使用DockerFile创建Docker Images

指令说明：

指令	说明
FROM	指定所创建镜像的基础镜像
MAINTAINER	指定维护者信息
RUN	运行命令
CMD	指定启动容器时默认执行的命令
LABEL	指定生成镜像的元数据标签信息
EXPOSE	声明镜像内服务所监听的端口
ENV	指定环境变量
ADD	赋值指定的<src>路径下的内容到容器中的<dest>路径下，<src>可以为URL；如果为tar文件，会自动解压到<dest>路径下
COPY	赋值本地主机的<scr>路径下的内容到容器中的<dest>路径下；一般情况下推荐使用COPY而不是ADD
ENTRYPOINT	指定镜像的默认入口
VOLUME	创建数据挂载点
USER	指定运行容器时的用户名或UID
WORKDIR	配置工作目录
ARG	指定镜像内使用的参数(例如版本号信息等)
ONBUILD	配置当前所创建的镜像作为其他镜像的基础镜像时，所执行的创建操作的命令
STOPSIGNAL	容器退出的信号
HEALTHCHECK	如何进行健康检查
SHELL	指定使用SHELL时的默认SHELL类型

dockerfile创建命令

dockerfile创建docker镜像
docker build -t build_repo/first_image /tmp/docker_builder
生成镜像标签为 ：build_repo/first_image
指定Dockerfile所在路径为： /tmp/docker_builder/

dockerfile案例：

```
# cat Dockerfile   ## bat dockerfile

FROM centos:7.2.1511

MAINTAINER houfei@sunline.cn

#COPY  jdk1.8.0_131 /usr/local/jdk1.8.0_131
COPY  cbs-ltts-full-1.6.1.0-SNAPSHOT /app/cbs-bat

#ENV JAVA_HOME=/usr/local/jdk1.8.0_131
#ENV PATH=$JAVA_HOME/bin:$PATH
#ENV export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
RUN yum install java-1.8.0-openjdk -y
#RUN  yum -y install kde-l10n-Chinese telnet && yum -y reinstall
glibc-common &&  yum clean all  && localedef -c -f UTF-8 -i zh_CN
zh_CN.utf8

ENV LC_ALL en_US.UTF-8
ENV LANG en_US.UTF-8
#ENV LC_ALL "zh_CN.UTF-8"

#RUN rm -rf /usr/local/tomcat/webapps/ROOT
#COPY ROOT /usr/local/tomcat/webapps/ROOT

LABEL version="v1.6.1.0"
LABEL description="1.6.1.0"
EXPOSE 8080

CMD /app/cbs-bat/bin/ltts start bat && tail -100f /etc/profile
```

```
# cat Dockerfile   # onl dockerfile

FROM centos:7.2.1511

MAINTAINER houfei@sunline.cn

#COPY  jdk1.8.0_131 /usr/local/jdk1.8.0_131
COPY  cbs-ltts-full-1.6.1.0-SNAPSHOT /app/cbs-onl

#ENV JAVA_HOME=/usr/local/jdk1.8.0_131
#ENV PATH=$JAVA_HOME/bin:$PATH
#ENV export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
RUN yum install java-1.8.0-openjdk -y
#RUN  yum -y install kde-l10n-Chinese telnet && yum -y reinstall
glibc-common &&  yum clean all  && localedef -c -f UTF-8 -i zh_CN
zh_CN.utf8

ENV LC_ALL en_US.UTF-8
ENV LANG en_US.UTF-8
#ENV LC_ALL "zh_CN.UTF-8"

#RUN rm -rf /usr/local/tomcat/webapps/ROOT
#COPY ROOT /usr/local/tomcat/webapps/ROOT

LABEL version="v1.6.1.0"
LABEL description="1.6.1.0"
EXPOSE 8080

CMD /app/cbs-onl/bin/ltts start onl && tail -100f /etc/profile
```

```
# cat Dockerfile    # dockerfile

FROM centos:7.2.1511

MAINTAINER houfei@sunline.cn

#COPY  jdk1.8.0_131 /usr/local/jdk1.8.0_131
RUN yum install java-1.8.0-openjdk -y
COPY  tomcat /app/bds

#ENV JAVA_HOME=/usr/local/jdk1.8.0_131
#ENV PATH=$JAVA_HOME/bin:$PATH
#ENV export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
RUN rm -rf /app/bds/webapps/ROOT
COPY ROOT /app/bds/webapps/ROOT

LABEL version="v1.6.1.0"
LABEL description="1.6.1.0"
EXPOSE 8080

CMD /app/bds/bin/catalina.sh run
```

常用指令：

RUN

运行指定命令。

格式为：RUN<command>或RUN ["executable","param1","param2"]。

注意：

后一个指令会被解析为json数组，所以必须使用双引号。

前者默认将在shell终端中运行命令，即/bin/sh -c；后者则使用exec执行，不会启动shell环境。

指定使用其他终端类型可以通过第二种方式实现，例如：

RUN ["/bin/bash","-c","echo hello"]

每条RUN指令将在当前镜像的基础上执行指定命令，并提交为新的镜像。当命令较长时可以使用\换行。例如：

```
RUN apt-get update \
&& apt-get install -y libsnappy-dev zliblg-dev libbz2-dev \
&& rm -rf /var/cache/apt
```

CMD

CMD指令用来指定启动容器时默认执行的命令。它支持三种格式：

```
1.CMD ["executable","param1","param2"] exec
2.CMD param1 param2 /bin/sh
3.CMD ["param1","param2"] ENTRYPOINT
DockerfileCMD(run)CMD
```

LABEL

LABEL指令用来生成用于生成镜像的元数据的标签信息。

格式为：LABEL <key>=<value> <key>=<value> <key>=<value> ...。

例如：

```
LABEL version="1.0"
LABEL description="This text illustrates \ that label-values can span
multiple lines."
```

EXPOSE

声明镜像内服务所监听的端口。

格式为：EXPOSE <port> [<port>...]

例如：

```
EXPOSE 22 80 443 3306
```

注意：

该命令只是起到声明租用，并不会自动完成端口映射。

在容器启动时需要使用-P(大写P)，Docker主机会自动分配一个宿主机未被使用的临时端口转发到指定的端口；使用-p(小写p)，则可以具体指定哪个宿主机的本地端口映射过来。

ENV

指定环境变量，在镜像生成过程中会被后续RUN指令使用，在镜像启动的容器中也会存在。

格式为：ENV <key><value>或ENV<key>=<value>...。

例如：

```
ENV GOLANG_VERSION 1.6.3
ENV GOLANG_DOWNLOAD_RUL
https://golang.org/dl/go$GOLANG_VERSION.linux-amd64.tar.gz
ENV GOLANG_DOWNLOAD_SHA256
cdd5e08530c0579255d6153b08fdb3b8e47caabbe717bc7bcd7561275a87aeb

RUN curl -fssL "$GOLANG_DOWNLOAD_RUL" -o golang.tar.gz && echo
"$GOLANG_DOWNLOAD_SHA256 golang.tar.gz" | sha256sum -c - && tar -C
/usr/local -xzf golang.tar.gz && rm golang.tar.gz

ENV GOPATH $GOPATH/bin:/usr/local/go/bin:$PATH

RUN mkdir -p "$GOPATH/bin" && chmod -R 777 "$GOPATH"
```

指令指定的环境变量在运行时可以被覆盖掉，如docker run --env <key>=<value> built_image。

ADD

该指令将复制指定的<src>路径下的内容到容器中的<dest>路径下。

格式为：ADD<src> <dest>

其中<src>可以使Dockerfile所在目录的一个相对路径(文件或目录)，也可以是一个URL，还可以是一个tar文件(如果是tar文件，会自动解压到<dest>路径下)。

<dest>可以使镜像内的绝对路径，或者相当于工作目录(WORKDIR)的相对路径。路径支持正则表达式，例如：

```
ADD *.c /code/
```

COPY

复制本地主机的<src>(为Dockerfile所在目录的一个相对路径、文件或目录)下的内容到镜像中的<dest>下。目标路径不存在时，会自动创建。路径同样支持正则。

格式为：COPY <src> <dest>

当使用本地目录为源目录时，推荐使用COPY。

ENTRYPOINT

指定镜像的默认入口命令，该入口命令会在启动容器时作为根命令执行，所有传入值作为该命令的参数。

支持两种格式：

```
1.ENTRYPOINT ["executable","param1","param2"] (exec)
2.ENTRYPOINT command param1 param2(shell)
```

此时，CMD指令指定值将作为根命令的参数。

每个Dockerfile中只能有一个ENTRYPOINT，当指定多个时，只有最后一个有效。

在运行时可以被--entrypoint参数覆盖掉，如docker run --entrypoint。

VOLUME

创建一个数据卷挂载点。

格式为：VOLUME ["/data"]

可以从本地主机或者其他容器挂载数据卷，一般用来存放数据库和需要保存的数据等。

USER

指定运行容器时的用户名或UID，后续的RUN等指令也会使用特定的用户身份。

格式为：USER daemon

当服务不需要管理员权限时，可以通过该指令指定运行用户，并且可以在之前创建所需要的用户。例如：

```
RUN groupadd -r nginx && useradd -r -g nginx nginx
```

要临时获取管理员权限可以用gosu或者sudo。

WORKDIR

为后续的RUN、CMD和ENTRYPOINT指令配置工作目录。

格式为：WORKDIR /path/to/workdir。

可以使用多个WORKDIR指令，后续命令如果参数是相对的，则会基于之前命令指定的路径。例如：

```
WORKDIR /a
WORKDIR b
WORKDIR c
RUN pwd
/a/b/c
```

ARG

指定一些镜像内使用的参数(例如版本号信息等)，这些参数在执行docker build命令时才以--build-arg<varname>=<value>格式传入。

格式为：ARG<name>[=<default value>]。

则可以用docker build --build-arg<name>=<value>来指定参数值。

ONBUILD

配置当所创建的镜像作为其他镜像的基础镜像的时候，所执行创建操作指令。
格式为：ONBUILD [INSTRUCTION]。
例如Dockerfile使用如下的内容创建了镜像image-A：

```
[...]
ONBUILD ADD . /app/src
ONBUILD RUN /usr/local/bin/python-build --dir /app/src
[...]
```

如果基于image-A镜像创建新的镜像时，新的Dockerfile中使用FROM image-A指定基础镜像，会自动执行ONBUILD指令的内容，等价于在后面添加了两条指令：

```
FROM image-A

# Automatically run the following
ONBUILD ADD . /app/src
ONBUILD RUN /usr/local/bin/python-build --dir /app/src
```

使用ONBUILD指令的镜像，推荐在标签中注明，例如：ruby:1.9-onbuild。

STOPSIGNAL
指定所创建镜像启动的容器接收退出的信号值。例如：

```
STOPSIGNAL singnal
```

HEALTHCHECK
配置所启动容器如何进行健康检查(如何判断是否健康)，自Docker 1.12开始支持。
格式有两种：

```
1.HEALTHCHECK [OPTIONS] CMD command 0
2.HEALTHCHECK NONE :
```

[OPTION]支持：

```
1.--inerval=DURATION (30s)
2.--timeout=DURATION (30s)
3.--retries=N  (3)
```

SHELL
指定其他命令使用shell时的默认shell类型。
格式为： SHELL ["executable","parameters"]
默认值为 ["bin/sh","-c"]。
注意：
对于Windows系统，建议在Dockerfile开头添加# escape=`来指定转移信息。