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 2](#_Toc24728057)

[1.1 虚拟机安装与配置 2](#_Toc24728058)

[1.1.1 安装VirtualBox和Linux系统 2](#_Toc24728059)

[1.1.2 建立与Windows可同步的文件夹 2](#_Toc24728060)

[1.2 安装docker 4](#_Toc24728061)

[二、 竞赛中使用docker 的必要命令 5](#_Toc24728062)

[2.1 启动docker 5](#_Toc24728063)

[2.2 Docker使用的hello World教程 5](#_Toc24728064)

[2.3 理解镜像与容器 7](#_Toc24728065)

[2.4 删除镜像 7](#_Toc24728066)

[三、 创建镜像标准样例 8](#_Toc24728067)

[3.1 选择镜像建议 8](#_Toc24728068)

[3.2 镜像必须包含的文件/文件夹 8](#_Toc24728069)

[3.3 Dockerfile文件编辑 9](#_Toc24728070)

[3.4 Requirements.txt文件 9](#_Toc24728071)

[3.5 run.py文件编辑 9](#_Toc24728072)

[3.6 Testmodel.py文件编辑 10](#_Toc24728073)

[3.7 创建镜像 10](#_Toc24728074)

[3.8 ★封装镜像★ 11](#_Toc24728075)

[3.9 测试docker 11](#_Toc24728076)

[四、 使用与建议 12](#_Toc24728077)

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。

虚拟机安装与配置

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>。

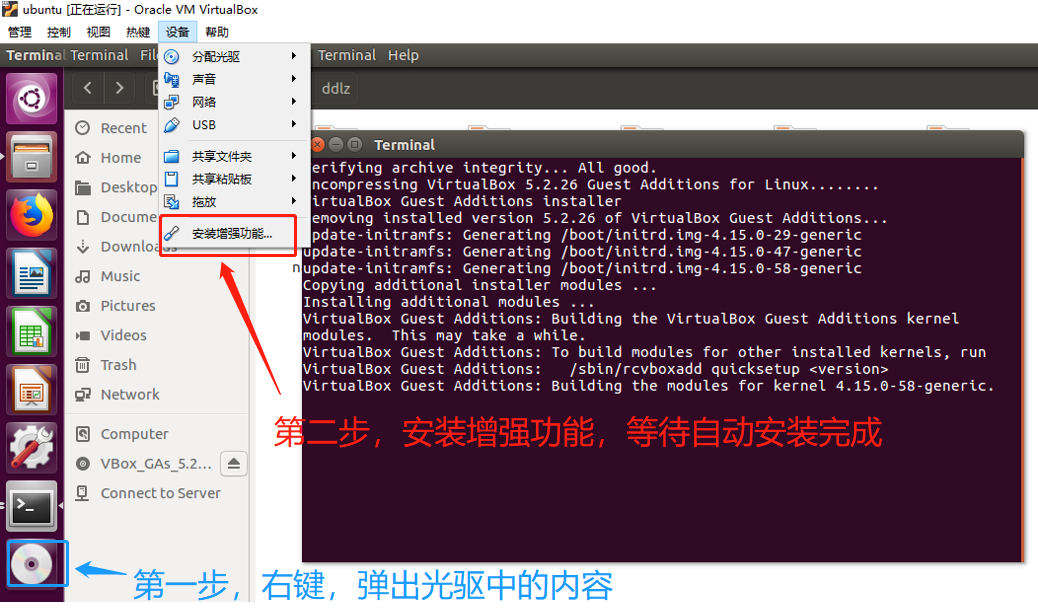
建立与Windows可同步的文件夹

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。**

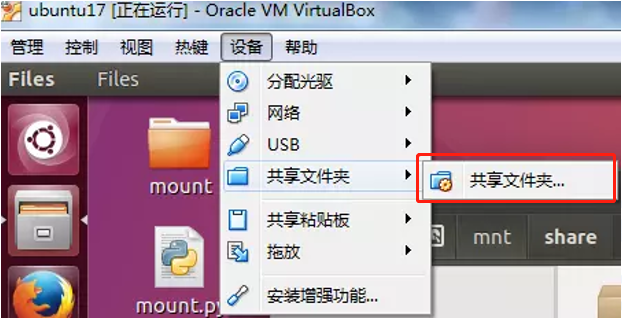
如下图，蓝色字体的“第一步”，红色字体的“第二步”。

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。

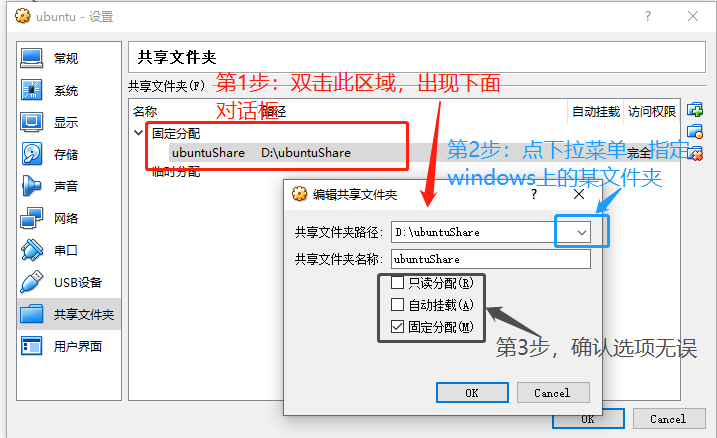


1. **配置共享文件夹**

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。



1. **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：

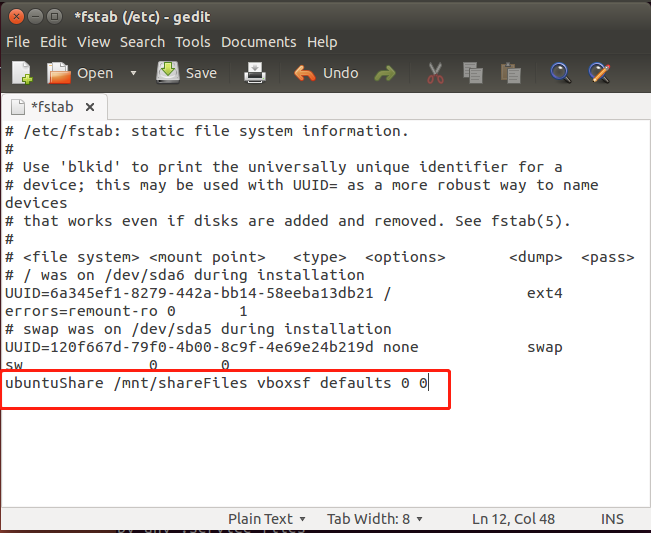
sudo mount -t vboxsf ubuntuShare /mnt/shareFiles

1. **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



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

1. **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"

1. **安装docker**

sudo apt-get update

sudo apt-get install docker-ce

1. **查看docker版本**

docker --version

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。

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

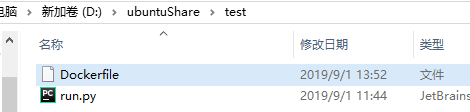
newgrp - docker

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。



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。**

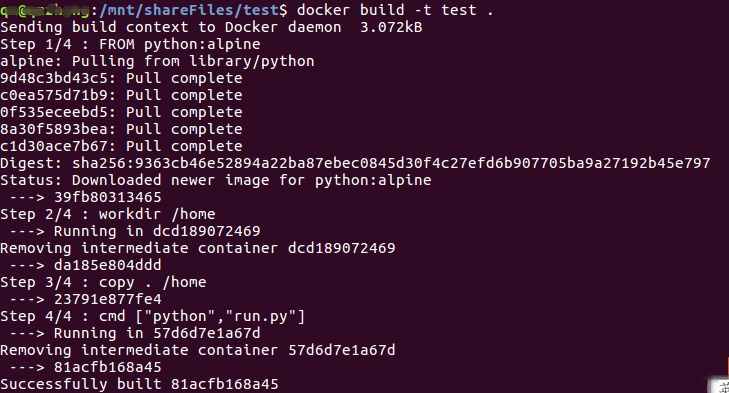
1. **创建镜像**

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“.”。**



1. **测试镜像**

有两种方式，我们先看简单的那种。

docker run test

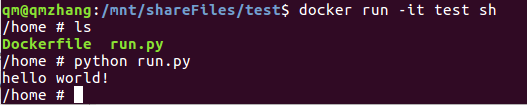
**注意：这里的test是指我们在build镜像的时候输入的名字**



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。



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。

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。

1. **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。

1. **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。

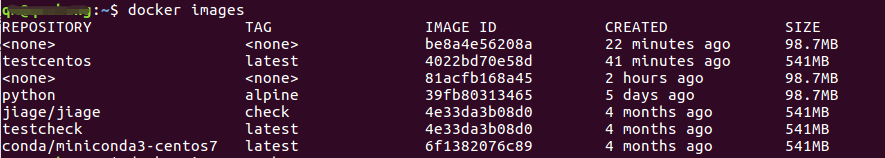
在我们第一次创建一个镜像的时候，通常会非常慢；但是当第二次运行创建，速度就会快很多，这也得益于“分层”执行。

删除镜像

1. **查看镜像命令**

docker images

或者docker image ls



1. **删除镜像命令**

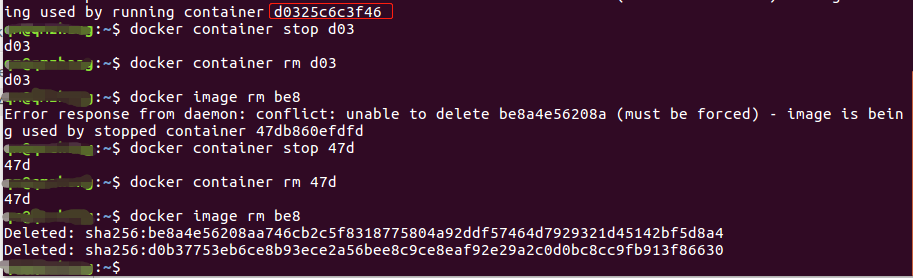
假设我们要删除第一个镜像，镜像id为be8a4e56208a（可用前三位代替），删除命令为

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。



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。



1. **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)

创建镜像标准样例

选择镜像建议

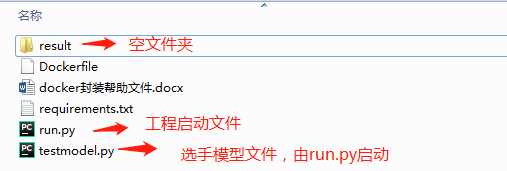
在上面的例子中，为了帮助大家快速理解镜像和容器，我们使用了python:alpine，因为它体积较小，构造速度较快。然而，它对于初学者来讲并不那么友好，pip安装会有较多麻烦。

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。

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



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！**

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。

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

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

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。

★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。

1. test\_docker:latest的含义
   1. test\_docker，为自己在3.4章节中创建的镜像名；
   2. latest为该镜像默认的标签。（此处我们不讲解标签方面的知识）
2. 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。

* 1. 测试docker

我们提交前需要测试封装好的docker是否能够运行，首先要通过挂载目录的形式来读取本地数据，然后运行docker。选手可以按照示例run.py文件中的路径相应在本地创建文件夹，在本地进行测试。

需要通过挂载本地的路径，来实现docker的访问，挂载命令如下：

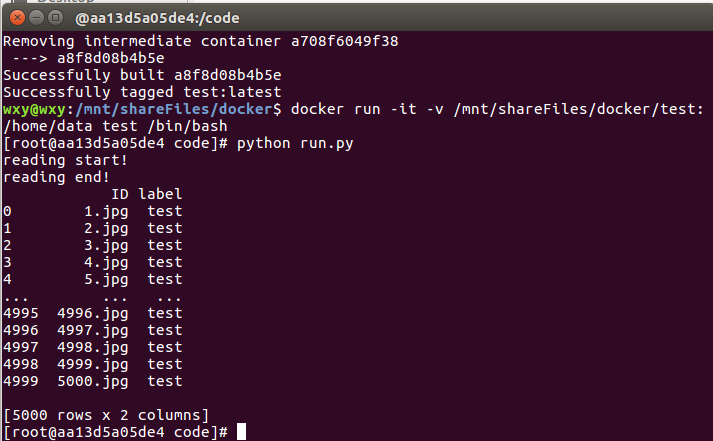
映射命令 -v 本地路径:docker路径

具体命令如下：

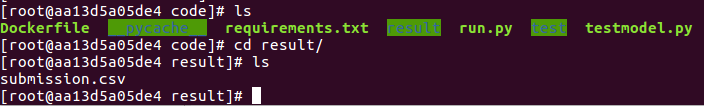
docker run -it -v /mnt/shareFiles/docker/test:/home/data test\_docker /bin/bash

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，test\_docker是要运行的docker镜像的名称。

完成上述命令后，已经进入到docker容器中，接下来运行python run.py测试成功即可。



最后我们确认result文件夹中生成了submission.csv结果文件，到此测试完成！



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