# Docker 必修课程 Dockerfile

## Dockerfile 简介

Dockfile 是一种被Docker程序解释的脚本文件，Dockerfile由一条一条的指令组成，每条指令对应Linux下面的一条命令，Docker程序将这些Dockerfile指令翻译真正的Linux命令；Dockerfile有自己书写格式和支持的命令，Docker程序解决这些命令间的依赖关系，类似于Makefile，Docker程序将读取Dockerfile，根据指令生成定制的image。

Dockerfile的指令是忽略大小写的，建议使用大写，使用 # 作为注释，每一行只支持一条连续的指令，每条指令可以携带多个参数。

Dockerfile 是一个文本文件，其内包含了一条条的指令(Instruction)，每一条指令构建一层， 因此每一条指令的内容，就是描述该层应当如何构建。

## Dockerfile 主要构成

Dockerfile 分为四部分：

* 基础镜像信息 (FROM)

* 维护者信息 (LABEL)，不推荐使用 (MAINTAINER)

* 镜像操作指令 (RUN)

* 容器启动时执行指令 (CMD)

首先看个例子

# stage 1  
FROM node-alpine as builder  
  
LABEL email=yangpeng2468@gmail.com  
  
RUN apk add --no-cache make gcc python-dev  
  
ARG NODE\_ENV  
ENV NODE\_ENV=${NODE\_ENV}  
  
USER node  
  
ADD --chown=node:node . /app  
  
WORKDIR /app  
  
RUN npm install \  
 && npm run build  
  
# stage 2  
FROM nginx:alpine  
  
COPY --from=builder /app/public /app/public  
COPY nginx.conf /etc/nginx/conf.d/default.conf  
  
EXPOSE 8080  
CMD ["nginx", "-g", "daemon off;"]

## Dockerfile 指令详解

#### FROM

格式为 FROM <image> 或 FROM <image>:<tag>

第一条指令必须为 FROM 指令。并且，如果在同一个 Dockerfile 中创建多个阶段时，可以使用多个 FROM 指令（每个阶段一次）

#### MAINTAINER

格式为 MAINTAINER <name>，指定维护者信息。已弃用，推荐使用 LABEL

#### LABEL

给构建的镜像打标签。格式：

#### RUN

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

推荐 RUN 把所有需要执行的 shell 命令写一行，例如:

RUN mkdir /app && \  
 echo "Hello World!" && \  
 touch /tmp/testfile

不推荐，如果 RUN 写多行会增加 docker image 体积。例如：

RUN mkdir /app  
RUN echo "Hello World!"  
RUN touch /tmp/testfile

#### CMD

支持三种格式

CMD ["executable","param1","param2"] 使用 exec 执行，推荐方式；

CMD command param1 param2 在 /bin/sh 中执行，提供给需要交互的应用；

CMD ["param1","param2"] 提供给 ENTRYPOINT 的默认参数；

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，只有最后一条会被执行。

#### EXPOSE

格式为 EXPOSE <port> [<port>...]

声明 Docker 服务端容器暴露的端口号，供外部系统使用。在启动容器时需要通过 -p指定端口号

#### ENV

格式为 ENV <key> <value>。 指定一个环境变量，会被后续 RUN 指令使用，并在容器运行时保持

#### ADD

格式为 ADD <src> <dest> ，在 docker ce 17.09以上版本支持 格式为 ADD --chown=<user>:<group> <src> <dest>

#### COPY

格式为 COPY <src> <dest>，在 docker ce 17.09以上版本支持 格式为 COPY --chown=<user>:<group> <src> <dest>

#### ENTRYPOINT

支持两种格式：

ENTRYPOINT ["executable", "param1", "param2"]

ENTRYPOINT command param1 param2（shell中执行）

#### VOLUME

格式为 VOLUME ["/data"]

创建一个可以从本地主机或其它容器挂载的挂载点，用来保持数据不被销毁

#### 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

容器不推荐使用 root 权限

#### WORKDIR

格式为 WORKDIR /path/to/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。例如

WORKDIR /a  
WORKDIR b  
WORKDIR c  
RUN pwd

则最后输出路径为 /a/b/c

#### ONBUILD

为他人做嫁衣，格式为 ONBUILD [INSTRUCTION]

配置当前所创建的镜像作为其它新创建镜像的基础镜像时，所执行的操作指令

#### HEALTHCHECK

健康检查，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 支持下列选项：

--interval=<间隔>：两次健康检查的间隔，默认为30秒；

--timeout=<时长>：健康检查命令运行超时时间，如果超过这个时间，本次健康检查就被 视为失败，默认30 秒；

--retries=<次数>：当连续失败指定次数后，则将容器状态视为unhealthy，默认3 次和CMD ,ENTRYPOINT一样， HEALTHCHECK 只可以出现一次，如果写了多个，只有最后一个生效

#### ARG

构建参数，格式：ARG<参数名>[=<默认值>]

构建参数 和 ENV的 效果一样，都是设置环境变量。所不同的是，ARG所设置的构建环境的环境变量，在将来容器运行时是不会存在这些环境变量的。但是不要因此就使用ARG保存密码之类的信息，因为docker history还是可以看到所有值的。

## 创建镜像

编写完 Dockerfile 后，通过 docker build 命令创建 docker image

例如：

$ docker build -t yangpeng2468/test:v1 . -f Dockerfile  
  
# -t 声明 docker image 名称  
# . 把当前目录加入到镜像中  
# -f 指定 Dockerfile 文件，默认为 Dockerfile 名称，如果是其它名称，需要使用 -f 来指定  
# --no-cache 不使用历史缓存

## 参考链接

[Docker-必修课程-Dockerfile](https://www.yp14.cn/2019/12/01/Docker-%E5%BF%85%E4%BF%AE%E8%AF%BE%E7%A8%8B-Dockerfile/)