# Maven插件构建docker

## 引入构建docker插件

<plugin>

<groupId>com.spotify</groupId>

<artifactId>docker-maven-plugin</artifactId>

<version>1.2.2</version>

</plugin>

## 在插件中增加相关配置

<configuration>

<!-- 镜像名称 -->

<imageName>spring-cloud-config-server:0.0.1</imageName>

<!-- 指定基础镜像等价于FROM指令 -->

<baseImage>java</baseImage>

<!-- docker 的 ENTRYPOINT指令 -->

<entryPoint>["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar","--spring.profiles.active=docker"]</entryPoint>

<resources>

<resource>

<targetPath>/</targetPath>

<!-- 需要指定复制的根目录 ${project.build.directory}表示target目录 -->

<directory>${project.build.directory}</directory>

<!-- 用于指定需要复制的文件 ${project.build.finalName}.jar指的是打包之后的jar包文件-->

<include>${project.build.finalName}.jar</include>

</resource>

</resources>

<!-- 指定需要推送的docker主机位置 -->

<dockerHost>http://192.168.60.204:2375</dockerHost>

</configuration>

此配置还可以简化成如下样子，直接指明Dockerfile文件路径即可：

<configuration>

<!-- 镜像名称 -->

<imageName>spring-cloud-config-server:0.0.1</imageName> <!-- 指明Dockerfile的所在目录-->

<dockerDirectory>${project.basedir}/src/main/docker</dockerDirectory>

<resources>

<resource>

<targetPath>/</targetPath>

<!-- 需要指定复制的根目录 ${project.build.directory}表示target目录 -->

<directory>${project.build.directory}</directory>

<!-- 用于指定需要复制的文件 ${project.build.finalName}.jar指的是打包之后的jar包文件-->

<include>${project.build.finalName}.jar</include>

</resource>

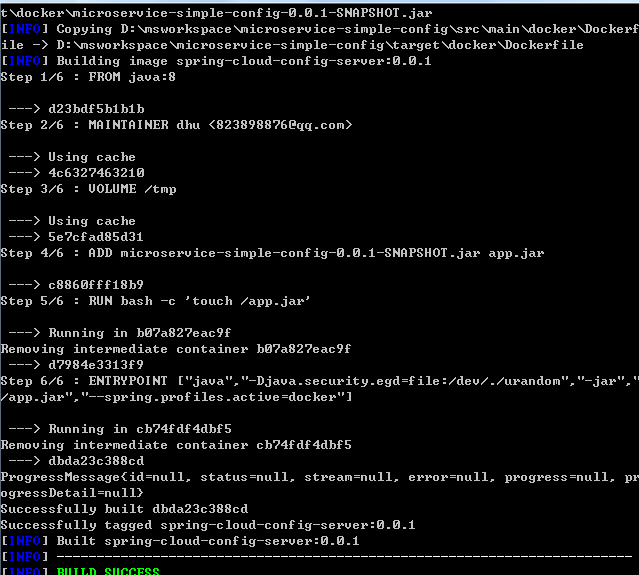
</resources>

<!-- 指定需要推送的docker主机位置 -->

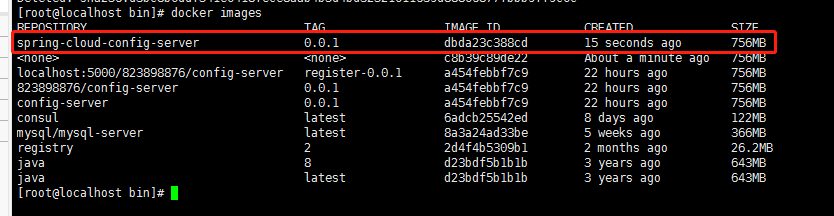
<dockerHost>http://192.168.60.204:2375</dockerHost>

</configuration>

## 3、通过命令mvn clean package:build生成镜像



在docker image查看生成的镜像：



## 4、将插件绑定在某个phase执行

<!-- 将步骤package绑定一个goal执行build 当执行mvn clean package 等价于 mvn clean package:build -->

<executions>

<execution>

<id>build-image</id>

<phase>package</phase>

<goals>

<goal>build</goal>

</goals>

</execution>

</executions>

具体细节参考maven的生命周期

## 推送镜像

通过docker push xxxx/repo:tag可以推送镜像，也可以用maven插件直接推送镜像

1. 修改maven的全局配置文件settings.xml，在其添加以下内容

<server>

<id>docker-hub</id>

<!-- docker hub 的用户名 -->

<username>823898876</username>

<!-- docker hub 的密码 -->

<password>823898876</password>

<configuration>

<!-- docker hub的邮箱 -->

<email>823898876@qq.com</email>

</configuration>

</server>

最终maven插件改成如下

<plugin>

<groupId>com.spotify</groupId>

<artifactId>docker-maven-plugin</artifactId>

<version>1.2.2</version>

<!-- 将步骤package绑定一个goal执行build 当执行mvn clean package 等价于 mvn clean package:build -->

<executions>

<execution>

<id>build-image</id>

<phase>package</phase>

<goals>

<goal>build</goal>

</goals>

</execution>

</executions>

<configuration>

<!-- 镜像名称 -->

<imageName>spring-cloud-config-server:0.0.1</imageName>

<!-- 以下配置均可用 dockerDirectory 带提 -->

<!-- 指定基础镜像等价于FROM指令 -->

<!-- <baseImage>java</baseImage> -->

<!-- docker 的 ENTRYPOINT指令 -->

<!-- 指明Dockerfile的所在目录-->

<dockerDirectory>${project.basedir}/src/main/docker</dockerDirectory>

<resources>

<resource>

<targetPath>/</targetPath>

<!-- 需要指定复制的根目录 ${project.build.directory}表示target目录 -->

<directory>${project.build.directory}</directory>

<!-- 用于指定需要复制的文件 ${project.build.finalName}.jar指的是打包之后的jar包文件-->

<include>${project.build.finalName}.jar</include>

</resource>

</resources>

<!-- 指定需要推送的docker主机位置 -->

<dockerHost>http://192.168.60.204:2375</dockerHost>

<!-- 与maven配置文件setting.xml中配置的server.id一致，用于推送镜像 -->

<serverId>docker-hub</serverId>

</configuration>

</plugin>

执行 mvn clean package docker:build -DpushImage

也可以推送到私有仓库只需要将镜像名称改成对应的仓库地址/用户/仓库名称:tag名称即可

也可以借助 imageTags标签

<imageTags>

<imageTag>0.0.1</imageTag>

<imageTag>latest</imageTag>

</imageTags>

这样就可以为同一个镜像指定两个标签，也可以在执行构建命令的时候，使用dockerImageTags参数指定标签名称，例如

mvn clean package docker:build -DpushImageTags -DdockerImageTags=latest -DdockerImageTags=0.0.1

如需重复构建相同标签名称的镜像，可将forceTags设为true，这样就会覆盖构建相同标签的镜像

<!-- 覆盖每次构建相同标签的镜像 -->

<forceTags>true</forceTags>