CMS第四次设计报告

1. **小组成员**

前端：吴梦凝19301050

后端：吴梦凝19301050

设计：任城仪19301045

1. **概要设计**

本系统基于第三次作业进行迭代，在微服务架构的基础上，添加卡夫卡消息队列，并通过spring cloud config配置中心，将微服务架构的配置文件上传至github进行集中配置。

本系统采用springboot+Mybatis+Spring cloud+openFeign+Eureka+Hystrix

+kafka+spring cloud config的方式进行实现。

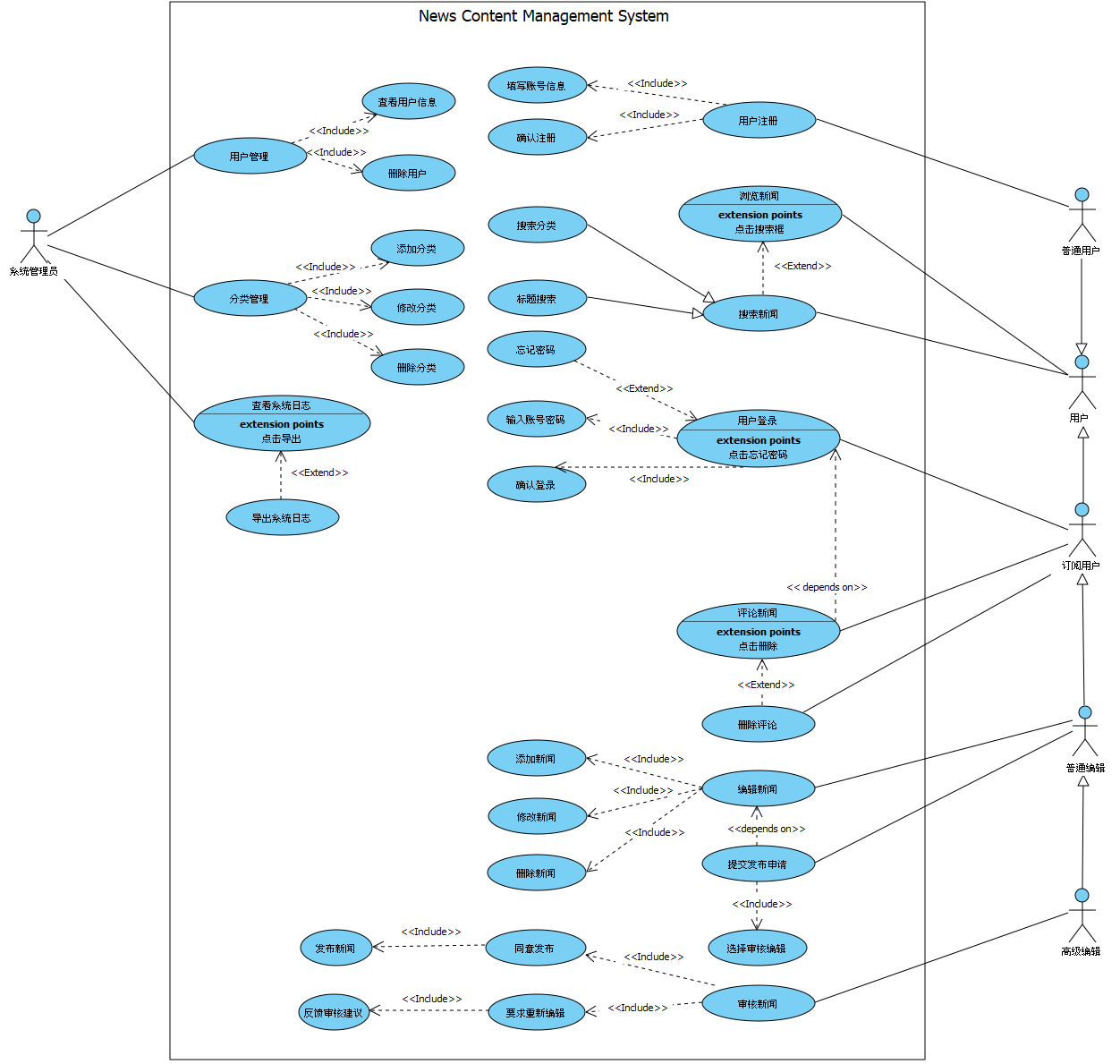
|  |  |
| --- | --- |
| 微服务架构 | 与第三次作业相同 |
| Kafka | 实现分布式分发订阅消息 |
| Zookeeper | 使用zookeeper进行统一管理 |
| Spring cloud config | 将微服务的配置文件上传至远程仓库中 |

1. **详细设计**
   1. **系统功能**

本新闻内容管理系统具有以下功能：

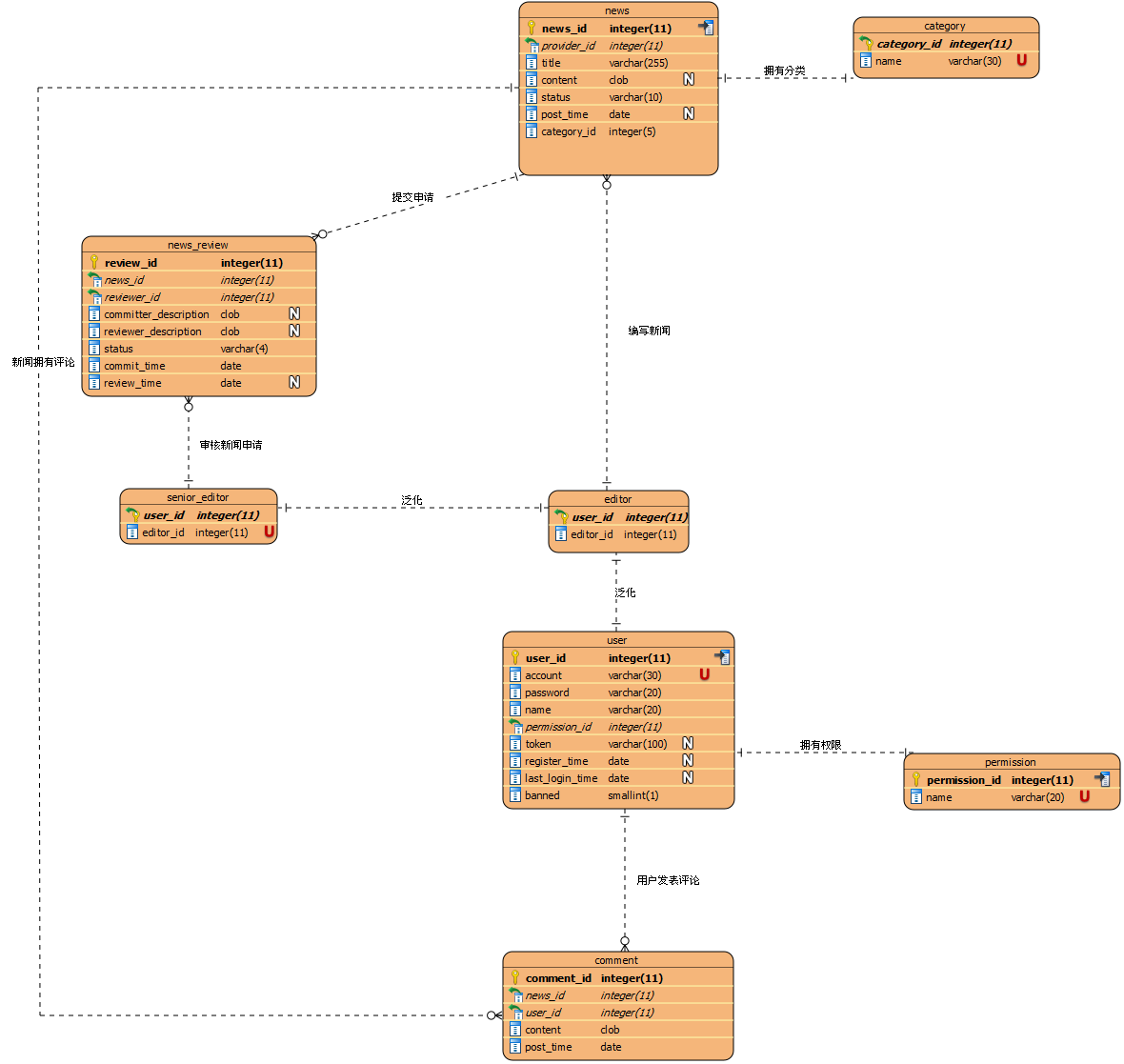
|  |  |
| --- | --- |
| 编号 | 功能描述 |
| 1 | 用户登录注册 |
| 2 | 用户阅览新闻 |
| 3 | 用户搜索新闻 |
| 4 | 用户添加评论信息 |
| 5 | 用户删除评论信息 |
| 6 | 编辑添加新闻 |
| 7 | 编辑提交发布申请 |
| 8 | 编辑管理自己发布的新闻 |
| 9 | 高级编辑审核新闻发布申请 |
| 10 | 高级编辑管理审核列表 |
| 11 | 管理员用户查看用户列表 |
| 12 | 管理员用户查看分类列表 |
| 13 | 管理员用户删除用户账号 |
| 14 | 管理员用户修改分类 |
| 15 | 管理员用户删除分类 |
| 16 | 管理员用户修改用户发言权限 |

具体用例图如下：



* 1. **数据库设计**

本系统主要包括用户，新闻，审核申请，权限和评论这几个实体，ER图如下：



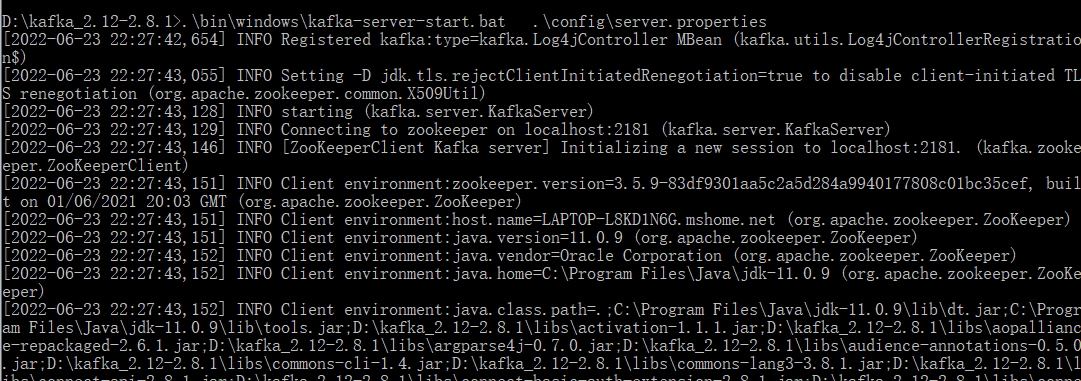
* 1. **Zookeeper分布式应用协调服务**

本系统使用zookeeper管理kafka的集群配置，具体流程如下：

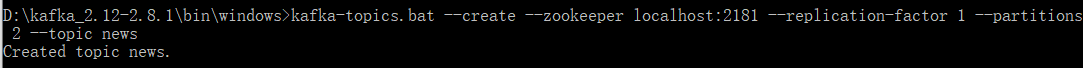
首先，启动zookeeper服务器：



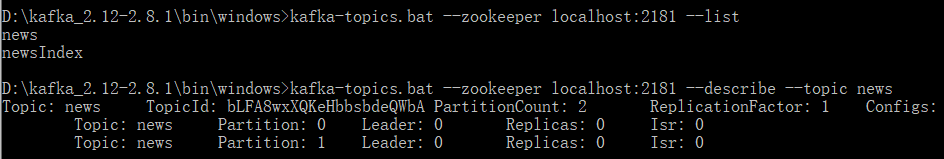
然后，启动kafka服务器：



创建新的topic：

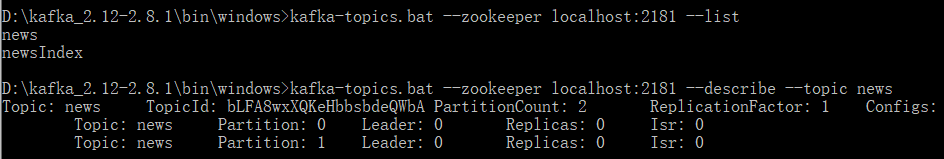


查看创建的topic信息：



* 1. **Kafka消息队列**

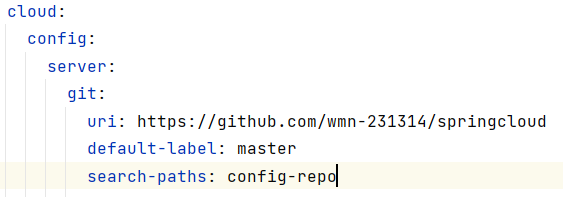
本系统在kafka服务器中注册了一个news的topic，并为其设置了两个partition，用于提高集群的可用性。



在新闻服务中，当前端发送新闻获取请求时，系统会通过kafka template向消息队列发送主题为news，内容为“kafka getting news“。消息到达队列后，将会广播给订阅方，此时订阅方为同微服务器。在controller中，有一个名为listener的函数用于拉取kafka队列发送的消息并打印。

* 1. **Spring cloud Config集中管理配置**

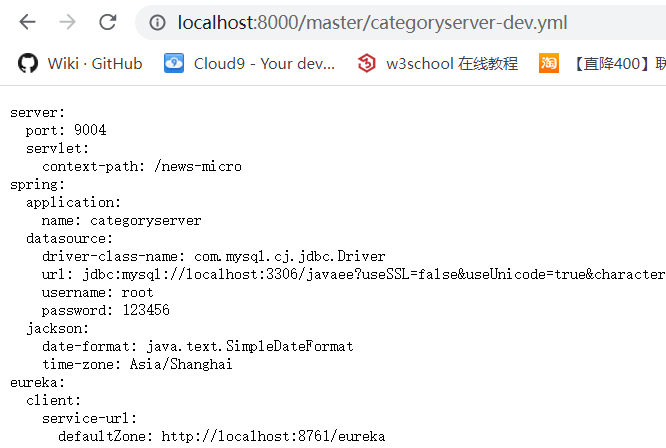
本系统通过github对配置进行集中管理。首先，我在github上创建了一个仓库名为springcloud，然后将其url和对应label配置在config-server中。如图：



然后，我将category-server作为测试config的客户端，将其application.yml删除，并通过配置bootstrap.yml，从github上获取对应的配置文件。如下：



从github获取配置截图如下：8000为config端口



最后，category仍能正常使用：

