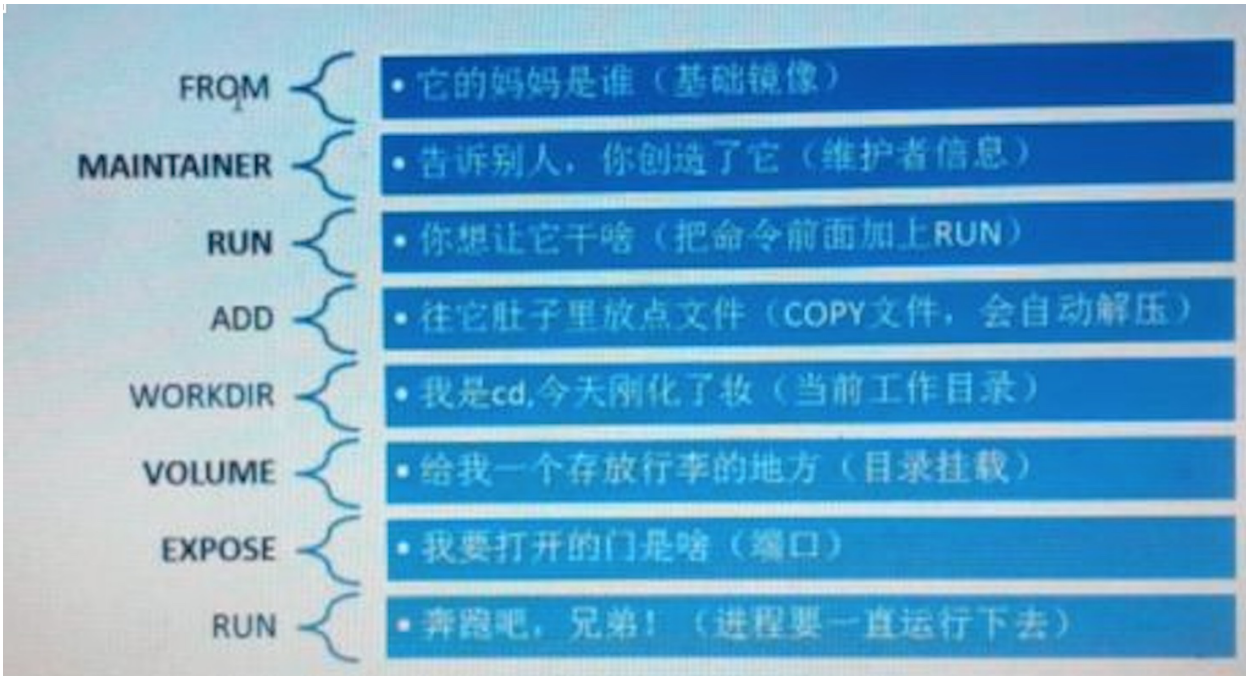
参考文档： 文档

1,Dockerfile介绍

- 目的: 知道dockerfile的作用,以及常见的指令即可
- 作用:
  - 就是一个脚本, 一旦运行可以构建镜像, 通过镜像创建容器的时候,会自动执行指令
  - 常见的指令: from, maintainer, run, env,add,copy,workdir,entrypoint,expose



2, from指令

- 目的: 可以知道from的作用,并创建新的镜像
- 操作流程:
  - 1, 在test文件加中创建Dockerfile文件,

```
touch Dockerfile
```

    - 编写如下内容

```
#1,基于哪个镜像创建新的镜像
from ubuntu
```

  - 2, 执行dockerfile文件

```
docker build -t 镜像名 ==.==
```

- 注意事项
  - 1, 创建的Dockerfile文件必须大写开头
  - 2, form一定要写在有效代码第一行

3, maintainer指令

- 目的: 可以给生成的镜像指定名字
- 操作流程:
  - 1, 编写dockerfile

```
#1,基于哪个镜像创建新的镜像
from ubuntu

```

```
#2,指定镜像的作者
maintainer laowang
```

- 2,执行dockerfile文件
- 3,查看镜像信息

```
docker inspect ubuntu2
```

4,run指令

- 目的: 可以编写run指令, 创建镜像和容器
- 特点: 创建镜像的时候会自动执行命令, 当通过镜像创建容器的时候, 容器中就有run命令的内容了
- 操作流程:
  - 1, 编写dockerfile文件

```
#1,基于哪个镜像创建新的镜像
from ubuntu

#2,指定镜像的作者
maintainer laowang

#3,执行命令
run apt-get update && apt-get install net-tools && apt-get install vim
```

- 2, 执行dockerfile文件

```
docker build -t ubuntu3:latest .
```

- 3, 创建容器测试即可

```
docker run -it ubuntu3 /bin/bash
```

5,env指令

- 目的: 可以定义环境变量,并输出
- 操作流程:
  - 1, 编写dockerfile文件

```
from ubuntu
env number 10
```

- 2, 执行dockerfile文件

```
# 使用变量
enco $number   # 10
```

6, add指令

- 目的: 可以向镜像中添加文件或者文件夹
- 操作流程:
  - 1, 在dockerfile同级的文件中创建a.txt, data.tar并编写dockerfile

```
#1,基于哪个镜像创建新的镜像
from ubuntu

#2,指定镜像的作者
maintainer laowang

#3,执行命令
#run apt-get update && apt-get install net-tools && apt-get install vim -y

#4,可以向镜像中添加文件,或者压缩包
add a.txt /home
add data.tar /home
```

  - 2, 执行dockerfile文件

```
docker build -t ubuntu6:latest .
```

  - 3, 创建容器,测试是否存在a.txt, data

```
docker run -it ubuntu6:latest /bin/bash
```

7, copy指令

- 目的: 可以拷贝文件,或者压缩包到镜像中
- 操作流程:
  - 1, 编写dockerfile

```
#1,基于哪个镜像创建新的镜像
from ubuntu

#2,指定镜像的作者
maintainer laowang

```

```
#3,执行命令
#run apt-get update && apt-get install net-tools && apt-get install vim -y

#4,可以向镜像中添加文件,或者压缩包
#add a.txt /home
#add data.tar /home

#5,向镜像中拷贝文件或者压缩包
copy a.txt /home
copy data.tar /home
```

- 2, 执行dockerfile

```
docker build -t ubuntu7:latest .
```

- 3, 创建镜像测试

```
docker run -it ubuntu7 /bin/bash
```

- 注意点:
  - 1, add拷贝压缩会解压
  - 2, copy不会解压压缩包

8, workdir指令

- 目的: 在创建容器之后, 直接可以进入到指定的文件夹中去
- 操作流程:
  - 1, 编写dockerfile文件

```
#1,基于哪个镜像创建新的镜像
from ubuntu

#2,指定镜像的作者
maintainer laowang

#3,执行命令
#run apt-get update && apt-get install net-tools && apt-get install vim -y

#4,可以向镜像中添加文件,或者压缩包
#add a.txt /home
#add data.tar /home

```

```
#5,向镜像中拷贝文件或者压缩包
#copy a.txt /home
#copy data.tar /home

#6,先创建文件夹目录,然后直接进去
run mkdir /home/data
run mkdir /home/data/data2
run mkdir /home/data/data2/data3
workdir /home/data/data2/data3
```

- 2,执行dockerfile

```
docker build -t ubuntu8:latest .
```

- 3, 创建容器测试

```
docker run -it ubuntu8 /bin/bash
```

10, helloworld部署

- 目的: 可以通过容器创建django程序并使用桥接和host模式访问容器
- 操作流程:
  - 1, 编写dockerfile文件

```
#1,基于哪个镜像创建新的镜像
from ubuntu_python

#2,指定镜像的作者
maintainer laowang


#7,对外提供端口
expose 8000
```

- 2, 执行dockerfile

```
docker build -t ubuntu_python1:latest .
```

- 3, 创建容器
  - 1, 桥接模式

```
docker run -itd -p 8000:8000  --name django2 ubuntu_python1
```

        - 注意点: 需要使用expose对外提供端口
    - 2, host模式

```
docker run -it --network=host ubuntu_python1 /bin/bash
```

- 3, 创建django程序, 设置allow_host = ["*"]
    - 创建demo01项目

```
python3 manage.py runserver 0.0.0.0:8000
```

    - 0.0.0.0: 既能通过本地127.0.0.1访问, 也能使用本机172.16.12.134

11, hellworld部署(dockerfile),entrypoint指令

- 目的: 可以使用dockerfile将helloworld进行部署
- 操作流程:
    - 1, 创建demo01程序, 和dockerfile同级
    - 2, 编写dockerfile

```
#1, 导入镜像
from ubuntu_python

#2,导入程序
add demo01.tar /home

#3,进入demo01中
workdir /home/demo01

#4,对外提供端口
expose 8000

#5,运行程序
entrypoint python3 manage.py runserver 0.0.0.0:8000
```

- 3, 执行dockerfile

```
docker build -t ubuntu_django2 .
```

- 4, 创建容器测试

```
docker run -d --network=host ubuntu_django2
```

- 注意点:
  - entrypoint: 容器创建就会自动执行指令

Dockerfile中RUN，CMD和ENTRYPOINT都能够用于执行命令，下面是三者的主要用途：

- RUN命令执行命令并创建新的镜像层，通常用于安装软件包
- CMD命令设置容器启动后默认执行的命令及其参数，但CMD设置的命令能够被docker run命令后面的命令行参数替换
- ENTRYPOINT配置容器启动时的执行命令（不会被忽略，一定会被执行，即使运行docker run时指定了其他命令）