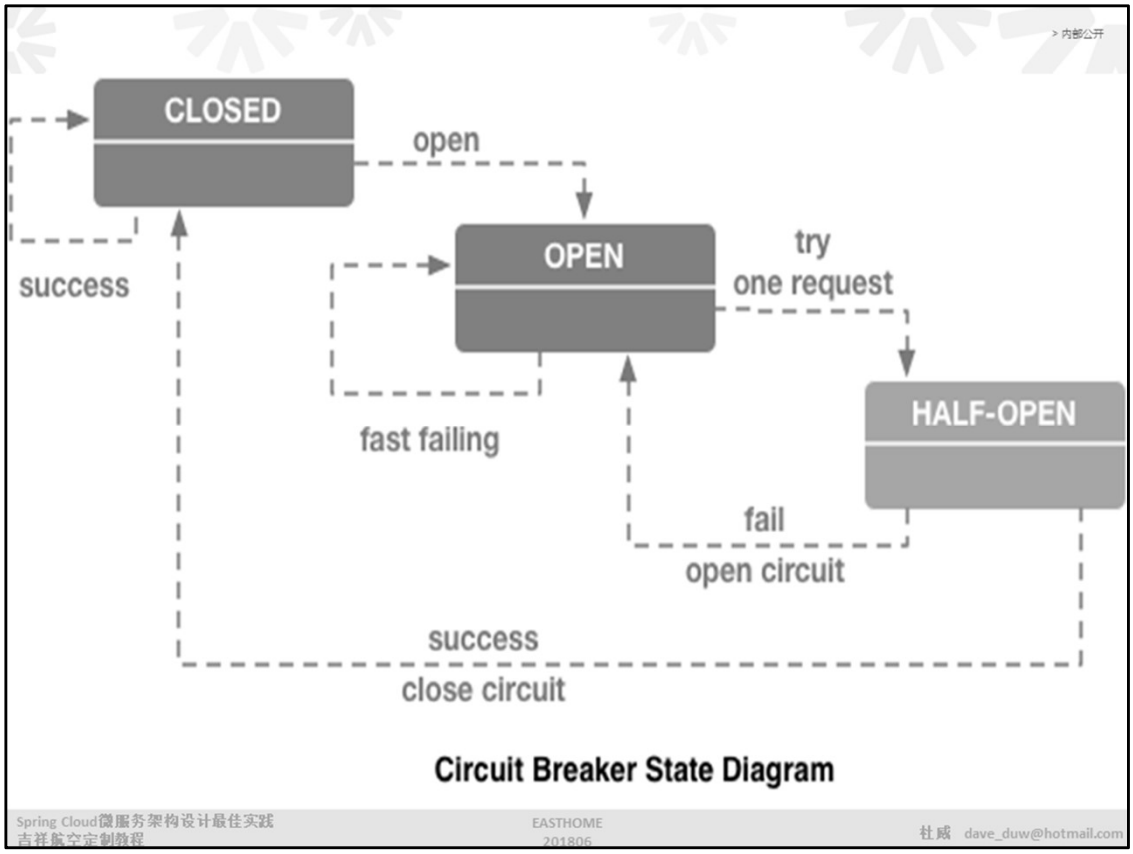
服务容错-最佳实践

- 电路熔断器模式(Circuit Breaker Patten)



该模式的原理类似于家里的电路熔断器，如果家里的电路发生短路，熔断器能够主动熔断电路，以避免灾难性损失。在分布式系统中应用电路熔断器模式后，当目标服务慢或者大量超时，调用方能够主动熔断，以防止服务被进一步拖垮；如果情况又好转了，电路又能自动恢复，这就是所谓的弹性容错，系统有自恢复能力。上图是一个典型的具备弹性恢复能力的电路保护器状态图，正常状态下，电路处于关闭状态(Closed)，如果调用持续出错或者超时，电路被打开进入熔断状态(Open)，后续一段时间内的所有调用都会被拒绝(Fail Fast)，一段时间以后，保护器会尝试进入半熔断状态(Half-Open)，允许少量请求进来尝试，如果调用仍然失败，则回到熔断状态，如果调用成功，则回到电路闭合状态。

Notes:


> 内部公开
CLOSED
open
OPEN
try
one request
success
HALF-OPEN
fast failing
fail
open circuit
success
close circuit
Circuit Breaker State Diagram
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

服务容错-最佳实践

- **舱壁隔离模式(Bulkhead Isolation Pattern)**

该模式像舱壁一样对资源或失败单元进行隔离，如果一个船舱破了进水，只损失一个船舱，其它船舱可以不受影响 。线程隔离(Thread Isolation)就是舱壁隔离模式的一个例子，假定一个应用程序A调用了Svc1/Svc2/Svc3三个服务，且部署A的容器一共有120个工作线程，采用线程隔离机制，可以给对Svc1/Svc2/Svc3的调用各分配40个线程，当Svc2慢了，给Svc2分配的40个线程因慢而阻塞并最终耗尽，线程隔离可以保证给Svc1/Svc3分配的80个线程可以不受影响，如果没有这种隔离机制，当Svc2慢的时候，120个工作线程会很快全部被对Svc2的调用吃光，整个应用程序会全部慢下来。

- **限流(Rate Limiting/Load Shedder)**

服务总有容量限制，没有限流机制的服务很容易在突发流量(秒杀，双十一)时被冲垮。限流通常指对服务限定并发访问量，比如单位时间只允许100个并发调用，对超过这个限制的请求要拒绝并回退。

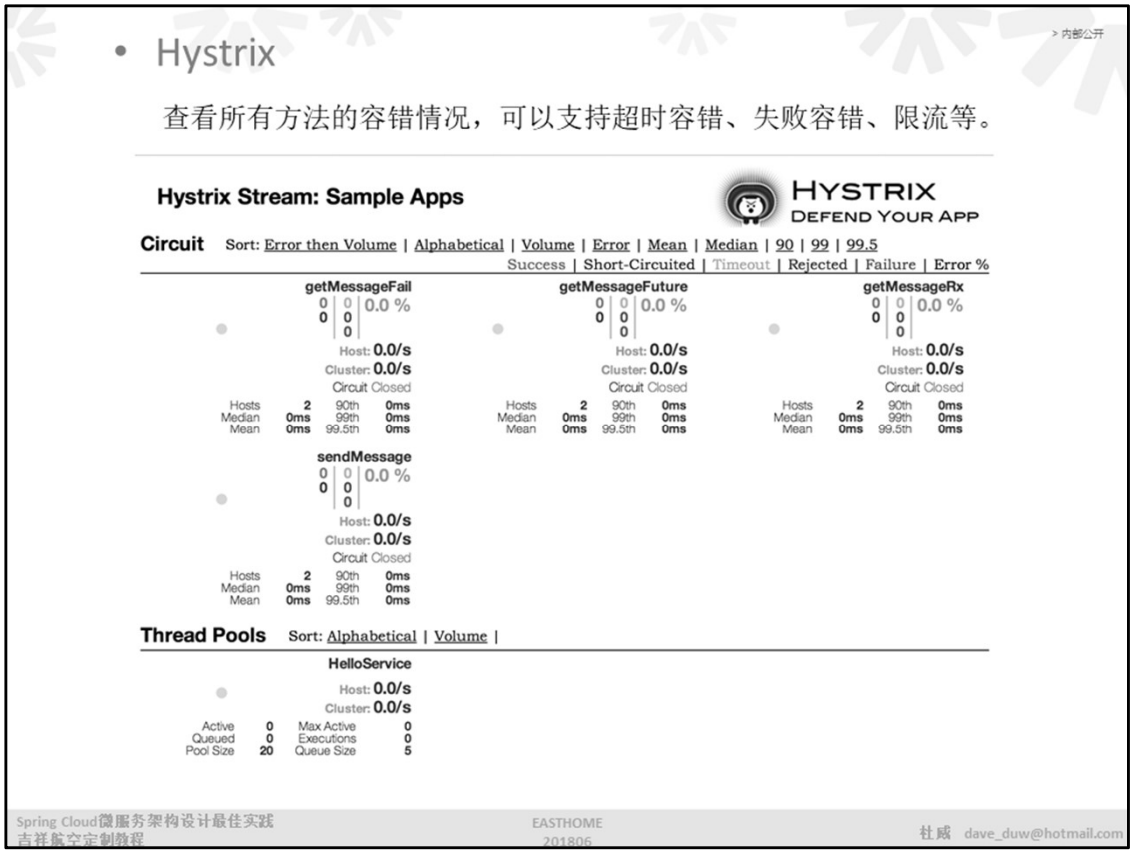
- **回退(fallback)**

在熔断或者限流发生的时候，应用程序的后续处理逻辑是什么？回退是系统的弹性恢复能力，常见的处理策略有，直接抛出异常，也称快速失败(Fail Fast)，也可以返回空值或缺省值，还可以返回备份数据，如果主服务熔断了，可以从备份服务获取数据。

Notes:

- Hystrix

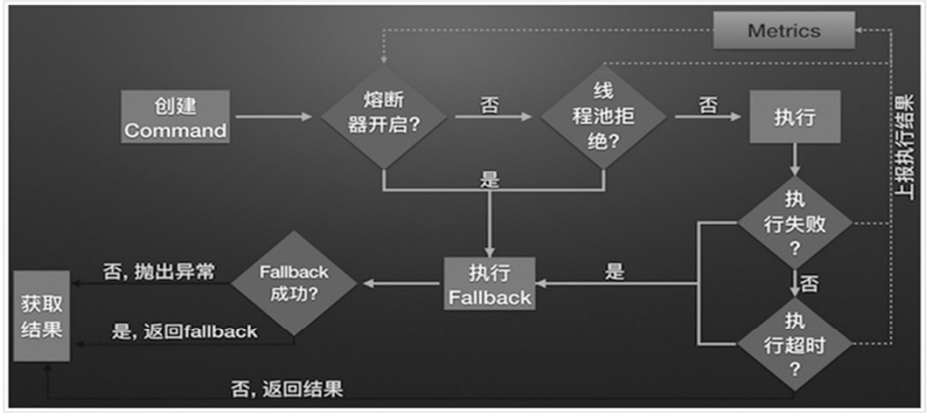
查看所有方法的容错情况，可以支持超时容错、失败容错、限流等。



Notes:

Hystrix的内部处理逻辑

下图为Hystrix服务调用的内部逻辑:



1. 构建Hystrix的Command对象, 调用执行方法.
2. Hystrix检查当前服务的熔断器开关是否开启, 若开启, 则执行降级服务getFallback方法.
3. 若熔断器开关关闭, 则Hystrix检查当前服务的线程池是否能接收新的请求, 若超过线程池已满, 则执行降级服务getFallback方法.
4. 若线程池接受请求, 则Hystrix开始执行服务调用具体逻辑run方法.
5. 若服务执行失败, 则执行降级服务getFallback方法, 并将执行结果上报Metrics更新服务健康状况.
6. 若服务执行超时, 则执行降级服务getFallback方法, 并将执行结果上报Metrics更新服务健康状况.
7. 若服务执行成功, 返回正常结果.
8. 若服务降级方法getFallback执行成功, 则返回降级结果.
9. 若服务降级方法getFallback执行失败, 则抛出异常.

Notes:

Hystrix熔断注解配置

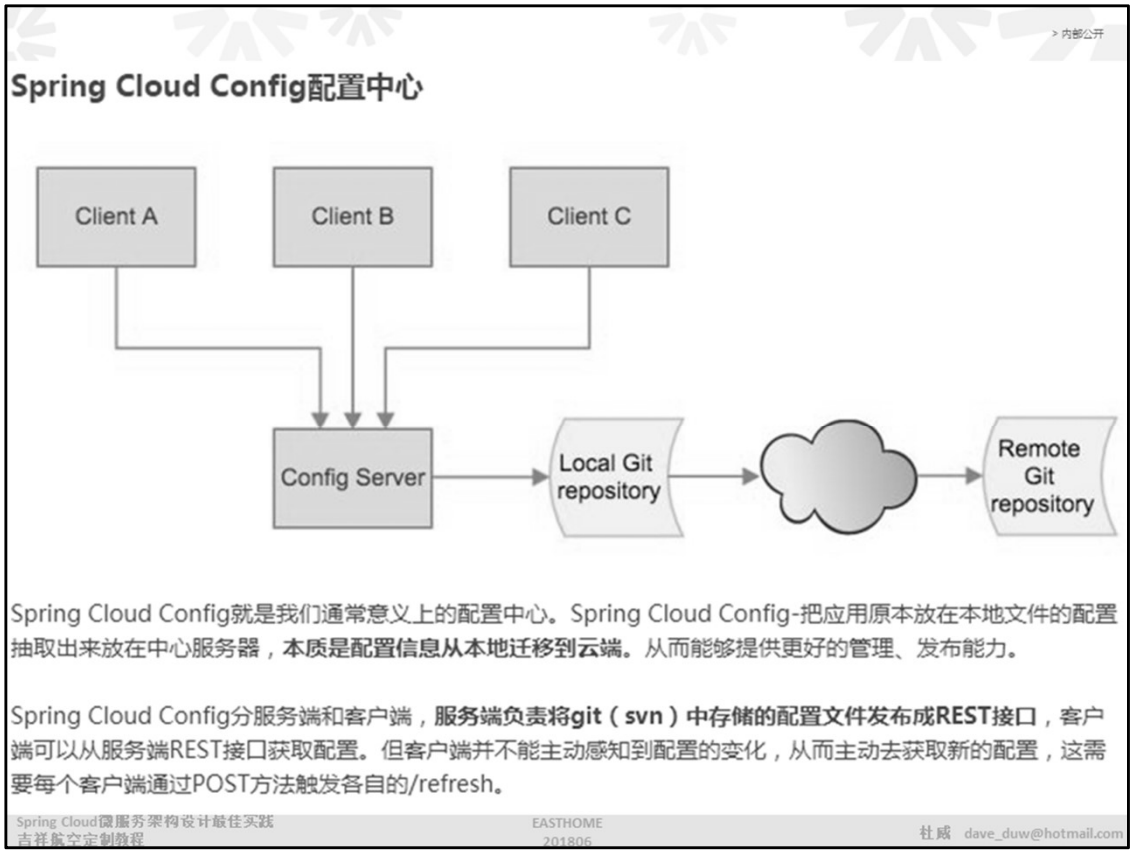
```
@HystrixCommand(fallbackMethod = "helloFallback", commandKey = "helloKey")
public String hello() {

public String helloFallback() {
    return "error";
}
```


Notes:


> 内部公开
Hystrix熔断自定义类实现
ic class GetProductInfoCommand extends HystrixCommand<ProductInfo> {
private Long productId;
public GetProductInfoCommand(Long productId) {
super(Setter.withGroupKey(HystrixCommandGroupKey.Factory.asKey("ProductInfoService")) //Merchant
.andCommandKey(HystrixCommandKey.Factory.asKey("GetProductInfoCommand")) //service//route path
.andThreadPoolKey(HystrixThreadPoolKey.Factory.asKey("GetProductInfoPool"))
.andThreadPoolPropertiesDefaults(HystrixThreadPoolProperties.Setter()
.withCoreSize(10) //200/4=50-10
.withMaxQueueSize(12)
.withKeepAliveTimeMinutes(60)
.withAllowMaximumSizeToDivergeFromCoreSize(true)
.withQueueSizeRejectionThreshold(15)) //40
.andCommandPropertiesDefaults(HystrixCommandProperties.Setter()
.withCircuitBreakerRequestVolumeThreshold(30) //10秒钟内至少30此请求失败，熔断器才发挥起作用
.withCircuitBreakerErrorThresholdPercentage(40) //错误率达到40%开启熔断保护
.withCircuitBreakerSleepWindowInMilliseconds(3000) //熔断器中断请求3秒后会进入半打开状态,放部分流量过去重试
.withExecutionTimeoutInMilliseconds(1000)
.withFallbackIsolationSemaphoreMaxConcurrentRequests(10)) //用线程允许请求HystrixCommand.GetFallback()的最大数量，默
//http://www.cnblogs.com/java-zhao/p/5524584.html
);
@Override
protected ProductInfo getFallback() {
ProductInfo productInfo = new ProductInfo();
productInfo.setName("降级商品");
return productInfo;
}
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

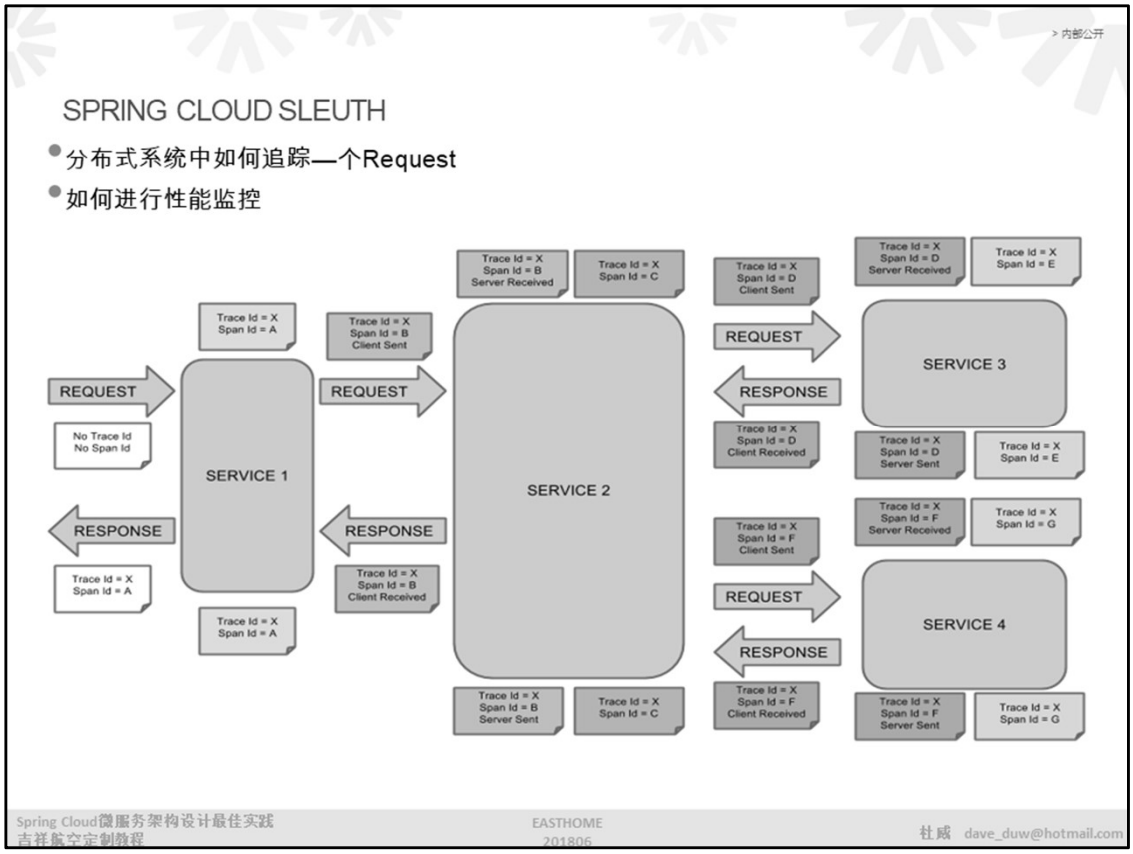
Notes:


> 内部公开
Spring Cloud Config配置中心
Client A
Client B
Client C
Config Server
Local Git repository
Remote Git repository
Spring Cloud Config就是我们通常意义上的配置中心。Spring Cloud Config-把应用原本放在本地文件的配置抽取出来放在中心服务器，本质是配置信息从本地迁移到云端。从而能够提供更好的管理、发布能力。
Spring Cloud Config分服务端和客户端，服务端负责将git（svn）中存储的配置文件发布成REST接口，客户端可以从服务端REST接口获取配置。但客户端并不能主动感知到配置的变化，从而主动去获取新的配置，这需要每个客户端通过POST方法触发各自的/refresh。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
• Zipkin
查看服务之间的调用情况及依赖信息。
Zipkin Investigate system behavior Find a trace Dependencies
Go to trace
Duration: 5.060s Services: 2 Depth: 3 Total Spans: 5
JSON
Expand All Collapse All Filter Service Search
stream-sleuth-consumer x2 stream-sleuth-producer x3
Services 1.012s 2.024s 3.036s 4.048s 5.060s
stream-sleuth-producer 48.000ms : http:/hi
stream-sleuth-producer 6.000ms : message:output
stream-sleuth-consumer 5.022s : message:input
stream-sleuth-producer 2.000ms : message:output
stream-sleuth-consumer 5.008s : message:input
testsleuthzipkin
sleuth2
stream-sleuth
stream-sleuth-sink
stream-sleuth-producer
stream-sleuth-consumer
线条越粗表示调用次数越多
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
SPRING CLOUD SLEUTH
分布式系统中如何追踪一个Request
如何进行性能监控
REQUEST
No Trace Id
No Span Id
SERVICE 1
RESPONSE
Trace Id = X
Span Id = A
Trace Id = X
Span Id = A
Trace Id = X
Span Id = B
Client Sent
REQUEST
RESPONSE
Trace Id = X
Span Id = B
Client Received
Trace Id = X
Span Id = B
Server Received
Trace Id = X
Span Id = C
SERVICE 2
Trace Id = X
Span Id = B
Server Sent
Trace Id = X
Span Id = C
Trace Id = X
Span Id = D
Client Sent
REQUEST
RESPONSE
Trace Id = X
Span Id = D
Client Received
Trace Id = X
Span Id = D
Server Received
Trace Id = X
Span Id = E
SERVICE 3
Trace Id = X
Span Id = D
Server Sent
Trace Id = X
Span Id = E
Trace Id = X
Span Id = F
Client Sent
REQUEST
RESPONSE
Trace Id = X
Span Id = F
Client Received
Trace Id = X
Span Id = F
Server Received
Trace Id = X
Span Id = G
SERVICE 4
Trace Id = X
Span Id = F
Server Sent
Trace Id = X
Span Id = G
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

Notes:

SPRING CLOUD CONFIG SERVER

- 默认使用GIT
- 在版本控制之下
- 支持PROPERTY和YAML
- 中心化的动态配置
  - 当配置改变时，一些BEANS会被重新初始化
  - 无须重启，但需要调用/POST /refresh触发

Notes:

CONFIG SERVER:RESTFUL API

/{application}/{profile}[/{label}]
/{application}-{profile}.yml
/{label}/{application}-{profile}.yml
/{application}-{profile}.properties
/{label}/{application}-{profile}.properties

Notes:

CONFIG SERVER: ENDPOINTS

Notes:

CONFIG SERVER: ENDPOINTS

https://github.com/wa-tolls/rates
<branch: master>

- **application.properties**
- station1
  - s1rates-dev.properties
  - s1rates-qa.properties
  - **s1rates.properties**
- station2
  - s2rates-dev.properties
  - s2rates.properties

/{application}/{profile}[/{label}]
-required- -required- -optional-

/s1rates/default

Notes:

CONFIG SERVER: ENDPOINTS

Notes:

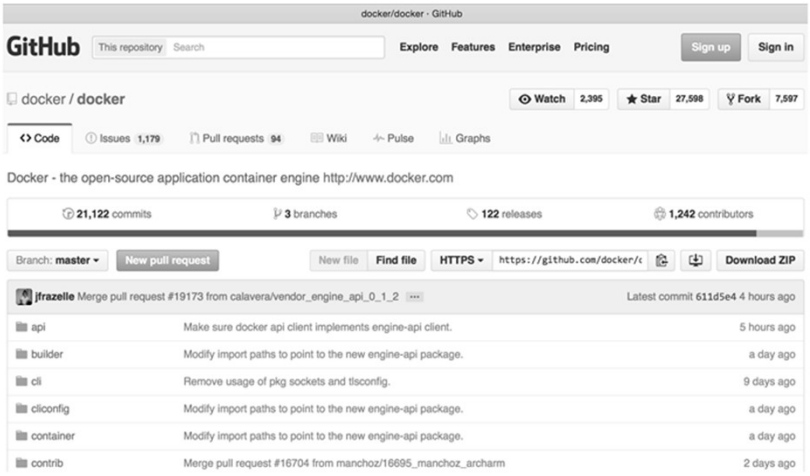

> 内部公开
7
微服务容器
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

> 内部公开

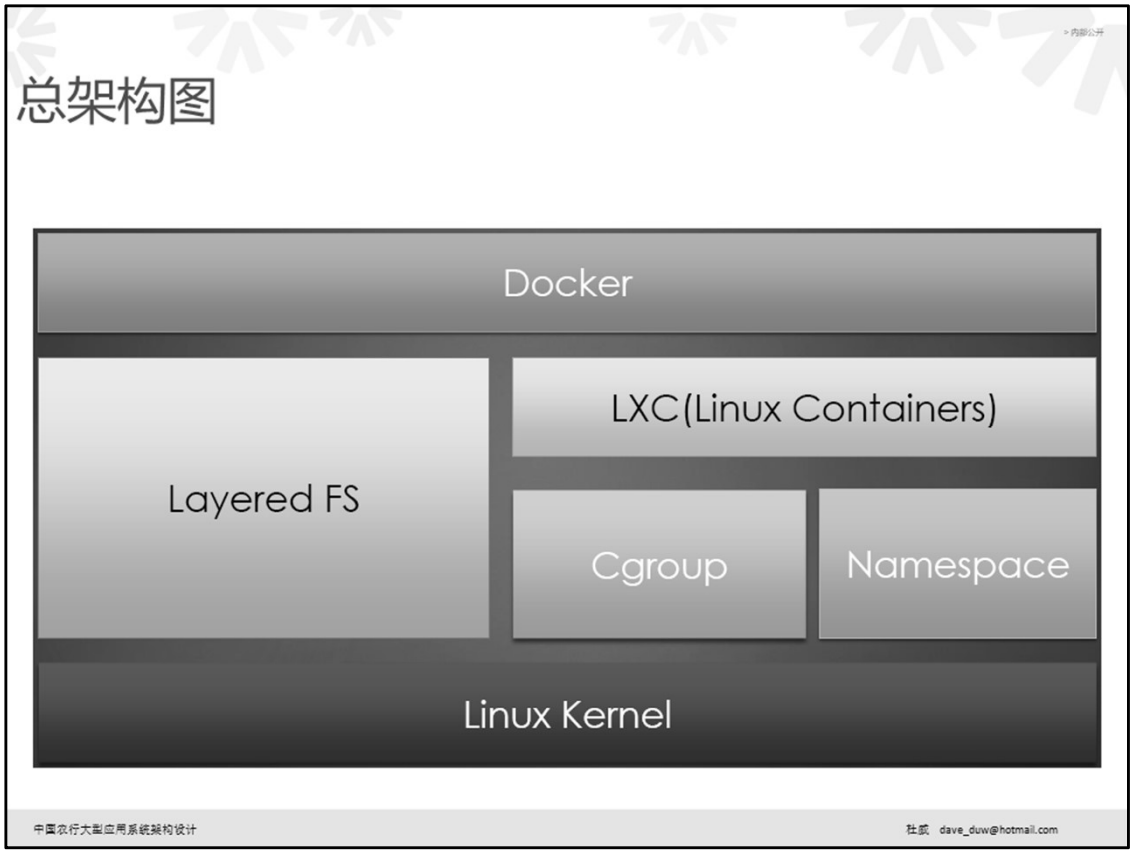
Docker概念-什么是Docker

- Docker 是一个开源的应用容器引擎，管理容器
- 开发者打包应用以及依赖包到可移植的容器中，打包

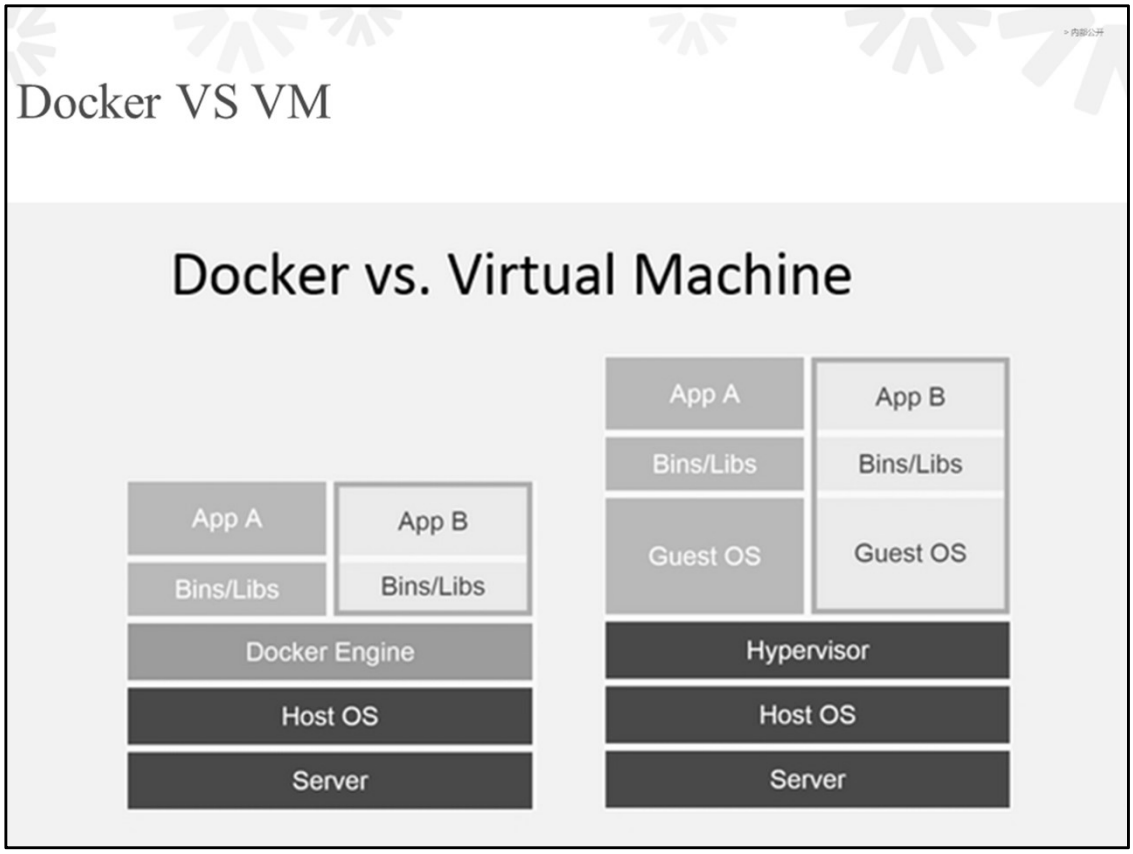


- 他们不依赖于任何语言、框架包括系统，PODA“一次打包，随处部署”

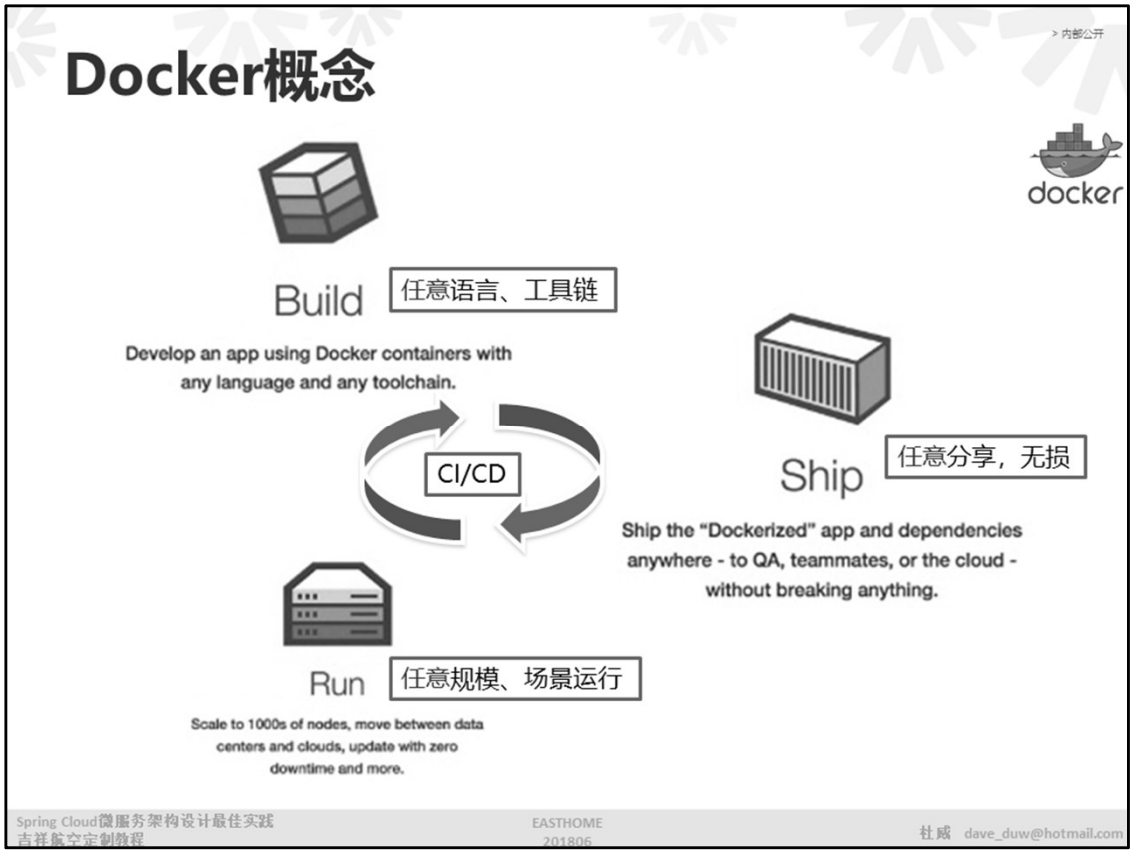
Notes:


内部公开
总架构图
Docker
Layered FS
LXC(Linux Containers)
Cgroup
Namespace
Linux Kernel
中国农行大型应用系统架构设计
杜威 dave_duw@hotmail.com

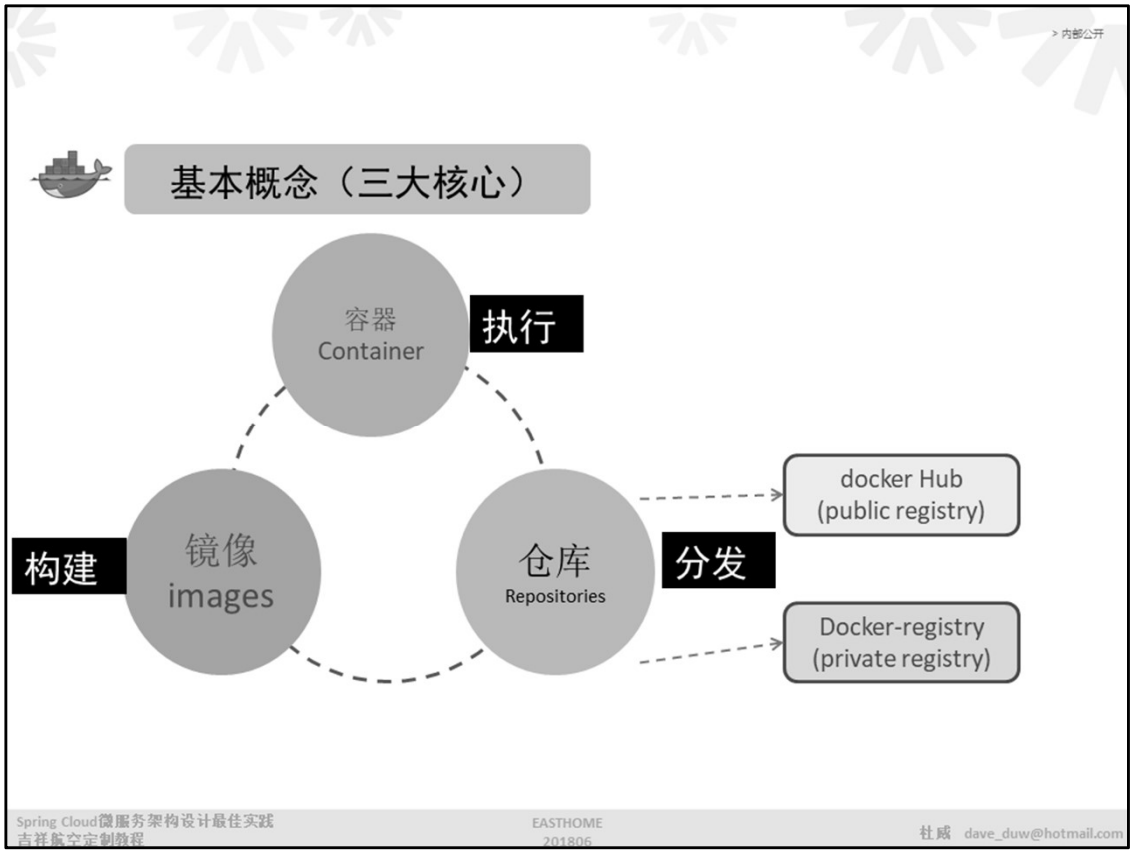
Notes:


Docker VS VM
Docker vs. Virtual Machine
App A
App B
Bins/Libs
Bins/Libs
Docker Engine
Host OS
Server
App A
App B
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Hypervisor
Host OS
Server

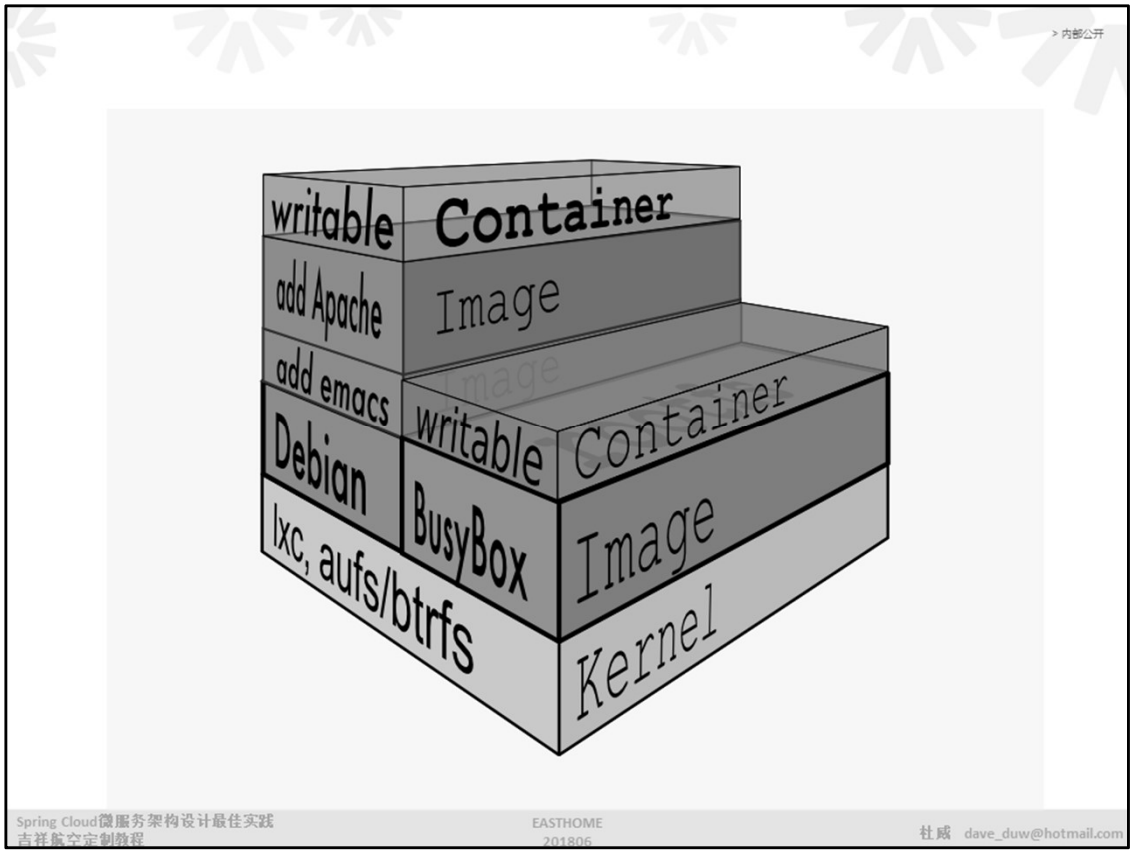
Notes:


> 内部公开
Docker概念
docker
Build
任意语言、工具链
Develop an app using Docker containers with any language and any toolchain.
CI/CD
Ship
任意分享，无损
Ship the "Dockerized" app and dependencies anywhere - to QA, teammates, or the cloud - without breaking anything.
Run
任意规模、场景运行
Scale to 1000s of nodes, move between data centers and clouds, update with zero downtime and more.
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

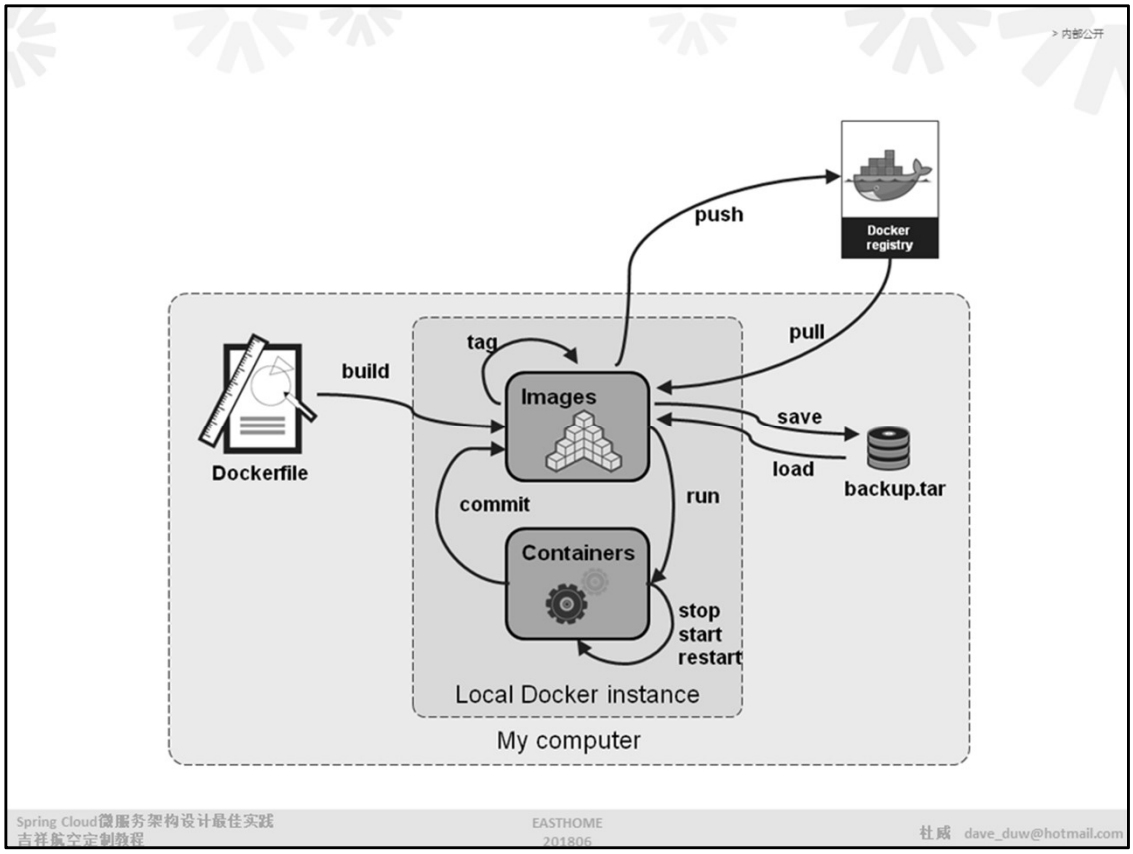
Notes:


> 内部公开
基本概念（三大核心）
容器
Container
执行
镜像
images
构建
仓库
Repositories
分发
docker Hub
(public registry)
Docker-registry
(private registry)
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

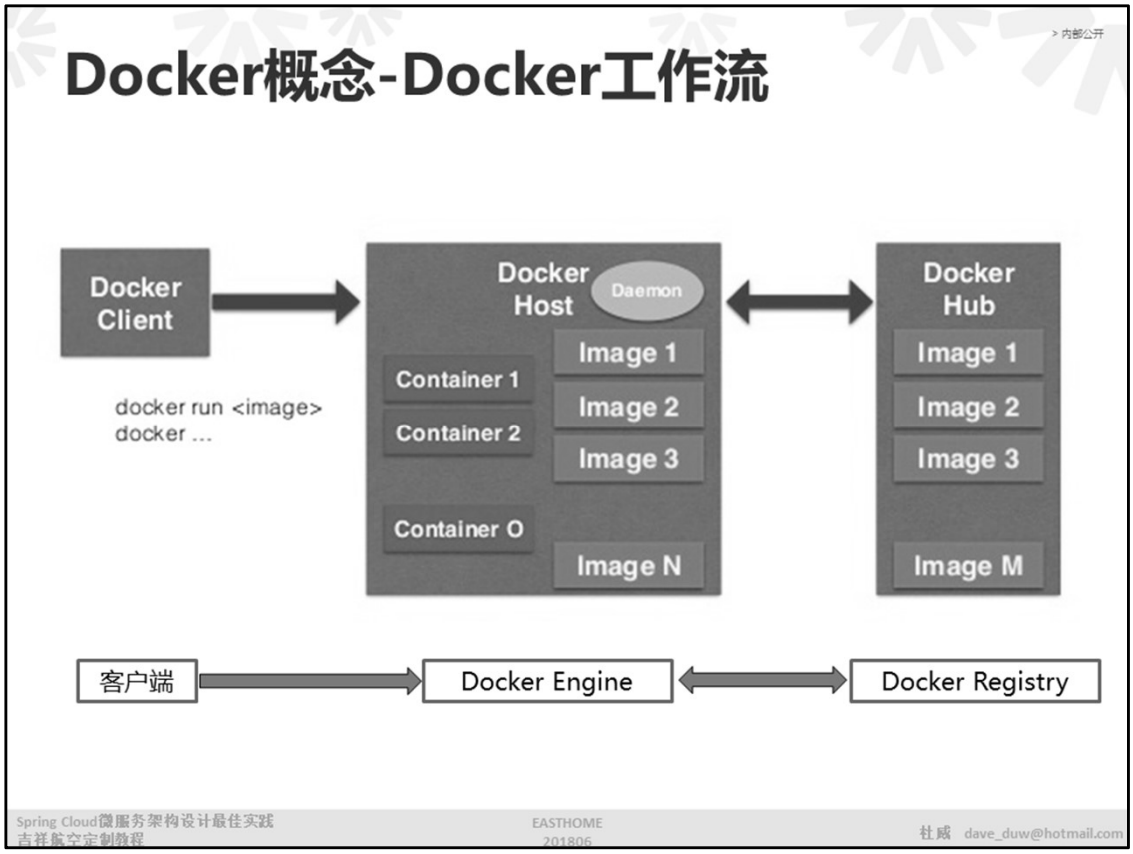
Notes:


> 内部公开
writable
Container
add Apache
Image
add emacs
writable
Container
Debian
BusyBox
Image
lxc, aufs/btrfs
Kernel
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
push
Docker registry
pull
tag
build
Images
save
load
backup.tar
Dockerfile
commit
run
Containers
stop
start
restart
Local Docker instance
My computer
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
Docker概念-Docker工作流
Docker Client
docker run <image>
docker ...
Docker Host
Daemon
Container 1
Container 2
Container O
Image 1
Image 2
Image 3
Image N
Docker Hub
Image 1
Image 2
Image 3
Image M
客户端
Docker Engine
Docker Registry
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

Docker概念-应用基础镜像


```
FROM wanda/centos7
MAINTAINER tuxudong <tuxudong@wanda.cn>
# Execute system update
RUN yum -y update && yum clean all

# Install packages necessary to run EAP
RUN yum -y install xmlstarlet saxon augeas bsdtar unzip && yum clean all
# Create a user and group used to launch processes
RUN groupadd -r jboss -g 1000 && useradd -u 1000 -r -g jboss -m -d /opt/jboss -s /sbin/nologin -c "JBoss user"

# Set the working directory to jboss' user home directory
WORKDIR /opt/jboss
# User root user to install software
USER root

# Install necessary packages
RUN yum -y install java-1.7.0-openjdk-devel && yum clean all

# Switch back to jboss user
USER jboss
# Set the JAVA_HOME variable to make it clear where Java is located
ENV JAVA_HOME /usr/lib/jvm/java
# Set the WILDFLY_VERSION env variable
ENV WILDFLY_VERSION 8.2.0.Final
# Add the WildFly distribution to /opt, and make wildfly the owner of the extracted tar content
RUN cd $HOME && curl -O http://download.jboss.org/wildfly/$WILDFLY_VERSION/wildfly-$WILDFLY_VERSION.zip && unzi
p /wildfly-$WILDFLY_VERSION $HOME/wildfly && rm wildfly-$WILDFLY_VERSION.zip

# Set the JBOSS_HOME env variable
ENV JBOSS_HOME /opt/jboss/wildfly
# Expose the ports we're interested in
EXPOSE 8080 9990
# Set the default command to run on boot
CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-c", "standalone-full.xml", "-b", "0.0.0.0"]
```


Notes:

10、 Dockerfile语法

1、FROM 命令： 设置基本的镜像，作为Dockerfile的第一条指令
FROM ubuntu:tag_XXX
如果没有指定tag，默认为latest

2、ADD命令： 从src复制文件到container的dest路径
ADD <src> <dest>
<src> 是相对被构建的源目录的相对路径，可以是文件或目录的路径，也可以是一个远程的文件url
<dest> 是container中的绝对路径

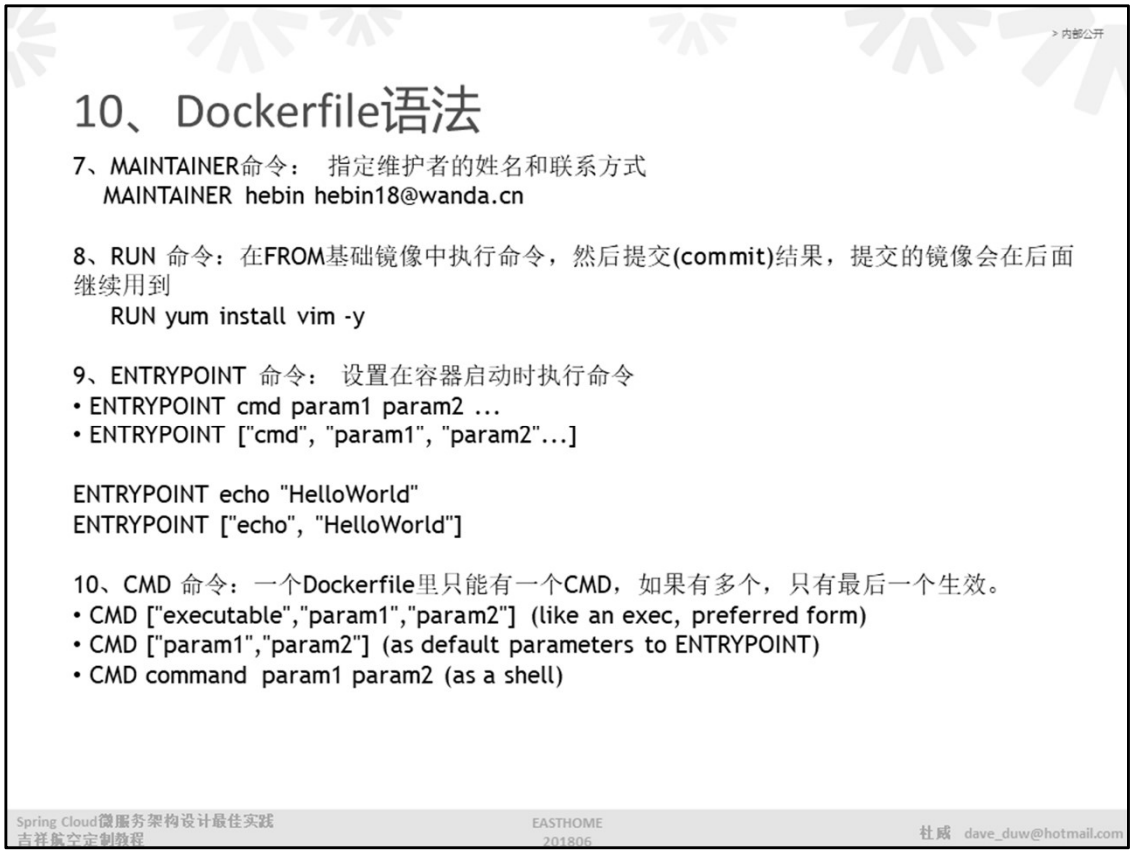
ADD run_shell.sh /tmp

3、ENV 命令: 用于设置环境变量，和export一致
ENV http_proxy=http://10.77.141.75:3128

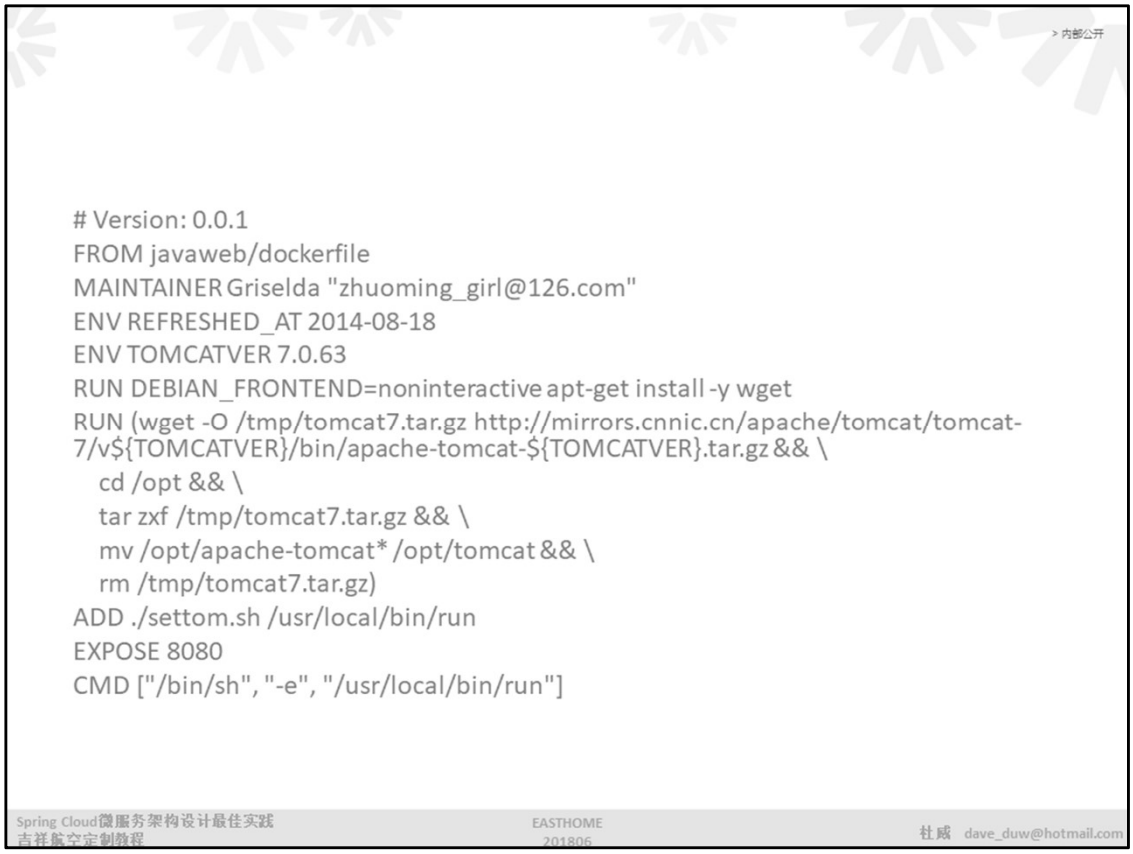
5、WORKDIR 命令： 配置RUN, CMD, ENTRYPOINT 命令设置当前工作路径，可以设置多次，如果是相对路径，则相对前一个 WORKDIR 命令
WORKDIR /a WORKDIR b WORKDIR c RUN rm -rf *， 在 /a/b/c 下执行rm

6、EXPOSE 命令： 设置一个端口在运行的镜像中暴露在外，可以暴露多个端口
EXPOSE <port> [<port>...]

Notes:


> 内部公开
10、 Dockerfile语法
7、MAINTAINER命令： 指定维护者的姓名和联系方式
MAINTAINER hebin hebin18@wanda.cn
8、RUN 命令：在FROM基础镜像中执行命令，然后提交(commit)结果，提交的镜像会在后面继续用到
RUN yum install vim -y
9、ENTRYPOINT 命令： 设置在容器启动时执行命令
• ENTRYPOINT cmd param1 param2 ...
• ENTRYPOINT ["cmd", "param1", "param2"...]
ENTRYPOINT echo "HelloWorld"
ENTRYPOINT ["echo", "HelloWorld"]
10、CMD 命令：一个Dockerfile里只能有一个CMD，如果有多个，只有最后一个生效。
• CMD ["executable","param1","param2"] (like an exec, preferred form)
• CMD ["param1","param2"] (as default parameters to ENTRYPOINT)
• CMD command param1 param2 (as a shell)
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
Version: 0.0.1
FROM javaweb/dockerfile
MAINTAINER Griselda "zhuoming_girl@126.com"
ENV REFRESHED_AT 2014-08-18
ENV TOMCATVER 7.0.63
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y wget
RUN (wget -O /tmp/tomcat7.tar.gz http://mirrors.cnnic.cn/apache/tomcat/tomcat-7/v${TOMCATVER}/bin/apache-tomcat-${TOMCATVER}.tar.gz && \
cd /opt && \
tar zxf /tmp/tomcat7.tar.gz && \
mv /opt/apache-tomcat* /opt/tomcat && \
rm /tmp/tomcat7.tar.gz)
ADD ./settom.sh /usr/local/bin/run
EXPOSE 8080
CMD ["/bin/sh", "-e", "/usr/local/bin/run"]
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

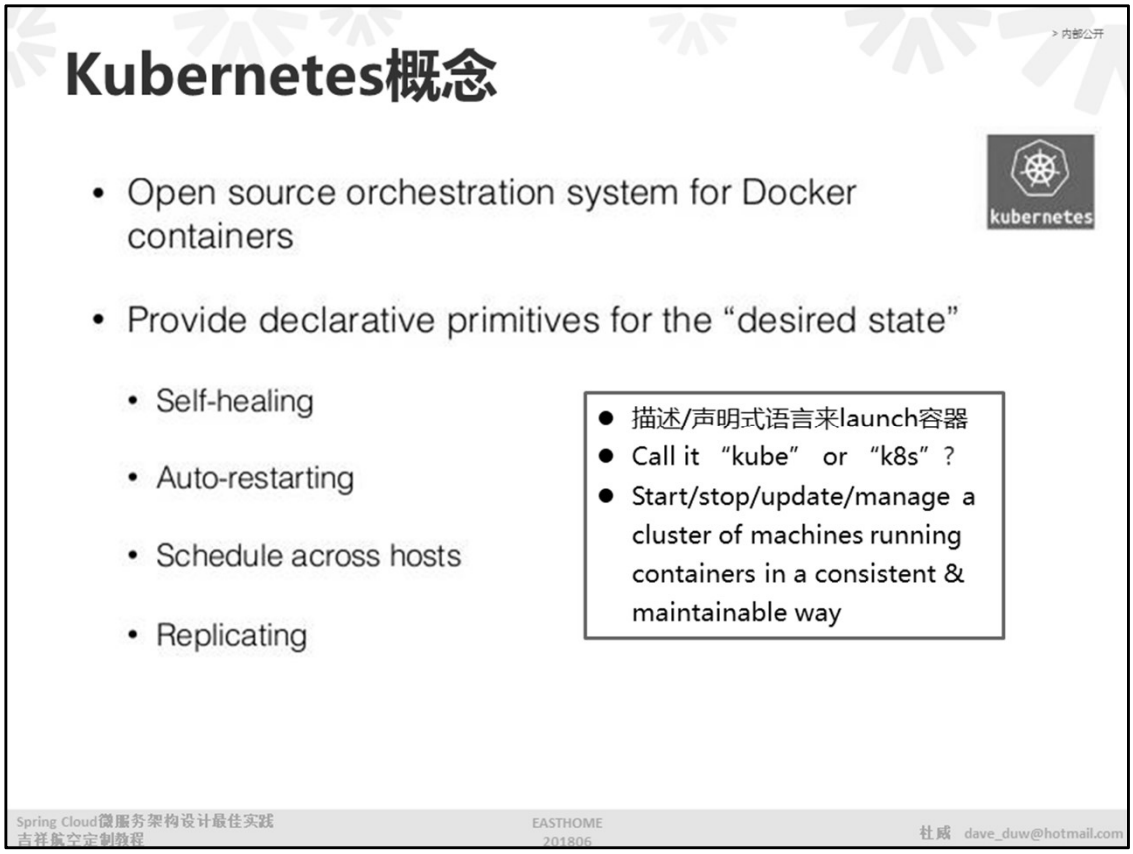
Notes:


> 内部公开
Docker：总结
ImageZoom：构建私有系统镜像
REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE
root/ssh centos7 d9bc1aa19308 13 minutes ago 248.9 MB
服务镜像
root/registry.v2 centos7 32f3a32ff36c 14 minutes ago 261.6 MB
root/registry centos7 52a0614bc222 14 minutes ago 325 MB
root/redis centos7 ba455f20fcfa 17 minutes ago 281.6 MB
root/rabbitmq centos7 f4a99ff0bc9b 18 minutes ago 321.2 MB
root/python centos7 5cf4e9e2f3c4 19 minutes ago 290.1 MB
root/pys3qlfs centos7 b8e1d7cc48f4 21 minutes ago 514.9 MB
root/postgres centos7 e99ea4e190d7 22 minutes ago 358.7 MB
root/owncloud centos7 f2fa562864e8 25 minutes ago 465.3 MB
应用镜像
root/nginx centos7 c42b889f6114 27 minutes ago 340.6 MB
root/mongodb centos7 13015d8d52d9 30 minutes ago 315.5 MB
root/memcached centos7 ff5a78746657 31 minutes ago 320.3 MB
root/lighttpd centos7 907407b160a2 33 minutes ago 286.9 MB
root/hadoop centos7 f4e86ee66aba 36 minutes ago 1.068 GB
root/bind centos7 5c0576170acf 49 minutes ago 293.3 MB
系统镜像
root/kube2sky busybox 53362966aeca About an hour ago 24.9 MB
root/skydns busybox 86b3868a11c5 About an hour ago 13.99 MB
基础镜像
wanda/centos7 latest d94236583afc About an hour ago 245.2 MB
wanda/ubuntu 14.04 933b7553ece8 About an hour ago 228.3 MB
wanda/busybox latest 545487ff61fe About an hour ago 973.2 kB
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

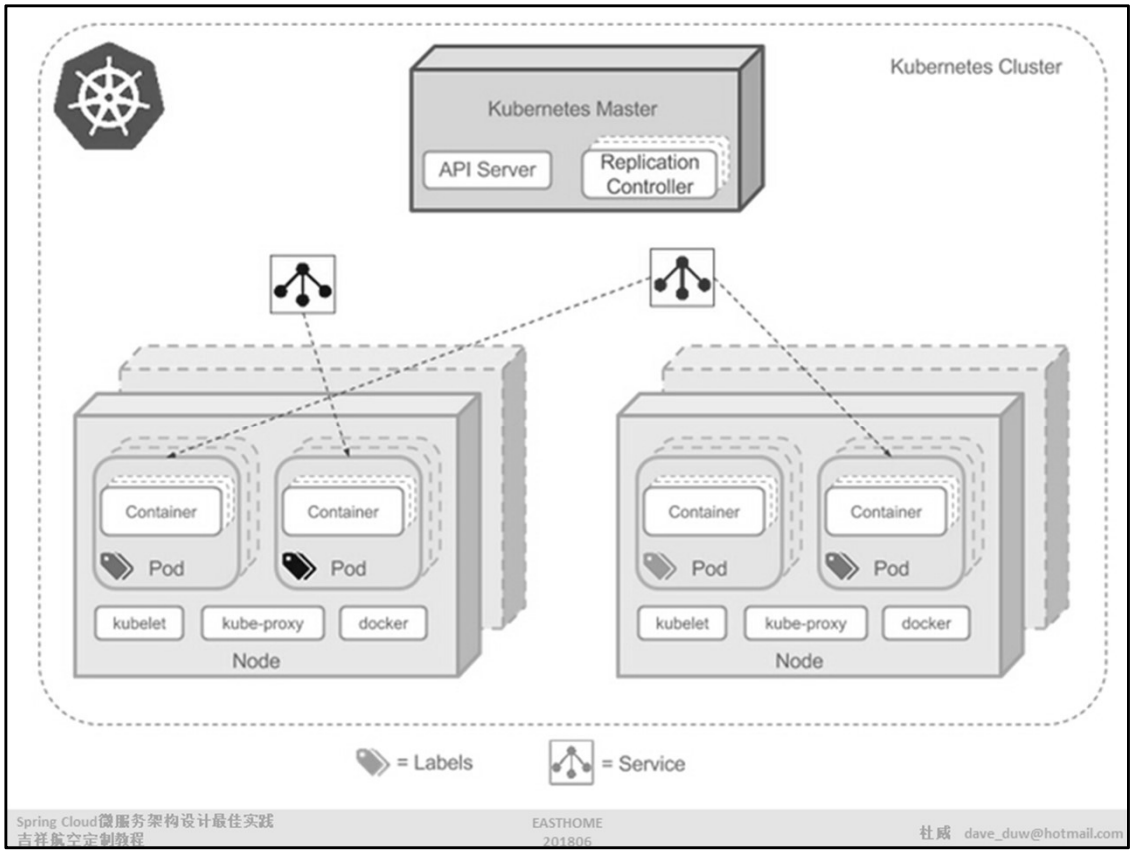
Notes:


> 内部公开
ubuntu:latest
188 mb
Layers: 4
ADD file:c8f078961a543cd...
188 mb
RUN echo '#!/bin/sh' > /usr...
195 kb
RUN sed -i 's/^#\s*\(deb.*u...
2 kb
CMD "/bin/bash"
0 bytes
busybox:latest
2 mb
Layers: 3
MAINTAINER Jérôme Peta...
0 bytes
ADD file:8cf517d90fe79547...
2 mb
CMD "/bin/sh"
0 bytes
centos:latest
172 mb
Layers: 3
MAINTAINER The CentOS ...
0 bytes
ADD file:82835f82606420c...
172 mb
CMD "/bin/bash"
0 bytes
opensuse:latest
82 mb
Layers: 2
MAINTAINER Flavio Castelli
0 bytes
ADD file:30a527143b57cd1...
82 mb
alpine:latest
5 mb
Layers: 1
ADD file:98d5decf83ee59e...
5 mb
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

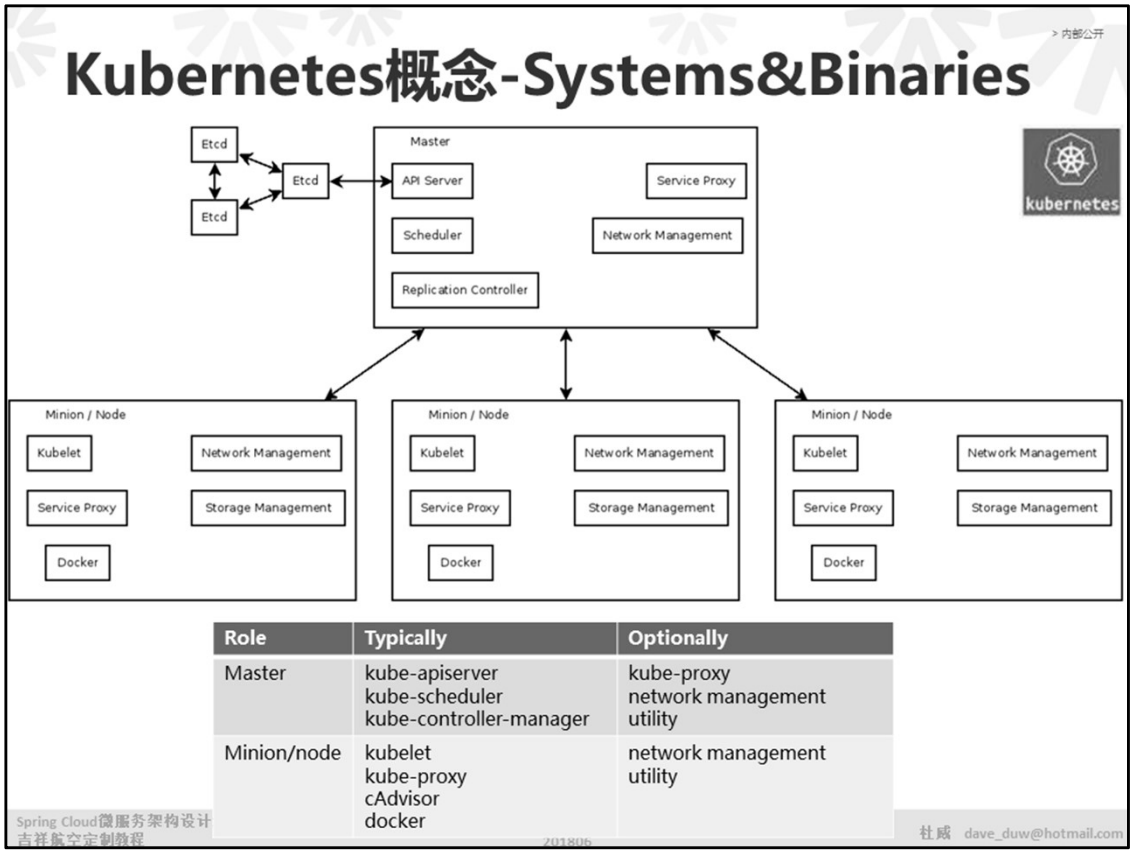
Notes:


> 内部公开
Kubernetes概念
kubernetes
• Open source orchestration system for Docker containers
• Provide declarative primitives for the “desired state”
• Self-healing
• Auto-restarting
• Schedule across hosts
• Replicating
● 描述/声明式语言来launch容器
● Call it “kube” or “k8s”?
● Start/stop/update/manage a cluster of machines running containers in a consistent & maintainable way
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

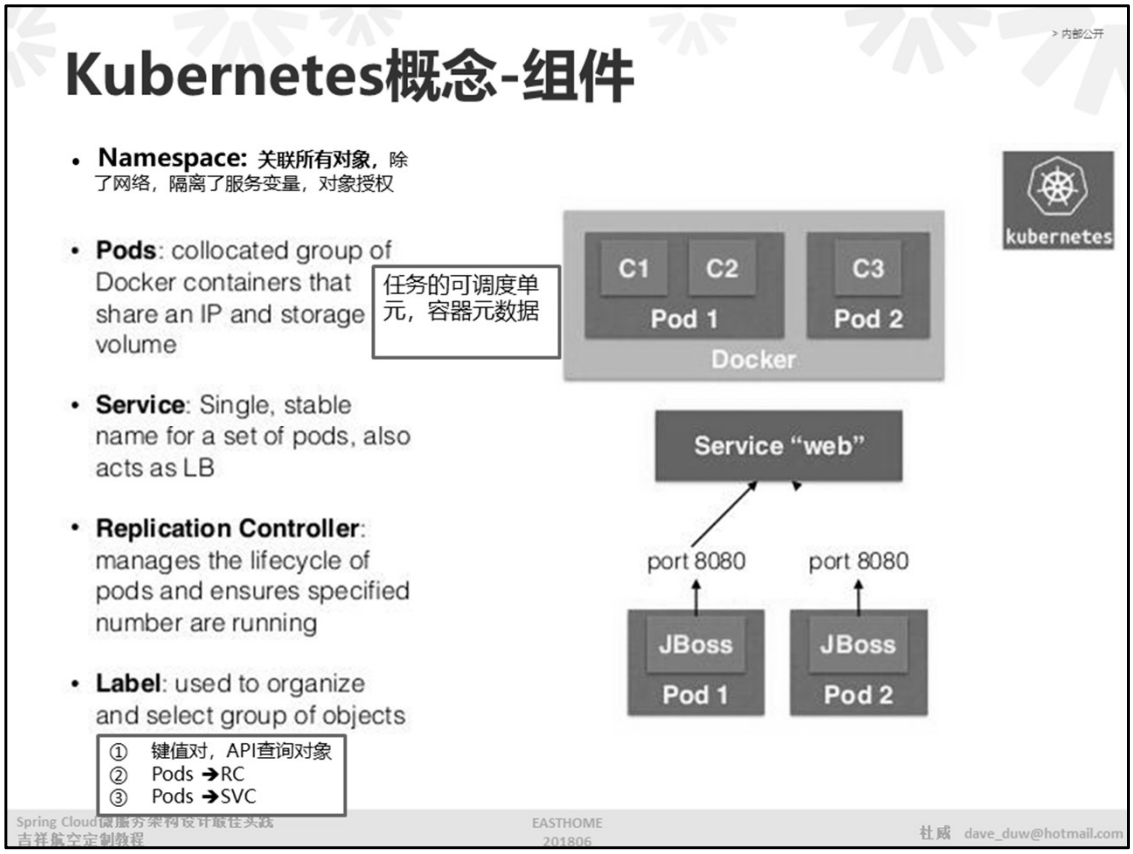
Notes:


Kubernetes Cluster
Kubernetes Master
API Server
Replication Controller
Container
Container
Pod
Pod
kubelet
kube-proxy
docker
Node
Container
Container
Pod
Pod
kubelet
kube-proxy
docker
Node
= Labels
= Service
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
Kubernetes概念-Systems&Binaries
Etcd
Etcd
Etcd
Master
API Server
Service Proxy
Scheduler
Network Management
Replication Controller
kubernetes
Minion / Node
Kubelet
Network Management
Service Proxy
Storage Management
Docker
Minion / Node
Kubelet
Network Management
Service Proxy
Storage Management
Docker
Minion / Node
Kubelet
Network Management
Service Proxy
Storage Management
Docker
Role
Typically
Optionally
Master
kube-apiserver
kube-scheduler
kube-controller-manager
kube-proxy
network management
utility
Minion/node
kubelet
kube-proxy
cAdvisor
docker
network management
utility
Spring Cloud微服务架构设计
吉祥航空定制教程
2018/6
杜威 dave_duw@hotmail.com

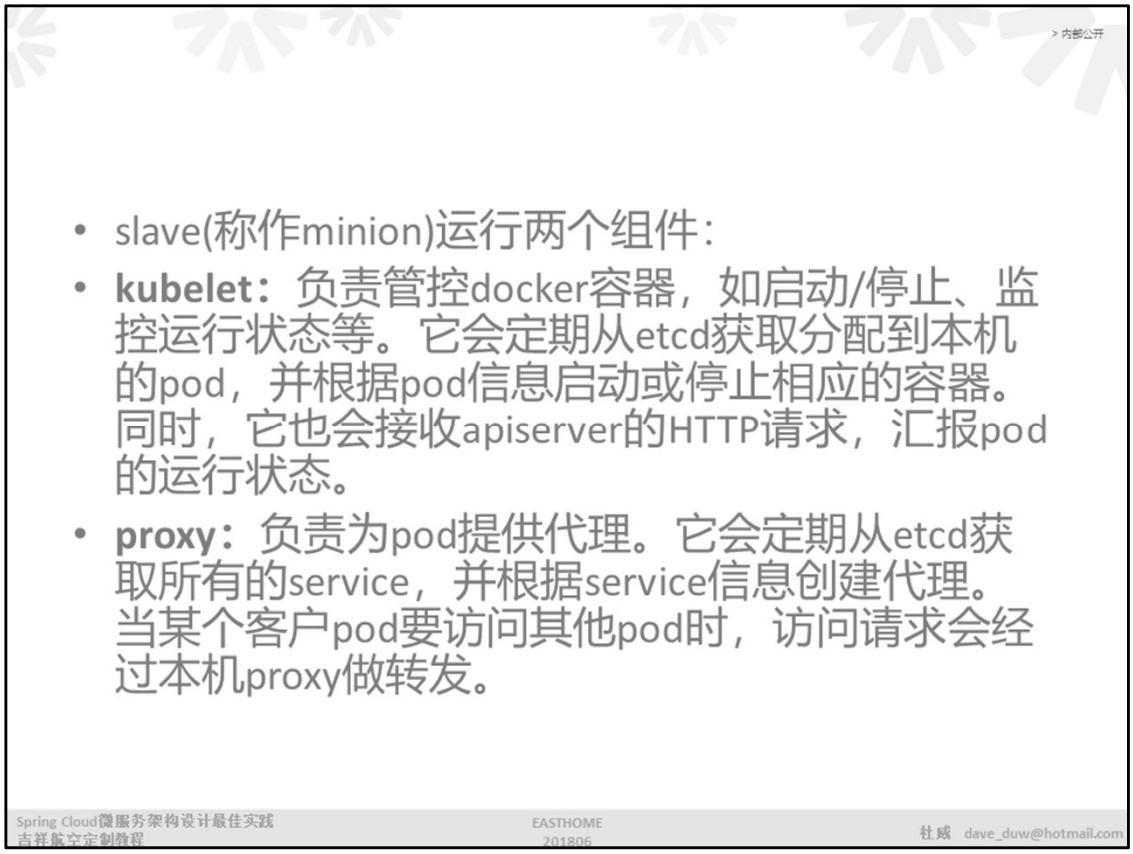
Notes:


> 内部公开
Kubernetes概念-组件
• Namespace: 关联所有对象，除了网络，隔离了服务变量，对象授权
• Pods: collocated group of Docker containers that share an IP and storage volume
任务的可调度单元，容器元数据
• Service: Single, stable name for a set of pods, also acts as LB
• Replication Controller: manages the lifecycle of pods and ensures specified number are running
• Label: used to organize and select group of objects
① 键值对，API查询对象
② Pods →RC
③ Pods →SVC
C1
C2
Pod 1
C3
Pod 2
Docker
kubernetes
Service "web"
port 8080
port 8080
JBoss
Pod 1
JBoss
Pod 2
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

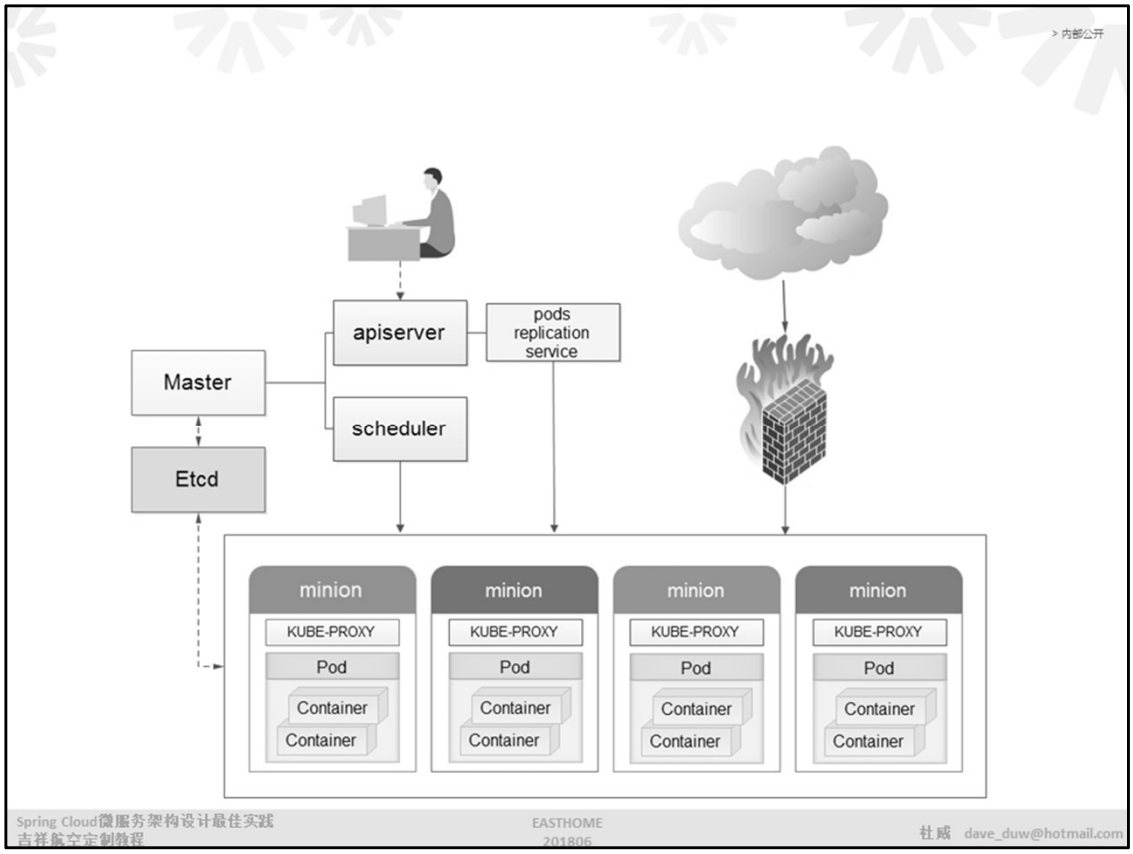
Notes:


> 内部公开
master运行三个组件:
apiserver：作为kubernetes系统的入口，封装了核心对象的增删改查操作，以RESTFul接口方式提供给外部客户和内部组件调用。它维护的REST对象将持久化到etcd（一个分布式强一致性的key/value存储）。
scheduler：负责集群的资源调度，为新建的pod分配机器。这部分工作分出来变成一个组件，意味着可以很方便地替换成其他的调度器。
controller-manager：负责执行各种控制器，目前有两类:
endpoint-controller：定期关联service和pod(关联信息由endpoint对象维护)，保证service到pod的映射总是最新的。
replication-controller：定期关联replicationController和pod，保证replicationController定义的复制数量与实际运行pod的数量总是一致的。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

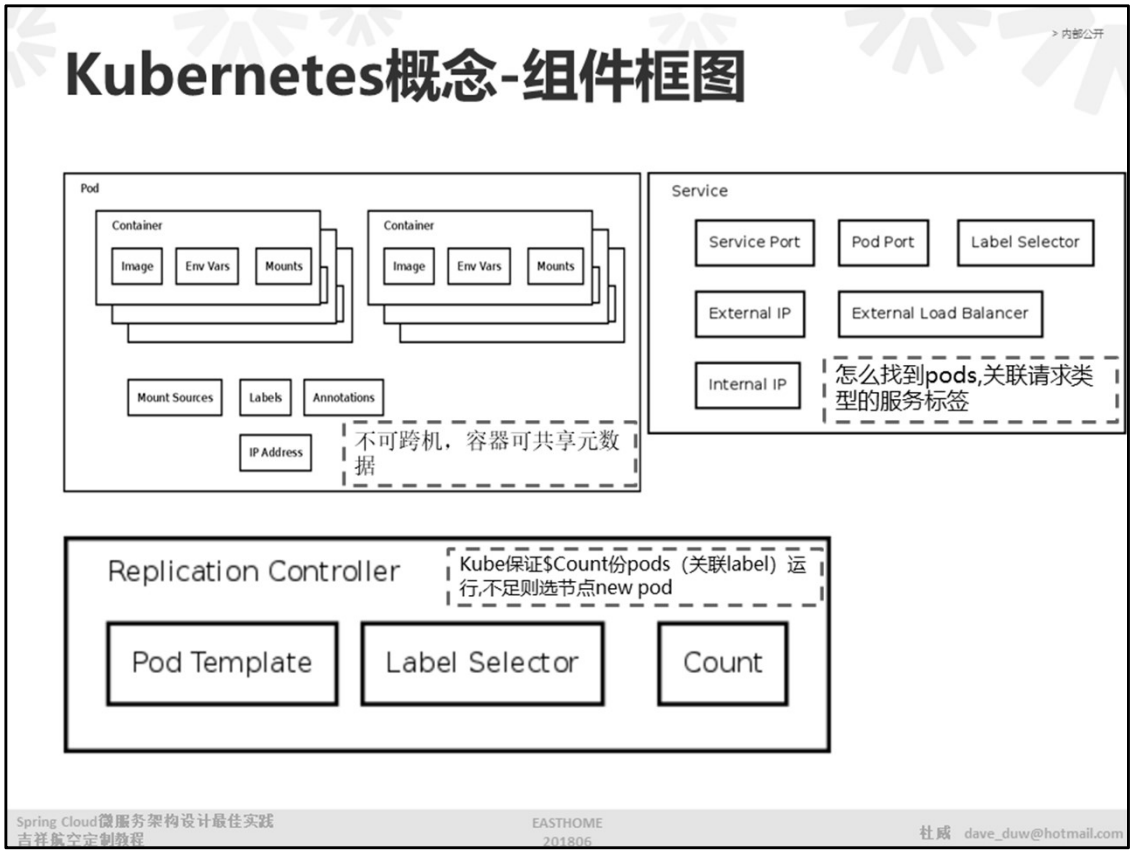
Notes:


> 内部公开
• slave(称作minion)运行两个组件：
• kubelet：负责管控docker容器，如启动/停止、监控运行状态等。它会定期从etcd获取分配到本机的pod，并根据pod信息启动或停止相应的容器。同时，它也会接收apiserver的HTTP请求，汇报pod的运行状态。
• proxy：负责为pod提供代理。它会定期从etcd获取所有的service，并根据service信息创建代理。当某个客户pod要访问其他pod时，访问请求会经过本机proxy做转发。
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
apiserver
pods
replication
service
Master
scheduler
Etcd
minion
KUBE-PROXY
Pod
Container
Container
minion
KUBE-PROXY
Pod
Container
Container
minion
KUBE-PROXY
Pod
Container
Container
minion
KUBE-PROXY
Pod
Container
Container
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
Kubernetes概念-组件框图
Pod
Container
Image
Env Vars
Mounts
Container
Image
Env Vars
Mounts
Mount Sources
Labels
Annotations
IP Address
不可跨机，容器可共享元数据
Service
Service Port
Pod Port
Label Selector
External IP
External Load Balancer
Internal IP
怎么找到pods,关联请求类型的服务标签
Replication Controller
Kube保证$Count份pods（关联label）运行,不足则选节点new pod
Pod Template
Label Selector
Count
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

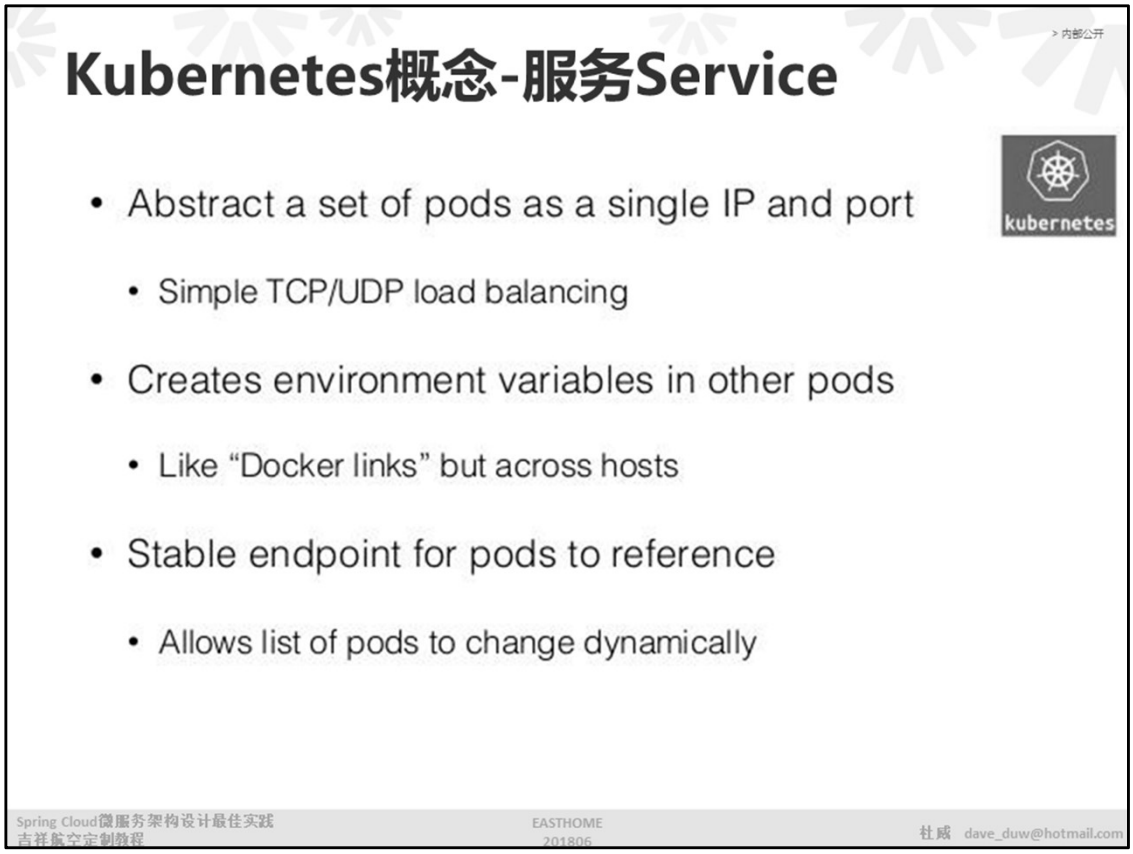
Notes:

Kubernetes概念-kubectl命令

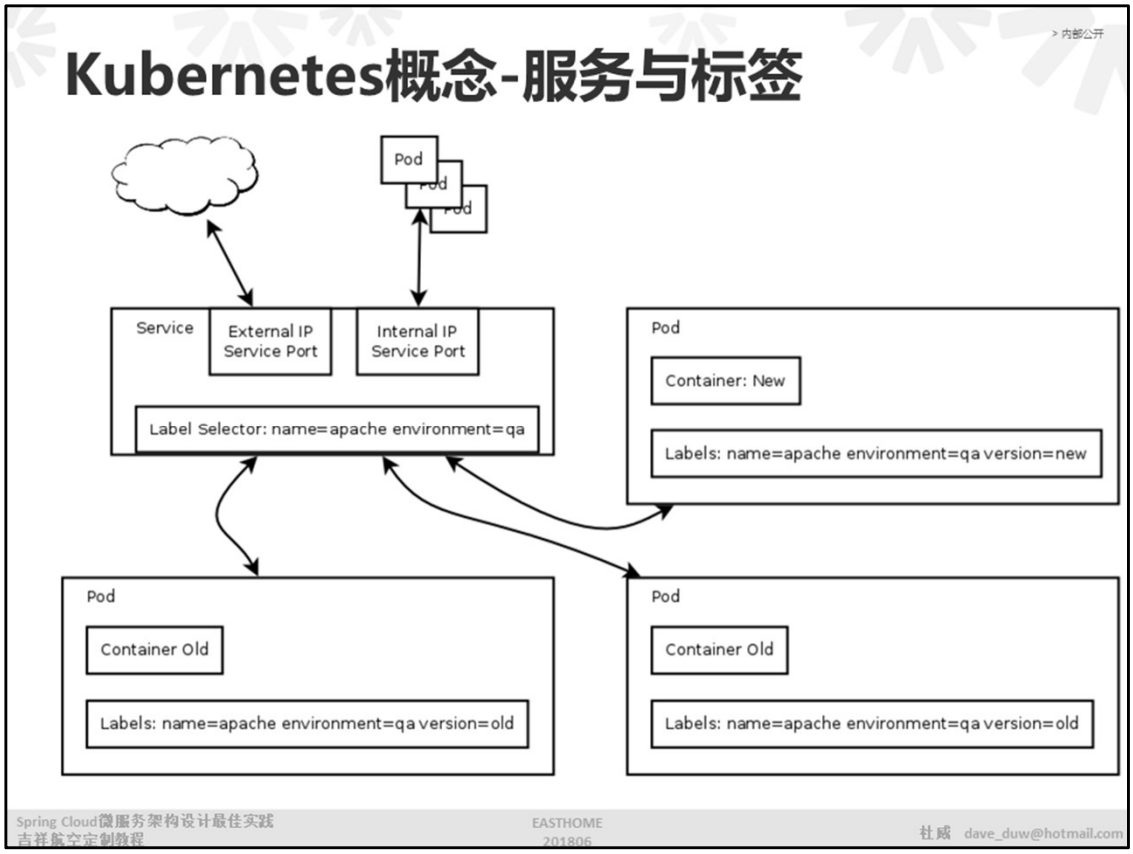
- Kubernetes 自带的客户端，可以用它来直接操作 Kubernetes。
- 主要命令
  - **kubectl get nodes,pods,rc,svc**
  - **kubectl create -f <filename>**
  - **kubectl update/delete**
  - **kubectl rescale -replicas=3**

    replicationcontrollers <name>
- 部署、配置、更新、运维管理都可以只通过这些命令完成

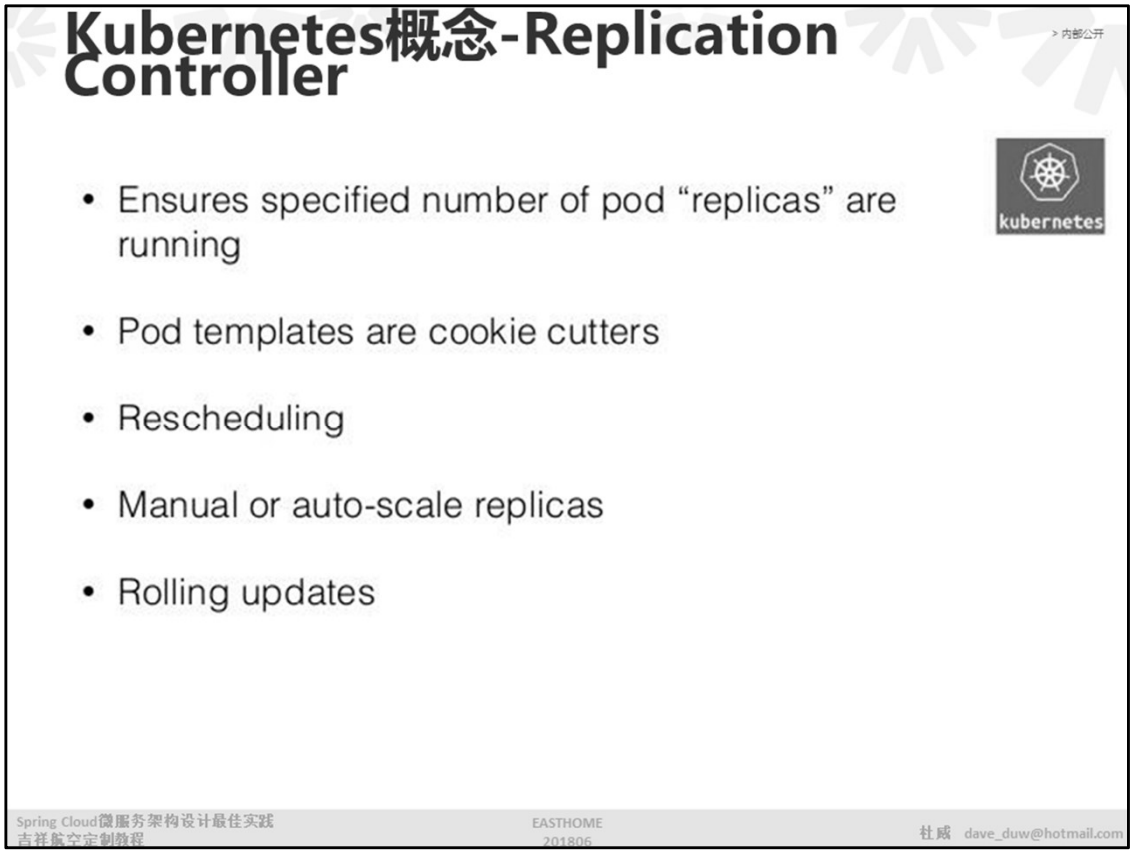
Notes:


> 内部公开
Kubernetes概念-服务Service
kubernetes
• Abstract a set of pods as a single IP and port
• Simple TCP/UDP load balancing
• Creates environment variables in other pods
• Like “Docker links” but across hosts
• Stable endpoint for pods to reference
• Allows list of pods to change dynamically
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

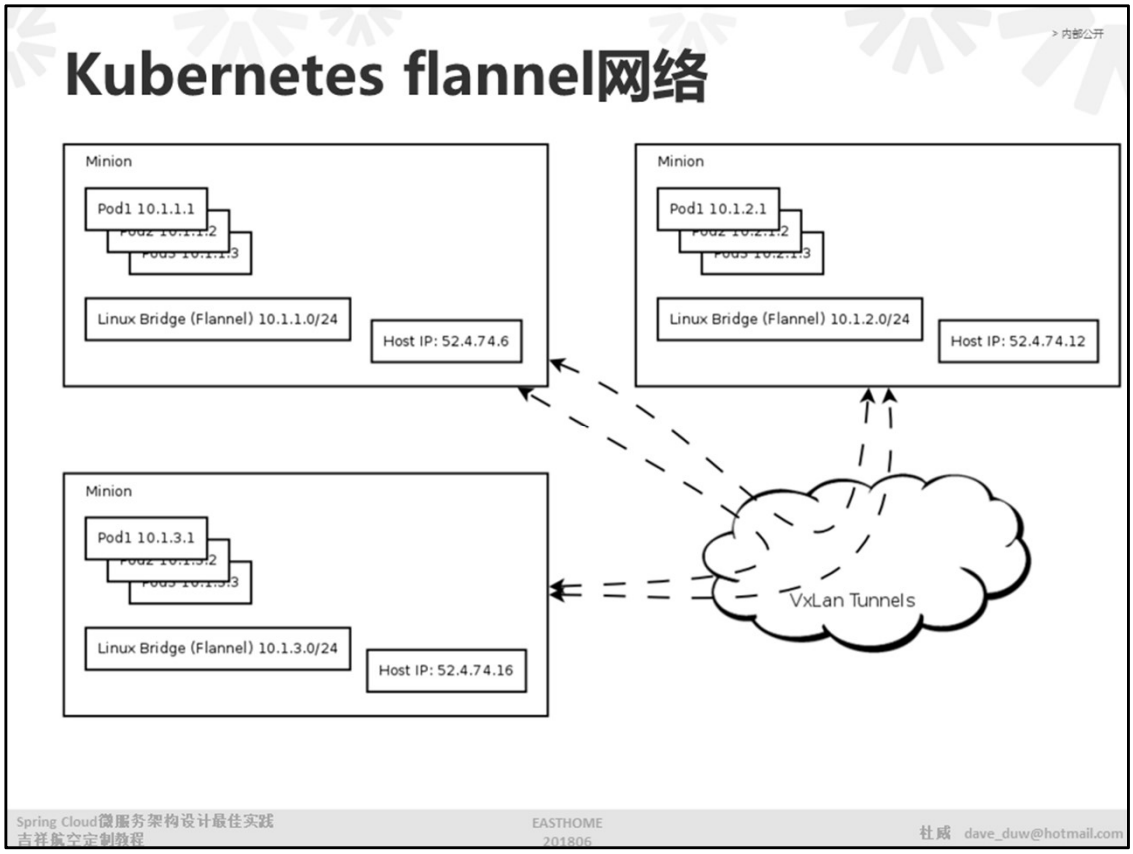
Notes:


> 内部公开
Kubernetes概念-服务与标签
Pod
Pod
Pod
Service
External IP
Service Port
Internal IP
Service Port
Label Selector: name=apache environment=qa
Pod
Container: New
Labels: name=apache environment=qa version=new
Pod
Container Old
Labels: name=apache environment=qa version=old
Pod
Container Old
Labels: name=apache environment=qa version=old
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

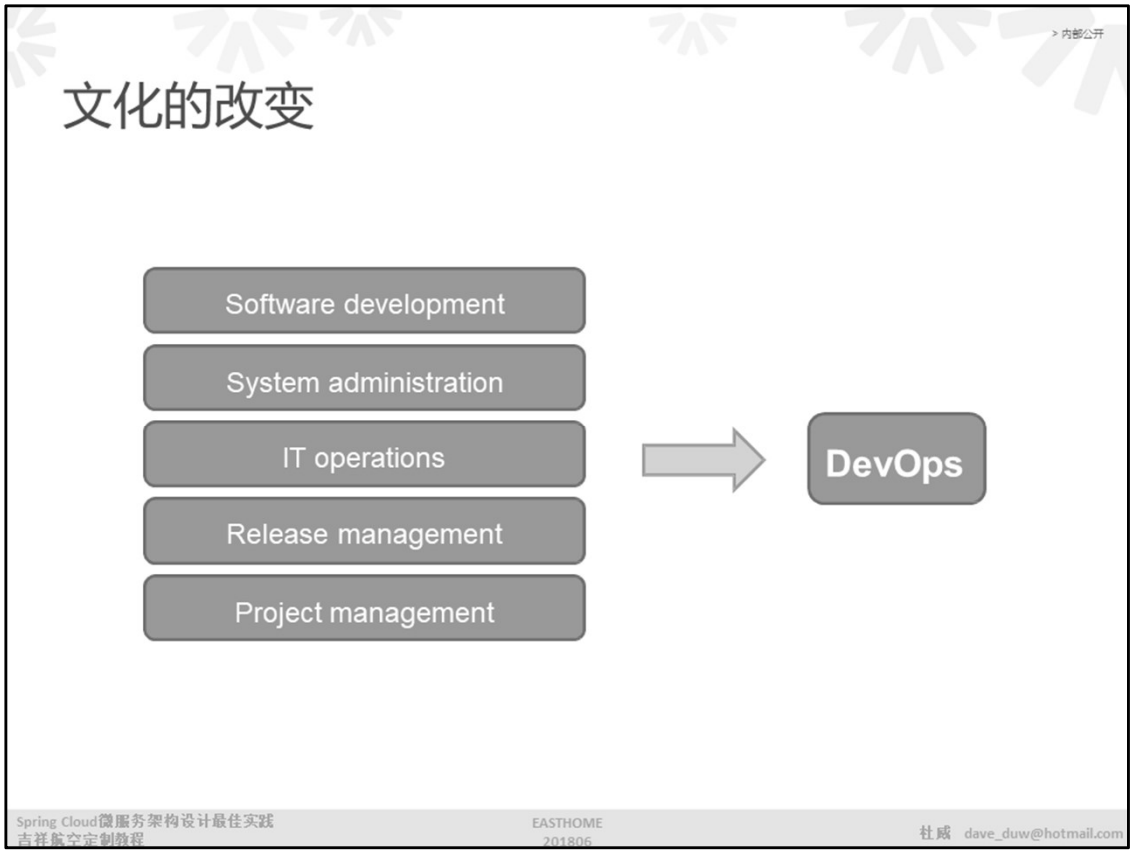

Kubernetes概念-Replication Controller
> 内部公开
kubernetes
• Ensures specified number of pod “replicas” are running
• Pod templates are cookie cutters
• Rescheduling
• Manual or auto-scale replicas
• Rolling updates
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
Kubernetes flannel网络
Minion
Pod1 10.1.1.1
Linux Bridge (Flannel) 10.1.1.0/24
Host IP: 52.4.74.6
Minion
Pod1 10.1.2.1
Linux Bridge (Flannel) 10.1.2.0/24
Host IP: 52.4.74.12
Minion
Pod1 10.1.3.1
Linux Bridge (Flannel) 10.1.3.0/24
Host IP: 52.4.74.16
VxLan Tunnels
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:

Notes:


> 内部公开
文化的改变
Software development
System administration
IT operations
Release management
Project management
DevOps
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

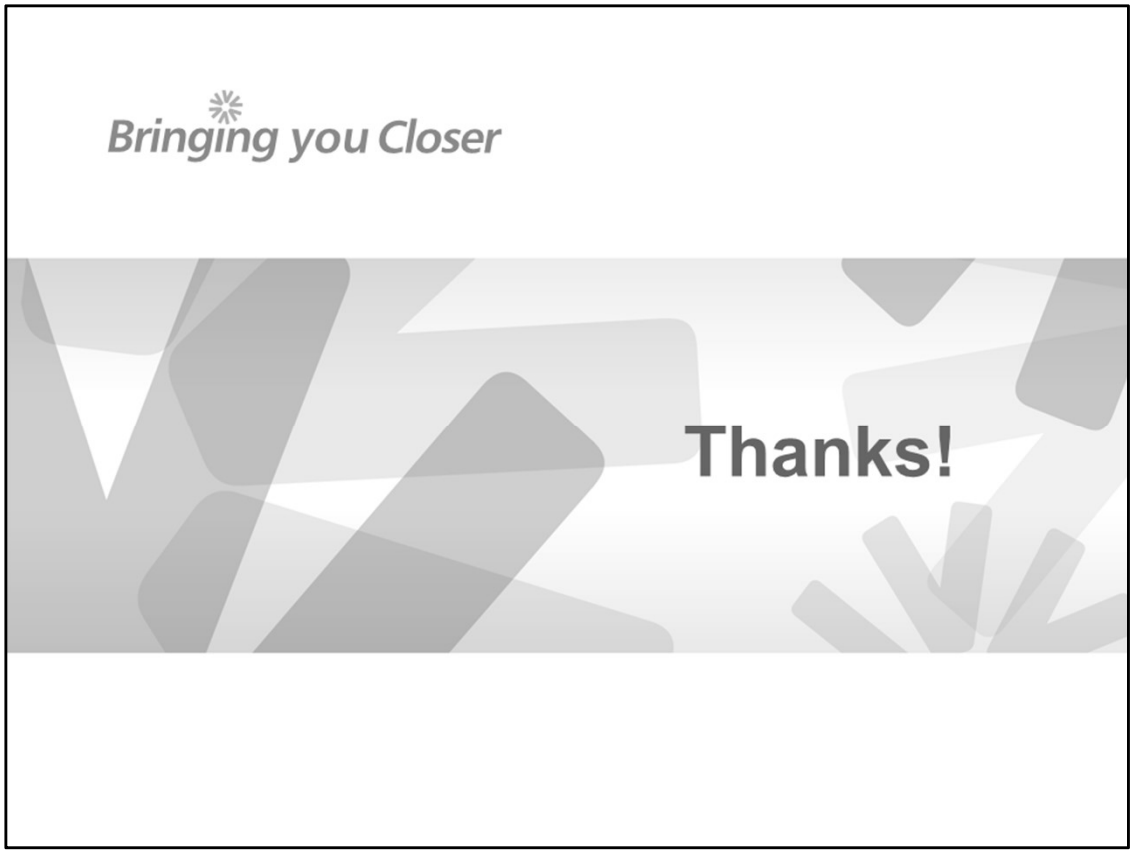
Notes:


> 内部公开
组织 - - 单体应用
• 按照技术能力组织
UI Team
App Logic Team
DBA Team
Organizational Structure
Web Tier
App Tier
DB
Application Architecture
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:


> 内部公开
组织 - - 微服务
• 按照业务职责组织
Login
Registration
Order
Accounts team
Personalization
Personalization team
Mobile
Mobile team
微服务所有权：需求、技术选择、开发、质量、部署、支持
Spring Cloud微服务架构设计最佳实践
吉祥航空定制教程
EASTHOME
201806
杜威 dave_duw@hotmail.com

Notes:



Notes: