一、Dockerfile：用来构造镜像文件，包含指令和说明

二、定制镜像

1、定制nginx镜像：完成后的镜像内有一个/usr/share/nginx/html/index.html文件

2、建立文件Dockerfile，vim打开

FROM nginx

RUN echo '这是一个本地构建的nginx镜像' > /usr/share/nginx/html/index.html



3、指令介绍

FROM：定制基于的镜像，nginx就是定制需要的基础镜像，后续操作基于nginx

RUN：用于执行后跟着的命令行命令

a）shell格式

RUN <命令行命令>

# <命令行命令> 等同于，在终端操作的 shell 命令。

b）exec格式：

RUN ["可执行文件", "参数1", "参数2"]

# 例如：

# RUN ["./test.php", "dev", "offline"] 等价于 RUN ./test.php dev offline

每一条命令都会在基础上新建一层，过多无异于的层，会造成镜像膨胀过大，例如

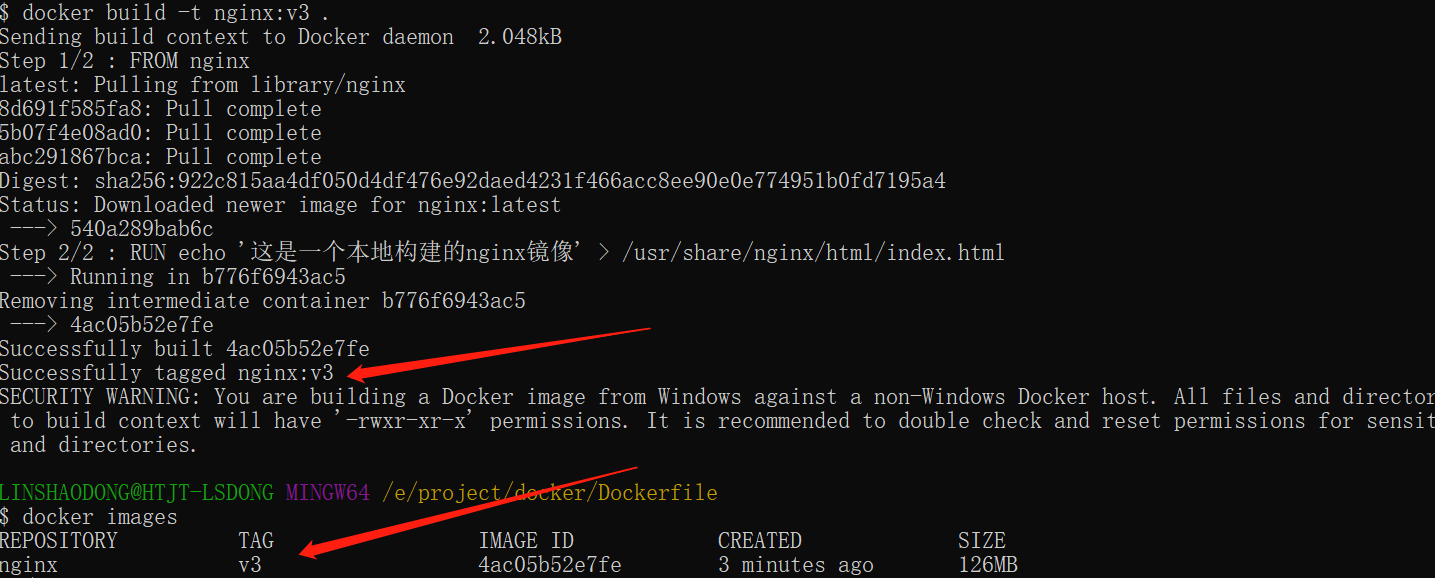
FROM centos  
RUN **yum install** **wget**  
RUN **wget** -O redis.tar.gz "http://download.redis.io/releases/redis-5.0.3.tar.gz"  
RUN **tar** -xvf redis.tar.gz  
以上执行会创建 3 层镜像。可简化为以下格式：  
FROM centos  
RUN **yum install** **wget** \  
    **&&** **wget** -O redis.tar.gz "http://download.redis.io/releases/redis-5.0.3.tar.gz" \  
    **&&** **tar** -xvf redis.tar.gz

使用&&符号连接命令，只会创建一层镜像

三、构建镜像：在Dockerfile文件存放的目录下，执行构建动作

注：最后.代表本次执行上下文路劲

$ docker build -t nginx:test .



四、上下文路径

1、介绍：上下文路径，是指 docker 在构建镜像，有时候想要使用到本机的文件（比如复制），docker build 命令得知这个路径后，会将路径下的所有内容打包。

2、解析：由于 docker 的运行模式是 C/S。我们本机是 C，docker 引擎是 S。实际的构建过程是在 docker 引擎下完成的，所以这个时候无法用到我们本机的文件。这就需要把我们本机的指定目录下的文件一起打包提供给 docker 引擎使用。

如果未说明最后一个参数，那么默认上下文路径就是 Dockerfile 所在的位置。

五、指令详解

1、复制指令：copy。从上下文目录中复制文件或目录到容器里指定路劲

a）格式

COPY [--chown=<user>:<group>] <源路径1>... <目标路径>

COPY [--chown=<user>:<group>] ["<源路径1>",... "<目标路径>"]

[--chown=<user>:<group>]：可选参数，复制后，改变容器中所属组和所有者

<源路径>：源文件或源目录，可以使用通配符，但需要满足Go的filepath.Match

COPY hom\* /mydir/

COPY hom?.txt /mydir/

<目标路径>：容器内指定的路径，如果不存在，会创建

2、复制指令：add，同copy使用格式一致（相同需求下，官方推荐copy），功能类似

a）优点：在执行 <源文件> 为 tar 压缩文件的话，压缩格式为 gzip, bzip2 以及 xz 的情况下，会自动复制并解压到 <目标路径>。

b）缺点：在不解压的前提下，无法复制 tar 压缩文件。会令镜像构建缓存失效，从而可能会令镜像构建变得比较缓慢。具体是否使用，可以根据是否需要自动解压来决定。

3、运行指令：cmd，类似run

a）cmd：docker run 时运行

b）run：docker build 时运行

c）作用：为启动容器指定默认允许程序，程序运行介绍，容器介绍。可以被docker run 命令行参数中指定要运行的程序覆盖

d）注意：存在多个cmd，仅最后一个生效

e）格式

CMD <shell 命令>

CMD ["<可执行文件或命令>","<param1>","<param2>",...]

CMD ["<param1>","<param2>",...] # 该写法是为 ENTRYPOINT 指令指定的程序提供默认参数

4、运行指令：entrypoint，类似cmd，但不会被docker run 指令覆盖，命令行参数会被当做参数送给entrypoint指令指定的程序

a）docker run 添加了 – entrypoint 就会被覆盖

b）优点：执行docker run 的时候可以指定entrypoint需要的参数

c）注意：存在多个，最后一个生效

e）格式

ENTRYPOINT ["<executeable>","<param1>","<param2>",...]

f）搭配cmd命令：一般变参才会使用cmd，这里cmd等于给entrypoint传参

FROM nginx

ENTRYPOINT ["nginx", "-c"] # 定参

CMD ["/etc/nginx/nginx.conf"] # 变参

不传参运行

$ docker run nginx:test

容器内会默认运行以下命令，启动主进程。

nginx -c /etc/nginx/nginx.conf

传参运行

$ docker run nginx:test -c /etc/nginx/new.conf

容器内会默认运行以下命令，启动主进程(/etc/nginx/new.conf:假设容器内已有此文件)

nginx -c /etc/nginx/new.conf

5、环境变量：env，定义环境变量，使用环境变量，和shell用法相似

a）格式：

ENV <key> <value>

ENV <key1>=<value1> <key2>=<value2>...

例：

ENV NODE\_VERSION 7.2.0

RUN curl -SLO "https://nodejs.org/dist/v$NODE\_VERSION/node-v$NODE\_VERSION-linux-x64.tar.xz" \

&& curl -SLO "https://nodejs.org/dist/v$NODE\_VERSION/SHASUMS256.txt.asc"

6、构建参数：arg，和env作用一致。但作用域不一样，arg只在Dockerfile中有效，也就是说只有docker build过程中有效

a）构建命令 docker build 中可以用 --build-arg <参数名>=<值> 来覆盖

b）格式

ARG <参数名>[=<默认值>]

7、定义匿名数据卷：volume，启动容器忘记挂在数据卷，会自动挂载到匿名卷

a）作用：避免重要数据，因重启丢失。避免容器不断变大

b）格式

VOLUME ["<路径1>", "<路径2>"...]

VOLUME <路径>

8、声明端口：expose

a）作用：帮助镜像使用者理解这个镜像服务的守护端口，以方便配置映射。在运行时使用随机端口映射时，也就是 docker run -P 时，会自动随机映射 EXPOSE 的端口

b）格式

EXPOSE <端口1> [<端口2>...]

9、指定工作目录：workdir，会在镜像每一层都存在（指定的目录必须是提前存在的）

a）docker build构建中，每个run都会创建一层，只有workdir创建的目录会一直存在

b）格式

WORKDIR <工作目录路径>

10、指定后续用户和用户组（必须提前存在）：user

a）格式：

USER <用户名>[:<用户组>]

11、指定命令监控docker容器服务运行状态：healthcheck

a）格式

HEALTHCHECK [选项] CMD <命令>：设置检查容器健康状况的命令

HEALTHCHECK NONE：如果基础镜像有健康检查指令，使用这行可以屏蔽掉其健康检查指令

HEALTHCHECK [选项] CMD <命令> : 这边 CMD 后面跟随的命令使用，可以参考 CMD 的用法。

12、延迟构建命令的执行（当前构建不会执行，下一个会使用）：onbuild

a）格式

ONBUILD <其它指令>