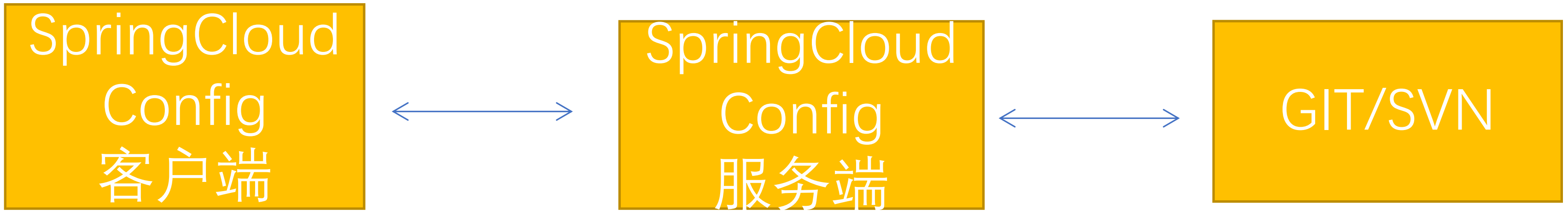
第十一章 Spring Cloud Config 集中配置

为什么需要SpringCloudConfig配置中心

- 随着线上项目变的日益庞大，每个项目都散落着各种配置文件，如果采用分布式的开发模式，需要的配置文件随着服务增加而不断增多。某一个基础服务信息变更，都会引起一系列的更新和重启，运维苦不堪言也容易出错。配置中心便是解决此类问题的灵丹妙药。

认识Spring Cloud Config

- Spring Cloud Config是Spring Cloud团队创建的一个项目，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。

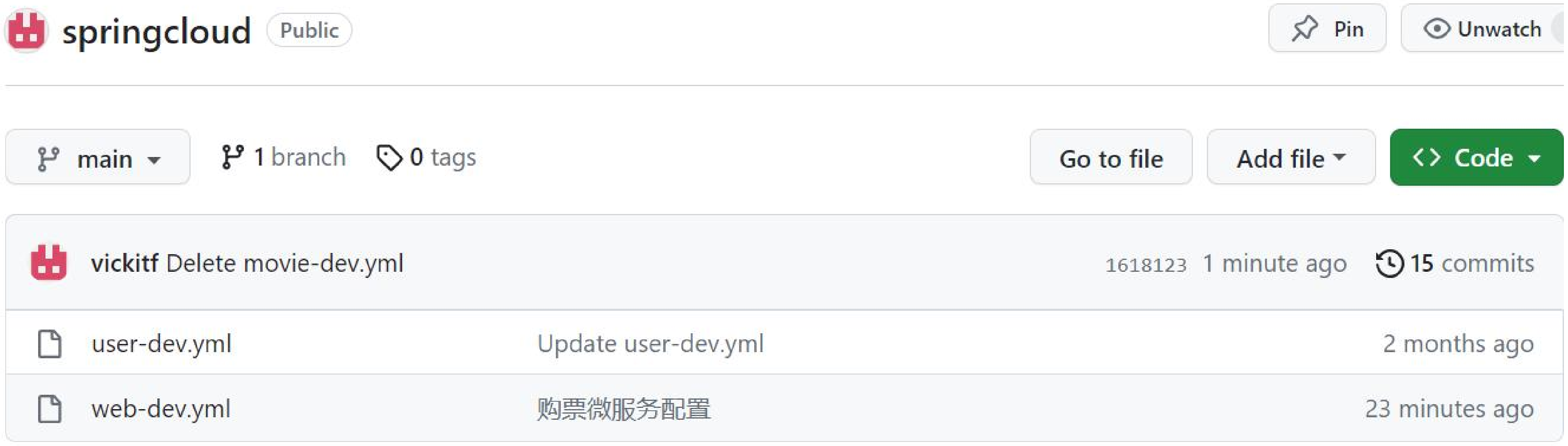


Spring Cloud Config-Git方式

- 第一步，把微服务application.yml抽离到github中。

user-dev.yml

web-dev.yml


springcloud Public

Pin | Unwatch

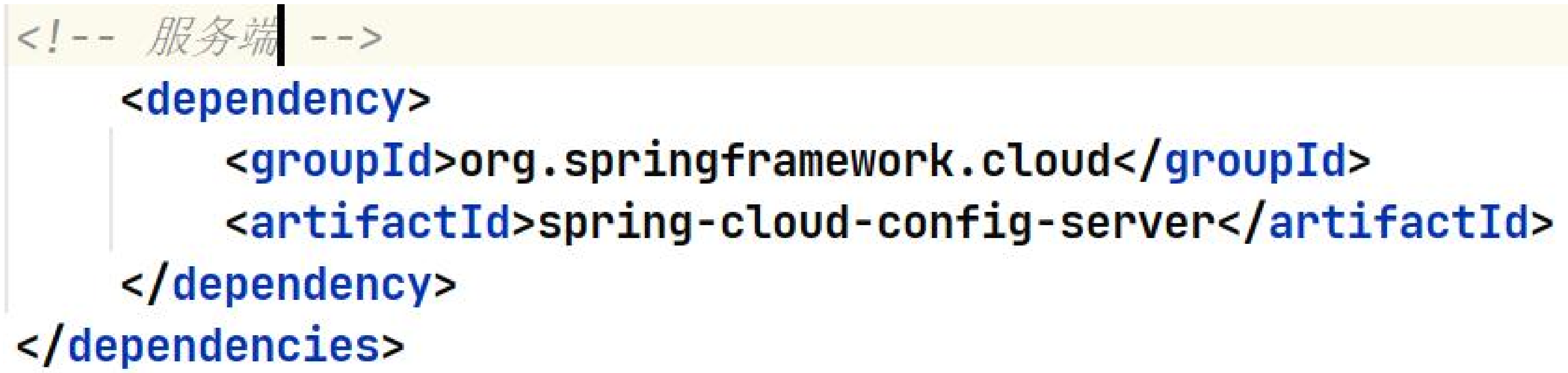
main | 1 branch | 0 tags | Go to file | Add file | Code

vickitf Delete movie-dev.yml 1618123 1 minute ago 15 commits

user-dev.yml	Update user-dev.yml	2 months ago
web-dev.yml	购票微服务配置	23 minutes ago

Spring Cloud Config-Git方式

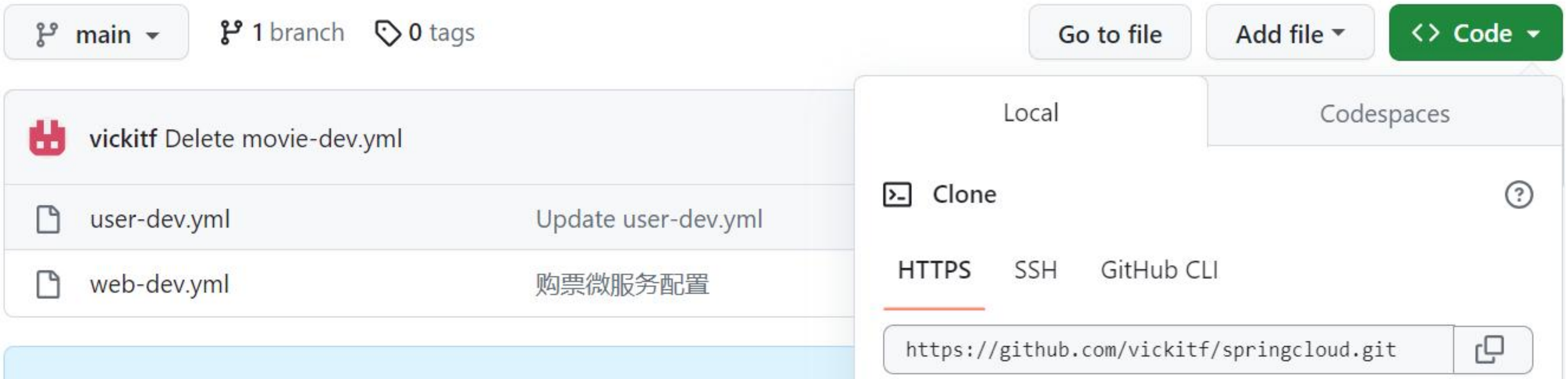
- **第二步，搭建Spring Cloud Config服务端**



```
<!-- 服务端 -->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-config-server</artifactId>
    </dependency>
</dependencies>
```

• 编写config配置文件

```yaml
server:
  port: 12000
spring:
  application:
    name: myshop-config
  cloud:
    config:
      server:
        git:
          uri: https://github.com/vickitf/springcloud.git
```



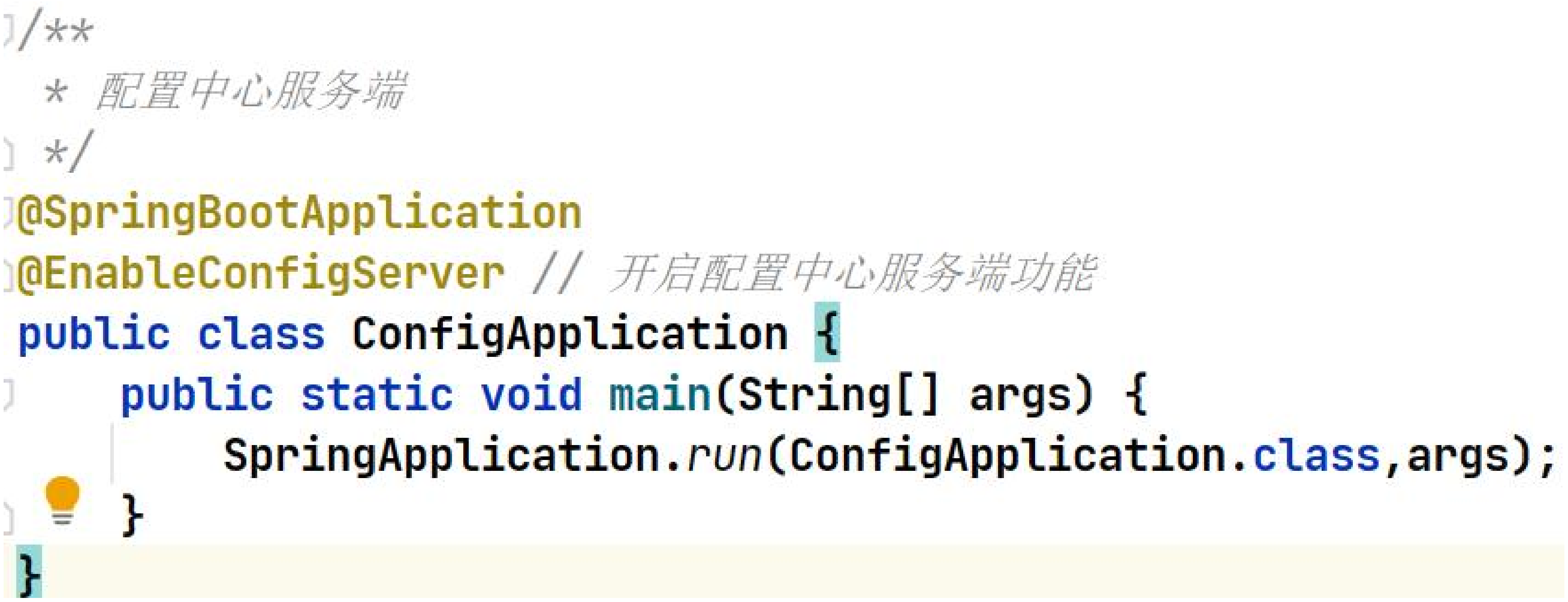
vickitf Delete movie-dev.yml

user-dev.yml Update user-dev.yml

web-dev.yml 购票微服务配置

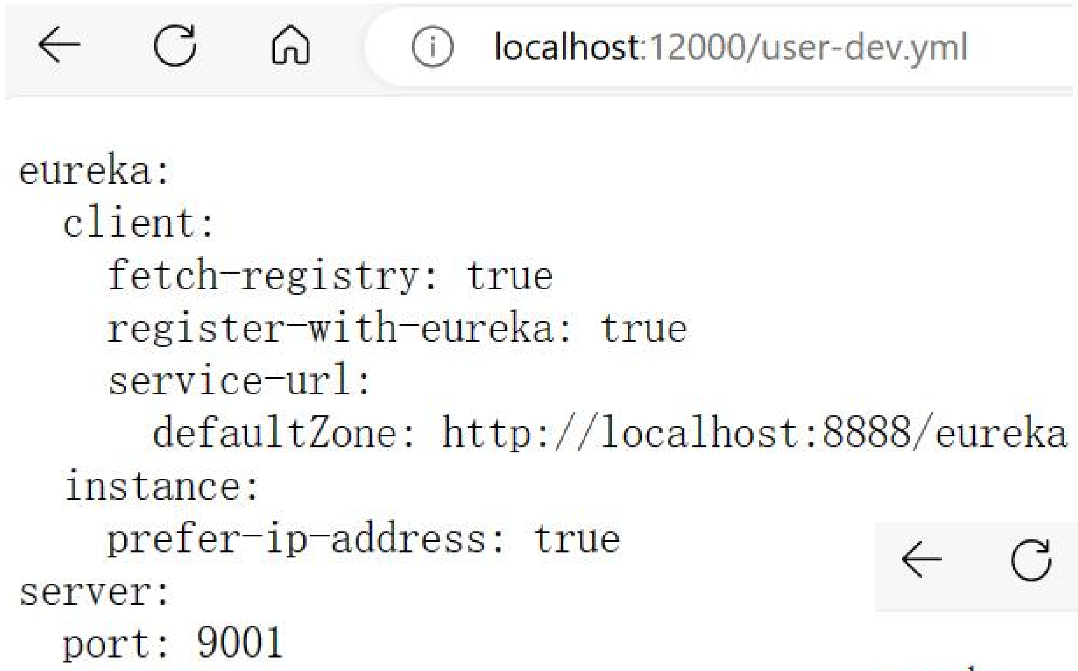
https://github.com/vickitf/springcloud.git

• 编写config项目的引导类



```java
/**
 * 配置中心服务端
 */
@SpringBootApplication
@EnableConfigServer // 开启配置中心服务端功能
public class ConfigApplication {
    public static void main(String[] args) {
        SpringApplication.run(ConfigApplication.class,args);
    }
}
```

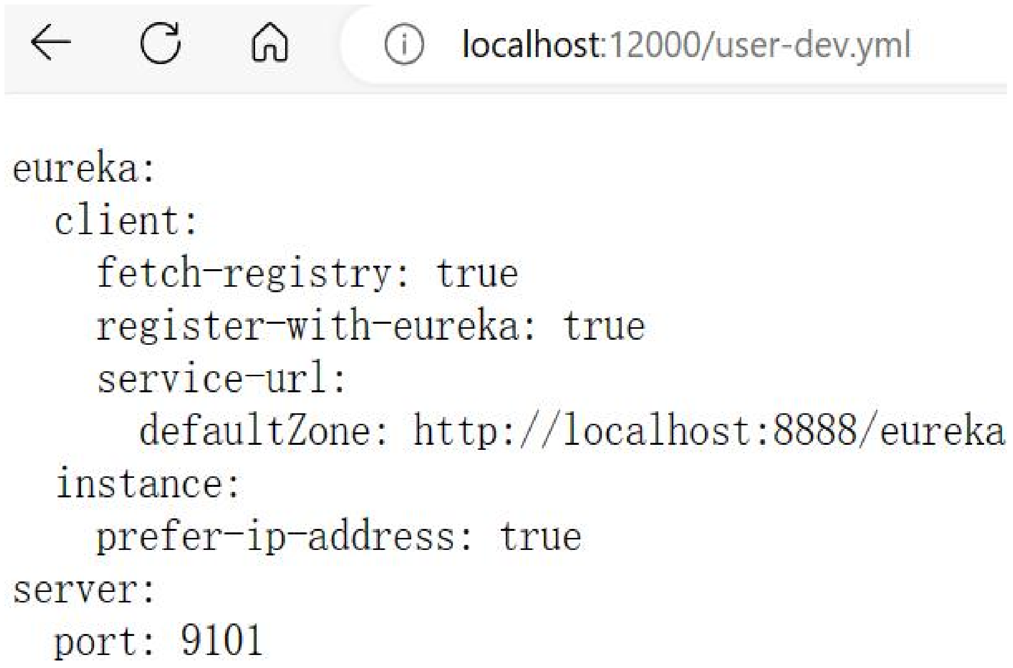
- **测试服务端是否能连上仓库**



localhost:12000/user-dev.yml

```
eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: http://localhost:8888/eureka
  instance:
    prefer-ip-address: true
server:
  port: 9001
```

- **仓库中port改为9101**

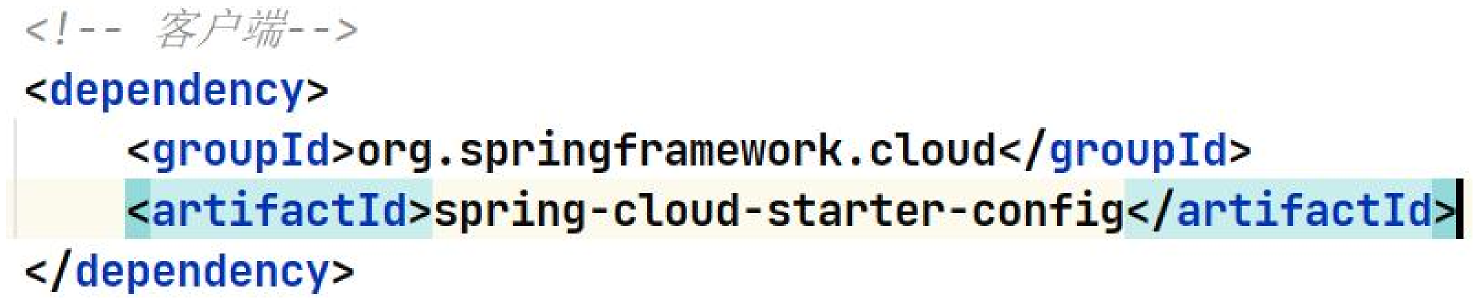


localhost:12000/user-dev.yml

```
eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: http://localhost:8888/eureka
  instance:
    prefer-ip-address: true
server:
  port: 9101
```

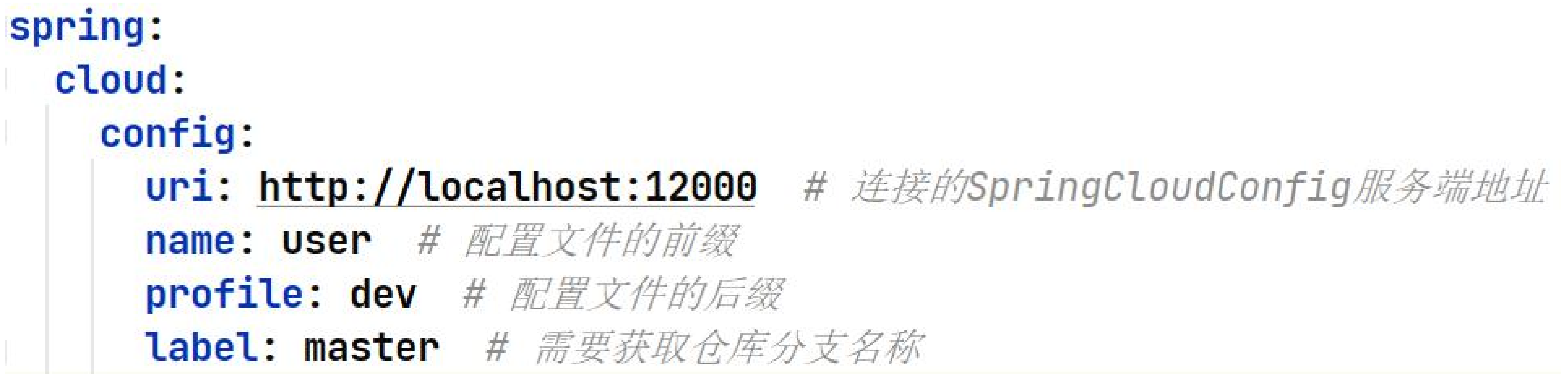
• 第三步，搭建Spring Cloud Config客户端

- 首先，导入客户端依赖



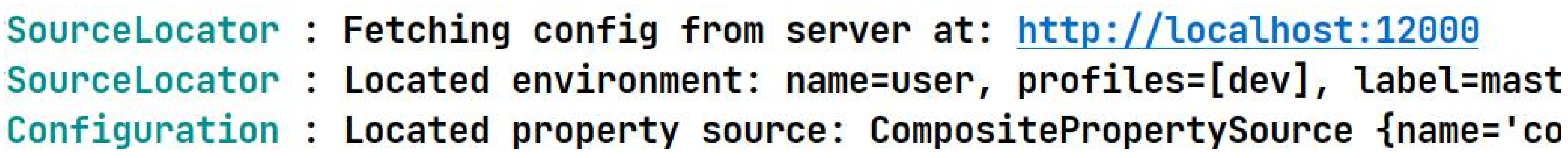
```xml
<!-- 客户端-->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
</dependency>
```

- resources包中建bootstrap.yml引导配置文件



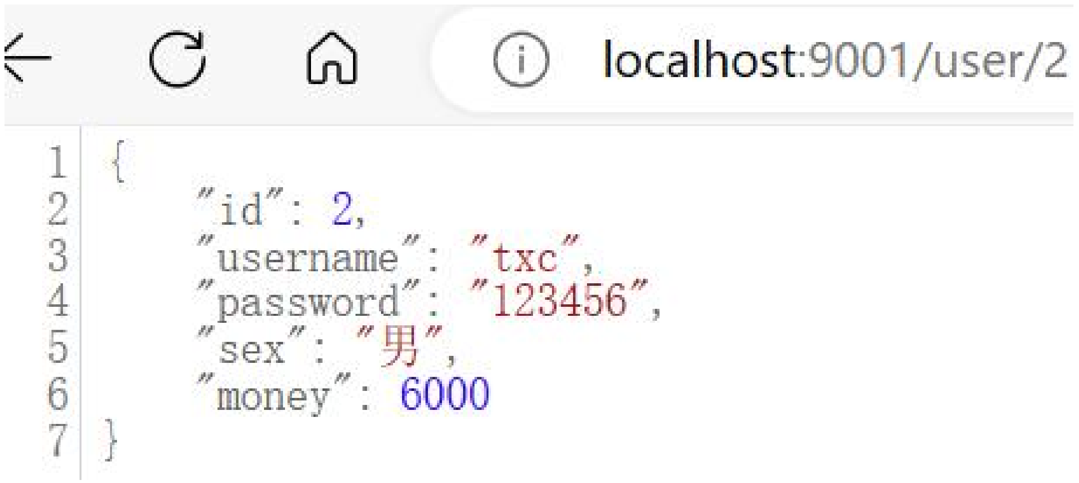
```yaml
spring:
  cloud:
    config:
      uri: http://localhost:12000  # 连接的SpringCloudConfig服务端地址
      name: user  # 配置文件的前缀
      profile: dev  # 配置文件的后缀
      label: master  # 需要获取仓库分支名称
```

- 启动用户微服务



```
SourceLocator : Fetching config from server at: http://localhost:12000
SourceLocator : Located environment: name=user, profiles=[dev], label=mast
Configuration : Located property source: CompositePropertySource {name='co
```

- 浏览器中访问用户微服务

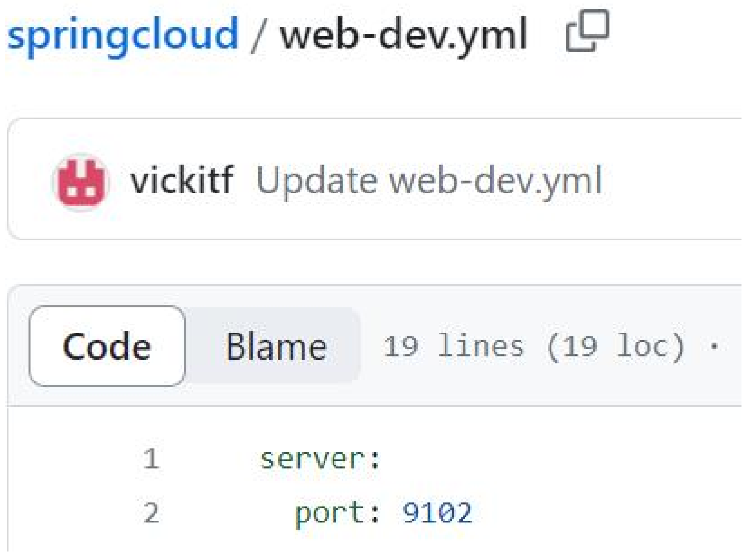


localhost:9001/user/2

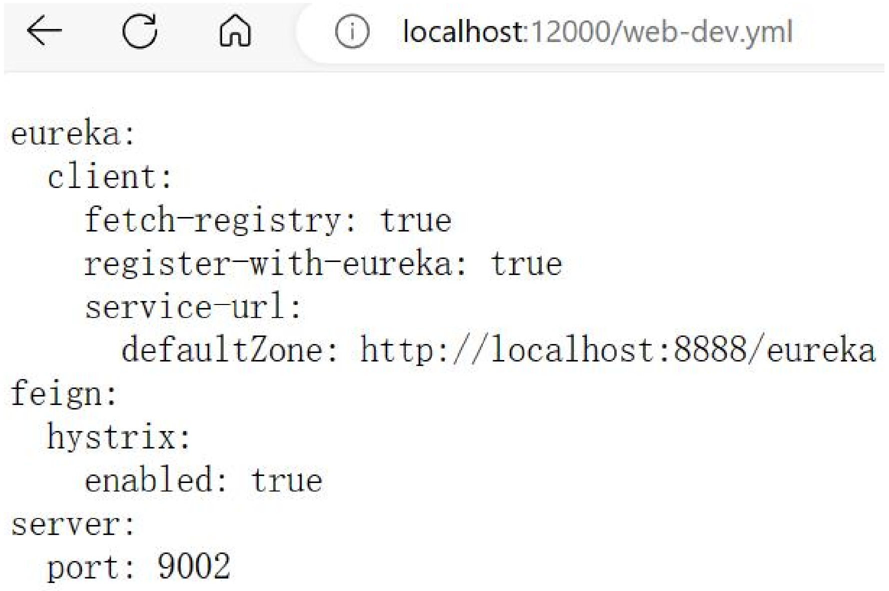
```
{
    "id": 2,
    "username": "txc",
    "password": "123456",
    "sex": "男",
    "money": 6000
}
```

- 同理改造购票微服务为config客户端

- **注意：仓库中配置文件的更改不是立即生效**



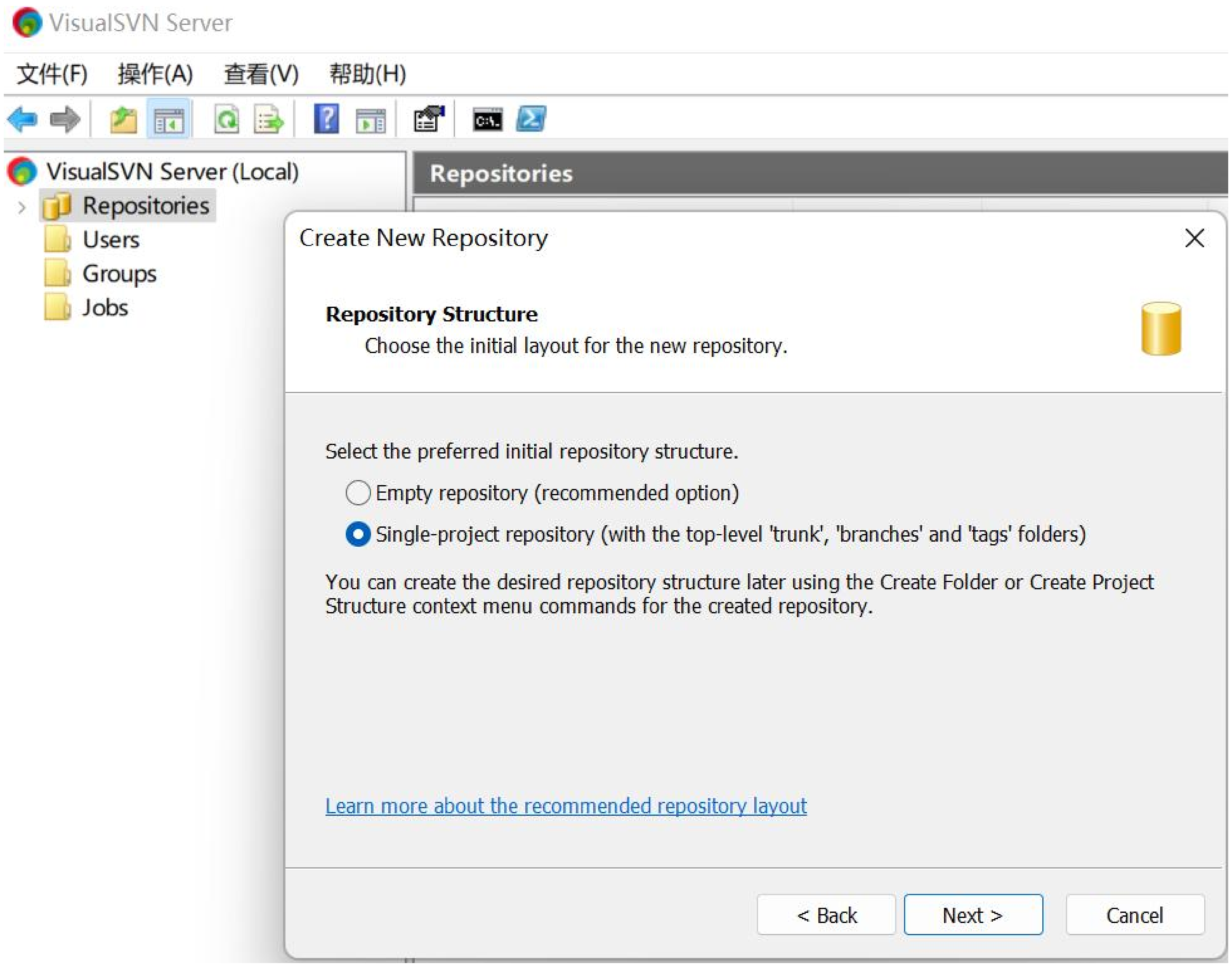
```
server:
  port: 9102
```



```
eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: http://localhost:8888/eureka
feign:
  hystrix:
    enabled: true
server:
  port: 9002
```

而是需要重启该微服务后生效

Spring Cloud Config-SVN方式



Create New Repository

Repository Structure

Choose the initial layout for the new repository.

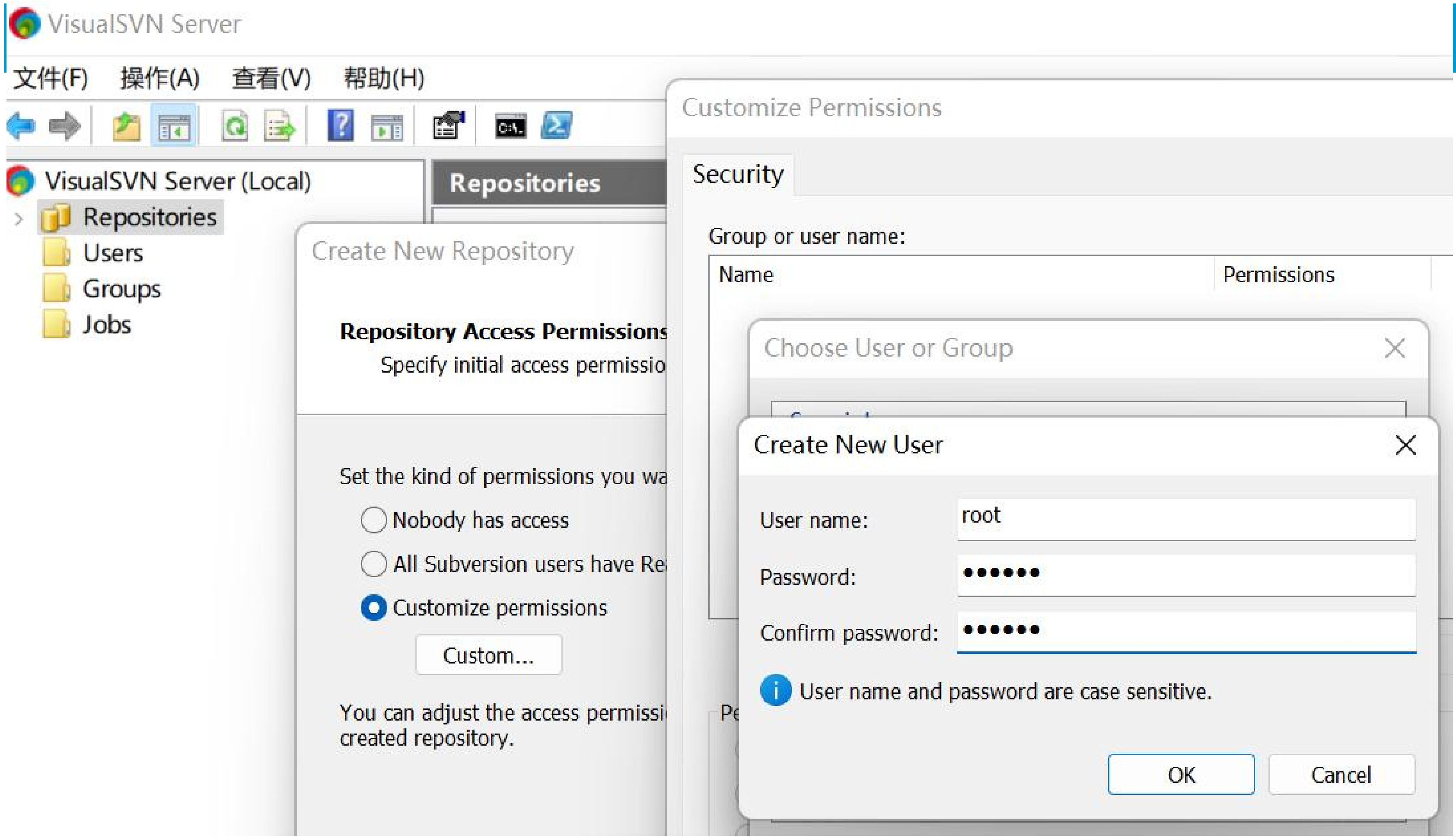
Select the preferred initial repository structure.

Empty repository (recommended option)

[✓ Single-project repository (with the top-level 'trunk', 'branches' and 'tags' folders)]

You can create the desired repository structure later using the Create Folder or Create Project Structure context menu commands for the created repository.

Learn more about the recommended repository layout



VisualSVN Server

文件(F) 操作(A) 查看(V) 帮助(H)

VisualSVN Server (Local)
- Repositories
- Users
- Groups
- Jobs

Repositories

Create New Repository

Repository Access Permissions
Specify initial access permissio

Set the kind of permissions you wa
- ( ) Nobody has access
- ( ) All Subversion users have Re
- [✓ Customize permissions]

Custom...

You can adjust the access permissi
created repository.

Customize Permissions

Security

Group or user name:

Name	Permissions

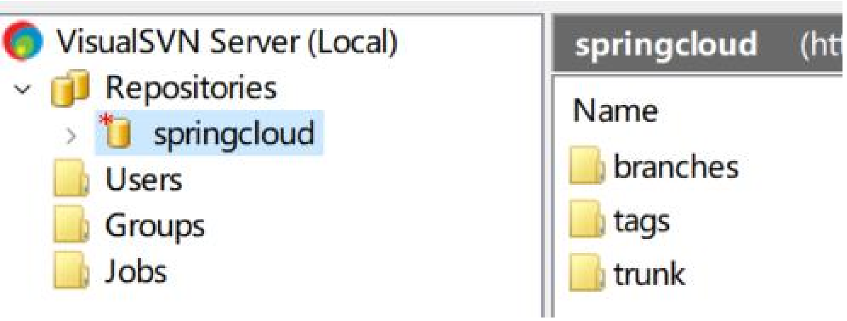
Choose User or Group

Create New User

User name: root
Password: ••••••
Confirm password: ••••••

User name and password are case sensitive.

OK Cancel

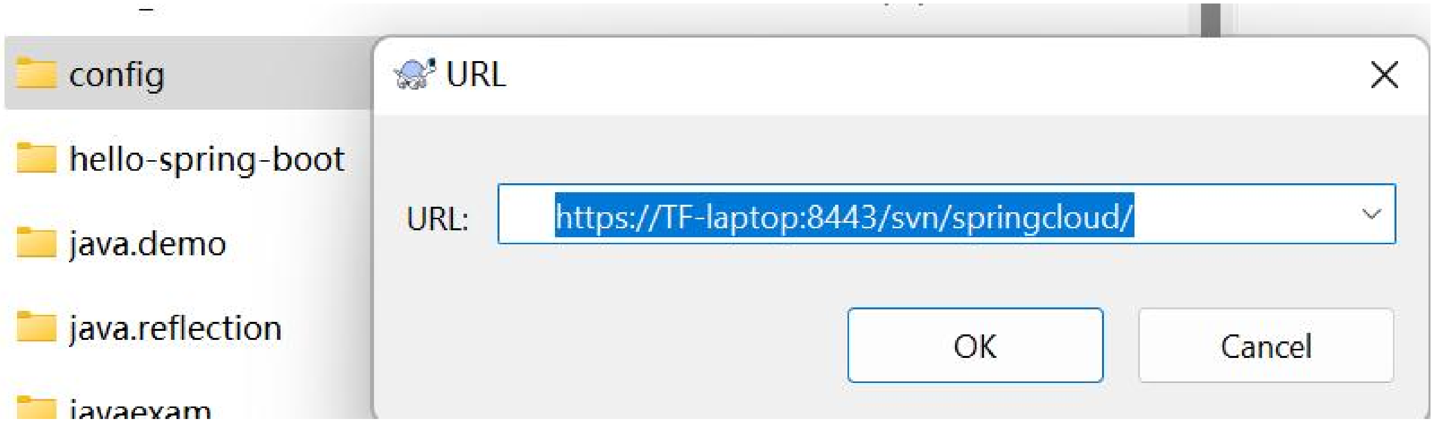


VisualSVN Server (Local)
- Repositories
  - springcloud
- Users
- Groups
- Jobs

springcloud (ht

Name
- branches
- tags
- trunk

使用TortoiseSVN将配置文件上传到仓库

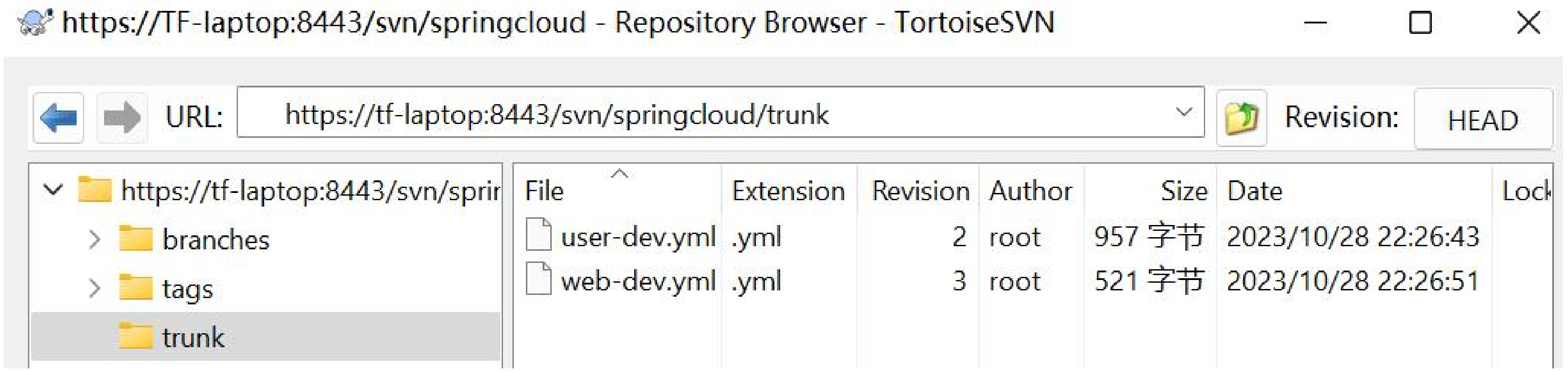


config
hello-spring-boot
java.demo
java.reflection
javaexam

URL

URL: https://TF-laptop:8443/svn/springcloud/

OK Cancel



https://TF-laptop:8443/svn/springcloud - Repository Browser - TortoiseSVN

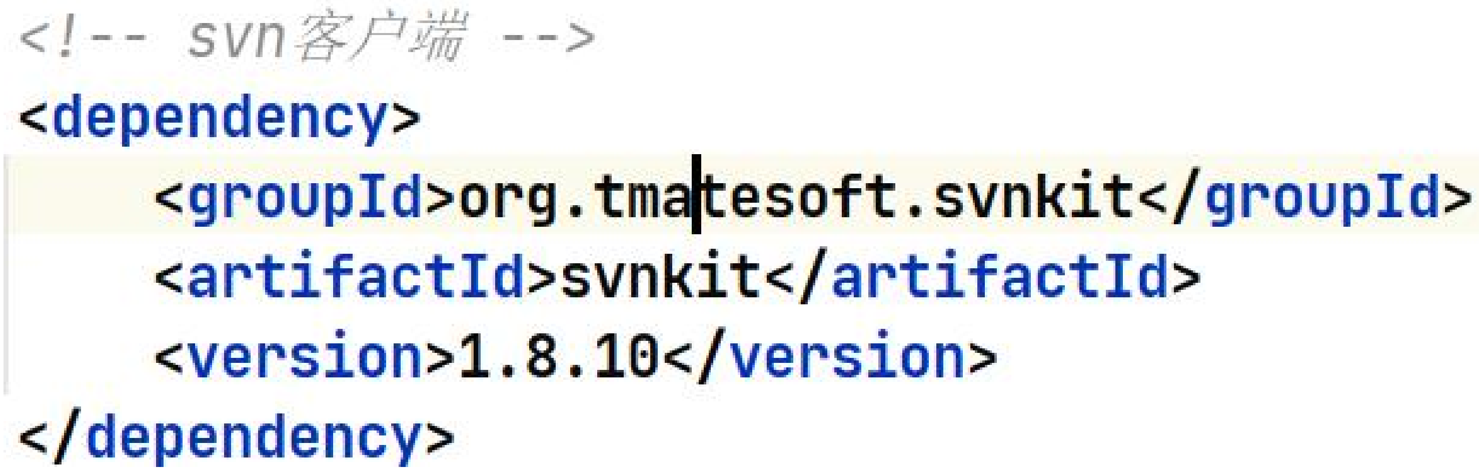
URL: https://tf-laptop:8443/svn/springcloud/trunk Revision: HEAD

https://tf-laptop:8443/svn/sprir
- branches
- tags
- trunk

File	Extension	Revision	Author	Size	Date	Lock
user-dev.yml	.yml	2	root	957 字节	2023/10/28 22:26:43	
web-dev.yml	.yml	3	root	521 字节	2023/10/28 22:26:51	

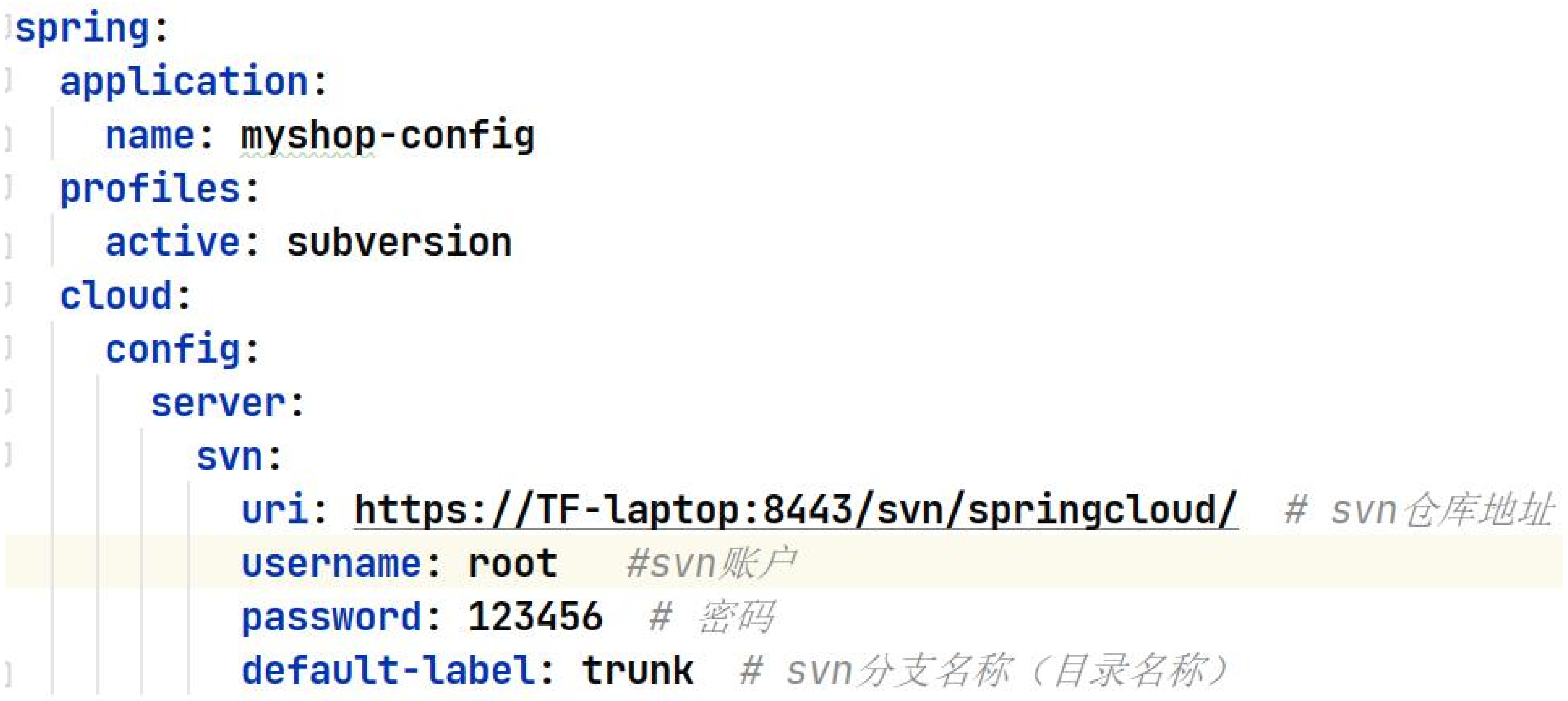
Spring Cloud Config服务端

- 导入SVN依赖



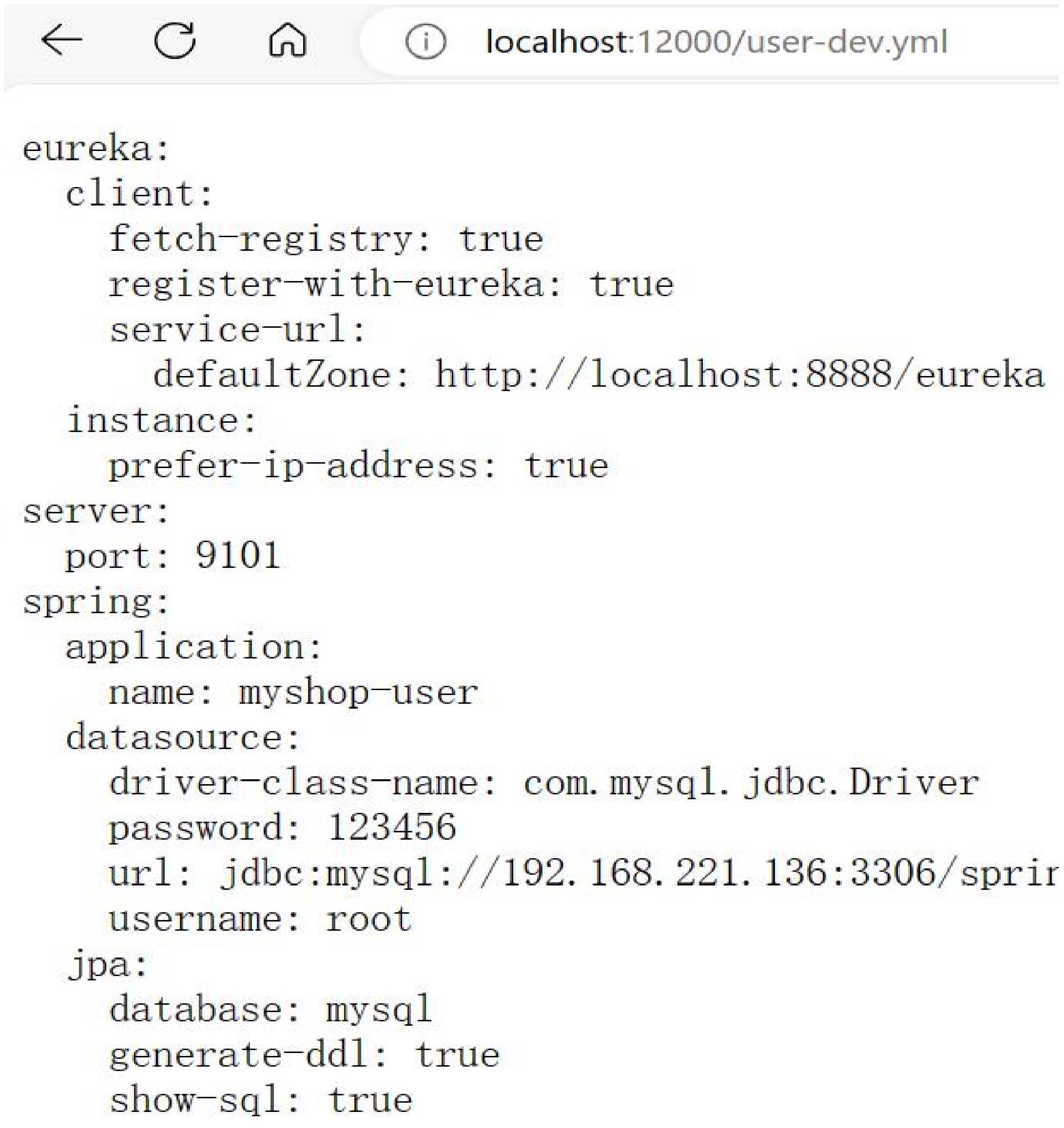
```xml
<!-- svn客户端 -->
<dependency>
    <groupId>org.tmatesoft.svnkit</groupId>
    <artifactId>svnkit</artifactId>
    <version>1.8.10</version>
</dependency>
```

• 修改config服务端配置文件



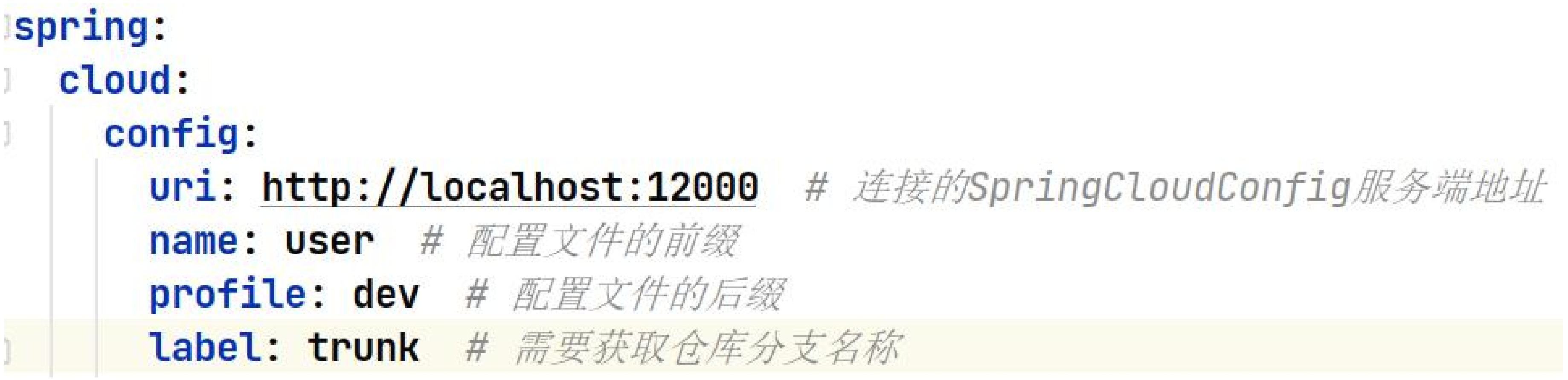
```yaml
spring:
  application:
    name: myshop-config
  profiles:
    active: subversion
  cloud:
    config:
      server:
        svn:
          uri: https://TF-laptop:8443/svn/springcloud/   # svn仓库地址
          username: root    #svn账户
          password: 123456  # 密码
          default-label: trunk  # svn分支名称（目录名称）
```

- 重启config，测试服务端是否能连上仓库

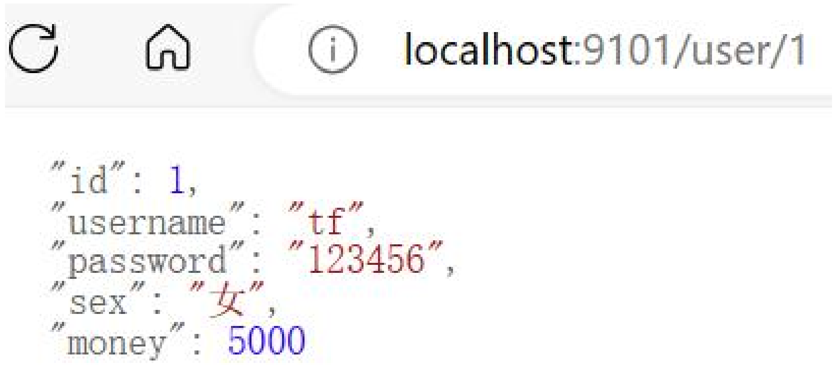

localhost:12000/user-dev.yml

```
eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: http://localhost:8888/eureka
  instance:
    prefer-ip-address: true
server:
  port: 9101
spring:
  application:
    name: myshop-user
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    password: 123456
    url: jdbc:mysql://192.168.221.136:3306/sprir
    username: root
  jpa:
    database: mysql
    generate-ddl: true
    show-sql: true
```

- **修改微服务端bootstrap.yml，并重启验证**



```yaml
spring:
  cloud:
    config:
      uri: http://localhost:12000  # 连接的SpringCloudConfig服务端地址
      name: user  # 配置文件的前缀
      profile: dev  # 配置文件的后缀
      label: trunk  # 需要获取仓库分支名称
```



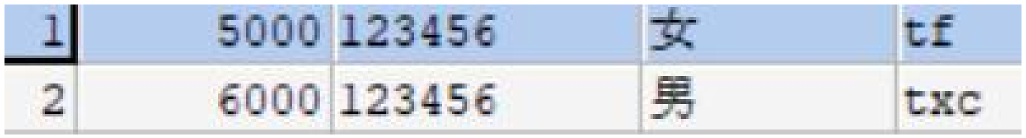
localhost:9101/user/1

```
"id": 1,
"username": "tf",
"password": "123456",
"sex": "女",
"money": 5000
```

Spring Cloud Bus消息总线

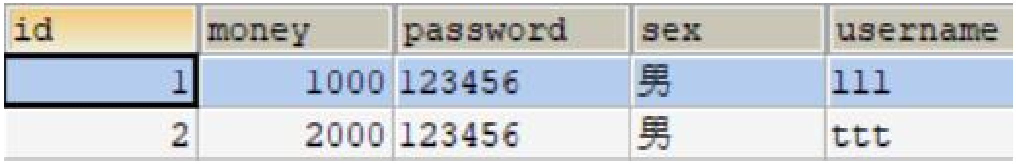
- 配置实时更新
- 更新数据库

```
server:
  port: 9001
spring:   # 服务名称，暂时没有用，讲到SpringCloud服务调用的时候才会有用。
  application:
    name: myshop-user
  datasource:
    url: jdbc:mysql://192.168.221.136:3306/springcloud?characterEncoding=UTF8&useSSL=false
```



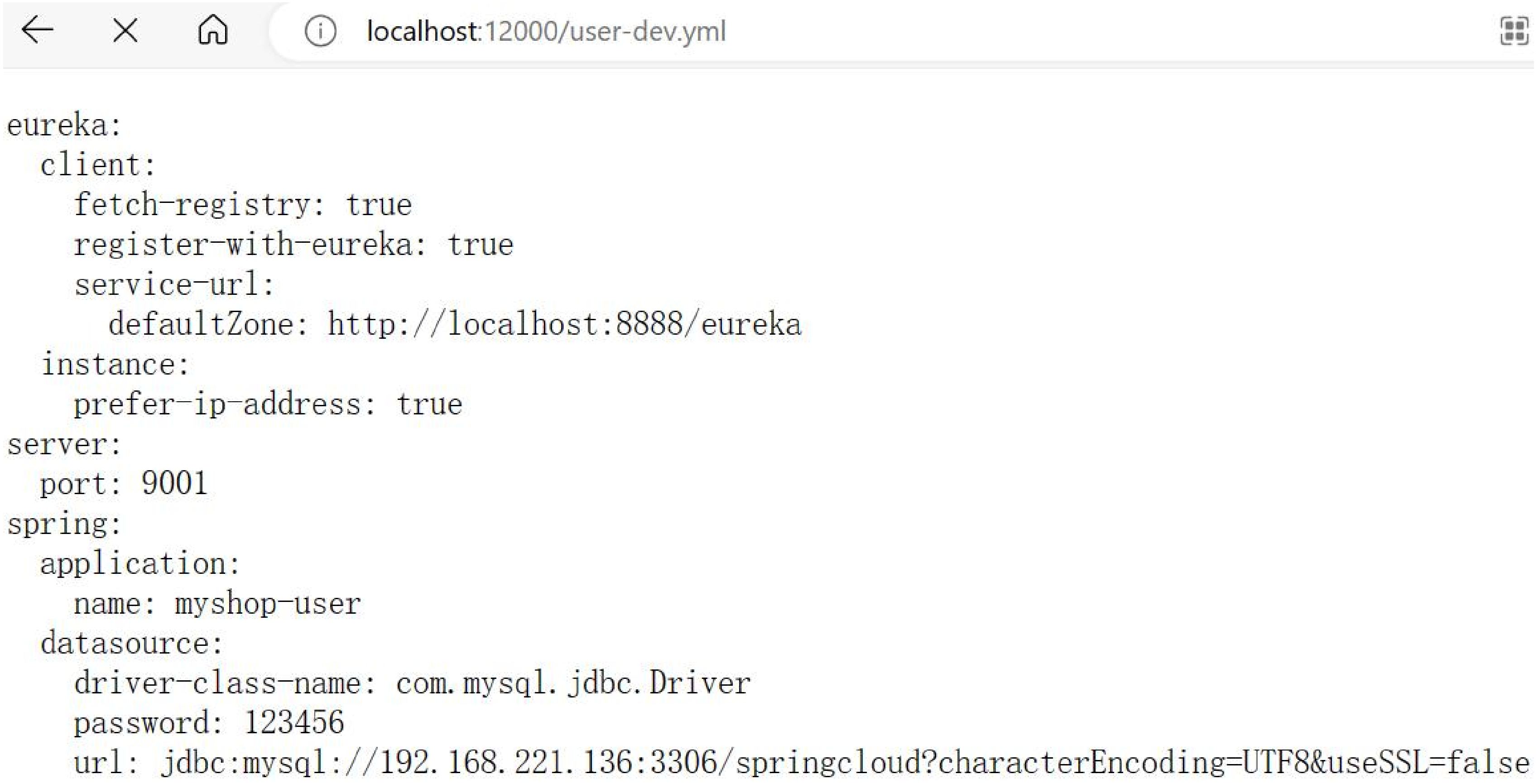
1	5000	123456	女	tf
2	6000	123456	男	txc

```
server:
  port: 9001
spring:   # 服务名称，暂时没有用，讲到SpringCloud服务调用的时候才会有用。
  application:
    name: myshop-user
  datasource:
    url: jdbc:mysql://192.168.221.136:3307/springcloud?characterEncoding=UTF8&useSSL=false
```



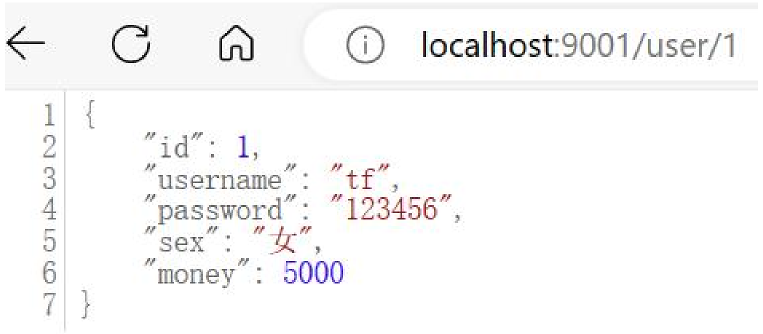
id	money	password	sex	username
1	1000	123456	男	111
2	2000	123456	男	ttt

- 仓库中修改配置后，通过服务端查看该配置文件是否修改生效



```yaml
eureka:
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      defaultZone: http://localhost:8888/eureka
  instance:
    prefer-ip-address: true
server:
  port: 9001
spring:
  application:
    name: myshop-user
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    password: 123456
    url: jdbc:mysql://192.168.221.136:3306/springcloud?characterEncoding=UTF8&useSSL=false
```

- 访问用户微服务，拿到的仍然是3306数据库的数据
- 重启用户微服务后才能拿到3307数据库的数据



```json
{
    "id": 1,
    "username": "tf",
    "password": "123456",
    "sex": "女",
    "money": 5000
}
```

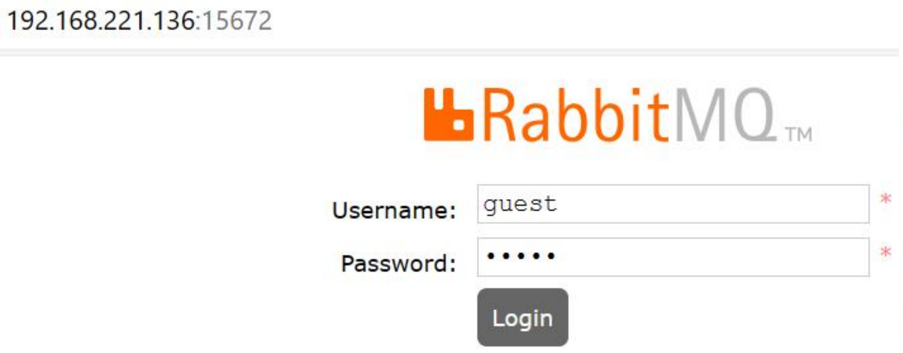
Spring Cloud Bus消息总线

- docker中创建rabbitmq容器

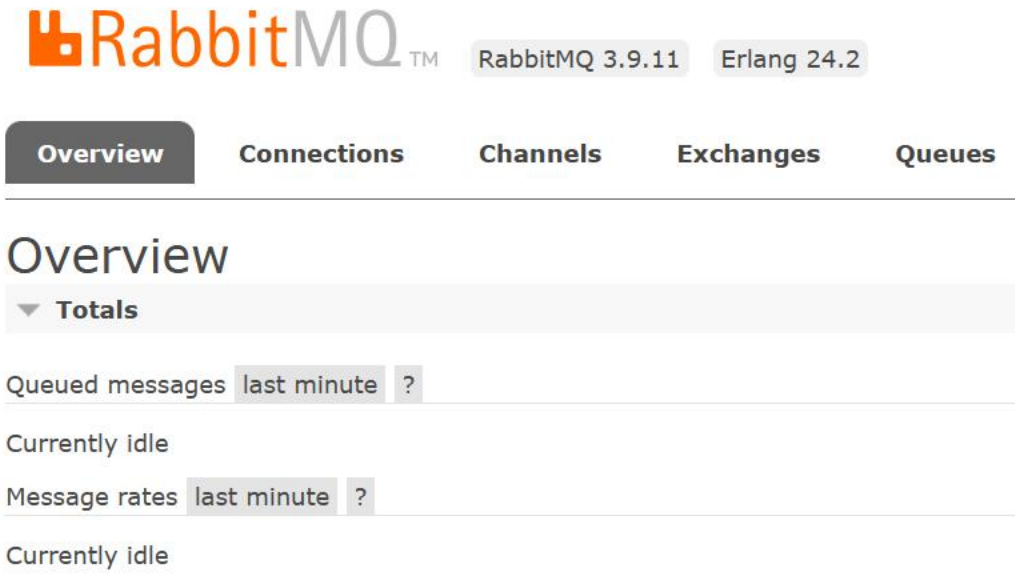
```
# docker pull rabbitmq:management
```

```
root@docker:/home/vicki# docker run -di --name=rabbitmq -p 5671:5671 -p 5672:5
672 -p 15671:15671 -p 15672:15672 -p 4369:4369 -p 25672:25672 rabbitmq:managem
ent
58a906b374d1bc017d3dfb473a05f615435be1ca8023d72b8f466405618ec6dc
```

- 访问rabbitmq


192.168.221.136:15672

Username: guest
Password: •••••

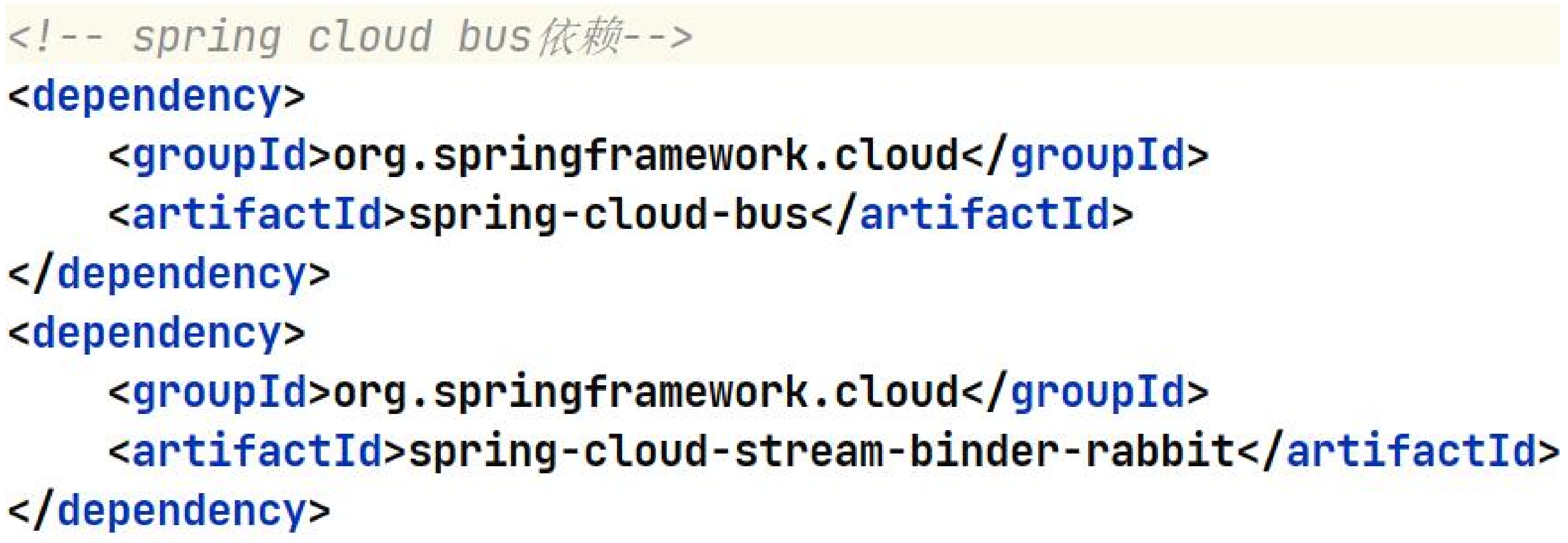


Queued messages last minute ?
Currently idle
Message rates last minute ?
Currently idle

Spring Cloud Bus消息总线

- Spring Cloud Bus服务端（spring cloud config中）

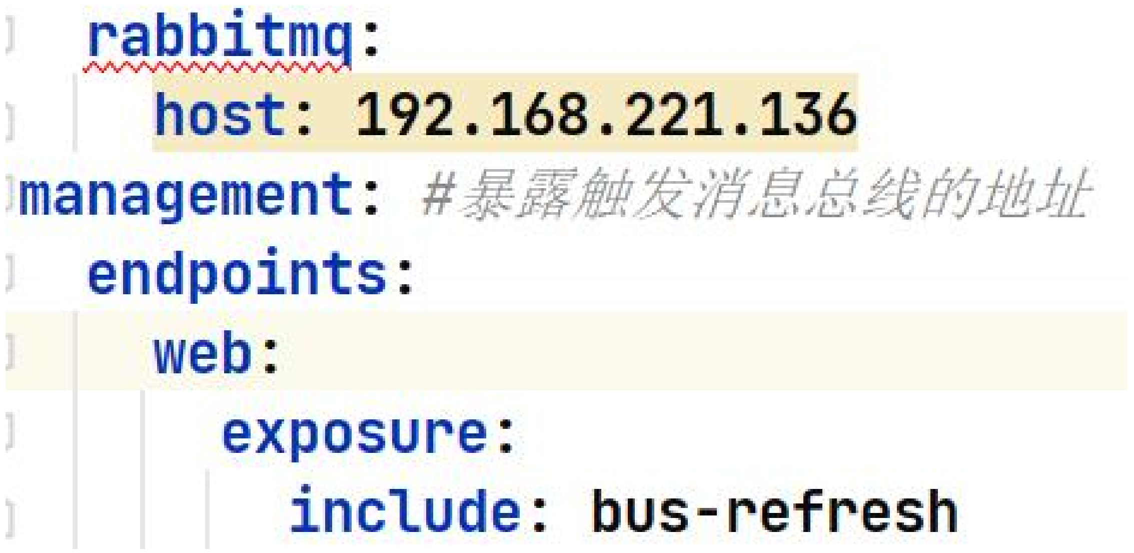
导入依赖



```xml
<!-- 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>
```

Spring Cloud Bus消息总线

配置文件中暴露springcloudbus的触发地址



```yaml
  rabbitmq:
    host: 192.168.221.136
management: #暴露触发消息总线的地址
  endpoints:
    web:
      exposure:
        include: bus-refresh
```

- Spring Cloud Bus客户端（用户微服务）

导入依赖

```xml
<!-- 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>
```

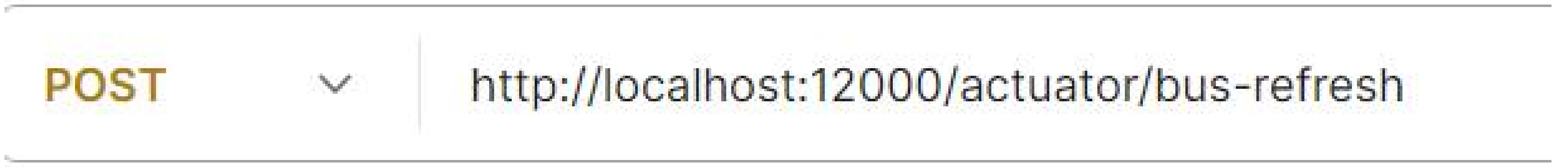
配置文件中配置rabbitmq地址

```yaml
rabbitmq:
  host: 192.168.221.136
```

- 配置文件中将数据库切换为3307

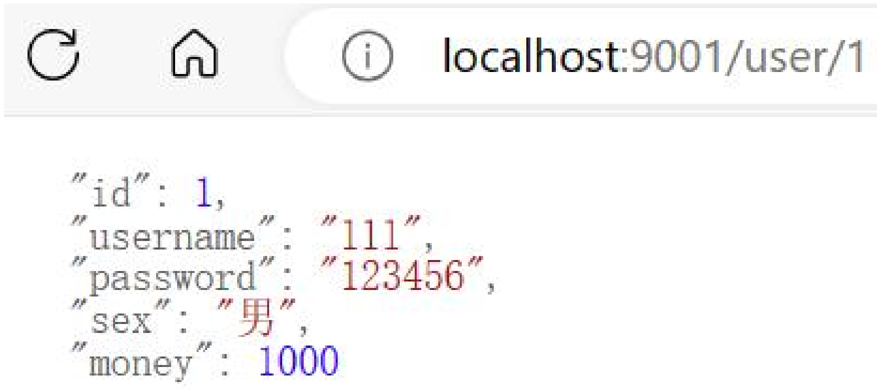
```
datasource:
  url: jdbc:mysql://192.168.221.136:3307/springcloud?characterEncoding=UTF8&useSSL=false
  driver-class-name: com.mysql.jdbc.Driver
```

- 触发地址更新



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

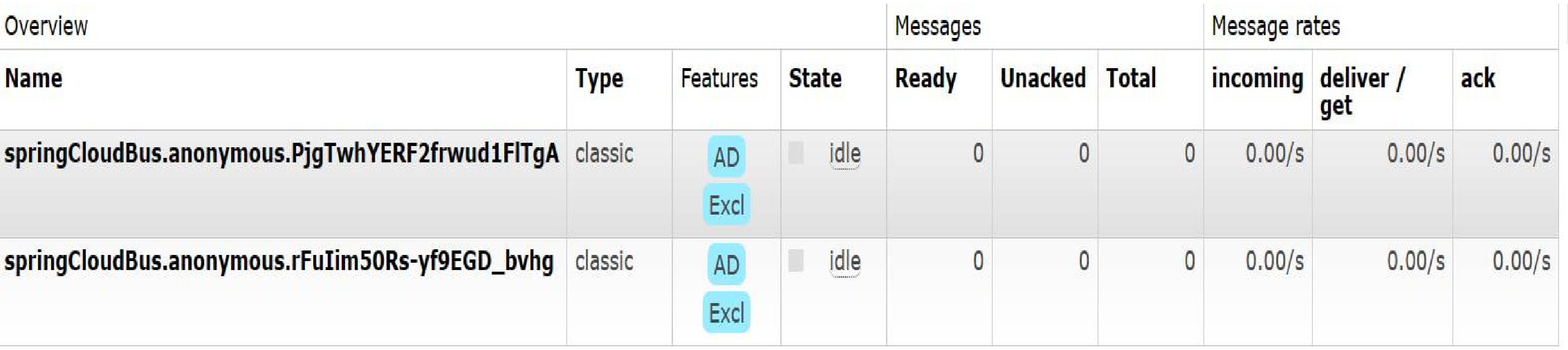
- 无需重启微服务，数据库切换到3307



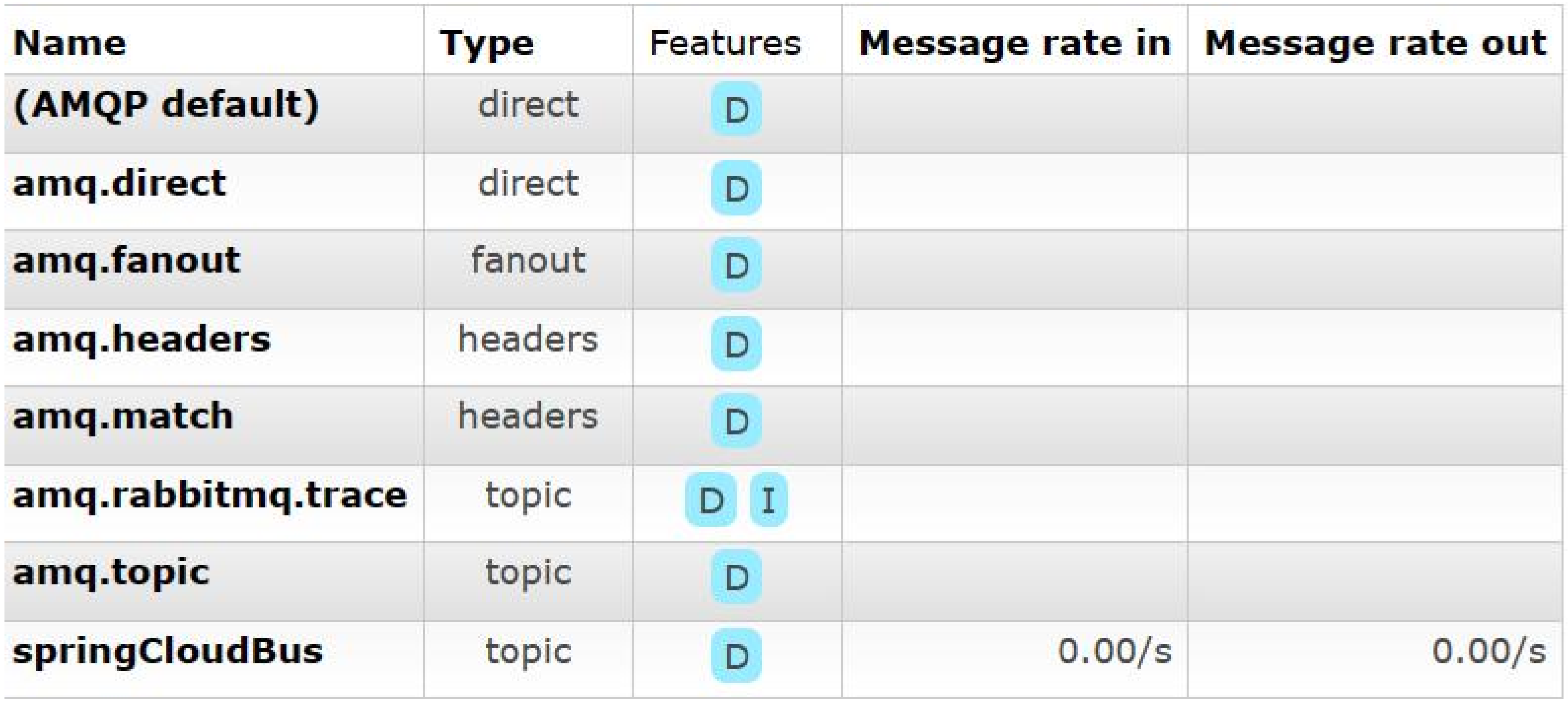
```
localhost:9001/user/1

"id": 1,
"username": "111",
"password": "123456",
"sex": "男",
"money": 1000
```

- rabbitmq中queues和Exchanges



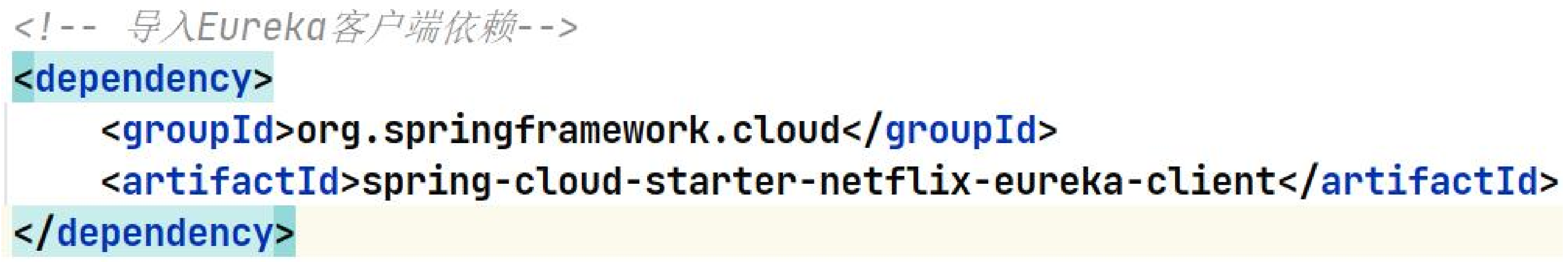
Overview				Messages			Message rates		
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
springCloudBus.anonymous.PjgTwhYERF2frwud1FlTgA	classic	AD Excl	idle	0	0	0	0.00/s	0.00/s	0.00/s
springCloudBus.anonymous.rFuIim50Rs-yf9EGD_bvhg	classic	AD Excl	idle	0	0	0	0.00/s	0.00/s	0.00/s



Name	Type	Features	Message rate in	Message rate out
(AMQP default)	direct	D		
amq.direct	direct	D		
amq.fanout	fanout	D		
amq.headers	headers	D		
amq.match	headers	D		
amq.rabbitmq.trace	topic	D I		
amq.topic	topic	D		
springCloudBus	topic	D	0.00/s	0.00/s

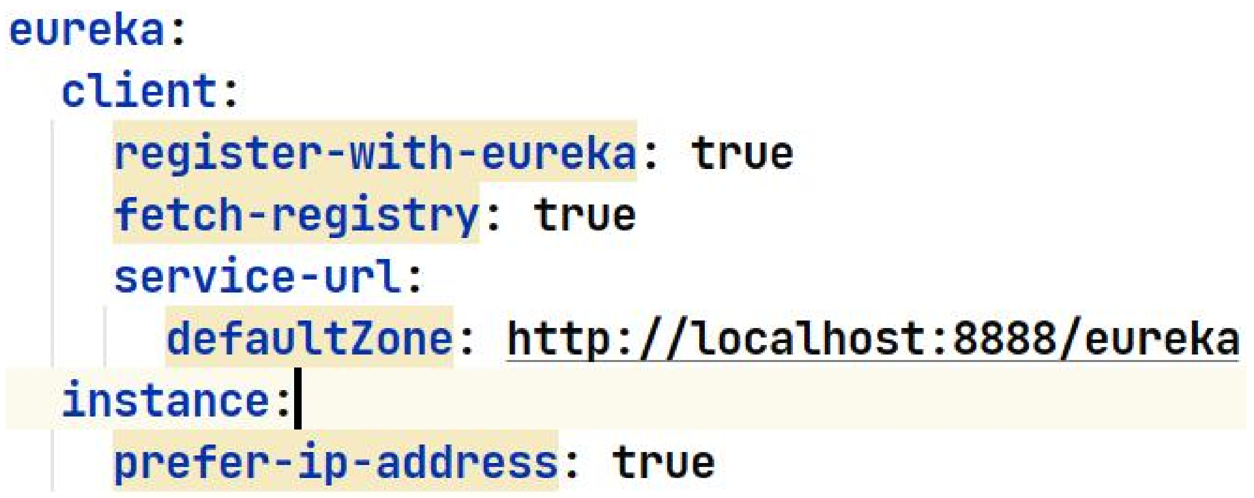
高可用配置中心

- 导入Eureka客户端依赖



```xml
<!--  导入Eureka客户端依赖-->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

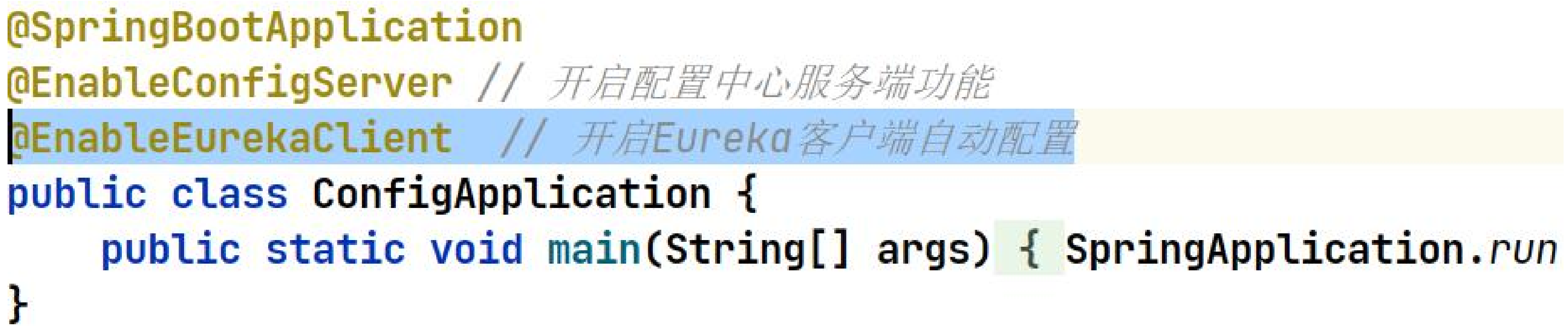
- 写配置文件



```yaml
eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8888/eureka
  instance:
    prefer-ip-address: true
```

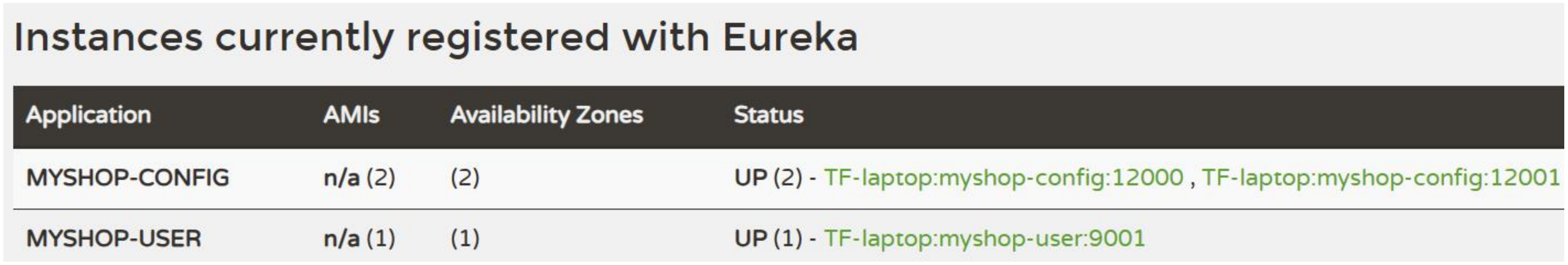
高可用配置中心

- config启动类加@EnableEurekaClient注解



```
@SpringBootApplication
@EnableConfigServer // 开启配置中心服务端功能
@EnableEurekaClient  // 开启Eureka客户端自动配置
public class ConfigApplication {
    public static void main(String[] args) { SpringApplication.run
}
```

- 开启多个配置中心实例

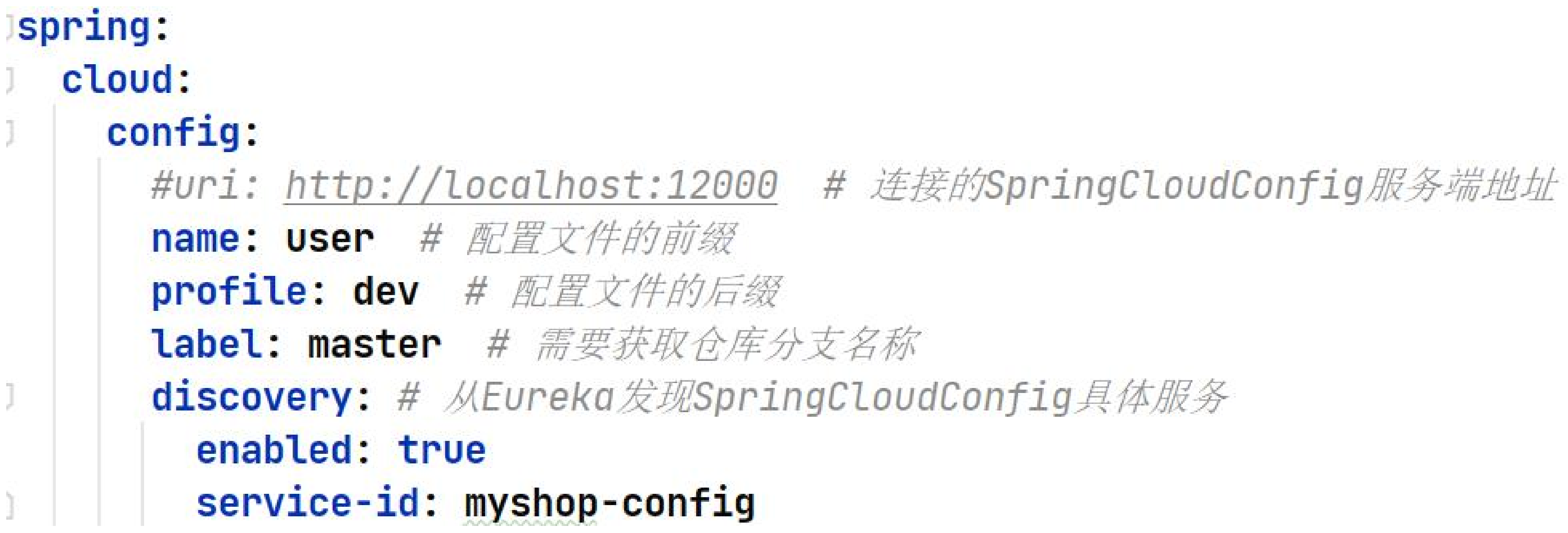


Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
MYSHOP-CONFIG	n/a (2)	(2)	UP (2) - TF-laptop:myshop-config:12000 , TF-laptop:myshop-config:12001
MYSHOP-USER	n/a (1)	(1)	UP (1) - TF-laptop:myshop-user:9001

高可用配置中心

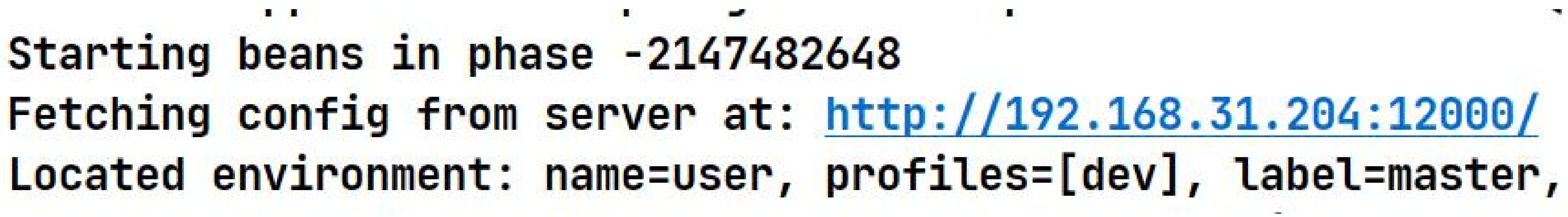
Spring Cloud Config客户端通过服务名连接config服务端



```
spring:
  cloud:
    config:
      #uri: http://localhost:12000  # 连接的SpringCloudConfig服务端地址
      name: user  # 配置文件的前缀
      profile: dev  # 配置文件的后缀
      label: master  # 需要获取仓库分支名称
      discovery: # 从Eureka发现SpringCloudConfig具体服务
        enabled: true
        service-id: myshop-config
```

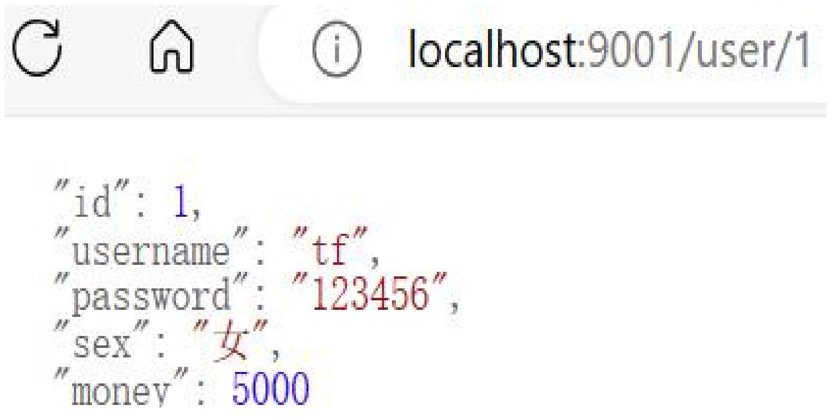
客户端bootstrap.yml中需要包含eureka的配置

启动微服务，在启动信息中可以看到连接的是哪一个config



```
Starting beans in phase -2147482648
Fetching config from server at: http://192.168.31.204:12000/
Located environment: name=user, profiles=[dev], label=master,
```

测试用户微服务可用



localhost:9001/user/1

```
"id": 1,
"username": "tf",
"password": "123456",
"sex": "女",
"money": 5000
```

停止端口为12000的config，用户微服务仍然可用。