**Dockerfile创建镜像**

Dockerfile由一行行命令语句组成，并且支持用“#”开头作为注释，一般的，Dockerfile分为四部分：基础镜像信息，维护者信息，镜像操作指令和容器启动时执行的指令。

Dockerfile指令：

**1、FROM**

格式：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中创建多个镜像时，可以使用多个FROM指令（每个镜像一次）

2、MAINTAINET

格式：MAINTAINET <name>

指定维护者的信息

3、RUN

格式：RUN <command> 或 RUN ["", "", ""]

每条指令将在当前镜像基础上执行，并提交为新的镜像。（可以用“\”换行）

4、CMD

格式：CMD ["","",""]

指定启动容器时执行的命令，每个Dockerfile只能有一条CMD指令，如果指定了多条指令，则最后一条执行。（会被启动时指定的命令覆盖）

5、EXPOSE

格式：EXPOSE <port>  [ <port> ...]

告诉Docker服务端暴露端口，在容器启动时需要通过 -p 做端口映射

6、ENV

格式：ENV <key> <value>

指定环境变量，会被RUN指令使用，并在容器运行时保存

7、ADD

格式：ADD  <src>  <dest>

复制指定的<src>到容器的<dest>中，<src>可以是Dockerfile所在的目录的一个相对路径；可以是URL，也可以是tar.gz（自动解压）

8、COPY

格式：COPY <src>  <dest>

复制本地主机的 <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）

9、ENTRYPOINT

格式：ENTRYPOINT ["","",""]

配置容器启动后执行的命令，并且不可被 docker run 提供的参数覆盖。（每个 Dockerfile 中只能有一个 ENTRYPOINT ，当指定多个时，只有最后一个起效）

10、VOLUME

格式：VOLUME ["/mnt"]

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

11、USER

格式：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。

12、WORKDIR

格式：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）

13、ONBUILD

格式：ONBUILD [INSTRUCTION]

配置当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

------------------分割线----------------------

1、创建Dockerfile文件

首先，需要创建一个目录来存放Dockerfile文件，目录名称可以任意，在目录里创建Dockerfile文件

mkdir Test

cd Test（ls查看该目录下的文件)

vim Dockerfile(vim简单使用：i进入insert，输入完毕，按ESC退出insert，ZZ或:wq保存并退出)

在Test文件夹下创建的dockerfile，内容如下：

#使用的是ubuntu的基础镜像，这里使用的是14.04版本，必须是第一条非注释指令

FROM ubuntu:14.04

#制定镜像的作者信息

MAINTAINER ERNEST xx “[1234567@qq.com](mailto:1234567@qq.com)”

#指定当前镜像中运行的命令u

RUN apt-get -y update && apt-get install -y nginx

RUN mkdir -p /var/www/html/website

ADD nginx/global.conf /etc/nginx/conf.d/

ADD nginx/nginx.conf /etc/nginx/nginx.conf

#运行该镜像的容器使用的端口

EXPOSE 80

2、从这个Dockfile里面看出：我们使用了ubuntu的基础镜像，然后下载了nginx，同时建立一个/var/www/html/website文件夹，然后又拷贝了宿主机上的两个文件，一个是global.conf，另一个是nginx.conf，这两个文件需要我们自己写。于是我们就要在Test下再建立一个叫nginx的文件夹，里面写上这两个文件

mkdir nginx

cd nginx

vim global.conf

vim nginx.conf

global.conf的内容如下：

server {

listen 0.0.0.0:80;

server\_name \_;

root /var/www/html/website;

index index.html index.htm;

access\_log /var/log/nginx/default\_access.log;

error\_log /var/log/nginx/default\_error.log;

}

nginx.conf的内容如下：

user www-data;

worker\_processes 4;

pid /run/nginx.pid;

events { }

http {

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error.log;

gzip on;

gzip\_disable "msie6";

include /etc/nginx/conf.d/\*.conf;

}

3、build这个镜像，比如这个镜像名叫做ljwtest/nginx:1.0，在Test文件夹里使用的命令语句就是：

cd ..

(cd ..回到Test文件夹)

docker build -t='ljwtest/nginx:1.0' . （. 表示当前路径）

docker images 查看已有镜像



有了镜像，再在Test文件夹里新增一个文件夹，比如就叫website，里面有一个文件叫index.html。

mkdir website

cd website

vim index.html

而index.html的内容如下：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>hello</title>

</head>

<body>

<p>hello world</p>

</body>

</html>

4、返回到Test目录cd ..

制作一个容器，制作容器命令是 # docker run -d -p 8088:80 --name test01 -v $PWD/website:/var/www/html/website ljwtest/nginx:1.0 nginx -g "daemon off;" ,这句话里规定容器的8080端口映射到宿主机的80端口，同时引入了当前目录的website目录到容器的/var/www/html/website目录，nginx也默认在前台进程进行。执行之后，#docker ps –a



**·**引入-v这个命令在容器里，可以随时调试内容，而不是每次都要重新打包生成镜像。这一点再调试阶段为我们提供了很大的方便。

**·**docker run命令里指定端口的格式是，" -p 容器端口:宿主机端口"。如果想要随机指定就是大写的P

**·** 有时候port这里却不显示端口映射的情况,这个情况是因为这个容器的status是exited，Docker 会在容器主进程结束后自动终止容器运行，而 nginx 启动后就会在后台运行，docker 以为 nginx 已经结束运行了，所以就会停止容器。

5、测试

**这时我们在docker主机上可通过如下方式访问容器的服务**

**方式一：**

**curl http://localhost:8088**

**方式二**：

curlhttp://123.207.215.55:8088 (123.207.215.55为本机IP)

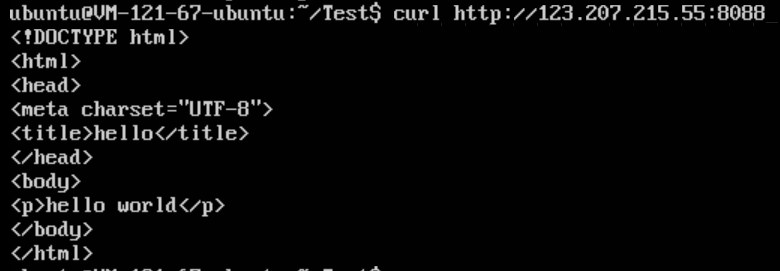
这样docker主机局域网内的机器就可以通过主机的地址和绑定的ip来访问到docker容器上的服务了。不过这存在的问题是，需要将容器端口与主机上端口的映射关系变成固定的。否则一旦容器重启，绑定端口就会变了。

**方式三**：

假设容器的ip地址为172.17.0.2(可通过docker inspect查看)

curl <http://172.17.0.2:80>

curl <http://123.207.215.55:8088>



在这台器上访问

