Dockerfile

- 手动编写一个dockerfile文件，必须符合file的规范
- 有了这个文件后，直接docker build 命令执行，获得一个新的自定义的镜像
- run

以下代码相当于java 中的object，是所有镜像的祖先类

```
From scratch
```

- 每条保留字的后面都必须使用大写字母且后面要紧随至少一个参数
- 指令按照从上到下顺序执行
- # 标识注释
- 每条指令都会创建一个新的镜像层，并对镜像进行提交

运行的大致流程：

- docker 从基础镜像运行一个容器
- 执行一条指令并对容器做出修改
- 执行类似docker commit 的操作提交一个新的镜像
- docker 在基于刚才提交的镜像运行一个新容器
- 执行docker 中的下一条指令直到所有指令都执行完毕

Dockerfile 保留字

FROM

```
基础镜像，当前镜像是基于哪个镜像的
```

MAINTAINER

```
镜像的维护者的姓名和邮箱
```

RUN

```
容器构建是需要运行的命令
```

EXPOSE

```
该容器暴露出对外的端口号
```

WORKDIR

```
指定在创建容器后，终端默认登录的进来工作目录，一个落脚点，
当我们运行这句代码的时候，docker run -it  zzyy/ubuntu
进入到ubuntu目录的 / 目录下面，之所以进入这里是在unbuntu 里面没有设置
WORKDIR (默认方式为进入到 / 目录)
如果在ubuntu 中设置 WORKDIR /data
那么当运行  docker run -it  zzyy/ubuntu
你进入ubuntu系统后的目录就是 /data  目录
```

ENV

```
构建镜像创建过程中设置的环境变量
```

COPY

```
拷贝
```

ADD

```
拷贝 + 解压缩，将宿主机目录下的文件拷贝到镜像且会自动处理URL和tar压缩包
```

CMD: 指定一个容器启动时候需要运行的命令

```
指令格式和RUN相似，也是两种格式
shell CMD<命令>
exec CMD [可执行文件，  参数1，参数2.。。]
```

ENTRYPOINT： 指定一个容器启动时候需要运行的命令

```
和CMD一样
和CMD唯一一点不同的是
在DockerFile中可以有多个CMD命令，但是只有最后一个生效，最后一个CMD命令之前的命令都会失效
entrypoint 不会被替换，
```

ONBUILD
只要子镜像继承了父镜像，并且子镜像得到了运行，那么父镜像的onbuild被触发

案例

在拉去的ubuntu镜像，是没有vim、ifconfig 等命令的，现在去制作一个自己的ubuntu镜像，具备这些命令

```
FROM centos
MAINTANIER zzyy<123@email.com>

ENV mypath /tmp

WORKDIR $mypath

RUN yum -y install vim
RUN yum -y install net-tools

EXPOSE 80

CMD echo $mypath
CMD echo “============ successful ==========”

CMD /bin/bash
```

curl http://www.baidu.com
获取网址的form表单，下载命令

```
FROM centos
RUN yum install -y curl
CMD [“curl”, “-s”, “http://ip.cn"]
```