**使用dockerfile-maven-plugin构建可以自动部署web项目的docker镜像**

要实现这个目标，主要分为两步：

1. dockerfile-maven-plugin的使用
2. dockerfile的编写

**环境：**

Maven

Docker

1. **dockerfile-maven-plugin的使用**

**步骤：**

1. 创建一个简单的基于maven控制的web项目
2. 修改项目根目录下pom.xml文件

添加：

<properties>

...

<docker.image.prefix>demo1</docker.image.prefix>

...

</properties>

<build>

<plugins>

...

<plugin>

<groupId>com.spotify</groupId>

<artifactId>dockerfile-maven-plugin</artifactId>

<version>1.4.0</version>

<executions>

<execution>

<id>default</id>

<goals>

<goal>build</goal>

<goal>push</goal>

</goals>

<configuration>

<repository>172.18.2.210:5000/demo</repository>

</configuration>

</execution>

</executions>

<configuration>

<repository>${docker.image.prefix}/${project.artifactId}</repository>

<buildArgs>

<JAR\_FILE>target/${project.build.finalName}.war</JAR\_FILE>

</buildArgs>

</configuration>

</plugin>

...

</plugins>

</build>

确认打包方式为war：

<packaging>war</packaging>

1. 在项目根目录下创建dockerfile文件
2. **dockerfile的编写**

**思路：**

1. 以原始centos:7镜像为基础搭建镜像
2. 将本地jdk和tomcat拷贝到镜像中去，此处需要将jdk和tomcat的压缩包放在项目根目录下
3. 将本地web项目的war包拷贝到tomcat目录下的webapps文件夹内，此处需要为dockerfile-maven-plugin设置参数，让其在自动build镜像时使用--build-arg<varname>=<value>来指定pom.xml中<JAR\_FILE>的值作为参数值
4. 配置环境变量
5. 设置启动时命令运行catalina.sh

**Dockerfile编写：**

# 选择基础镜像

FROM centos:7

MAINTAINER salo

# 将本地jdk和tomcat拷贝到镜像中去

ADD apache-tomcat-9.0.8.tar.gz /usr/local/

ADD jdk-8u171-linux-x64.tar.gz /usr/local/

# 设置工作目录，会对RUN,ADD,COPY,CMD,ENTRYPOINT等生效

WORKDIR /usr/local/apache-tomcat-9.0.8

# 配置环境变量

ENV JAVA\_HOME /usr/local/jdk1.8.0\_171

ENV CLASSPATH $JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

ENV CATALINA\_HOME /usr/local/apache-tomcat-9.0.8

ENV CATALINA\_BASE /usr/local/apache-tomcat-9.0.8

ENV PATH $PATH:$JAVA\_HOME/bin:$CATALINA\_HOME/lib:$CATALINA\_HOME/bin

# 设置参数名为JAR\_FILE，在build镜像时可以使用--build-arg<varname>=<value>来指定参数值，本次将打包好的项目文件为变量值

ARG JAR\_FILE

# 将${JAR\_FILE}文件，即打包好的项目文件，复制为镜像中的app.jar文件

ADD ${JAR\_FILE} webapps/demo.war

# 设置启动参数

CMD ./bin/catalina.sh run

**效果：**

在项目根目录下输入mvn package打包项目，可在本地镜像中通过docker images找到构建好的镜像172.18.2.210:5000/demo:latest(镜像名由pom.xml中<comfiguration>设置

运行镜像docker run -itd -p 8085:8080 --name demo 172.18.2.210:5000/demo

访问172.18.2.210:8085/demo



Web项目成功部署

**附：**

Pom.xml文件及dockerfile文件

