

---

# Docker 封装帮助文件

前言：熟悉 docker 的选手，可直接跳转阅读第三部分。

## 目录

|                                     |    |
|-------------------------------------|----|
| 一、 安装 docker 环境.....                | 3  |
| 1.1 虚拟机安装与配置.....                   | 3  |
| 1.1.1 安装 VirtualBox 和 Linux 系统..... | 3  |
| 1.1.2 建立与 Windows 可同步的文件夹.....      | 3  |
| 1.2 安装 docker.....                  | 5  |
| 二、 竞赛中使用 docker 的必要命令.....          | 6  |
| 2.1 启动 docker.....                  | 6  |
| 2.2 Docker 使用的 hello World 教程.....  | 6  |
| 2.3 理解镜像与容器.....                    | 8  |
| 2.4 删除镜像.....                       | 8  |
| 三、 创建镜像标准样例.....                    | 9  |
| 3.1 选择镜像建议.....                     | 9  |
| 3.2 镜像必须包含的文件/文件夹.....              | 9  |
| 3.3 Dockerfile 文件编辑.....            | 10 |
| 3.4 Requirements.txt 文件.....        | 10 |
| 3.5 run.py 文件编辑.....                | 10 |
| 3.6 Testmodel.py 文件编辑.....          | 11 |

|                 |    |
|-----------------|----|
| 3.7 创建镜像.....   | 12 |
| 3.8 挂载目录.....   | 12 |
| 3.9 ★封装镜像★..... | 13 |
| 四、 使用与建议.....   | 13 |

## 一、安装 docker 环境

建议在 linux 系统中配置 docker 环境，会更加简单。

如果只有 windows 怎么办？可以装个虚拟机呀。Hyper-V？额，还是算了，关掉来装 VirtualBox。

### 1.1 虚拟机安装与配置

#### 1.1.1 安装 VirtualBox 和 Linux 系统

- (1) 官网下载 <https://www.virtualbox.org/wiki/Downloads>；
- (2) 可选择一直 next；然后运行 VirtualBox；
- (3) 配置并安装 Linux 系统，可参考博客 <https://www.cnblogs.com/rocedu/p/6012545.html>。

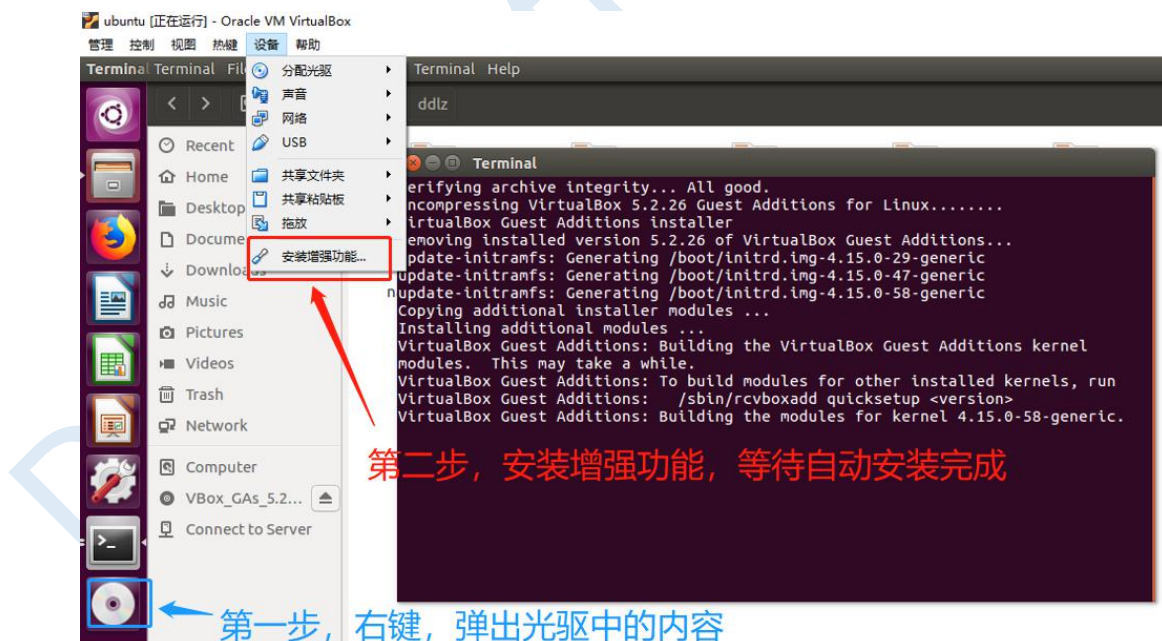
#### 1.1.2 建立与 Windows 可同步的文件夹

为什么要建立“共享文件夹”？可以在 windows 下面直接编辑呀~~

##### (1) 配置虚拟机，安装增强功能。

如下图，蓝色字体的“第一步”，红色字体的“第二步”。

此处测试版本为 VirtualBox5.2 版本，输入开机密码之后，就能自动安装完毕。

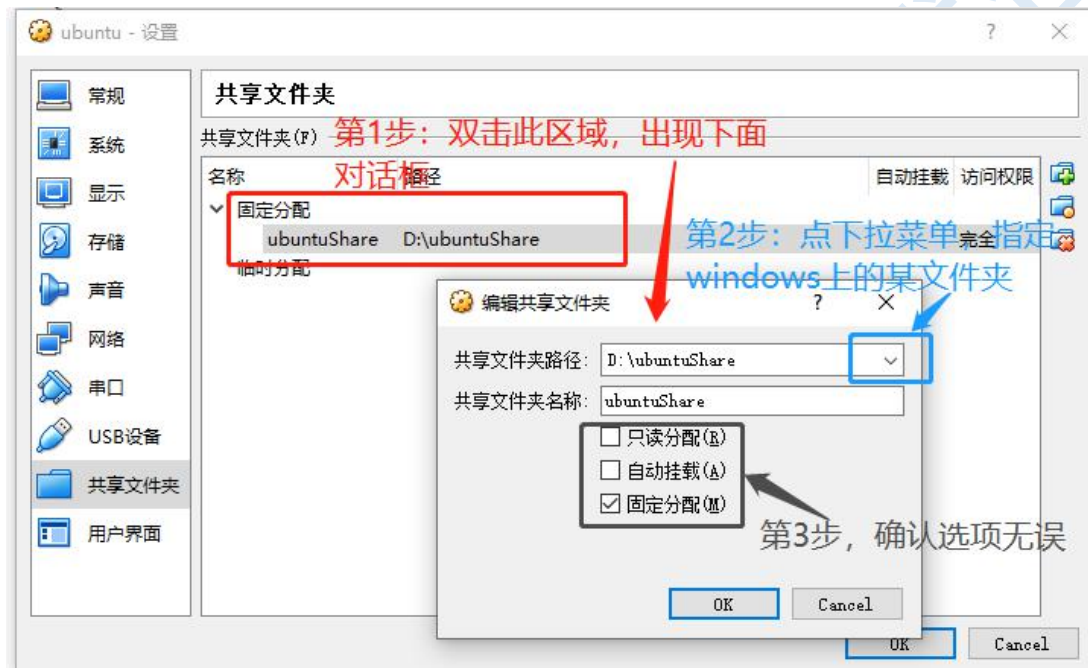


##### (2) 配置共享文件夹

按图中操作，点击“共享文件夹”。



然后，指定需要共享/同步的文件夹。共有 3 步。



### (3) 将此文件夹挂载到 ubuntu 系统中

先在 ubuntu 上建立一个文件夹，比如 /mnt/shareFiles，命令如下：

```
sudo mkdir /mnt/shareFiles
```

然后挂载命令为：

```
sudo mount -t vboxsf "第二步中的共享文件夹名称" "第三步中在 ubuntu 中新建立的文件夹"
```

注：共享文件夹名称，请在第(2)步中的第二幅图中找。

在本例中，完整的命令为：

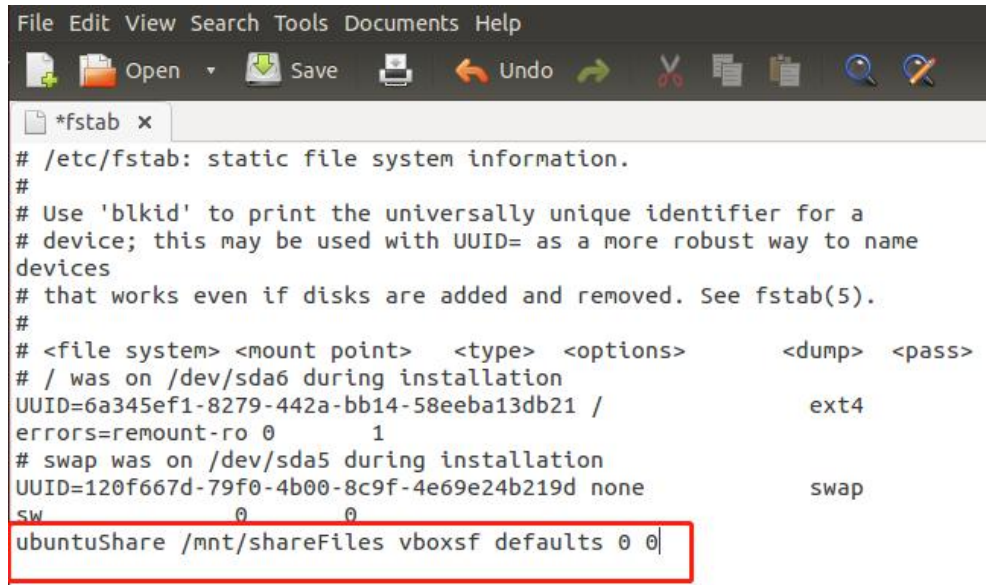
```
sudo mount -t vboxsf ubuntuShare /mnt/shareFiles
```

### (4) 开机自动挂载设置

在 linux 终端输入：`sudo gedit /etc/fstab`

然后，完成下图中的编辑，只需增加最下面的一行

```
ubuntuShare /mnt/shareFiles vboxsf defaults 0 0
```



```
File Edit View Search Tools Documents Help
Open Save Undo
*fstab x
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name
# devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options>          <dump> <pass>
# / was on /dev/sda6 during installation
UUID=6a345ef1-8279-442a-bb14-58eeba13db21 /              ext4
errors=remount-ro 0           1
# swap was on /dev/sda5 during installation
UUID=120f667d-79f0-4b00-8c9f-4e69e24b219d none            swap
sw              0           0
ubuntuShare /mnt/shareFiles vboxsf defaults 0 0
```

## 1.2 安装 docker

可直接参考网络中提供的教程，也可直接看下面我们的简化整理。

<https://segmentfault.com/a/1190000014066388>

此处简化如下：

### (1) 安装 https 相关的软件包

打开终端，输入以下命令：

```
sudo apt-get update # 先更新一下软件源库信息
```

```
# 然后直接复制粘贴下面的命令即可
```

```
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
```

### (2) 设置 apt 仓库地址，添加阿里云的 apt 仓库

```
curl -fsSL https://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-key
add -
```

```
sudo add-apt-repository \
    "deb [arch=amd64] https://mirrors.aliyun.com/docker-ce/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
```

### (3) 安装 docker

```
sudo apt-get update
```

```
sudo apt-get install docker-ce
```

#### (4) 查看 **docker** 版本

```
docker --version
```

## 二、 竞赛中使用 **docker** 的必要命令

注：docker 基本概念，例如什么是 image，什么是 container，请自行查阅资料。

### 2.1 启动 **docker**

Ubuntu 16 启动 docker 的方法（只需做一次即可）：

```
sudo systemctl enable docker
```

```
sudo systemctl start docker
```

避免每次都使用 sudo 的方法：建立 docker 组，将用户加入到 docker 组中

```
sudo groupadd docker
```

```
sudo usermod -aG docker $USER
```

```
sudo gpasswd -a ${USER} docker
```

```
sudo service docker restart
```

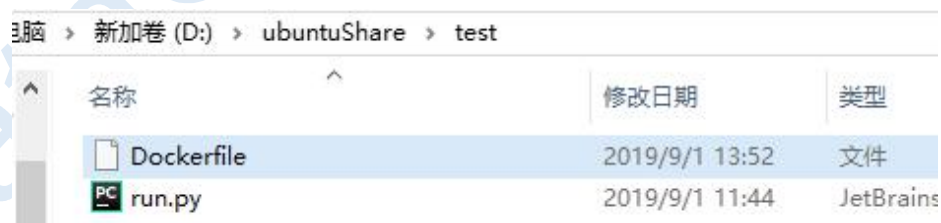
```
newgrp - docker
```

### 2.2 Docker 使用的 **hello World** 教程

先不管每行命令的意思，直接操作一遍，后面讲解。

#### (1) 编写镜像文件

在共享目录下新建文件夹，如 test，再新建两个文件，如下图。其中 **Dockerfile** 文件名不可更改。



| 名称         | 修改日期           | 类型        |
|------------|----------------|-----------|
| Dockerfile | 2019/9/1 13:52 | 文件        |
| run.py     | 2019/9/1 11:44 | JetBrains |

观察 Dockerfile 内容（先不管 python:alpine）：

```
FROM python:alpine
```

```
workdir /home
```

```
copy . /home
```

```
cmd ["python","run.py"]
```

观察 run.py 内容：

```
print('hello world!')
```



注意：**copy** 命令中，**.**与**/**之间是有空格的。

## (2) 创建镜像

在 linux 环境下，进入此目录，右键点击空白区域，打开终端 Terminal

输入以下命令：

```
docker build -t test .
```

注意：**test** 后面有一个空格，还有一个 **"."**。

```
qm@qmzhang:/mnt/shareFiles/test$ docker build -t test .
Sending build context to Docker daemon 3.072kB
Step 1/4 : FROM python:alpine
alpine: Pulling from library/python
9d48c3bd43c5: Pull complete
c0ea575d71b9: Pull complete
0f535ecee5bd5: Pull complete
8a30f5893bea: Pull complete
cid30ace7b67: Pull complete
Digest: sha256:9363cb46e52894a22ba87ebec0845d30f4c27efd6b907705ba9a27192b45e797
Status: Downloaded newer image for python:alpine
--> 39fb80313465
Step 2/4 : workdir /home
--> Running in dcd189072469
Removing intermediate container dcd189072469
--> da185e804ddd
Step 3/4 : copy . /home
--> 23791e877fe4
Step 4/4 : cmd ["python","run.py"]
--> Running in 57d6d7e1a67d
Removing intermediate container 57d6d7e1a67d
--> 81acfb168a45
Successfully built 81acfb168a45
```

## (3) 测试镜像

有两种方式，我们先看简单的那种。

```
docker run test
```

注意：**这里的 test** 是指我们在 **build** 镜像的时候输入的名字

```
qm@qmzhang:/mnt/shareFiles/test$ docker run test
hello world!
```

第二种是交互式运行，如下命令：

```
docker run -it test sh
```

效果如下：我们进入了一个新的系统，目录为/home，可输入命令，如 ls、python xx.py 等。

```
qm@qmzhang:/mnt/shareFiles/test$ docker run -it test sh
/home # ls
Dockerfile  run.py
/home # python run.py
hello world!
/home #
```

在这个/home 目录下，包含我们一开始创建的 Dockerfile 和 run.py 两个文件，这是 Dockerfile 中两个命令作用的结果：

```
workdir /home
```

```
copy . /home
```

意思分别是：新建工作目录/home，以及把**当前目录下**的所有内容 copy 到/home 下。

## 2.3 理解镜像与容器

### (1) 镜像与容器

从上例，我们可以看出，镜像就像一个系统的安装包，我们可以把需要的内容，一并加在这个镜像里面，从而创建我们自己独特的镜像（即，包含我们自定义内容的安装包）。

`docker run` 命令，就相当于我们把这个镜像安装到了某个硬件上，成为一个独立的系统，此时也就是我们常说的容器。因此，容器，就是应用镜像的结果。一个镜像，可以同时启动多个容器。

`docker run -it XXX sh`，就要进入到 XXX 这个容器内，就像进入了一台新的电脑。

也可以将 `sh` 替换为 `/bin/bash`，以这种方式运行，可以查看当前容器的 id。

### (2) 镜像之 FROM `python:alpine`

`FROM python:alpine` 命令是指，我们从 image 仓库中，在 `python` 位置拉取了一个标签为 `alpine` 的系统，作为我们这个镜像的底层系统。

也就是说，我们可以在别的镜像的基础上，再进行封装。

### (3) 镜像的层

上面的 `Dockerfile` 一共有 4 个命令，在创建镜像的时候，当一个命令被执行，就会有进度显示 `step 1/4`、`step 2/4` 等，这个既是分步执行的意思，也是分层执行。

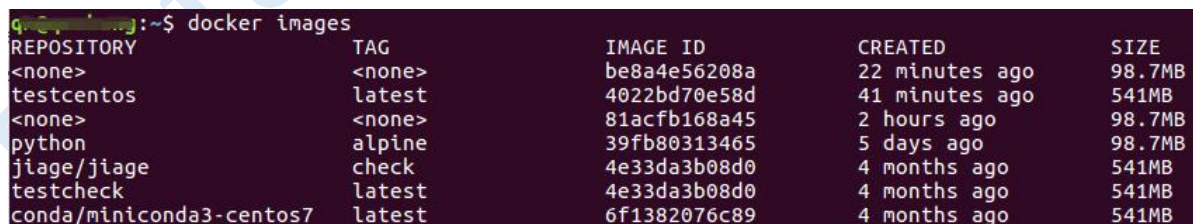
在我们第一次创建一个镜像的时候，通常会非常慢；但是当第二次运行创建，速度就会快很多，这也得益于“分层”执行。

## 2.4 删除镜像

### (1) 查看镜像命令

`docker images`

或者 `docker image ls`



| REPOSITORY               | TAG    | IMAGE ID     | CREATED        | SIZE   |
|--------------------------|--------|--------------|----------------|--------|
| <none>                   | <none> | be8a4e56208a | 22 minutes ago | 98.7MB |
| testcentos               | latest | 4022bd70e58d | 41 minutes ago | 541MB  |
| <none>                   | <none> | 81acfb168a45 | 2 hours ago    | 98.7MB |
| python                   | alpine | 39fb80313465 | 5 days ago     | 98.7MB |
| jiage/jiage              | check  | 4e33da3b08d0 | 4 months ago   | 541MB  |
| testcheck                | latest | 4e33da3b08d0 | 4 months ago   | 541MB  |
| conda/miniconda3-centos7 | latest | 6f1382076c89 | 4 months ago   | 541MB  |

### (2) 删除镜像命令

假设我们要删除第一个镜像，镜像 id 为 `be8a4e56208a`（可用前三位代替），删除命令为

`docker image rm be8`

但是我们发现，它并不能被删除，因为有依赖于它的容器正在运行。如下图。



```
root@~$ docker image rm be8
Error response from daemon: conflict: unable to delete be8a4e56208a (cannot be forced) - image is being used by running container d0325c6c3f46
```

此时，我们需要先停止正在运行的容器，然后再删除容器，之后才能删除镜像。停止容器试用 stop 命令，删除容器则用 rm。

```
ing used by running container d0325c6c3f46
root@~$ docker container stop d03
d03
root@~$ docker container rm d03
d03
root@~$ docker image rm be8
Error response from daemon: conflict: unable to delete be8a4e56208a (must be forced) - image is being used by stopped container 47db860efdfd
root@~$ docker container stop 47d
47d
root@~$ docker container rm 47d
47d
root@~$ docker image rm be8
Deleted: sha256:be8a4e56208aa746cb2c5f8318775804a92ddf57464d7929321d45142bf5d8a4
Deleted: sha256:d0b37753eb6ce8b93ece2a56bee8c9ce8eaf92e29a2c0d0bc8cc9fb913f86630
root@~$
```

### (3) 批量删除镜像命令 (须谨慎使用)

步骤 1：批量停止容器

```
docker container stop $(docker container ls -aq)
```

步骤 2：批量删除容器：

```
docker container rm $(docker container ls -aq)
```

步骤 3：批量删除镜像：

```
docker image rm $(docker image ls -aq)
```

## 三、创建镜像标准样例

### 3.1 选择镜像建议

在上面的例子中，为了帮助大家快速理解镜像和容器，我们使用了 python:alpine，因为它体积较小，构造速度较快。然而，它对于初学者来讲并不那么友好，pip 安装会有较多麻烦。

对于没有经验的小伙伴，可直接使用 **from conda/miniconda3-centos7**，python 版本是 3.7。

### 3.2 镜像必须包含的文件/文件夹

- (1) Dockerfile：用户规定镜像的构建规则；若缺失，镜像不能建立。
- (2) run.py：用户模型/工程的入口文件；若缺失，选手模型不能执行。
- (3) 选手的源代码：可以是单个 py 文件，也可以是工程文件，需要有 run.py 启动。
- (4) requirements.txt：若缺失，选手模型运行可能出现依赖包缺失的错误；requirements 里面的依赖包，也可以放到 Dockerfile 中安装。

如下图



### 3.3 Dockerfile 文件编辑

```
from conda/miniconda3-centos7
workdir /code
copy . /code
run pip install -i https://pypi.tuna.tsinghua.edu.cn/simple -r requirements.txt
```

红色部分代码，不可更改！

### 3.4 Requirements.txt 文件

根据自己的需求，选择尽可能少的依赖包，以减小镜像的大小，缩短上传时间和计算等待时间。

### 3.5 run.py 文件编辑

run.py 文件是整个工程文件的启动文件，具体形式可参照示例文件。

```
import numpy as np
import os
import pandas as pd
import testmodel
```

```
if __name__ == "__main__":
```

```
##### 不可修改区域开始
#####
testpath = '/home/data/' #测试集路径。包含验证码图片文件
```

```
result_folder_path = '/result/submission.csv'    #结果输出文件路径
#####      不      可      修      改      区      域      结      束
#####

### 调用自己的工程文件，并这里生成结果文件（datafram）
result = testmodel.model(testpath)
print(result)
# 注意路径不能更改，index 需要设置为 None
result.to_csv(result_folder_path, index=None)
### 参考代码结束：输出标准结果文件
```

### 3.6 Testmodel.py 文件编辑

Testmodel.py 文件是选手的模型文件，在 run.py 文件中调用此文件用于读取模型并生成结果文件。

```
import numpy as np
```

```
import pandas as pd
```

```
def model(testpath):
```

```
    #your model goes here
```

```
    # 在这里放入或者读入模型文件
```

```
    pass
```

```
    # the format of result-file
```

```
# 这里可以生成结果文件

ids = [str(x) + ".jpg" for x in range(1, 11)]

labels = ['test'] * 10

df = pd.DataFrame([ids, labels]).T

df.columns = ['ID', 'label']

return df
```

### 3.7 创建镜像

在 Dockerfile 所在目录下打开终端，输入命令：

```
docker build -t test_docker .
```

其中 test\_docker 可根据自己喜好命名。

### 3.8 挂载目录

我们需要通过挂载目录的形式，来读取本地数据运行。选手可以按照示例 run.py 文件中的路径相应在本地图建文件夹，在本地图进行测试。

需要通过挂载本地的路径，来实现 docker 的访问，挂载命令如下：

```
映射命令 -v 本地路径:docker 路径
```

具体命令如下：

```
docker run -it -v /mnt/shareFiles/finalTest:/home/data user9 /bin/bash
```

上面命令实现功能就是，将本地路径 /mnt/shareFiles/finalTest 下的内容，映射到 /home/data 下，如果 docker 中不存在此文件夹，则新创建一个。

```
exit
qm@qmqzhang:/mnt/shareFiles$ docker run -it user9 /bin/bash
root@a54ceaddf98a:/code# ls
Dockerfile Train model.py requirements.txt result run.py
root@a54ceaddf98a:/code# exit
exit
qm@qmqzhang:/mnt/shareFiles$ docker run -it -v /mnt/shareFiles/finalTest:/home/data user9 /bin/bash
root@80e823d3134b:/code# ls
Dockerfile Train model.py requirements.txt result run.py
root@80e823d3134b:/code# cd ..
root@80e823d3134b:/# ls
bin boot code dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@80e823d3134b:/# cd home/
root@80e823d3134b:/home# cd data/
root@80e823d3134b:/home/data# ls
ddlz_evaluation.py image result1.csv result11.csv result4.csv result8.csv result9.csv test.csv track
root@80e823d3134b:/home/data# exit
exit
qm@qmqzhang:/mnt/shareFiles$ docker run -it user9 /bin/bash
root@2fb727dd25cc:/code# ls
Dockerfile Train model.py requirements.txt result run.py
root@2fb727dd25cc:/code# cd ..
root@2fb727dd25cc:/# ls
bin boot code dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@2fb727dd25cc:/# cd home/
root@2fb727dd25cc:/home# ls
root@2fb727dd25cc:/home# exit
exit
```

### 3.9 ★封装镜像★

用一个命令，就可以简单解决。

```
docker save test_docker:latest | gzip > test_docker_today.tar.gz
```

其中蓝色、斜体的部分，需要替换为自己的文件名。

(a) test\_docker:latest 的含义

- a) test\_docker，为自己在 3.4 章节中创建的镜像名；
- b) latest 为该镜像默认的标签。（此处我们不讲解标签方面的知识）

(b) test\_docker\_today 为自定义的名字。此处也可指定路径，若不指定路径，则为终端（terminal）当前显示的路径。

最后将生成的结果文件提交即可。

## 四、使用与建议

欢迎大家转发、传阅。

有任何建议，可发送至邮箱 [competition@datacastle.cn](mailto:competition@datacastle.cn)