dockerfile实践学习

> 需求：
>
> 让你安装一个mysql，且启动
>
> 如果是虚拟机部署形式的话，如下
>
> 1.开启vmware
>
> 2.运行某一个虚拟机，centos7
>
> 3.centos7安装mysql yum install mysql-server
>
> 4.通过脚本，或者命令，启动mysql即可
>
> 部署缓慢，且修改了宿主机的环境，删除较为麻烦，占用宿主机的一个3306端口

我们要学的是基于容器去运行mysql

1.开始vmware

2.运行虚拟机centos7（宿主机）

3.安装docker容器软件

4.获取mysql镜像即可，docker pull mysql:tag **（你无法自由控制，该mysql的基础镜像是什么发行版，你获取的镜像，是别人定制好，你下载使用的，debian，你希望得到一个基于centos7.8的发行版，运行mysql）**

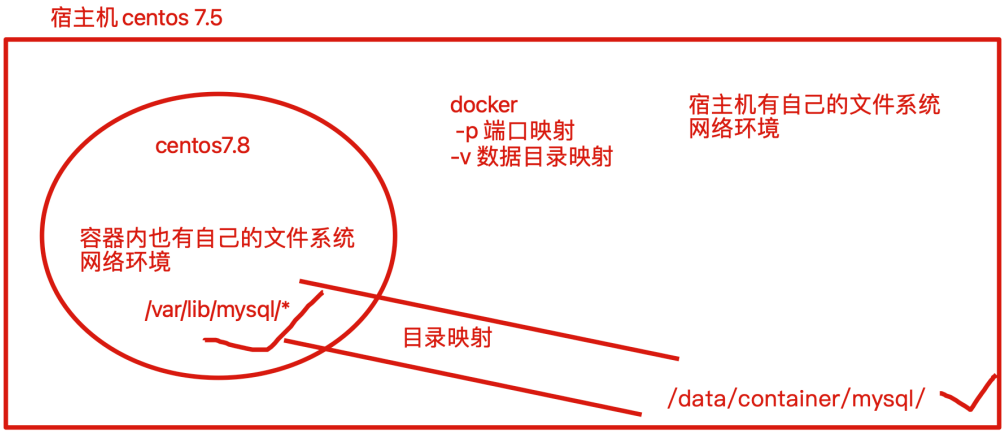
5.直接运行该镜像，通过端口映射，运行mysql，`docker run mysql:5.6` **（容器能够运行，必须在容器内，有一个进程在前台运行，该容器内，有mysql正在前台运行）**

6.访问宿主机的一个映射端口，访问到容器内的mysql

> 想自定义镜像，就得自己写脚本，也就是dockfile了

dockfile指令学习

```
1
2  FROM 这个镜像的妈妈是谁？（指定基础镜像）
3
4  MAINTAINER 告诉别人，谁负责养它？（指定维护者信息，可以没有）
5
6  RUN 你想让它干啥（在命令前面加上RUN即可）
7
8  ADD 添加宿主机的文件到容器内，还多了一个自动解压的功能
9
10 # RUN tar -zxf /opt/xx.tgz # 报错！该tgz文件不存在！！
11
12 COPY 作用和ADD是一样的，都是拷贝宿主机的文件到容器内，COPY就是仅仅
   拷贝
13
14 WORKDIR 我是cd,今天刚化了妆（设置当前工作目录）
15
16 VOLUME 给它一个存放行李的地方（设置卷，挂载主机目录）
17
18 EXPOSE 它要打开的门是啥（指定对外的端口），在容器内暴露一个窗口，端
   口 EXPORT 80
19
20 CMD 奔跑吧，兄弟！（指定容器启动后的要干的事情）
```



dockerfile实践

需求，通过dockerfile，构建nginx镜像，且运行容器后，生成的页面，是"超哥带你学习docker"

```
# 1. 创建Dockerfile，注意文件名，必须是这个
[root@yc_docker01 learn_docker]# pwd
/learn_docker
[root@yc_docker01 learn_docker]# cat Dockerfile
FROM nginx
RUN echo '<meta charset=utf8>超哥带你用docker运行nginx服务.' > /usr/share/nginx/html/index.html
[root@yc_docker01 learn_docker]#

# 2.构建Dockerfile
docker build .

# 3.修改镜像名字
[root@yc_docker01 learn_docker]# docker tag b4200a856253 my_nginx

# 构建出的镜像如下
```

```
[root@yc_docker01 learn_docker]# docker images
REPOSITORY                    TAG         IMAGE ID
CREATED              SIZE
my_nginx                      latest      b4200a856253
About a minute ago   133MB

# 4.运行该镜像
docker run -d -p 80:80  my_nginx

# 5.查看宿主机的80端口
http://10.211.55.21/
超哥带你用docker运行nginx服务.
```

> 如下指令该怎么用

```
# COPY
copy chaoge.py /home/

# 支持多个文件，以及通配符形式复制，语法要满足Golang的filepath.Match
copy chaoge* /tmp/cc?.txt. /home/


# ADD
ADD chaoge.tgz /home/
RUN  linux命令（xxx修改命令）

# CMD在容器内运行某个命令，启动程序
# 该镜像在运行容器实例的时候，执行的具体参数是什么

CMD ["参数1","参数2"]

CMD ["/bin/bash"]
```

```
18
19 # 该容器运行时，执行的命令
20 # 等同于命令行的直接操作 docker run -it centos   cat /etc/os-release
21 CMD ["cat","/etc/os-release"]
22
23
24 # CMD有关启动程序的坑
25
26
```

把宿主机安装，启动nginx的理念放入到dockerfile

1. RUN yum install nginx
2. RUN 配置文件修改 sed
3. `RUN systemctl start nginx` ❌ 容器内的程序必须是前台运行，你的容器是启动不了的
4. 正确的应该是 CMD ["nginx","-g","daemon off;"]

> docker面试题
>
> ENTRYPOINT和CMD的区别以及用法

ENTRYPOINT和CMD的玩法

```
1 作用和CMD一样，都是在指定容器启动程序以及参数。
2
3 当指定了ENTRYPOINT之后，CMD指令的语义就有了变化
4 而是吧CMD的内容当作参数传递给ENTRYPOINT指令。
5
6 CMD    xxxx
7 ENTRYPOINT   xxx
8
```

```
9
10 # 1. 准备一个dockerfile
11 FROM centos:7.8.2003
12 RUN rpm --rebuilddb &&  yum install epel-release -y
13 RUN rpm --rebuilddb &&  yum install curl -y
14 CMD ["curl","-s","http://ipinfo.io/ip"]
15
16 # 用法如下
17 docker run  my_centos curl -s http://ipinfo.io/ip
18 #
19 dockr run my_centos
20
21 # 2.构建镜像
22 docker build .
23
24 # 3.查看结果
25  ---> Running in b00811eb9124
26 Removing intermediate container b00811eb9124
27  ---> 4dc77fbc02e3
28 Successfully built 4dc77fbc02e3
29
30 # 4.检查镜像
31 [root@yc_docker01 learn_docker]# docker tag 4dc77fbc02e3
   centos_curl
32 [root@yc_docker01 learn_docker]# docker images |grep curl
33 centos_curl                    latest     4dc77fbc02e3
   About a minute ago   462MB
34
35 # 5.运行镜像，生成容器记录，没有前台运行，因此立即挂了
36 [root@yc_docker01 learn_docker]# docker run centos_curl
37 18.167.165.145[root@yc_docker01 learn_docker]#
38
39 # 6.上述运行正确，但是我想再传入一个参数，该怎么办
40 发现是无法直接传入参数的，该形式是覆盖镜像中的cmd
```

```
就好比把该docker镜像，当做一个环境，去执行后面的命令
[root@yc_docker01 learn_docker]# docker run centos_curl pwd
/
[root@yc_docker01 learn_docker]#
[root@yc_docker01 learn_docker]# docker run centos_curl -I
docker: Error response from daemon: OCI runtime create
failed: container_linux.go:380: starting container process
caused: exec: "-I": executable file not found in $PATH:
unknown.


# 7.想要正确的给容器传入一个 -I 参数该怎么办
希望容器内能正确完整，这个命令的执行
curl -s http://ipinfo.io/ip  -I


# 8.解决办法1
给容器传入新的，完整的命令
# 这是投机取消的办法，不合适
[root@yc_docker01 learn_docker]# docker run centos_curl curl
-s http://ipinfo.io/ip -I
HTTP/1.1 200 OK
access-control-allow-origin: *
content-type: text/html; charset=utf-8
content-length: 13
date: Tue, 29 Jun 2021 08:42:49 GMT
x-envoy-upstream-service-time: 1
Via: 1.1 google

# 9.正确的姿势应该是使用 ENTRYPOINT
修改dockerfile，如下

FROM centos:7.8.2003
RUN rpm --rebuilddb &&  yum install epel-release -y
```

```
RUN rpm --rebuilddb &&  yum install curl -y
ENTRYPOINT ["curl","-s","http://ipinfo.io/ip"]

# 10.重新构建镜像
[root@yc_docker01 learn_docker]# docker build .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM centos:7.8.2003
 ---> afb6fca791e0
Step 2/4 : RUN rpm --rebuilddb &&  yum install epel-release -y
 ---> Using cache
 ---> 8936b4cae9bc
Step 3/4 : RUN rpm --rebuilddb &&  yum install curl -y
 ---> Using cache
 ---> 4f650e03058d
Step 4/4 : ENTRYPOINT ["curl","-s","http://ipinfo.io/ip"]
 ---> Running in 0226ffc5eace
Removing intermediate container 0226ffc5eace
 ---> 2e1fc7f4cf92
Successfully built 2e1fc7f4cf92

# 11.重新运行该镜像，看结果，以及传入新的参数
[root@yc_docker01 learn_docker]# docker tag 2e1fc7f4cf92 centos_curl_new

# 此时发现，超哥没有在忽悠大家，传入的CMD指令，当做了ENTRYPOINT的参数
# 其实容器内，执行的完整命令，是 curl -s http://ipinfo.io/ip -I
[root@yc_docker01 learn_docker]# docker run centos_curl_new -I
HTTP/1.1 200 OK
access-control-allow-origin: *
content-type: text/html; charset=utf-8
content-length: 13
```

```
date: Tue, 29 Jun 2021 08:46:23 GMT
x-envoy-upstream-service-time: 1
Via: 1.1 google


```

ARG和ENV指令

设置环境变量

容器1 nginx

容器2 php-fpm

容器3 mysql

```
dockerfile 脚本，shell脚本

ENV NAME="yuchao"
ENV AGE="18"
ENV MYSQL_VERSION=5.6

后续所有的操作，通过$NAME 就可以直接获取变量值曹邹了，维护dockerfile脚本时更友好，方便
ADD
COPY
EXPOSE

ARG和ENV一样 设置环境变量
区别在于ENV 无论是在镜像构建时，还是容器运行，该变量都可以使用
ARG只是用于构建镜像需要设置的变量，容器运行时就消失了


```

VOLUME

容器在运行时，应该保证在存储层不写入任何数据，运行在容器内产生的数据，我们推荐是挂载，写入到宿主机上，进行维护。

```
# mount /mnt

VOLUME /data  # 将容器内的/data文件夹 ，在容器运行时，该目录自动挂载为匿名卷，任何向该目录中写入数据的操作，都不会被容器记录，保证的容器存储层无状态理念

# Dockerfile1
FROM centos
MAINTAINER chaoge
VOLUME ["/data1","/data2"]

# 该容器运行时候，这2个目录自动和宿主机的目录做好映射关系
docker build .

# 运行该镜像
docker run 13cee5b66d58
# 查看生成的容器信息
[root@yc_docker01 learn_docker]# docker ps -a|head -2
CONTAINER ID   IMAGE                                COMMAND
         CREATED          STATUS                          PORTS
  NAMES
5b12bf9d63e9   13cee5b66d58                         "/bin/bash"
         15 seconds ago   Exited (0) 15 seconds ago
  romantic_haslett

# docker inspect命令查看

docker inspect 5b12bf9d63e9
```