docker-compose 容器挂载权限问题

原创xiaoguangtouqiang 最后发布于2018-10-23 16:05:42 阅读数 8145 收藏

展开

项目中遇到docker-compose启动springboot的应用，挂载的日志目录没有权限写入的问题；后来查了很多资料，终于有点眉目了，做个记录，希望遇到的朋友少踩点坑~;

1.问题描述

项目框架是使用jhipster生成的，现在需要把生成的日志挂载出去，以便查看日志记录；首先修改下logback-spring.xml的配置，将系统生成的日志文件都存放在项目的logs目录下面；代码如下

然后把这个目录挂载到linux中的/temp/logs/bbb/目录中；下面是app.yml代码

```
version: '2'
services:
    aiops-app:
        image: aiops
        volumes:
            - /temp/logs/bbb/:/home/jhipster/logs/
        environment:
            # - _JAVA_OPTIONS=-Xmx512m -Xms256m
            - SPRING_PROFILES_ACTIVE=prod,swagger
            - SPRING_DATASOURCE_URL=jdbc:mysql://test-mysql:3306/aiops?
useUnicode=true&characterEncoding=utf8&useSSL=false
            - JHIPSTER_SLEEP=10 # gives time for the database to boot before the
application
        ports:
            - 9999:8080
```

这里把logs目录挂载出去到/temp/logs/bbb/这个目录下面；再看下dockerFile文件，这个文件用来把java 应用打包成docker镜像的；如下

```
FROM openjdk:8-jre-alpine

ENV SPRING_OUTPUT_ANSI_ENABLED=ALWAYS \
    JHIPSTER_SLEEP=0 \
    JAVA_OPTS=""

# Add a jhipster user to run our application so that it doesn't need to run as
root
RUN adduser -D -s /bin/sh jhipster
```

```
WORKDIR /home/jhipster

ADD entrypoint.sh entrypoint.sh
RUN chmod 755 entrypoint.sh && chown jhipster:jhipster entrypoint.sh
USER jhipster

ADD *.war app.war

ENTRYPOINT ["./entrypoint.sh"]

EXPOSE 8080
```

这里创建了一个用户jhipster ,设置了工作目录/home/jhipster;然后 把启动命令entrypoint.sh复制到镜像中；并赋权entrypoint.sh为jhipster用户所有，然后使用jhipster用户来启动应用；下面是entrypoint.sh shell脚本的具体内容

```
#!/bin/sh
echo "The application will start in ${JHIPSTER_SLEEP}s..." && sleep ${JHIPSTER_SLEEP}
exec java ${JAVA_OPTS} -Djava.security.egd=file:/dev/./urandom -jar "${HOME}/app.war" "$@"
```

这里的通过java -jar 来启动app,这里的${HOME}目录指的是linux的用户目录，/home/{用户名}，因为当前的用户是jhipster，所以这里的app目录其实是/home/jhipster/app.war；这里的${JHIPSTER_SLEEP}是上面的环境变量JHIPSTER_SLEEP=0；通过docker-compose -f app.yml up -d 命令来启动之后，就报错了，具体如图所示，在linux中会报错，docker在window上只能挂载到用户目录c:\\users\\test\\,不知为啥测试下来不会报错的；

2.报错原因及解决方案

错误的原因是，docker-compose启动的时候会分别在容器和宿主机上创建目录，在宿主机上是使用docker进程来创建目录的，默认使用的角色是root，目录挂载的时候会把这个用户同步到容器里面的目录，所以容器里面logs目录权限也是root，但是容器里面的应用是使用jhipster用户来运行的，当然没有往logs目录里面写的权限，所以就报错了；因为这两个角色是会同步的，所以解决的办法就是修改宿主机的日志权限为docker里面用户的权限；可以使用如下命令修改宿主机日志目录的权限

```
chown -R 1000:1000 "/temp/logs/bbb/"
```

修改前后对比，如图所示

可以看到修改之后默认的bbb目录权限从root变成jhipster，这里的1000就是docker中用户的id，默认docker用户的id是1000；修改之后docker容器中的目录logs角色也变成jhipster，这样权限就统一了，应用运行使用jhipster,日志文件目录权限也是

jhipster,自然可以把日志写到这个目录中了；

这种方式的好处是修改起来简单,缺点是需要额外手工去操作，如果涉及到多个目录的话就很麻烦了，这里介绍另外一种方法

3.使用root权限进入命令行，然后修改权限，再使用jhipster用户启动应用；

基本思路是，使用root用户将容器中的logs目录赋权限为jhipster，然后再使用jhipster用户来启动应用即可；这里需要有个gosu命令，具体的地址是https://github.com/tianon/gosu，这个命令用来替换sudo，它是"su"和"sudo"命令的轻量级替代品，并解决了它们在tty和信号传递中的一些问题。首先需要修改下dockerFile文件，添加sudo命令支持，如下所示

```
FROM openjdk:8-jre-alpine

ENV SPRING_OUTPUT_ANSI_ENABLED=ALWAYS \
    JHIPSTER_SLEEP=0 \
    JAVA_OPTS=""

# Add a jhipster user to run our application so that it doesn't need to run as root
RUN adduser -D -s /bin/sh jhipster
WORKDIR /home/jhipster
ADD *.war app.war
ADD entrypoint.sh entrypoint.sh
RUN chmod 755 entrypoint.sh && chown jhipster:jhipster entrypoint.sh

#使用root用户，下载sogu工具
USER root
ENV GOSU_VERSION 1.11
RUN set -eux; \
        \
        apk add --no-cache --virtual .gosu-deps \
                dpkg \
                gnupg \
        ; \
        \
        dpkgArch="$(dpkg --print-architecture | awk -F- '{ print $NF }')"; \
        wget -O /usr/local/bin/gosu "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch"; \
        wget -O /usr/local/bin/gosu.asc "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch.asc"; \
        \
# verify the signature
        export GNUPGHOME="$(mktemp -d)"; \
# for flaky keyservers, consider https://github.com/tianon/pgp-happy-eyeballs,
```

ala https://github.com/docker-library/php/pull/666
	gpg --keyserver ha.pool.sks-keyservers.net --recv-keys B42F6819007F00F88E364FD4036A9C25BF357DD4; \
	gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu; \
	command -v gpgconf && gpgconf --kill all || :; \
	rm -rf "$GNUPGHOME" /usr/local/bin/gosu.asc; \
	\
clean up fetch dependencies
	apk del --no-network .gosu-deps; \
	\
	chmod +x /usr/local/bin/gosu; \
verify that the binary works
	gosu --version; \
	gosu nobody true

ENTRYPOINT ["./entrypoint.sh"]

EXPOSE 8080

相比之前的内容，添加了切换到root，并下载gosu工具的代码；具体的下载命令可以在gosu的github上也有，地址是https://github.com/tianon/gosu/blob/master/INSTALL.md，注意使用alpine的版本，因为java应用构建是基于这个的，从dockerFile第一句话可以看出；

之后再到entrypoint.sh中赋权，并使用jhipster启动应用，具体内容如下；这里为什么要切换到root用户，如果不切换的话使用chown会报错

#!/bin/sh
chown -R 1000:1000 "logs"
echo "The application will start in ${JHIPSTER_SLEEP}s..." && sleep ${JHIPSTER_SLEEP}
exec gosu jhipster java ${JAVA_OPTS} -Djava.security.egd=file:/dev/./urandom -jar "app.war" "$@"
这样就解决了；这种方式的话，每次打包镜像的时候会去下载gosu，会慢一点，但是不用自己修改权限了；

总结：文章参考了https://www.cnblogs.com/jackluo/p/5783116.html，但是文章里面下载gosu的部分不能正常使用；所以做了一些修改