https://www.jianshu.com/p/2f3be7781451

Anaconda使用总结

2016.7.19 PeterYuan

序

Python易用，但用好却不易，其中比较头疼的就是包管理和Python不同版本的问题，特别是当你使用Windows的时候。为了解决这些问题，有不少发行版的Python，比如WinPython、Anaconda等，这些发行版将python和许多常用的package打包，方便pythoners直接使用，此外，还有virtualenv、pyenv等工具管理虚拟环境。

个人尝试了很多类似的发行版，最终选择了Anaconda，因为其强大而方便的包管理与环境管理的功能。该文主要介绍下Anaconda，对Anaconda的理解，并简要总结下相关的操作。

Anaconda概述

Anaconda是一个用于科学计算的Python发行版，支持 Linux, Mac, Windows系统，提供了包管理与环境管理的功能，可以很方便地解决多版本python并存、切换以及各种第三方包安装问题。Anaconda利用工具/命令conda来进行package和environment的管理，并且已经包含了Python和相关的配套工具。

这里先解释下conda、anaconda这些概念的差别。conda可以理解为一个工具，也是一个可执行命令，其核心功能是包管理与环境管理。包管理与pip的使用类似，环境管理则允许用户方便地安装不同版本的python并可以快速切换。Anaconda则是一个打包的集合，里面预装好了conda、某个版本的python、众多packages、科学计算工具等等，所以也称为Python的一种发行版。其实还有Miniconda，顾名思义，它只包含最基本的内容——python与conda，以及相关的必须依赖项，对于空间要求严格的用户，Miniconda是一种选择。

进入下文之前，说明一下conda的设计理念——conda将几乎所有的工具、第三方包都当做package对待，甚至包括python和conda自身！因此，conda打破了包管理与环境管理的约束，能非常方便地安装各种版本python、各种package并方便地切换。

Anaconda的安装

**Anaconda的下载页参见官网下载，Linux、Mac、Windows均支持。**

安装时，会发现有两个不同版本的Anaconda，分别对应Python 2.7和Python 3.5，两个版本其实除了这点区别外其他都一样。后面我们会看到，安装哪个版本并不本质，因为通过环境管理，我们可以很方便地切换运行时的Python版本。（由于我常用的Python是2.7和3.4，因此倾向于直接安装Python 2.7对应的Anaconda）

下载后直接按照说明安装即可。这里想提醒一点：尽量按照Anaconda默认的行为安装——不使用root权限，仅为个人安装，安装目录设置在个人主目录下（Windows就无所谓了）。这样的好处是，同一台机器上的不同用户完全可以安装、配置自己的Anaconda，不会互相影响。

对于Mac、Linux系统，Anaconda安装好后，实际上就是在主目录下多了个文件夹（~/anaconda）而已，Windows会写入注册表。安装时，安装程序会把bin目录加入PATH（Linux/Mac写入~/.bashrc，Windows添加到系统变量PATH），这些操作也完全可以自己完成。以Linux/Mac为例，安装完成后设置PATH的操作是

**# 将anaconda的bin目录加入PATH，根据版本不同，也可能是~/anaconda3/bin**

**echo 'export PATH="~/anaconda2/bin:$PATH"' >> ~/.bashrc**

**# 更新bashrc以立即生效**

**source ~/.bashrc**

配置好PATH后，可以通过which conda或conda --version命令检查是否正确。假如安装的是Python 2.7对应的版本，运行python --version或python -V可以得到Python 2.7.12 :: Anaconda 4.1.1 (64-bit)，也说明该发行版默认的环境是Python 2.7。

Conda的环境管理

Conda的环境管理功能允许我们同时安装若干不同版本的Python，并能自由切换。对于上述安装过程，假设我们采用的是Python 2.7对应的安装包，那么Python 2.7就是默认的环境（默认名字是root，注意这个root不是超级管理员的意思）。

假设我们需要安装Python 3.4，此时，我们需要做的操作如下：

# 创建一个名为python34的环境，指定Python版本是3.4（不用管是3.4.x，conda会为我们自动寻找3.4.x中的最新版本）

conda create --name python34 python=3.4

# 安装好后，使用activate激活某个环境

activate python34 # for Windows

source activate python34 # for Linux & Mac

# 激活后，会发现terminal输入的地方多了python34的字样，实际上，此时系统做的事情就是把默认2.7环境从PATH中去除，再把3.4对应的命令加入PATH

# 此时，再次输入

python --version

# 可以得到`Python 3.4.5 :: Anaconda 4.1.1 (64-bit)`，即系统已经切换到了3.4的环境

# 如果想返回默认的python 2.7环境，运行

deactivate python34 # for Windows

source deactivate python34 # for Linux & Mac

# 删除一个已有的环境

conda remove --name python34 --all

用户安装的不同python环境都会被放在目录~/anaconda/envs下，可以在命令中运行conda info -e查看已安装的环境，当前被激活的环境会显示有一个星号或者括号。

说明：有些用户可能经常使用python 3.4环境，因此直接把~/anaconda/envs/python34下面的bin或者Scripts加入PATH，去除anaconda对应的那个bin目录。这个办法，怎么说呢，也是可以的，但总觉得不是那么elegant……

如果直接按上面说的这么改PATH，你会发现conda命令又找不到了（当然找不到啦，因为conda在~/anaconda/bin里呢），这时候怎么办呢？方法有二：1. 显式地给出conda的绝对地址 2. 在python34环境中也安装conda工具（推荐）。

Conda的包管理

Conda的包管理就比较好理解了，这部分功能与pip类似。

例如，如果需要安装scipy：

# 安装scipy

conda install scipy

# conda会从从远程搜索scipy的相关信息和依赖项目，对于python 3.4，conda会同时安装numpy和mkl（运算加速的库）

# 查看已经安装的packages

conda list

# 最新版的conda是从site-packages文件夹中搜索已经安装的包，不依赖于pip，因此可以显示出通过各种方式安装的包

conda的一些常用操作如下：

# 查看当前环境下已安装的包

conda list

# 查看某个指定环境的已安装包

conda list -n python34

# 查找package信息

conda search numpy

# 安装package

conda install -n python34 numpy

# 如果不用-n指定环境名称，则被安装在当前活跃环境

# 也可以通过-c指定通过某个channel安装

# 更新package

conda update -n python34 numpy

# 删除package

conda remove -n python34 numpy

前面已经提到，conda将conda、python等都视为package，因此，完全可以使用conda来管理conda和python的版本，例如

# 更新conda，保持conda最新

conda update conda

# 更新anaconda

conda update anaconda

# 更新python

conda update python

# 假设当前环境是python 3.4, conda会将python升级为3.4.x系列的当前最新版本

补充：如果创建新的python环境，比如3.4，运行conda create -n python34 python=3.4之后，conda仅安装python 3.4相关的必须项，如python, pip等，如果希望该环境像默认环境那样，安装anaconda集合包，只需要：

# 在当前环境下安装anaconda包集合

conda install anaconda

# 结合创建环境的命令，以上操作可以合并为

conda create -n python34 python=3.4 anaconda

# 也可以不用全部安装，根据需求安装自己需要的package即可

设置国内镜像

如果需要安装很多packages，你会发现conda下载的速度经常很慢，因为Anaconda.org的服务器在国外。所幸的是，清华TUNA镜像源有Anaconda仓库的镜像，我们将其加入conda的配置即可：

# 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

执行完上述命令后，会生成~/.condarc(Linux/Mac)或C:\Users\USER\_NAME\.condarc文件，记录着我们对conda的配置，直接手动创建、编辑该文件是相同的效果。

跋

Anaconda具有跨平台、包管理、环境管理的特点，因此很适合快速在新的机器上部署Python环境。总结而言，整套安装、配置流程如下：

* 下载Anaconda、安装
* 配置PATH（bashrc或环境变量），更改TUNA镜像源
* 创建所需的不用版本的python环境
* Just Try!

cheat-sheet 下载：  
Conda cheat sheet

参考资料

* Anaconda Homepage
* Anaconda Documentation
* Conda Docs

作者：PeterYuan  
链接：https://www.jianshu.com/p/2f3be7781451  
來源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。