### 走一遍

systemctl stop firewalld.service #停止firewall

systemctl disable firewalld.service #禁止firewall开机启动

firewall-cmd --state #查看默认防火墙状态（关闭后显示notrunning，开启后显示running）

# @EnableDiscoveryClient与@EnableDiscoveryClient的区别？

SpringCLoud中的“Discovery Service”有多种实现，比如：eureka, consul, zookeeper。

1，@EnableDiscoveryClient 注解是基于 spring-cloud-commons依赖，并且在classpath中实现；

2，@EnableEurekaClient 注解是基于 spring-cloud-netflix依赖，只能为eureka作用；

简单的说，就是如果选用的注册中心是eureka，那么就推荐@EnableEurekaClient，如果是其他的注册中心，那么推荐使用@EnableDiscoveryClient

# springcloud的yml常用配置

### eureka常用参数：

eureka.client.register-with-eureka: 是否向注册中心注册自己（true/false）

eureka.client.fetch-registry: 是否获取服务清单（一般服务消费者设置为true，默认为true)

eureka.client.registry-fetch-interval-seconds=30 检测服务的时间，单位秒

eureka.client.serviceUrl.defaultZone=http://localhost:8080/eurekaServer 指定

enable-self-preservation: 是否开启自我保护机制

eureka.client.eureka.enabled: 是否去用eureka客户端

eureka.client.registryFetchIntervalSecondes: 从eureka服务端获取注册信息的间隔时间/s

eureka.client.healthcheck.enabled: 是否开启健康监测

自我保护机制的工作机制是如果在15分钟内超过85%的客户端节点都没有正常的心跳，那么Eureka就认为客户端与注册中心出现了网络故障，Eureka Server自动进入自我保护机制，此时会出现以下几种情况：

1、Eureka Server不再从注册列表中移除因为长时间没收到心跳而应该过期的服务。

2、Eureka Server仍然能够接受新服务的注册和查询请求，但是不会被同步到其它节点上，保证当前节点依然可用。

3、当网络稳定时，当前Eureka Server新的注册信息会被同步到其它节点中。

因此Eureka Server可以很好的应对因网络故障导致部分节点失联的情况，而不会像ZK那样如果有一半不可用的情况会导致整个集群不可用而变成瘫痪

### zuul:网关配置

zuul.routes.zuul-url.path=/zuul-url/\*\*

zuul.routes.zuul-url.url=http://localhost:8080/

zuul.routes.zuul-url.path中的zuul-url是路由的名字，可自行替换。

### 面向服务的路由

zuul.routes.zuul-url.path=/zuul-url/\*\*

zuul.routes.zuul-url.serviceId=test-service

上面的配置方式等同于 zuul.routes.test-service=/zuul-url/\*\*

### 单实例配置

zuul.routes.zuul-url.path=/zuul-url/\*\*

zuul.routes.zuul-url.serviceId=test-service

### 多实例配置

zuul.routes.zuul-url.path=/zuul-url/\*\*

zuul.routes.zuul-url.serviceId=test-service

test-service.ribbon.listOfserver=http://localhost:8080/,http://localhost:8081/

通过使用ribbon实现了路由转发时的负载均衡策略。

由于zuul会自动为在eurekaServer中的每个服务都创建映射关系，但是当我们不想对某个服务进行路由转发时，可以通过

zuul.ignored-services来设置一个服务名匹配表达式，从而达到该服务不被转发。

另外，在实际工作中，可能会有多个路由规则映射到同一路由上，

比如，zuul.routes.test-service.path=/test-service/\*\*

          zuul.routes.test-service.serviceId=test-service

          zuul.routes.test-service.path=/test-service/test1\*\*

          zuul.routestest-service.serviceId=test-service-test1

这样如果访问/test-service/test1\*\*，上面两个都会被匹配到。

因为路由规则是存放到LinkedHashMap里的。是进行排序的，而properties文件是无法保证顺序的。为了保证根据指定的路由规则进行转发。我们可以使用yml文件进行配置，读取yml文件时是从上至下进行读取的。

忽略表达式

zuul.ignored-patterns:该表达式可以设置某个路由不被api网关进行路由。

zuul.ignored-patterns=/\*\*/hello/\*\*

如果访问http://localhost:8080/test-service/hello/test,那么该url就不会被路由了。

### 路由前缀

zuul提供了zuul.prefix

zuul.prefix=/api   默认情况下，代理前缀不起作用，需要设置 zuul.stripPrefix=false。或者设置zuul.routes.<route>.strip-prefix=true 对指定的路由进行设置。

### cookie问题

默认情况下，使用zuul进行路由的时候，会过滤掉http请求头的敏感信息，其中就包括cookie。防止传递到下游的外部服务器，可以设置zuul.sensitiveHeaders=,将该值设置为空，表示不过滤敏感信息。但是实际情况下有的服务会用到这些信息，而有的不需要，比如

登录功能需要用到cookie。这样我们可是设置

1.对指定路由开启自定义敏感头

zuul.routes.<router>.customSensitiveHeaders=true

2.将指定路由的敏感头设置为空

zuul.routes.<router>.sensitiveHeaders=

### 重定向

zuul.addHostHeader=true

### Hystrix和Ribbon

hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds=60000

该参数可以用来设置api网关中路由转发请求的HystrixCommand执行超时时间，单位毫秒，

ribbon.ConnectTimeout=60000，如果该值小于hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds，会重新连接，知道到达hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds的时间。否则直接抛出异常。

ribbon.ReadTimeout=10000

bootstrap.yml 是被一个父级的 Spring ApplicationContext 加载的。这个父级的 Spring ApplicationContext是先加载的，在加载application.yml 的 ApplicationContext之前。

为何需要把 config server 的信息放在 bootstrap.yml 里？

当使用 Spring Cloud 的时候，配置信息一般是从 config server 加载的，为了取得配置信息（比如密码等），你需要一些提早的或引导配置。因此，把 config server 信息放在 bootstrap.yml，用来加载真正需要的配置信息。同时，某些配置如果已经加载过，那么后续后相同的配置，也不会加载了。

# 应用名称配置:

spring:

application:

name: microservice-springcloud-config-client

# git地址配置

server:

port: 3344

spring:

application:

name: microservice-springcloud-server-config-server

cloud:

config:

server:

git:

uri: https://github.com/1iqueen/microservice-springcloud-server-config.git

username: 1018661791@qq.com

password: 20111701020207a

spring:

application:

name: microservice-springcloud-config-client

cloud:

config:

name: microservice-springcloud-config-client #需要从github上读取的资源名称

profile: dev #本次访问的配置项

label: master

uri: http://localhost:3344 #服务启动通过3344间接与github获取连接

# 数据源配置

spring:

application:

name: MICROSERVICE-SPRINGCLOUD-DEPT-PROVIDER

datasource:

type: com.alibaba.druid.pool.DruidDataSource # 当前数据源操作类型

driver-class-name: org.gjt.mm.mysql.Driver # mysql驱动包

url: jdbc:mysql://192.168.106.34:3306/cloudDB01 # 数据库名称

username: root

password: root

dbcp2:

min-idle: 5 # 数据库连接池的最小维持连接数

initial-size: 5 # 初始化连接数

max-total: 5 # 最大连接数

max-wait-millis: 200 # 等待连接获取的最大超时时间

# Mybatis配置

mybatis:

config-location: classpath:mybatis/mybatis.cfg.xml # mybatis配置文件所在路径

type-aliases-package: com.lk.springcloud.entities # 所有Entity别名类所在包

mapper-locations:

- classpath:mybatis/mapper/\*\*/\*.xml # mapper映射文件

# 微服务信息

info:

app.name: microservice-springcloud-provider-8001

company.name: www.lk.com

build.artifactId: $project.artifactId$

build.version: $project.version$

# eureka配置

eureka:

instance:

instance-id: microservice-springcloud-dept-provider-8001

prefer-ip-address: true #访问路径可以显示IP地址

client: #客户端注册进eureka服务列表内

service-url:

defaultZone: http://eureka7001.com:7001/eureka/,http://eureka7002.com:7002/eureka/,http://eureka7003.com:7003/eureka/,

# defaultZone: http://localhost:7001/eureka/

eureka:

instance:

# hostname: localhost #eureka服务端的实例名称

hostname: eureka7001.com

client:

register-with-eureka: false #false表示不向注册中心注册自己。

fetch-registry: false #false表示自己端就是注册中心，我的职责就是维护服务实例，并不需要去检索服务

service-url:

defaultZone: http://eureka7002.com:7002/eureka/,http://eureka7003.com:7003/eureka/

#defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/

#设置与Eureka Server交互的地址查询服务和注册服务都需要依赖这个地址（单机）。

# zuul配置

zuul:

prefix: /zs/

ignored-services: "\*" # 使用"\*"注解表示所有程序原始访问服务路径忽略 microservicecloud-dept

routes:

mydept.serviceId: MICROSERVICE-SPRINGCLOUD-DEPT-PROVIDER

mydept.path: /mydept/\*\*

# 配置激活的环境

spring:

application:

name: account-service-provider

mvc:

throw-exception-if-no-handler-found: true

resources:

add-mappings: false

profiles:

active: '@activatedProperties@'

# 配置日志级别

logging:

level:

cn:

skytech:

skycloud:

provider:

account: DEBUG

logging:

level:

cn:

skytech:

skycloud:

receive:

account:

UserFeignClient: debug

# 是否允许debug

debug: true

# Ribbon设置

ribbon:

ReadTimeout: 600000

ConnectTimeout: 600000

# Fegin设置

feign:

httpclient:

enabled: true

okhttp:

enabled: false