**相关Linux的准备工作**

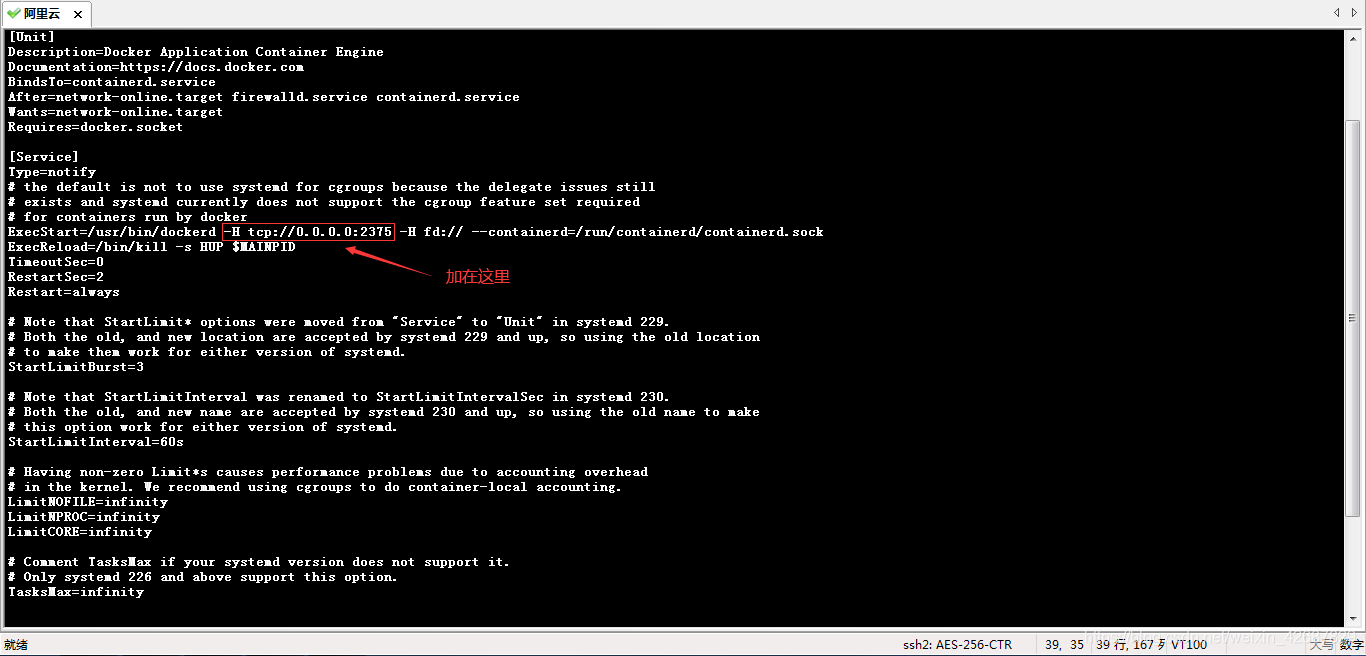
1、配置docker远程连接端口

编辑服务器上的docker.service文件

vi /usr/lib/systemd/system/docker.service

找到如下图所示的地方，添加以下东西（**版本需为centos7以上**）：

-H tcp:*//0.0.0.0:2375*



如果版本是centos7以下的则直接在这一行的最后添加即可，centos7以下版本添加后的全部代码为：

ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:2375

2、保存修改退出，重启docker

1. systemctl daemon-reload
2. service docker restart

3、测试远程连接是否正常

curl http:*//localhost:2375/version*

如果出现以下内容则能正常连接：



4、开放端口

需要将2375端口进行开放才能被远程连接，如果是阿里云主机的话，可以直接登录阿里云去进行开放：



如果是虚拟机的话，可以用以下命令进行开放：

firewall-cmd --zone=public --add-port=2375/tcp --permanent

然后用浏览器访问Linux主机的IP:2375/version，测试端口是否开放成功

5、在docker里装MySQL数据库（如果已经装过则不用）

1. *#拉取MySQL镜像*
2. docker pull mysql:5.7
4. *#使用镜像开启服务（PASSWORD后面是root用户的密码）*
5. docker run -d -p 3308:3306 --name mysql5.7 -e MYSQL\_ROOT\_PASSWORD=123456 mysql:5.7

然后看MySQL是否已启动：



如果已启动，则可以用工具去连接，**注意端口用3308**，因为MySQL的3306端口映射成了3308

6、在docker里装redis（如果已经装过则不用）

1. *#拉取redis镜像*
2. docker pull redis
4. *#启动redis*
5. *#redis-server --appendonly yes：在容器执行redis-server启动命令，并打开redis持久化配置*
6. *#requirepass "123456"：配置redis连接密码*
7. docker run --name redis -p 6379:6379 -d --restart=always redis redis-server --appendonly yes --requirepass "123456"

然后看redis是否已启动：

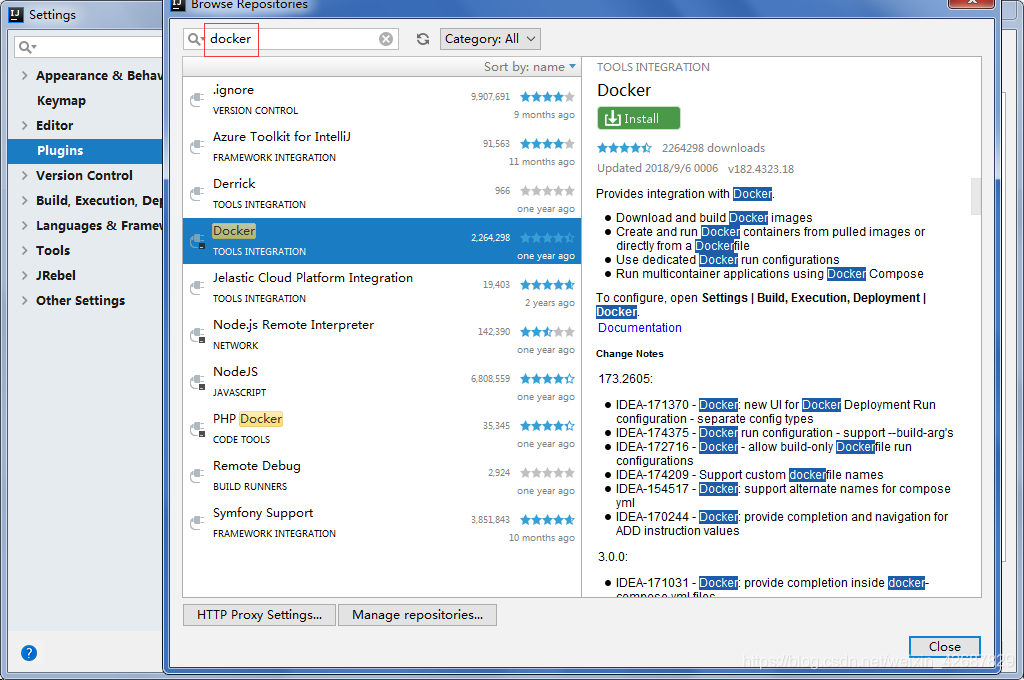


如果已启动，则可以用工具去连接，端口用6379

**相关idea的操作**

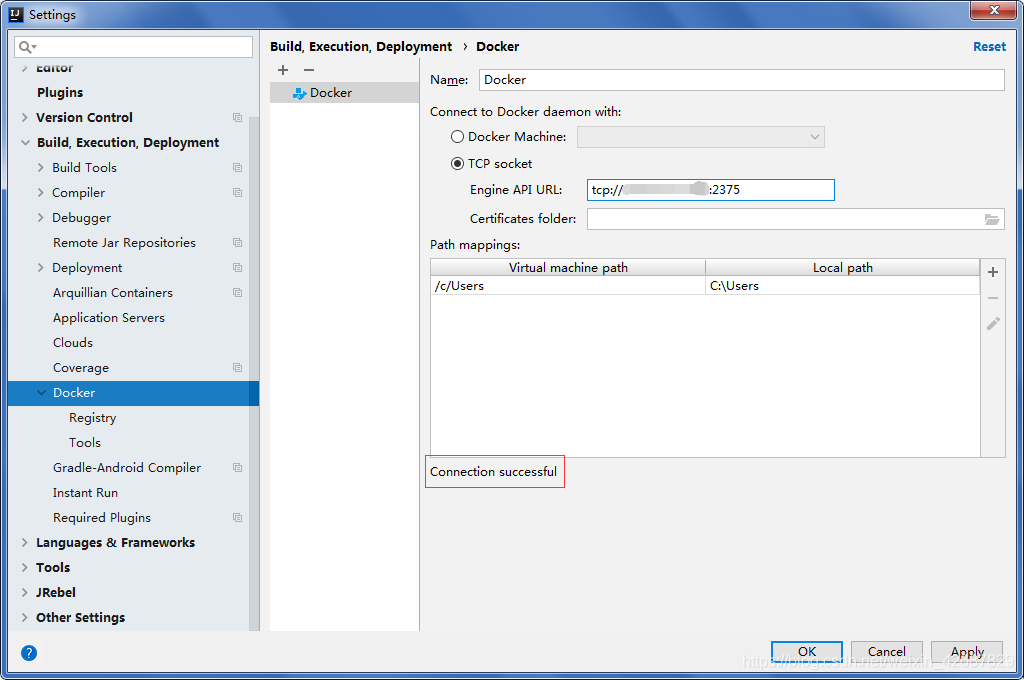
1、下载docker插件

在File-》Setting-》Plugins-》Browse Repositones进行下载：

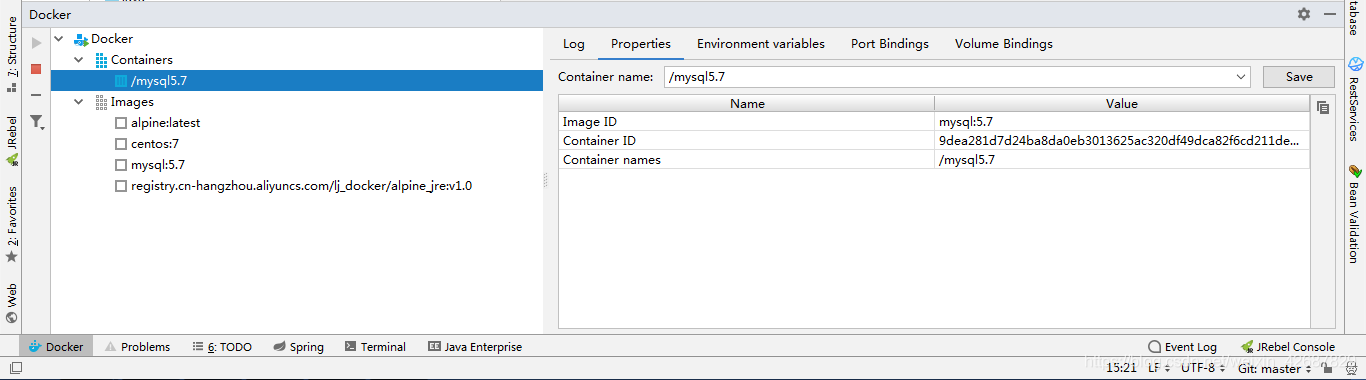


2、使用docker插件连接到Linux上的docker容器

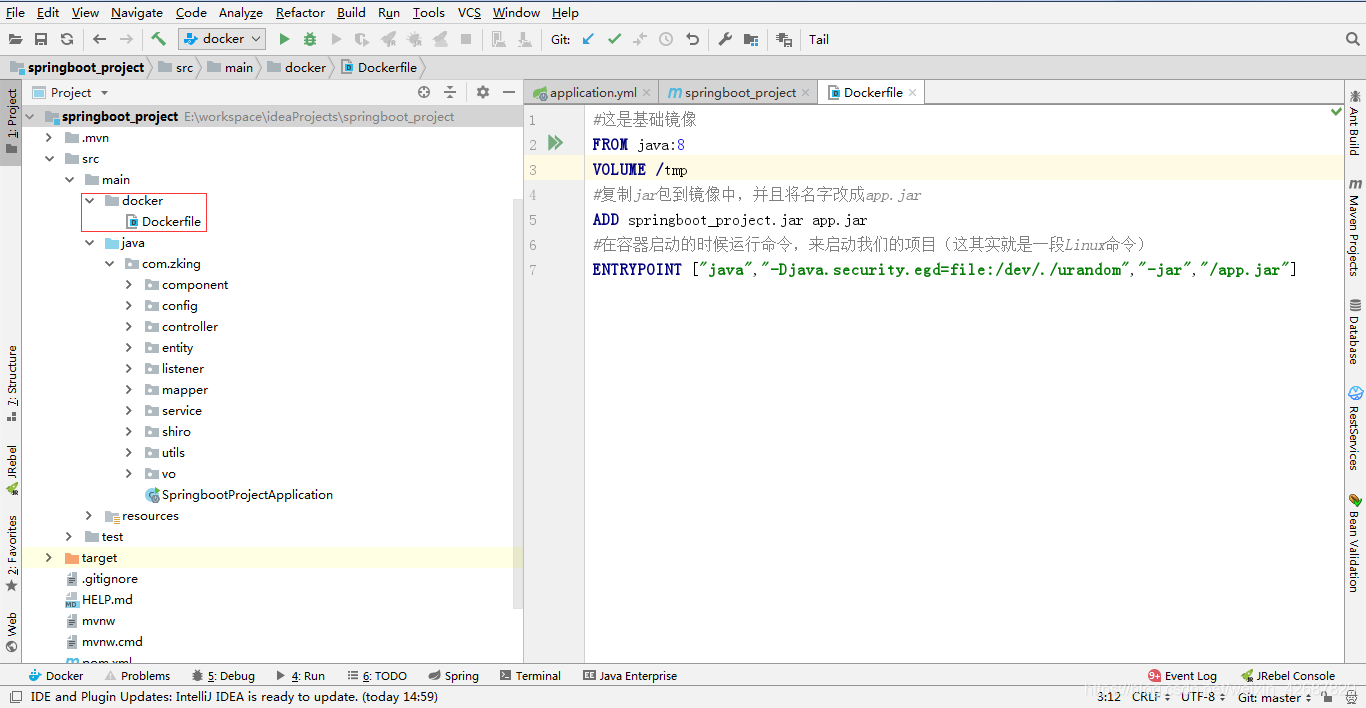
在File-》Settings-》Build, Execution, Deployment-》Docker里用Linux主机的IP:2375进行连接：



连接成功后可以看到docker里所有的镜像以及容器：



3、在项目中添加一个docker文件夹并创建Dockerfile文件



Dockerfile文件的内容如下：

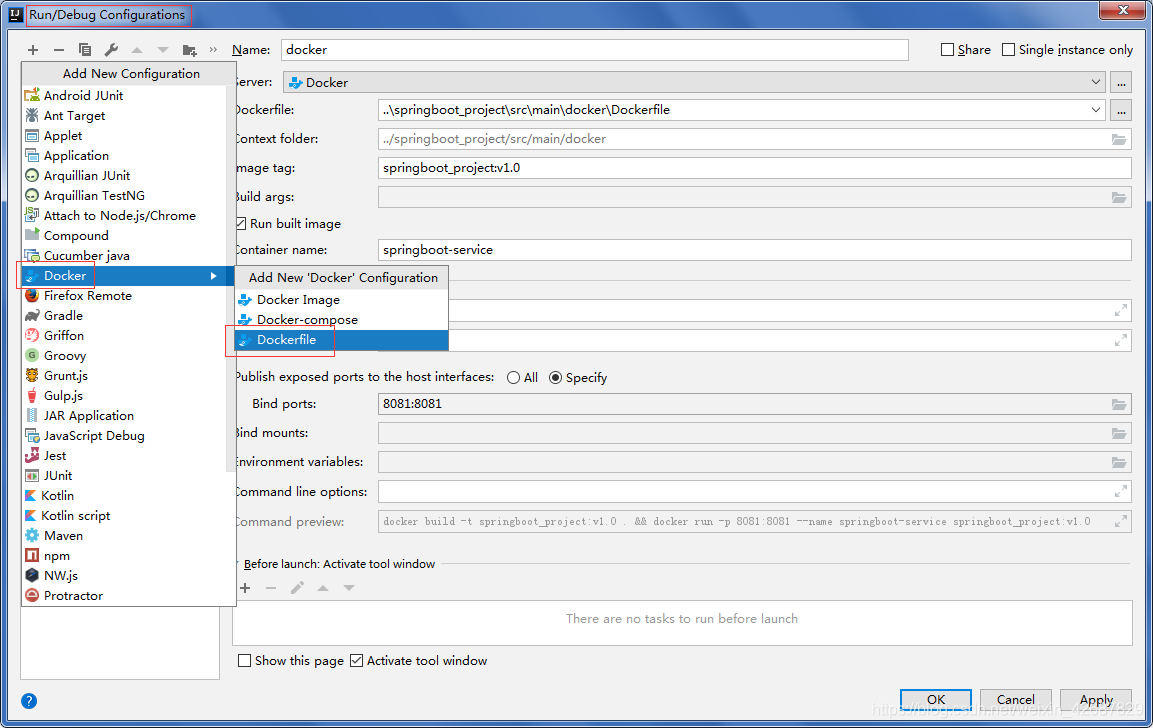
1. *#这是基础镜像*
2. FROM java:8
3. VOLUME /tmp
4. *#复制jar包到镜像中，并且将名字改成app.jar*
5. ADD springboot\_project.jar app.jar
6. *#在容器启动的时候运行命令，来启动我们的项目（这其实就是一段Linux命令）*
7. ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]

建议先登录Linux用命令拉取java:8基础镜像，这样以免到后面build镜像的时候才拉取，造成用时很长

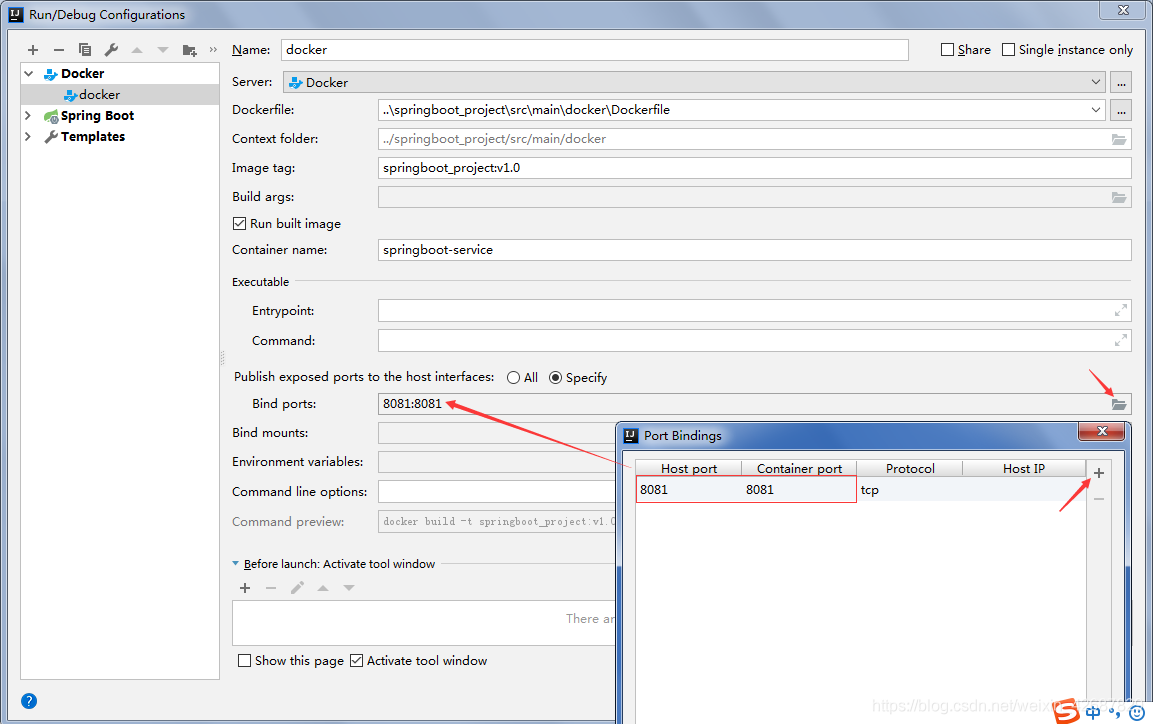
4、在pom.xml里添加docker相关配置

1. <plugin>
2. <groupId>com.spotify</groupId>
3. <artifactId>docker-maven-plugin</artifactId>
4. <version>1.0.0</version>
5. <configuration>
6. <dockerDirectory>src/main/docker</dockerDirectory>
7. <resources>
8. <resource>
9. <targetPath>/</targetPath>
10. <directory>${project.build.directory}</directory>
11. <include>${project.build.finalName}.jar</include>
12. </resource>
13. </resources>
14. </configuration>
15. </plugin>
16. <plugin>
17. <artifactId>maven-antrun-plugin</artifactId>
18. *<!--绑定mvn package命令，当执行package这个maven命令打包项目的同时*
19. *会把target目录下的jar包给copy到docker目录去-->*
20. <executions>
21. <execution>
22. <phase>package</phase>
23. <configuration>
24. <tasks>
25. <copy todir="src/main/docker" file="target/${project.artifactId}.${project.packaging}"></copy>
26. </tasks>
27. </configuration>
28. <goals>
29. <goal>run</goal>
30. </goals>
31. </execution>
32. </executions>
33. </plugin>

5、添加一个docker配置



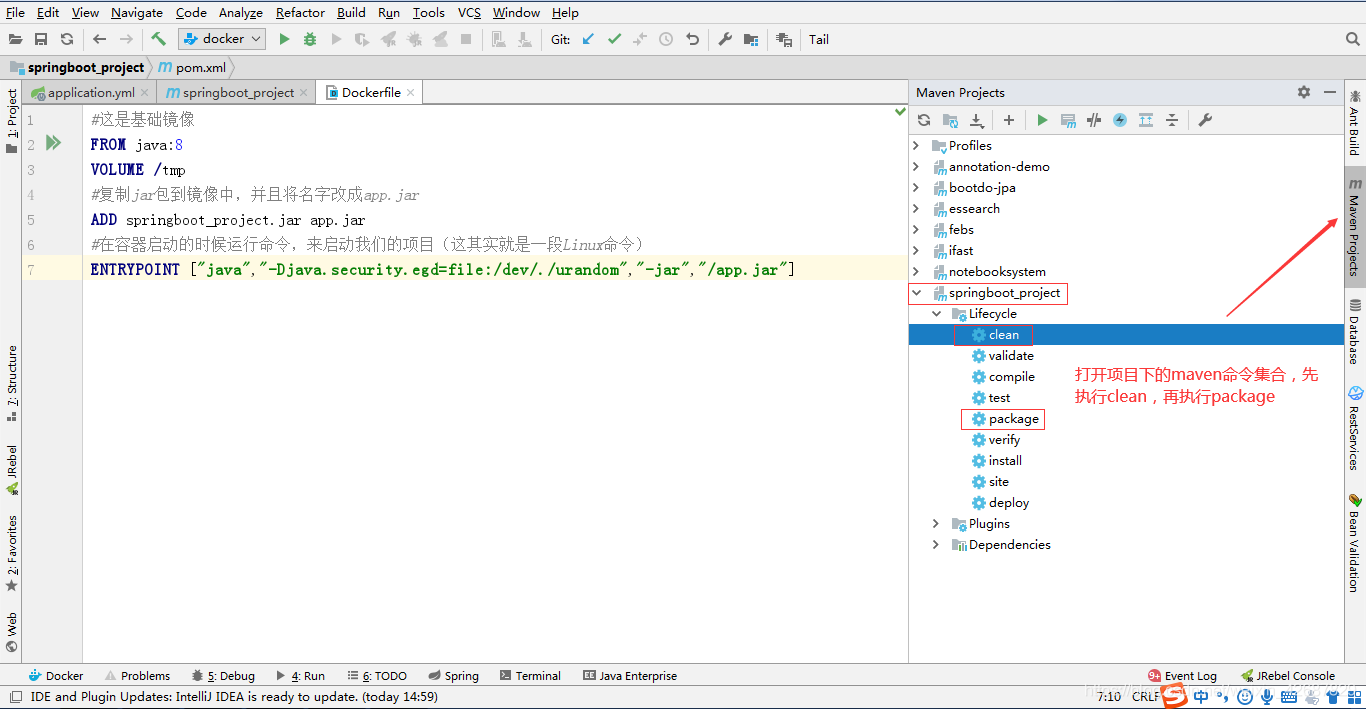
具体配置如下：



部分解释如下：

1. 1.Name：这个根据自己的项目进行命名
2. 2.Server：这个就是远程的服务器上的docker
3. 3.Dockerfile：这个就是自己项目的Dockerfile文件
4. 4.Image Tag：不能重复根据自己项目来定义
5. 5.Container name：创建你自己生成容器的名字
6. 6.Bindports：绑定服务器对外提供的端口和自己项目的端口(阿里云的端口需要在后台自行管理打开)
7. 格式为[宿主机端口]:[容器内部端口]
8. 7.Bind mounts：静态资源存放路径的映射，根据自己的项目自行映射
9. 格式为[宿主机目录]:[容器内部目录]
10. 8.Command line options：这是docker内部的机制当某个容器需要其他运行容器的时候需要在这里配置
11. 语法：--link 运行容器的名称: 自己定义的名称（多个的容器的话需要使用空格分隔）

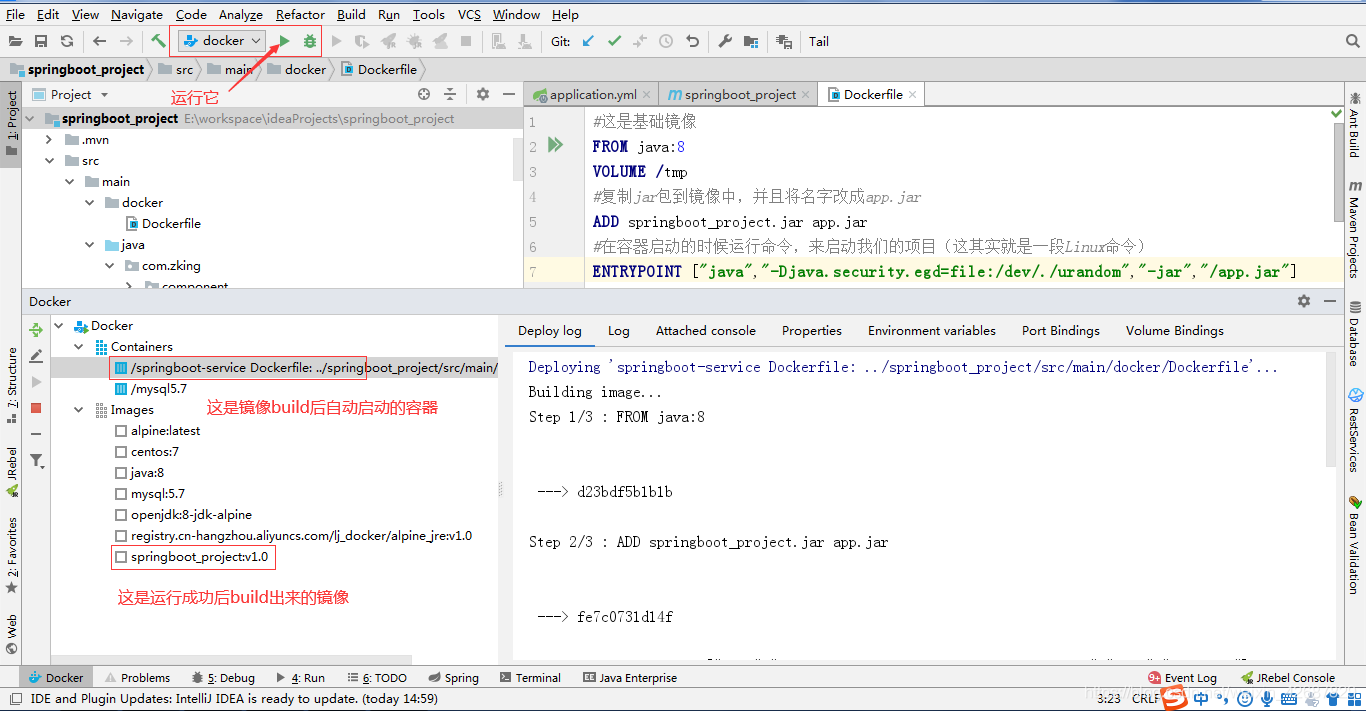
6、将项目打成jar包

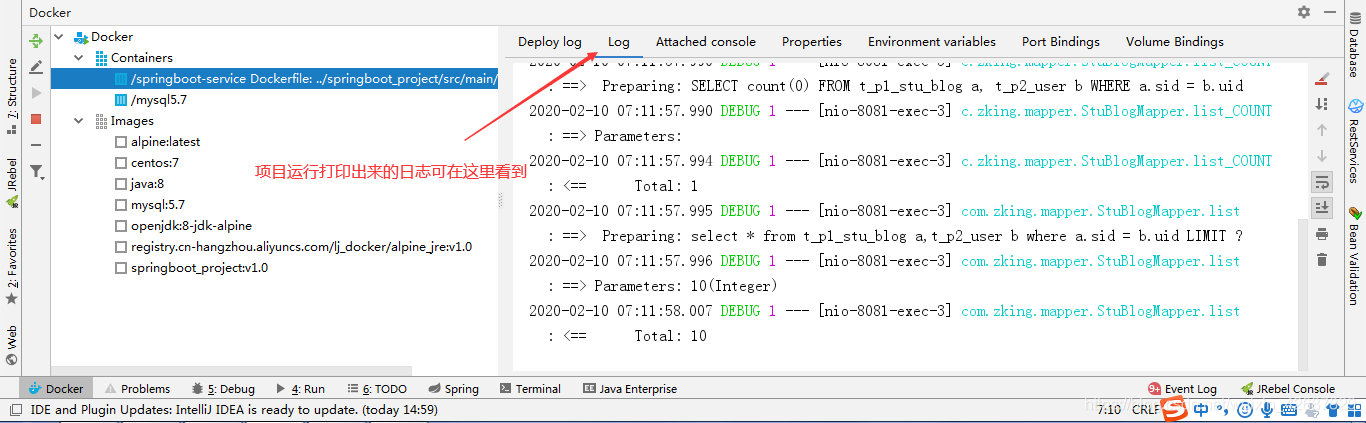


打包成功后，项目的target目录跟docker目录下都有一个jar包

7、运行docker来build镜像

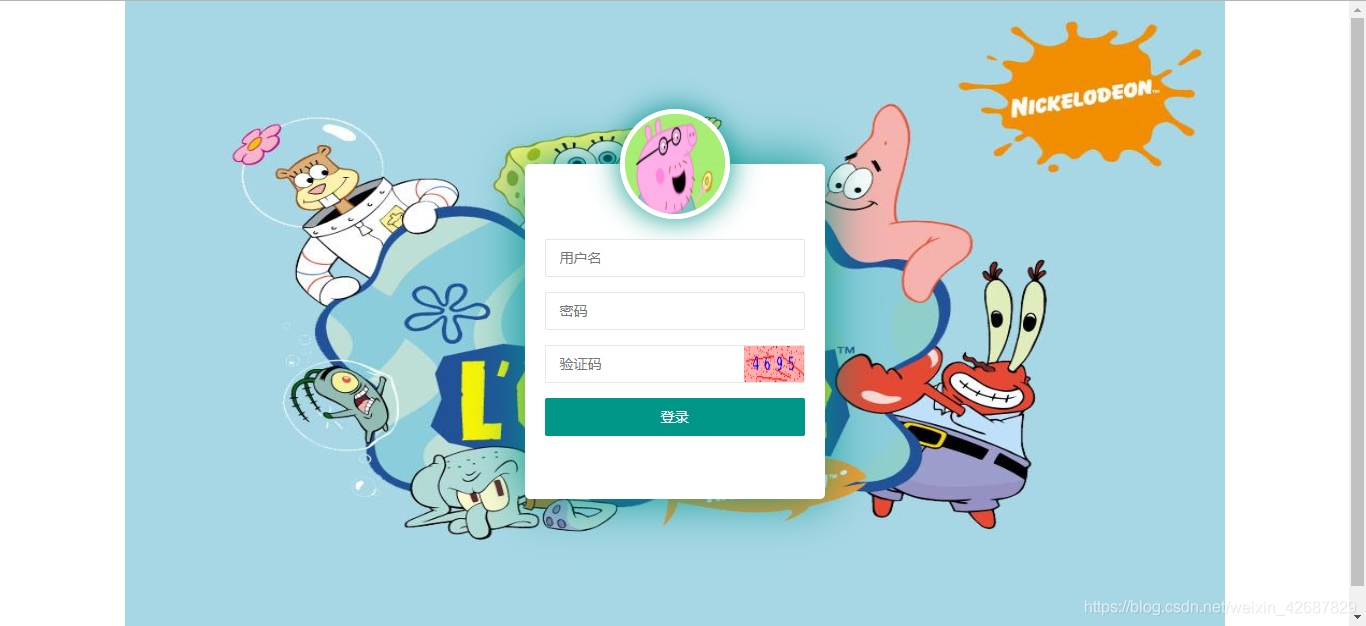
运行之前配置好的Dockerfile，打包镜像，并将镜像部署到远程docker运行：





8、用浏览器访问项目看效果

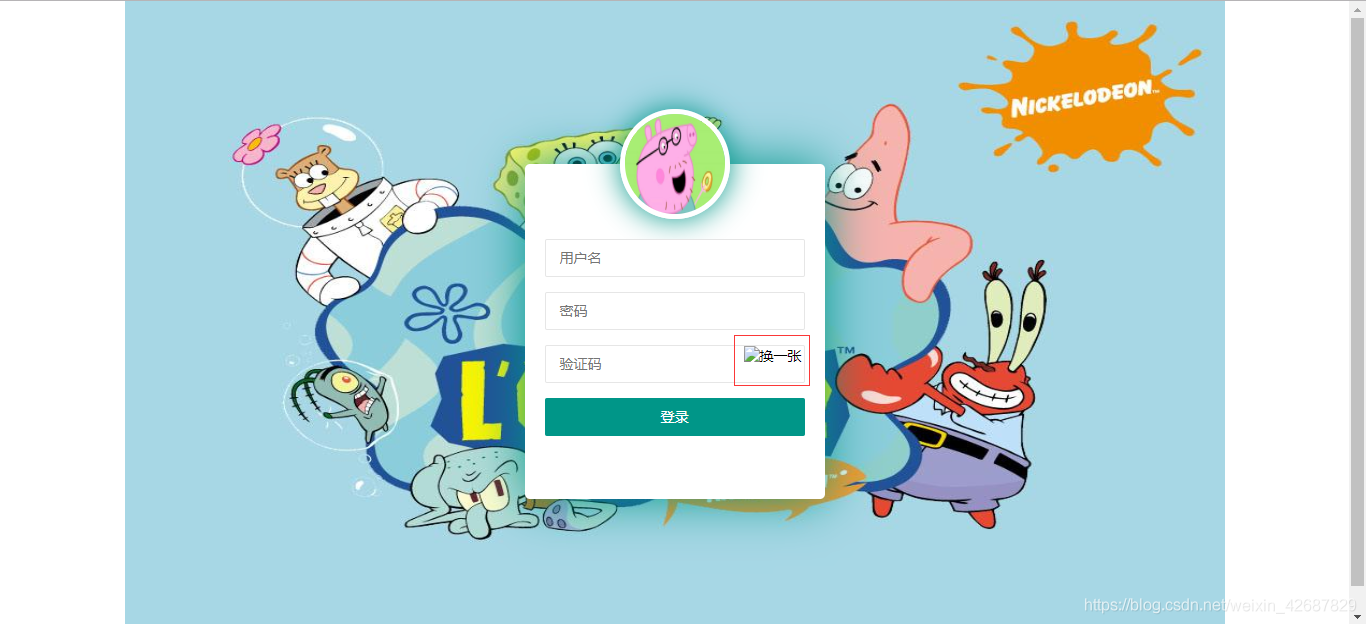
用浏览器访问Linux主机IP+端口+项目名，效果如下：





**部署期间遇到的问题**

项目部署成功后访问登录页面，调用后端接口生成的图片验证码显示不出来，看日志在报空指针异常，如下图：



产生原因：我开始用的jdk基础镜像是openjdk:8-jdk-alpine，这个版本的jdk相比于我们平时用的jdk是不完整的，有些包是没有的，所以导致空指针异常的出现

解决方案：将基础镜像换成java:8

