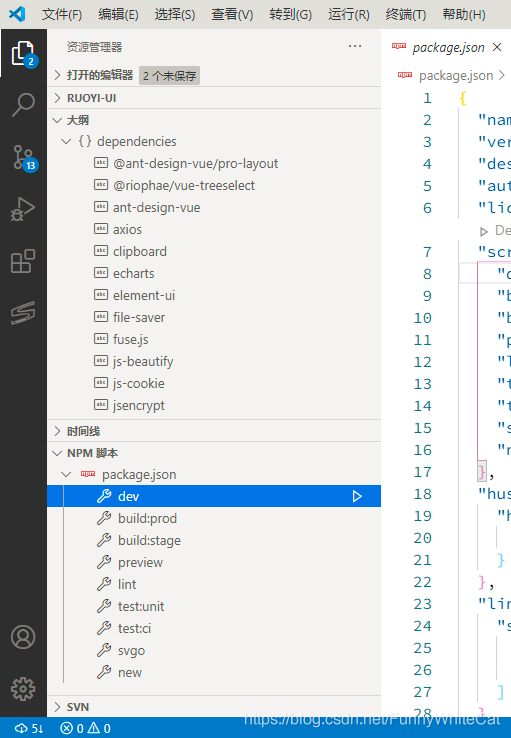
**1.启动**

 按照官网的步骤，

1. 导入项目，创建数据库和表，启动nacos。接下来修改nacos=》配置管理=》配置列表，把里面用到数据库连接的地方全部换成自己的数据库，并修改用户名和密码，包括：ruoyi-system-dev.yml、ruoyi-auth-dev.yml、ruoyi-gen-dev.yml、ruoyi-job-dev.yml
2. 下载、安装、启动Redis（注意，2020-07-22的时候官网文档没写这一步，但是必须要启动redis），将nacos配置列表所有用到Redis的地方换成自己的链接，包括：ruoyi-gateway-dev.yml、ruoyi-auth-dev.yml、ruoyi-system-dev.yml
3. 启动前端。我遇到了某个包下载不下来的情况（ejs），解决办法是单独安装一下那个包，或者复制一个放到 node\_modules 下也行。node 版本最好不要太低，我用的12.16.3

补充：vscode 打开 ruoyi-ui后，左侧目录会出现 NPM脚本，点击 dev 右侧的小三角就能启动前端了，简单省事。如下图：



**2.代码生成**

使用代码生成的时候，数据库表一定要配置主键，不然可能会遇到一些一眼看过去很奇妙的问题，我遇到的是新增和修改调用的是同一个接口，表现出来就是有一个功能异常。后期熟悉了很容易找见问题，但刚接触的时候确实摸不着头脑。

**3.协同开发一个模块**

多个人协同开发同一个模块的情况多少会遇到。如果不做修改的话，在 nacos 上看到的是一个服务有多个实例，nacos 会根据配置的权重自动做均衡。这种情况下调试就会变得很麻烦。我没采用的解决办法是：假设开发ruoyi-house 模块

1. 修改 resources 目录下的 bootstrap.yml文件中定义的应用名称，比如改成：ruoyi-house-m1
2. nacos 上克隆一份原配置，修改名称，这里的情况就是改成 ruoyi-house-m1-dev.yml
3. 修改 ruoyi-ui 下的 vue.config.js 的 proxy，将要开发模块的请求全部映射到本地，不走网关。这样别人访问不到自己的服务，不用担心调试的时候误入。但是自己可以访问其他模块，不影响自己的开发。参考配置如下：
4. proxy: {
5. *// detail: https://cli.vuejs.org/config/#devserver-proxy*
6. [process.env.VUE\_APP\_BASE\_API + '/house']: {
7. target: `http://localhost:9204`,
8. changeOrigin: true,
9. pathRewrite: {
10. ['^' + process.env.VUE\_APP\_BASE\_API + '/house']: ''
11. }
12. },
13. [process.env.VUE\_APP\_BASE\_API]: {
14. target: `http://localhost:8080`,
15. changeOrigin: true,
16. pathRewrite: {
17. ['^' + process.env.VUE\_APP\_BASE\_API]: ''
18. }
19. }
20. }

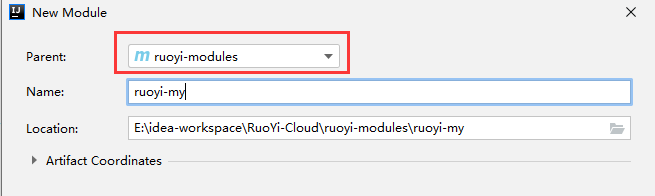
因为是从上到下顺序匹配的，所以顺序一定不能变。同样的也可以实现同一模块的某个接口继续走网关，剩下的走本地，如下

1. proxy: {
2. *// detail: https://cli.vuejs.org/config/#devserver-proxy*
3. [process.env.VUE\_APP\_BASE\_API + '/house/bedroom']: {
4. target: `http://localhost:8080`,
5. changeOrigin: true,
6. pathRewrite: {
7. ['^' + process.env.VUE\_APP\_BASE\_API]: ''
8. }
9. },
10. [process.env.VUE\_APP\_BASE\_API + '/house']: {
11. target: `http://localhost:9204`,
12. changeOrigin: true,
13. pathRewrite: {
14. ['^' + process.env.VUE\_APP\_BASE\_API + '/house']: ''
15. }
16. },
17. [process.env.VUE\_APP\_BASE\_API]: {
18. target: `http://localhost:8080`,
19. changeOrigin: true,
20. pathRewrite: {
21. ['^' + process.env.VUE\_APP\_BASE\_API]: ''
22. }
23. }
24. }

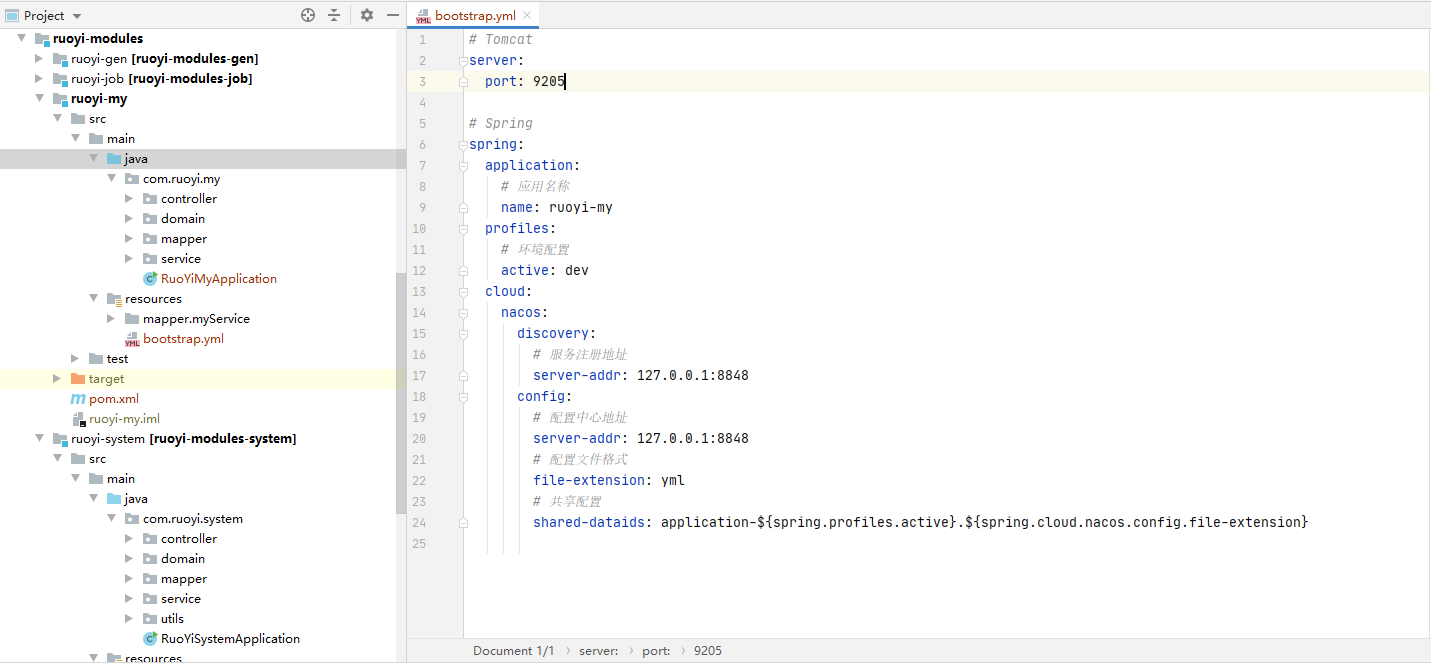
**4.新建模块**

**新增以com.ruoyi开头的模块**

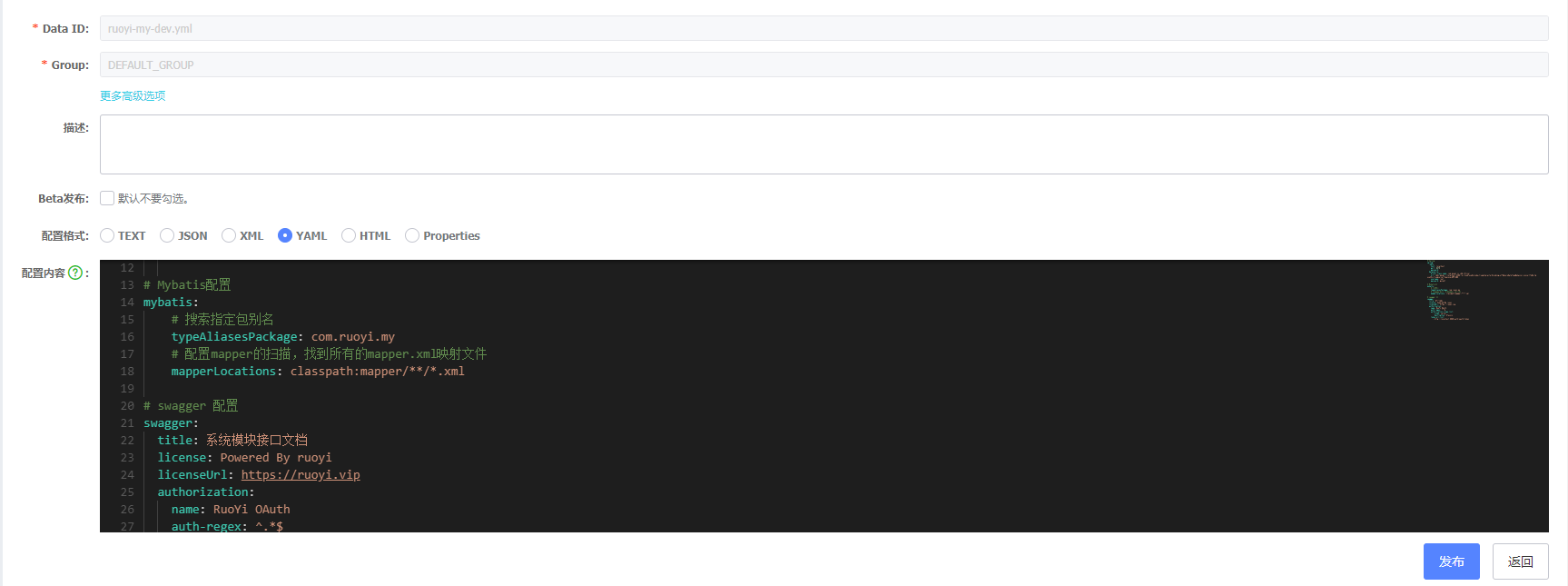
在 ruoyi-modules 上右键，new=》module，Parent 选择 ruoyi-modules，name 以 ruoyi-my 为例。如下图：

新建模块

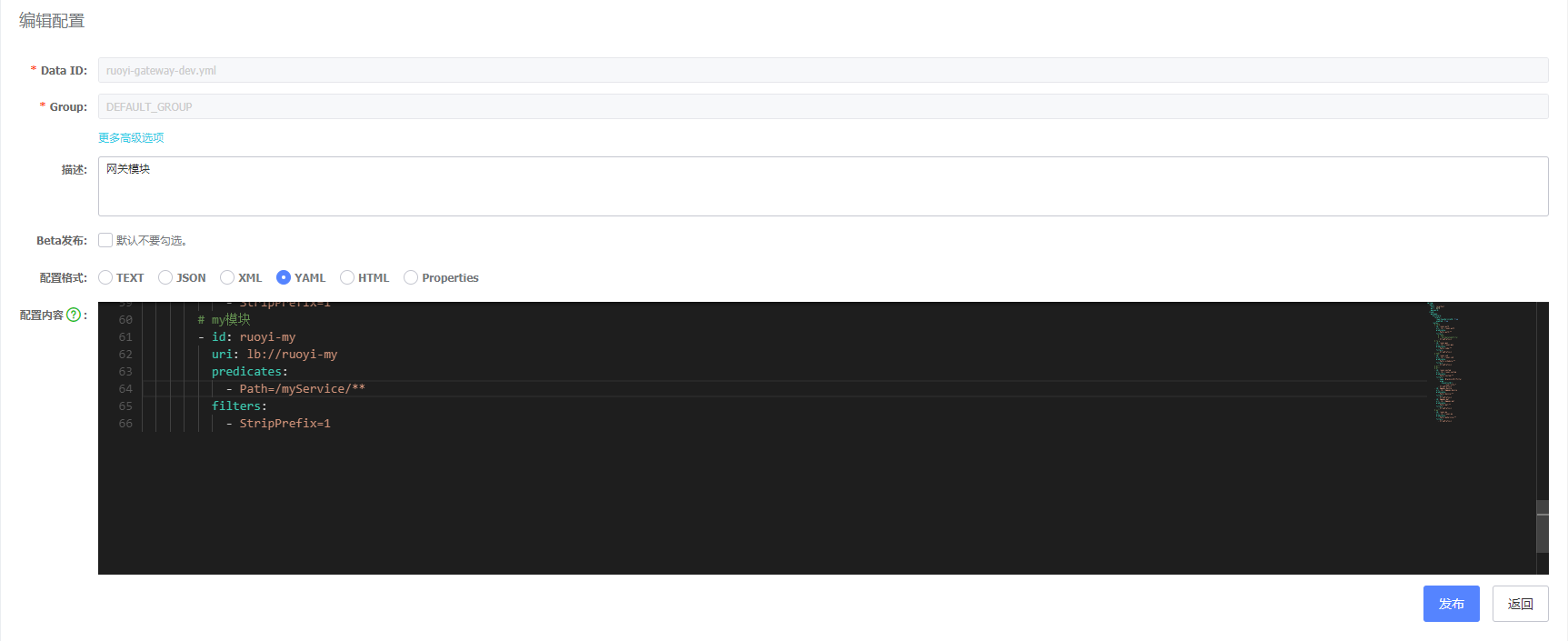
接下来修改 ruoyi-my 的 pom 文件，直接复制 ruoyi-system 的 dependencies 节点和 build 节点。创建包结构，创建启动类（复制 system 模块的启动类修改名称），创建 bootstrap.yml（同样复制 system 模块，修改端口和服务名称）。如下图：

目录结构和bootstrap.yml

接下来登陆 nacos，克隆 ruoyi-system-dev.yml ，命名为 ruoyi-my-dev.yml，修改 mybatis.typeAliasesPackage，改为 com.ruoyi.my，发布。如下图：

nacos上的配置文件

然后修改 ruoyi-gateway-dev.yml，添加新增的模块，发布。如下图：

网关配置

关于上图中出现的 myService 的说明：myService 是和 api 中的 url 关联的。api如下：

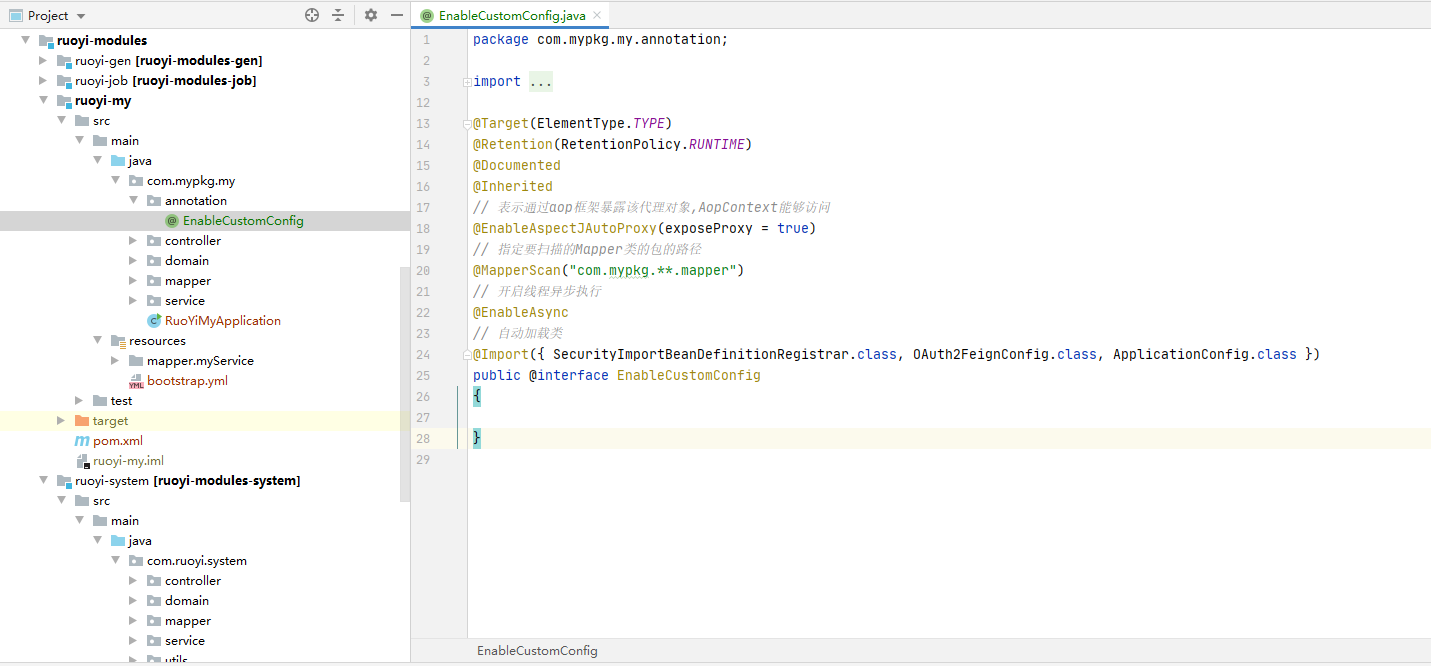
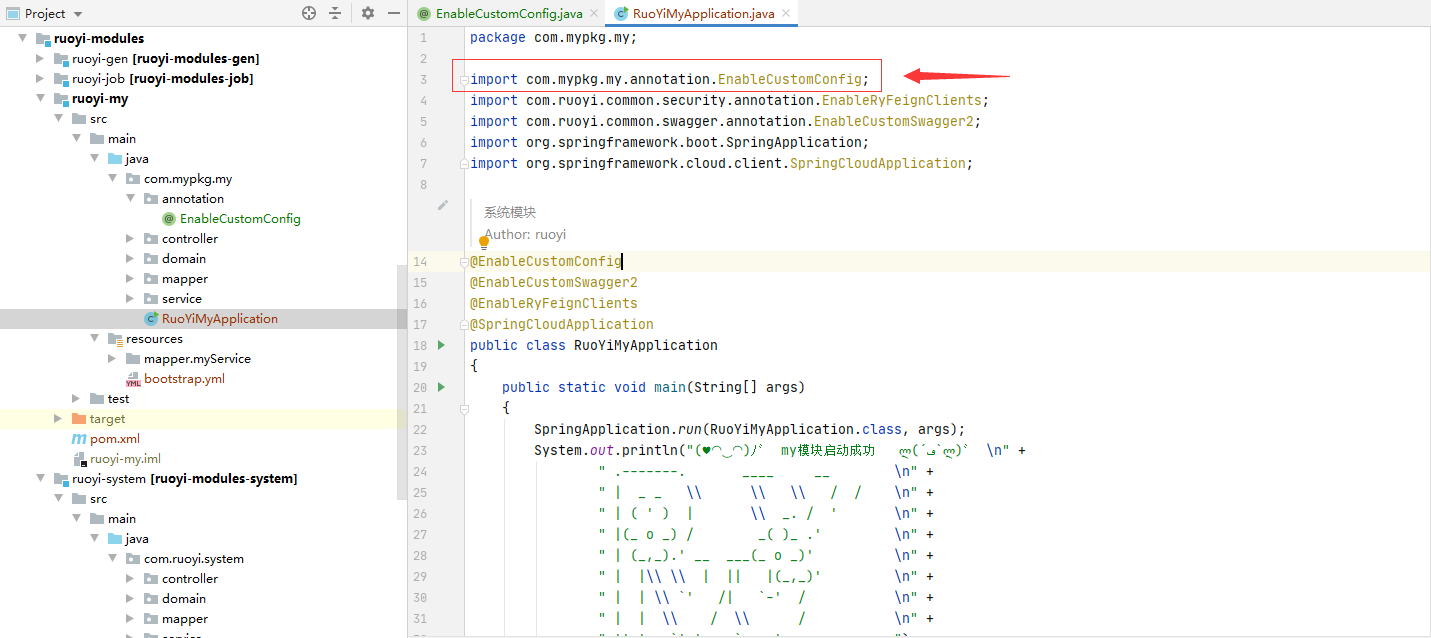
1. *// 查询测试用列表*
2. export function listTest(query) {
3. return request({
4. url: '/myService/test/list',
5. method: 'get',
6. params: query
7. })
8. }

启动新模块后就可以访问了。

**新增不以 com.ruoyi 开头的模块**

在“新增 com.ruoyi 开头的模块”的基础上做以下修改：以 com.mypkg.my 为例

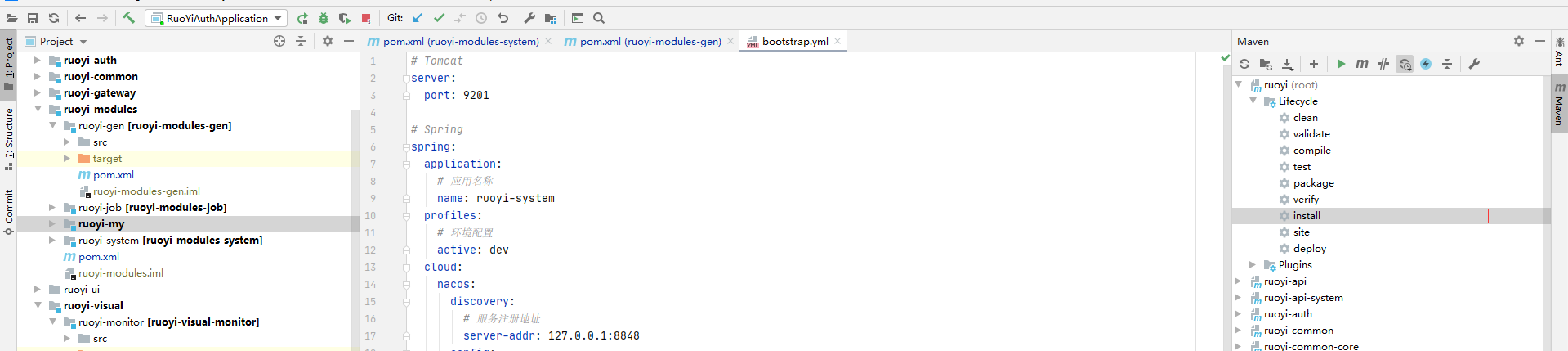
1. 修改 nacos 上对应配置文件，修改 mybatis.typeAliasesPackage 为 com.mypkg.my
2. 新建 annotation 包，将 ruoyi-common 模块下 ruoyi-common-security子模块下的com.ruoyi.common.security.annotation.EnableCustomConfig.java 类复制过来，修改 @MapperScan 的值为 com.mypkg.\*\*.mapper。接着将启动类上的 @EnableCustomConfig 注解改引用 annotation 包下的 EnableCustomConfig。如下图：

EnableCustomConfig 启动类

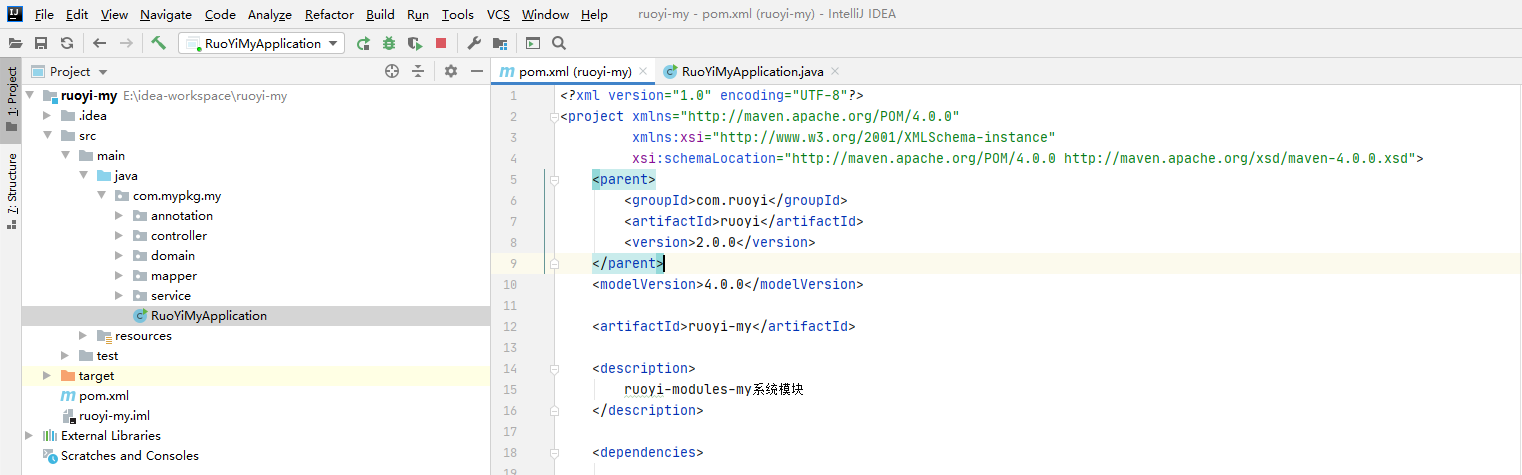
 启动项目即可。

**5.将模块抽取为单独的项目**

首先，运行 mvn install 命令，将 ruoyi 的包安装到本地仓库，因为之后会用到这些包。直接在 idea 里运行就可以。如下图：

install ruoyi 各模块

以“新增不以 com.ruoyi 开头的模块” 为例。先将 ruoyi-my 文件夹复制到一个新的目录，然后用 idea打开。然后修改 pom.xml 文件，修改 parent 节点，指向 ruoyi。如下：

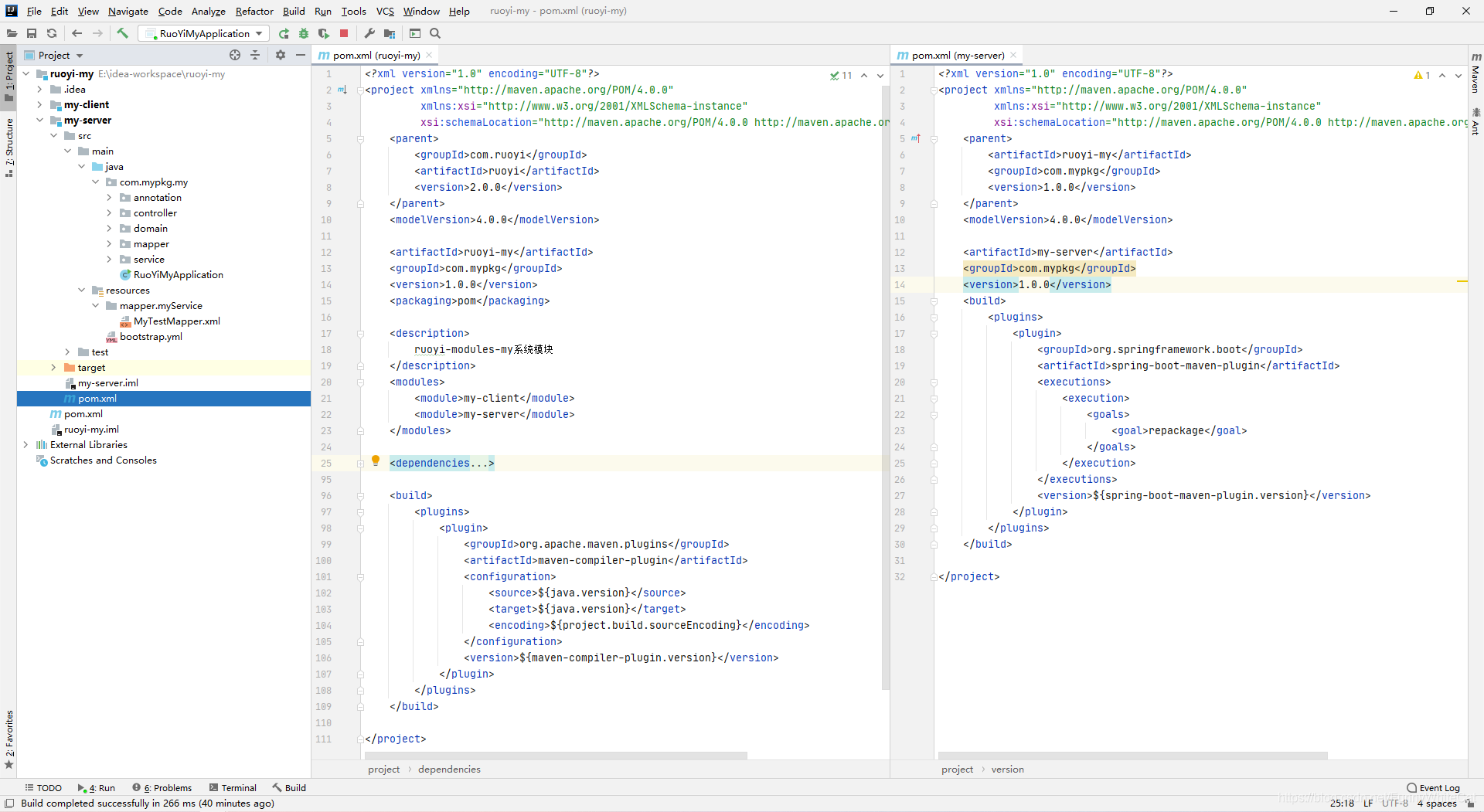
修改pom的parent节点

运行启动类，结束。

如果是以“新增以 com.ruoyi开头的模块” 为基础，那么需要添加“新增不以 com.ruoyi 开头的模块”的第2步，迁移 EnableCustomConfig 类。

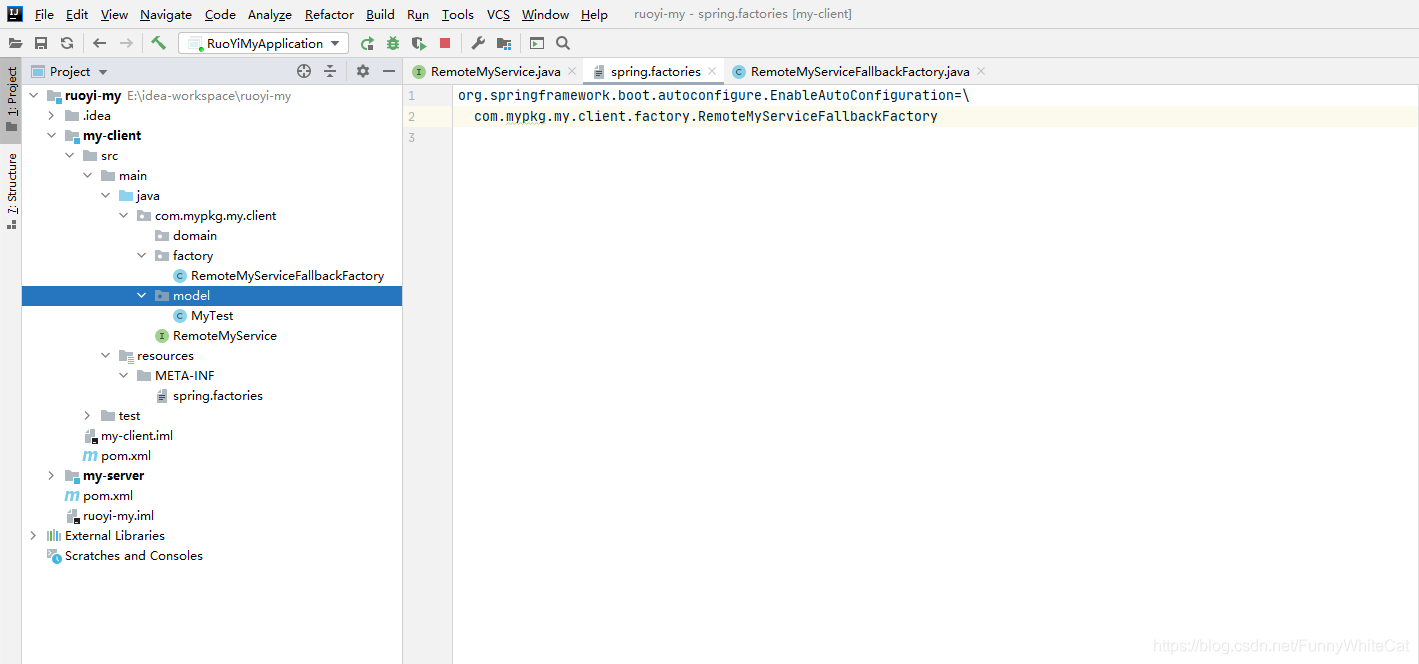
**6.服务接口的定义和服务间调用**

以“新增不以 com.ruoyi 开头的模块” 为例。在原项目里建两个模块：my-client 和 my-server 。my-clinet 里编写对外提供的远程调用接口。将原来的逻辑迁移进 my-server 。迁移的时候要修改两个地方：主 pom 的 build 节点和 my-server pom 的 build 节点。将原主 pom 的build 节点迁移至 my-server 的pom里。主pom改用 maven-compiler-plugin 。另外，建议指定3个pom的 groupId 、artifactId、version，这样别的模块引用的时候不会混乱。如下：

主pom和my-server的pom

1. // 主pom
2. <build>
3. <plugins>
4. <plugin>
5. <groupId>org.apache.maven.plugins</groupId>
6. <artifactId>maven-compiler-plugin</artifactId>
7. <configuration>
8. <source>${java.version}</source>
9. <target>${java.version}</target>
10. <encoding>${project.build.sourceEncoding}</encoding>
11. </configuration>
12. <version>${maven-compiler-plugin.version}</version>
13. </plugin>
14. </plugins>
15. </build>

 my-client 的创建参照 ruoyi-api 即可。有一点需要注意，新建的 fallbackFactory 接口要在 resource.META-INF.spring.factories 里注册，如下图：



 重复“新增不以 com.ruoyi 开头的模块” ，新建一个项目 ruoyi-my2，ruoyi-my2 调用 ruoyi-my 创建的服务。

首先 mvn install ruoyi-my 项目。ruoyi-my2 的 pom 文件中添加 ruoyi-my 项目的 my-client 的引用，不需要整个引用 ruoyi-my。然后在 annotation 包中新建接口 EnableRyFeignClients，修改启动类对 EnableRyFeignClients 的引用，改引用自定义的 EnableRyFeignClients。接着修改 EnableRyFeignClients 的 basePackages。因为 OpenFeign 默认只扫描启动类所在目录及子目录，所以默认扫不到其他包里定义的远程调用接口。EnableRyFeignClients 内容如下：

1. import java.lang.annotation.\*;
3. */\*\**
4. *\* 自定义feign注解*
5. *\* 添加basePackages路径*
6. *\**
7. *\* @author ruoyi*
8. *\*/*
9. @Target(ElementType.TYPE)
10. @Retention(RetentionPolicy.RUNTIME)
11. @Documented
12. @EnableFeignClients
13. public @interface EnableRyFeignClients
14. {
15. String[] value() default {};
17. String[] basePackages() default { "com.ruoyi", "com.mypkg" };
19. Class<?>[] basePackageClasses() default {};
21. Class<?>[] defaultConfiguration() default {};
23. Class<?>[] clients() default {};
24. }

 ruoyi-my2 的 pom 文件中添加对 my-client 的引用：

1. <dependency>
2. <groupId>com.mypkg</groupId>
3. <artifactId>my-client</artifactId>
4. <version>1.0.0</version>
5. </dependency>

然后就可以调用 RemoteMyService 服务了。

**7.部署运行**

* **jar 包运行**

打包后直接 java -jar \*\*\*.jar 会报异常：

1. 2020-08-01 22:04:19.429 ERROR 15556 --- [ main] c.a.c.n.c.NacosPropertySourceBuilder : parse data from Nacos error,dataId:application-dev.yml,data:#请求处理的超时时间
2. ......
3. org.yaml.snakeyaml.error.YAMLException: java.nio.charset.MalformedInputException: Input length = 1

 参考文章[NACOS MalformedInputException 无法读取中文配置问题](https://www.jianshu.com/p/81e0f5b78adf)，解决办法是运行时指定编码格式：

java -D'file.encoding=utf-8' -jar .\ruoyi-auth-2.0.0.jar

* **Redis 只能 localhost 访问，本机 IP 访问报错**

报错内容：io.lettuce.core.RedisException: java.io.IOException: 远程主机强迫关闭了一个现有的连接。

参考[redis本机能访问 远程不能访问的问题](https://www.cnblogs.com/powerwu/articles/8480203.html)，解决办法：

1. 给 Redis 添加密码
2. 注释掉 bind 127.0.0.1

* **docker 运行 nacos**

 开发环境下 nacos 使用单机版。配置参考了[docker内搭建单机模式的nacos](https://blog.csdn.net/li495512592/article/details/89434190)，但是原文章写的比较早，参数已经发生了变化，所以以下文为准（2020-08-02版）。建议直接参考官方的参数说明：[nacos-docker](https://github.com/nacos-group/nacos-docker/tree/ba41672f2f3db1b5e730f77a4b3cfedde1ae40f8#common-property-configuration)**。**

1. 拉取镜像

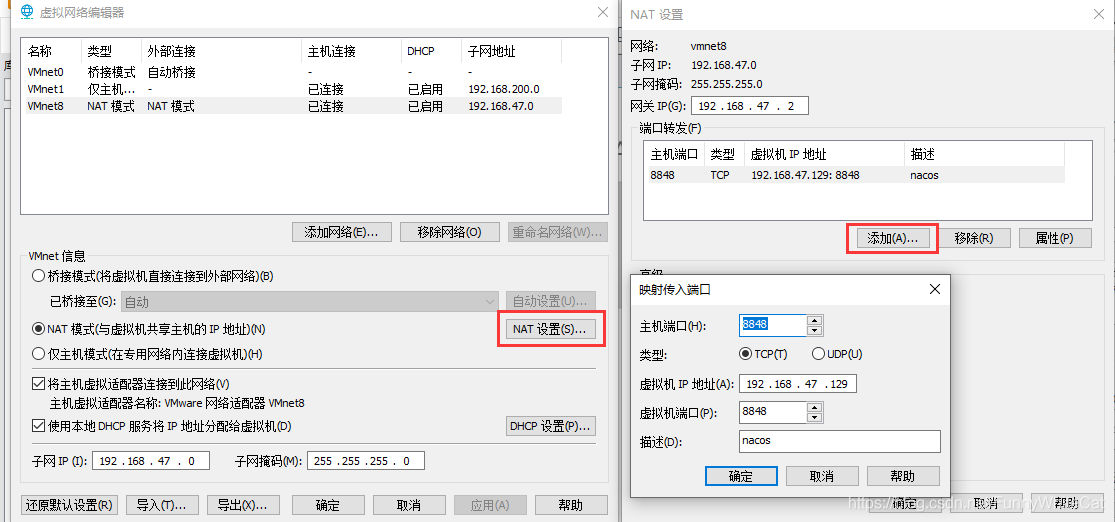
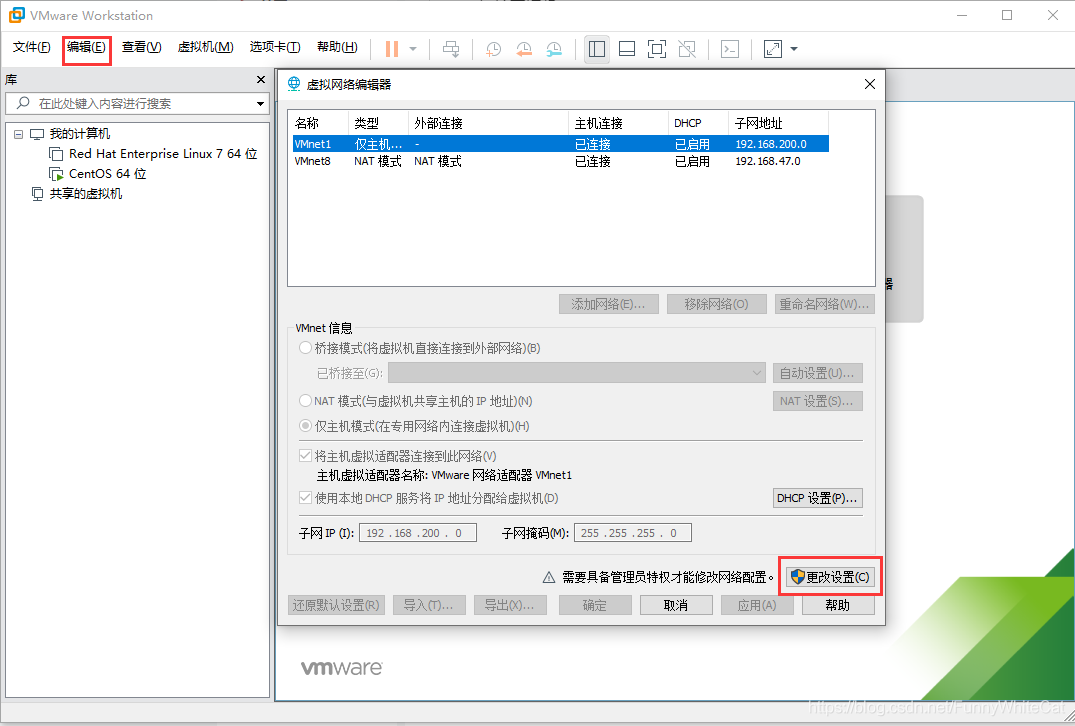
docker pull nacos/nacos-server

1. 启动镜像

docker run --env-file=env.list --name nacos -d -p 8848:8848 nacos/nacos-server

env.list 内容

* 1. MODE=standalone
  2. SPRING\_DATASOURCE\_PLATFORM=mysql
  3. MYSQL\_SERVICE\_DB\_NAME=ry-config
  4. MYSQL\_SERVICE\_HOST=192.168.1.109
  5. MYSQL\_SERVICE\_USER=root
  6. MYSQL\_SERVICE\_PASSWORD=root

如果使用的虚拟机（我用的vmware），则需要将主机的 8848 端口映射到虚拟机的 8848 端口，这样其他电脑才能访问到 docker 里的 nacos。步骤是：编辑=》虚拟网络编辑器=》更改设置=》NAT设置=》添加，

* **docker 运行 ruoyi 模块**

运行网关模块，发现没有注册到nacos。运行命令： docker logs 容器id，发现报错： java.net.NoRouteToHostException: No route to host (Host unreachable)。解决办法：防火墙开放相应端口。

* **开发环境 docker 部署注意事项**

默认情况下，docker 内部网络和外部是隔离的。启动的时候讲宿主机的 8080 端口映射到 docker 内网关模块的 8080 端口，这样外部通过网关访问容器内的服务没有问题，内部服务间的调用也没有问题。但是我本地访问 docker 内部的服务的时候就访问不到了，因为 docker 内服务的 IP 都是172.\*。我遇到的报错提示是：请求 auth 时超时。解决办法是修改docker的网络为 host，这样服务会共用宿主机的 IP ，访问就正常了。docker 的四种网络模式的区别可以参看：[Docker四种网络模式](https://www.jianshu.com/p/22a7032bb7bd)。