2021.3.11 dockerfile 构建 nginx 镜像

1）建立工作目录

```
[root@server04 ~]# mkdir nginx
[root@server04 ~]# cd nginx
```

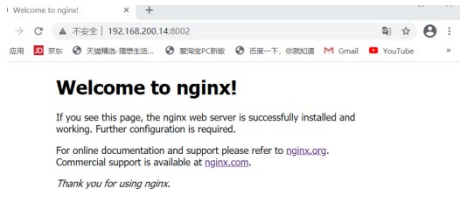
2）编写 dockerfile 文件

```
[root@server04 nginx]# vim run.sh   #启动 nginx 的脚本
/usr/local/sbin/nginx
```

```
[root@server04 nginx]# vim Dockerfile
FROM centos:7
MAINTAINER  Crushlinux <crushlinux@163.com>

RUN yum install -y wget proc-devel net-tools gcc zlib zlib-devel make openssl-devel
RUN wget http://nginx.org/download/nginx-1.19.0.tar.gz
RUN tar zxf nginx-1.19.0.tar.gz -C /usr/src && cd /usr/src/nginx-1.19.0 && ./configure --prefix=/usr/local/nginx && make && make install
RUN echo "daemon off; ">>/usr/local/nginx/conf/nginx.conf
RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

ADD run.sh /run.sh
RUN chmod 775 /run.sh
EXPOSE 80
EXPOSE 443

CMD ["/run.sh"]
[root@server04 nginx]# docker build -t nginx:new .  #开始构建镜像
[root@server04 nginx]# docker run -d -p 8002:80 --name nginx-test --restart=always nginx:new
#启动 nginx 容器，并做一个端口关联
```

3）测试容器

Web 访问 192.168.200.14:8002 的端口



dockerfile 构建 tomcat 镜像

建立工作目录

```
[root@server04 ~]#  mkdir  tomcat
```

```
[root@server04 ~]#  cd    tomcat
```

编写 Dockerfile 文件

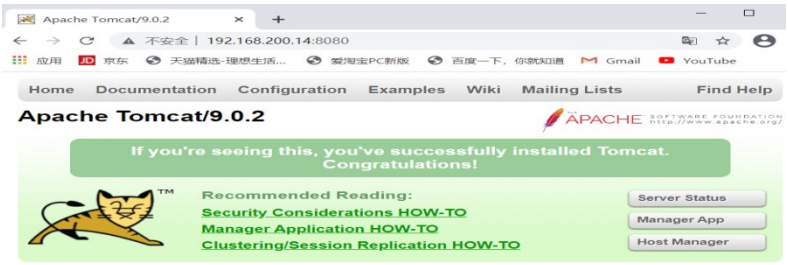
上传 JDK 和 tomcat 软件包（注意：Jik 要 1.8 版本）

```
[root@crushlinux ~]# tar xf jdk-7u65-linux-x64.tar.gz -C tomcat/   #解压软件包
[root@crushlinux ~]# tar xf apache-tomcat-9.0.2.tar.gz -C tomcat/  #解压软件包

[root@crushlinux tomcat]# ls
apache-tomcat-9.0.2  jdk1.7.0_65

[root@server04 ~]#  vim Docekrfile
FROM centos:7
MAINTAINER  Crushlinux <crushlinux@163.com>
ADD jdk1.8.0_191 /usr/local/java
ENV JAVA_HOME /usr/local/java
ENV JAVA_BIN /usr/local/java/bin
ENV JRE_HOME /usr/local/java/jre
ENV PATH $PATH:/usr/local/java/bin:/usr/local/java/jre/bin
ENV CLASSPATH /usr/local/java/jre/bin:/usr/local/java/lib:/usr/local/java/jre/lib/charsets.jar

ADD apache-tomcat-9.0.2 /usr/local/tomcat
RUN chmod 755 /usr/local/tomcat/bin/startup.sh
RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

EXPOSE 8080

CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]
```

构建镜像

```
[root@server04 tomcat]# docker build -t tomcat:new .
[root@server04  tomcat]#  docker   run  -d  -p  8080:8080  --name  tomcat-test  --restart=always  tomcat:new  #启动 tomcat 容器，设置关联端口
```

Web 访问 192.168.200.14:8080 端口



Dockerfile 构建 Redis 镜像

建立工作目录

```
[root@server04 tomcat]# mkdir   redis
[root@server04 tomcat]# cd   redis
[root@server04 redis]# vim Dockerfile

FROM centos:7
MAINTAINER  Crushlinux <crushlinux@163.com>

RUN yum -y install epel-release && yum -y install redis
RUN sed -i -e 's@bind 127.0.0.1@bind 0.0.0.0@g' /etc/redis.conf
RUN sed -i -e 's@protected-mode yes@protected-mode no@g'/etc/redis.conf
RUN echo "requirepass 123456" >> /etc/redis.conf

RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
EXPOSE 6379

CMD [ "/usr/bin/redis-server" , "/etc/redis.conf"]
```

构建镜像

```
[root@server04 redis]# docker build -t redis:new  .
[root@server04 redis]# docker run -d -p 6397:6397 --name redis-test --restart=always  redis:new
```

Linux 机器做测试

```
[root@server04 redis]# rpm -ivh epel-release-latest-7.noarch.rpm  #安装一个 repo 源
[root@server04 redis]# yum -y install redis  #安装 redis，使用它的客户端功能
[root@server04 redis]# redis-cli  -h localhost -a 123456  #如果可以连接上说明容器运行成功，时可用的
```

Docker 构建 mysql 镜像

建立工作目录

```
[root@server04 ~]# mkdir mysql
[root@server04 ~]# cd  mysql
```

编写 Dockerfile 文件

```
FROM centos:7
MAINTAINER  Crushlinux <crushlinux@163.com>

RUN yum install -y mariadb mariadb-server mariadb-devel   #安装 mariadb

ENV MARIADB_USER root   #定义用户
```

```
ENV MARIADB_PASS 123456  #定义面膜
ENV LC_ALL en_US.UTF-8   #

ADD db_init.sh /root/db_init.sh  #编写启动脚本

RUN chmod 775 /root/db_init.sh
RUN /root/db_init.sh

EXPOSE 3306   #指定端口

CMD ["mysqld_safe"]  #启动 mysql 服务
```

编写 mysql 初始化脚本

```
#!/bin/bash
mysql_install_db --user=mysql
sleep 3
mysqld_safe &
sleep 3
#mysqladmin -u "$MARIADB_USER" password "$MARIADB_PASS"
mysql -e "use mysql; grant all privileges on *.* to '$MARIADB_USER '@'%' identified by '$MARIADB_PASS' with
grant option;"
h=$(hostname)
mysql -e "use mysql; update user set password=password('$MARIADB_PASS') where user='$MARIADB_USER' and host='$h';"
mysql -e "flush privileges;"
```

测试环节

```
[root@server04mysql]# docker run -d -p 3306:3306 --name mysql-test --restart=always mysql:new  #启动 mysql 容器
[root@server04 mysql]# netstat -lntp | grep 3306  #查看端口
tcp    0   0 0.0.0.0:3306          0.0.0.0:*         LISTEN     8058/docker-prox
[root@server04 mysql]# mysql -h 192.168.200.14 -u root -p123456  #登录 mysql 库，登录成功，说可以使用
```

Docker 构建 lnmp 镜像

建立工作目录

```
[root@crushlinux ~]# mkdir lnmp
[root@crushlinux ~]# cd lnmp/
```

编写 Dockerfile 文件

```
FROM centos:7
MAINTAINER  Crushlinux <crushlinux@163.com>
RUN rpm -ivh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm

RUN yum -y install nginx
RUN rpm --rebuilddb && yum -y install mariadb-devel mariadb-server mariadb php php-fpm

RUN sed -i '/^user/s/nginx/nginx\ nginx/g' /etc/nginx/nginx.conf
RUN sed -i '10cindex index.php index.html index.htm ;' /etc/nginx/conf.d/default.conf
RUN sed -i '30,36s/#//' /etc/nginx/conf.d/default.conf
RUN sed -i '31s/html/\/usr\/share\/nginx\/html/' /etc/nginx/conf.d/default.conf
RUN sed -i '/fastcgi_param/s/scripts/usr\/share\/nginx\/html/' /etc/nginx/conf.d/default.conf



RUN sed -i '/^user/s/apache/nginx/g' /etc/php-fpm.d/www.conf
RUN sed -i '/^group/s/apache/nginx/g' /etc/php-fpm.d/www.conf

ADD db_init.sh /root/db_init.sh
RUN chmod 775 /root/db_init.sh
RUN /root/db_init.sh

ADD index.php /usr/share/nginx/html/index.php

RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

EXPOSE 80
EXPOSE 9000
EXPOSE 3306

ADD run.sh /run.sh
RUN chmod 775 /run.sh
CMD ["/run.sh"]
```

初始化数据库脚本 db_init.sh

```
[root@server04 lnmp]# vim db_init.sh
/usr/bin/mysqld_safe &
sleep 3
mysql -e "use mysql; grant all privileges on *.* to '$MARIADB_USER'@'%' identified by '$MARIADB_PASS' with grant option;"
h=$(hostname)
mysql -e "use mysql; update user set password=password('$MARIADB_PASS') where user='$MARIADB_USER' and host='$h';"
```

```
mysql -e "flush privileges;"
```

配置一个测试页 index.php

```
[root@server04 lnmp]# vim index.php
<?php
phpinfo();
?>
```

配置启动脚本 run.sh

```
[root@server04 lnmp]# vim run.sh
#!/bin/bash
/usr/sbin/nginx && /usr/sbin/php-fpm && /usr/bin/mysqld_safe
```

开始构建

```
[root@server04 lnmp]#  docker build -t lnmp:new  .
[root@server04 lnmp]# docker run -d -p 80:80 -p 3306:3306 -p 9000:9000 --name lnmp-test  --restart=always lnmp:new
```

web 访问 192.168.200.14

Dockerfile 面试题

CMD，ENTRYPOINT 与 RUN 命令的对比

CMD 命令时在容器启动后执行的命令。一个 dockerfile 可以有多个 CMD，但是只执行最后一个 CMD 生效，当容器启动时如果指定了命令，那么 CMD 的命令将被忽略

```
FROM alpine:latest
WORKDIR /workdir
ENV    name "Docker"
CMD    echo $name
```

生成新的镜像 secondtonone1/alpine-cmd

```
docker build -t secondtonone1/alpine-cmd .
```

生成后生成容器

```
docker run --rm --name cmd secondtonone1/alpine-cmd
```

可以看到输出 docker 了

接下来我们在容器启动时后边增加一个命令

```
docker run --rm -it --name cmd secondtonone1/alpine-cmd sh
```

此时进入了容器内部，执行了 sh 命令。Dockerfile 中的 cmd 被忽略了。

RUN 命令是在构建镜像时执行的命令，我们可以安装一些应用。

```
FROM ubuntu:18.04
WORKDIR /workdir
RUN    apt-get update
RUN    apt-get install -y net-tools
CMD    netstat
```

生成镜像

```
docker build -f Dockerfile -t cmd2 .
```

生成容器并启动

```
docker run -it --rm    cmd2
```

可以看到容器启动后调用了 cmd 命令 netstat

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
```

ENTRYPOINT 和 CMD 不同，他不会被 docker 启动后执行的命令覆盖

```
FROM ubuntu:18.04
WORKDIR /workdir
RUN    apt-get update
RUN    apt-get install -y net-tools
ENTRYPOINT    netstat
```

ENTRYPOINT 和 CMD 不同他不会被 docker 启动后执行的命令覆盖

```
FROM ubuntu:18.04
WORKDIR /workdir
RUN    apt-get update
RUN    apt-get install -y net-tools
ENTRYPOINT    netstat
```

生成镜像

```
docker build -f Dockerfile -t cmd3 .
```

生成容器并启动

```
docker run -it --rm    cmd3 /bin/bash
```

可以看到容器启动后并没有执行/bin/bash 命令，而是调用了 ENTRYPOINT 命令 netstat

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address                Foreign Address              State
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
```

RUN 和 CMD 支持参数形式命令

```
FROM ubuntu:18.04
WORKDIR /workdir
ENV    name "Docker"
RUN    ["/bin/bash", "-c", "apt-get update"]
RUN    ["/bin/bash", "-c", "apt-get install -y net-tools"]
CMD    ["/bin/bash","-c","echo Hello $name !"]
```

生成镜像

```
docker build -f ./Dockerfile -t cmd4 .
```

运行容器

```
docker run -it --rm cmd4
```

可以看到输出了 Hello, Docker!