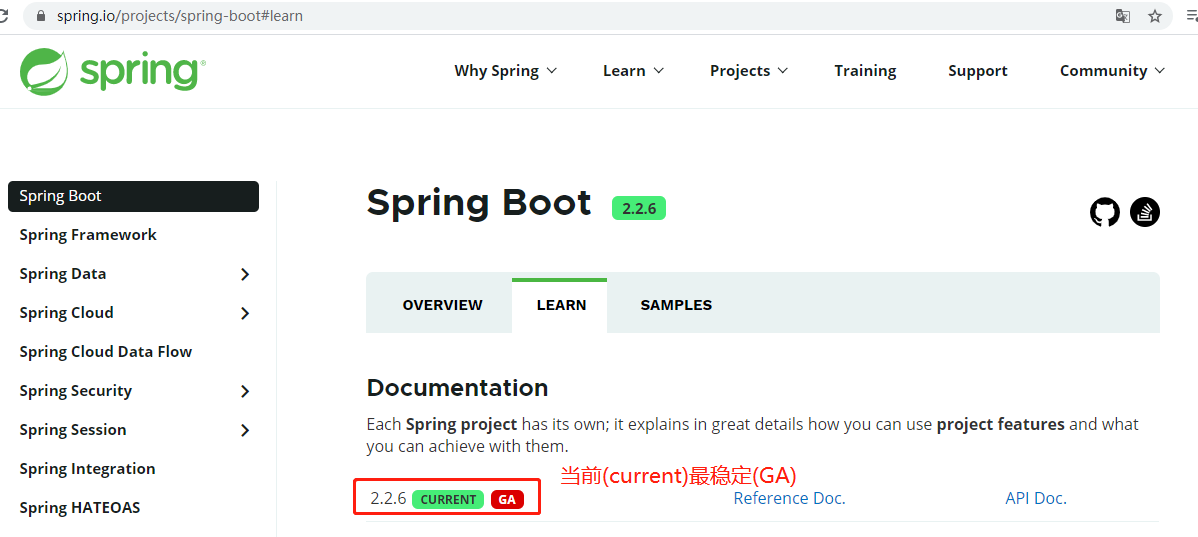
# 版本选择

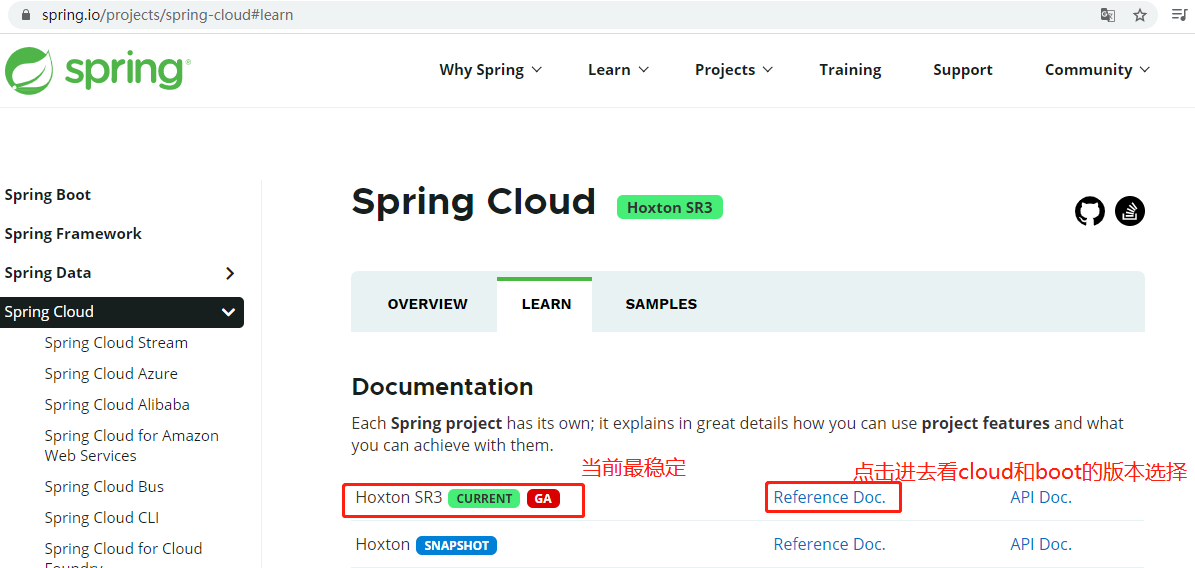
## springBoot

<https://spring.io/projects/spring-boot>



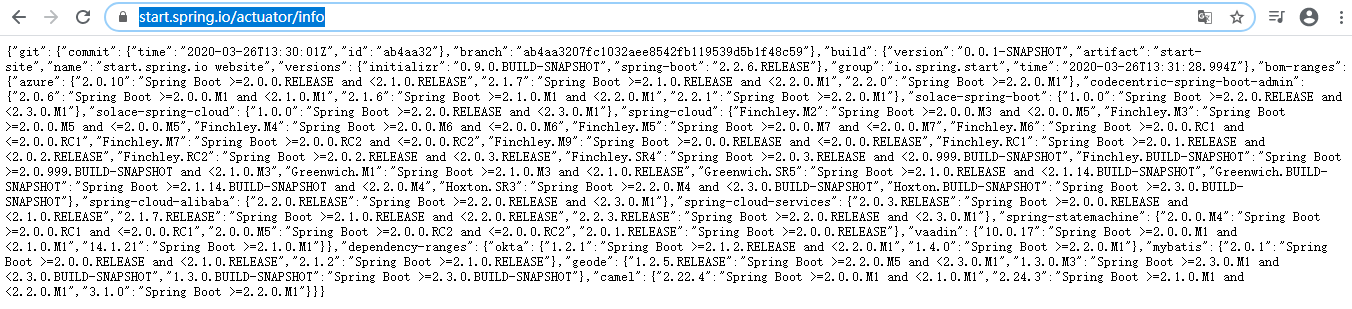
## SpringCloud

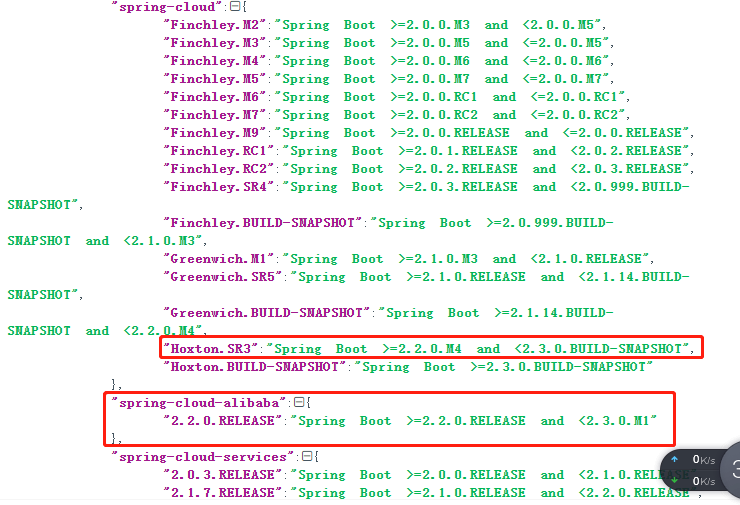
<https://spring.io/projects/spring-cloud>



## 1.3 springBoot和springCloud版本对应查看

<https://start.spring.io/actuator/info>

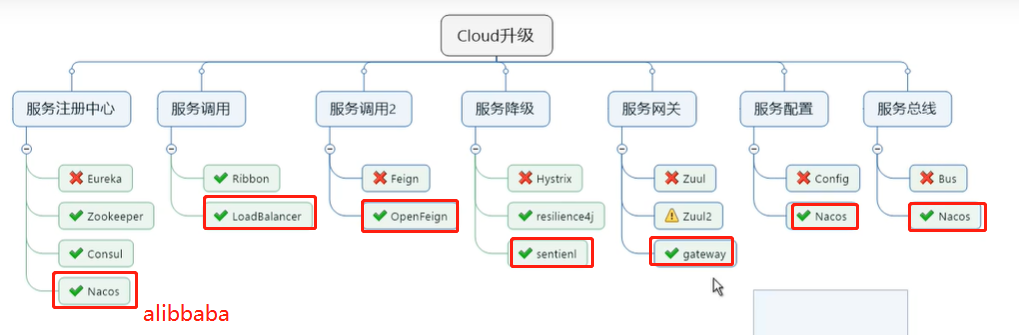




## 1.4 本次版本选择

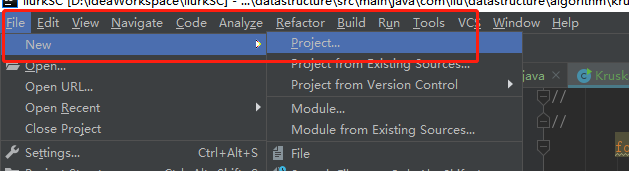
|  |  |
| --- | --- |
| springCloud | Hoxton.SR1 |
| springBoot | 2.2.2.RELEASE |
| Cloud alibaba | 2.1.0.RELEASE |
| Jdk/java | 1.8/java8 |
| Maven | 3.5及以上(3.6.0) |
| Mysql | 5.7及以上(5.8) |

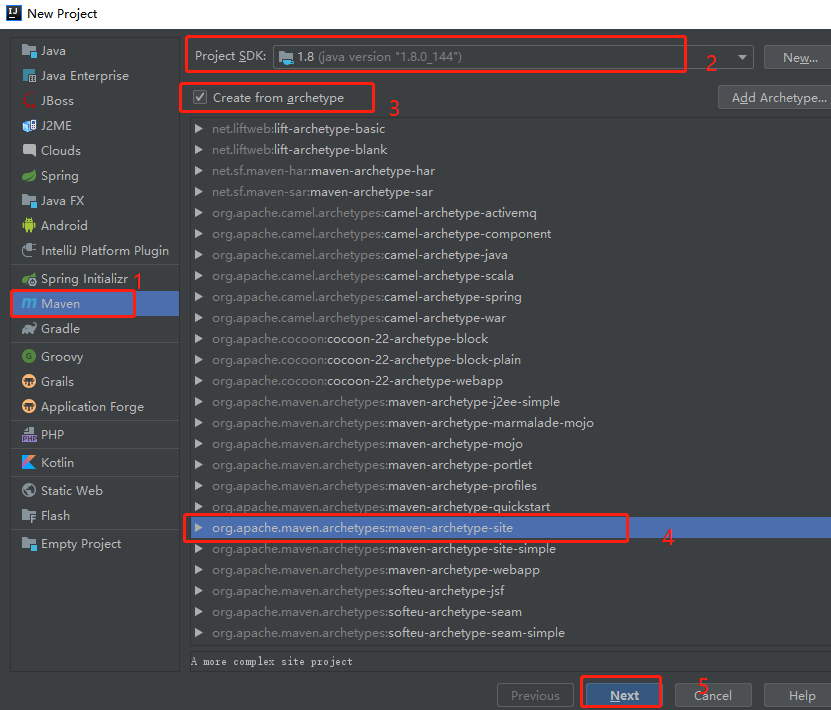
# springCloud各组件技术选型

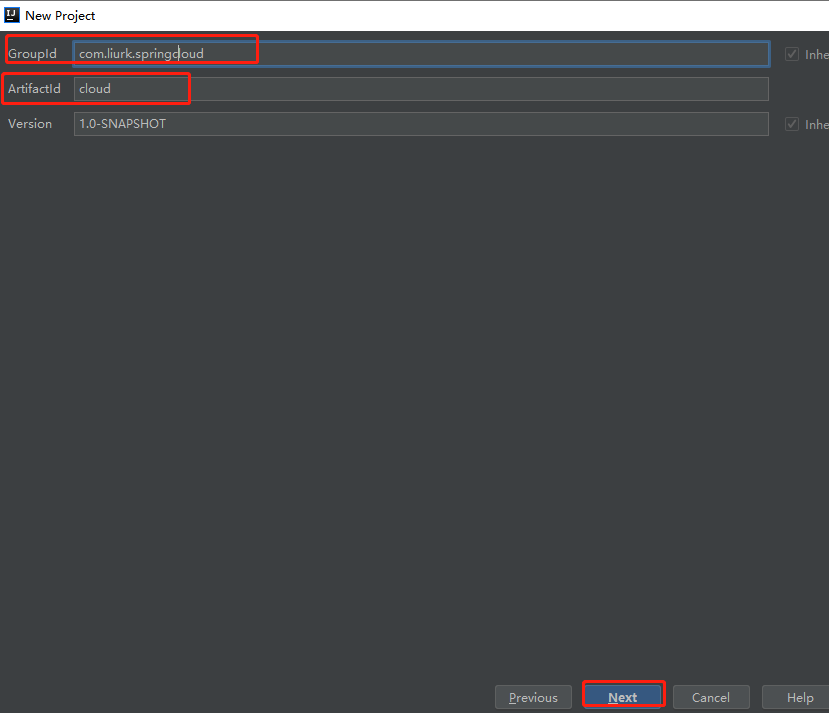


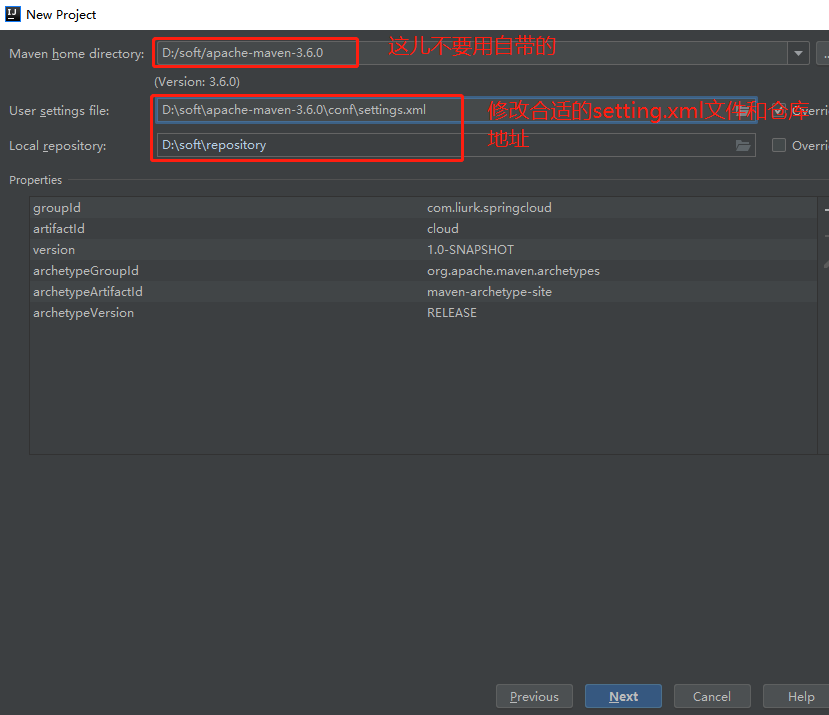
# 创建父工程

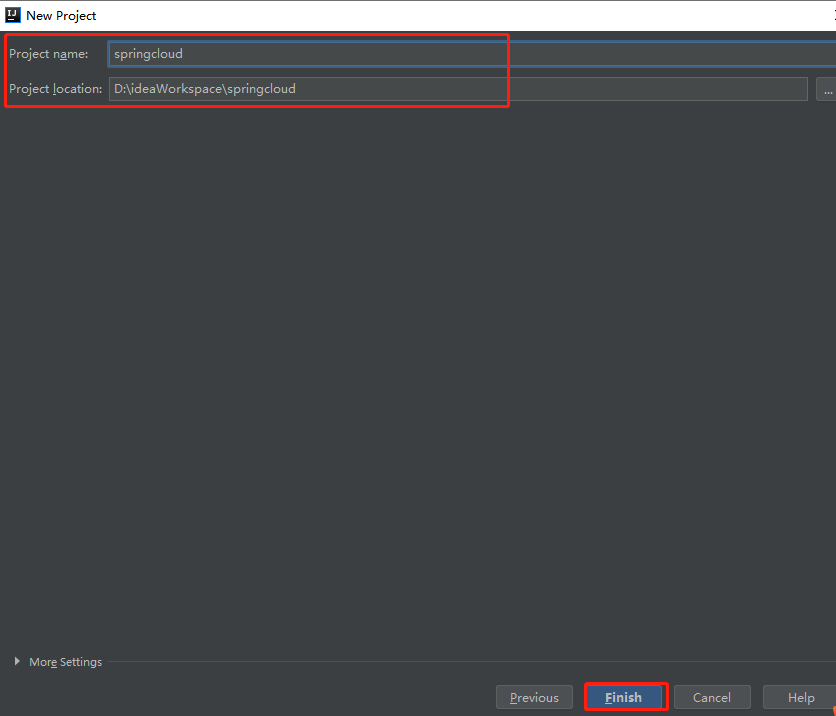
## 3.1 创建springcloud父工程

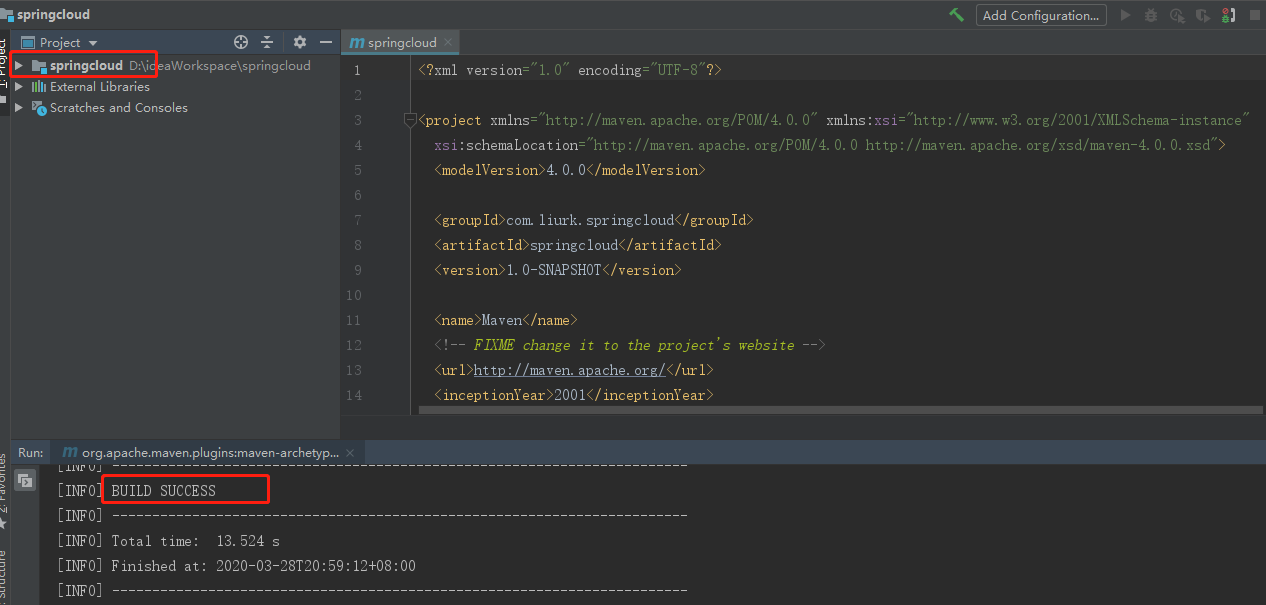




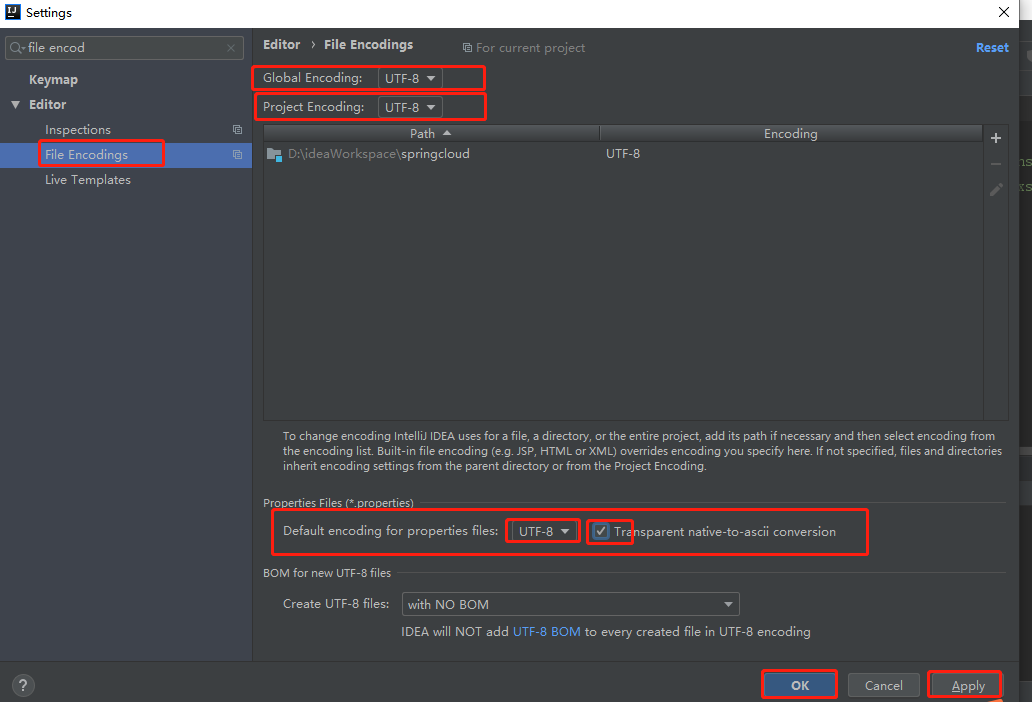


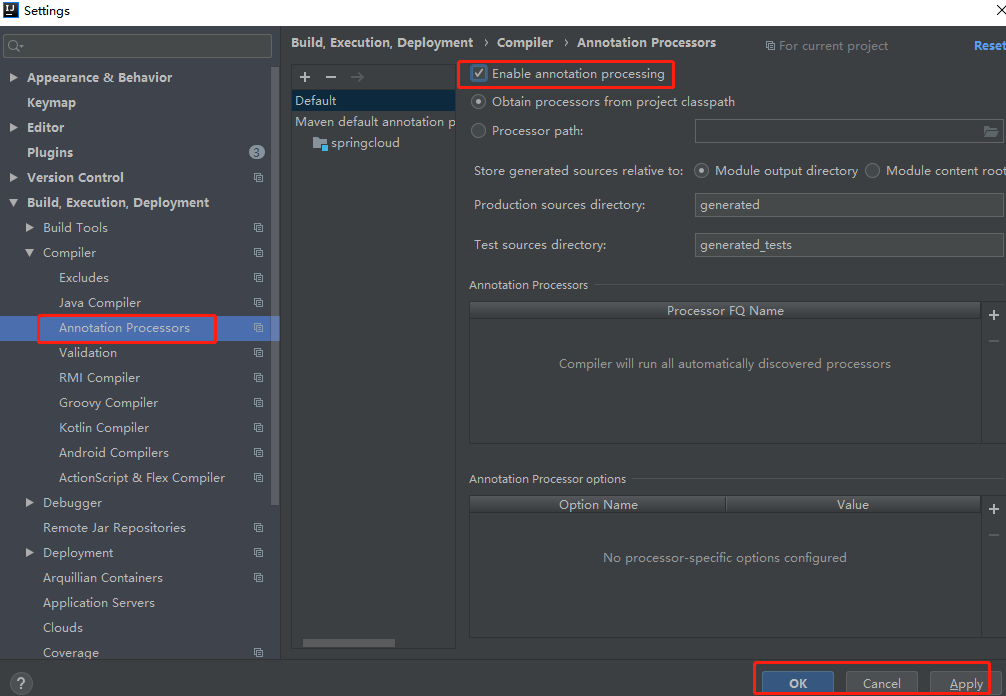


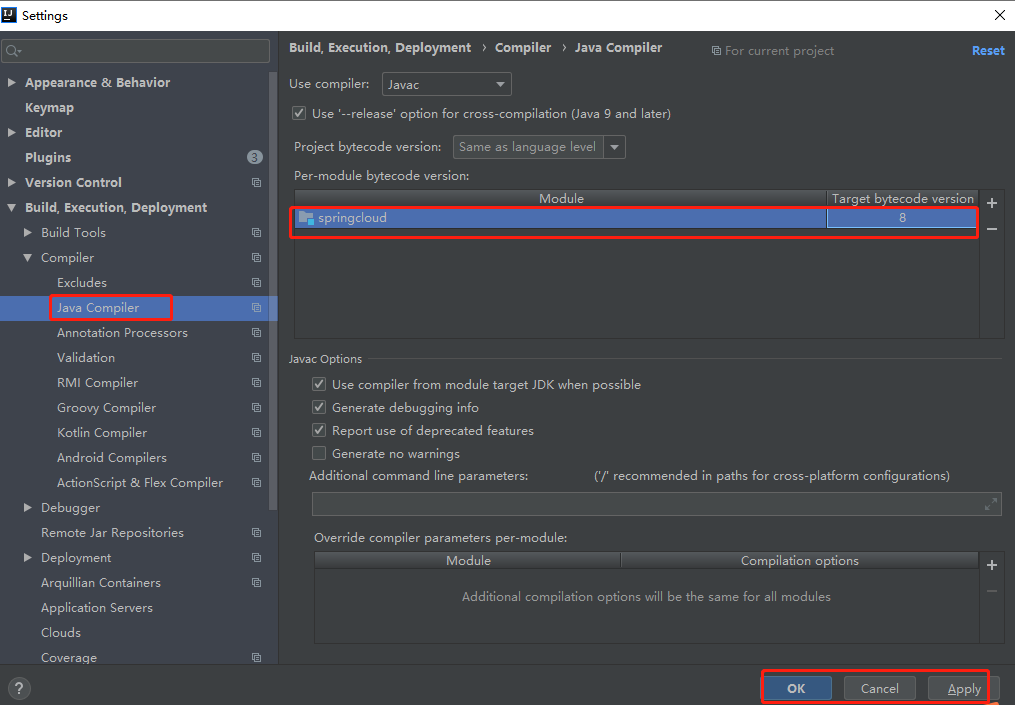




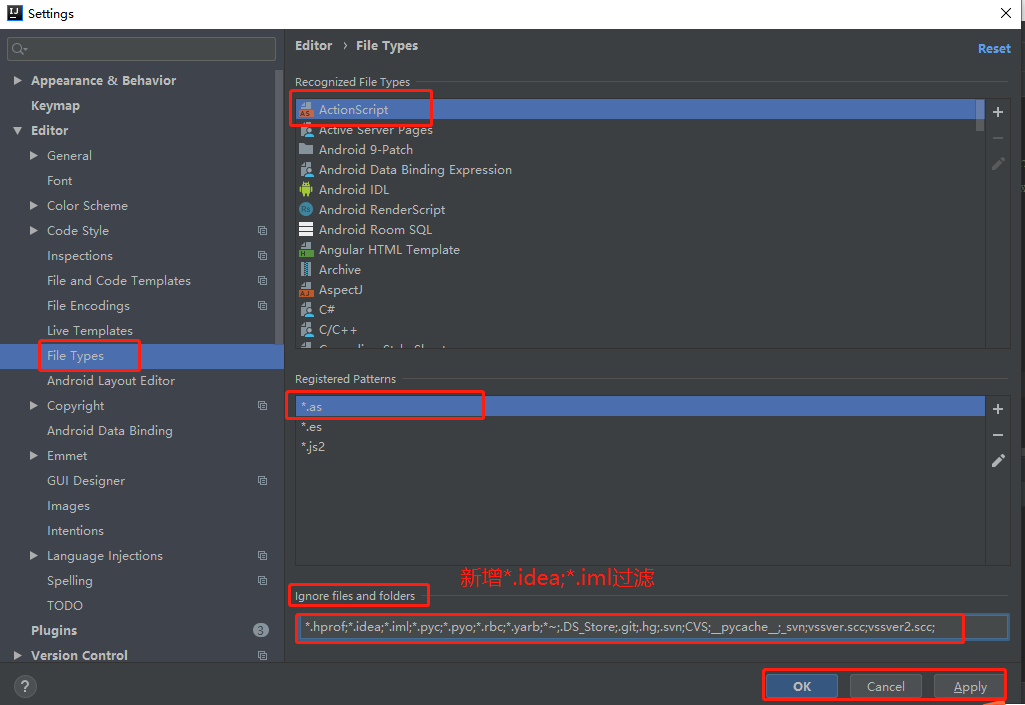
## 3.2 字符编码、注解激活生效、java编译版本





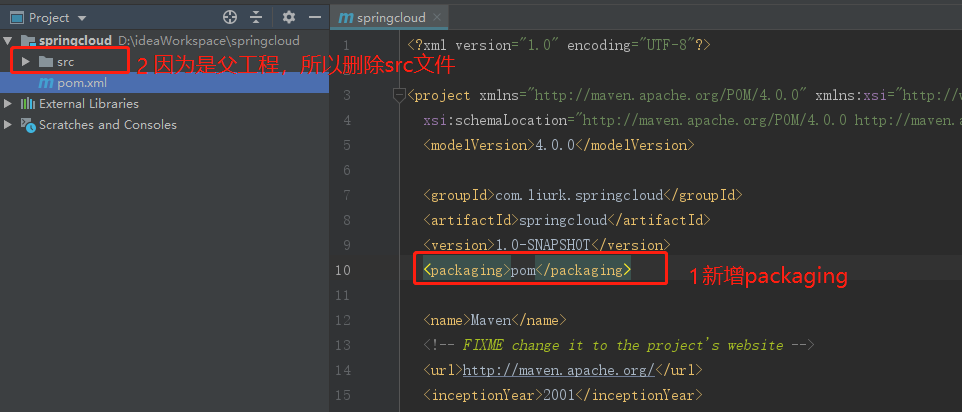


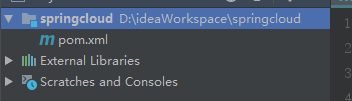
## 3.3 File Types过滤(\*.idea;\*.iml;)



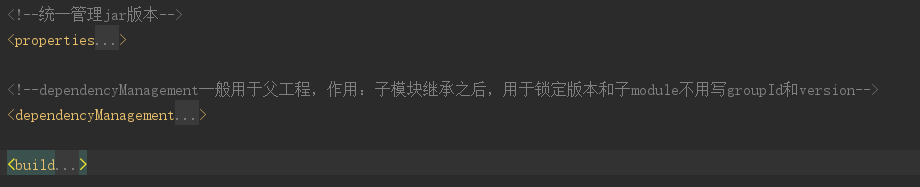
## 3.4 父工程pom.xml

### 3.4.1 修改pom.xml和删除src文件





### 3.4.2 pom.xml添加依赖



### 3.4.3 dependencyManagement和dependencies的区别

（1）dependencyManagement

maven使用dependencyManagement元素来提供了一种管理依赖版本号的方式

一般会出现在项目的父工程

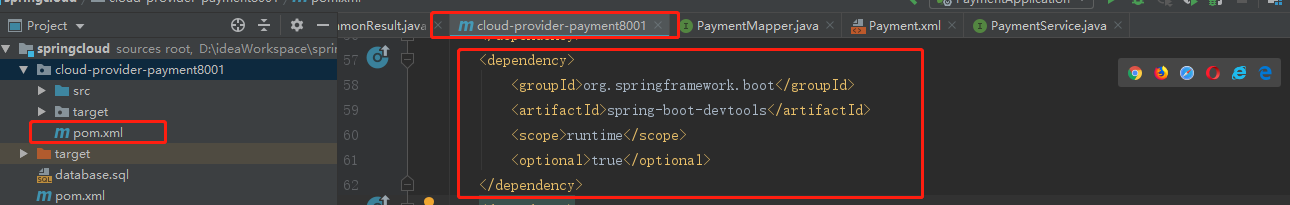
使用pom.xml中的dependencyManagement元素能让所有子项目中引用一个依赖而不显式的列出版本号，maven会沿着父子层次向上走，直到找到一个拥有dependencyManagement元素的项目，然后就会使用dependencyManagement元素中指定的版本号

这样做的好处是：如果有多个子项目都引用同一依赖，则可以避免在每个使用的子项目里都声明一个版本号，这样想升级或者切换另一个版本时，只需要在顶层父容器里更新，而不需要一个一个修改子项目。当某个子项目需要另外的一个版本时，只需要声明version即可，如果子项目中制定了版本号，那么会使用子项目中指定的jar版本

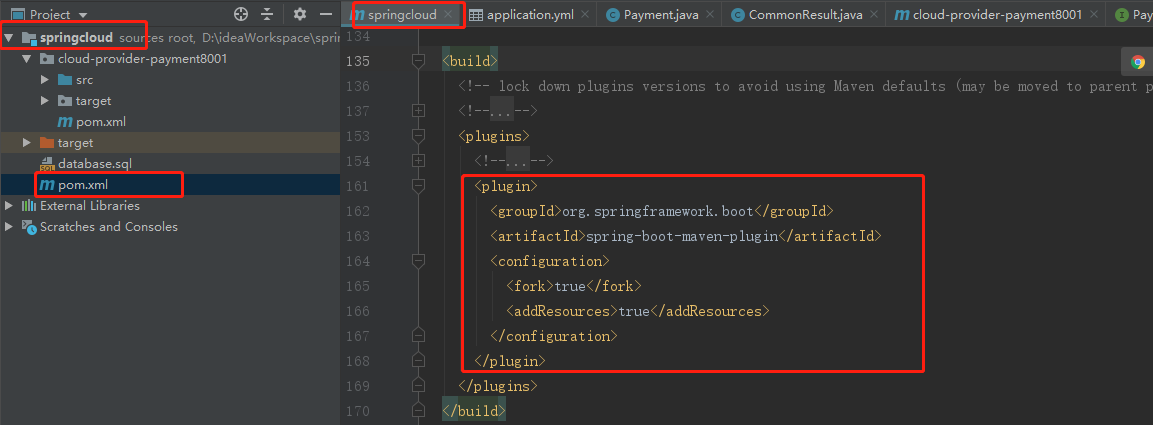
注意：dependencyManagement里只是声明依赖，并不实现引入，因此子项目需要显示的声明需要用的依赖

# 热部署Devtools

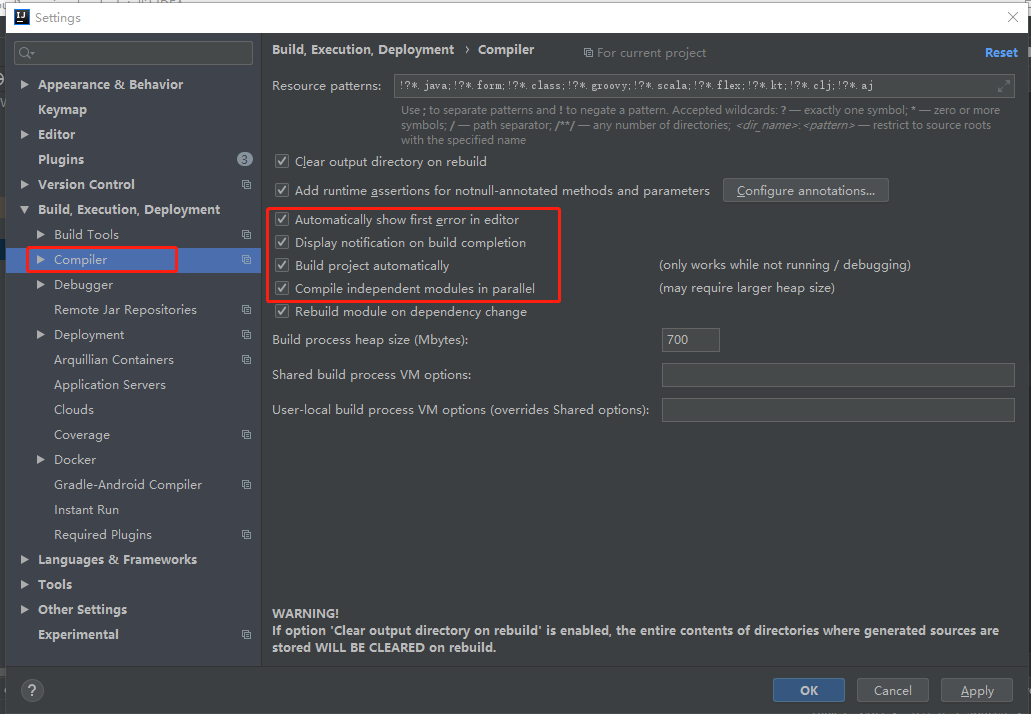
## 在子module添加依赖



## 在父工程中添加maven插件

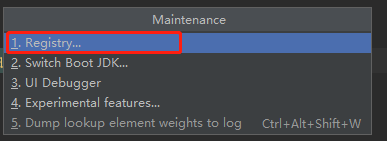


## 开启自动编译

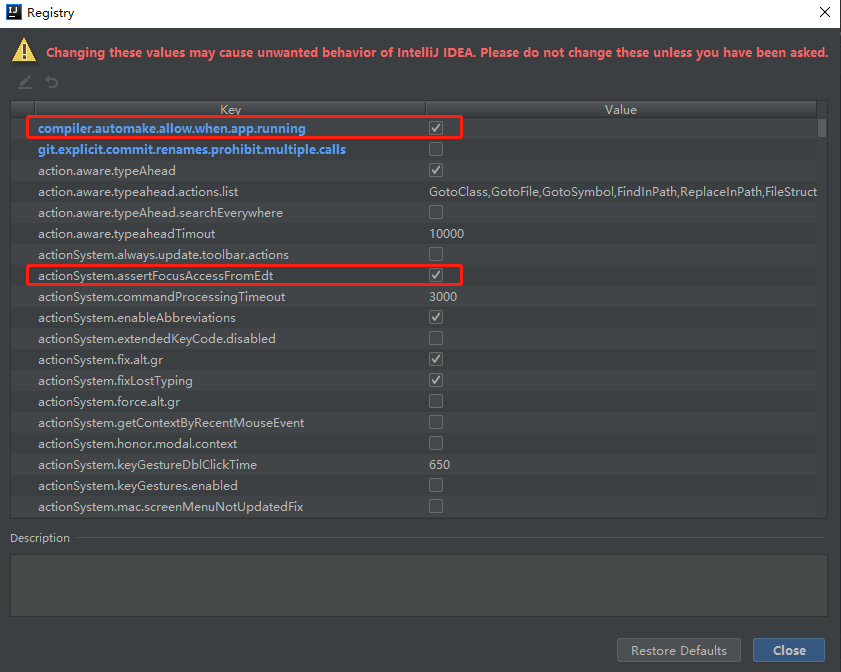


## Update the value of

快捷键：ctrl+alt+shift+/ 会有个弹框，点击Registry

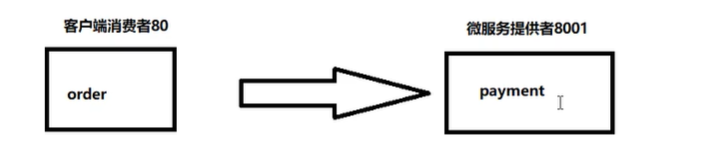


勾选如下两个



## 重启idea

# Restful微服务工程构建

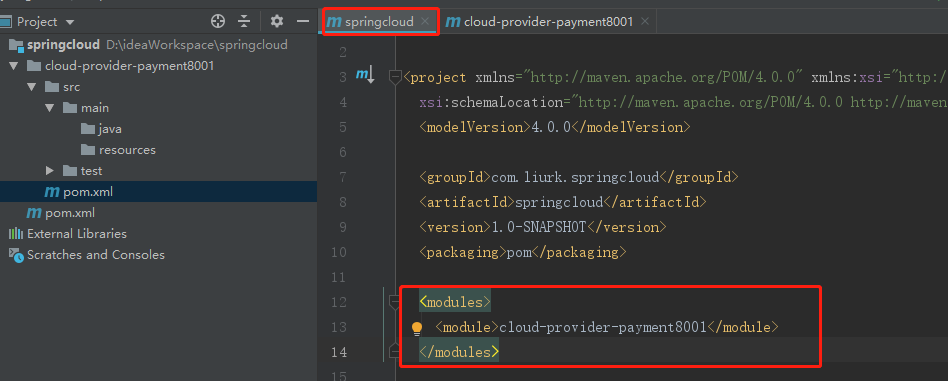


## 新建支付模块payment

### 创建module



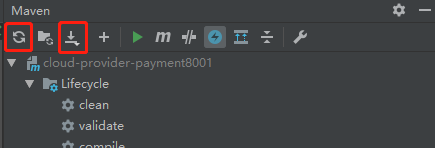
创建好后再查看父工程的pom.xml，多了modules模块



### 修改pom文件

### 编写application.yml

如果driver-class-name: com.mysql.jdbc.Driver报错的话，可能是jar包没有下载下来，需要手动下载和刷新一下依赖



### 启动类

### 业务类

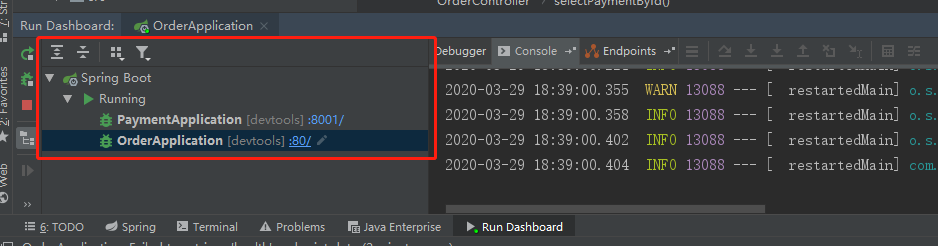
## 消费者订单模块

cloud-cousumer-order80

# RunDashboard

当微服务多了之后，为了方便查看微服务，需要使用RunDashboard

当启动多个微服务后RunDashboard会自动出现

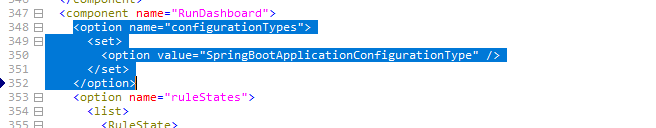


如果没有自动出现的话，修改workspace.xml文件(.idea下面)



添加如下配置

|  |
| --- |
| <option name="configurationTypes">  <set>  <option value="SpringBootApplicationConfigurationType" />  </set>  </option> |



添加后重启idea，就会出现RunDashboard，方便微服务的启动管理

可以右键debug启动对应的微服务

# 工程重构-提取公共内容到common模块

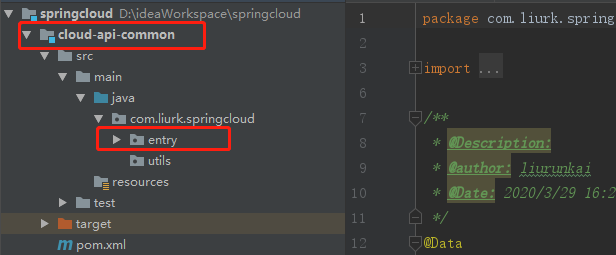
## 新建cloud-api-common模块

用于存放公共的东西：实体类、工具类等

## 修改pom文件

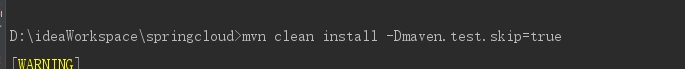
## cloud-api-common模块添加公共内容

将支付模块cloud-provider-payment8001和订单模块cloud-cousumer-order80的entry拷贝到cloud-api-common模块



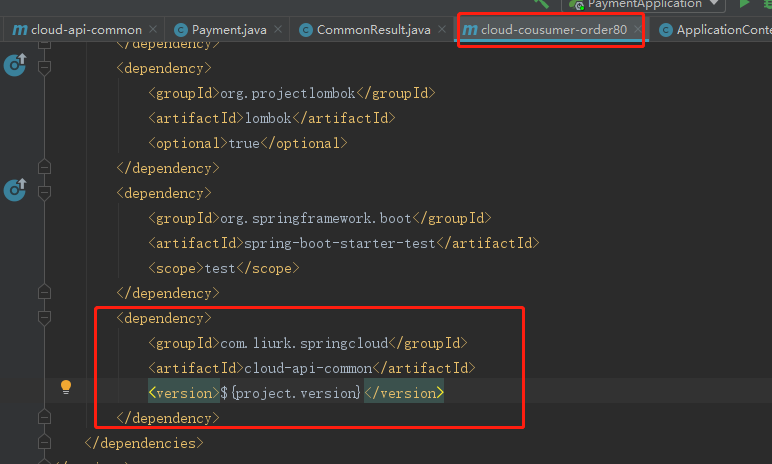
## 将cloud-api-common模块clean install

mvn clean install -Dmaven.test.skip=true



## 支付模块和订单模块改造

1. 将支付模块cloud-provider-payment8001和订单模块cloud-cousumer-order80的entry删除
2. 在对应的pom.xml文件中添加cloud-api-common依赖

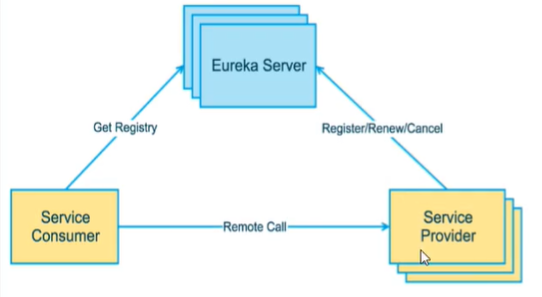


## 再次使用postman测试

测试支付模块cloud-provider-payment8001

和订单模块cloud-cousumer-order80

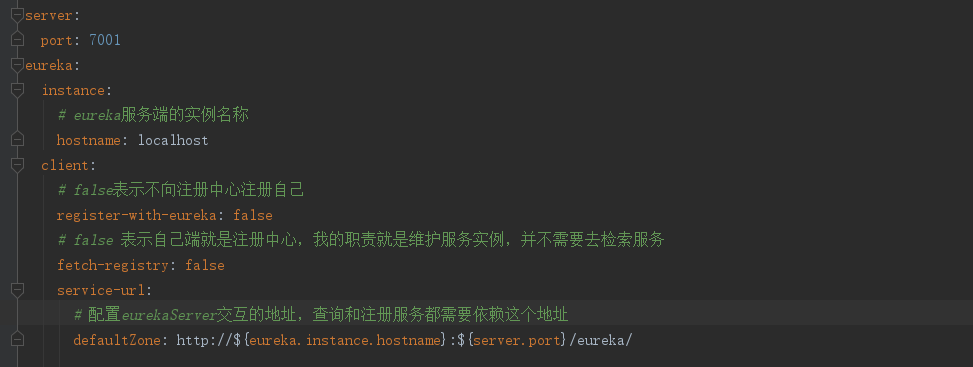
# 服务注册-Eureka



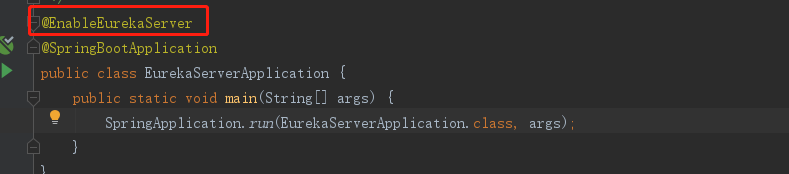
## 搭建eurekaServer单机版

### 新建module，导入pom依赖

### 配置application.yam文件

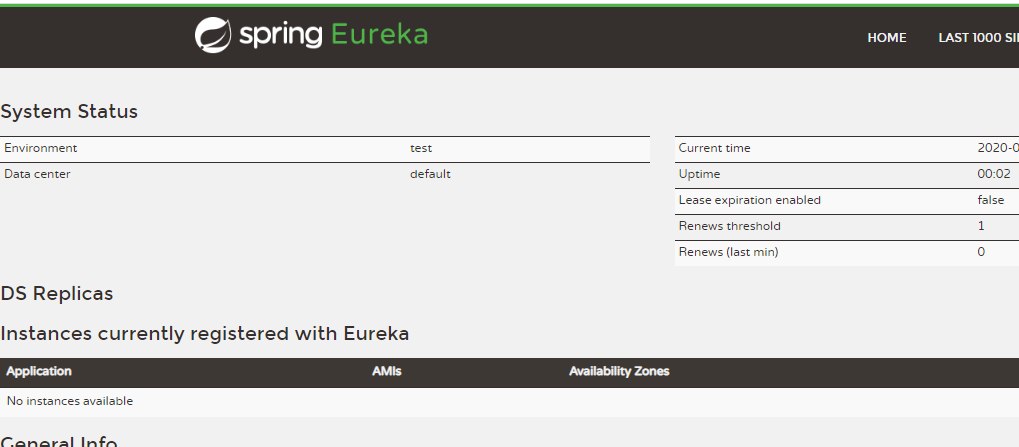


### 启动类



### 访问

<http://localhost:7001>



## 改造生产者cloud-provider-payment8001模块

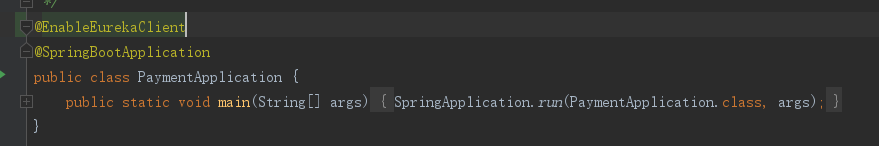
### 修改pom文件



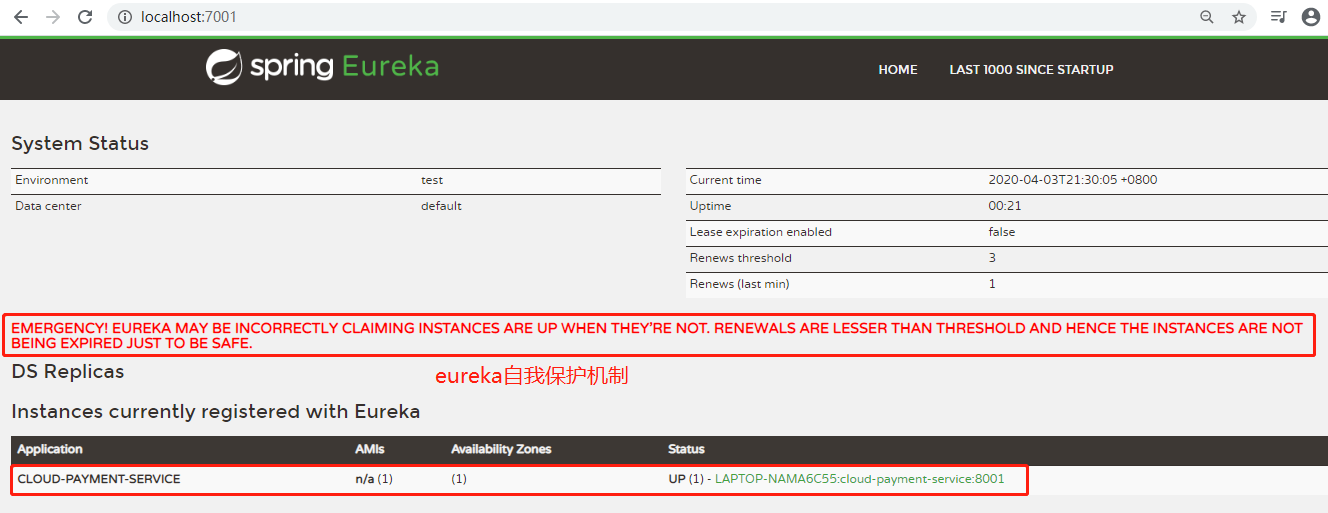
### 修改application.yml文件



### 修改启动类添加注解

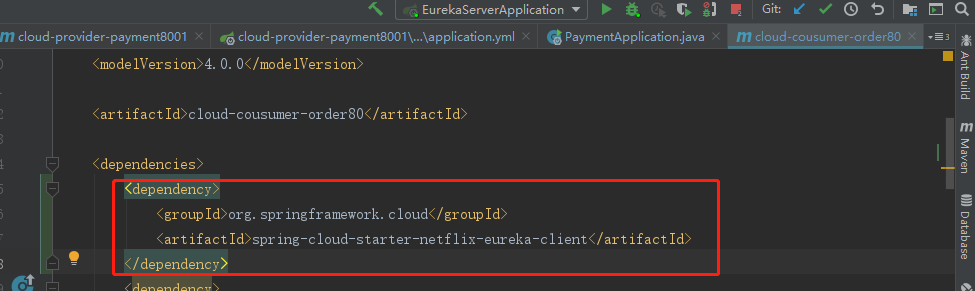


### 启动payment模块，访问eureka服务



### 将cloud-cousumer-order80模块注册到eureka

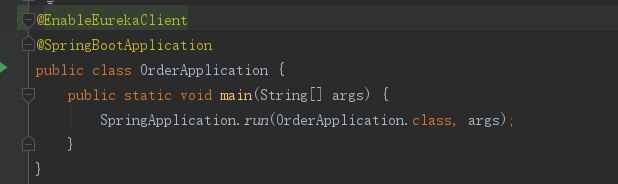
#### 添加pom依赖



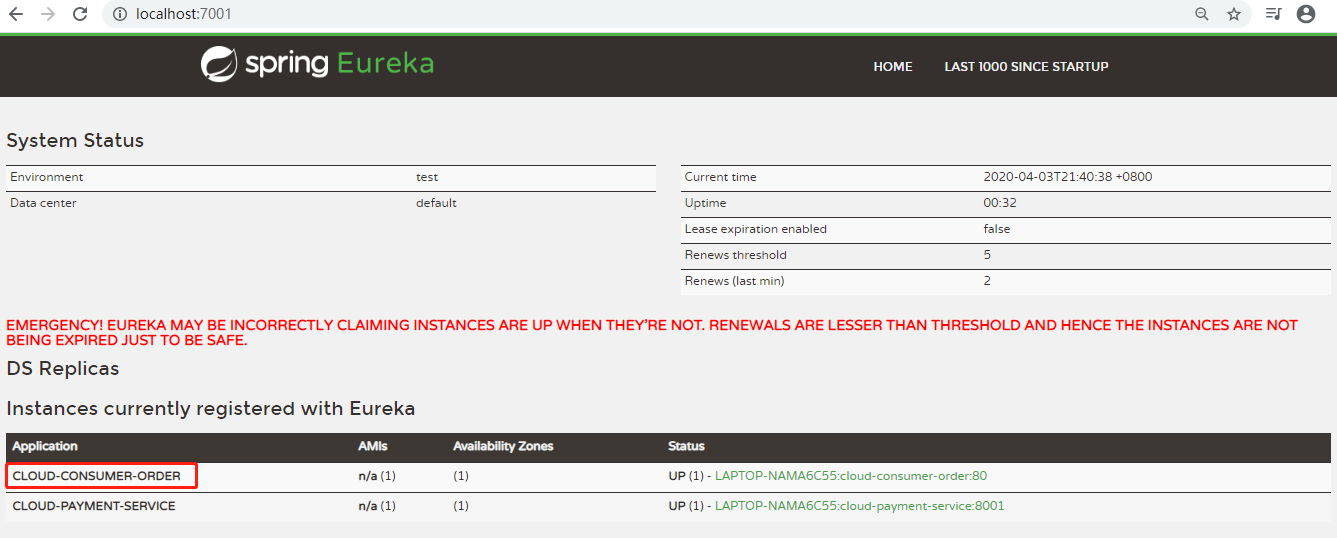
#### 修改application.yml



#### 启动类加注解



#### 访问测试



## 搭建eurekaServer集群版

### eurekaServer集群原理

服务注册：将服务信息注册进服务注册中心

服务发现：从注册中心获取服务信息

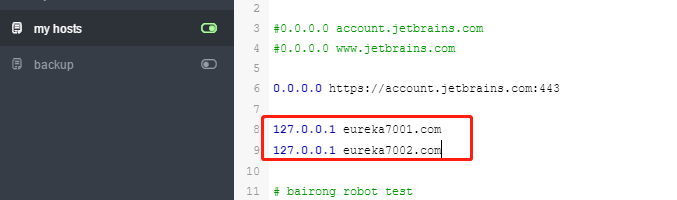
实质：存储key为服务名称，获取value为调用地址

1. 启动服务注册中心，将服务提供者(payment)注册到注册中心
2. 消费者order服务在需要调用接口时，使用服务别名去注册中心获取实际的RPC远程调用地址
3. 消费者获取到远程调用地址后，底层使用httpClient技术实现远程调用
4. 消费者获取到服务地址后会缓存在本地jvm内存中，默认每间隔30s获取一次服务调用地址

### 新建module，导入pom依赖

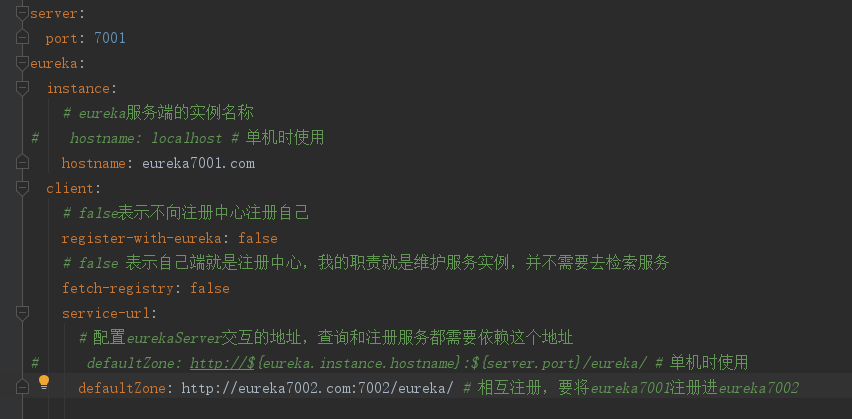
### 修改hosts配置

说明：因为是在自己电脑上测试eureka集群，所以修改hosts文件映射，如果是生产环境，eureka不在同一台服务器，就不需要修改hosts配置



### 修改application.yml

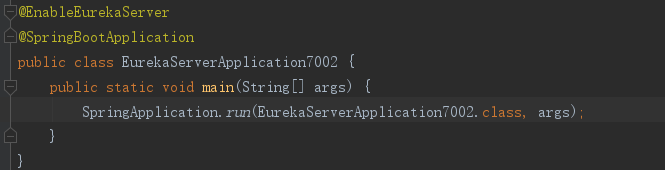
先修改cloud-eureka-server7001的配置文件



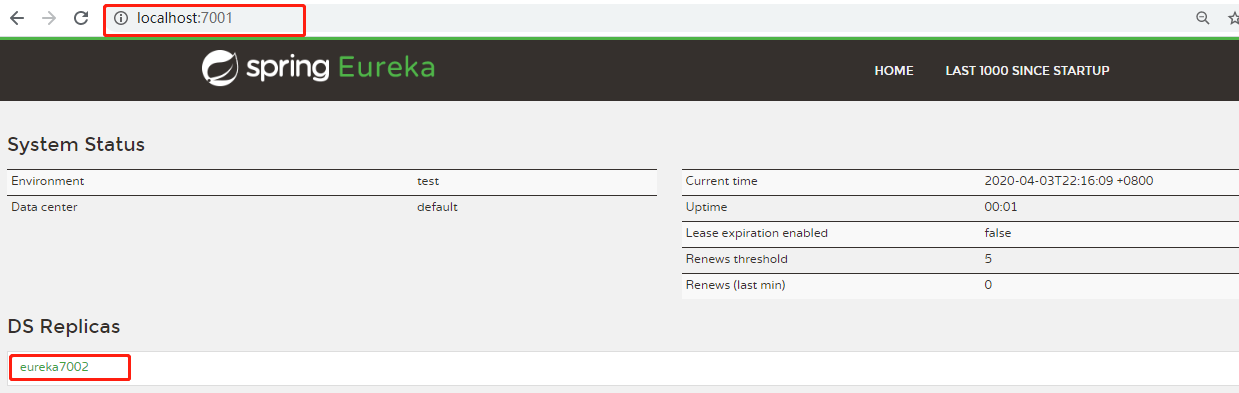
修改cloud-eureka-server7002的配置文件

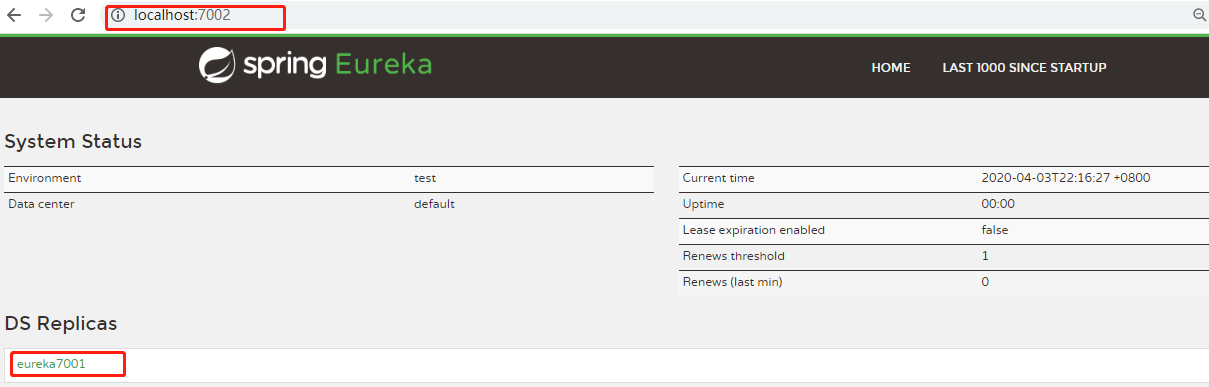


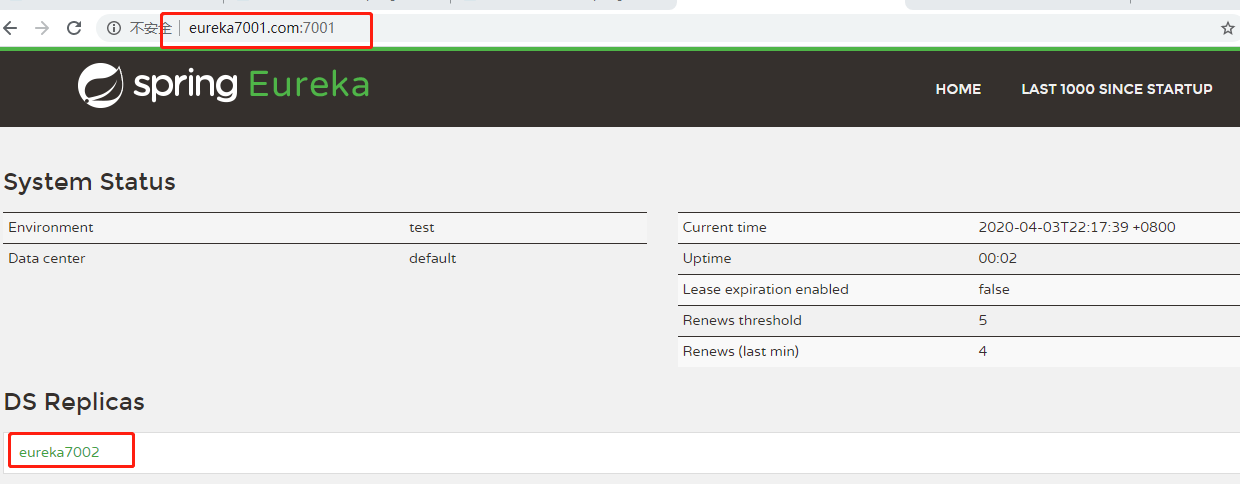
### 启动类

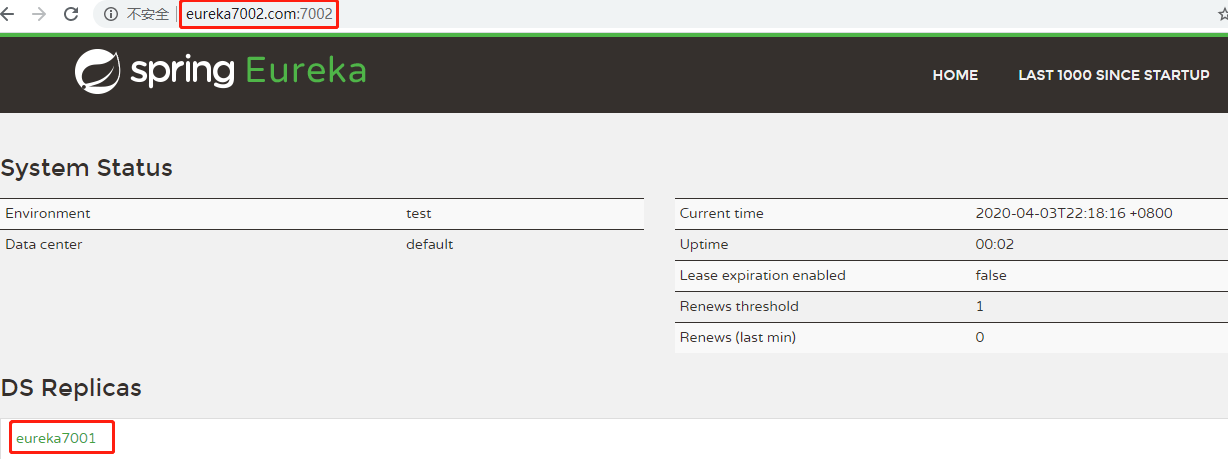


### 启动eureka7001和eureka7002测试









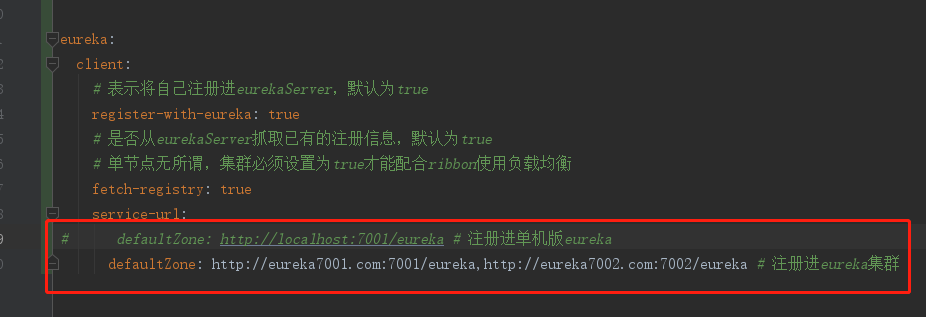
### 将cloud-provider-payment8001注册进eureka集群

修改application.yml

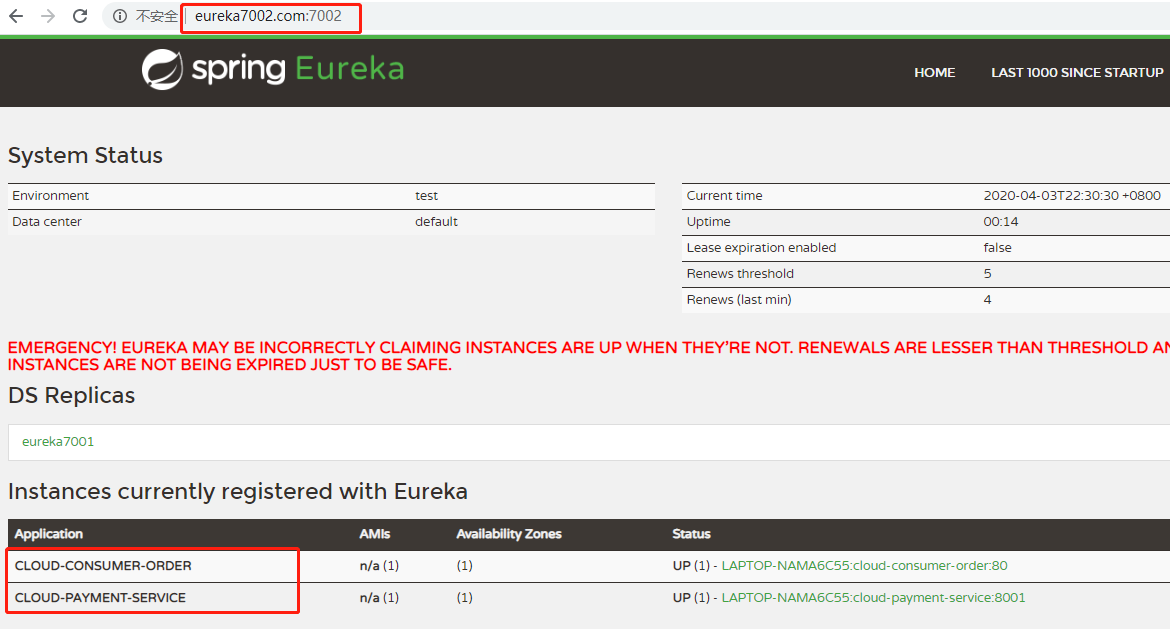


### 将cloud-cousumer-order80注册进eureka集群

修改application.yml



### 启动测试



## 搭建cloud-provider-payment8001集群

### 修建cloud-provider-payment8002 module

### 拷贝8001的pom文件

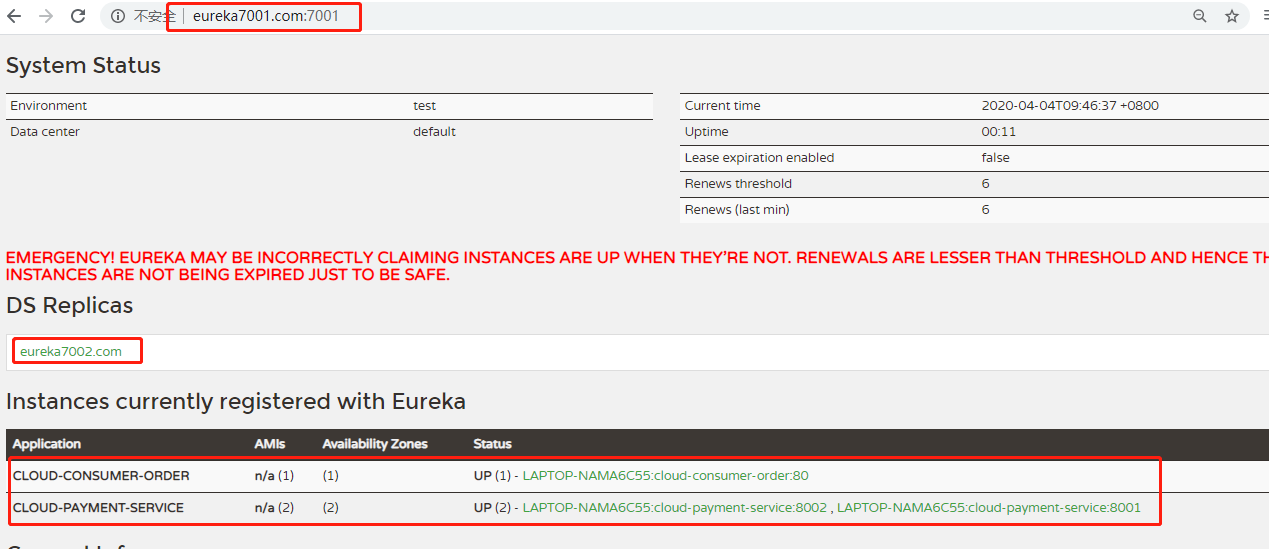
### application.yml配置文件

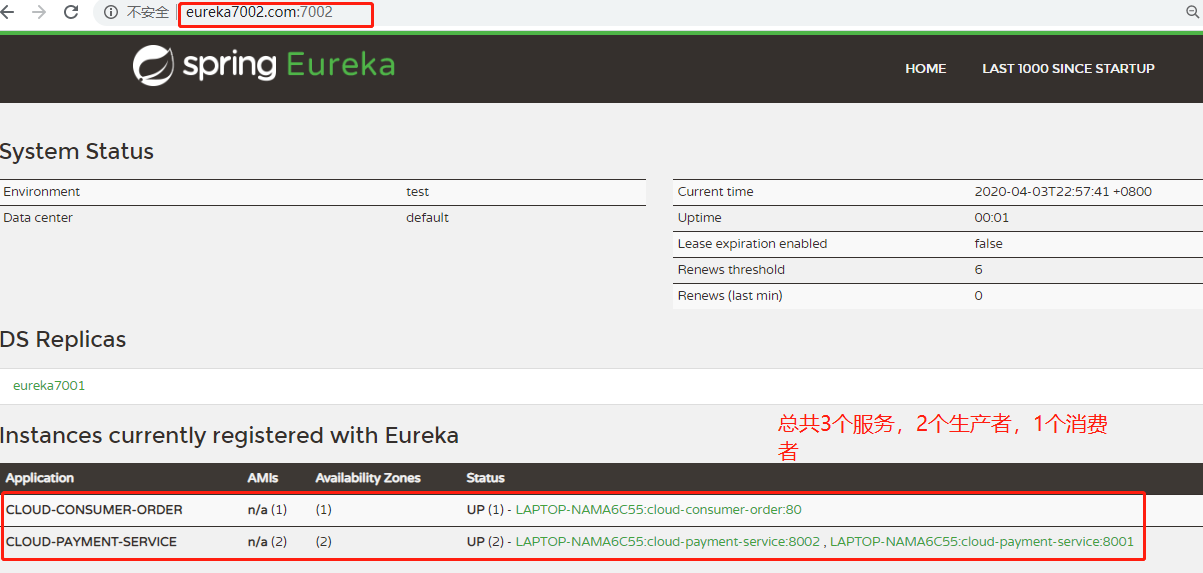
修改端口号为8002

### 主启动类

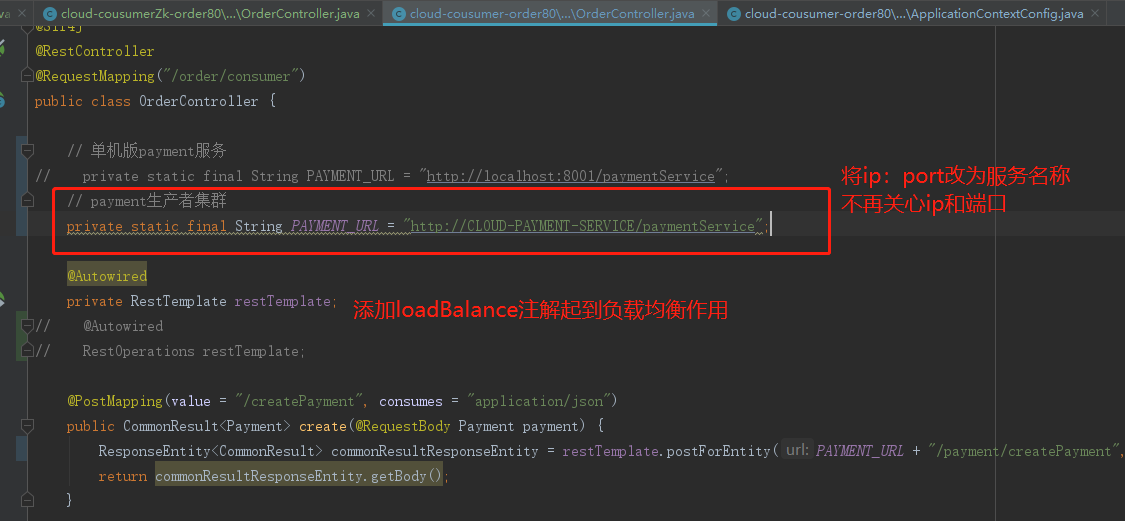
### 拷贝8001的业务代码

### 启动7001、7002、8001、8002测试



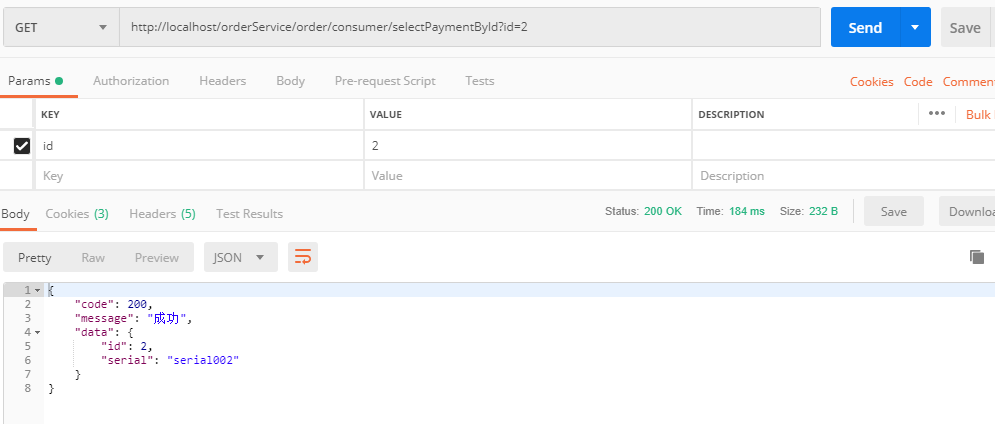


### 修改消费者order的调用方式





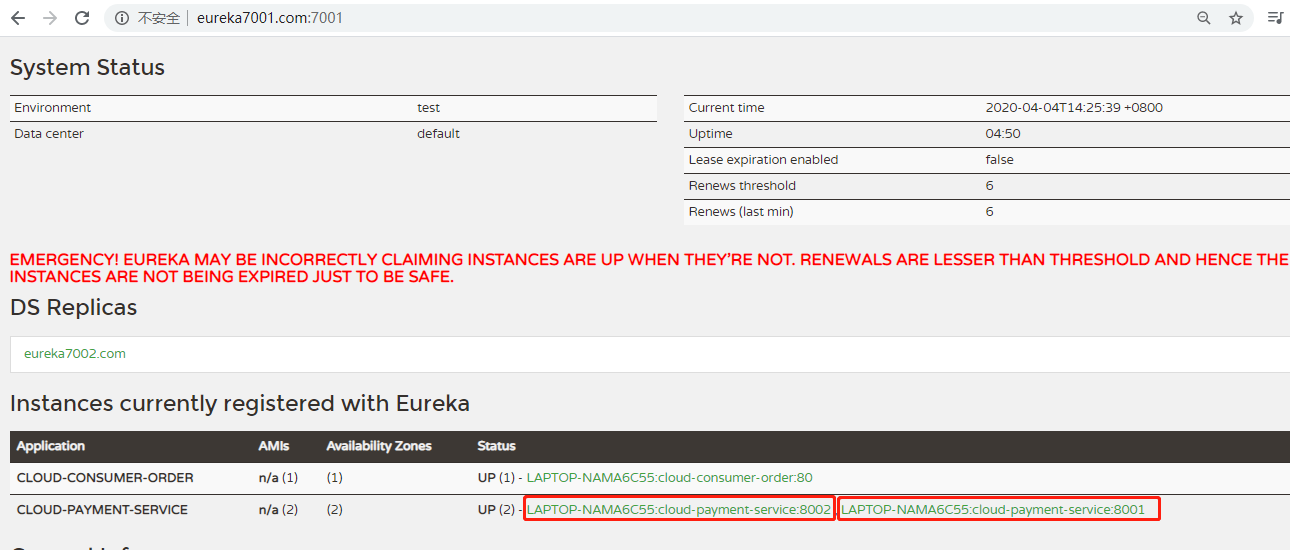
### 启动测试



## actuator信息完善

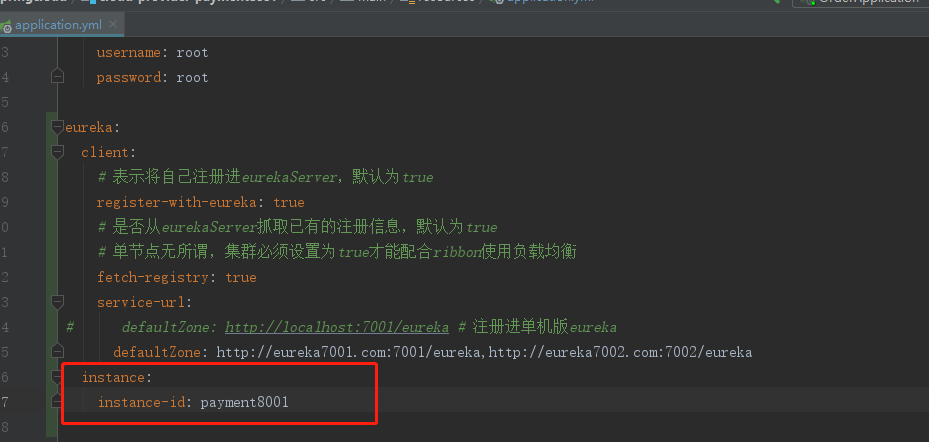
### 主机名称:服务名称修改

修改前

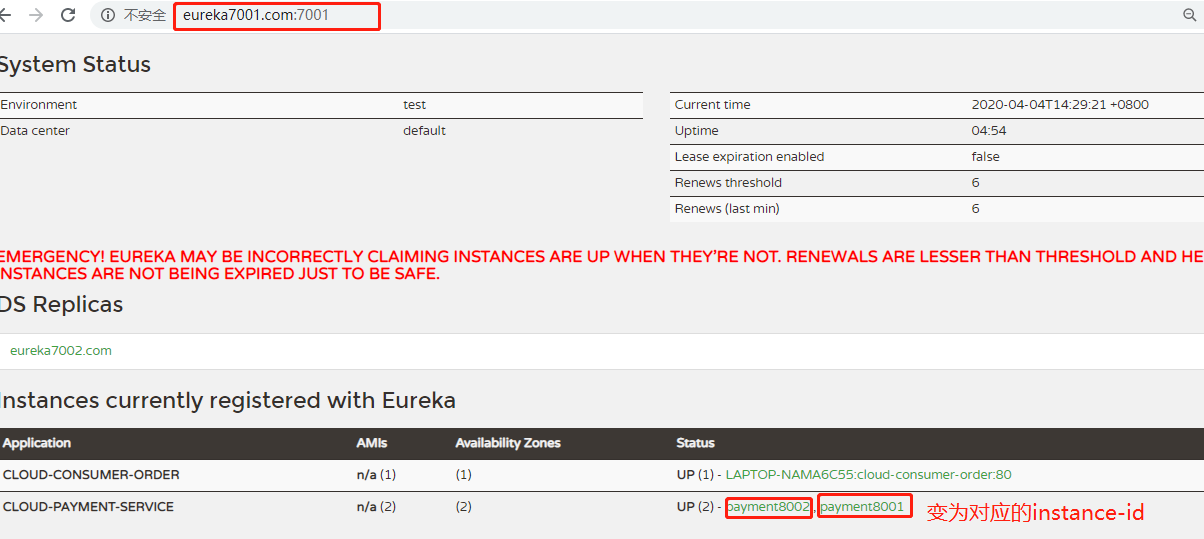


修改payment8001和payment8002的application.yml

添加如下配置：instance

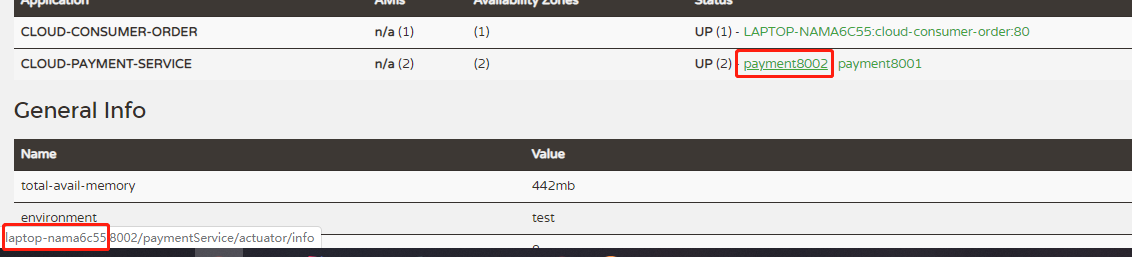


修改后



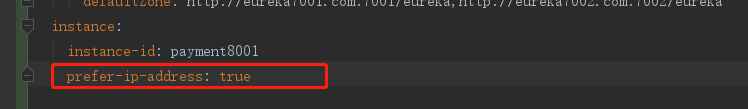
### 访问路径可以显示ip地址

未修改application.yml前



修改application.yml

在instance-id下添加显示ip的配置



修改后



## 服务发现Discovery

对于注册进eureka服务里面的微服务，可以通过服务发现来获得该服务的信息

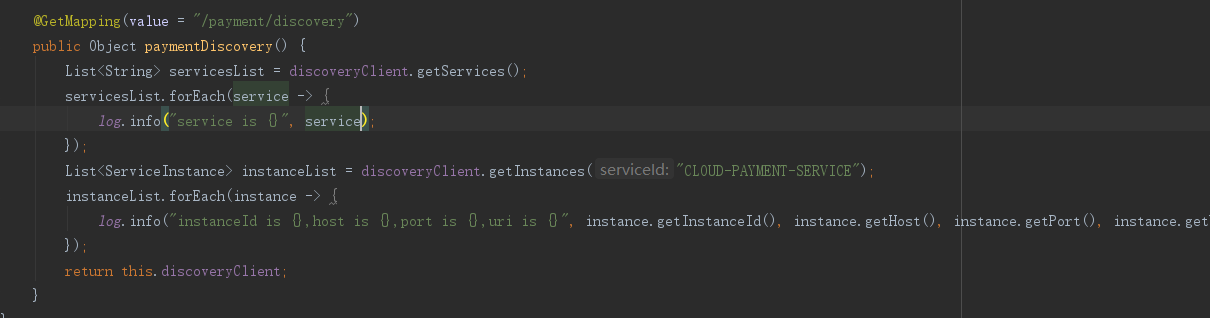
### 修改payment8001和payment8002的controller

注入discoverryClient，注意导包

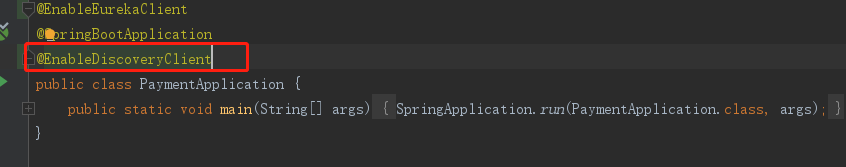




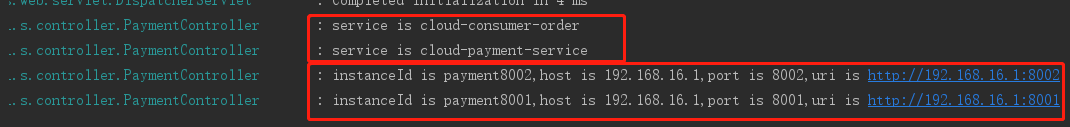
获取服务信息



### 修改启动类添加注解

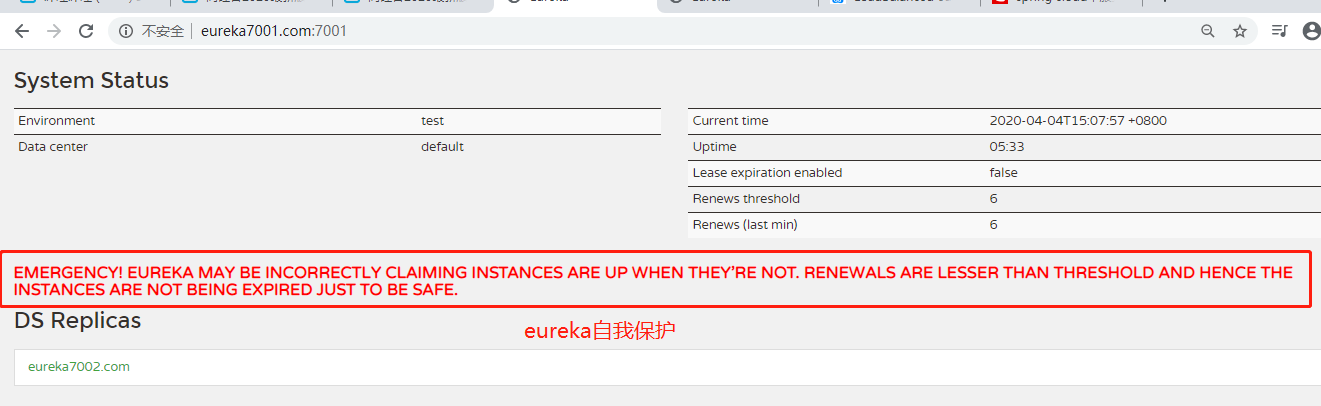


### 启动测试



## Eureka自我保护

### 8.7.1 什么是eureka自我保护



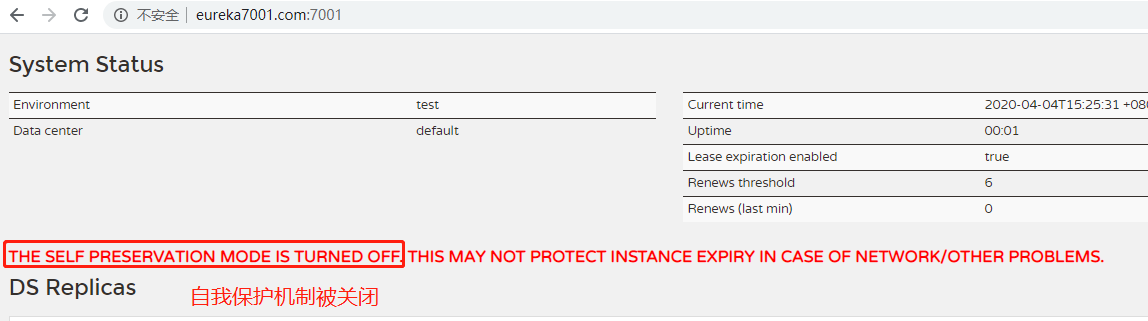
某个微服务在某一时刻不可用，eureka不会立刻清理该微服务，依旧会对该微服务的信息保存，属于CAP里面的AP(P：分区容错性，A：高可用)分支

### 8.7.2 禁止eureka自我保护

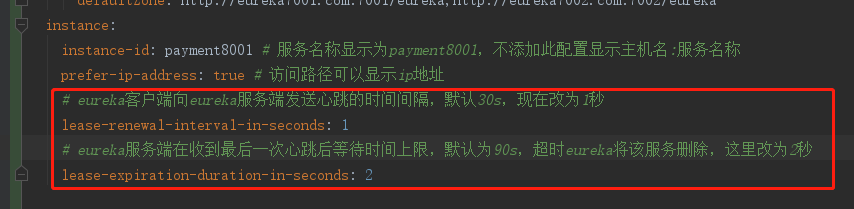
修改application.yml



### 8.7.3 启动eureka服务测试



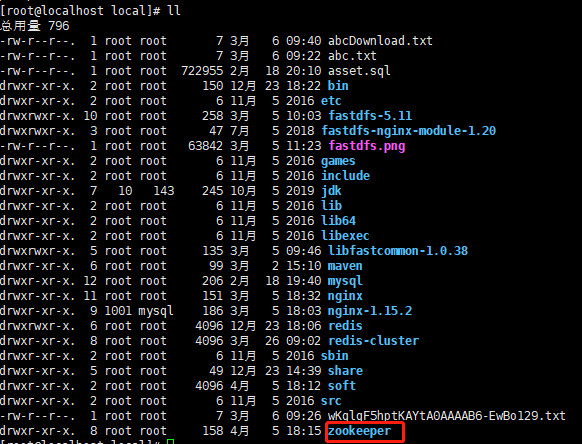
### 8.7.4 修改payment的application.yml



# 服务注册-zookeeper

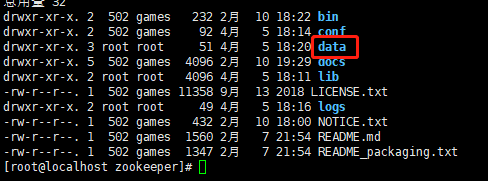
## 安装zookeeper

### 下载并解压到/usr/local

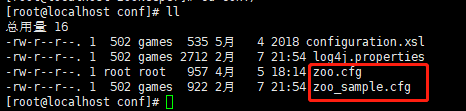


### 修改配置文件

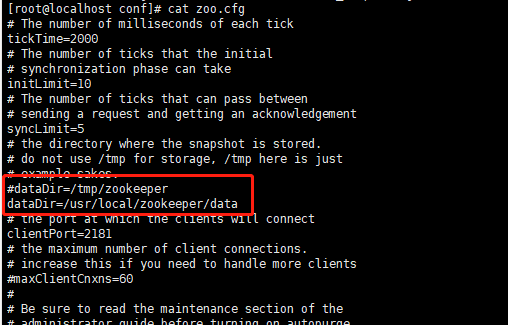
创建data目录 mkdir -p /usr/local/zookeeper/data



进入/usr/local/zookeeper/conf目录，复制zoo\_sample.cfg并重命名为zoo.cfg



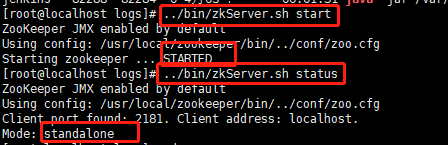
修改zoo.cfg：vim zoo.cfg，将dataDir目录改为刚刚创建的data目录



### 启动zookeeper并查看zookeeper启动状态

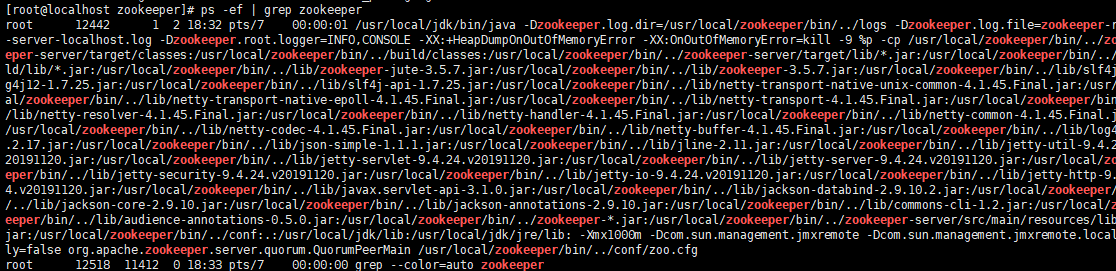
./zkServer.sh start

./zkServer.sh status



### 查看zookeeper进程id

ps -ef | grep zookeeper

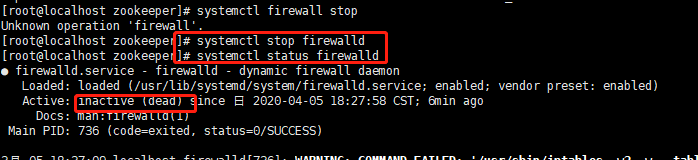


### 查看zookeeper端口

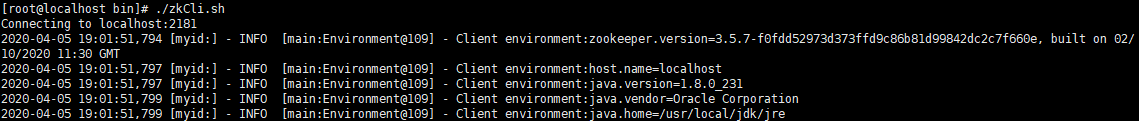
netstat -anp | grep 2181



### 关闭防火墙并查看防火墙状态



### 连接zookeeper



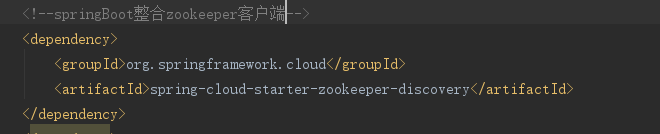
### 查看zk节点，默认只有zookeeper一个节点



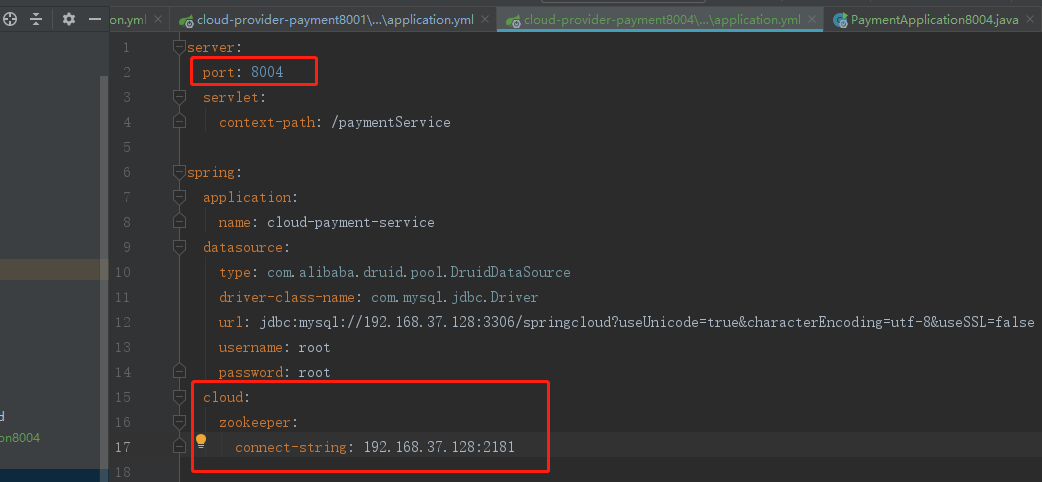
## 将生产者注册进zookeeper

### 创建cloud-provider-payment8004模块

### 修改pom文件，添加zookeeper依赖



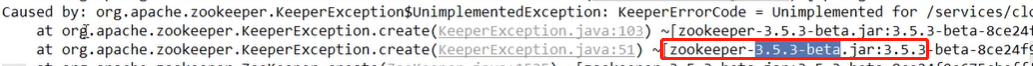
### 修改application.yml



### 启动类



说明：我的zookeeper版本是3.5.7，如果zookeeper版本低于3.5.3的话，如3.4.9在项目启动的时候可能会报如下错误，jar包冲突：

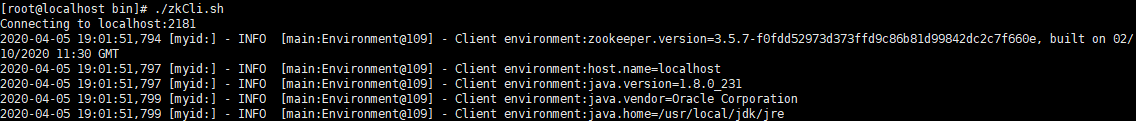


这时需要修改pom依赖，需要先排除3.5.3的依赖，然后在指定zookeeper版本为3.4.9



### 启动项目，连接zk客户端，查看服务是否注册成功

./zkCli.sh



如果注册成功，使用ls /可以看到服务(services)，默认只有zookeeper

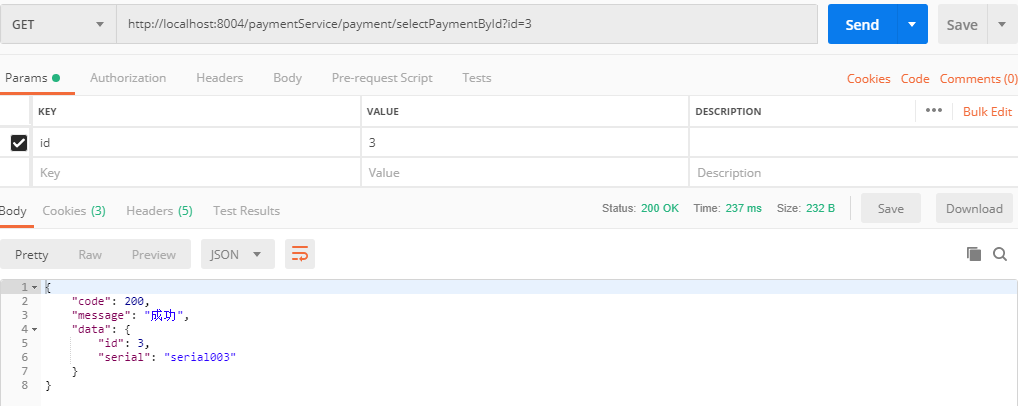


查看服务ls /services ，可以看到我们的服务名称

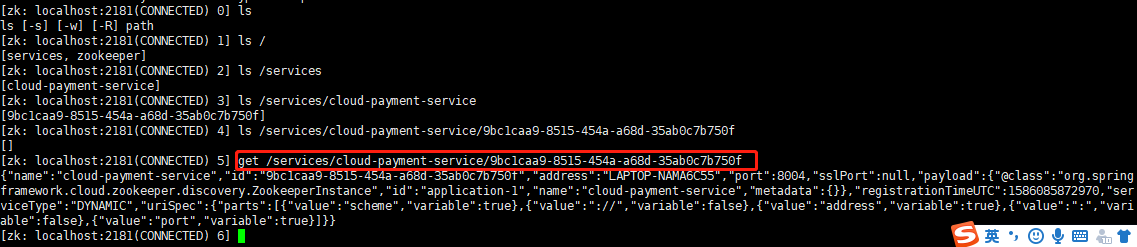


### 测试

1. 通过postman测试



1. 通过连接zookeeper客户端测试



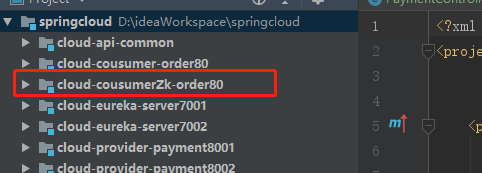
### 结论

zk节点分为4种：临时节点、持久节点、临时带序号节点、持久带序号节点

微服务注册进zookeeper的节点是临时节点即：当服务停止后，在一段时间内zookeeper没有收到心跳，zookeeper会将微服务清除(注意：不是立即清除，而是过了一会儿没有收到心跳才清除，保证了数据一致性，即CAP里的CP)

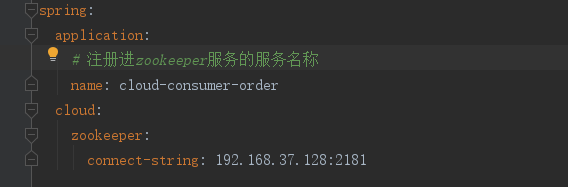
## 消费者(订单服务注册进zookeeper)

### 新建module消费者cloud-cousumerZk-order80

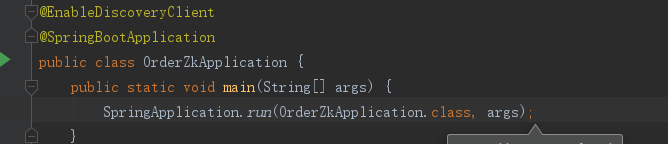


### 导入pom依赖

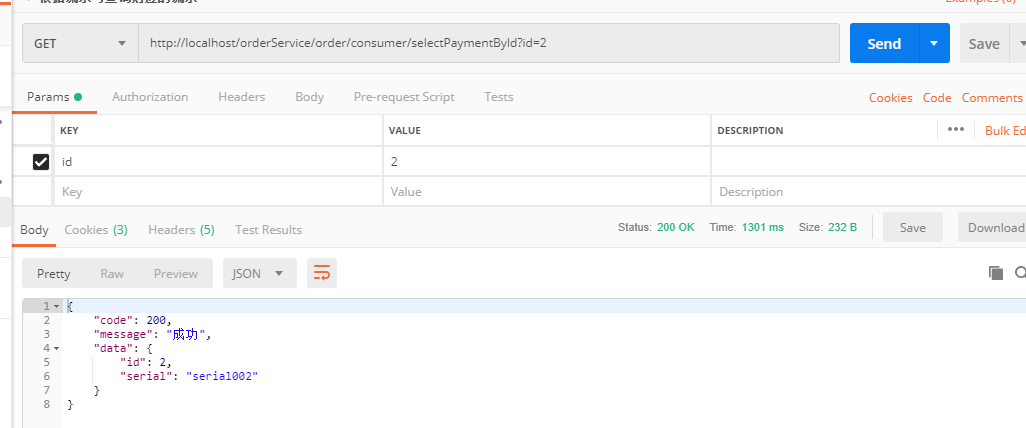
### 修改application.yml

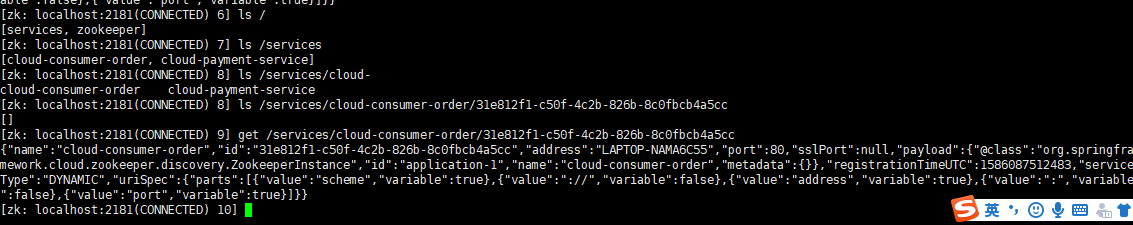


### 启动类



### 测试





# 服务注册-consul

## consul简介

consul是一套开源的分布式服务注册与发现和配置管理系统，由HashiCorp公司用GO语言编写

Consul提供了微服务系统中的服务治理、配置中心、控制总线等功能。这些功能中的每一个都可以根据需要单独使用，也可以一起使用构建全方位的服务网络。Consul提供了一种完整的网络服务解决方案

## 安装并运行consul

### 10.2.1 安装consul

下载地址：<https://www.consul.io/downloads.html>

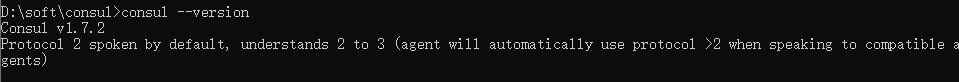




解压到consul文件夹下，解压后是个exe文件

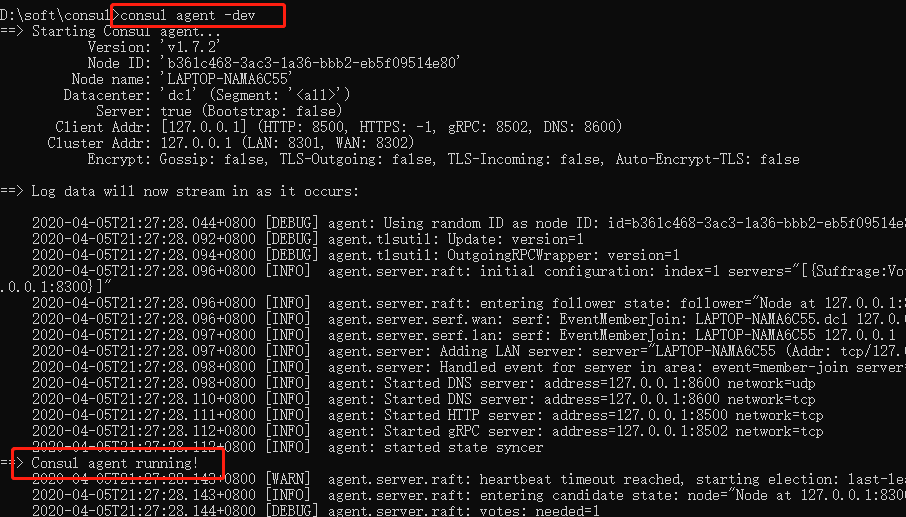
进入exe文件所在的cmd命令窗口，查看consul版本：consul --version





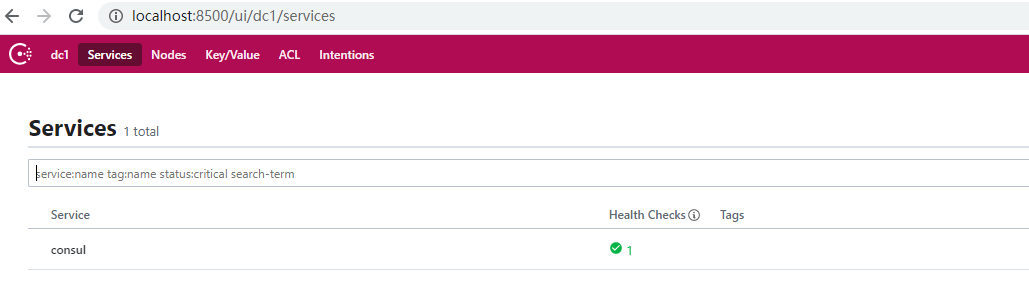
### 10.2.2 运行consul：端口默认8500

consul agent -dev



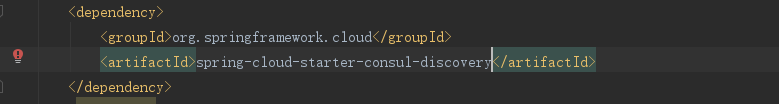
## 访问测试consul

<http://ocalhost:8500>

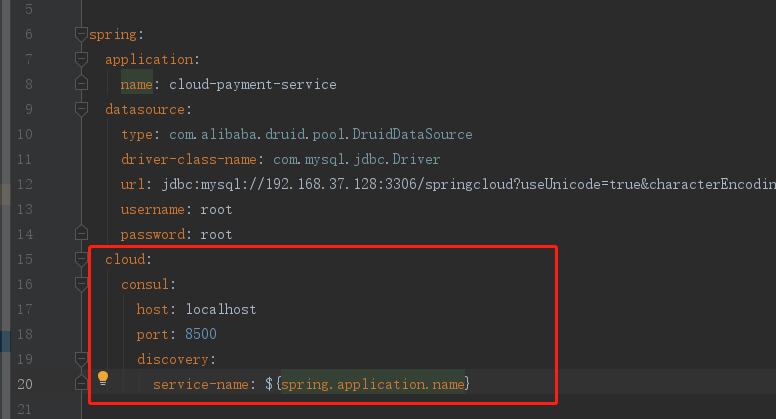


## 生产者注册进consul

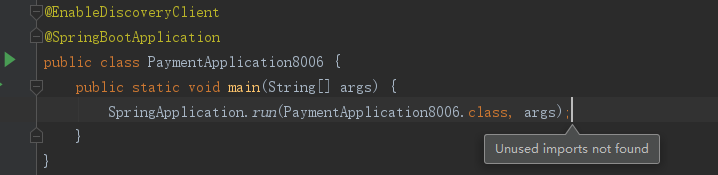
### 创建module并导入依赖



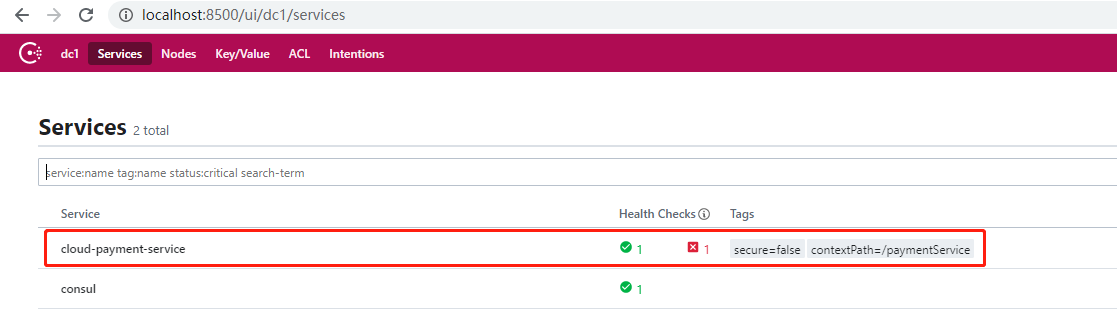
### 修改pom文件

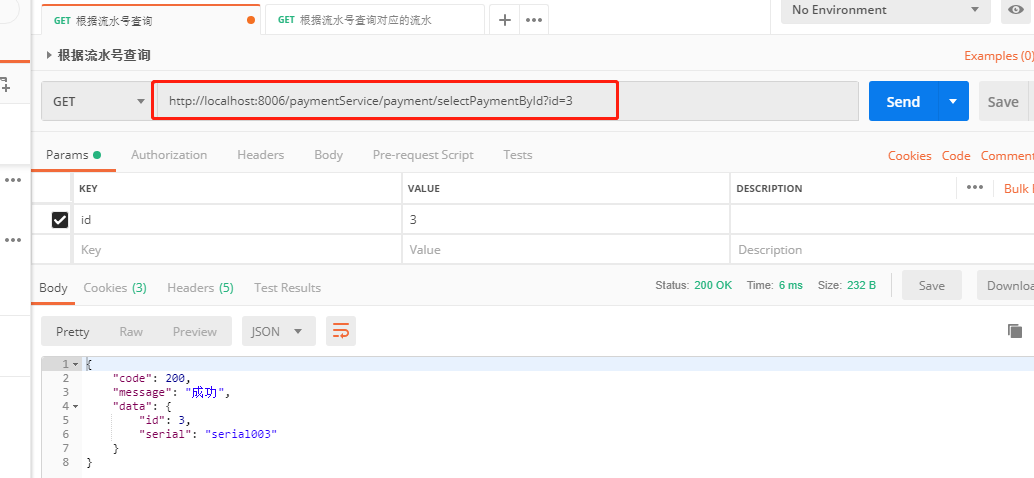


### 启动类



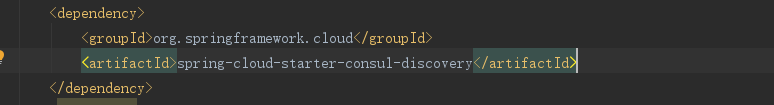
### 测试





## 消费者注册进consul

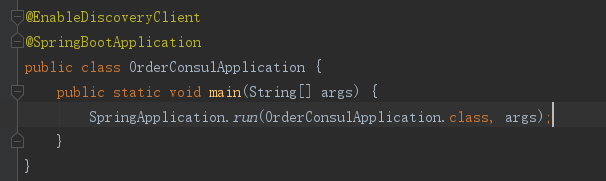
### 创建module并导入依赖



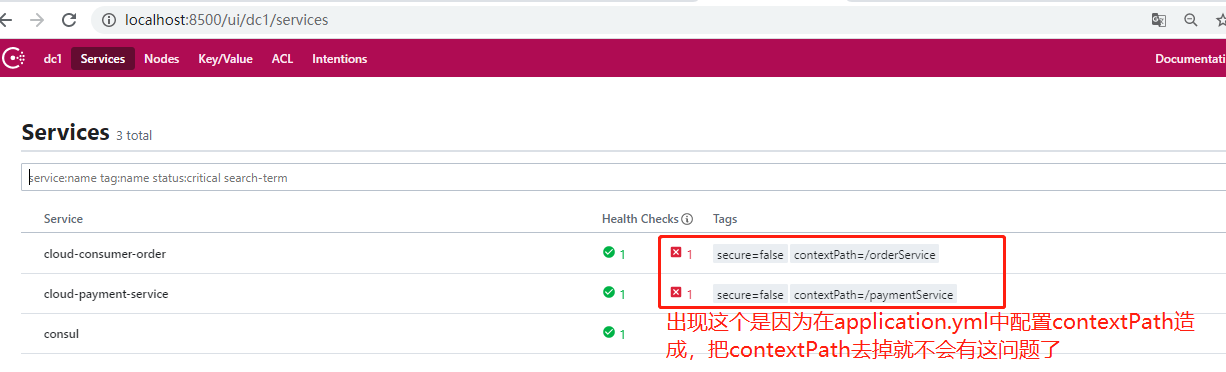
### 修改pom



### 启动类

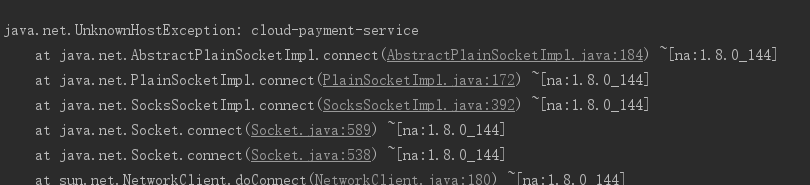


### 测试



### 自己遗留问题

当消费者通过服务注册名访问生产者时，报错：UnknownHostException



# eureka、zookeeper、consul异同

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 组件名 | 语言 | CAP | 服务健康检查 | 对外暴露接口 | springCloud集成 |
| Eureka | Java | AP | 支持 | http | 已集成 |
| Consul | Go | CP | 支持 | http/dns | 已集成 |
| Zookeeper | Java | CP | 支持 | 客户端 | 已集成 |

# 服务调用ribbon

* 1. 啊
  2. 啊

1. P36