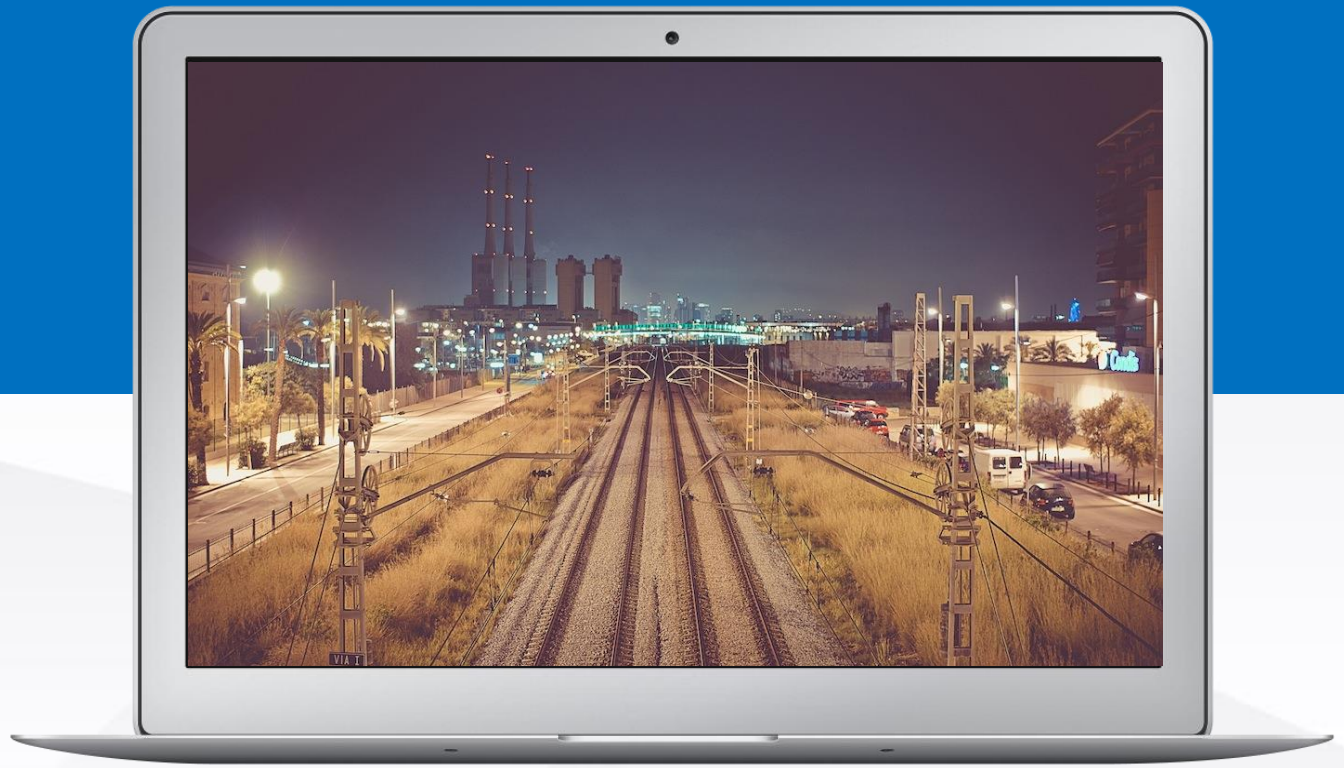
八六三软件股份有限公司
863 Software Co., Ltd.

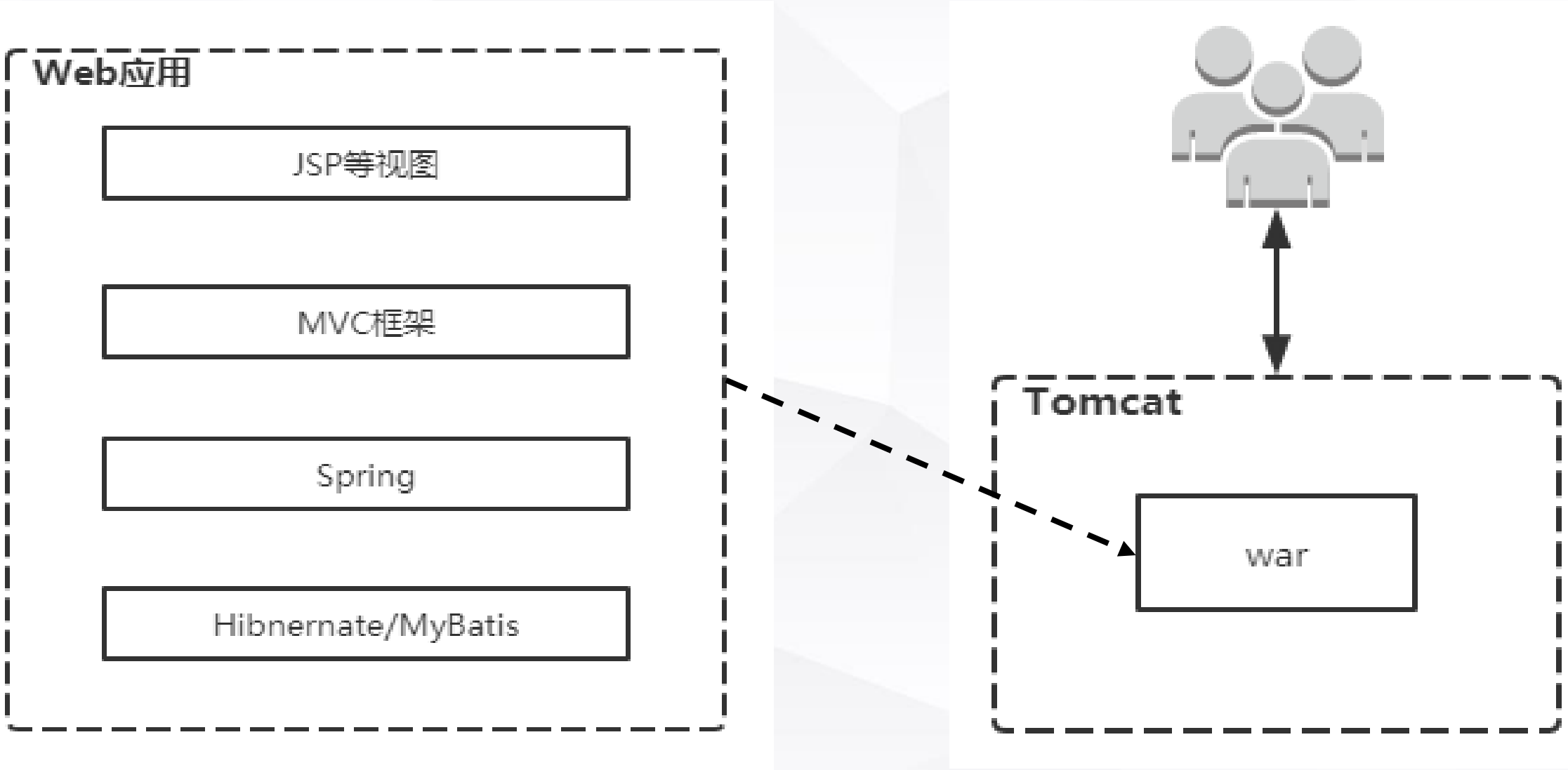
SpringCloud

目录

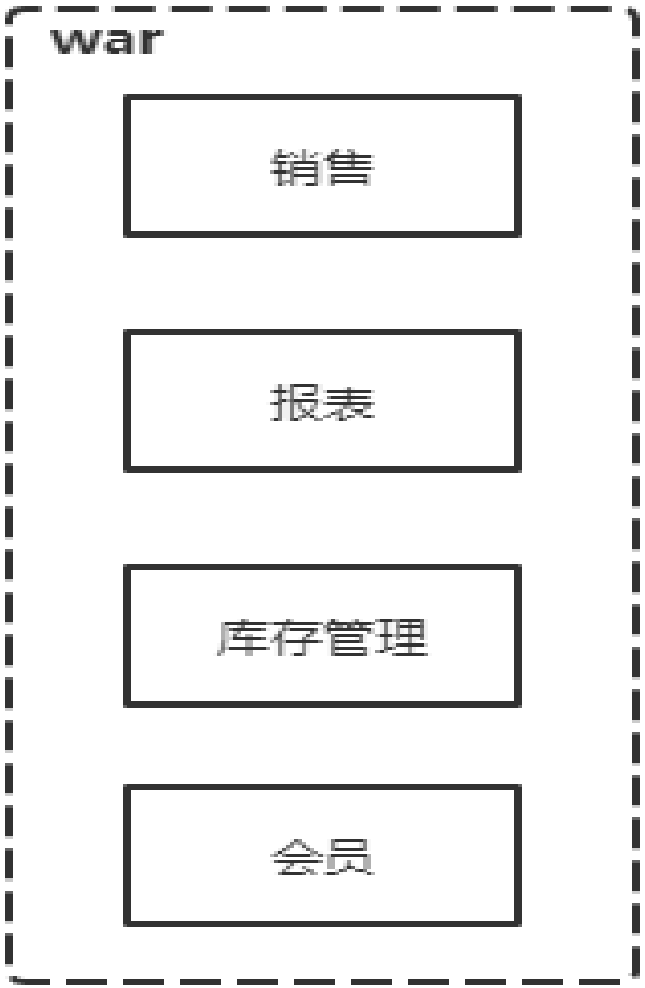
一、微服务架构



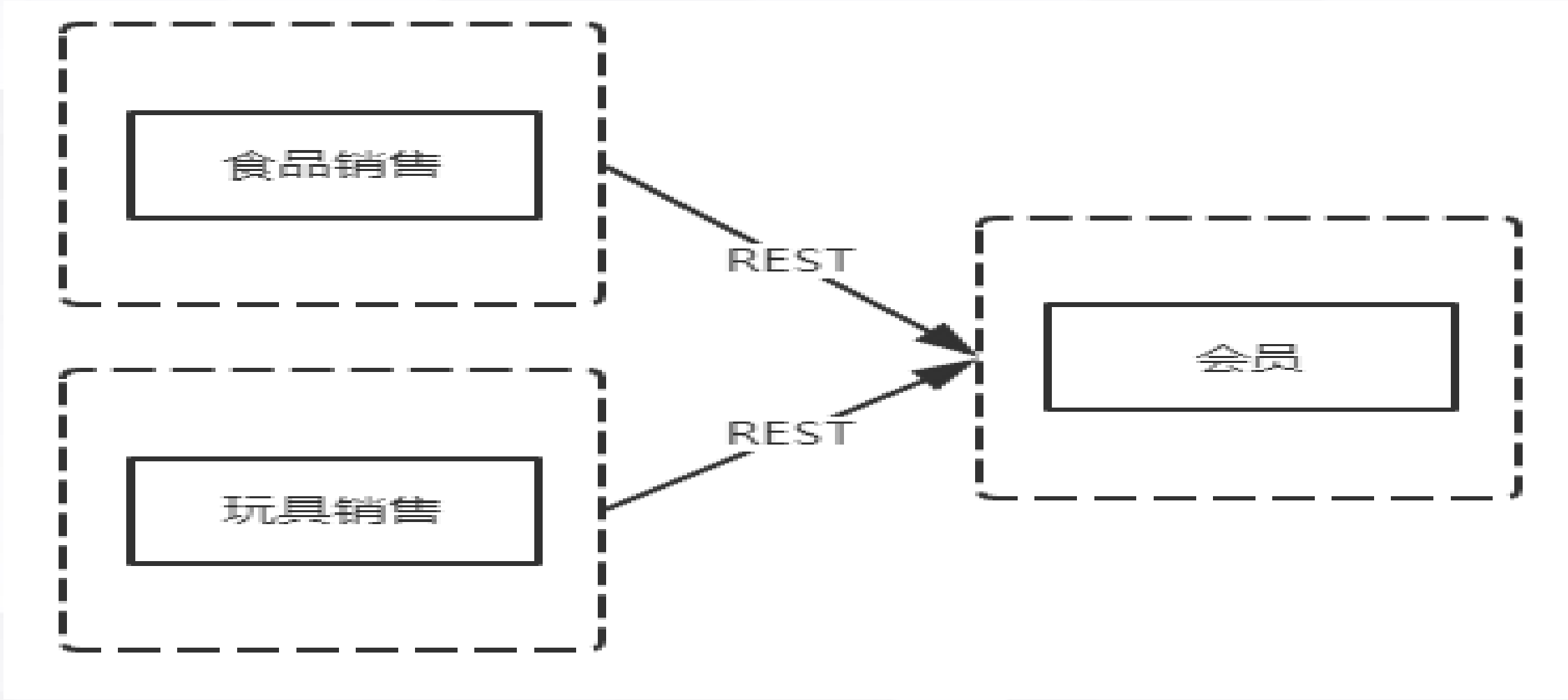
单体架构



单体架构优化



微服务架构



什么是微服务

- 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，并且能够被独立地部署到生产环境、类生产环境等。
- 另外，应尽量避免统一的、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。
- 微服务是一种架构风格，一个大型复杂软件应用由一个或多个微服务组成。系统中的各个微服务可被独立部署，各个微服务之间是松耦合的。每个微服务仅关注于完成一件任务并很好地完成该任务。在所有情况下，每个任务代表着一个小的业务能力。

微服务架构好处

- 使每个微服务组件都是简单灵活的，能够独立部署。应用不需要一个庞大的应用服务器来支撑。
- 可以由一个小团队负责更专注专业，相应的也就更高效可靠。
- 微服务之间是松耦合的，微服务内部是高内聚的，每个微服务很容易按需扩展。
- 微服务架构与语言工具无关，自由选择合适的语言和工具，高效的完成业务目标即可。

微服务挑战

八六三软件股份有限公司
863 Software Co., Ltd.

运维复杂

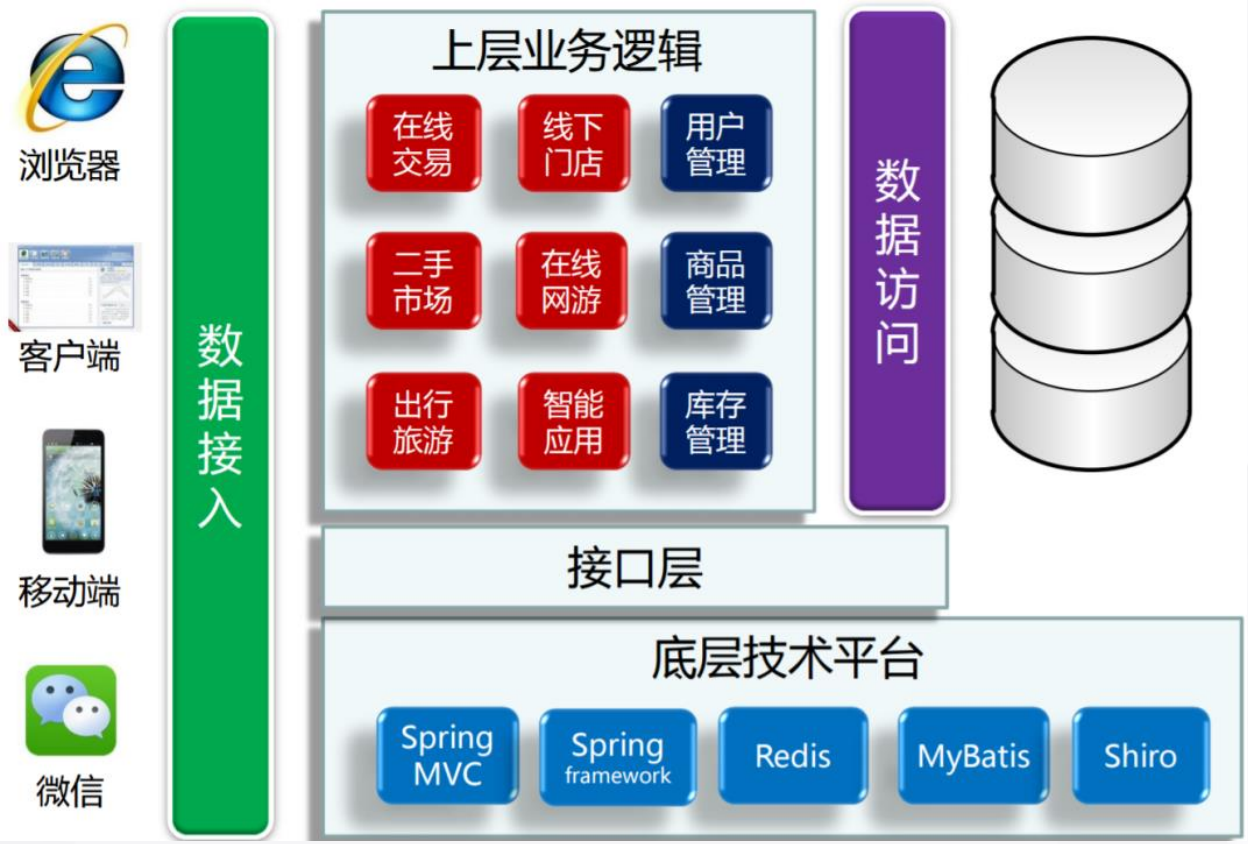
数据一致性问题

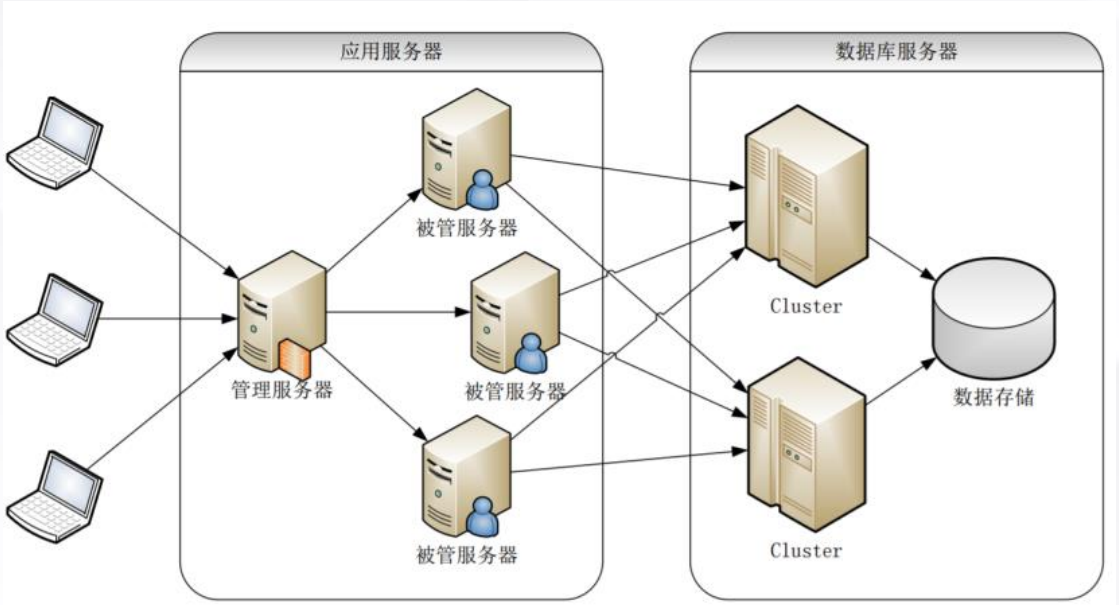
集成测试复杂

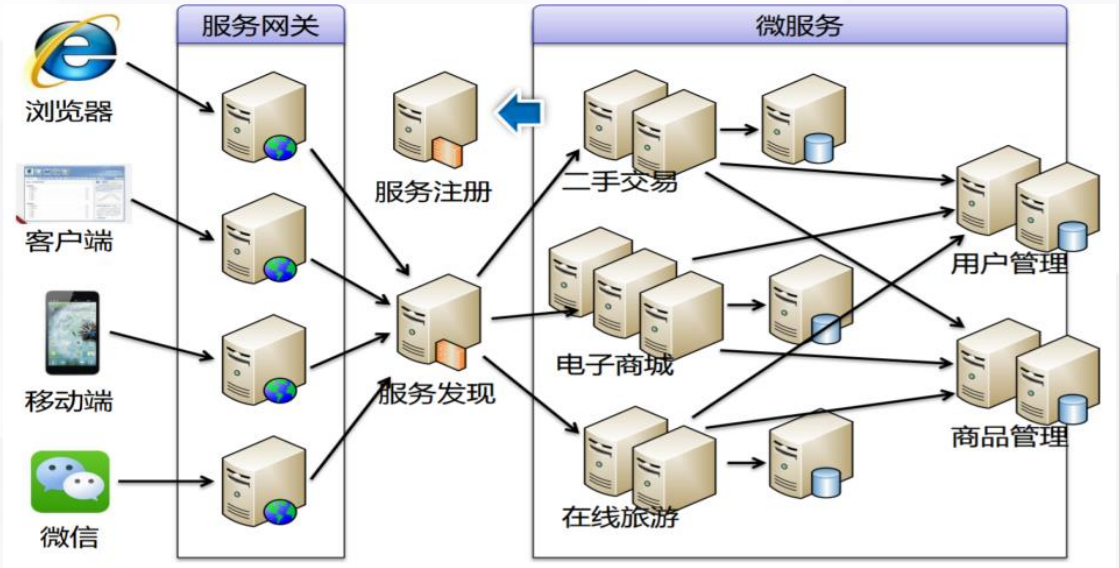
重复代码

监控困难

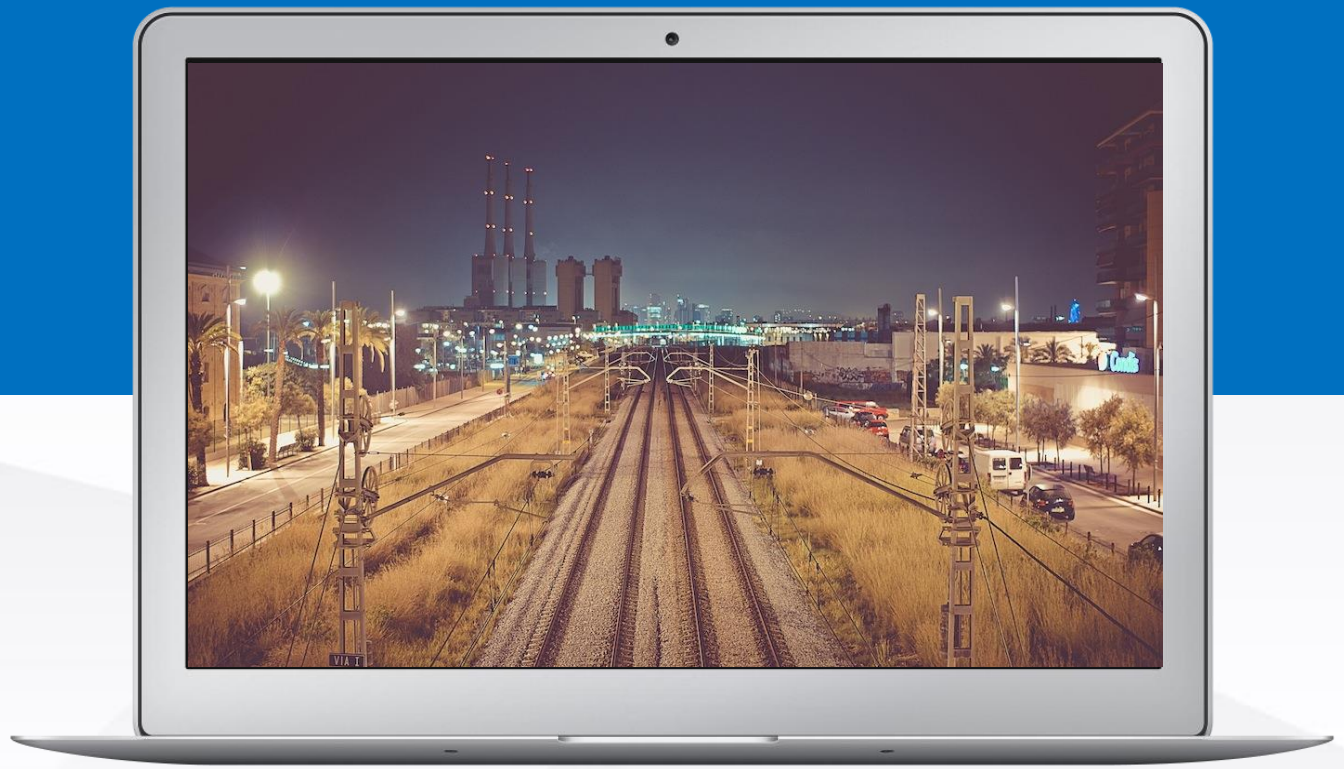
微服务架构示例


浏览器
客户端
移动端
微信
数据接入
上层业务逻辑
在线交易
线下门店
用户管理
二手市场
在线网游
商品管理
出行旅游
智能应用
库存管理
数据访问
接口层
底层技术平台
Spring MVC
Spring framework
Redis
MyBatis
Shiro


应用服务器
数据库服务器
被管服务器
管理服务器
被管服务器
被管服务器
Cluster
Cluster
数据存储


服务网关
微服务
浏览器
客户端
移动端
微信
服务注册
服务发现
二手交易
电子商城
在线旅游
用户管理
商品管理

三、SpringCloud概述



Spring Cloud概述

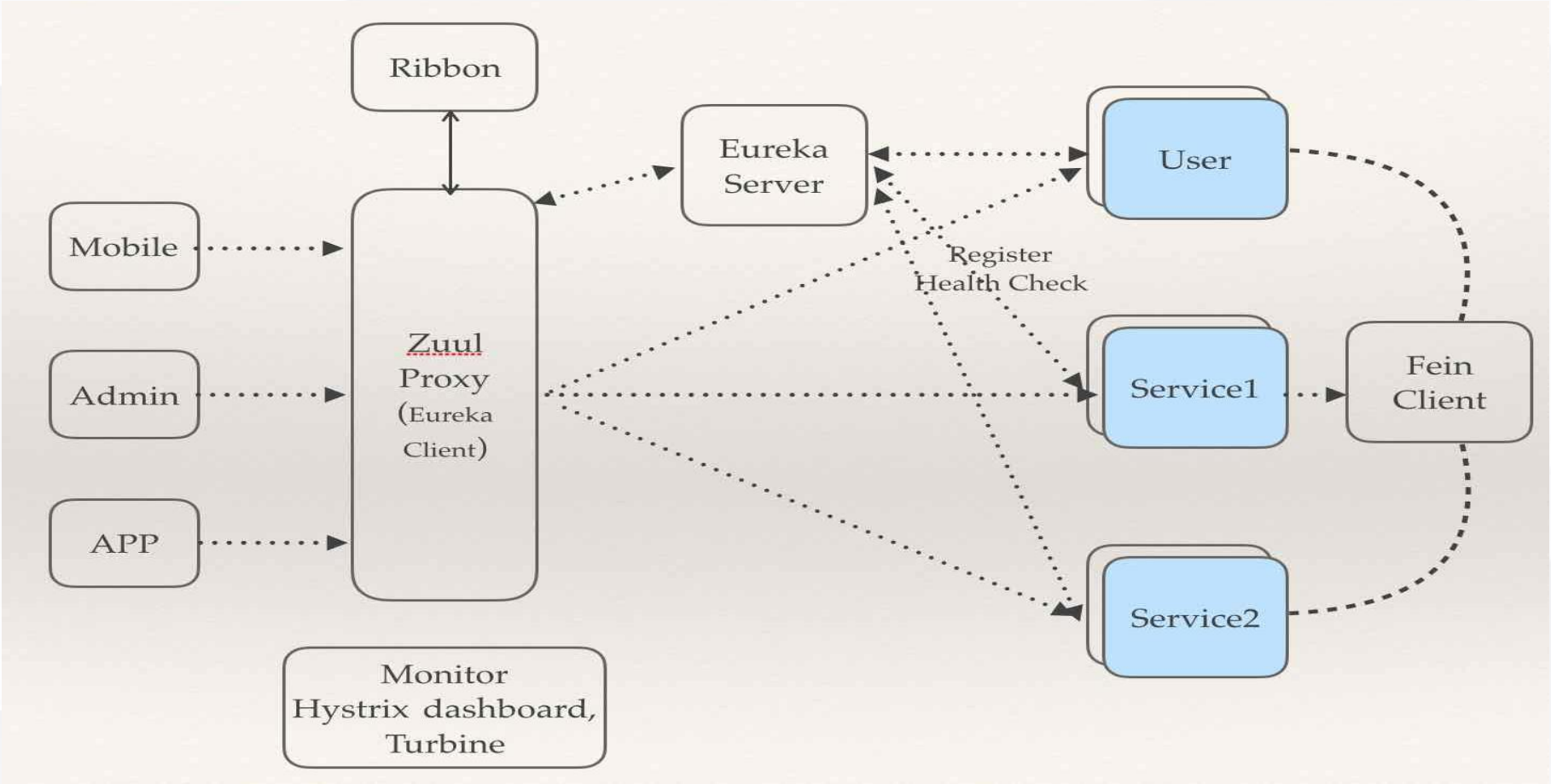
Spring Cloud是**一系列框架的有序集合**。它利用Spring Boot的开发便利性巧妙地**简化**了分布式系统基础设施的开发，如**服务发现与注册、配置中心、消息总线、负载均衡、断路器（熔断器）、数据监控**等，都可以用Spring Boot的开发风格做到一键启动和部署。

Spring**并没有重复制造轮子**，它只是将目前各家公司开发的比较成熟、经得起实际考验的服务框架组合起来，通过Spring Boot风格进行再封装屏蔽掉了复杂的配置和实现原理，最终给开发者留出了一套**简单易懂、易部署和易维护**的**分布式**系统**开发工具包**

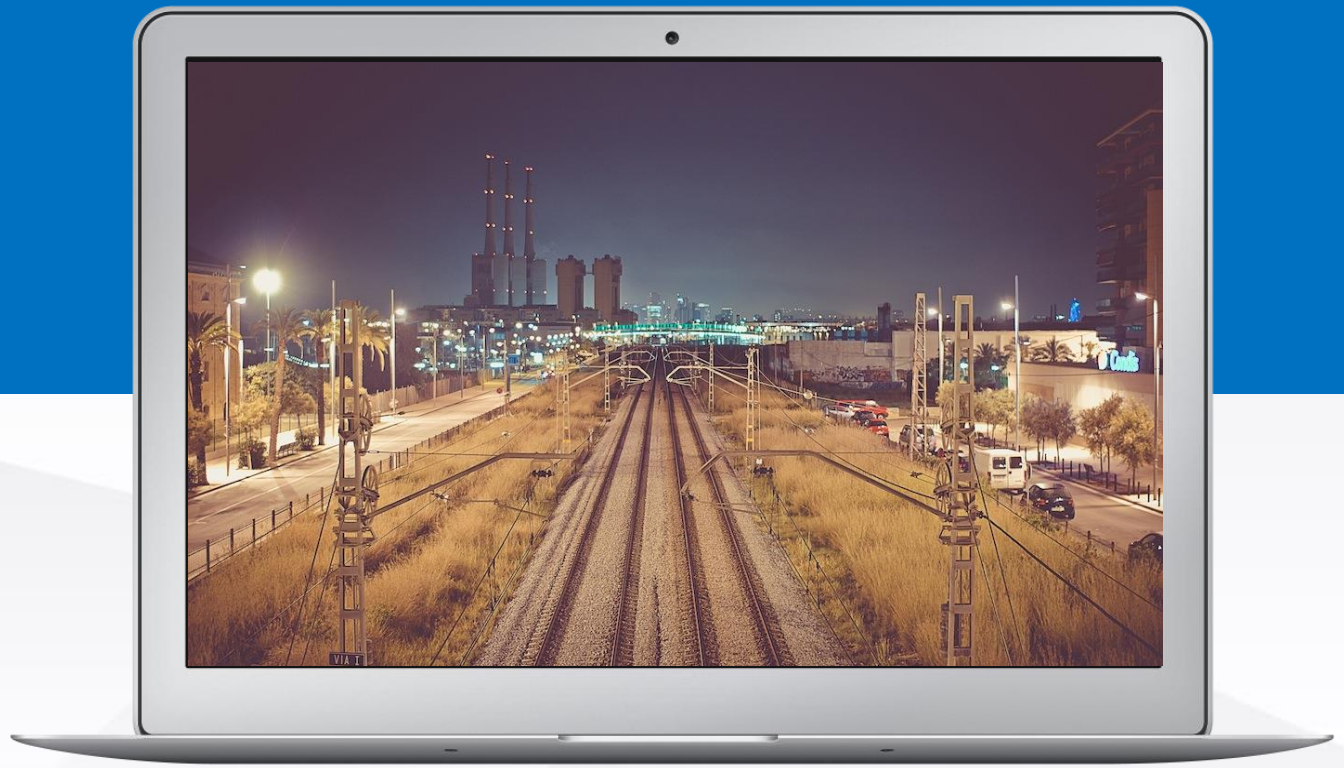
Netflix与Spring Cloud

- Netflix是一家在全球范围内提供流视频服务的公司，截止到2016年已经拥有8300+万订阅用户，每天播放时间达到了1亿2千万小时，是北美互联网峰值下载量的1/3。
- Netflix组件是由Netflix公司开发并开源的一套微服务框架，这套架构在Netflix公司大规模**分布式微服务**环境中经过数年的生产环境检验被证明是可靠的。
- Spring Cloud是基于Spring Boot的一整套实现微服务的框架。
- Spring Cloud Netflix是对Netflix分布式服务开发框架的封装，包括服务发现和注册、负载均衡、断路器、REST客户端、请求路由等

Spring cloud netflix



三、SpringCloud组件



Spring Cloud核心功能

八六三软件股份有限公司 863 Software Co., Ltd.

Eureka：基于REST服务的分布式中间件，主要用于服务管理。云端服务发现，一个基于 REST 的服务，用于定位服务，以实现云端中间层服务发现和故障转移。

Ribbon：负载均衡框架。提供云端负载均衡，有多种负载均衡策略可供选择，可配合服务发现和断路器使用。

Hystrix：容错框架，通过添加延迟阀值以及容错的逻辑，来帮助我们控制分布式系统间组件的交互。熔断器，容错管理工具，旨在通过熔断机制控制服务和第三方库的节点,从而对延迟和故障提供更强大的容错能力。

Feign：一个REST客户端，目的是为了简化Web Service客户端的开发。是一种声明式、模板化的HTTP客户端。

Zuul：为微服务集群提供过代理、过滤、路由等功能。 是在云平台上提供动态路由,监控,弹性,安全等边缘服务的框架。Zuul 相当于是设备和 Netflix 流应用的 Web 网站后端所有请求的前门。

Spring Cloud Config：将配置信息中央化保存，分布式配置中心组件,支持配置服务放在配置服务的内存中（即本地），也支持放在远程Git、SVN。

Spring Cloud Bus：事件、消息总线，用于在集群（例如，配置变化事件）中传播状态变化，可与Spring Cloud Config联合实现热部署。Spring Cloud Zookeeper对Zookeeper的封装，使之能配合其它 Spring Cloud项目使用，一般当作注册中心

其他请参照：https://www.springcloud.cc/

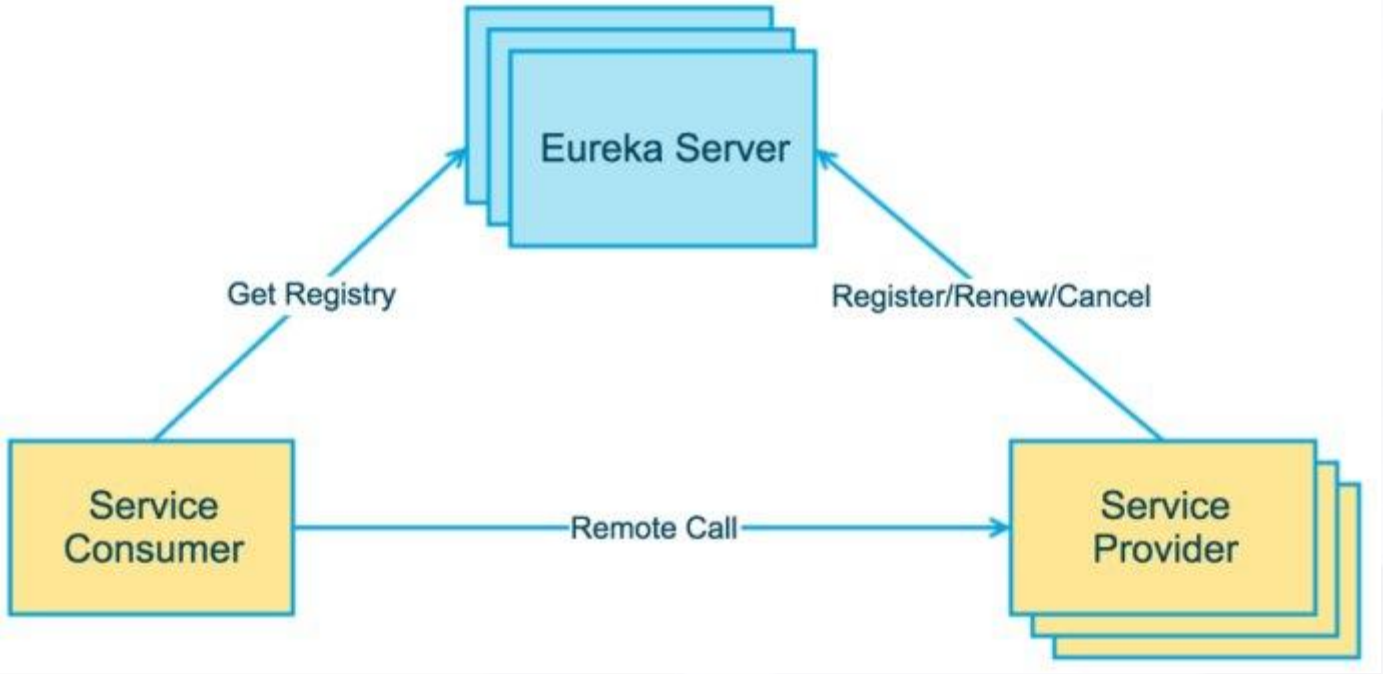
Eureka

Eureka由两个组件组成：Eureka服务器和Eureka客户端。Eureka服务器用作服务注册服务器。Eureka客户端是一个java客户端，用来简化与服务器的交互、作为轮询负载均衡器，并提供服务的故障切换支持。

Eureka的吸引力来源于以下几点：

- **开源**：大家可以对实现一探究竟，甚至修改源码。
- **可靠**：经过Netflix多年的生产环境考验，使用应该比较靠谱省心
- **功能齐全**：不但提供了完整的注册发现服务，还有Ribbon等可以配合使用的服务。
- **基于Java**：对于Java程序员来说，使用起来，心里比较有底。
- **Spring Cloud：**可以使用Spring Cloud, 与Eureka进行了很好的集成，使用起来非常方便。

Eureka



Eureka的高级架构图

1 **Register：服务注册**

当Eureka客户端向Eureka Server注册时，
它提供自身的元数据，比如IP地址、端口，
运行状况指示符URL，主页等。

2 **Renew：服务续约**

Eureka客户会每隔30秒发送一次心跳来续约。 通过续约来告知Eureka Server该Eureka客户仍然存在，没有出现问题。 正常情况下，如果Eureka Server在90秒没有收到Eureka客户的续约，它会将实例从其注册表中删除。

3 **Fetch Registries：获取注册列表信息**

Eureka客户端从服务器获取注册表信息，并将其缓存在本地。客户端会使用该信息查找其他服务，从而进行远程调用。该注册列表信息定期（每30秒钟）更新一次。每次返回注册列表信息可能与Eureka客户端的缓存信息不同， Eureka客户端自动处理。

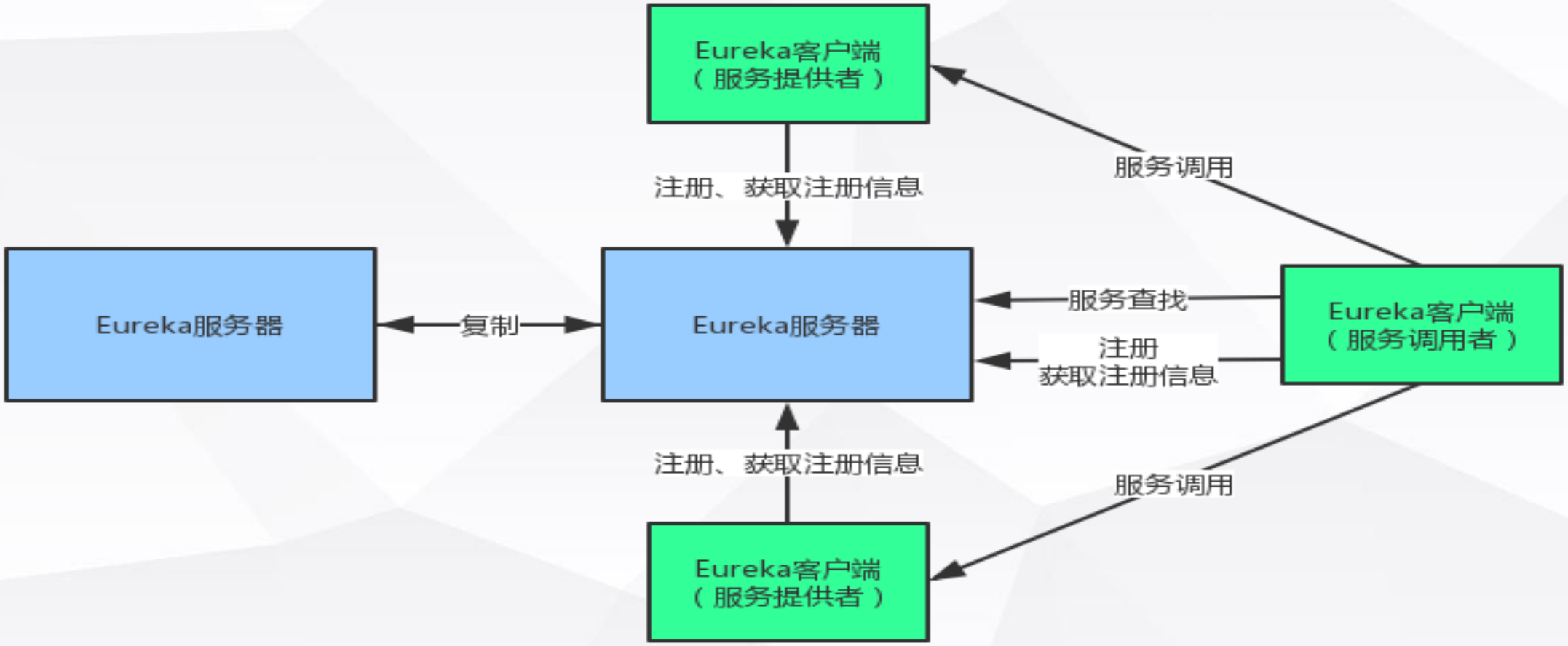
4 **Cancel：服务下线**

Eureka客户端在程序关闭时向Eureka服务器发送取消请求。 发送请求后，该客户端实例信息将从服务器的实例注册表中删除。该下线请求不会自动完成，它需要调用以下内容：
DiscoveryManager.getInstance().shutdownComponent();

5 **Eviction 服务剔除**

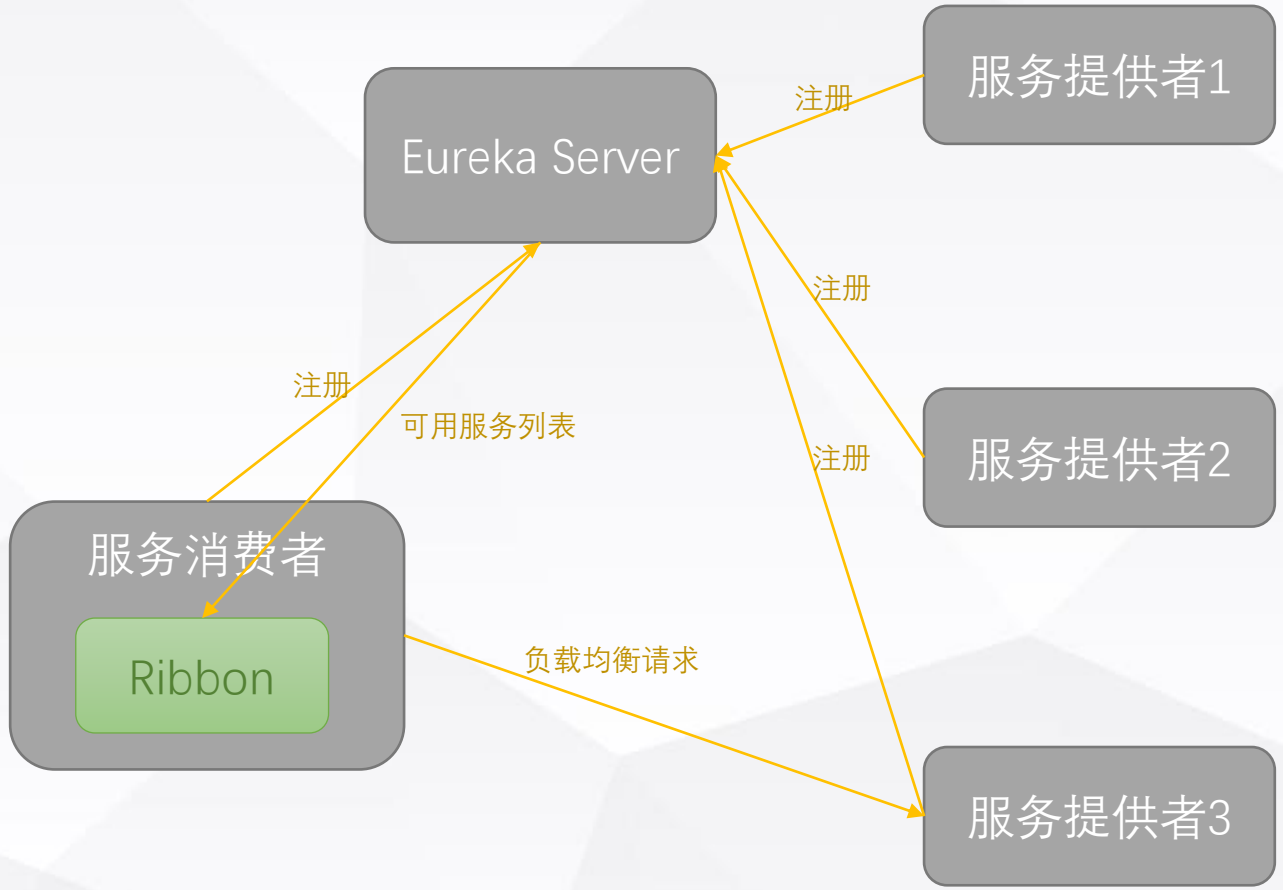
在默认的情况下，当Eureka客户端连续90秒没有向Eureka服务器发送服务续约，即心跳，Eureka服务器会将该服务实例从服务注册列表删除，即服务剔除。

Eureka架构


Eureka客户端
（服务提供者）
注册、获取注册信息
服务调用
Eureka服务器
复制
Eureka服务器
服务查找
注册
获取注册信息
Eureka客户端
（服务调用者）
注册、获取注册信息
服务调用
Eureka客户端
（服务提供者）

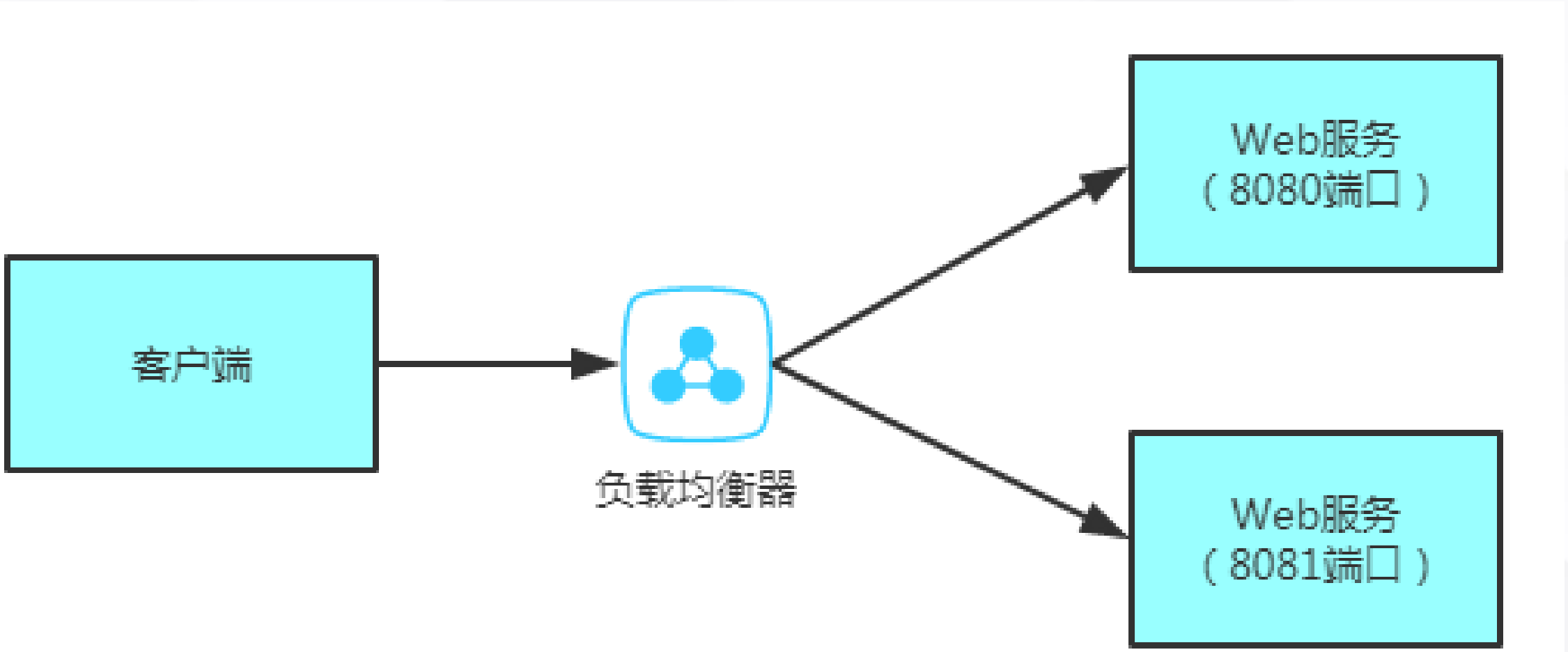
Ribbon简介

- 负载均衡框架，支持可插拔式的负载均衡规则
- 支持多种协议，如HTTP、UDP等
- 提供负载均衡客户端



Ribbon架构图

Ribbon程序



负载均衡器组件

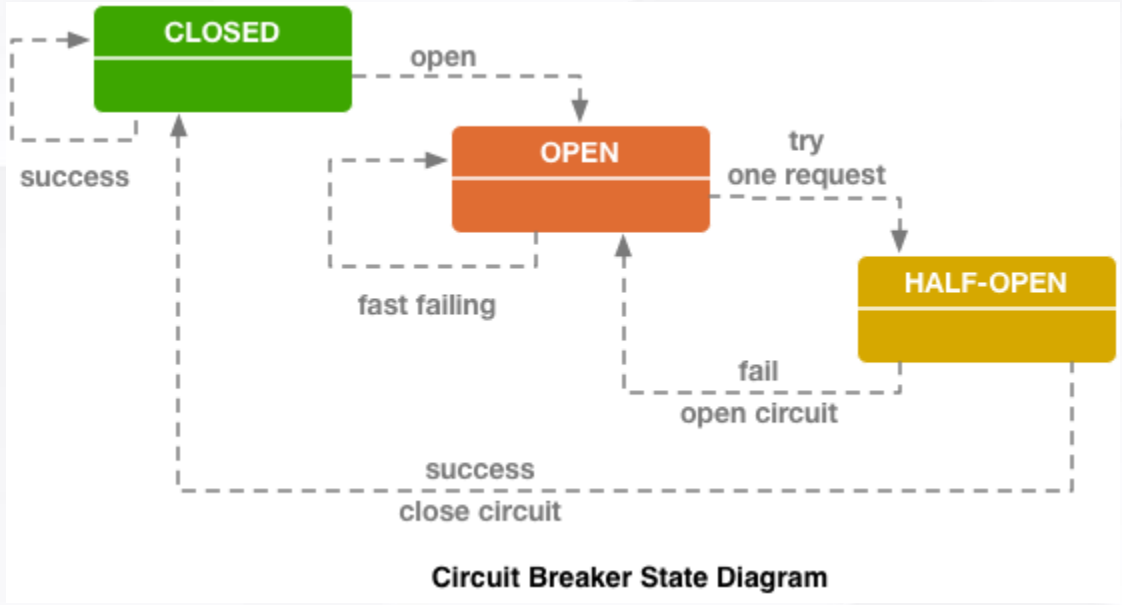
- 一个负载均衡器，至少提供以下功能：
  - 要维护各个服务器的IP等信息
  - 根据特定逻辑选取服务器
- 为了实现基本的负载均衡功能，Ribbon的负载均衡器有三大子模块：
  - Rule
  - Ping
  - ServerList

Hystrix

Hystrix 是一个帮助解决分布式系统交互时**超时处理**和**容错**的类库，它同样拥有保护系统的能力

在微服务架构中，我们将业务拆分成一个个的服务，服务与服务之间可以相互调用。为了保证其高可用，单个服务又必须集群部署。由于**网络原因或者自身**的原因，服务并不能保证服务的100%可用，如果单个服务出现问题，调用这个服务就会出现网络延迟，此时若有大量的网络涌入，会形成任务累计，导致服务瘫痪，甚至导致服务“雪崩”。为了解决这个问题，就出现熔断器模型。

Hystrix作用



Circuit Breaker State Diagram

Hystrix架构图

当Hystrix Command请求后端服务失败数量超过一定比例(默认50%), 断路器会切换到开路状态(Open). 这时所有请求会直接失败而不会发送到后端服务. 断路器保持在开路状态一段时间后(默认5秒), 自动切换到半开路状态(HALF-OPEN). 这时会判断下一次请求的返回情况, 如果请求成功, 断路器切回闭路状态(CLOSED), 否则重新切换到开路状态(OPEN).

1 服务雪崩效应形成的原因

- 服务提供者不可用
- 重试加大流量
- 服务调用者不可用

2 Hystrix的设计原则包括:

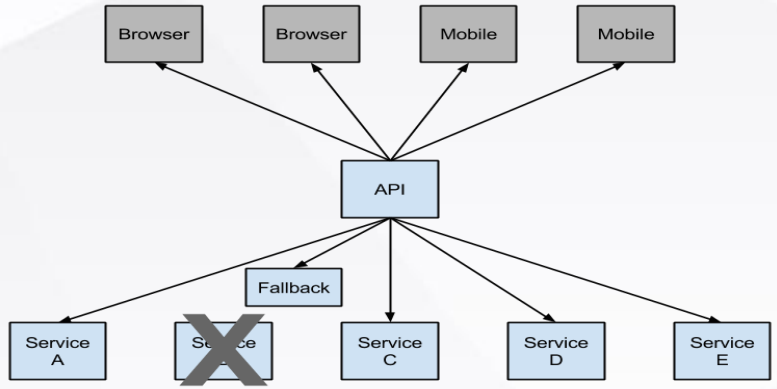
- 资源隔离
- 熔断器

3 熔断器的概念

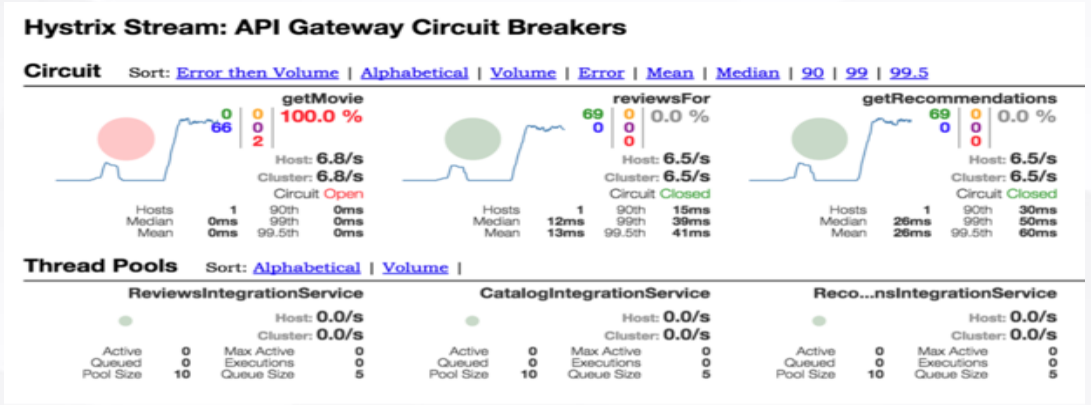
熔断器即断路器，断路器(Cricuit Breaker)是一种能够在远程服务不可用时自动熔断(打开开关)，并在远程服务恢复时自动恢复(闭合开关)的设施，Spring Cloud通过Netflix的Hystrix组件提供断路器、资源隔离与自我修复功能。

Hystrix系列

- **Hystrix** 监控和断路器。我们只需要在服务接口上添加Hystrix标签，就可以实现对这个接口的监控和断路器功能。
- **Hystrix Dashboard** 监控面板，他提供了一个界面，可以监控各个服务上的服务调用所消耗的时间等。
- **Hystrix Turbine** 监控聚合，使用Hystrix监控，我们需要打开每一个服务实例的监控信息来查看。而Turbine可以帮助我们把所有的服务实例的监控信息聚合到一个地方统一查看。这样就不需要挨个打开一个个的页面一个个查看。



Hystrix



Hystrix Dashboard

Feign介绍

- Feign是一个声明式的伪Http客户端，它使得写Http客户端变得更简单。使用Feign，只需要创建一个接口并注解。它具有可插拔的注解特性，可使用Feign 注解和JAX-RS注解。Feign支持可插拔的编码器和解码器。Feign整合了Ribbon 与Hystrix，并和Eureka结合，能够实现负载均衡和断路器等效果。
- 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。

简而言之：

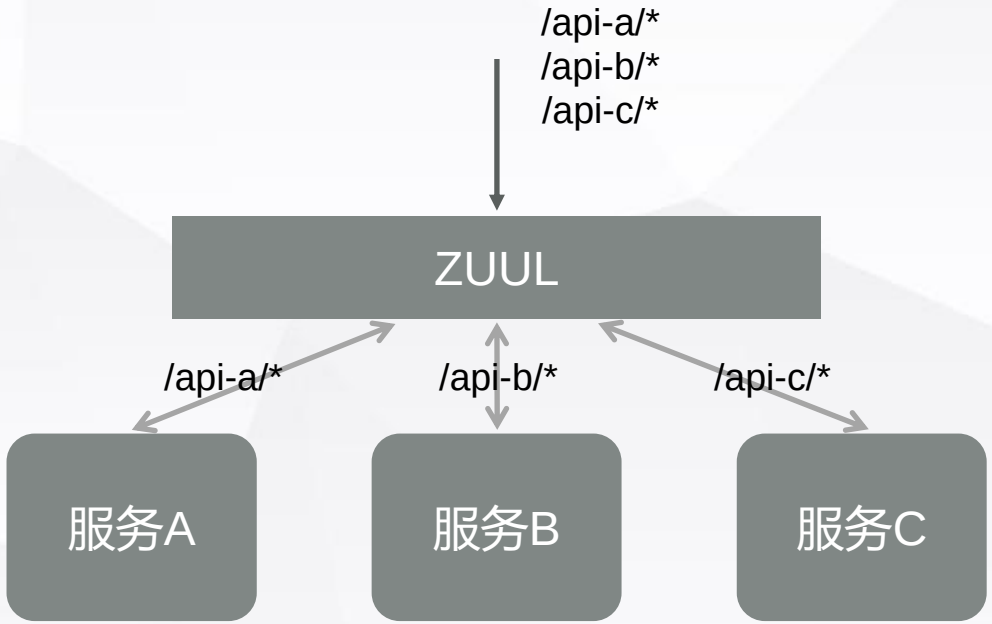
- Feign 采用的是基于接口的注解
- Feign 整合了ribbon、 Hystrix

Zuul

Zuul的主要功能是路由转发和过滤器。路由功能是微服务的一部分，所有的客户端请求通过这个网关访问后台的服务。它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。比如 / api/user转发到到user服务，/api/shop转发到到shop服务。zuul默认和Ribbon结合实现了负载均衡的功能

是Netflix的一个子项目

- 提供代理、过滤、路由等功能
- 天生自带Hystrix 和Ribbon 支持

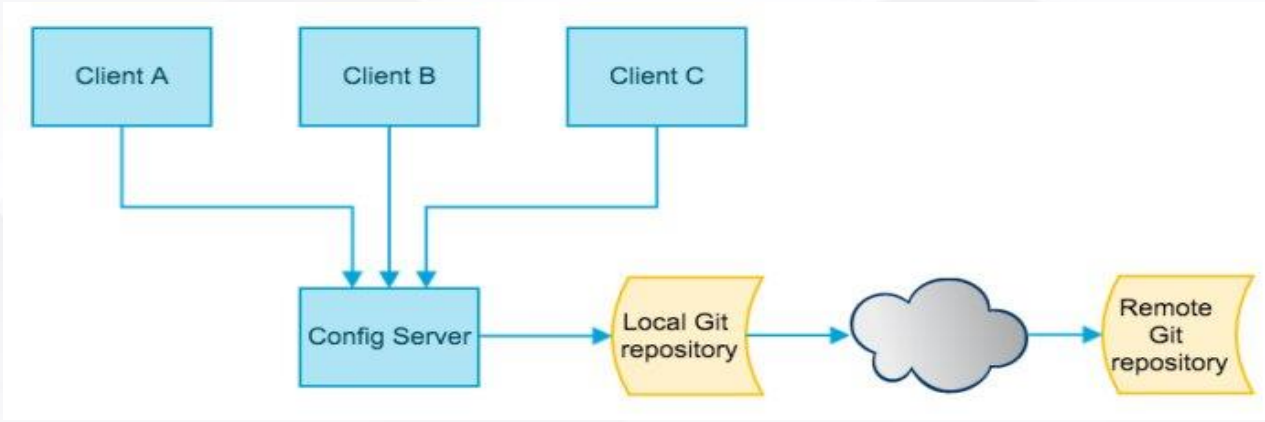


SpringCloudConfig介绍

用来为分布式系统中的基础设施和微服务应用提供**集中化的外部配置支持**， 它分为**服务端**与**客户端**两个部分。

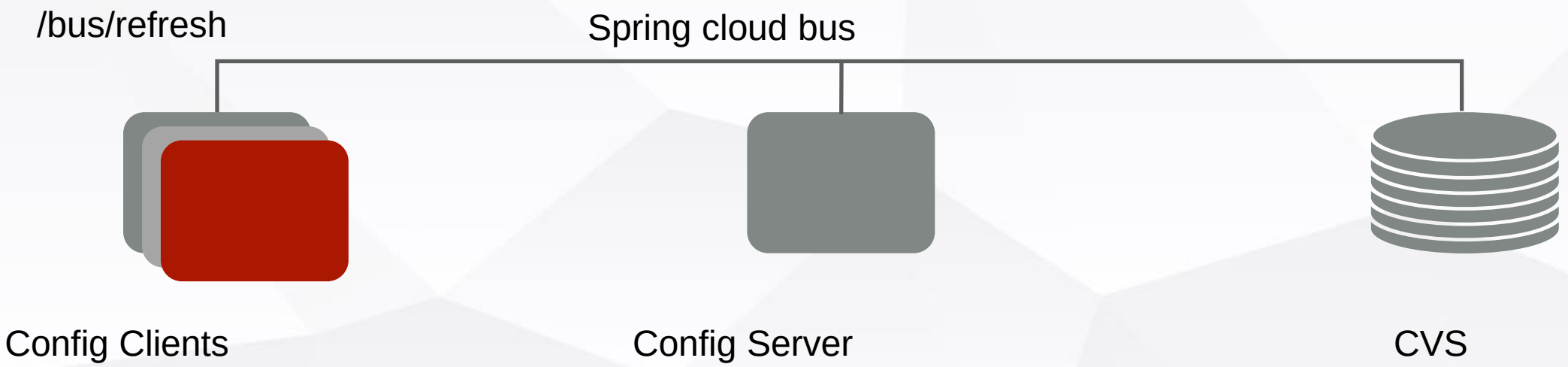
服务端也称为**分布式配置中心**， 它是一个独立的微服务应用，用来连接配置仓库并为客户端提供获取配置信息、加密/解密信息等访问接口；

而客户端则是微服务架构中的**各个微服务应用**或基础设施， 它们通过指定的配置中心来管理应用资源与业务相关的配置内容，并在启动的时候从配置中心获取和加载配置信息。

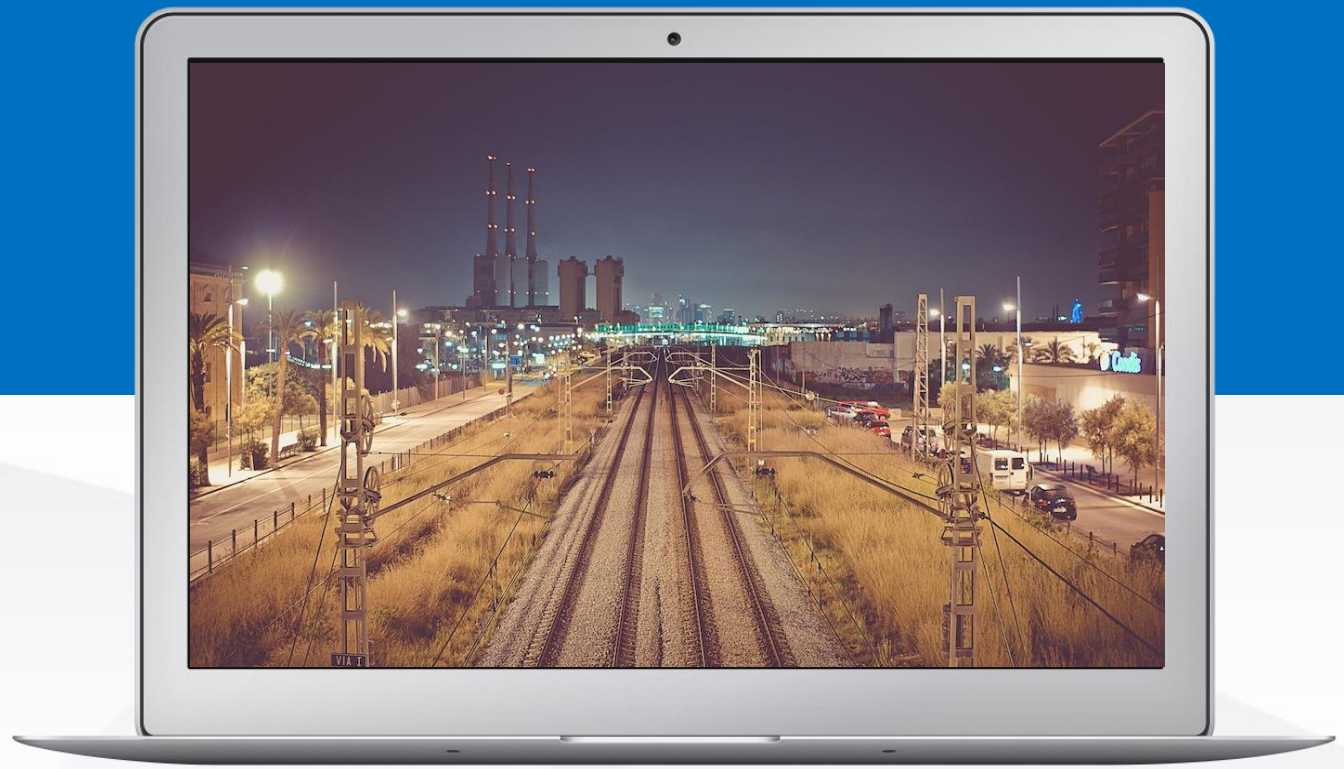


SpringCloudBus

事件、消息总线，用于广播应用状态变更到分布式系统中的各个关联的节点。应用节点不直接相互通讯，而通过消息总线来实现通知,用于在集群（例如，配置变化事件）中传播状态变化，可与Spring Cloud Config联合实现热部署。



四、 SpringCloud案例



演示