前言

首先要搭建好python开发环境，Anaconda 是一个非常不错的环境管理器和包管理器。相比pip 安装第三放模块有时需要安装多个依赖包，Anaconda实在是要方便很多。例如，安装pandas模块适，pip中要依次安装NumPy, dateutil, pytz ,setuptools, 而Anaconda 中只要安装一个pandas包其他依赖包就会自动安装。 conda可以管理不同的运行环境，方便我们在不同版本的 python 间切换使用 。非常好的一项功能就是共享环境。

## 目录

[**目录**2](#_Toc58662027)

[1. Python 的安装5](#_Toc58662028)

[1.1 Windwos5](#_Toc58662029)

[1.2 Linux5](#_Toc58662030)

[1.2.1 yum更新yum源5](#_Toc58662031)

[1.2.2 安装Python5](#_Toc58662032)

[1.2.3 创建python虚拟环境6](#_Toc58662033)

[2. Anaconda 的安装和下载7](#_Toc58662034)

[2.1 Windows 安装7](#_Toc58662035)

[2.2 3.2 Linux下安装：8](#_Toc58662036)

[2.2.1 下载8](#_Toc58662037)

[2.2.2 安装8](#_Toc58662038)

[2.2.3 配置环境变**量**8](#_Toc58662039)

[3. conda安装及使用10](#_Toc58662040)

[3.1 conda 的安装11](#_Toc58662041)

[3.1.1 换国内源12](#_Toc58662042)

[3.1.2 临时使用镜像13](#_Toc58662043)

[3.1.3 版本导致的问题13](#_Toc58662044)

[3.2 Anaconda 的使用13](#_Toc58662045)

[3.2.1 Anaconda安装后开始菜单配置问题13](#_Toc58662046)

[3.2.2 管理包14](#_Toc58662047)

[3.2.3 管理环境14](#_Toc58662048)

[3.2.4 共享环境15](#_Toc58662049)

[3.2.5 jupyter notebook 的配置16](#_Toc58662050)

[3.2.6 matplotlib 中文字体配置18](#_Toc58662051)

[4. pip的安装使用18](#_Toc58662052)

[4.1 pip的安装18](#_Toc58662053)

[4.1.1 windows下18](#_Toc58662054)

[4.1.2 Linux下19](#_Toc58662055)

[4.1.3 安装包时临时配置20](#_Toc58662056)

[4.2 pip 的使用20](#_Toc58662057)

[4.2.1 安装包20](#_Toc58662058)

[4.2.2 查看所有版本20](#_Toc58662059)

[4.2.3 指定版本20](#_Toc58662060)

[4.2.4 卸载包21](#_Toc58662061)

[4.2.5 查看安装包信息21](#_Toc58662062)

[4.2.6 列出所有安装包21](#_Toc58662063)

[4.2.7 检测更新21](#_Toc58662064)

[4.2.8 更新包21](#_Toc58662065)

[4.2.9 环境导出导入21](#_Toc58662066)

[5. TensorFlow&Pytorch GPU版本安装22](#_Toc58662067)

[5.1 TesnsorFlow37](#_Toc58662068)

[5.2 Pytorch39](#_Toc58662069)

# Python 的安装

## Windwos

如果不想用Anaconda 可以到官网直接下载python

Python 官网下载地址 <https://www.python.org/downloads/>

安装时添加环境变量

安装后确保环境变量中有如下的地址

根据自己的实际路径

D:\programs\python3.8\Scripts\

D:\programs\python3.8\

## Linux

### yum更新yum源

yum update

* 安装Python 3.7.9所需的依赖否则安装后没有pip3包

yum install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel libffi-devel gcc make

* 在官网下载所需版本，这里用的是3.7.7版本

wget <https://www.python.org/ftp/python/3.7.9/Python-3.7.9.tgz>

* 国内镜像源下载

wget https://npm.taobao.org/mirrors/python/3.7.9/Python-3.7.9.tgz

### 安装Python

* 解压

tar -xvf Python-3.7.9.tgz

rm Python-3.7.9.tgz

* 配置编译

cd Python-3.7.9

* 配置编译的的路径（这里--prefix是指定编译安装的文件夹）

./configure --prefix=/usr/local/python3

* 执行该代码后，会编译安装到 /usr/local/bin/ 下，且不用添加软连接或环境变量

./configure --enable-optimizations

make && make install

* 添加软连接

ln -s /usr/local/python3/bin/python3 /usr/bin/python3

ln -s /usr/local/python3/bin/pip3 /usr/bin/pip3

* 将/usr/local/python3/bin加入PATH

vim /etc/profile

* 然后在文件末尾添加

export PATH=$PATH:/usr/local/python3/bin

* 修改完后，还需要让这个环境变量在配置信息中生效，执行命令

source /etc/profile

### 创建python虚拟环境

* 安装virtualenv

　 yum install python-virtualenv

* 创建python虚拟环境

　　virtualenv env

* 执行后，在本地会生成一个与虚拟环境同名的文件夹
* 如果你的系统里安装有不同版本的python，可以使用--python参数指定虚拟环境的python版本：

　 virtualenv --python=/usr/local/python3/bin/python3 env

* 启动虚拟环境

　 source bin/activate

* 退出虚拟环境

　 deactivate #

# Anaconda 的安装和下载

## Windows 安装

官网地址：

<https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html>

官网下载地址：

<https://www.anaconda.com/products/individual>

官网下载慢的同学可移步到清华镜像源：

<https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/>

官网默认的版本一般是稳定版本，大家可以参考官网的版本到清华镜像源下载对应的版本

安装步骤参考附件：Anaconda的下载和使用.pdf

## 3.2 Linux下安装：

### 下载

wget <https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2020.11-Linux-x86_64.sh>

具体网址根据版本自行选择

### 安装

bash Anaconda3-2020.11-Linux-x86\_64.sh

新版的安装在一路yes 最后是更改安装地址，默认是

Prefix= /root/anaconda3

可以自定义

/root/anaconda3>>>/home/anaconda3

最后一步有一个init初始化，选择yes的话可以自动conda命令到环境变量

默认是添加在.bashrc

手动激活

source ~/.bashrc

### 配置环境变**量**

root 用户下

vi /etc/profile

在文件最后加入如下语句（路径需要根据自己的安装位置更改）

PATH**=**$PATH:/root/anaconda3/bin #路径名跟自己实际情况而定

export PATH

按住shift键+:键，输入wq，保存文件并退出。最后使用如下命令刷新环境变量即可：

source /etc/profile

echo $PATH

或

vim ~/.bash\_profile

在最后一行加上

export PATH=$PATH:/home/nan/Anaconda3

激活

source ~/.bash\_profile

禁止自动进入base环境

conda config --set auto\_activate\_base false

# Miniconda的安装和下载

Miniconda 是Anaconda的简化版，只有部分基础功能用户可根据自己的需求安装所需的包。

官网下载地址：

<https://docs.conda.io/en/latest/miniconda.html>

官网下载慢的同学可移步到清华镜像源：

<https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/?C=M&O=D>



## 3.2 Linux下安装：

### 下载

wget https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/Miniconda3-py39\_4.10.3-Linux-x86\_64.sh

具体网址根据版本自行选择

### 安装

bash Miniconda3-py39\_4.10.3-Linux-x86\_64.sh

取消自动进入base环境

conda config --set auto\_activate\_base false

环境变量会自动添加到~/.bashrc

可以手动激活source ~/.bashrc或者进入bash 命令

conda env list

测试命令

### Docker 安装 miniconda

miniconda 体谅下很适合部署在docker容器中

官网参考

<https://docs.anaconda.com/anaconda/user-guide/tasks/docker/>

搜索镜像

docker search continuumio

拉取镜像

docker pull continuumio/miniconda3

以后台方式启动镜像创建容器

docker run -itd --name="anaconda3\_jupyter" -p 8888:8888 continuumio/miniconda3 /bin/bash

如果使用gpu

docker run –-gpus all -itd --name="anaconda3\_jupyter" -p 8888:8888 docker pull gpuci/miniconda-cuda:11.0-devel-centos7 /bin/bash

# Conda配置使用

## conda 的初始化

**root 权限下**

如果用户从来没有使用过conda config 命令，就不会有配置文件，当用户第一次运行 conda config命令时，将会在用户的家目录创建该文件，即一个名为.condarc的文本文件，一般表示 conda 应用程序的配置文件，在用户的家目录之下：

windows：C:\users\username\.condarc

Linux：/home/username/.condarc）

注意：condarc配置文件，是一种可选的（optional）运行期配置文件，其默认情况下是不存在的，但当用户第一次运行 conda config命令时，才会在用户的家目录创建该文件。我可以通过conda config 命令来配置该文件，也完全可以自己手动编辑也可以。

手动编辑

channels:

http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/

<http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/>

http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch/

http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/msys2/

ssl\_verify: true

### 换国内源

* 查看源：conda config --show-sources



* 添加清华源：

conda config --add channels <http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/>

conda config --add channels <http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/>

移除默认源

conda config --remove channels defaults



* 是否显示channel的来源

conda config --set show\_channel\_urls yes/no

如果是：conda config --set show\_channel\_urls yes 则配置文件中为 show\_channel\_urls: True。这表示在使用conda search package或者是conda install package的时候会显示这个包是来自于哪一个镜像源。

当然我也可以不显示，则为：conda config --set show\_channel\_urls no 则配置文件中为 show\_channel\_urls: False

* 移除源

conda config --remove-key channels

或

conda config --remove channels http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/

### 临时使用镜像

conda install -c 镜像源 包名

例如：

conda install -c <http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/> pandas

### 版本导致的问题

Anaconda3 2020.11(Python 3.8.5)之前的版本镜像源使用https 开头

<https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/>

## Anaconda 的使用

#### Anaconda安装后开始菜单配置问题

win+R 运行cmd，输入一下命令

conda update menuinst #更新菜单栏 出现 conda install -f console\_shortcut ipython ipython-notebook ipython-qtconsole launcher sp

#### 管理包

conda list #查看已安装内容

conda upgrade --all #更新所有包

conda insatll package\_name #安装包

conda search search\_term 模糊查找包 如查找numpy可输入conda search numpy

python setup.py install #进入下载好的第三方库路径下运行

conda isntall numpy=1.10 #指定所需的包版本

conda remove package\_name #卸载包

conda update package\_name #更新包

#### 管理环境

* 创建环境和包

conda create -n env\_name package\_names

simple: conda create -n py2 pandas

* 指定版本

conda create -n py2 python=2

* 克隆环境

conda create -n new\_env --clone old\_env

* 进入环境
* Windows 上

conda activate my\_env

* OSX/Linux上

source activate my\_env

* 在环境中安装包的命令与前面的一样：

conda install package\_name

* 离开环境
* Windows上

conda deactivate

* OSX/Linux上

source deactivate

#### 共享环境

* 当前命令窗口路径下输出环境中所有包的名称

conda env export>environment.yaml

#### 进入你的环境，

activate py3

* 使用以下命令更新你的环境

conda env update -f=/path/to/environment.yaml

-f表示你要导出的文件在本地的路径 所以/path/to/environment.yaml要更换成你本地的实际路径

#### 列出环境

conda env list

#### 删除指定环境

conda env remove -n env\_name

conda remove -n env\_name --all

### jupyter notebook 的安装配置

Anaconda3 base 环境自带jupyter 和 spyder， 如果安装的是Miniconda需要自行安装配置

#### conda 环境中安装jupyter noterbook

pip install jupyter notebook

#### 安装环境自动关联包

让jupyter 可以切换多个python环境

conda install nb\_conda

如果镜像源找不到最新本本移步官网

<https://anaconda.org/conda-forge/nb_conda>

conda install -c conda-forge nb\_conda

<https://anaconda.org/anaconda/nb_conda>

conda install -c anaconda nb\_conda

进入虚拟环境

conda activate python\_env

给虚拟环境安装ipykernel

pip install ipykernel

#### 进入notebook

jupyter notebook



jupyter notebook 中使用Tab键补全代码

#### 为jupyter notebook 添加目录

conda install -c conda-forge jupyter\_contrib\_nbextensions

或

pip install jupyter\_nbextensions\_configurator

方法：运行Jupyter Notebook, 在打开的Notebook界面里, 会发现多了一个Nbextensions,点击这个tab勾选Table of Contents (有的版本是toc2). 然后创建或者打开一个Jupter Notebook

### jupyter lab 的安装

jupyter lab 可实现终端控制

pip install jupyterlab

启动

Jupyter lab



### matplotlib 中文字体配置

notebook 中输入

import matplotlib

matplotlib.matplotlib\_fname()

查找配置文件路径

在font 字体文件夹中的ttf文件夹中添加SimHei.ttf文件

修改matplotlibrc文件

font.family 去掉注释#

font.sans-serif去掉注释# 添加SimHei

axes.unicode\_minus : False 改为True

删除C：\User\用户名下的matplotlib缓存

重启jupyter Notebook

# pip的安装使用

## pip的安装

### windows下

* 查看当前镜像源

pip config list

* 更换镜像源

pip config set global.index-url <https://pypi.tuna.tsinghua.edu.cn/simple>

pip config set install.trusted-host pypi.tuna.tsinghua.edu.cn

pip config set global.ssl\_verify false

有时候镜像源找不到包可以更换镜像源

pip config set global.index-url <https://pypi.douban.com/simple>

pip config set install.trusted-host pypi.douban.com

网易<https://mirrors.163.com/pypi/simple/>

阿里https://mirrors.aliyun.com/pypi/simple/

* 查看镜像源配置地址以及当前使用的源

pip config -v list

pip使用配置文件的搜索路径优先级是按照上述 list 的 “从下往上” 进行的

如果下面的最下面的pip.ini文件不存在就从倒数第二个开始

命令行默认配置的是类似C:\Users\Nan\AppData\Roaming\pip\pip.ini



* 移除镜像源

pip config unset global.index-url <https://pypi.tuna.tsinghua.edu.cn/simple>

pip config unset install.trusted-host [tuna.tsinghua.edu.cn/simple](https://pypi.tuna.tsinghua.edu.cn/simple)

pip config set global.index-url <https://pypi.org/simple>

pip config set install.trusted-host pypi.org/simple

或按照下面的方法

直接在user目录中创建一个pip目录，再新建文件pip.ini。（例如：C:\Users\WQP\pip\pip.ini）内容同上。

注意：一定要确保系统环境变量中的有pip.ini文件所在的路径

### Linux下

在用户目录之下： ~/.pip/pip.conf (没有就创建一个文件夹及文件。文件夹要加“.”，表示是隐藏文件夹)

添加如下内容：

[global]

iindex-url = https://pypi.douban.com/simple #豆瓣源，可以换成其他的源

trusted-host = pypi.douban.com #添加豆瓣源为可信主机，要不然可能报错

disable-pip-version-check = true #取消pip版本检查，排除每次都报最新的

pip timeout = 120

### 安装包时临时配置

pip install -i 原地址 包名称

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple numpy

常见的源有

豆瓣：http://pypi.douban.com/simple/

清华：<https://pypi.tuna.tsinghua.edu.cn/simple>

## pip 的使用

### 安装包

pip install package\_name

### 查看所有版本

pip install package\_name==

### 指定版本

pip install package\_name ==1.0.4

### 卸载包

pip uninstall package\_name

### 查看安装包信息

pip show package\_name

### 列出所有安装包

pip list

### 检测更新

pip list -outdated

### 更新包

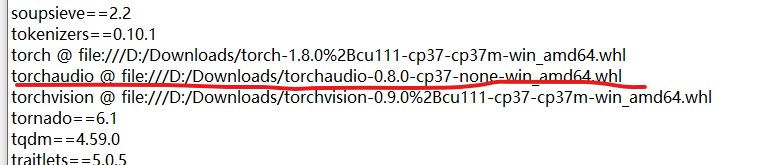
pip install --upgrade package\_name

### 环境导出导入

对于不使用conda的用户，可以使用这段命令 将一个txt文件导出并包括在其中

pip freeze >environment.txt

在 conda 环境中使用 pip freeze > requirements.txt 命令导出已安装的模块，其中部分模块显示了 @ file:///...，而不是具体的版本号如下图



当你遇到此类问题时，可以暂时考虑使用如下命令生成 requirements.txt 文件

pip list --format=freeze > requirements.txt

以管理员身份运行cmd下的命令，安装你刚导出来的environment.txt /path/environment.txt导出的文件在本地的实际路径

pip install -r /path/environment.txt

# TensorFlow&Pytorch 安装

## 显卡配置

TensorFlow和Pytorch GPU 版本安装前还要要根据自己的显卡安装对应的cuda 和CUDNN版本

CUDA是深度学习的sdk CUDNN是神经网络的sdk

另外要注意1.0版本和2.0版本支持的cuda版本也不同

NVIDIA 官网： <https://developer.nvidia.com/cuda-gpus>

NVIDIA 驱动 <https://www.nvidia.com/Download/index.aspx>

### windows

#### 驱动



**CUDA** 对应的驱动版本：[https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html](https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html)

安装好驱动后查看驱动版本



#### CUDA 安装

**CUDA Toolkit Archive**: <https://developer.nvidia.com/cuda-toolkit-archive>



**cuDNN** : <https://developer.nvidia.com/rdp/cudnn-download>

cuDNN 可能得通过科学上网不然速度实在慢的惊人



cuDNN 需要注册账号登录



解压cuda对应的cudnn



将以上的文件移到你的cuda安装目录下

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.1



添加环境变量

类似的：

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.1

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.1\bin

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.1\lib\x64

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.1\libnvvp



查看GPU状态

nvidia-smi

# 查看cuda 版本

nvcc –version

### Linux

#### 安装 NVIDIA 显卡驱动

Centos 为例 ubuntu GUI 软件商店一般就能搜到

* 查看系统内核版本

uname -r

* ·查看显卡列信息

lspci| grep -i vga

* 安装依赖

sudo yum install gcc dkms

sudo yum install kernel-devel

yum install dnf

dnf groupinstall "Development Tools"

dnf install libglvnd-devel elfutils-libelf-devel

安装完成后，执行

rpm -qa|grep gcc

rpm -qa|grep kernel

检查安装版本，这里可能遇到的情况有kernel存在两个版本，这时候要卸载一个，确保存在的kernel与kernel-devel和kernel-header包的版本号一致

卸载命令(不检查依赖关系直接删除)

例如

rpm -e --nodeps kernel-3.10.0-1160.el7.x86\_64

* 屏蔽 nouveau 驱动

vim /etc/modprobe.d/nvidia-installer-disable-nouveau.conf

* 添加

blacklist nouveau

options nouveau modeset=0

保存退出

vim /lib/modprobe.d/nvidia-installer-disable-nouveau.conf

* 添加

blacklist nouveau

options nouveau modeset=0

如果不手动添加安装过程可能会提示自动添加，然后重启生效

备份 initramfs 镜像

mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).img.bak

* 建立新的镜像

dracut /boot/initramfs-$(uname -r).img $(uname -r)

* 查看nouveau有没有被禁用

系统默认安装nouveau kernel driver, 与NVIDIA驱动冲突，所以要先检查其是否被禁用，执行命令

lsmod | grep nouveau

有输出信息说明没有被禁用，禁用方法如下

* 改为终端模式重启

systemctl set-default multi-user.target

init 3

reboot

* 安装

chmod +x ./NVIDIA-Linux-x86\_64-460.32.03.run

./NVIDIA-Linux-x86\_64-460.32.03.run -no-x-check -no-nouveau-check -no-opengl-files

–no-opengl-files 只安装驱动文件，不安装OpenGL文件。这个参数最重要

–no-x-check 安装驱动时不检查X服务

–no-nouveau-check 安装驱动时不检查nouveau

如果安装后再进入图形界面显示器不亮可尝试在图形界面中安装

* 查看GPU状态

nvidia-smi

Ubuntu

sudo apt-add-repository -r ppa:graphics-drivers/ppa

sudo apt remove nvidia\*

sudo apt autoremove

sudo ubuntu-drivers autoinstall

如果仍然对保留的软件包有问题，可能是因为在启用PPA的同时更新了一些软件包，在这种情况下，运行

sudo apt install aptitude

sudo aptitude install <name\_of\_package\_with\_conflicts>

#### cuda





安装

chmod +x sh cuda\_11.0.3\_450.51.06\_linux.run

sudo sh cuda\_11.0.3\_450.51.06\_linux.run

安装过程只选择安装cuda即可

安装完成后会有提示添加环境变量

vim ~/.bashrc

export PATH=$PATH:/usr/local/cuda-11.0/bin

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/cuda-11.0/lib64

source ~/.bashrc

**cuDNN**



解压

tar -xzvf cudnn-11.0-linux-x64-v8.0.5.39.tgz

复制到cuda 的安装目录

sudo cp cuda/include/cudnn.h /usr/local/cuda/include

sudo cp cuda/lib64/libcudnn\* /usr/local/cuda/lib64

sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn\*

如果运行tensorflow 缺少上个版本的组件（例如libcusolver.so.10），请自行下载copy到cuda-11.1/lib64 目录

卸载

在cuda的安装目录如 /usr/local/cuda-11.1/bin运行以下命令

cuda-uninstaller

**ubuntu**

* 安装

wget https://mirrors.aliyun.com/nvidia-cuda/ubuntu2004/x86\_64/cuda-ubuntu2004.pin

sudo mv cuda-ubuntu2004.pin /etc/apt/preferences.d/cuda-repository-pin-600

sudo apt-key adv --fetch-keys https://mirrors.aliyun.com/nvidia-cuda/ubuntu2004/x86\_64/7fa2af80.pub

sudo add-apt-repository "deb https://mirrors.aliyun.com/nvidia-cuda/ubuntu2004/x86\_64/ /"

sudo apt-get update

sudo apt-get -y install cuda-11-1

* 卸载

sudo apt-get remove --auto-remove nvidia-cuda-toolkit

sudo apt-get purge --auto-remove nvidia-cuda-toolkit

* Uninstall just nvidia-cuda-toolkit

sudo apt-get remove nvidia-cuda-toolkit

* Uninstall nvidia-cuda-toolkit and it's dependencies

sudo apt-get remove --auto-remove nvidia-cuda-toolkit

* Purging config/data

sudo apt-get purge nvidia-cuda-toolkit or sudo apt-get purge --auto-remove nvidia-cuda-toolkit

cuDNN 与centos类似

## Nvidia-docker 安装

### Docker 安装

Linux 中建议用docker 构建cuda 环境，这是比较快捷和有效的方法

官网地址：<https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html#docker>

设置稳定存储库yum源为阿里docker源

yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

#### 查看已安装的docker列表

yum list installed |grep docke

#### 删除已安装的docker

yum -y remove docker-ce.x86\_64

使用Docker Engineshequ社区的存储库安装docker。

在新主机上首次安装 Docker Engine-Community 之前，需要设置 Docker 仓库。之后，您可以从仓库安装和更新 Docker。

设置仓库

安装所需的软件包。

yum- utils 提供了 yum-config-manager ，并且 device mapper 存储驱动程序需要 device-mapper-persistent-data 和 lvm2。

sudo yum install -y yum-utils device-mapper-persistent-data lvm2

#### 设置稳定存储库yum源为阿里docker源

yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

#### 安装 Docker Engine-Community

* 安装最新版本的 Docker Engine-Community 和 containerd，或者转到下一步安装特定版本：

sudo yum install docker-ce docker-ce-cli containerd.io

如果提示您接受 GPG 密钥，请选是

Docker 安装完默认未启动。并且已经创建好 docker 用户组，但该用户组下没有用户。

要安装特定版本的 Docker Engine-Community，请在存储库中列出可用版本，然后选择并安装：

列出并排序您存储库中可用的版本。此示例按版本号（从高到低）对结果进行排序。

yum list docker-ce --showduplicates | sort -r

docker-ce.x86\_64 3:18.09.1-3.el7 docker-ce-stable

docker-ce.x86\_64 3:18.09.0-3.el7 docker-ce-stable

docker-ce.x86\_64 18.06.1.ce-3.el7 docker-ce-stable

docker-ce.x86\_64 18.06.0.ce-3.el7 docker-ce-stable

通过其完整的软件包名称安装特定版本，该软件包名称是软件包名称（docker-ce）加上版本字符串（第二列），从第一个冒号（:）一直到第一个连字符，并用连字符（-）分隔。例如：docker-ce-18.09.1。

验证安装是否成功(有client和service两部分表示docker安装启动都成功了)

```

docker version

```

#### 启动并加入开机启动

sudo systemctl enable docker.service

* docker启动

sudo systemctl start docker

* 重启docker

sudo systemctl restart docker

### 设置NVIDIA Container Toolkit

以下以centos7 为示例，其他版本可参照官网

#### 设置稳定的存储库和GPG密钥

distribution=$(. /etc/os-release;echo $ID$VERSION\_ID) \

&& curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.repo | sudo tee /etc/yum.repos.d/nvidia-docker.repo

#### 更新软件包清单后，nstall nvidia-docker2软件包（和依赖项）

sudo yum clean expire-cache

sudo yum install -y nvidia-docker2

#### 设置默认运行时后，重新启动Docker守护程序以完成安装：

sudo systemctl restart docker

#### 可以通过运行基本CUDA容器来测试有效的设置：

sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi

这应该导致控制台输出如下所示



## Anaconda 管理cuda环境

用conda 管理cuda是最简单快捷的， 但是往往最新版的cuda还是的自己官网下载安装

#### 进入虚拟环境

conda activate env

#### 搜索cuda版本

conda search cudatoolkit -c conda-forge

#### 选择需要的版本安装

conda install -c conda-forge cudatoolkit=11.0

#### 搜索cuDNN

conda search cuDNN -c conda-forge

#### 选择需要的版本安装

conda install -c conda-forge cuDNN=8.0

如果直接安装cuDNN 会自动安装对应的cuda 最高版本，这时可能出现不是对应的tensorflow支持版本，所以按照tensorflow对应版本先装cuda。

## TesnsorFlow

官网介绍： [**https://www.tensorflow.org/install/pip**](https://www.tensorflow.org/install/pip)



对应版本信息查看<https://tensorflow.google.cn/install/source>





安装好后可在python 编译器里查看是否可以运行

import tensorflow as tf

tf.test.is\_gpu\_available()

返回True 证明成功

## Pytorch

官网介绍：<https://pytorch.org/get-started/locally/>

根据自己的配置选择对用的版本

如果下载慢的话可以直接移步官网用迅雷下载



例如：



安装后python 编译器输入

import torch

print(torch.cuda.is\_available())

返回True 证明成功