Dockerfile定义镜像内容，是自动构建 docker 镜像的配置文件。Dockerfile 是由一行行命令语句组成，并且支持以#开头的注释行。

一般来说，我们可以将 Dockerfile 分为四个部分：

跨越多行指令 \ 以及`

基础镜像(父镜像)信息指令 FROM

维护者信息指令 MAINTAINER

镜像操作指令 RUN 、 EVN 、 ADD 和 WORKDIR 等

容器启动指令 CMD 、 ENTRYPOINT 和 USER 等

Dockerfile 中所有的命令都是以下格式：INSTRUCTION argument ，指令 (INSTRUCTION) 不分大小写，但是推荐大写。

FROM命令

FROM 是用于指定基础的 images ，一般格式为 FROM <image> or FORM <image>:<tag> ，所有的 Dockerfile 都用该以 FROM 开头，FROM 命令指明 Dockerfile 所创建的镜像文件以什么镜像为基础，FROM 以后的所有指令都会在 FROM 的基础上进行创建镜像。可以在同一个 Dockerfile 中多次使用 FROM 命令用于创建多个镜像。比如我们要指定 python 2.7 的基础镜像，我们可以像如下写法一样：

FROM python:2.7

MAINTAINER命令

MAINTAINER 是用于指定镜像创建者和联系方式，一般格式为 MAINTAINER <name> 。例如

MAINTAINER Test <[Test8@gmail.com](mailto:Test8@gmail.com)>

COPY命令

COPY 是用于复制宿主主机的 <src> (为 Dockerfile 所在目录的相对路径) 到容器中的 <dest>。

当使用本地目录为源目录时，推荐使用 COPY 。一般格式为 COPY <src><dest> 。例如我们要拷贝主机当前目录到容器中的 /app 目录下，我们可以这样操作：

COPY . /app

COPY ./index.html /usr/share/nginx/html/index.html # 拷贝宿主当前目录下的index.html到image目录/usr/share/nginx/html下并且名为index.html, 文件名不可缺少

ADD命令：类似于COPY命令，ADD命令从URL中获取资源

WORKDIR命令

WORKDIR 用于配合 RUN，CMD，ENTRYPOINT 命令设置当前工作路径。可以设置多次，如果是相对路径，则相对前一个 WORKDIR 命令。默认路径为/。一般格式为 WORKDIR /path/to/work/dir 。例如我们设置/app 路径，我们可以进行如下操作：

WORKDIR /app

RUN命令

RUN 用于容器内部执行命令。每个 RUN 命令相当于在原有的镜像基础上添加了一个改动层，原有的镜像不会有变化。一般格式为 RUN <command> 。例如我们要安装 python 依赖包，我们做法如下：

RUN pip install -r requirements.txt

EXPOSE

EXPOSE 命令用来指定对外开放的端口。一般格式为 EXPOSE <port> [<port>...]

开放5000端口：

EXPOSE 5000

ENTRYPOINT

ENTRYPOINT 可以让你的容器表现得像一个可执行程序一样。**一个 Dockerfile 中只能有一个 ENTRYPOINT，如果有多个，则最后一个生效。**

ENTRYPOINT 命令也有两种格式：

ENTRYPOINT ["executable", "param1", "param2"] ：推荐使用的 exec形式

ENTRYPOINT command param1 param2 ：shell 形式

例如下面这个，我们要将 python 镜像变成可执行的程序，我们可以这样去做：

ENTRYPOINT ["python"]

CMD

CMD 命令用于启动容器时默认执行的命令，CMD 命令可以包含可执行文件，也可以不包含可执行文件。不包含可执行文件的情况下就要用 ENTRYPOINT 指定一个，然后 CMD 命令的参数就会作为ENTRYPOINT的参数。

CMD 命令有三种格式：

CMD ["executable","param1","param2”]: 推荐使用的 exec 形式。Container启动时将执行 executable param1 param2

CMD ["param1","param2”]: 无可执行程序形式

CMD command param1 param2: shell 形式。

一个 Dockerfile 中只能有一个CMD，如果有多个，则最后一个生效。而 CMD 的 shell 形式默认调用 /bin/sh -c 执行命令。

P.s

RUN命令在 image 文件的构建阶段执行，执行结果都会打包进入 image 文件；CMD命令则是在容器启动后执行。另外，一个 Dockerfile 可以包含多个RUN命令，但是只能有一个CMD命令。注意，指定了CMD命令以后，docker container run命令就不能附加命令了，否则它会覆盖CMD命令。例如命令行传入的参数覆盖：docker run busybox /bin/echo Hello Docker 则/bin/echo Hello Docker会把 CMD 里的命令覆盖。

例如我们要启动 /app.py ，我们可以用如下命令实现：

CMD ["app.py"]

ONBUILD（为他人做嫁衣裳）  
格式：ONBUILD <其它指令>。

ONBUILD 是一个特殊的指令，它后面跟的是其它任何指令，比如 RUN, COPY 等，而这些指令，在当前此次镜像构建时并不会被执行。只有当以当前镜像为基础镜像，去构建下一级镜像的时候才会被执行。这样的作用是当我们为一个application构建镜像，而使用这个带有ONBUILD的dockerfile时，ONBUILD后面接的操作例如拷贝package.son时，才会执行，并且最重要的是拷贝的是application自身的package.json而不是基础镜像的package.json.

Dockerfile 中的其它指令都是为了定制当前镜像而准备的，唯有 ONBUILD 是为了帮助别人定制自己而准备的。

一个实例流程为：首先我们用一个dockerfile1构建一个image.该dockerfile1中image为Docker Hub中centos.然后在dockerfile1中定义ONBUILD修饰的指令。然后docker build创建一个名为centos-mycentos的本地image.接下来我们再写一个dockerfile2,这个dockerfile2的镜像为FROM centos-mycentos,然后使用这个dockerfile2构建一个image.此时构建这个image的时候，便会执行dockerfile1的ONBUILD指令。因为此时满足ONBUILD指定条件。

🡪

.dockerignore文件中使用正则表达式表示不想被涵盖入container的文件。常见忽略的文件有.git, passwd敏感信息文件，占用空间大的文件。