Dockerfile

1、Dockerfile 文件常见指令

① FROM

> 格式为：FROM 或FROM :
> 使用哪个基础镜像启动构建流程，第一条指令必须为FROM指令，在同一个Dockerfile中创建多个镜像时，可以使用多个FROM指令（每个镜像一次）

```
FROM centos
```

构建镜像

```
docker build -t centos01 .
```

启动容器

```
docker run -it --name cos centos01
```

② MAINTAINER（被废弃）

> 格式为：MAINTAINER
> 指定维护者信息，类似javadoc注释中的@author,用于声明创作者，一般都放在文件比较靠上的位置

```
FROM centos
MAINTAINER shadow
```

构建镜像

```
docker build -t centos02 .
```

启动容器

```
docker run -it --name cos centos02
```

③ RUN

> 两种格式：
>
> - shell 格式 ： RUN
> - exec 格式：RUN ["executable"，"paraml"，"param2"]
>   **构建镜像时运行的命令**
>   每条RUN指令将在当前镜像基础上执行指定命令，当命令较长时可以使用""来换行

```
FROM centos
MAINTAINER shadow
#RUN yum install -y vim # shell 格式
RUN ["yum","install","-y","vim"] # exec 格式
#RUN ["/bin/bash","-c","yum install -y vim"]
```

构建镜像

```
docker build -t centos03 .
```

④ EXPOSE

> 格式为：EXPOSE [...]
> 告诉Docker服务端容器暴露的端口号，供互联系统使用

```
FROM centos
MAINTAINER shadow
RUN ["yum","install","-y","vim"]
EXPOSE 80
```

构建镜像

```
docker build -t centos04 .
```

⑤ ENV

> 格式为：ENV
> 指定一个环境变量，在后续的指令中可以使用，并在容器运行时保持

```
FROM centos
MAINTAINER shadow
RUN ["yum","install","-y","vim"]
EXPOSE 80
ENV name shadow
```

构建镜像

```
docker build -t centos05 .
```

⑥ WORKDIR

> 格式为：WORKDIR
> 为后续的RUN、CMD、ENTRYPOINT指令配置工作目录；可以使用多个WORKDIR指令，后续命令如果参数是相对路径，则会基于之前命令指定的路径

```
FROM centos
MAINTAINER shadow
RUN ["yum","install","-y","vim"]
EXPOSE 80
ENV name shadow
WORKDIR /$name
```

构建镜像

```
docker build -t centos06 .
```

⑦ VOLUME

> 格式为：VOLUME []
> 创建一个可以从本地主机或其他容器挂载的挂载点， 一般用来存放数据库和需要保持的数据等

```
FROM centos
MAINTAINER shadow
RUN ["yum","install","-y","vim"]
EXPOSE 80
ENV name shadow
WORKDIR /$name
VOLUME /$name
```

构建镜像

```
docker build -t centos07 .
```

⑧ ADD

> 格式为：ADD
> 将复制指定的到容器中的；其中可以是 Dockerfile 所在目录的一个相对路径：也可以是一个URL；还可以是一个tar文件（自动解压为目录）

```
FROM centos
MAINTAINER shadow
RUN ["yum","install","-y","vim"]
EXPOSE 80
ENV name shadow
WORKDIR /$name
VOLUME /$name
ADD apache-maven-3.6.2-bin.tar.gz /usr/local
```

构建镜像

```
docker build -t centos08 .
```

⑨ COPY

> 格式为： COPY
> 复制本地主机的 (Dockerfile所在目录的相对路径）到容器中的；当使用本地目录为源目录时，推荐使用COPY

```
FROM centos
MAINTAINER shadow
RUN yum install -y vim
EXPOSE 80
ENV name shadow
WORKDIR /$name
VOLUME /$name
# 将 apache-maven-3.6.2-bin.tar.gz 包放在同级目录下
COPY apache-maven-3.6.2-bin.tar.gz /usr/local
# 构建镜像时运行命令，解压缩maven，并创建链接
```

```
RUN cd /usr/local &&  \
  tar -zxvf apache-maven-3.6.2-bin.tar.gz && \
  rm -f apache-maven-3.6.2-bin.tar.gz && \
  ln -s /usr/local/apache-maven-3.6.2/bin/mvn /usr/bin/mvn && \
  ln -s /usr/local/apache-maven-3.6.2 /usr/local/apache-maven && \
  mkdir -p /usr/local/apache-maven/repo
# maven 配置文件放在同级目录下,修改好 mirror
COPY settings.xml /usr/local/apache-maven/conf/settings.xml
```

构建镜像

```
docker build -t centos09 .
```

⑩ CMD

> 三种格式：
>
> - CMD ["executable","paraml","param2"] 使用exec执行，推荐方式
> - CMD command paraml param2在/bin/sh中执行，提供给需要交互的应用
> - CMD ["paraml"，"param2"] 提供给ENTRYPOINT的默认参数
>
> **启动/运行容器时运行的命令**
> 指定启动容器时执行的命令，每个Dockerfile只能有一条CMD命令。如果指定了多条命令，则只有最后一条会被执行；如果用户启动容器时指定了运行的命令，则会覆盖CMD指定的命令

```
FROM centos
MAINTAINER shadow
# 镜像构建时执行
RUN ["yum","install","-y","vim"]
ENV name shadow
CMD ["echo","$name"]
```

构建镜像

```
docker build -t centos10 .
```

⑪ ENTRYPOINT

> 两种格式：
>
> - ENTRYPOINT["executable","paraml","param2"]
> - ENTRYPOINT command paraml param2
>
> 配置容器启动后执行的命令，并且不可被Docker RUN叫提供的参数覆盖；每个Dockerfile中只能有一个ENTRYPOINT，当指定多个时，只有最后一个生效

```
FROM centos
MAINTAINER shadow
RUN ["yum","install","-y","vim"]
ENV name shadow
ENTRYPOINT ["top","-b"]
CMD ["-c"]
```

构建镜像

```
docker build -t centos11 .
```

⑫ **USER**

> 格式为：USER daemon
> 指定运行容器时的用户名或UID，后续的RUN也会使用指定用户；当服务不需要管理员权限时，可以通过该命令指定运行用户

```
# 基础镜像 jenkins/jnlp-slave:latest
FROM jenkins/jnlp-slave:latest
MAINTAINER shadow
# 切换到 root 账户进行操作
USER root
# 安装 maven
COPY apache-maven-3.6.2-bin.tar.gz .
# 将 apache-maven-3.6.2-bin.tar.gz 包放在同级目录下
RUN tar -zxf apache-maven-3.6.2-bin.tar.gz && \
  mv apache-maven-3.6.2 /usr/local && \
  rm -f apache-maven-3.6.2-bin.tar.gz && \
  ln -s /usr/local/apache-maven-3.6.2/bin/mvn /usr/bin/mvn && \
  ln -s /usr/local/apache-maven-3.6.2 /usr/local/apache-maven && \
  mkdir -p /usr/local/apache-maven/repo
# maven 配置文件放在同级目录下
COPY settings.xml /usr/local/apache-maven/conf/settings.xml
# 宿主机创建 jenkins 用户
USER jenkins
```

构建镜像

```
docker build -t centos12 .
```

⑬ **ONBUILD**

> 格式为：ONBUILD [INSTRUCTION]
> 配置当创建的镜像作为其他新创建镜像的基础镜像时，所执行的操作指令

```
FROM centos
MAINTAINER shadow
ONBUILD RUN ["echo","build centos"]
```

构建镜像

```
docker build -t centos13 .
```

使用 centos13 作为基础镜像

```
FROM centos13
MAINTAINER shadow
```

2、镜像构建

1. 创建 Dockerfile 文件

2. 执行构建命令

```
# 在 Dockerfile 所在目录下执行
docker build -t name:tag .
# 任意位置 Dockerfile
docker build -t name:tag -f /path/Dockerfile /path
```

3、docker build 构建流程

- (1)提取Dockerfile(evaluator.go/RUN)。
- (2)将Dockerfile按行进行分析(parser/parser.go/Parse)，每行第一个单词(如CMD、FROM等)叫作command。根据command，将之后的字符串用对应的数据结构进行接收。
- (3)根据分析的command，在dispatchers.go中选择对应的函数进行处理(dispatchers.go)。
- (4)处理完所有的命令，如果需要打标签，则给最后的镜像打上tag，结束。

4、Dockerfile 逆向

通过docker history image可以看到该镜像的历史来源。即使没有Dockerfile，也可以通过history来逆向产生Dockerfile