

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）

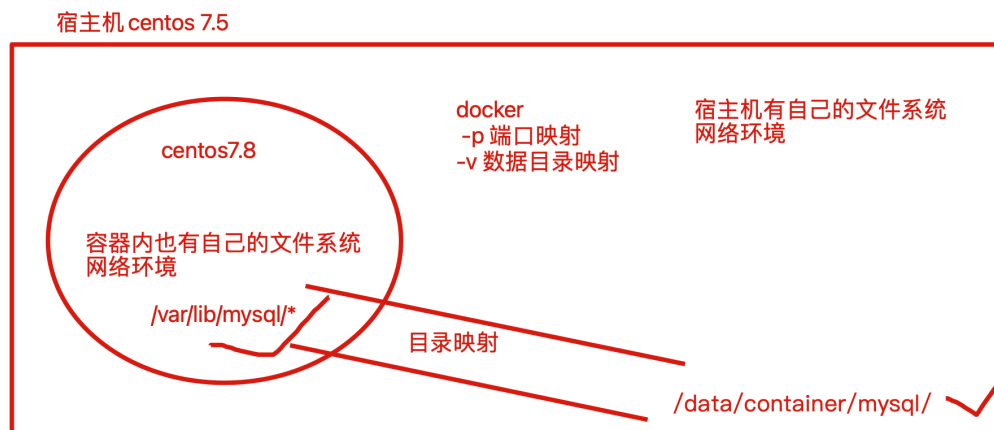
5.直接运行该镜像，通过端口映射，运行mysql，`docker run mysql:5.6`（容器能够运行，必须在容器内，有一个进程在前台运行，该容器内，有mysql正在前台运行）

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
   □ 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 # 1. 创建Dockerfile, 注意文件名, 必须是这个
2 [root@yc_docker01 learn_docker]# pwd
3 /learn_docker
4 [root@yc_docker01 learn_docker]# cat Dockerfile
5 FROM nginx
6 RUN echo '<meta charset=utf8>超哥带你用docker运行nginx服务.' >
  /usr/share/nginx/html/index.html
7 [root@yc_docker01 learn_docker]#
8
9 # 2.构建Dockerfile
10 docker build .
11
12 # 3.修改镜像名字
13 [root@yc_docker01 learn_docker]# docker tag b4200a856253
  my_nginx
14
15 # 构建出的镜像如下
```

```

16 [root@yc_docker01 learn_docker]# docker images
17 REPOSITORY          TAG          IMAGE ID
18 my_nginx             latest       b4200a856253
19 About a minute ago   133MB
20 # 4. 运行该镜像
21 docker run -d -p 80:80 my_nginx
22
23 # 5. 查看宿主机的80端口
24 http://10.211.55.21/
25 超哥带你用docker运行nginx服务.

```

如下指令该怎么用

```

1 # COPY
2 copy chaoge.py /home/
3
4 # 支持多个文件，以及通配符形式复制，语法要满足Golang的
  filepath.Match
5 copy chaoge* /tmp/cc?.txt. /home/
6
7
8 # ADD
9 ADD chaoge.tgz /home/
10 RUN linux命令 (xxx修改命令)
11
12 # CMD在容器内运行某个命令，启动程序
13 # 该镜像在运行容器实例的时候，执行的具体参数是什么
14
15 CMD ["参数1","参数2"]
16
17 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
```

```
41 就好比把该docker镜像，当做一个环境，去执行后面的命令
42 [root@yc_docker01 learn_docker]# docker run centos_curl pwd
43 /
44 [root@yc_docker01 learn_docker]#
45 [root@yc_docker01 learn_docker]# docker run centos_curl -I
46 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.
47
48
49 # 7.想要正确的给容器传入一个 -I 参数该怎么办
50 希望容器内能正确完整，这个命令的执行
51 curl -s http://ipinfo.io/ip -I
52
53
54 # 8.解决办法1
55 给容器传入新的，完整的命令
56 # 这是投机取消的办法，不合适
57 [root@yc_docker01 learn_docker]# docker run centos_curl curl
   -s http://ipinfo.io/ip -I
58 HTTP/1.1 200 OK
59 access-control-allow-origin: *
60 content-type: text/html; charset=utf-8
61 content-length: 13
62 date: Tue, 29 Jun 2021 08:42:49 GMT
63 x-envoy-upstream-service-time: 1
64 Via: 1.1 google
65
66 # 9.正确的姿势应该是使用 ENTRYPOINT
67 修改dockerfile，如下
68
69 FROM centos:7.8.2003
70 RUN rpm --rebuilddb && yum install epel-release -y
```

```
71 RUN rpm --rebuilddb && yum install curl -y
72 ENTRYPOINT ["curl","-s","http://ipinfo.io/ip"]
73
74 # 10.重新构建镜像
75 [root@yc_docker01 learn_docker]# docker build .
76 Sending build context to Docker daemon 2.048kB
77 Step 1/4 : FROM centos:7.8.2003
78 ---> afb6fca791e0
79 Step 2/4 : RUN rpm --rebuilddb && yum install epel-release -
y
80 ---> Using cache
81 ---> 8936b4cae9bc
82 Step 3/4 : RUN rpm --rebuilddb && yum install curl -y
83 ---> Using cache
84 ---> 4f650e03058d
85 Step 4/4 : ENTRYPOINT ["curl","-s","http://ipinfo.io/ip"]
86 ---> Running in 0226ffc5eace
87 Removing intermediate container 0226ffc5eace
88 ---> 2e1fc7f4cf92
89 Successfully built 2e1fc7f4cf92
90
91 # 11.重新运行该镜像，看结果，以及传入新的参数
92 [root@yc_docker01 learn_docker]# docker tag 2e1fc7f4cf92
centos_curl_new
93
94 # 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
95 # 其实容器内，执行的完整命令，是 curl -s http://ipinfo.io/ip -I
96 [root@yc_docker01 learn_docker]# docker run centos_curl_new -
I
97 HTTP/1.1 200 OK
98 access-control-allow-origin: *
99 content-type: text/html; charset=utf-8
100 content-length: 13
```



```
101 date: Tue, 29 Jun 2021 08:46:23 GMT
102 x-envoy-upstream-service-time: 1
103 Via: 1.1 google
104
105
```

ARG和ENV指令

设置环境变量

容器1 nginx

容器2 php-fpm

容器3 mysql

```
1  dockerfile 脚本, shell脚本
2
3  ENV NAME="yuchao"
4  ENV AGE="18"
5  ENV MYSQL_VERSION=5.6
6
7  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
8  ADD
9  COPY
10 EXPOSE
11
12 ARG和ENV一样 设置环境变量
13 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
14 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
15
16
17
```

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。

```

1  # mount /mnt
2
3  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
4
5  # Dockerfile1
6  FROM centos
7  MAINTAINER chaoge
8  VOLUME ["/data1", "/data2"]
9
10 # 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
11 docker build .
12
13 # 运行该镜像
14 docker run 13cee5b66d58
15 # 查看生成的容器信息
16 [root@yc_docker01 learn_docker]# docker ps -a | head -2
17 CONTAINER ID    IMAGE                                COMMAND
    CREATED        STATUS              PORTS
    NAMES
18 5b12bf9d63e9    13cee5b66d58                                "/bin/bash"
    15 seconds ago  Exited (0) 15 seconds ago
    romantic_haslett
19
20 # docker inspect命令查看
21
22 docker inspect 5b12bf9d63e9

```

23

24

1. 容器数据挂载的方式，通过`dockerfile`，指定`VOLUME`目录
2. 通过`docker run -v` 参数，直接设置需要映射挂载的目录

EXPOSE

指定容器运行时对外提供的端口服务，

- 帮助使用该镜像的人，快速理解该容器的一个端口业务

-

```
1 | docker port 容器
2 | docker run -p 宿主机端口: 容器端口
3 | docker run -P # 作用是随机宿主机端口: 容器内端口
```

-

WORKDIR

用于在`dockerfile`中，目录的切换，更改工作目录

```
1 | WORKDIR /opt
```

USER

用于改变环境，用于切换用户

```
1 | USER root
2 | USER chaoge
```

构建一个网站镜像

1.nginx, 修改首页内容, htm了 网站就跑起来了, web server, 提供web服务, 提供代理转发, 提供网关, 限流托等。。apache

2. web framework, web框架, 一般由开发, 通过某个开发语言, 基于某个web框架, 自己去开发一个web站点, python,django框架

1.用python语言, 基于flask web框架, 开发一个自己的网站, 写了一个后端的网站代码

2.开发dockerfile, 部署该代码, 生成镜像

3.其他人基于该镜像, docker run就可以在电脑跑起来你这个网站

网站的router, 也理解为path

<http://pythonav.cn/hello>

10.211.55.21 宿主机

10.211.55.21:8080/hello

```
1 | # 1.在宿主机下, 准备一个目录, 准备好dockerfile, 你的代码
2 | # 写一个flask的 python代码
3 | # 创建代码文件
4 | [root@yc_docker01 learn_docker]# cat chaoge_flask.py
5 | #coding:utf8
6 | from flask import Flask
7 | app=Flask(__name__)
```

```
8 @app.route('/hello')
9 def hello():
10     return "hello from docker,i am chaoge."
11 if __name__=="__main__":
12     app.run(host='0.0.0.0',port=8080)
13 [root@yc_docker01 learn_docker]#
14
15
16 # 2.编写dockerfile
17 touch Dockerfile
18
19
20 # 3.检查代码文件环境, 以及内容
21 [root@yc_docker01 learn_docker]# pwd
22 /learn_docker
23 [root@yc_docker01 learn_docker]#
24 [root@yc_docker01 learn_docker]# ls
25 chaoge_flask.py  Dockerfile
26 [root@yc_docker01 learn_docker]#
27
28
29 [root@yc_docker01 learn_docker]# cat Dockerfile
30 FROM centos:7.8.2003
31 RUN curl -o /etc/yum.repos.d/CentOS-Base.repo
32     https://mirrors.aliyun.com/repo/Centos-7.repo;
33 RUN curl -o /etc/yum.repos.d/epel.repo
34     http://mirrors.aliyun.com/repo/epel-7.repo;
35 RUN yum makecache fast;
36 RUN yum install python3-devel python3-pip -y
37 RUN pip3 install -i https://pypi.douban.com/simple flask
38 COPY chaoge_flask.py /opt
39 WORKDIR /opt
40 EXPOSE 8080
41 CMD ["python3","chaoge_flask.py"]
```

```
40
41
42
43 [root@yc_docker01 learn_docker]#
44
45
46 # 4.构建镜像
47 docker build --no-cache -t 'yuchao163/my_flask_web' .
48
49 # 最后的结果类似如下，才是正确
50 Removing intermediate container 02cdb179b35d
51 ---> 1235b732483a
52 Successfully built 1235b732483a
53 Successfully tagged yuchao163/my_flask_web:latest
54
55
56
57 # 5.运行镜像，生成容器
58 docker run -d --name my_flask_web_1 -p 90:8080
    yuchao163/my_flask_web
59
60 # 6.访问宿主机，看容器内的flask web网站
61 http://10.211.55.21:90/hello
62
63 hello from docker,i am chaoge.
64
65
66 # 7.如何修改网站的内容，这个程序是抛在容器里了
67
```

如何修改该网站的内容

1.修改宿主机的代码，以及dockerfile，重新构建

1 | # 课后作业，完成效果

2. 你可以进入到以及运行的容器内，修改代码，重启容器即可

```
1 # 1.进入容器
2 [root@yc_docker01 learn_docker]# docker exec -it 4efde9370b50
  bash
3 [root@4efde9370b50 opt]#
4
5 # 2.修改容器内的程序
6 [root@4efde9370b50 opt]# cat chaoge_flask.py
7 #coding:utf8
8 from flask import Flask
9 app=Flask(__name__)
10 @app.route('/hello')
11 def hello():
12     return "hello from docker,i am yuchao, xin ku tong xue men
  le .zhun bei xiu xi .heiheihei!!!"
13 if __name__=="__main__":
14     app.run(host='0.0.0.0',port=8080)
15 [root@4efde9370b50 opt]#
16
17 # 3.退出容器，重启容器
18 [root@yc_docker01 learn_docker]# docker restart 4efde9370b50
19 4efde9370b50
20 [root@yc_docker01 learn_docker]# docker ps
21 CONTAINER ID   IMAGE                                COMMAND
22                CREATED        STATUS        PORTS
23                NAMES
24 4efde9370b50   yuchao163/my_flask_web             "python3 chaoge_flas..."
25        6 minutes ago   Up 3 seconds   0.0.0.0:90->8080/tcp, :::90-
26        >8080/tcp    my_flask_web_1
```

```
23  
24  
25 | hello from docker,i am yuchao, xin ku tong xue men le .zhun  
    bei xiu xi .heiheihei!!!  
26  
27
```

再次复习docker的好处，用法

比如安装一个etcd、nacos，都是比较复杂的一些软件

需要依赖于go语言环境，比如需要依赖于java环境，在自己的机器安装好对应的开发环境，以及对应的版本，以及各种依赖。。

tomcat, jdk环境

当你有了docker,

`docker pull tomcat` # 这些主流的镜像都可以直接找到，并且该镜像中，就已经打包好了java环境

`docker pull nacos` # 打包好了各种依赖环境

`docker run tomcat xxxx.....` # 直接就可以访问tomcat了

docker容器管理总结

```
1 # 运行镜像，且进入容器内
2 [root@yc_docker01 ~]# docker run -it ubuntu bash
3 root@cb070de7a747:/#
4
5 # 容器运行web程序
6 # 注意端口使用，数字大一点 8000以后开始使用
7
8 [root@yc_docker01 ~]# docker run --name my_nginx_7070 -d --
  restart=always -p 7070:80 nginx
9 25634416c9ebf2fb074dc7a7378cfc9c653cfc569f1254fc97f0c056adb57e
  6d
10
11 # docker run 镜像id 这是前台运行容器
12
13 # 查看容器内日志，实时刷新
14
15 docker logs -f
16
17 # 查看运行时，以及挂掉的容器记录
18
19 docket ps 在运行的容器
20 # 等同于
21 docker container ls
22
```

```
23
24 docker ps -a 挂掉以及活着的容器
25 docker container ls -a
26
27 # 停止启动
28 docker start
29 docker stop
30
31 # 进入容器内
32 docker exec -it 容器id bash
33
34
35 # 删除容器
36 docker rm 容器id
37 docker rm `docker ps -aq`
38 # 强制杀死容器
39 docker rm -f 容器id
40
41 # 查看容器内进程信息的命令
42 docker top 容器id
43
44 # 查看容器内资源
45 docker stats 容器id
46
47 # 查看容器的具体信息
48 [root@yc_docker01 data]# docker inspect ed4e6852f620
49
50 # 获取容器内的ip地址，容器的格式化参数
51 [root@yc_docker01 data]# docker inspect ed4e6852f620 | grep
172
52         "Gateway": "172.17.0.1",
53         "IPAddress": "172.17.0.3",
54         "Gateway": "172.17.0.1",
55         "IPAddress": "172.17.0.3",
```

```
56 [root@yc_docker01 data]#  
57  
58 # 拿到容器内ip  
59 docker inspect --format '{{.NetworkSettings.IPAddress}}'  
    ed4e6852f620  
60  
61  
62  
63
```