# Anaconda使用手册

## 基本知识

Anaconda是一个开源的软件包管理系统和环境管理系统，管理python环境，用于安装多个版本的软件包及其依赖关系，并在它们之间轻松切换。

Anaconda默认是干净的，而且无法直接使用系统已安装各种库。例如：Anaconda默认不带opencv包，而且无法直接使用系统已安装的opencv，需要用户在Anaconda下再次安装，命令如下：(如果出错可以多重覆几次)

conda install --channel https://conda.anaconda.org/menpo opencv3

或者直接

conda install opencv3

除了python外，Anaconda还默认安装了下面几个工具：

1）Anaconda Navigtor：用于管理工具包和环境的图形用户界面，后续涉及的众多管理命令也可以在 Navigator 中手工实现。

2）Jupyter notebook ：基于web的交互式计算环境，可以编辑易于人们阅读的文档，用于展示数据分析的过程。

3）qtconsole ：一个可执行 IPython 的仿终端图形界面程序，相比 Python Shell 界面，qtconsole 可以直接显示代码生成的图形，实现多行代码输入执行，以及内置许多有用的功能和函数。

4）spyder ：全称为the Scientific PYthon Development EnviRonment，一个使用Python语言、跨平台的、科学运算集成开发环境，是一个免费的交互式开发环境。在anaconda安装完成之后，直接在终端输入spyder命令即可启动。

## 安装Anaconda

1）下载anaconda，网址为：https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/。该镜像站中包括了大部分anaconda的版本，在这里，我用的是“Anaconda2-2.4.1-Linux-x86\_64.sh”，希望大家统一一下使用这个。

以“Anaconda2-2.4.1-Linux-x86\_64.sh”为例，解释一下文件名的意义：Anaconda2说明是基于Python2的版本，2.4.1是具体的版本号，Linux是指适用于linux系统，x86\_64表明适用于64位系统（如果只有x86.sh，表明适用于32位系统）。

2）在linux下，直接运行该文件即可安装，安装中出现“proceed ([y]/n)?”，则选择 “y”。

3）安装成功之后需要先退出linux系统，再次登录linux系统后，命令栏输入“conda --version”，Conda会返回你安装Anaconda软件的版本。这样就安装成功了。

## 使用conda

一般流程是：先新建环境（见3.1的第2条），再激活环境（见3.1的第3条），配置并安装需要的包（遵循3.2），退出环境（见3.1的第3条）。

### 3.1 基本使用

* 删除anaconda：由于anaconda在linux下是安装在一个文件夹里/root/anaconda ,如果安装过程中出错问题，或者想更新另一个版本，删除anaconda也很方便，执行下面命令

|  |
| --- |
| rm -rf ~/anaconda |

* 建立一个 conda 环境，conda create -n <env\_name> python=2.7。

|  |
| --- |
| 例1：conda create -n ydwu-tmp python=2.7  # 新建一个名为ydwu-tmp的环境，并指定python版本为2.7，若不指定python版本则使用系统默认python版本。  例2：conda create -n <env\_name> numpy matplotlib python=2.7:  # 新建env\_name的同时安装必要的包.并不需要在新建环境时就安装包,一般都可以在新建环境之后再做。  例3：conda env create -f environment.yml  #使用yml来安装环境 |

* 激活/关闭环境，即打开/关闭一个环境（linux系统）：

|  |
| --- |
| 激活环境：source activate <env\_name>  关闭环境：source deactivate <env\_name>  若是Windows系统，则activate <env\_name> 和 deactivate <env\_name>。 |

* 查看当前系统下的所有环境。

|  |
| --- |
| conda info -e  即conda info --envs |

* 移除指定环境

|  |
| --- |
| conda remove -n <env\_name> --all  conda remove --name python27 --all，进行删除 |

* 复制一个环境

|  |
| --- |
| conda create -n <env\_name1> --clone <env\_name2>  在本地的conda里已经有一个AAA的环境，我想创建一个新环境跟它一模一样的叫BBB：  conda create -n BBB --clone AAA |

* 查看anaconda的版本

conda --v

* 通过conda -h或conda –-help，可以查看更多指令。
* 导出环境到文件

conda env export > environment.yaml

* 从文件导入环境

conda env create -f environment.yaml

### 3.2 python包管理(可以替代pip)

Anaconda默认不带opencv包，而且无法直接使用系统已安装的opencv，需要用户在Anaconda下再次安装，命令如下：(如果出错可以多重覆几次)

conda install --channel https://conda.anaconda.org/menpo opencv3

或者直接

conda install opencv3

* conda下载包很慢，增加源

|  |
| --- |
| 在以下的使用过程中你会发现使用conda下载包的速度非常的慢，因为使用的是国外的服务器，所以这里要设置为国内的镜像。使用下面的配置命令即可：  # 添加Anaconda的TUNA镜像  conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/  # TUNA的help中镜像地址加有引号，需要去掉 |
| # 设置搜索时显示通道地址  conda config --set show\_channel\_urls yes |

* 查看安装时自带的Python扩展包（package）

|  |
| --- |
| 查看当前环境的Python扩展包（package）conda list  查看指定某环境下安装的包：conda list -n <env\_name> |

* 查看可用软件包

conda search

* 安装软件包。

给某个特定环境安装包有两种方式：

|  |
| --- |
| 一是切换到该环境下直接安装，如“source activate <env\_name> && conda install <package-name>”。 |
| 二是安装时指定环境参数-n，如“conda install -n <env\_name> <package-name>”。 |

如果需要指定包的版本，需要使用 [package-name]=x.x。

|  |
| --- |
| 例如：装一下sklearn。  conda install scikit-learn  如果需要指定版本，也可以直接用 [package-name]=x.x 来指定，即  conda install scikit-learn==0.24.2  还可以指定channel，即  conda install --channel https://conda.anaconda.org/menpo opencv3 |

* 更新包

|  |
| --- |
| conda update <package-name>  注意，有时候包是建立在pip的基础上的(可以用pip list查看)，这些包的更新时通过“pip install -U setuptools” |

* 卸载包

conda remove <package-name>

## anconda的文件结构介绍

anaconda2/会在HOME目录中，具体例子就是在“/home/${user}/anaconda2"

bin, include, lib, share里面是conda默认环境(即root环境)的文件。

|  |  |
| --- | --- |
| bin | 属于root环境。  这里都是些可执行的文件。就如同使用linux命令时，调用的是/bin/目录下的文件；当使用conda命令时，实际调用的是该bin/目录下的文件。  $ which pip  >> /home/ydwu/anaconda2/bin/pip" |
| compiler\_compat |  |
| condabin |  |
| conda-meta | 属于root环境  在root环境下，所安装的所有的python库。可以通过“conda list”去查看。 |
| doc |  |
| envs | 用户(你)所创建的所有的python环境都在这。可以通过“conda info -e”去查看。  举个例子：对于环境名为bj-tf\_quan，其路径是anaconda2/envs/bj-tf\_quan/，该目录的结构如下：  ├── envs  │   ├── bin  │   ├── conda-meta  │   ├── include  │   ├── lib  │   ├── share  │   └── ssl  注意，bj-tf\_quan环境中的anaconda2/envs/bj-tf\_quan/bin相当于root环境中的anaconda2/bin，其他几个子目录也如此。 |
| etc | “etc”真正代表的是“etcetera”(附加物)。一般地，最为重要的目录是“bin”，(即“binaries”)；而“etc”中是一些次等重要的文件，包含琐碎的且广泛的系统配置文件，如启动、关机和管理。 |
| include | (属于root环境)：公共的头文件目录 |
| lib | (属于root环境)：公共基础库 |
| libexec | 这个目录下的文件及文件夹应该都可以放置在lib下；可以理解为lib的补充。 |
| LICENSE.txt | 版本许可 |
| licensing |  |
| man |  |
| mkspecs | 与QT的配置有关（QT是跨平台C++图形用户界面应用程序开发框架） |
| phrasebooks | "短语集" |
| pkgs | 里面是解压的软件安装包，可以在conda环境中进行链接。每个软件包驻留在与其规范名称相对应的子目录中。 |
| plugins | 插件 |
| pyodbc.pyi |  |
| qml | Qt有关 |
| resources |  |
| sbin | 只有超级用户才可以使用bin |
| share | 属于root环境)  可以在不同环境中共享的库或文件。举个例子：  在root环境中，which jupyter；  在bj-tf\_quan环境中，which jupyter。结果一样。 |
| shell |  |
| ssl | SSL证书  SSL(Secure Sockets Layer 安全套接层),及其继任者传输层安全（Transport Layer Security，TLS）是为网络通信提供安全及数据完整性的一种安全协议。TLS与SSL在传输层对网络连接进行加密。 |
| translations |  |
| var | .conda/中有一个environments.txt文件，里面存储着从安装conda以来，所有环境名的列表。 |

## 参考资料：

http://python.jobbole.com/86236/