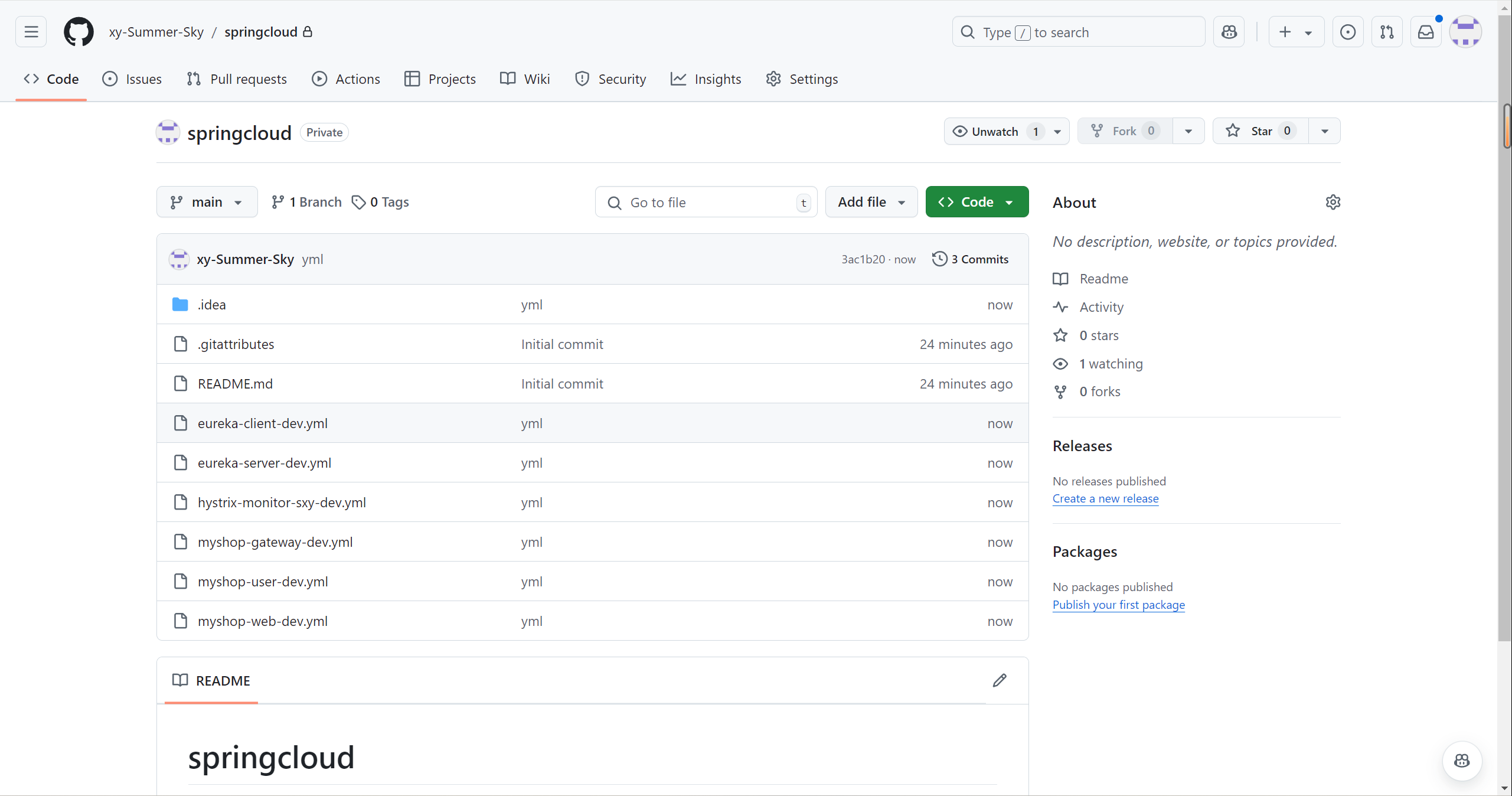
# Spring Cloud Config集中配置

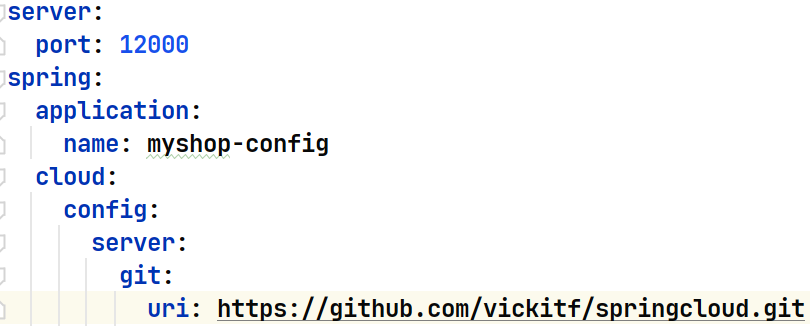
1. Spring Cloud Config-Git方式
2. 把微服务application.yml抽离到github中。创建一个仓库springcloud存放各微服务的配置文件，配置文件命名规则为：微服务名-环境名称，例如：user-dev.yml



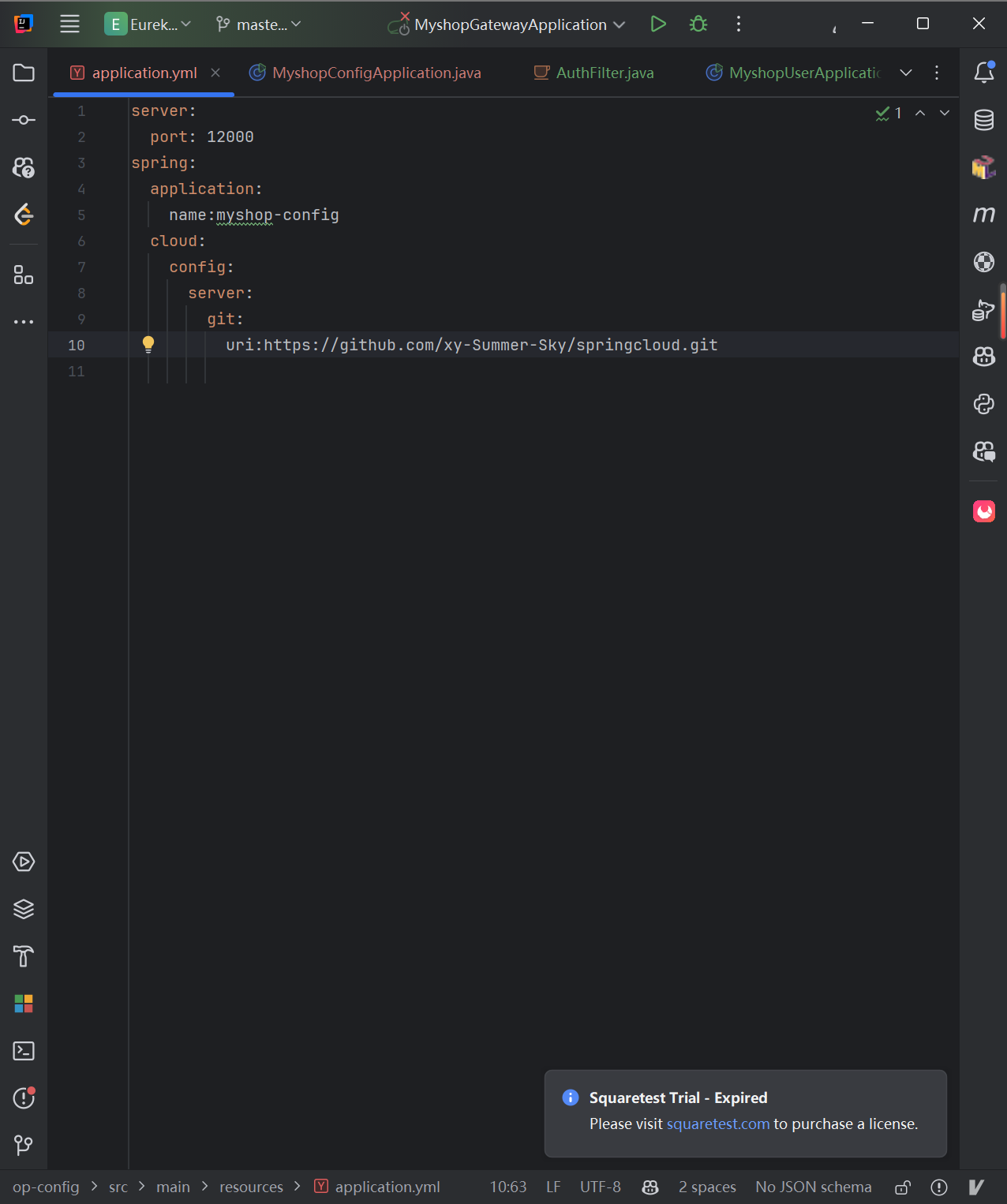
1. 搭建Spring Cloud Config服务端
2. 新建config服务端微服务，例如myshop-config，导入以下依赖：

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-config-server</artifactId>  
</dependency>

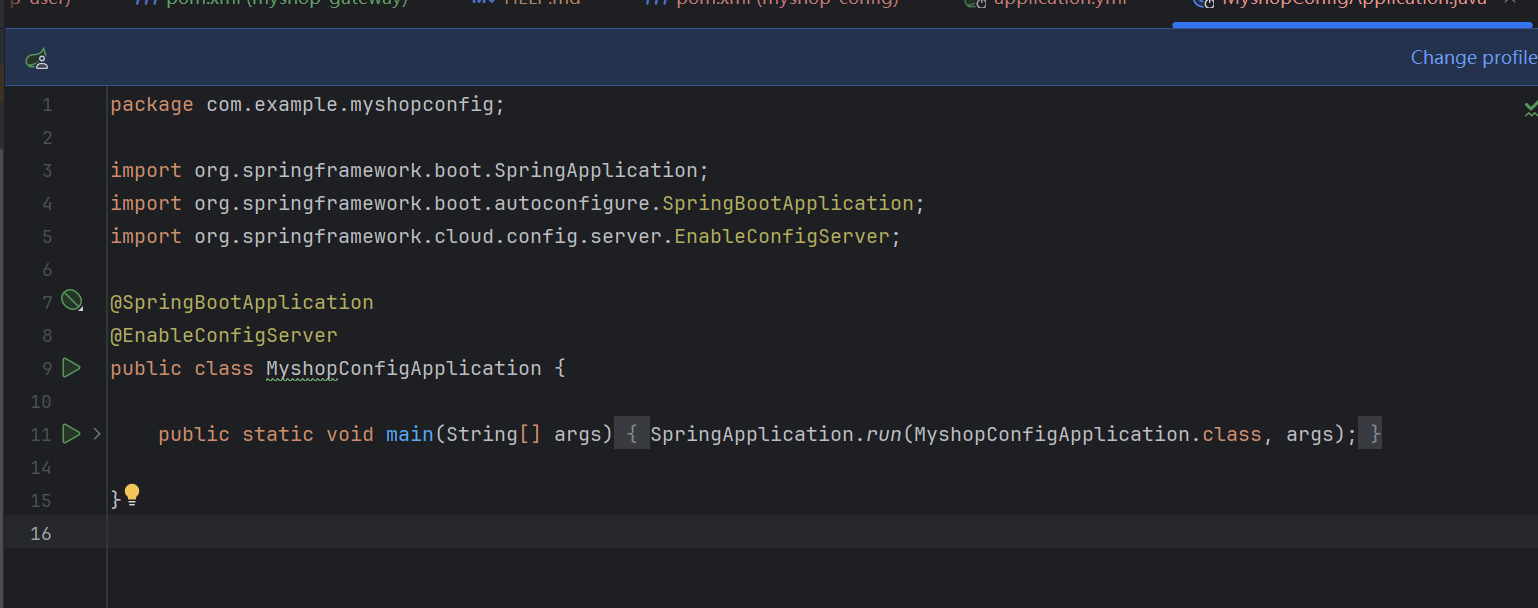
1. 编写config配置文件



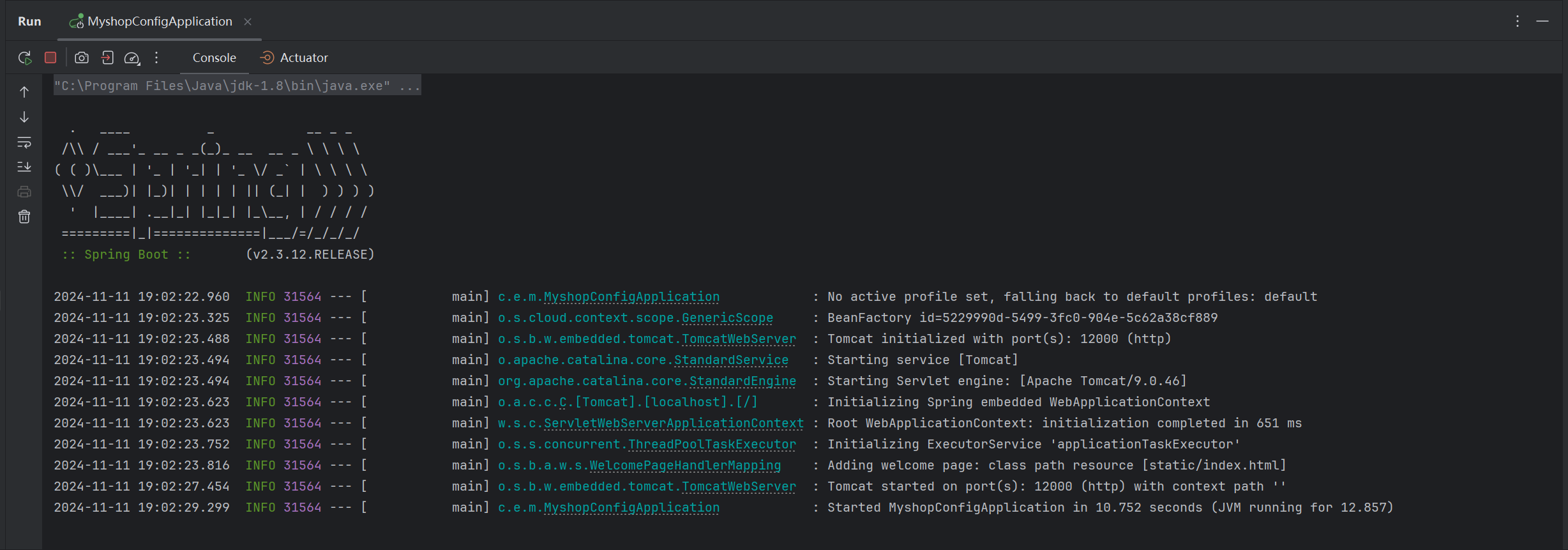
uri为仓库地址

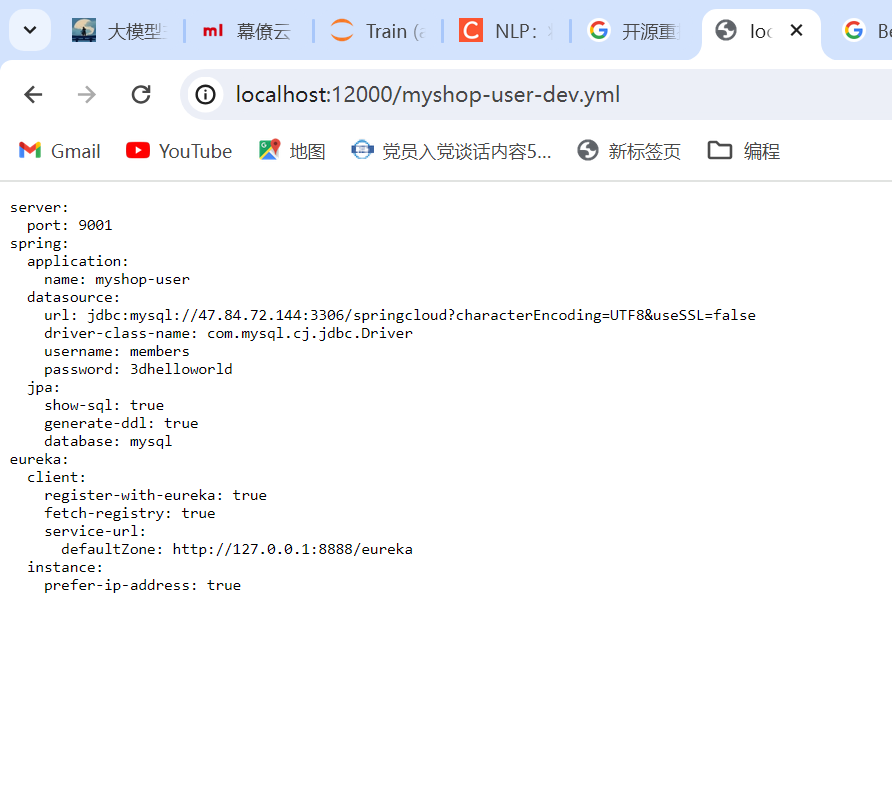


1. 编写config项目的引导类，添加@EnableConfigServer ，开启配置中心服务端功能



1. 启动myshop-config，测试服务端是否能连上仓库，请求localhost:12000/user-dev.yml是否能查看到仓库存储的配置文件（需要截图测试结果）



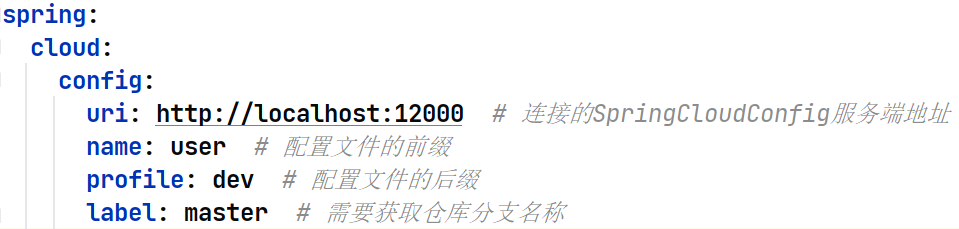


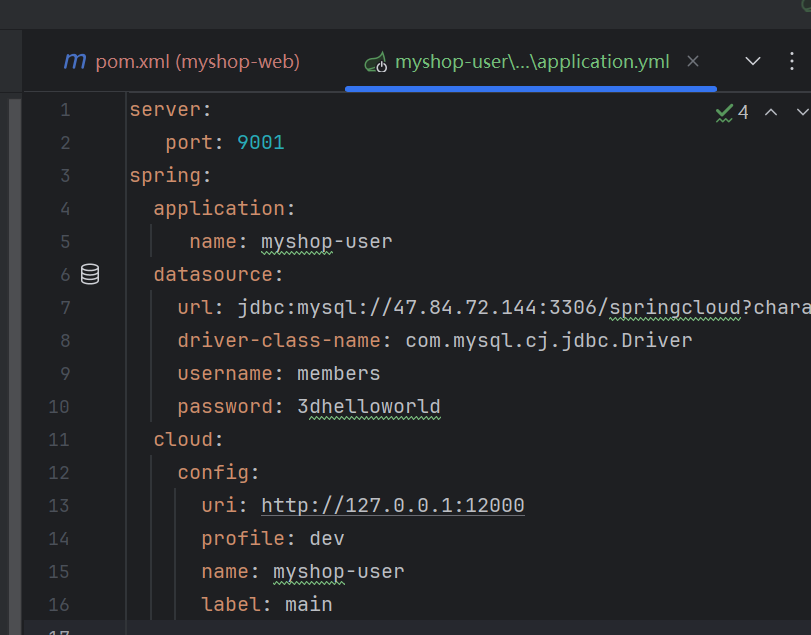
1. 搭建Spring Cloud Config客户端（myshop-user, myshop-user均为客户端）
2. 导入客户端依赖

<!-- 客户端-->  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-config</artifactId>  
</dependency>

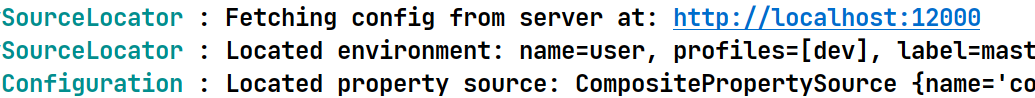
1. resources包中建bootstrap.yml引导配置文件

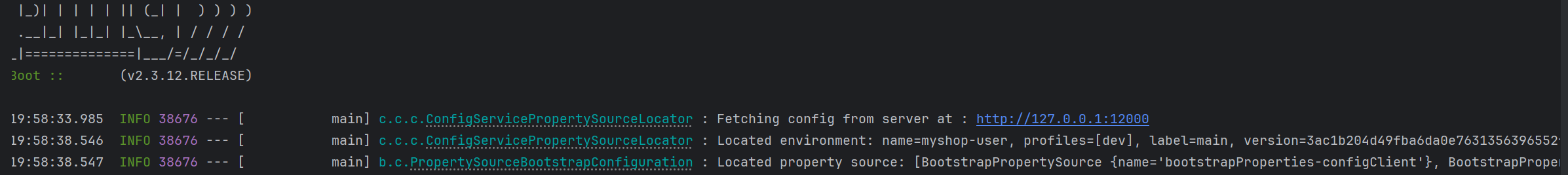
注：name，profile，label需改为自己配置文件信息

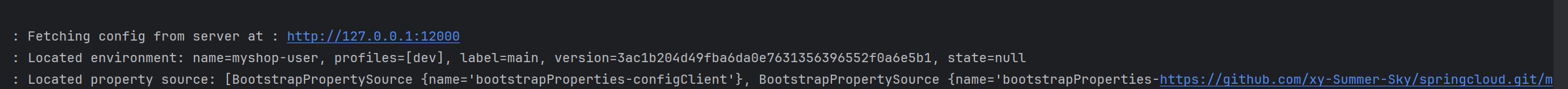




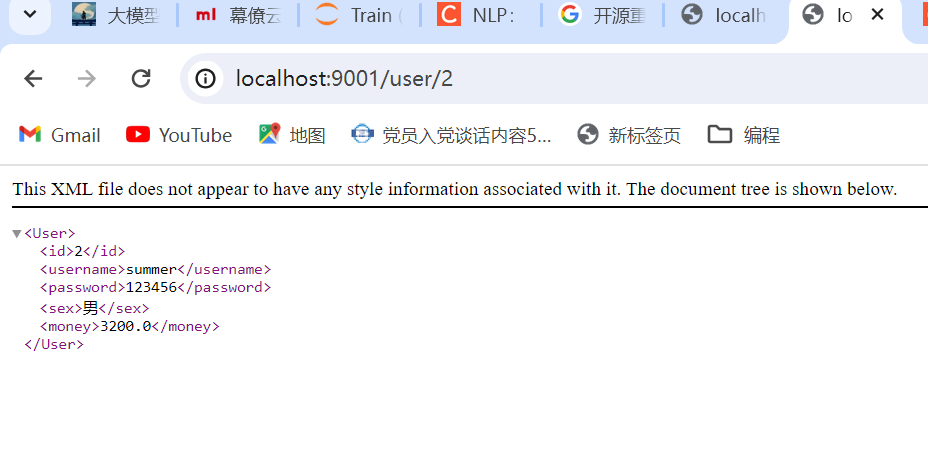
1. 启动微服务，观察是否连上config服务端（截图）



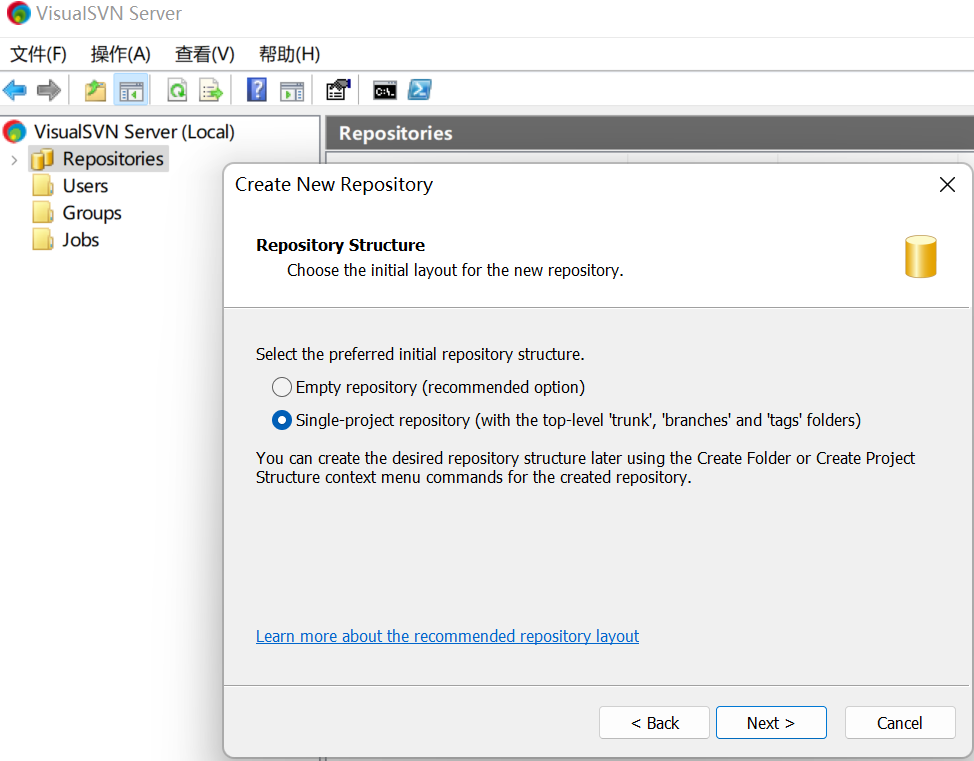


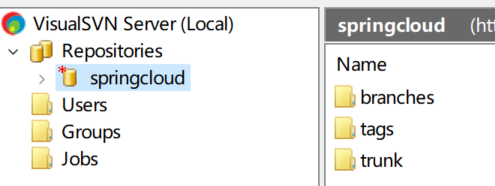


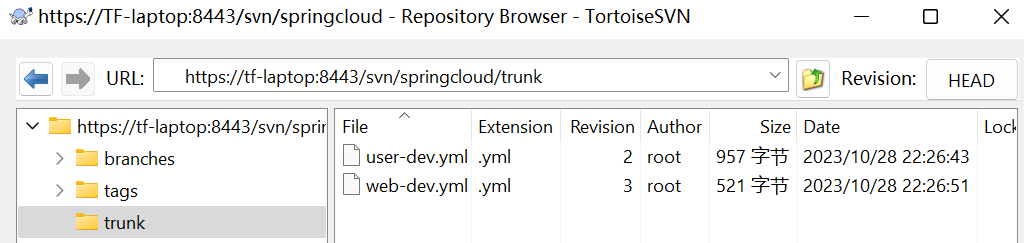
1. 测试。请求用户微服务，若能成功访问，则config配置成功。（截图）

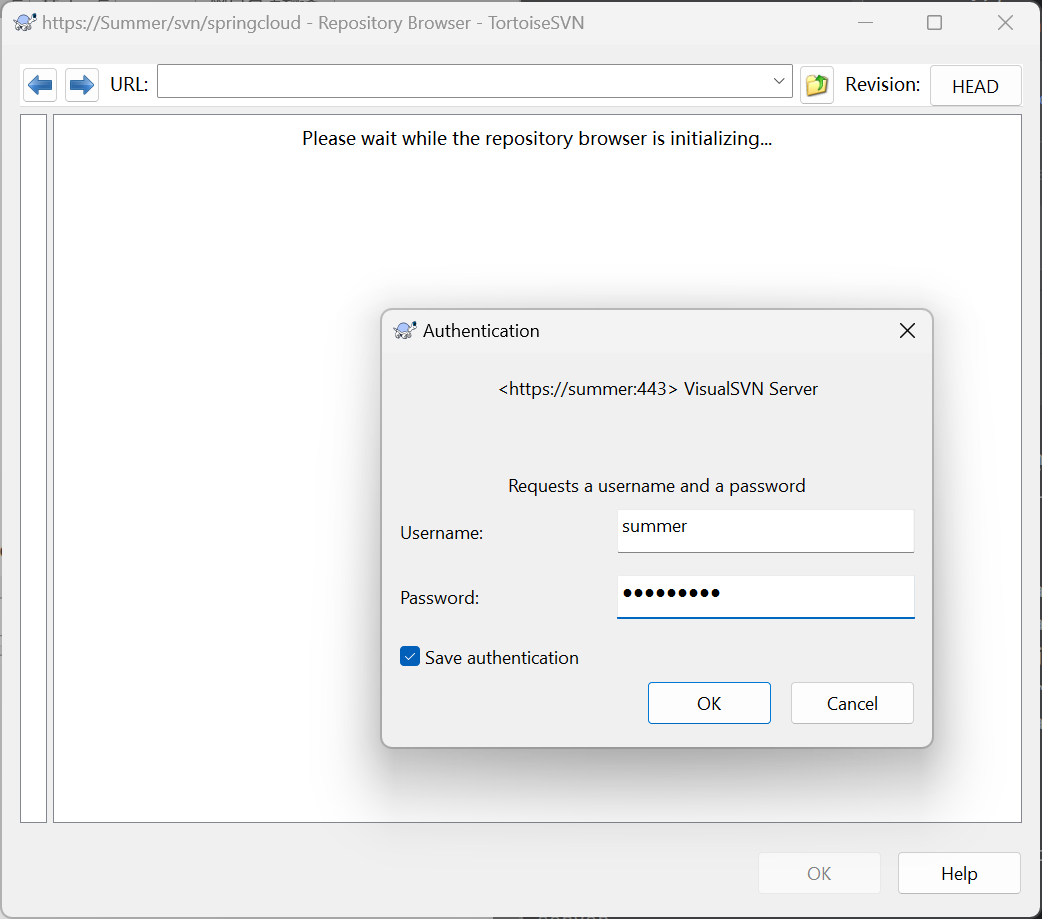


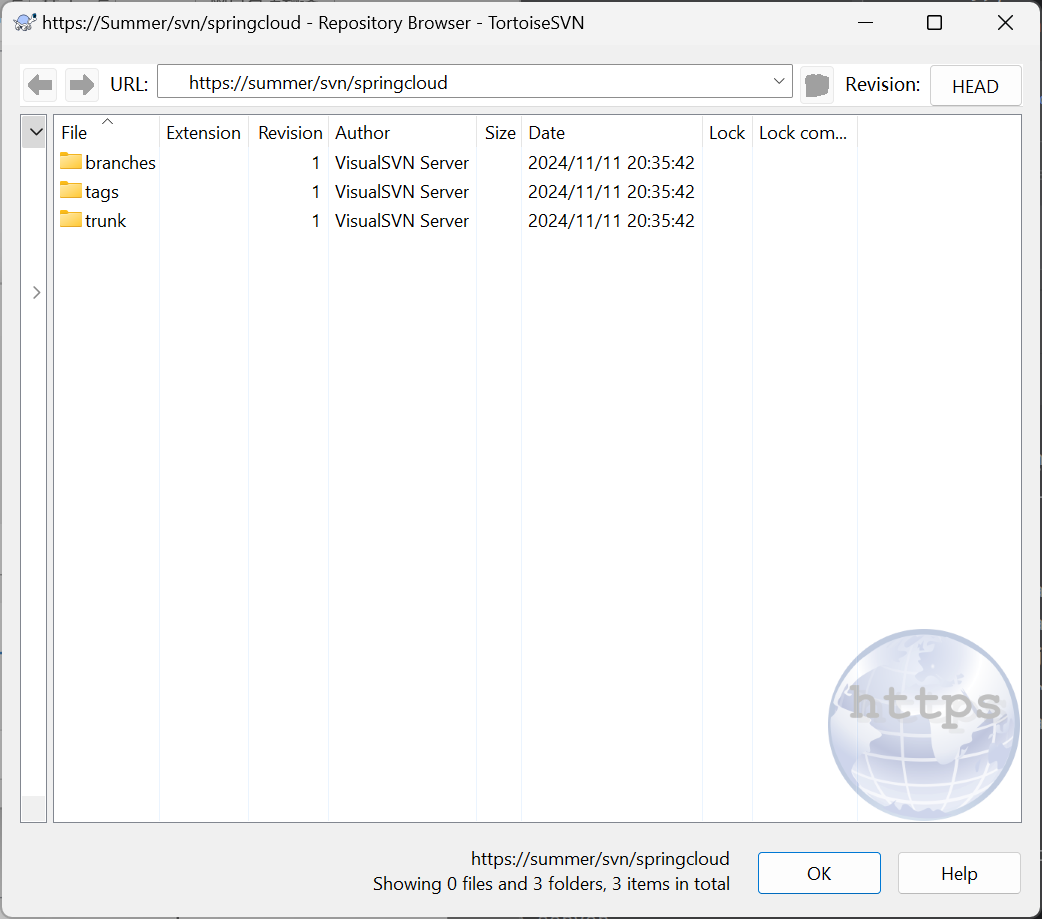
1. Spring Cloud Config-SVN方式
2. 创建SVN仓库，并将配置文件上传至SVN仓库

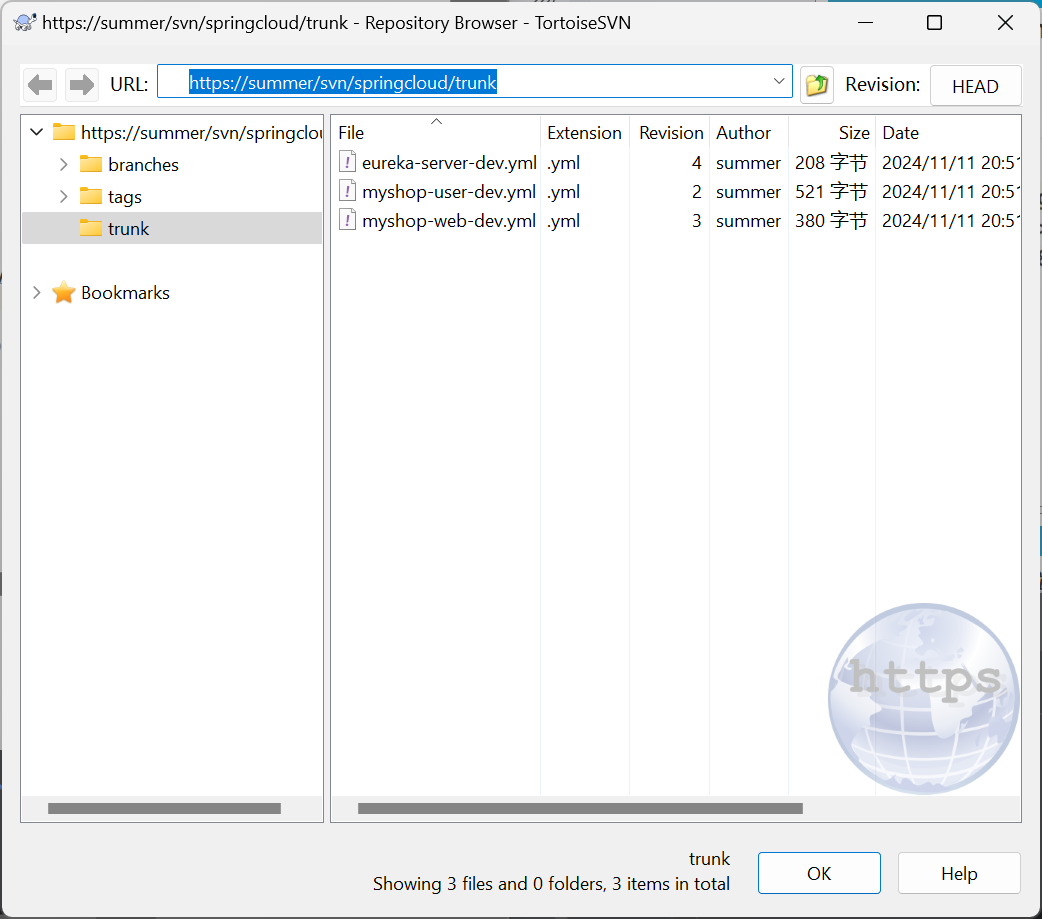


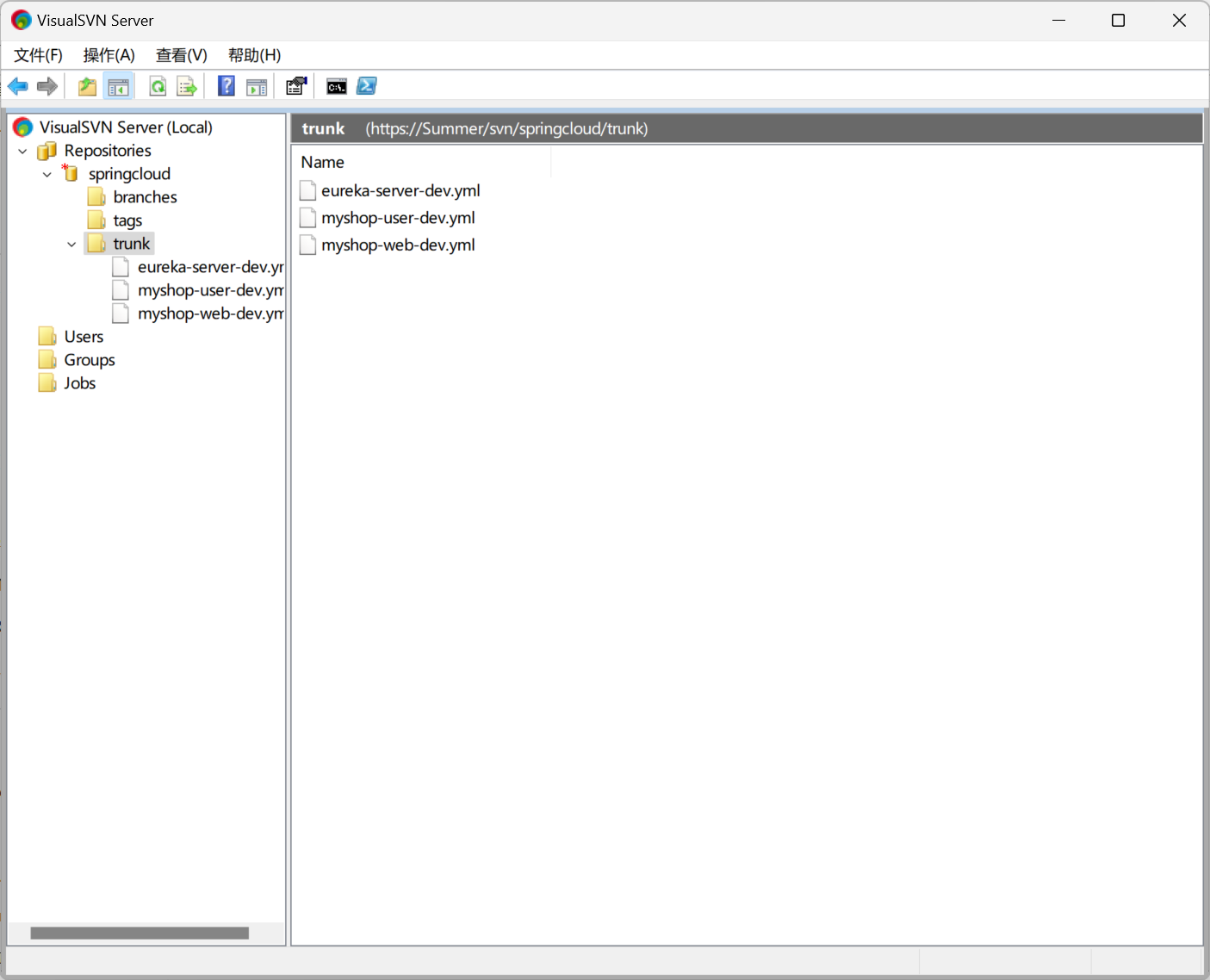










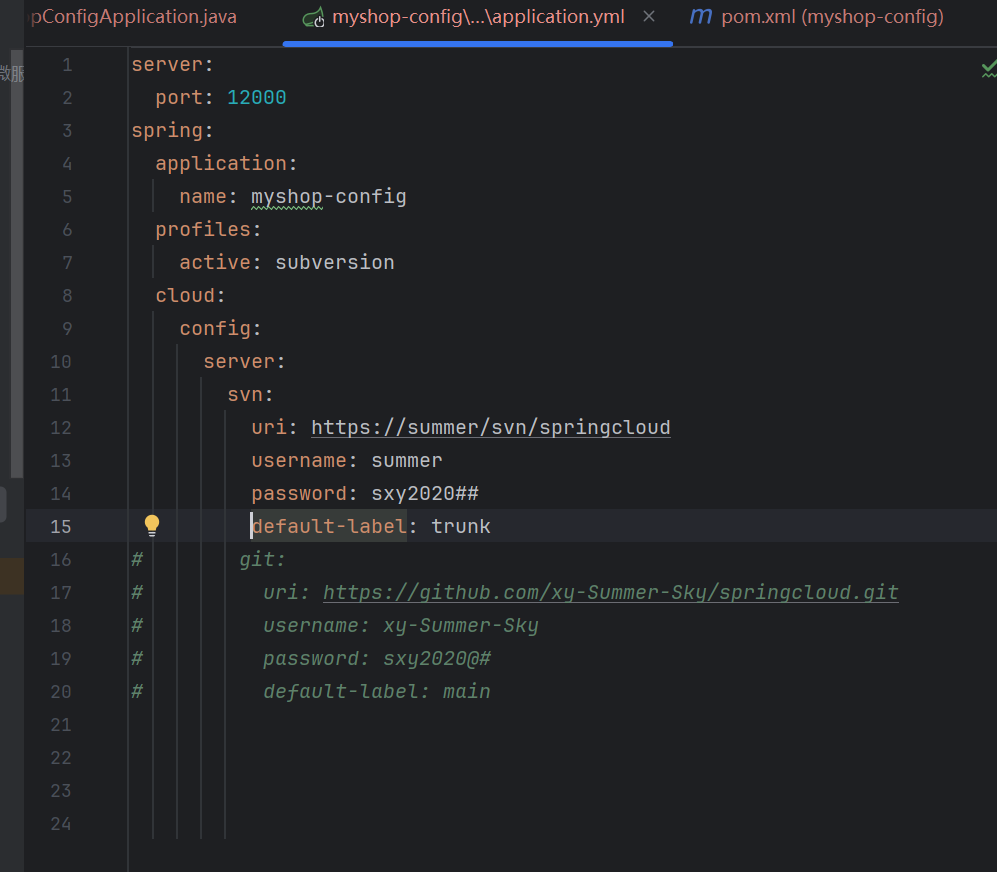


1. myshop-config导入SVN依赖

<!-- svn服务端 -->  
<dependency>  
 <groupId>org.tmatesoft.svnkit</groupId>  
 <artifactId>svnkit</artifactId>  
 <version>1.8.10</version>  
</dependency>

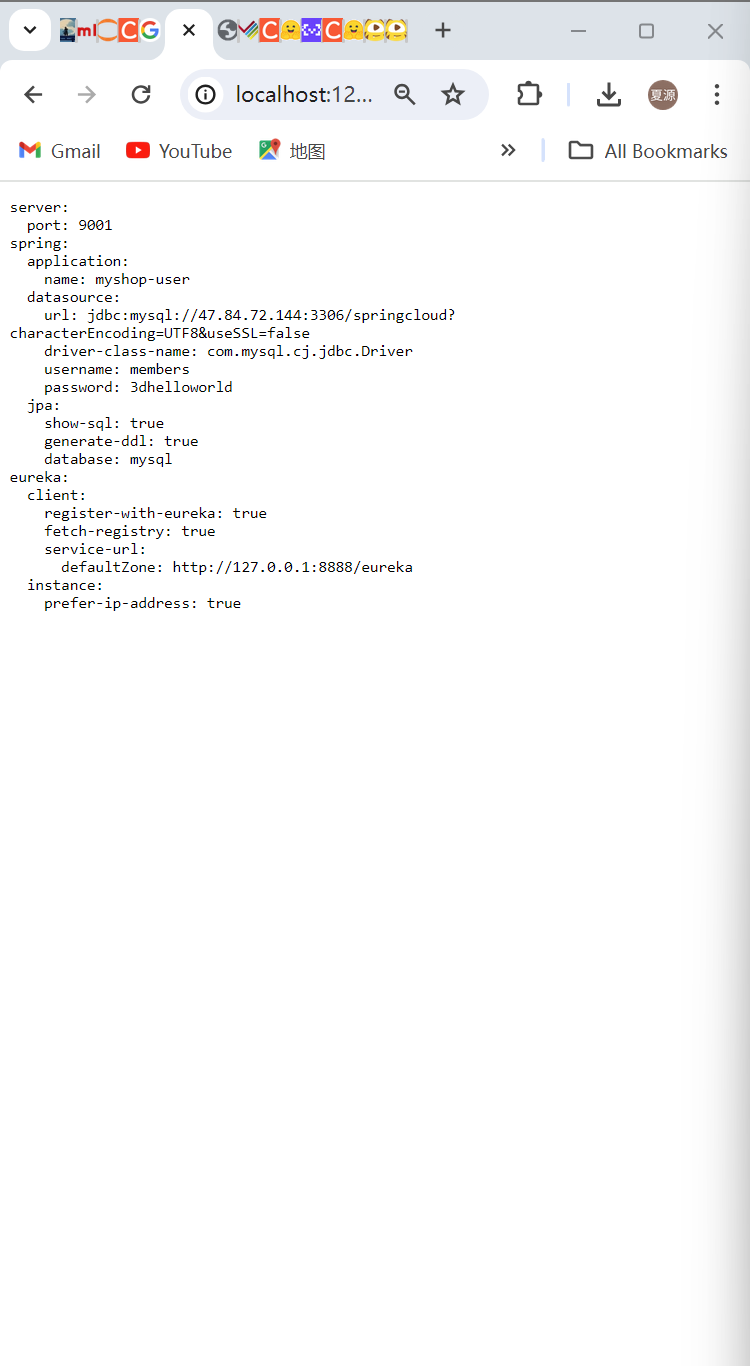
1. 修改config服务端配置文件

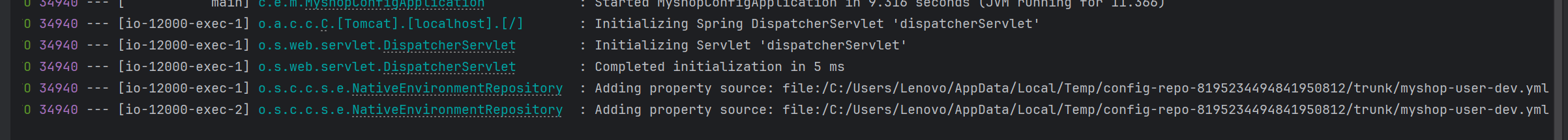


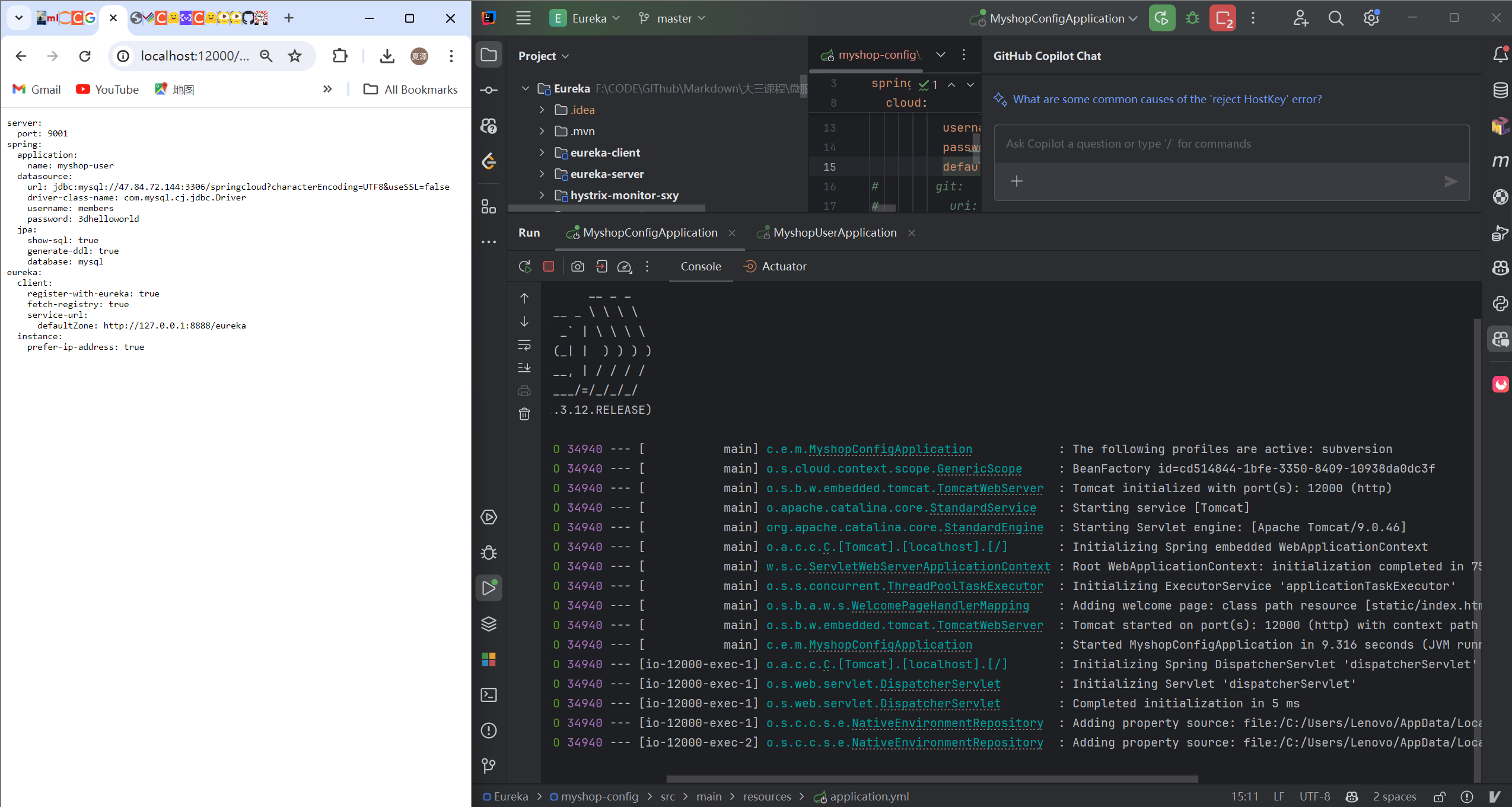


1. 重启config，测试服务端是否能连上仓库（截图）

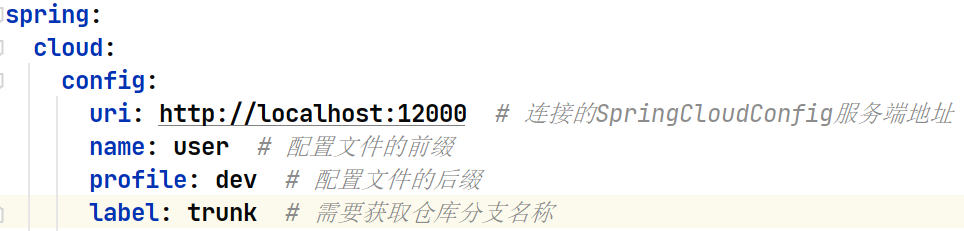




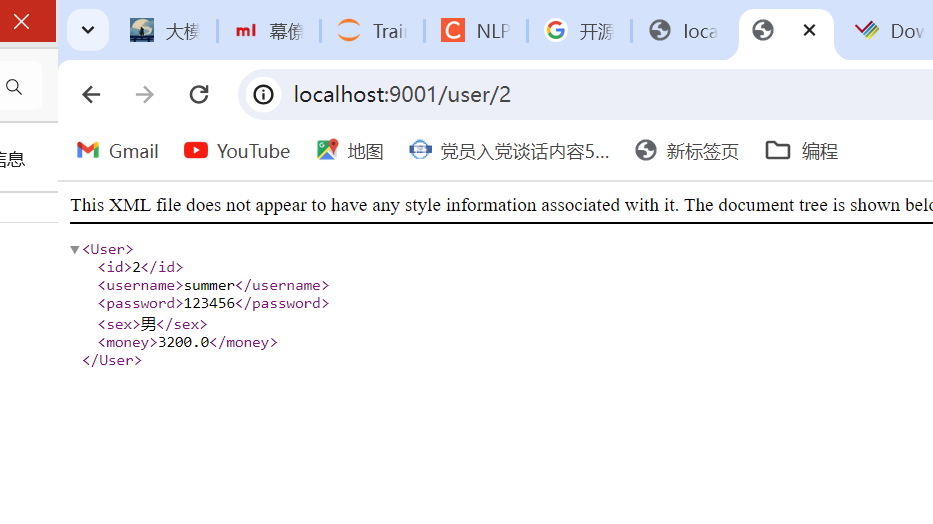


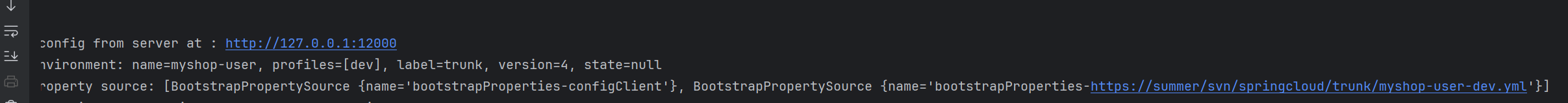


1. 修改微服务端bootstrap.yml，并重启微服务，验证服务能否从SVN获取配置文件，记录请求localhost:9001/user/2的返回结果（截图）





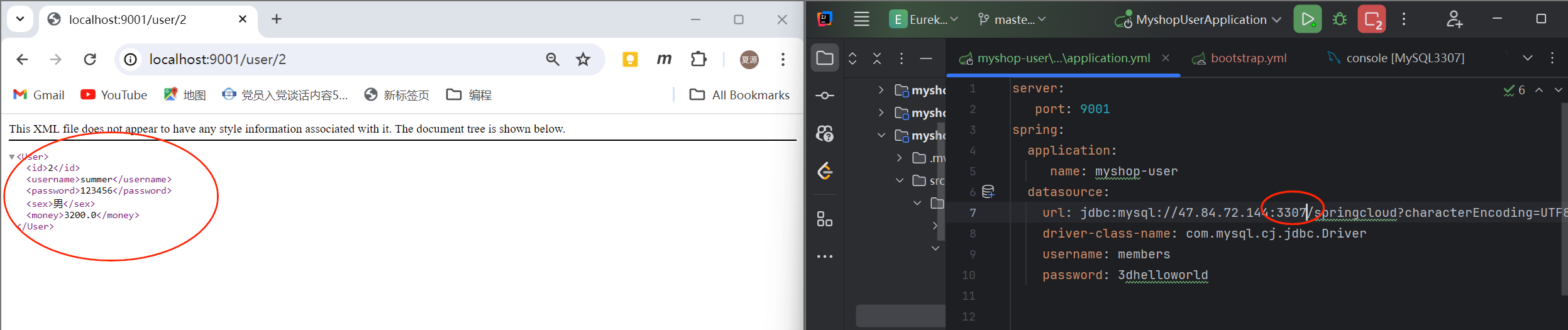




1. Spring Cloud Bus消息总线
2. 新建一个端口为3307的同名数据库，但3307与3306两数据库表内容不同
3. SVN仓库中修改用户微服务配置，将数据源改为3307的数据库，通过服务端查看该配置文件是否修改生效？

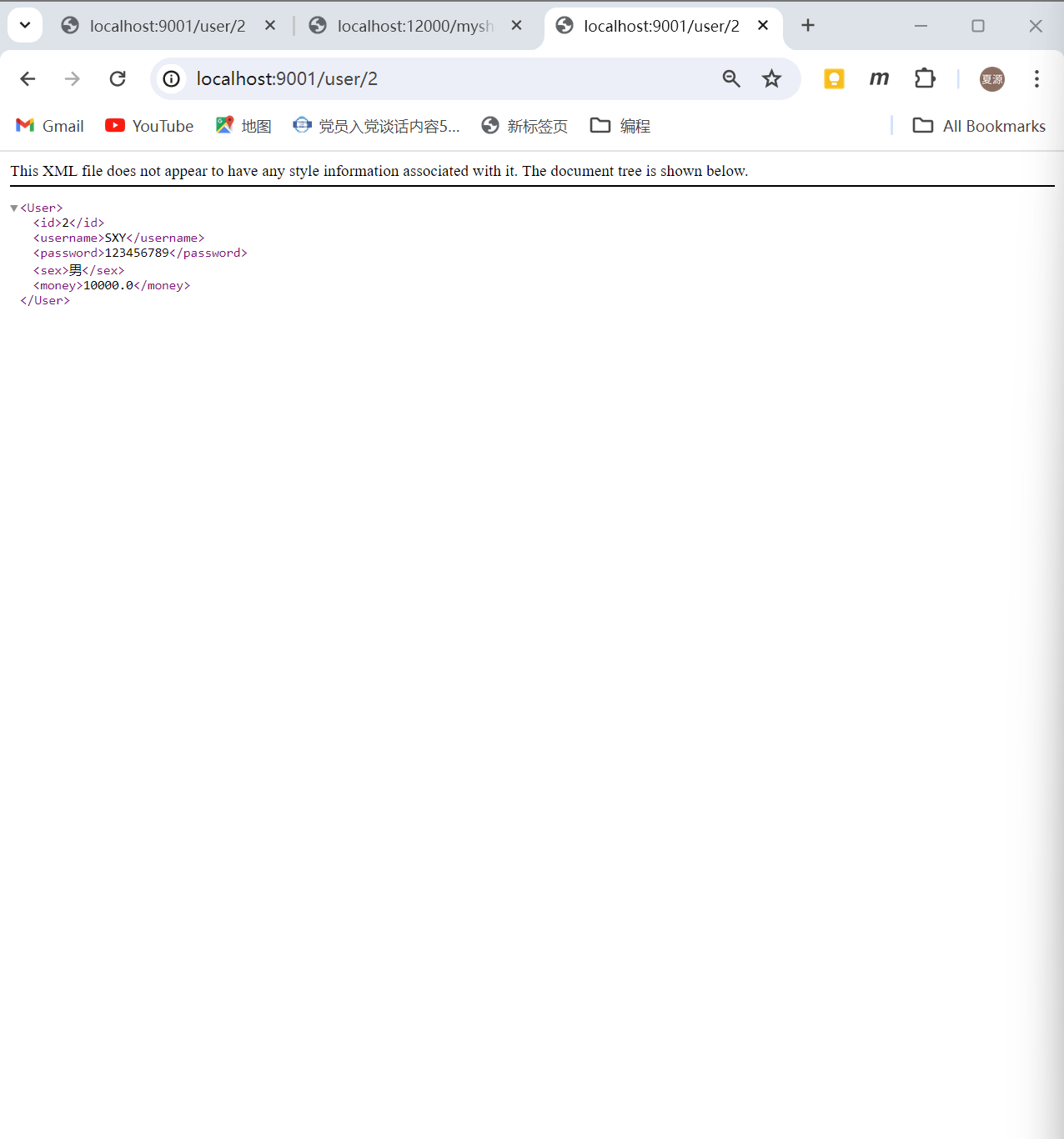
配置文件改变，但是不生效（即依旧是旧连接）

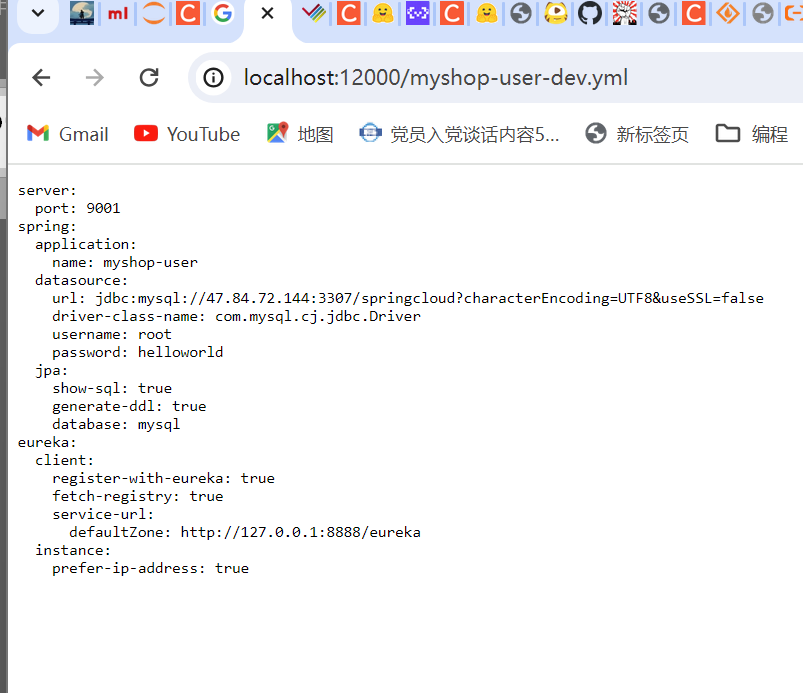


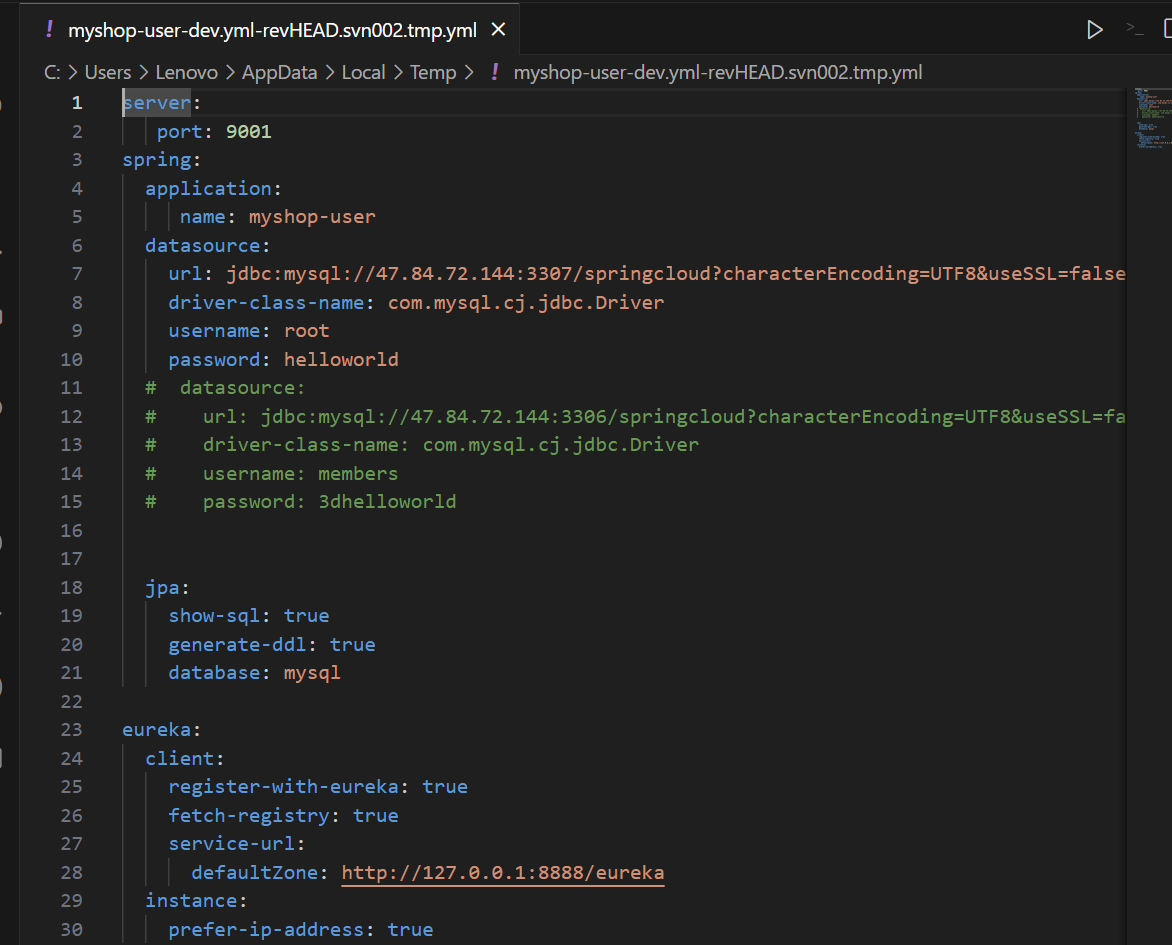




更新仓库的配置文件重启才会生效







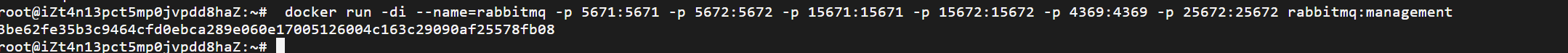
思考如何使修改的配置文件生效？（截图+回答）

1. docker中创建rabbitmq容器

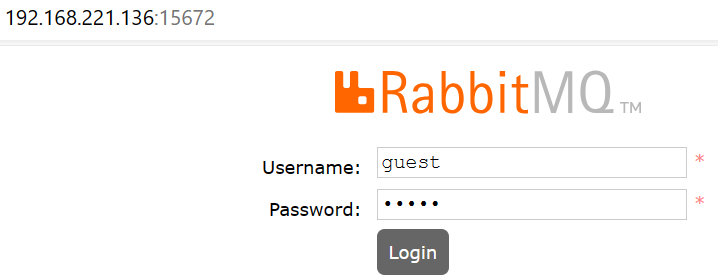
1）下载rabbitmq镜像：docker pull rabbitmq:management

2）查看已下载的镜像：docker images

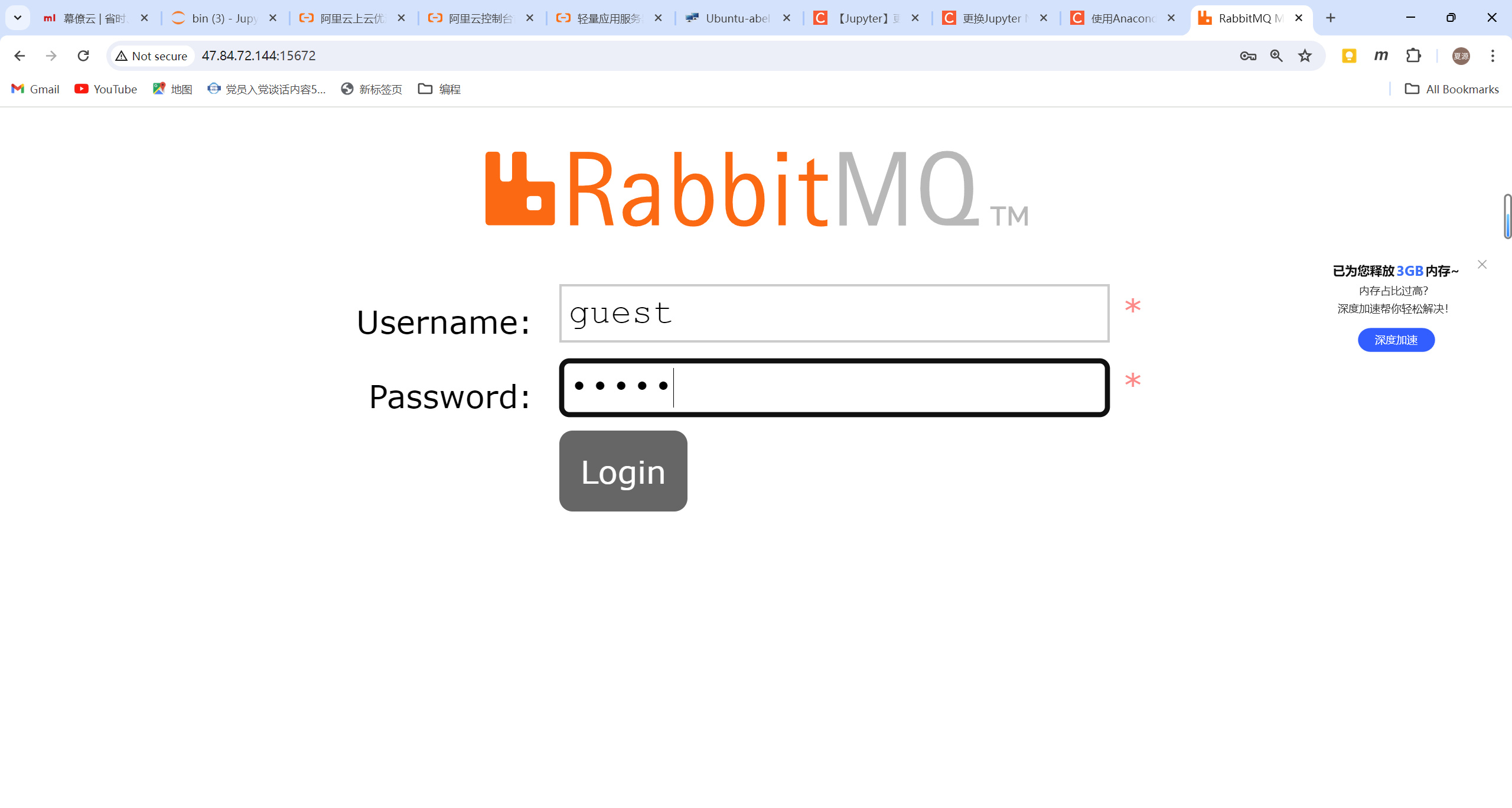
3）启动rabbitmq: docker run -di --name=rabbitmq -p 5671:5671 -p 5672:5672 -p 15671:15671 -p 15672:15672 -p 4369:4369 -p 25672:25672 rabbitmq:management



4）访问rabbitmq，用户名密码均为guest



我的登录



1. 配置Spring Cloud Bus服务端（myshop-config服务中配置）
2. 导入依赖

<!-- spring cloud bus依赖-->  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-bus</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-stream-binder-rabbit</artifactId>  
</dependency>

1. 配置文件中暴露springcloudbus的触发地址

spring:  
 rabbitmq:  
 host: 192.168.221.136 # rabbitmq所在虚机地址

management: #暴露触发消息总线的地址  
 endpoints:  
 web:  
 exposure:  
 include: bus-refresh

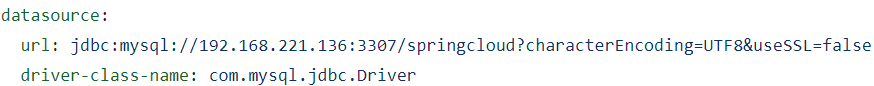
1. 配置Spring Cloud Bus客户端（用户微服务，购票微服务中配置）
2. 导入依赖

<!-- springcloudbus客户端-->  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-bus</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-stream-binder-rabbit</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>

1. 配置文件中配置rabbitmq地址

spring:  
 rabbitmq:  
 host: 192.168.221.136 # rabbitmq所在虚机地址

1. 配置文件中将数据库切换为3307



1. Postman中触发地址更新

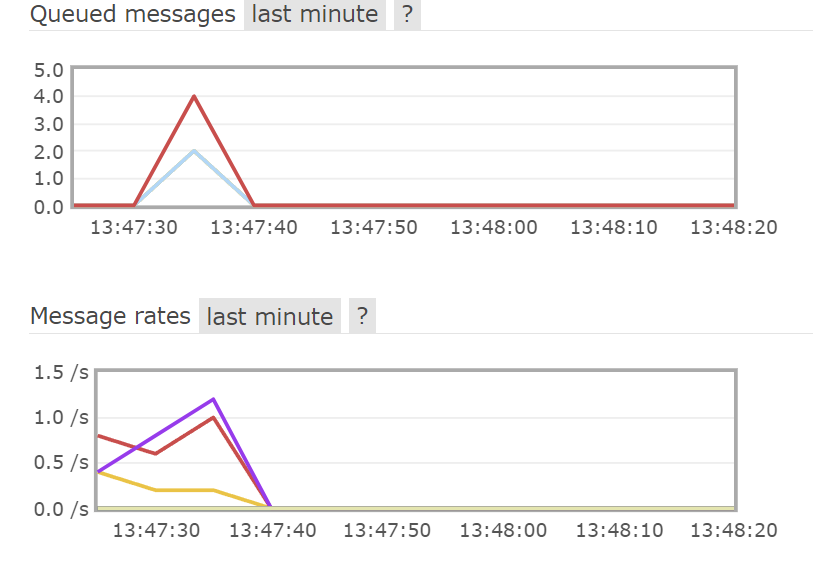
http://localhost:12000/actuator/bus-refresh

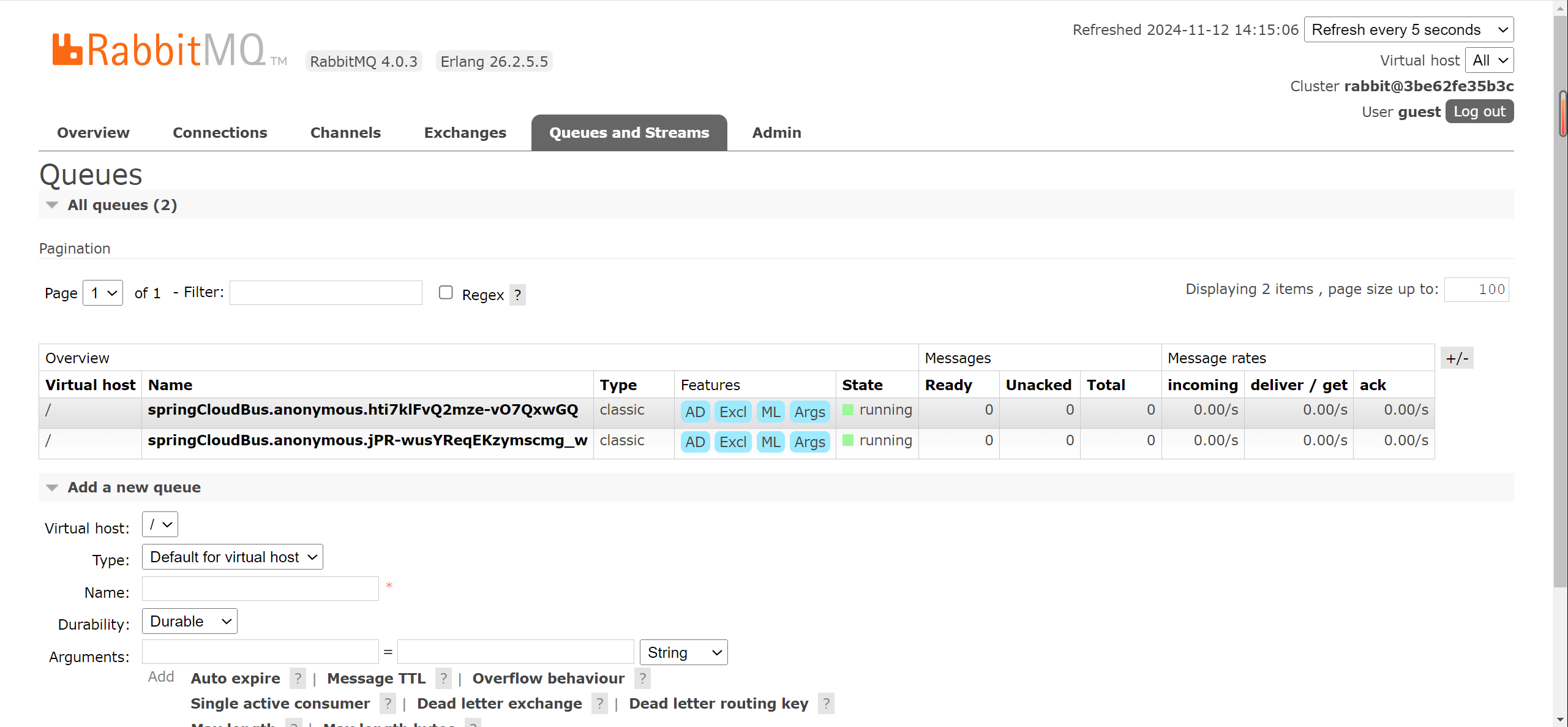


无需重启微服务，请求localhost:9001/user/2，记录请求结果，观察修改的配置文件（切换到3307数据库）是否生效？（截图）



观察rabbitmq界面，是否生成配置传输的queues？（截图）



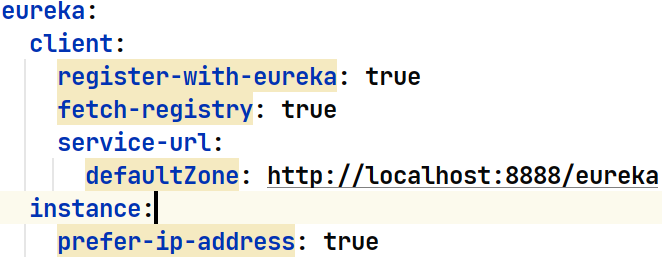


1. 高可用配置中心
2. myshop-config配置中心注册到Eureka

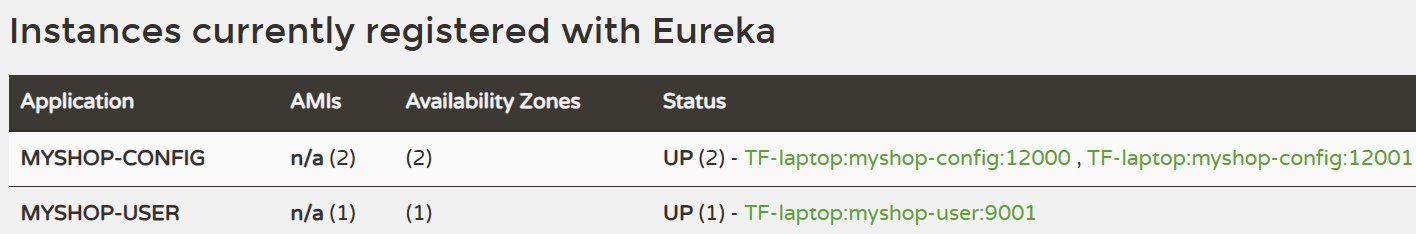
1）导入Eureka客户端依赖



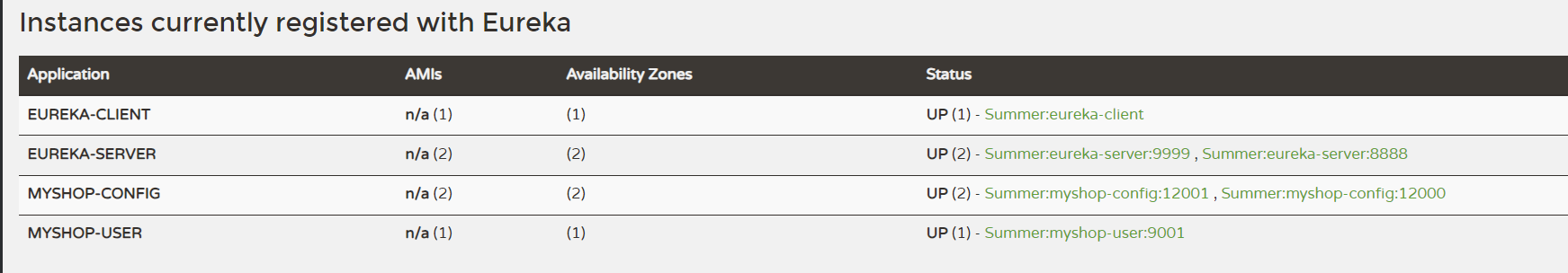
2）myshop-config配置文件添加Eureka配置

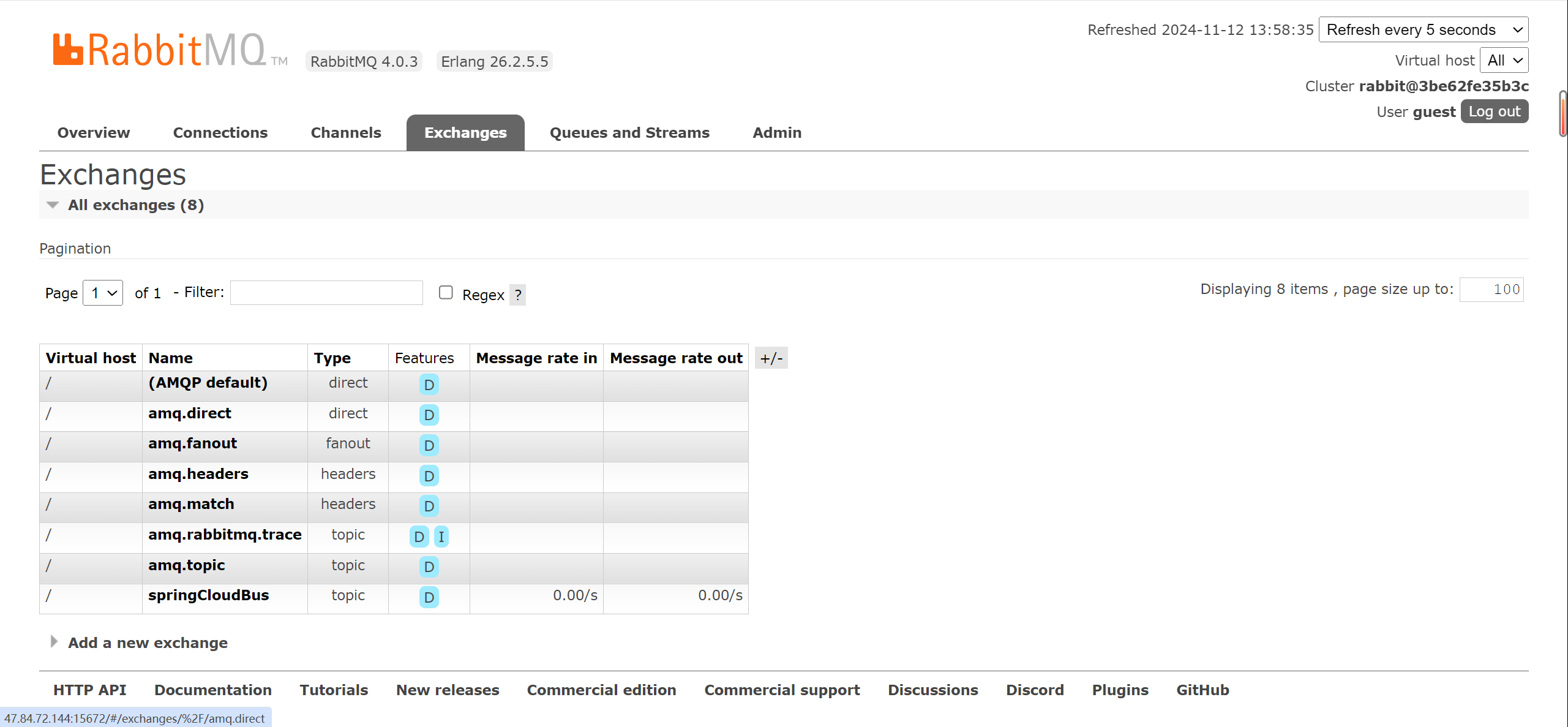


1. config启动类加@EnableEurekaClient注解
2. 开启多个配置中心实例（截图）



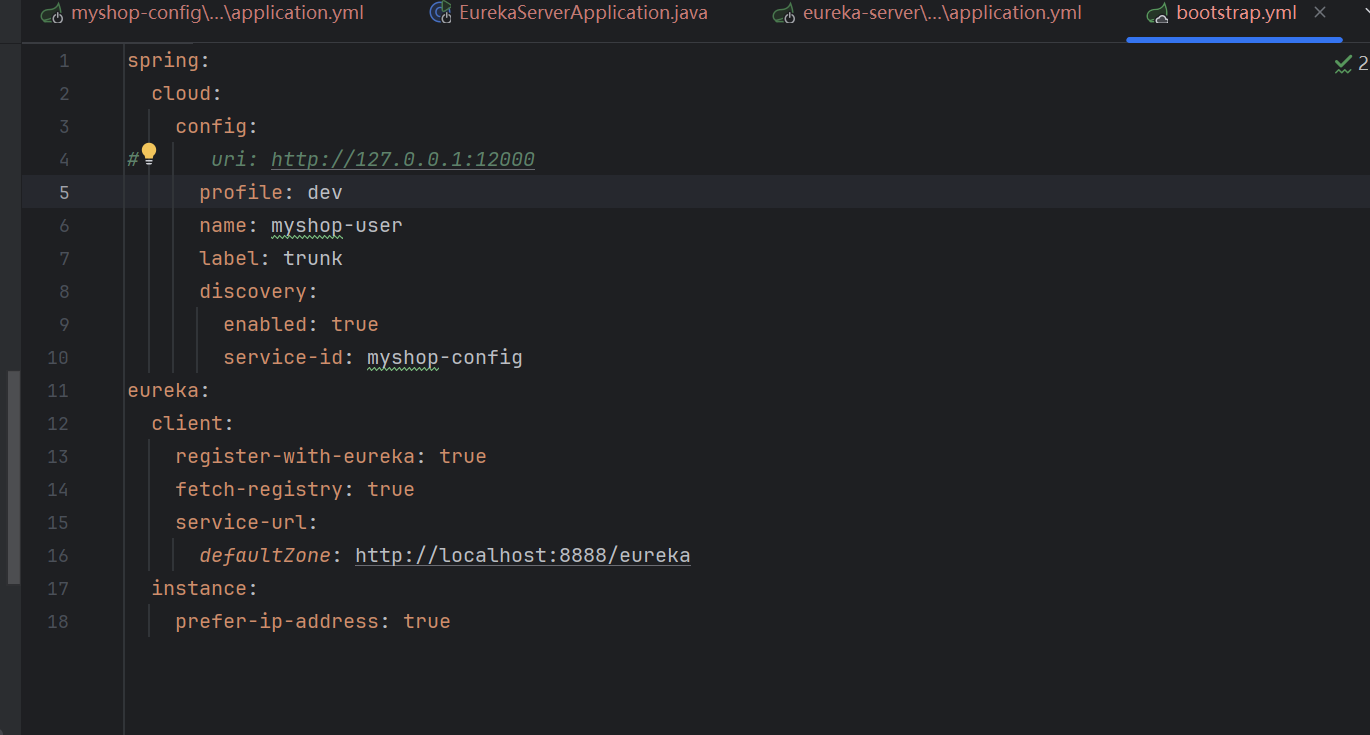
我的截图





1. 修改Spring Cloud Config客户端引导配置文件bootstrap.yml

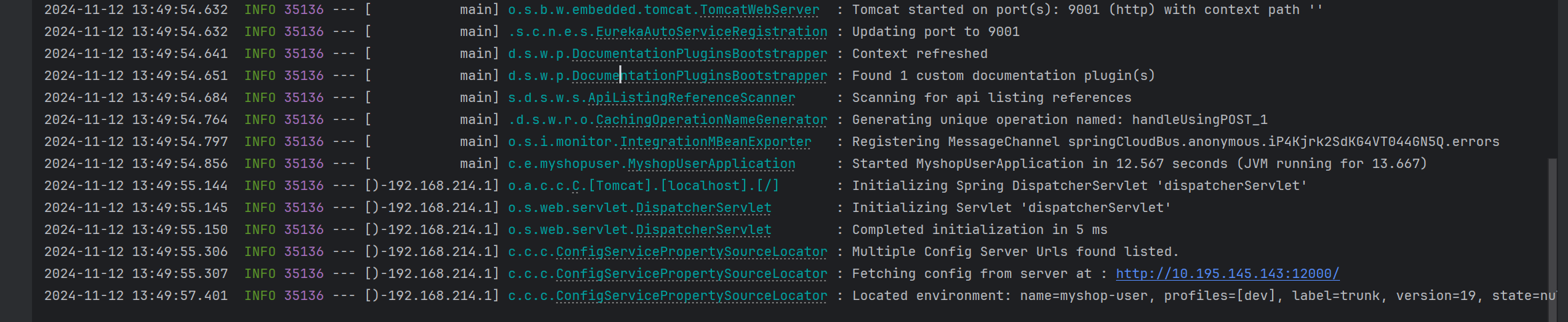
myshop-config注册到Eureka之后，客户端需通过服务名访问配置中心，且bootstrap.yml中需要包含eureka的配置







1. 启动微服务，记录启动信息中连接的是哪一个config（截图）



1. 测试用户微服务可用性。停止一个config，观察微服务是否仍可用？（截图+说明）

如server控制台所示，12000端口config已经取消挂载



但依然可以访问调用服务

