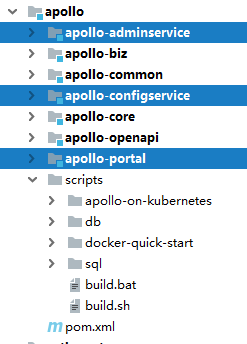
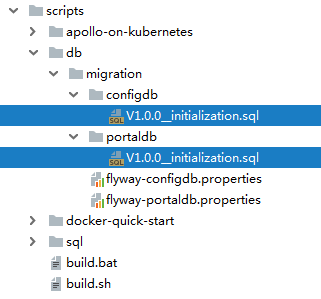
### 一、Apollo代码简介

Apollo代码结构如下所示：



主要功能分为三块：apollo-adminservice/apollo-configservice/apollo-portal，apollo-biz/apollo-common/apollo-core/apollo-openapi为公共包。

scripts为打包编译工具及数据库脚本，需要注意以下两个脚本，这是apollo数据库初始化脚本：



这两个sql脚本会分别建立ApolloConfigDB和ApolloPortalDB两个数据库。

apollo-adminservice：后台服务，真正用于修改工程配置的服务，不同环境都需要部署一套，对应的数据库为ApolloConfigDB

Apollo-configservice：注册中心，用于注册adminservice和portal，不同环境都需要部署一套，对应的数据库为ApolloConfigDB

apollo-portal：统一前台服务，用于显示前台页面及相关配置功能，多个环境只需部署一套，对应的数据库为ApolloPortalDB

（注意：ApolloConfigDB每个环境也都需要部署一套）

### 二、IDE启动Apollo

1. 启动apollo-configservice

通过类com.ctrip.framework.apollo.configservice.ConfigServiceApplication启动，vm配置 为：

-Dserver.port=8091

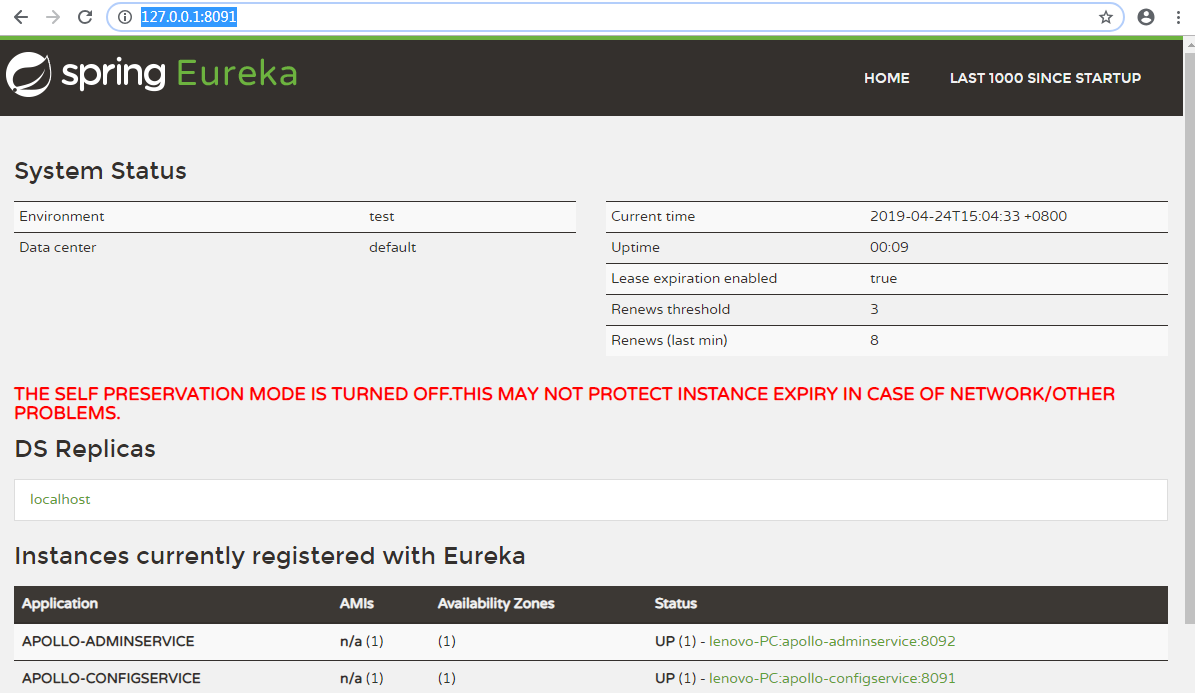
-Dapollo\_profile=github -Dspring.datasource.url=jdbc:mysql://10.192.19.116:3306/ApolloConfigDB?characterEncoding=utf8

-Dspring.datasource.username=root

-Dspring.datasource.password=123456

-Dlogging.file=D:/app/apollo-config.log

启动后进入http://127.0.0.1:8091/ 可以看到



1. 启动apollo-adminservice

通过类com.ctrip.framework.apollo.adminservice.AdminServiceApplication启动，vm配置 为：

-Dserver.port=8092

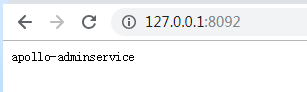
-Dapollo\_profile=github -Dspring.datasource.url=jdbc:mysql://10.192.19.116:3306/ApolloConfigDB?characterEncoding=utf8

-Dspring.datasource.username=root

-Dspring.datasource.password=123456

-Dlogging.file=D:/app/apollo-admin.log

启动后进入http://127.0.0.1:8092/ 可以看到



1. 启动apollo-portal

通过类com.ctrip.framework.apollo.portal.PortalApplication启动，vm配置 为：

-Dapollo\_profile=github,auth

-Ddev\_meta=http://localhost:8091/

-Dserver.port=8090

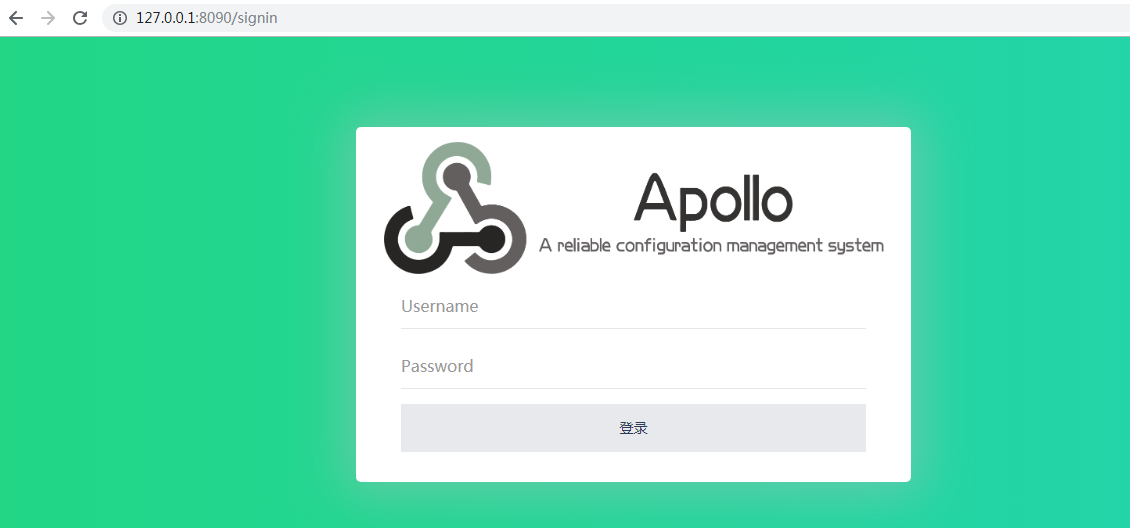
-Dspring.datasource.url=jdbc:mysql://10.192.19.116:3306/ApolloPortalDB?characterEncoding=utf8

-Dspring.datasource.username=root

-Dspring.datasource.password=123456

-Dlogging.file=D:/app/apollo-portal.log

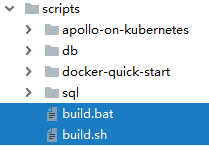
启动后进入http://127.0.0.1:8090/ 可以看到



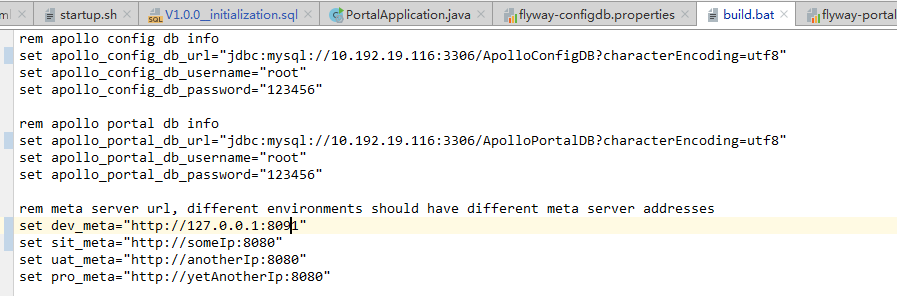
用户名密码为apollo/admin

### 三、打包部署Apollo

1. 打包



修改build.bat或build.sh里的数据库配置及环境配置，环境配置修改为对应环境的apollo-configservice地址，如下所示：



执行build.bat或build.sh进行打包，在三大模块下的target文件夹里能分别拿到一个zip文件。

1. 部署

将上一步中的zip包上传到服务器解压，执行scripts/startup.sh即可。需要注意的是apollo-portal及其连接的数据库ApolloPortalDB在多个环境下只需部署一套，形成统一配置系统。apollo-adminservice/apollo-configservice及其连接的数据库ApolloConfigDB不同的环境都需要部署一套，也就是需要部署多套，apollo-portal和apollo-adminservice/apollo-configservice是一对多的关系。部署策略如下：

1. 读取xml文件

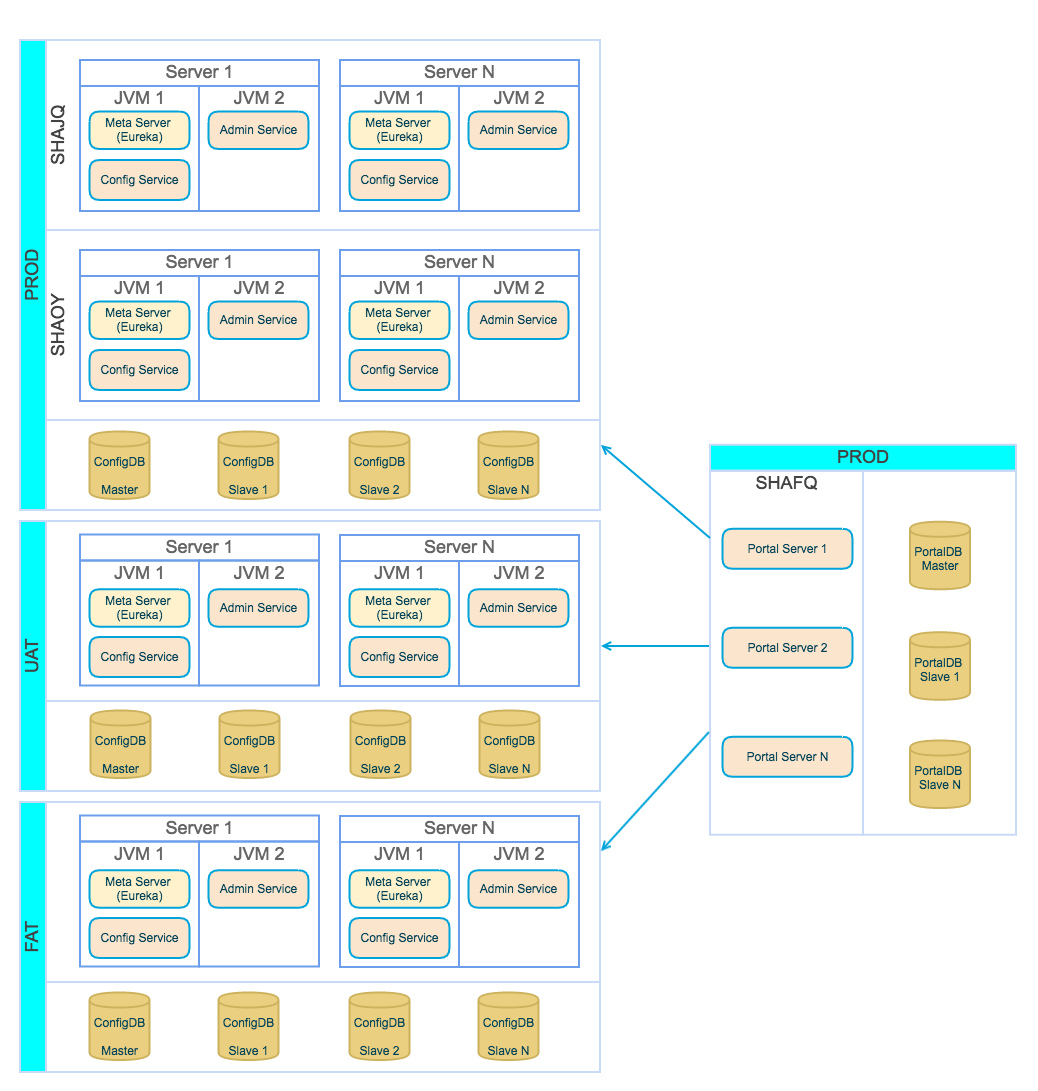
String somePublicNamespace = "nearlySearchMapConfig.xml";

ConfigBean.getName();

ConfigFile configFile = ConfigService.*getConfigFile*(somePublicNamespace, ConfigFileFormat.***XML***);

String content = configFile.getContent();

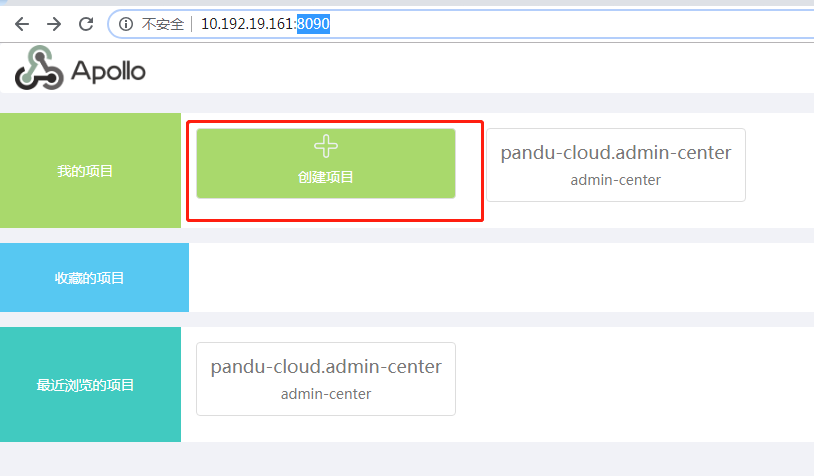
System.***out***.println(content);



### 四、Apollo的使用

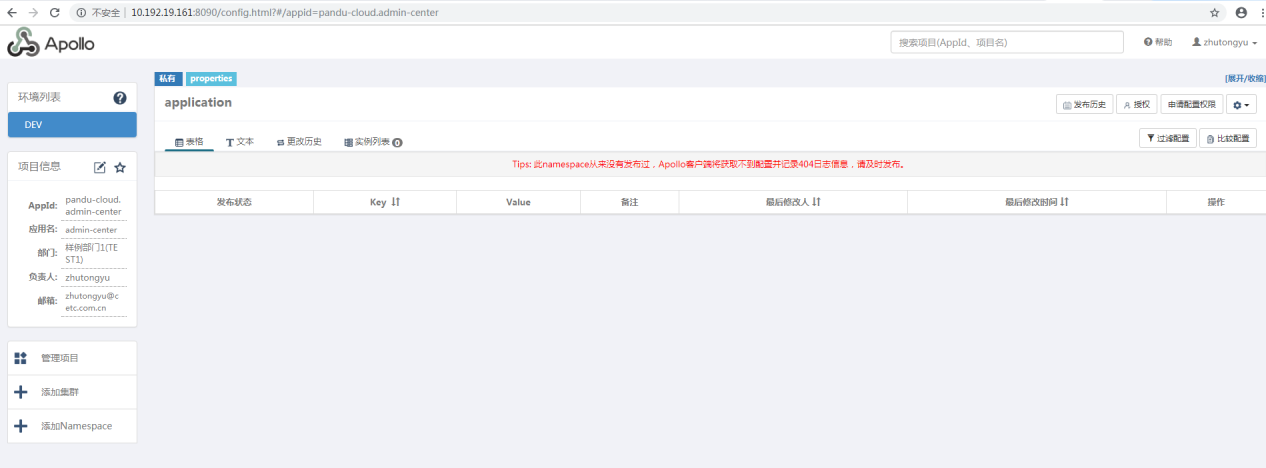
1. Apollo配置

访问apollo-portal，创建项目





应用id必须唯一，并且需要在对应的工程里做配置，例如我在这里配置应用id为pandu-cloud.admin-center并提交，我们就可以对该项目进行配置了：



填写配置后点击发布即可生效

1. 对应项目的配置（只针对springboot项目做说明）

使用apollo配置的相关项目需要引入以下包：

<**dependency**>  
 <**groupId**>com.ctrip.framework.apollo</**groupId**>  
 <**artifactId**>apollo-client</**artifactId**>  
 <**version**>1.3.0</**version**>  
</**dependency**>

并且在application.yml文件中做如下配置：

注意：这里的app.id对应的就是在apollo创建的项目id，namespaces对应apollo里配置的namespaces

**app**:  
 *# 如果有private的命名空间需要配置（默认application空间是private的）* **id**: pandu-cloud.admin-center  
**apollo**:  
 *# 指定apollo-configservice* **meta**: http://10.192.19.161:8091  
 **bootstrap**:  
 **enabled**: true  
 *#对应apollo里配置的namespaces* **namespaces**: base,application

注意：应用支撑项目里application.yml都做了备份，备份文件为application.yml.bak,在不适用apollo时，可删除application.yml将application.yml.bak重命名为application.yml,修改配置时，要同步修改备份文件里的配置，以备不时之需