```
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
-

```
docker port 容器
docker run -p 宿主机端口：容器端口
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

```
WORKDIR /opt
```

USER

用于改变环境，用于切换用户

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

```
@app.route('/hello')
def hello():
    return "hello from docker,i am chaoge."
if __name__=="__main__":
    app.run(host='0.0.0.0',port=8080)
[root@yc_docker01 learn_docker]#


# 2.编写dockerfile
touch Dockerfile


# 3.检查代码文件环境，以及内容
[root@yc_docker01 learn_docker]# pwd
/learn_docker
[root@yc_docker01 learn_docker]#
[root@yc_docker01 learn_docker]# ls
chaoge_flask.py  Dockerfile
[root@yc_docker01 learn_docker]#


[root@yc_docker01 learn_docker]# cat Dockerfile
FROM centos:7.8.2003
RUN curl -o /etc/yum.repos.d/CentOS-Base.repo
https://mirrors.aliyun.com/repo/Centos-7.repo;
RUN curl -o /etc/yum.repos.d/epel.repo
http://mirrors.aliyun.com/repo/epel-7.repo;
RUN yum makecache fast;
RUN yum install python3-devel python3-pip  -y
RUN pip3 install -i https://pypi.douban.com/simple flask
COPY chaoge_flask.py  /opt
WORKDIR /opt
EXPOSE 8080
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

```
# 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.进入容器
[root@yc_docker01 learn_docker]# docker exec -it 4efde9370b50 bash
[root@4efde9370b50 opt]#

# 2.修改容器内的程序
[root@4efde9370b50 opt]# cat chaoge_flask.py
#coding:utf8
from flask import Flask
app=Flask(__name__)
@app.route('/hello')
def hello():
    return "hello from docker,i am yuchao, xin ku tong xue men le .zhun bei xiu xi .heiheihei!!!"
if __name__=="__main__":
    app.run(host='0.0.0.0',port=8080)
[root@4efde9370b50 opt]#

# 3.退出容器，重启容器
[root@yc_docker01 learn_docker]# docker restart 4efde9370b50
4efde9370b50
[root@yc_docker01 learn_docker]# docker ps
CONTAINER ID   IMAGE                    COMMAND
   CREATED         STATUS         PORTS
             NAMES
4efde9370b50   yuchao163/my_flask_web   "python3 chaoge_flas…"
  6 minutes ago   Up 3 seconds   0.0.0.0:90->8080/tcp, :::90->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

```
# 运行镜像，且进入容器内
[root@yc_docker01 ~]# docker run -it ubuntu bash
root@cb070de7a747:/#

# 容器运行web程序
# 注意端口使用，数字大一点 8000以后开始使用

[root@yc_docker01 ~]# docker run --name my_nginx_7070  -d --restart=always   -p 7070:80  nginx
25634416c9ebf2fb074dc7a7378cfc9c653cfc569f1254fc97f0c056adb57e6d

# docker run 镜像id  这是前台运行容器

# 查看容器内日志，实时刷新

docker logs -f

# 查看运行时，以及挂掉的容器记录

docket ps   在运行的容器
# 等同于
docker container ls

```

```
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",
```

```
[root@yc_docker01 data]#

# 拿到容器内ip
docker inspect --format '{{.NetworkSettings.IPAddress}}'
ed4e6852f620


```