知识回顾：

1. SpringCloud微服务框架，使用约定大于配置思想，注解手段，共同作用，把公用通用，模式化代码都底层实现，开发人员工作量急剧减少，而且很简单就拥有超强能力。
2. Eureka实现注册中心，解耦，非硬编码。动态性。动态维护列表，当服务开启，它会自动注册到列表中（动态扩展），如果这个服务宕机，动态从列表中移除（动态删除）。因为心跳机制形成动态。由客户端发出心跳。短链接（Request Response请求响应模式，每次用户请求会创建新的Request对象。），长链接（复用第一次通道，不需重复创建请求、响应对象。TCP/IP三次握手。）。Chat聊天websocket。Ping
3. Ribbon负载均衡，Nginx反向代理（安全性，屏蔽内网中真实服务器所在地址）前置机器，踏板机，机器双网卡（一个对外网暴露，另外一个网卡就和内网通讯）Nginx编程，编程语言，脚本语言，替代shell编程。（xxx.sh）lua语言。黑名单、白名单、限流（超过负载能力，变慢，拒绝访问，宕机）令牌桶算法1000。默认算法：轮询，随机，最少请求数。
4. Feign声明式WebService。两种方式：
   1. RestTemplate需要学习新的API，学习成本高。代码维护成本高。
   2. Feign接口体现，接口中方法都是跟业务相关。代码后期维护方便。Sb1.5.4@PathVariable（”name”）String name; 2.1.1已经不需要。
   3. Feign底层消费者怎么去访问提供者的？httpclient

今天任务：

Springboot 1天，SpringCloud 5天

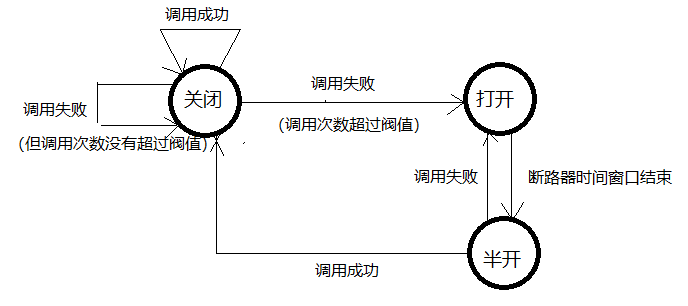
1. Hystrix断路器
2. Zuul APIGateway API网关
3. Sidecar 异构开发语言支持
4. NodeJS



传统项目中try/catch，捕获异常，程序总有出错时候

微服务，微服务有可能访问不了了。

1. 超时，timeout实际开发中必须设置。调用业务了吗？超时机制会先调用业务，其实已经操作的内容，只是不能完成。（1分钟、5分钟、30分钟、1天、7天、15天、30天）
2. 断路器，熔断器。保护资源，服务已经不堪重负，宕机，无法响应。很短暂超时（1s）。每次访问时，先测一下。如果失败，继续直接返回。一旦通了，业务又正常访问。



断路器的三种状态：

1. 关闭，服务是正常，断路器没起作用。
2. 打开，直接快速响应，返回，通知用户服务不可用。用户就不浪费时间。
3. 半开，有错误时，继续尝试，直到没有错误。转成“关闭”状态

服务升级，动态在做。



京东价格微服务宕机。给用户返回什么呢？

返回一个假价格！

双11，马云3w+服务器服务，万兆宽带。销售系统CRM，人财物管理系统ERP，办公自动化系统OA。降级，把非关键业务都停止。过了双11，服务要回复。

Hystrix基于Eureka、Ribbon、Feign

1. 引入Feign，先引入Hystrix
2. SpringCloud没有直接引入Hystrix，引入javanica封装过的Hystrinx

封装后访问api简单易用

@HystrixCommand(fallbackMethod = "helloFallback")

对应写这个方法，参数，和返回值，必须和之前controller对应方法值一样

@EnableCircuitBreaker 断路器

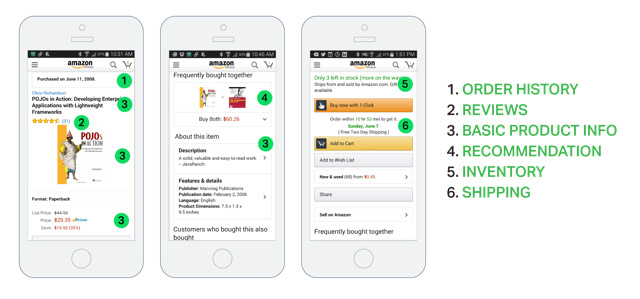
断路器第一次访问有“bug”，第一次，访问编译，超时。第一次出现访问断路器。不管。

API GateWay API网关

程序员功能，写API卖钱。

例如：天气预报

1. 授权，类似shiro，增加很多filter，形成filter链。
2. 收费，根据调用次数来收费。
3. 转发请求，所有的请求都必须经过API网关，对这个请求不做业务处理，只负责转发到指定服务，springmvc,nginx



商城购买书籍，

1. order history 历史订单
2. reviews 历史浏览信息
3. basic product info 商品信息
4. recommendation 商品评论
5. inventory 仓储
6. shipping 物流

亚马逊浏览商品页面，调用上百个微服务。

1. 转发
2. 微服务A调用B，B调用C。。。，微服务雪崩
3. 把这些请求合起来，变成一次请求，把所有结果合并，一次返回
4. 消除异构问题。技术对web支持不够好。Thrit序列化方式。ProtocolBuffer谷歌序列化方式，DUBBO

单点风险，API GateWay，

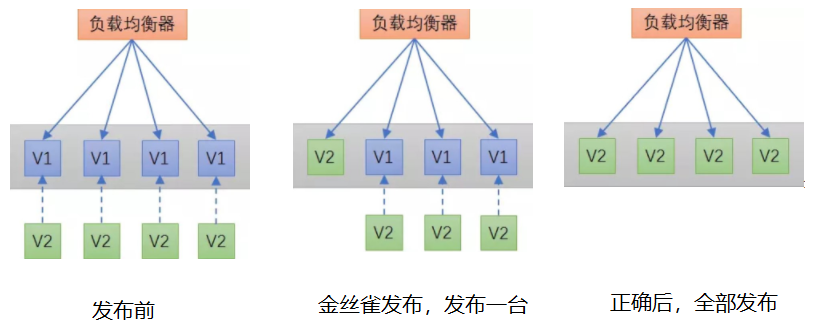
Nginx 5w/s

Zuul 2.0 100w/s

F5硬件 50w/s

单点风险：Zuul，Nginx，Eureka，Config，SSO，redis，集群

灰度测试，金丝雀测试（测试）



传统方式，停机，全部部署新的程序v2，启动服务。

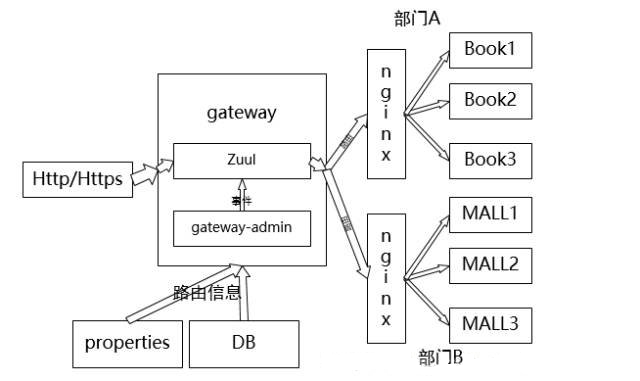
分流，逐步升级，某一个点先把V1升级成V2，用户就被程序进行分流，大量还是按之前业务来完成V1，少量用户就被分流V2上。如果V2程序BUG满天飞，可以直接退回，影响用户非常少。如果V2程序稳定运行一段，全面升级为V2。

程序有多个版本！

税率，每年给企业定税率。2018年1.0，2019年1.2

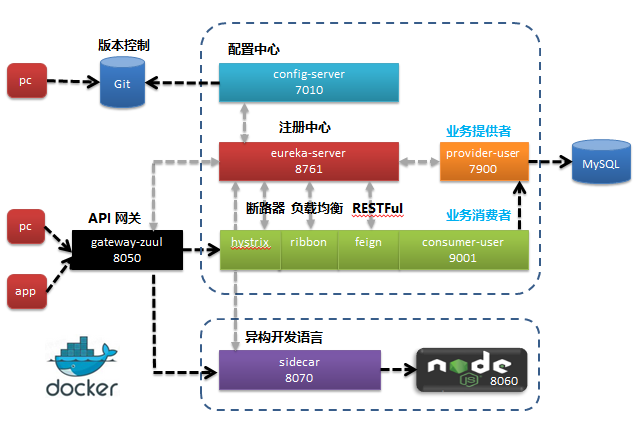
2018年账，1.0

2019年1月1号升新的税率，



Zuul所有的用户的请求必须通过API网关。

Zuul整合Eureka/Ribbon/Feign/Hystix，代码有所变化。（会影响之前运行代码）



Zuul映射配置

zuul:

routes:

app-a: 随便起，不能重名

path: /user/\*\* 所有请求访问时增加一级路径，/\*\*，支持后面多级

serviceId: provider-user 服务application.name



Zuul使用时，就要自己来集成Hystrix。Fallback方法，写在zuul中。

Zuul配置断路器？

1. 单独写一个类，实现接口
2. 在类上加上注解，@Component，Spring扫描时创建实例
3. New一个ClientHttpResponse（httpclient）
4. 设置返回Content（json+U8），错误代码，错误消息。。。

异构语言调用

不同开发语言调用，实际开发中必然遇到问题。

1. 别的开发语言，NodeJS

大前端，js干后端事情，js天生脚本语言（不让它访问本地资源）

NodeJS允许js访问客户端少部分资源。

Nginx,Redis,Docker（go）,Ribbitmq（Erlang）,hadoop（java）,storm(scala)。

Java纯，生态链

岗位：前端工程师

鄙视链：

架构师》高级程序员/系统分析师》中级程序员》项目经理》初级程序员

前端工程师（js/css/html）、测试工程师（他对业务了解不如开发）、运维工程师（环境）

sidecar:

port: 8060 就是NodeJS端口

health-uri: <http://localhost:8060/health.json> NodeJS启动服务健康检查的地址

这个服务不能直接访问，它和Zuul集成，通过Zuul访问。

在Zuul中增加sidecar的映射

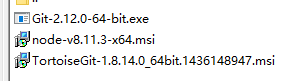
app-sidecar: #通过sidecar调用nodejs的服务

path: /sidecar/\*\*

serviceId: sidecar

小结：

1. SpringCloud Hystrix断路器，熔断器。异常处理。Try/catch。Try已经去做业务，只是做业务时出错了。Hystrix如果业务正常访问，断路器就不使用。当服务不能正常访问时，断路器才开始工作，相当于出错后，回调一个预先设置好的方法。方法是有要求的，返回值和返回值类型必须和调用方法一致，参数必须和调用方法的参数一致，包括参数类型。方法名不同，配置的回调名称。fallbackMethod=name。一旦修改好了，或者升级完成。重启加入Eureka中。断路器每次都会尝试一下，服务是否可以访问了，一旦服务正常，立即启用，不在使用断路器。
2. Hystrix三种状态：关闭，半开（过一定时间窗口），打开（出错次数超过阀值，默认15次，防止网络涌动）
3. GateWay网关 Zuul2.0 100w/s链接转发。降级（电商网站、CRM销售管理系统、ERP人财物管理系统、OA办公自动化系统。。。在大促时，对非业务关键的系统暂时停掉，大促后再启用。断路器。）限流Nginx网关（小漏斗，1000令牌）
4. Sidecar异构开发语言的支持。NodeJS大前端技术。类似tomcat。RESTFul+json



Git版本控制，代码管理

练习：

1. 安装Git，DOS命令窗口
2. Tortoise小乌龟，可视化操作
3. https://github.com/ 大家注册一个自己的账号