dockerfile用于生成docker镜像

**使用 Dockerfile 定制镜像**

**Dockerfile**

在一个空目录下，新建一个名为 Dockerfile 文件，并在文件内添加以下内容：

FROM nginx

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

**生成镜像**

执行如下命令生成镜像

$ docker build -t nginx:test .

.是Dockerfile 文件所在路径，称为上下文路径，生成镜像时会打包该路径下的所有文件，所以无关的文件不要放到此目录

**指令详解**

**FROM**

定制的镜像所依赖的基础镜像，示例：

FROM nginx

**RUN**

生成镜像时，在容器内执行的shell命令，示例：

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

每行RUN命令都是独立的进程，都会生成一个新的容器，后一行RUN是在前一行RUN的容器上执行的

来看一段代码：

RUN cd /app

RUN echo "hello"> world.txt

world.txt 文件在 /app 目录下吗？答案是不在，正确的做法是

RUN cd /app && echo "hello"> world.txt

**COPY**

生成镜像时，复制上下文路径中的目录或文件到容器内，格式如下：

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

示例：

COPY hom?.txt /mydir/

**CMD**

容器运行时，需要执行的shell命令，格式如下：

CMD <shell 命令>

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

CMD ["<param1>","<param2>",...]  # 该写法是为 ENTRYPOINT 指令提供默认参数

注：CMD 指令可被 docker run 命令行参数所覆盖，如果 Dockerfile 中如果存在多个 CMD 指令，仅最后一个生效。

**ENTRYPOINT**

类似CMD，但其不会被docker run 命令行参数覆盖，格式如下：

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

一般的用法如下：

FROM nginx

ENTRYPOINT ["nginx", "-c"] # 定参

CMD ["/etc/nginx/nginx.conf"] # 变参

当docker run不指定参数是，默认运行的命令为：

$ nginx -c /etc/nginx/nginx.conf

当docker run指定参数时，CMD被覆盖，执行的命令为：

$ nginx -c 参数

注：如果 Dockerfile 中如果存在多个 ENTRYPOINT 指令，仅最后一个生效。

**ENV**

设置环境变量，定义了环境变量，那么在后续的指令中，就可以使用这个环境变量。

通过ENV设置的环境变量，容器内也会存在该环境变量

格式：

ENV <key> <value>

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

示例：

ENV NODE\_VERSION 7.2.0

RUN curl -SLO "/v$NODE\_VERSION/linux-x64.tar.xz"

**VOLUME**

数据挂载，将匿名的主机目录挂载到容器的目录，可以防止因为容器的重启导致数据丢失，与docker run -v \*\*:\*\* 功能一致，但dockerfile是镜像，无法指定对应的主机目录

示例（为容器的/data1，/data2建立挂载点，对应的主机目录是随机匿名的）：

VOLUME ["/data1","/data2"]

**EXPOSE**

声明端口，会将容器的端口随机映射到主机端口，与docker run -p \*\*:\*\* 功能一致，但dockerfile是镜像，无法指定对应的主机端口

示例：

EXPOSE [80]

**WORKDIR**

指定工作目录，没有docker会创建

示例，你想之后的RUN都在 /app 下工作，可以执行如下命令

WORKDIR /app

RUN echo "hello"> world.txt

RUN echo "world"> world.txt