

- [Docker有啥用](#)
- [Docker的安装](#)
- [设置国内镜像](#)
- [测试Docker](#)
 - [运行hello-world](#)
 - [运行python](#)
 - [使用CentOS](#)
 - [安装CentOS](#)
 - [后台运行CentOS](#)
 - [attach运行着的镜像](#)
 - [重启或者终止](#)
 - [制作自己的Docker镜像](#)
 - [使用进阶](#)
 - [挂载物理机目录](#)
 - [端口映射](#)
 - [如果你需要AI](#)
 - [你需要自己定制镜像吗？](#)
- [总结](#)

Docker有啥用

Docker是一款开源应用容器引擎，我们可以认为Docker是运行于物理主机之上的虚拟机，在这个虚拟机上，可以运行各种各样的程序（甚至操作系统）。为什么要在虚拟机上运行程序，程序不是直接可以装在物理机上（比如windows系统）中吗？下面我们来看几个典型应用场景：

1. 你需要学习Linux操作系统，但你只有一台Windows电脑。
2. 你需要学习ROS（机器人操作系统），但你只有一台Windows电脑。
3. 你需要学习vue开发，为此你需要安装Node.js、NPM、vue-cli等一系列工具，但你只有一台Windows电脑。
4. 你在Windows电脑上开发了一个网站，该网站需要Python、Nginx服务器、MySQL数据库等运行环境，你希望在同学的电脑上部署该网站，但你的同学是一个果粉，他只有Mac电脑。
5. 你是一个新手程序员，你想学习Python、C++、Java，但你不会安装开发环境。

以上的问题均可以使用Docker轻松解决。我们可以将Linux、ROS、Python、C++、Java等打包为Docker镜像（Docker Image），然后将镜像在Docker虚拟机上运行（如下图所示）。用户在使用时，尽管使用的是Docker虚拟机中的程序，但用户无法分辨程序是运行在物理机上还是虚拟机上。由于Docker是跨平台的，因此只要物理机可以安装Docker，那么就能运行任何Docker镜像。

物理主机

Docker

Python镜像

C++镜像

Linux镜像

Java镜像

简言之，对于场景1-5，只要有对应的Docker镜像，就能立刻运行镜像，然后开始使用软件。这就为跨平台部署、软件配置等工作提供了极大的便利。对于新手程序员的，如果不想把时间浪费在环境的安装、配置上，使用Docker也可以极大简化学习过程。同时，新手程序员也可以在Docker上学习各种软件的安装和配置，而不用担心搞坏物理主机（因为虚拟机是运行在物理机上的沙箱SandBox，你可以随便折腾，搞坏了重新建一个就行）。

说一千道一万，不如实际用一遍。下面我们就来学习一下Docker的使用。本文档不如[菜鸟Docker教程](#)全面，建议你在阅读本文档之后，进一步阅读[菜鸟Docker教程](#)。或者在遇到问题是，阅读[菜鸟Docker教程](#)。

Docker的安装

在使用Docker之前，你需要学会安装Docker。各种操作系统安装Docker的教程可以参考：

1. [Windows](#)
2. [Mac](#)
3. [CentOS](#)
4. [官方文档](#)，菜鸟教程可能较官方已经过时，因此如果遇到安装问题，建议直接按照官方文档安装

Windows和Mac的安装都非常傻瓜化，下载安装包，一通Next即可。

注意！！

使用Windows10 home edition安装docker会遇到一些麻烦，以下是某位同学的安装流程，供各位参考。

首先，进到官网下载Windows版本的Docker Desktop，此时Docker已经有了，环境没配置好。然后，以管理员身份打开PowerShell并运行dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart，这是安装WSL 1。接下来，看系统是否满足运行WSL 2的要求：

若要更新到 WSL 2，需要运行 Windows 10。

- 对于 x64 系统：版本 1903 或更高版本，采用 内部版本 18362 或更高版本。
- 对于 ARM64 系统：版本 2004 或更高版本，采用 内部版本 19041 或更高版本。
- 低于 18362 的版本不支持 WSL 2。使用 [Windows Update 助手](#)更新 Windows 版本。

若要检查 Windows 版本及内部版本号，选择 Windows 徽标键 + R，然后键入“winver”，选择“确定”。（或者在 Windows 命令提示符下输入 `ver` 命令）。更新到“设置”菜单中的[最新 Windows 版本](#)。

① 备注

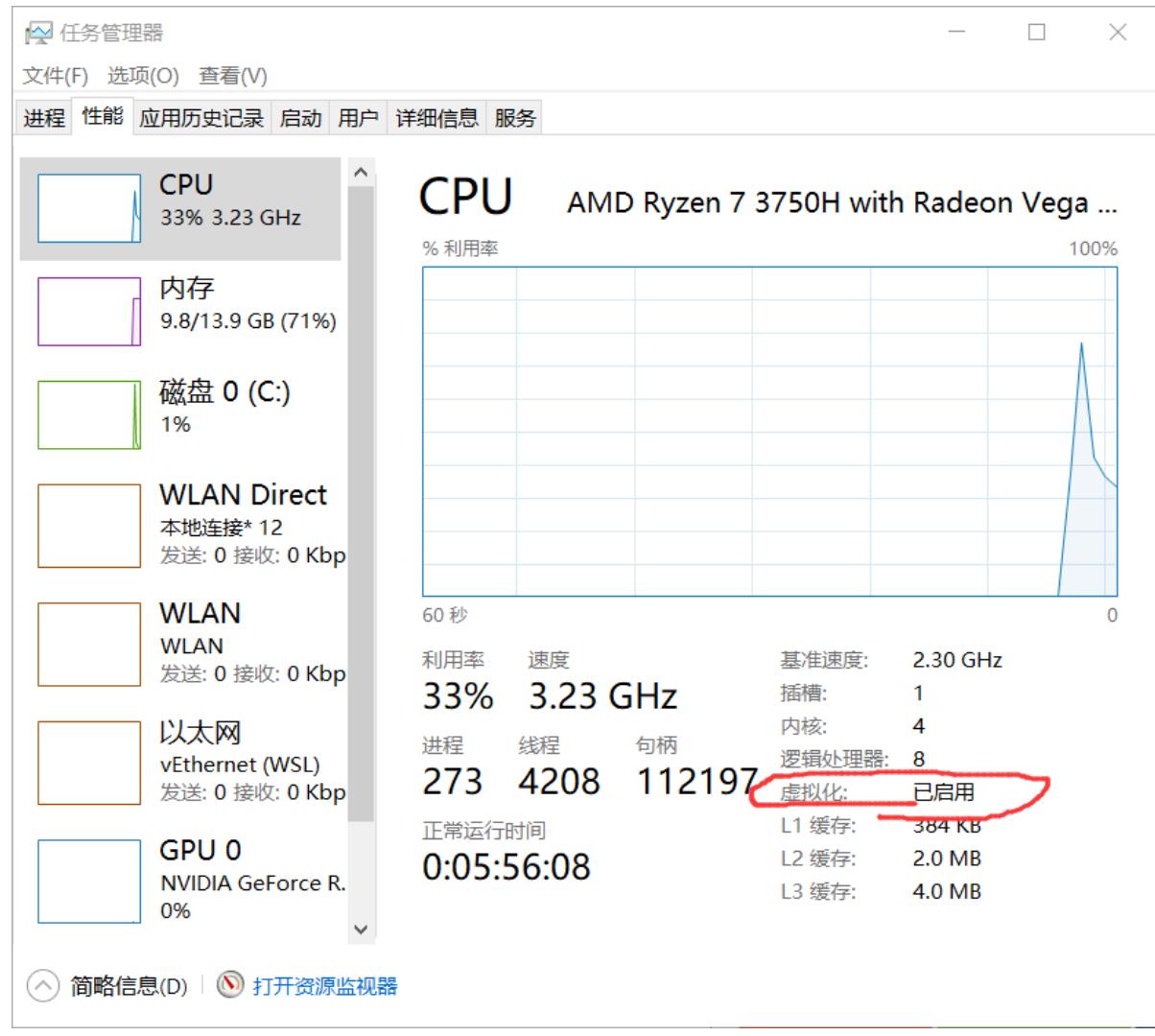
如果运行的是 Windows 10 版本1903 或 1909，请在 Windows 菜单中打开“设置”，导航到“更新和安全性”，然后选择“检查更新”。内部版本号必须是 18362.1049+ 或 18363.1049+，次要内部版本号需要高于 .1049。阅读详细信息：[WSL 2 即将支持 Windows 10 版本 1903 和 1909](#)。请参阅[疑难解答说明](#)。

然后再次以管理员身份在PowerShell里面运行：

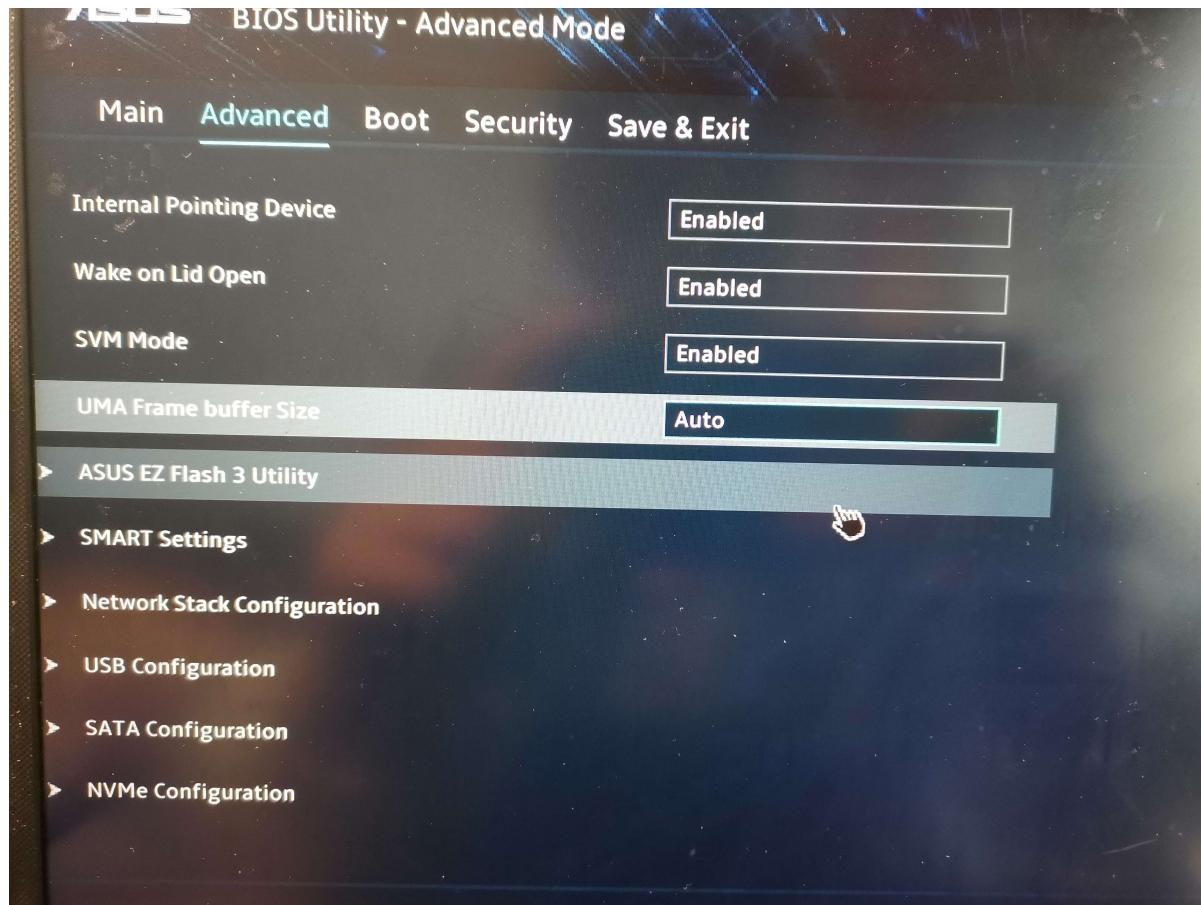
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart，重启电脑，以完成WSL安装并更新到WSL 2（一定要在这里重启）。

接下来下载Linux内核更新包（这是适用于x64计算机的）https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi

如果是其他机型，应该要下载相应的其他包。安装完成要点击运行完成安装，到这里应该就算完成了。要想知道CPU是否打开了VT，可以打开任务管理器，



看这个地方是否是“已启用”，如果不是的话，就需要重启电脑，然后使劲按“F2”，也可能是“DELETE”键——不同电脑不一样，进入BIOS界面。在Advanced Mode里找到跟VT (Virtual Technology) 有关的东西，



比如我这里是SVM Mode，把它设置为Enabled模式，就算启动了VT。但我这里遇到了个坑，就是SVM下面的UMA Frame buffer size在我设置SVM的时候自动变成了512M，然后电脑就进入不了系统——黑屏，我把它改成Auto就可以了。进入系统之后，WSL 2就能用了，Docker应该就能成功运行了（不能的话可能需要多打开几次Docker，我就是这样，前几次竟然还不行）。

设置国内镜像

Docker镜像的一个重要来源是[DockerHub](#)，这个网站在国内访问起来很慢，因此需要设置国内的加速。设置方法参考如下教程：

1. [Docker镜像加速](#)

如果你对网速有信心，或者你搞不懂镜像的设置，可以跳过这一步。使用时除了慢点，也没啥大毛病。

测试Docker

安装好Docker，也设置好镜像，下面就可以测试一下Docker的使用。我们来做两个例子。

运行hello-world

切换到命令行下（对windows命令行操作不熟悉的同学，可以看[这里](#)），执行如下指令：

```
1 | docker run hello-world
```

如果执行成功，你将看到如下输出(可能会略有区别)：

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Already exists
Digest: sha256:1a523af650137b8accdaed439c17d684df61ee4d74feac151b5b337bd29e7eec
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

```
https://hub.docker.com/
```

For more examples and ideas, visit:

```
https://docs.docker.com/get-started/
```

执行如下指令：

```
1 | docker images
```

你可以看到一个名称为 `hello-world` 的镜像已经从 DockerHub 上拉取下来了，`docker run` 执行了这个镜像。

fauria/vsftpd	latest	001e3f235a25	7 months ago	318MB
<u>hello-world</u>	latest	bf756fb1ae65	11 months ago	13.3kB
schnitzler/mysqldump	latest	ec7428bc6574	17 months ago	41MB
openjdk	8-jdk-alpine	a3562aa0b991	19 months ago	105MB
willfarrell/crontab	latest	f449b626ddf4	21 months ago	181MB

运行 python

下面我们用 Docker 来使用 python，先利用 `search` 指令查找 DockerHub 上的 python 镜像

```
1 | docker search python
```

显示如下结果（可能会有差异）：

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
python	Python is an interpreted, interactive, objec...	5695	[OK]	
django	Django is a free web application framework, ...	1028	[OK]	
pypy	PyPy is a fast, compliant alternative implem...	257	[OK]	
nikolaik/python-nodejs	Python with Node.js	56		[OK]
joyzoursky/python-chromedriver	Python with Chromedriver, for running automa...	54		[OK]
arm32v7/python	Python is an interpreted, interactive, objec...	53		
circleci/python	Python is an interpreted, interactive, objec...	42		
centos/python-35-centos7	Platform for building and running Python 3.5...	38		
centos/python-36-centos7	Platform for building and running Python 3.6...	30		
hylang	Hy is a Lisp dialect that translates express...	29	[OK]	
arm64v8/python	Python is an interpreted, interactive, objec...	24		
revolutionsystems/python	Optimized Python Images	18		
centos/python-27-centos7	Platform for building and running Python 2.7...	17		
bitnami/python	Bitnami Python Docker Image	10		[OK]
publicisworldwide/python-conda	Basic Python environments with Conda.	6		[OK]
dockershelf/python	Repository for docker images of Python. Test...	5		[OK]
d3fk/python_in_bottle	Simple python:alpine completed by Bottle+Req...	4		[OK]
clearlinux/python	Python programming interpreted language with...	4		
i386/python	Python is an interpreted, interactive, objec...	3		
ppc64le/python	Python is an interpreted, interactive, objec...	2		
centos/python-34-centos7	Platform for building and running Python 3.4...	2		
amd64/python	Python is an interpreted, interactive, objec...	1		
ccitest/python	CircleCI test images for Python	0	[OK]	
s390x/python	Python is an interpreted, interactive, objec...	0		
saagie/python	Repo for python jobs	0		

我们安装并运行第一个

```
1 | docker pull python
```

该指令只从网上拉取python但不运行（拉取过程如下图）

```
Using default tag: latest
latest: Pulling from library/python
6c33745f49b4: Downloading [=====] 41.63MB/50.4MB
c87cd3c61e27: Download complete
05a3c799ec37: Download complete
a61c38f966ac: Downloading [=====] 24.56MB/51.83MB
c2dd6d195b68: Downloading [==>] 14.25MB/192.3MB
29b9446ae7bd: Waiting
09cf96c794f9: Waiting
f674fd97fba7: Waiting
ffc24df6b7b8: Waiting
```

要想运行还需要执行

```
1 | docker run python
```

居然啥也没有！！其实正确的指令是：

```
1 | docker run -i -t python
```

-i 表示以 interactive 方式运行， -t 表示为用户分配一个 tty (百度一下 tty 是啥意思)，于是就能开启交互式的 python 窗口，如下图：

```
Python 3.9.1 (default, Dec 12 2020, 13:15:12)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 'hello world'
'hello world'
>>> 1+2
3
>>> a=[1,2,3,4]
>>> print(a)
[1, 2, 3, 4]
>>> █
```

没有安装任何Python软件，但我们先做可以用Python编程了！在刚才那条指令中，`python`是DockerHub上的Docker镜像的名字。`docker pull`拉取了`python`镜像，`docker run`执行了`python`镜像。现在我们应该记住两个常用的docker指令：

1. `docker pull [镜像名称]`，从DockerHub上拉取一个镜像。
2. `docker run [镜像名称]`，运行镜像，如果镜像不存在，则从DockerHub上拉取。

使用CentOS

安装CentOS

[CentOS](#)是一款Linux操作系统。Linux操作系统在服务器端、移动设备和嵌入式设备中广泛使用，是目前使用最为广泛的操作系统（Android手机均运行Linux）。要在个人电脑上安装CentOS是非常复杂的，例如B站上的视频，也是用VMware这款虚拟机来安装（视频见[这里](#)）。现在我们用Docker来安装、使用CentOS。首先搜索一下：

```
1 | docker search centos
```

有很多CentOS镜像

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
centos	The official build of CentOS.	6318	[OK]	
ansible/centos7-ansible	Ansible on Centos7	132	[OK]	
consol/centos-xfce-vnc	Centos container with "headless" VNC session...	123	[OK]	
jdeathe/centos-ssh	OpenSSH / Supervisor / EPEL/IUS/SCL Repos - ...	116	[OK]	
centos/systemd	systemd enabled base container.	87	[OK]	
centos/mysql-57-centos7	MySQL 5.7 SQL database server	86		
imagine10255/centos6-lnmp-php56	centos6-lnmp-php56	58	[OK]	
tutum/centos	Simple CentOS docker image with SSH access	46		
centos/postgresql-96-centos7	PostgreSQL is an advanced Object-Relational ...	45		
kinogmt/centos-ssh	CentOS with SSH	29	[OK]	
pivotaldata/centos-gpdb-dev	CentOS image for GPDB development. Tag names...	13		
guyton/centos6	From official centos6 container with full up...	10	[OK]	
centos/tools	Docker image that has systems administration...	7	[OK]	
drecom/centos-ruby	centos ruby	6	[OK]	
pivotaldata/centos	Base centos, freshened up a little with a Do...	5		
darksheer/centos	Base Centos Image -- Updated hourly	3		[OK]
pivotaldata/centos-mingw	Using the mingw toolchain to cross-compile t...	3		
mamohr/centos-java	Oracle Java 8 Docker image based on Centos 7	3		[OK]
pivotaldata/centos-gcc-toolchain	CentOS with a toolchain, but unaffiliated wi...	3		
mcaughton/centos-base	centos base image	1		[OK]
blacklabelops/centos	CentOS Base Image! Built and Updates Daily!	1		[OK]
indigo/centos-maven	Vanilla CentOS 7 with Oracle Java Developmen...	1		[OK]
pivotaldata/centos6.8-dev	CentosOS 6.8 image for GPDB development	0		
pivotaldata/centos7-dev	CentosOS 7 image for GPDB development	0		
smartentry/centos	centos with smartentry	0		[OK]

我们使用第一个

```
1 | docker pull centos
```

下载完成后，使用 `docker images` 指令，可以看到下载后的centos镜像，如下：

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
python	latest	0611cf846c85	2 days ago	885MB
nginx	latest	0efd8175f035	4 days ago	140MB
centos	latest	300e315adb2f	6 days ago	209MB
mysql	latest	b8ee0929b718	2 months ago	448MB
<none>	<none>	ef6b9db7043a	2 months ago	448MB
nodejs	<none>	2bc782858686	2 months ago	448MB

用 `docker run` 启动镜像

```
1 | docker run -i -t centos
```

然后就可以进入centos系统，并执行指令，如下：

```
[root@09d0381e8d6a /]# ls      运行ls指令
bin dev etc home lib lib64 lost+found media mnt opt proc root run sbin srv sys tmp usr var
[root@09d0381e8d6a /]# cat /etc/issue   运行cat
\$
Kernel \r on an \m

[root@09d0381e8d6a /]# rpm -q centos-release    运行rpm
package centos-release is not installed
[root@09d0381e8d6a /]# lsb_release -a    运行lsb_release
bash: lsb_release: command not found
[root@09d0381e8d6a /]# lsblk      运行lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda 254:0 0 59.6G 0 disk
`-vda1 254:1 0 59.6G 0 part /etc/hosts
[root@09d0381e8d6a /]# df      运行df
Filesystem 1K-blocks Used Available Use% Mounted on
overlay 61255492 16352180 41761988 29% /
tmpfs 65536 0 65536 0% /dev
tmpfs 1018048 0 1018048 0% /sys/fs/cgroup
shm 65536 0 65536 0% /dev/shm
/dev/vda1 61255492 16352180 41761988 29% /etc/hosts
tmpfs 1018048 0 1018048 0% /proc/acpi
tmpfs 1018048 0 1018048 0% /sys/firmware
[root@09d0381e8d6a /]# yum      运行yum
Failed to set locale, defaulting to C.UTF-8
usage: yum [options] COMMAND

List of Main Commands:

alias          List or create command aliases
autoremove     remove all unneeded packages that were originally installed as dependencies
check          check for problems in the packagedb
check-update   check for available package upgrades
clean          remove cached data
deplist        List package's dependencies and what packages provide them
distro-sync    synchronize installed packages to the latest available versions
downgrade     Downgrade a package
group          display, or use, the groups information
```

现在你还不理解centos能做什么，目前你只需要知道centos是和Windows一样强大的操作系统（应该是更强大）。

在centos中执行 `exit` 就可以退出，退出之后centos就终止了。

后台运行CentOS

如果我们希望centos一直运行呢？下面我们来看看如何让centos在后台运行。

执行如下指令：

```
1 | docker run -d -i -t --name centosA centos
2 | docker run -d -i -t --name centosB centos
```

每次执行都会出现了一串奇怪的数字。请注意 `-d` 这个参数，它的含义是让运行的虚拟机在后台运行。简言之，执行 `exit` 会退出centos，但centos操作系统不会终止运行。我们执行如下指令就可以看到运行中的centos：

```
1 | docker container ls
```

或者

```
1 | docker ps
```

可以看到运行着的docker镜像

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2d132a932c16	centos	"/bin/bash"	About a minute ago	Up About a minute		centosA
54b309f8b694	centos	"/bin/bash"	About a minute ago	Up About a minute		centosB

`--name` 的作用则是为运行的镜像加上名字，方便后续的操作。

attach运行着的镜像

现在执行操作 `docker attach centosA` 就可以进入名称为 `centosA` 的docker镜像。如果执行 `docker attach centosB` 就可以进入名称为 `centosB` 的docker镜像。

试试如下操作：

1. docker attach centosA
2. touch centosA.txt
3. ls /
4. exit

接着

1. docker attach centosA
2. touch centosB.txt
3. ls /
4. exit

效果如下：

```
liuji@LiuJi-MacBook-Pro-2018 ~ % docker attach centosA
[root@2d132a932c16 /]# touch centosA.txt
[root@2d132a932c16 /]# ls /                                         在 centosA 中创建 centosA.txt
bin  centosA.txt  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@2d132a932c16 /]# exit
exit
liuji@LiuJi-MacBook-Pro-2018 ~ % docker attach centosB
[root@54b309f8b694 /]# touch centosB.txt
[root@54b309f8b694 /]# ls /                                         在 centosB 中创建 centosB.txt
bin  centosB.txt  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@54b309f8b694 /]# exit
exit
```

重启或者终止

注意：在执行上述操作时如果遇到centos已经终止，可以执行 `restart` 重新启动镜像：

```
1 | docker restart centosA
2 | docker restart centosB
```

如果想要终止镜像的执行则可以使用 `kill` 指令：

```
1 | docker kill centosA
2 | docker kill centosB
```

制作自己的Docker镜像

Docker最强大的功能在于用户可以定制自己的镜像。考虑下面的应用场景：

```
1 | 刘老师每年给大一新生上Python程序设计课程，需要CentOS系统的计算机，并且安装Python、pip  
    (Python的包管理工具)、numpy、scipy、matplotlib和jupyter等工具。
```

刘老师现在有两个选择：

1. 在实验室中找一台机器，配置好环境，然后把这台机器的环境，克隆到其他计算机。
2. 制作一个Docker镜像。

刘老师一定会选择第二种方式，因为第一种方式需要花费刘老师半天的时间，而第二种方式只需要几分钟。你没看错，第一种方式需要半天，第二种方式只要几分钟。下面给出做法：

1. 找一个文本编辑器，比如记事本。
2. 编写一个名称为Dockerfile的文件。

Dockerfile(见文件[doc/docker/code/centos-python1/Dockerfile](#))中包含如下内容：

```
1 #以centos镜像为基础
2 FROM centos
3 #安装epel (好像这一步不是必须的)
4 RUN yum -y install epel-release
5 #安装python3.8
6 RUN yum -y install python38
7 #用pip3安装numpy scipy matplotlib sympy pandas jupyter
8 RUN pip3 install -i https://mirrors.aliyun.com/pypi/simple/ numpy scipy
matplotlib sympy pandas jupyter
```

你可以直接拷贝上述代码。下面我们生成Docker镜像，在Dockerfile所在目录，执行如下指令：

```
1 docker build -t centos-python1:latest .
```

千万注意最后有一个`.`，表示Dockerfile在执行指令的当前目录。如果你在本文档的根目录，那么执行：

```
1 docker build -t centos-python1:latest doc/docker/code/centos-python1
```

`-t` 表示给生成的镜像加一个名字，格式是`name:tag`，`centos-python1:latest` 就是镜像的名称。执行这个指令之后，你就能够看到镜像的生成过程，如下：

```

liuji@LiuJi-MacBook-Pro-2018 centos-python1 % docker build -t centos-python1:latest .
Sending build context to Docker daemon 2.048kB
Step 1/4 : FROM centos
--> 300e315adb2f
Step 2/4 : RUN yum -y install epel-release
--> Running in fc79f78b94ce
CentOS Linux 8 - AppStream           2.3 MB/s | 6.2 MB  00:02
CentOS Linux 8 - BaseOS             1.2 MB/s | 2.3 MB  00:01
CentOS Linux 8 - Extras              6.0 kB/s | 8.6 kB  00:01
Dependencies resolved.

=====
Package          Architecture Version   Repository  Size
=====
Installing:
epel-release     noarch      8-8.el8    extras       23 k

Transaction Summary
=====
Install 1 Package

Total download size: 23 k
Installed size: 32 k
Downloading Packages:
epel-release-8-8.el8.noarch.rpm      92 kB/s | 23 kB  00:00
-----
Total                      10 kB/s | 23 kB  00:02
warning: /var/cache/dnf/extras-cfb2f07b0021b7e/packages/epel-release-8-8.el8.noarch.rpm: Header V3 RSA
CentOS Linux 8 - Extras           1.6 MB/s | 1.6 kB  00:00
Importing GPG key 0x8483C65D:
Userid      : "CentOS (CentOS Official Signing Key) <security@centos.org>"
Fingerprint: 99DB 70FA E1D7 CE22 7FB6 4882 05B5 55B3 8483 C65D
From        : /etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing      : 1/1
Installing    : epel-release-8-8.el8.noarch 1/1
Running scriptlet: epel-release-8-8.el8.noarch 1/1
Verifying     : epel-release-8-8.el8.noarch 1/1

```

之后用 `docker images` 指令可以查看到生成的镜像

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<u>centos-python1</u>	<u>latest</u>	8e96fd81d2d5	9 seconds ago	811MB
python	latest	0611cf846c85	2 days ago	885MB
nginx	latest	0efd8175f035	4 days ago	140MB
centos	latest	300e315adb2f	7 days ago	209MB
mysql	latest	b8ee0929b718	2 months ago	448MB

下面是激动人心的时刻，执行 `docker run` 启动该镜像，就可以使用 python（安装了 numpy scipy matplotlib sympy pandas jupyter）

```
liuji@LiuJi-MacBook-Pro-2018 ~ % docker run -i -t centos-python1:latest
[root@d577a083e33c /]# ipython
Python 3.8.3 (default, Aug 31 2020, 16:03:14)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.19.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import numpy as np

In [2]: a=np.array([1,2,3])
...
In [3]: b=np.array([4,5,6])      用numpy求a、b向量的内积

In [4]: np.dot(a,b)
Out[4]: 32

In [5]: import sympy

In [6]: x=sympy.Symbol("x")

In [7]: f=x**3+x**2+x+1      用sympy求微分

In [8]: sympy.diff(f,x)
Out[8]: 3*x**2 + 2*x + 1
```

刘老师只需要把写好的Dockerfile拷贝给同学们，就可以为所有人配置好环境（前提是每个人都安装好Docker，并学会基本的使用）。如果对老师安装的环境不满意，调整也非常简单，只需要修改Dockerfile并重新生成镜像即可。

使用进阶

挂载物理机目录

Docker镜像运行在虚拟机中（顺便说一句，运行的镜像称为容器Container），虚拟机无法直接访问物理机的硬盘，物理机也无法直接访问虚拟机的文件。这样具有更高的安全性，不会因为虚拟机程序的崩溃导致物理机的问题。但偶尔我们也需要在虚拟机中访问物理机的文件，因此就需要用到挂载物理机目录。方法很简单，在启动容器时，加上一个参数`-v`，指令如下：

```
1 | docker run -i -t -v [物理机目录]:[虚拟机目录] centos-python1
```

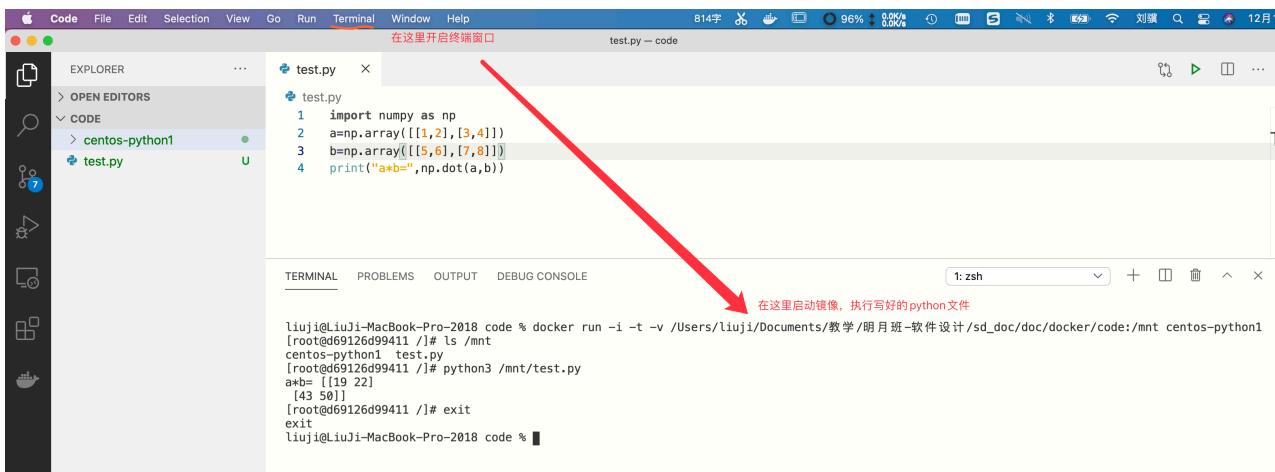
该指令将[物理机目录]映射到[虚拟机目录]，例如：

```
1 | docker run -i -t -v /Users/liuji/Documents/教学/明月班-软件设计/sd_doc/doc/docker/code/centos-python1:/mnt centos-python1
```

就将`/Users/liuji/Documents/教学/明月班-软件设计/sd_doc/doc/docker/code/centos-python1`这个目录映射到了虚拟机的`/mnt`目录下。我们可以看看效果：

```
[liuji@LiuJi-MacBook-Pro-2018 ~ % docker run -i -t -v /Users/liuji/Documents/教学/明月班-软件设计/sd_doc/doc/docker/code/centos-python1:/mnt centos-python1
[root@970622c6d3b0 /]# ls /mnt
Dockerfile
```

下面我们可以利用物理机的VSCode编辑一个python文件，然后在虚拟机中执行。请看截图：



端口映射

jupyter notebook是基于网页的交互式Python开发工具。在刚才的镜像中，我们已经安装了jupyter。执行 `jupyter notebook --allow-root` 就可以启动（如下图）。

```
[root@970622c6d3b0 /]# jupyter notebook --allow-root
[I 01:54:37.514 NotebookApp] Serving notebooks from local directory: /
[I 01:54:37.514 NotebookApp] Jupyter Notebook 6.1.5 is running at:
[I 01:54:37.514 NotebookApp] http://localhost:8888/?token=2db6463a0c9c413f0f5495a4f44b2a3a9a35f96a349f950d
[I 01:54:37.514 NotebookApp] or http://127.0.0.1:8888/?token=2db6463a0c9c413f0f5495a4f44b2a3a9a35f96a349f950d
[I 01:54:37.514 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 01:54:37.518 NotebookApp] No web browser found: could not locate runnable browser.
[C 01:54:37.518 NotebookApp]

To access the notebook, open this file in a browser:
  file:///root/.local/share/jupyter/runtime/nbsviewer-21-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/?token=2db6463a0c9c413f0f5495a4f44b2a3a9a35f96a349f950d
  or http://127.0.0.1:8888/?token=2db6463a0c9c413f0f5495a4f44b2a3a9a35f96a349f950d
```

但通过物理机的浏览器无法访问到notebook。其原因在于notebook所在的8888端口，无法在物理机访问，因此我们需要把端口映射到物理机中。启动时加入 `-p` 参数，如下：

```
1 | docker run -p 8888:8888 -i -t centos-python1
```

另外远程访问jupyter notebook存在一些安全限制，为了解决这些安全限制，需要做一些配置（此处约有2000字的配置）。同样，我们可以用Docker简化这个配置过程。

我们生成一个新的Dockerfile([doc/docker/code/centos-python2/Dockerfile](#))，源代码如下：

```
1 #以centos镜像为基础
2 FROM centos
3 #拷贝.jupyter目录到/root/.jupyter目录
4 RUN mkdir -p /root/.jupyter
5 COPY .jupyter /root/.jupyter
6 #安装epel (好像这一步不是必须的)
7 RUN yum -y install epel-release
8 #安装python3.8
9 RUN yum -y install python38
10 #用pip3安装numpy scipy matplotlib sympy pandas jupyter
11 RUN pip3 install -i https://mirrors.aliyun.com/pypi/simple/ numpy scipy
matplotlib sympy pandas jupyter
```

生成镜像需要在目录[code/centos-python2/](#)下面，因为这个目录中有一个`.jupyter`目录。在生成镜像时该目录会拷贝到镜像中（使用`COPY`指令）。具体怎么配置的？别管那么多，反正运行如下指令生成新的镜像即可。

```
1 docker build -t centos-python2:latest .
```

之后运行`centos-python2`镜像

```
1 docker run -i -t -p 8888:8888 centos-python2
```

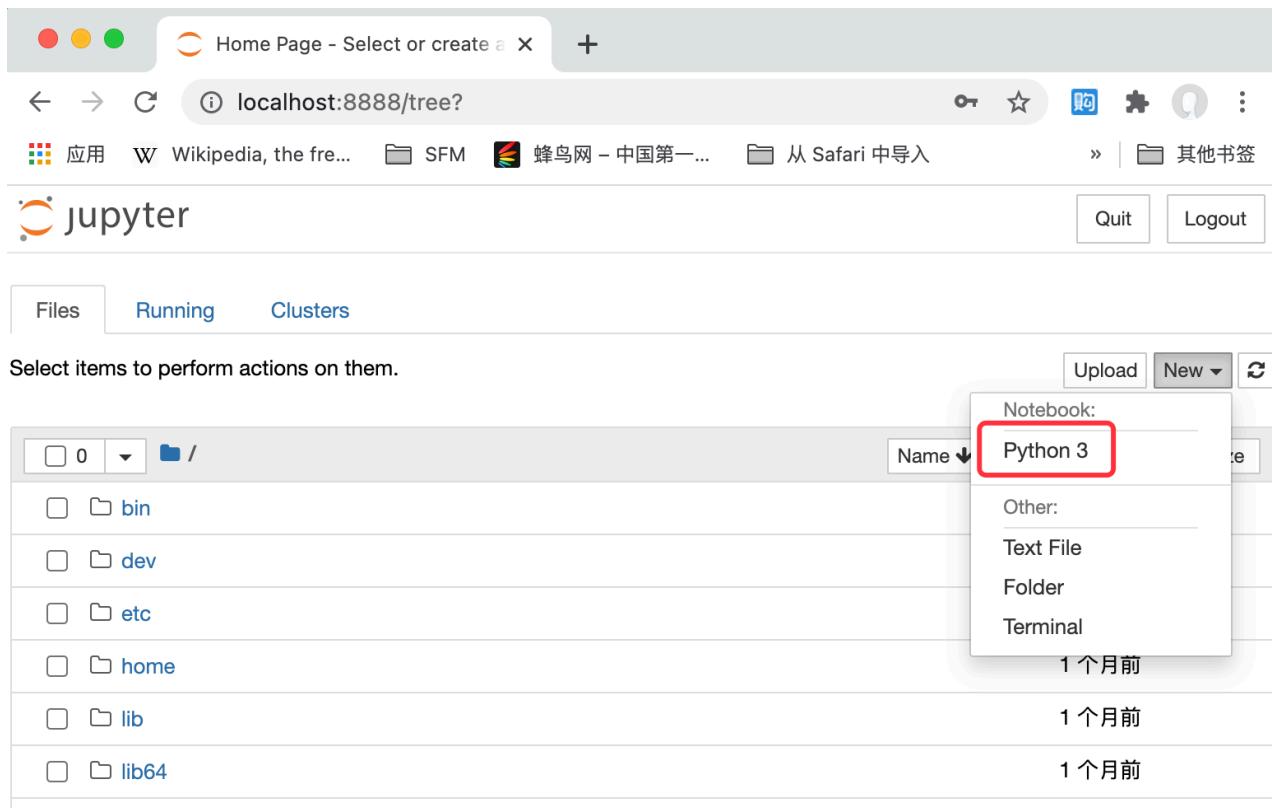
在虚拟机中执行`jupyter notebook --allow-root`

```
liuji@liuji-MacBook-Pro-2018 ~ % docker run -i -t -p 8888:8888 centos-python2
[root@e9e72d5a01f4 ~]# jupyter notebook --allow-root
[I 03:59:15.953 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret
[W 03:59:16.175 NotebookApp] WARNING: The notebook server is listening on all IP addresses and not using encryption. This is not recommended.
[I 03:59:16.177 NotebookApp] Serving notebooks from local directory: /
[I 03:59:16.178 NotebookApp] Jupyter Notebook 6.1.5 is running at:
[I 03:59:16.178 NotebookApp] http://e9e72d5a01f4:8888/
[I 03:59:16.178 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

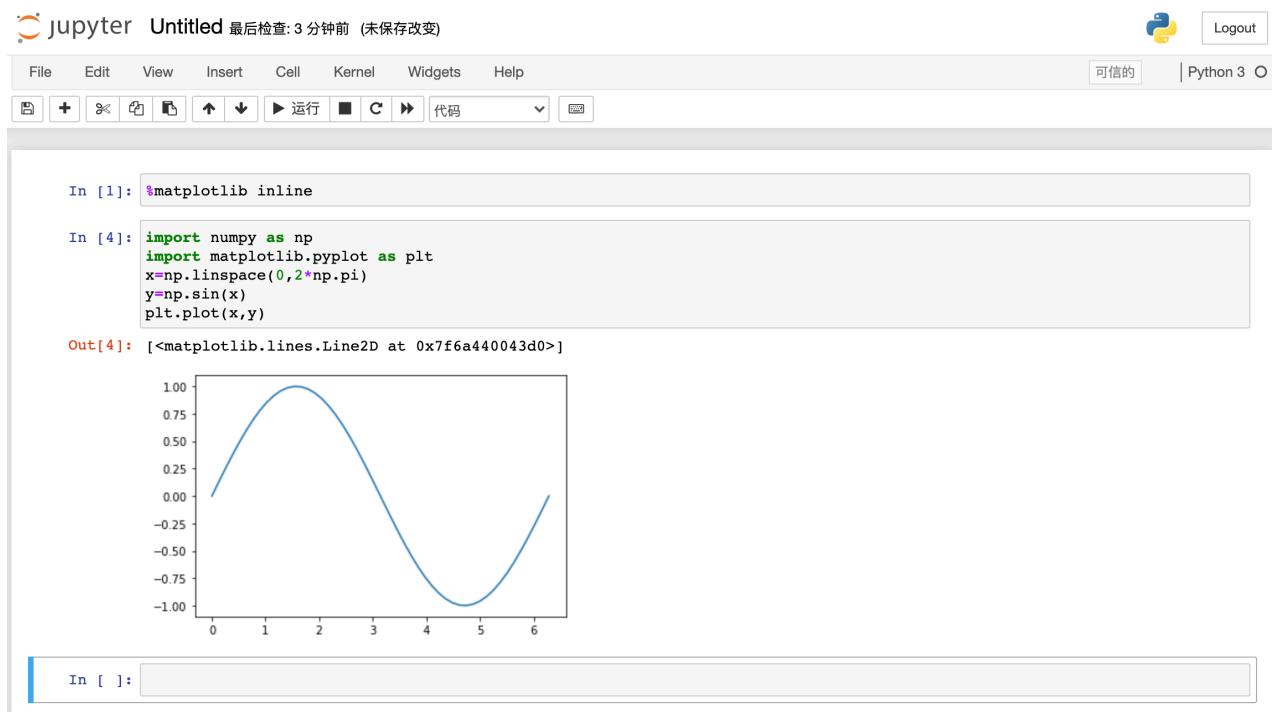
之后进入浏览器，访问<http://localhost:8888>，输入密码123，即可进入jupyter notebook。



接着创建python代码



执行一个酷一点的



如果你需要AI

给你一段Dockerfile

```
1 FROM centos
2 #拷贝.jupyter目录到/root/.jupyter目录
3 RUN mkdir -p /root/.jupyter
4 COPY .jupyter /root/.jupyter
```

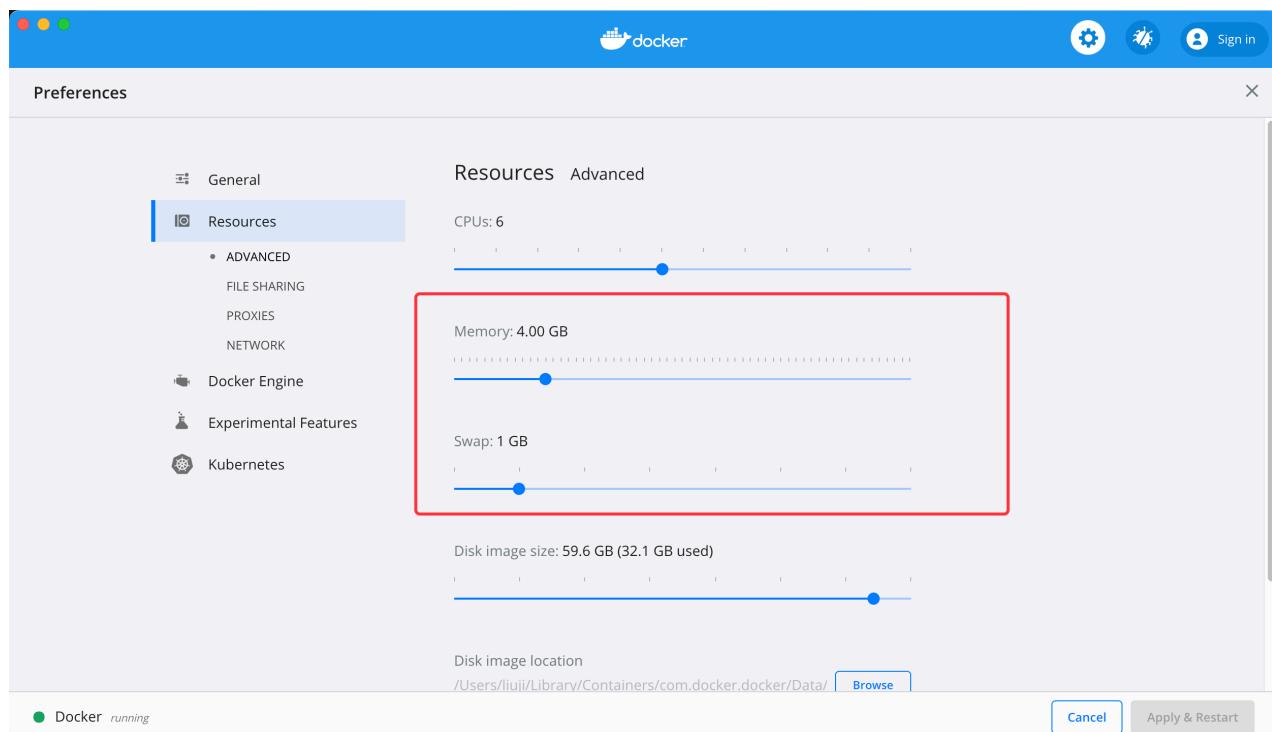
```

5 #安装epel (好像这一步不是必须的)
6 RUN yum -y install epel-release
7 #安装python3.8
8 RUN yum -y install python38
9 #用pip3安装numpy scipy matplotlib sympy pandas jupyter
10 RUN pip3 install -i https://mirrors.aliyun.com/pypi/simple/ numpy scipy
    matplotlib sympy pandas jupyter
11 #安装TensorFlow
12 RUN pip3 install -i https://mirrors.aliyun.com/pypi/simple/ tensorflow
13 #安装Pytorch
14 RUN pip3 install -i https://mirrors.aliyun.com/pypi/simple/ torch
15 RUN pip3 install -i https://mirrors.aliyun.com/pypi/simple/ torchvision

```

生成这个镜像（你并不需要理解上面的代码，虽然理解起来不难），自己试试。启动jupyter notebook，接下来就可以训练AI了。

友情提示：上述代码构建时需要的内存较大，提升docker的内存限制，以下是MacOS中的方法：



在命令行中执行，`docker stats` 指令，可以查看当前运行的容器的内存占用情况。`docker run` 的 `-m` 参数可以提高或者限制容器的内存占用。

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
878c91e074af	relaxed_euler	0.00%	1.309MiB / 2.93GiB	0.04%	766B / 0B	1.27MB / 0B	1

你需要自己定制镜像吗？

通常是不需要的，DockerHub上能找到各种镜像，课程中老师会给你提供各种镜像，百度上可以找到各种Dockerfile。有了Docker，世界真的变得很轻松。

总结

通过本文档你应该掌握如下的知识：

1. 如何pull docker镜像 (`docker pull`)
2. 如何run docker镜像 (`docker run`)
3. 如何build docker镜像 (`docker build`)
4. 如何挂载目录 (`-v`参数)
5. 如何映射端口 (`-p`参数)

请照着这个文档做一遍，有问题参考[菜鸟Docker教程](#)，或者上网百度。