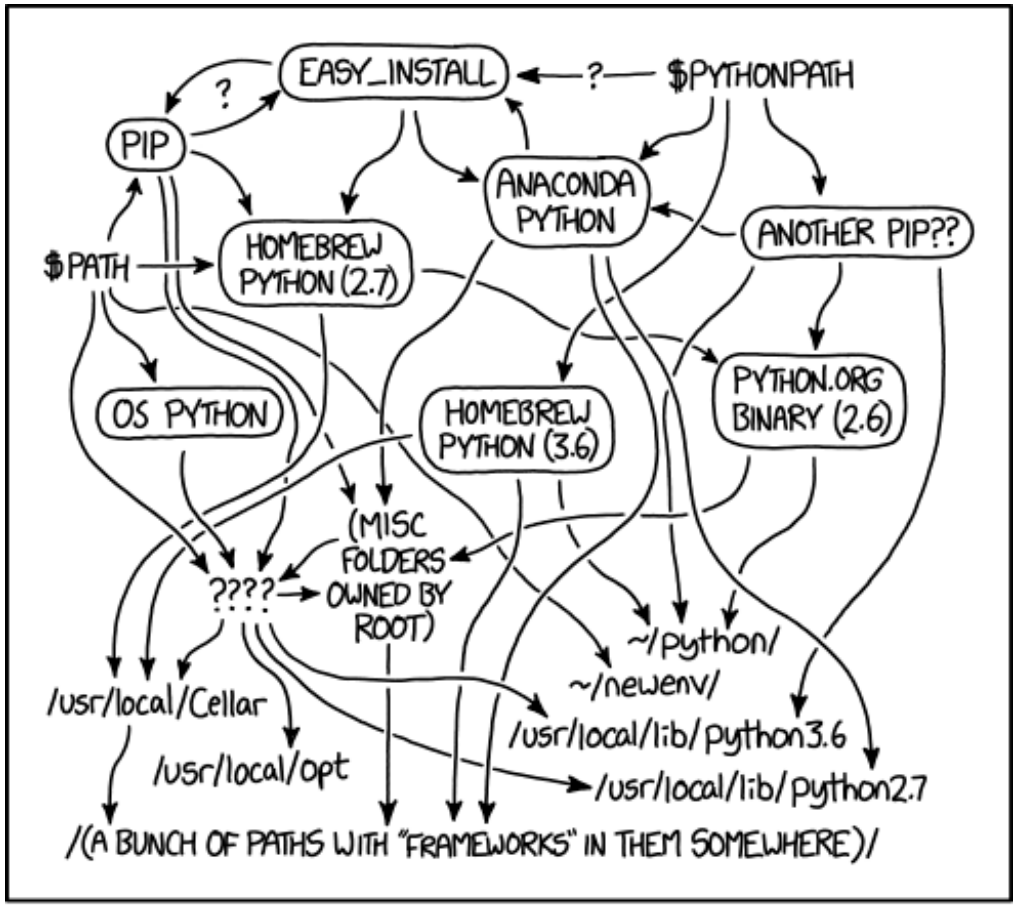
Customizing Python Environment on TSCC

Manu Shantharam
San Diego Supercomputer Center
September 2, 2021

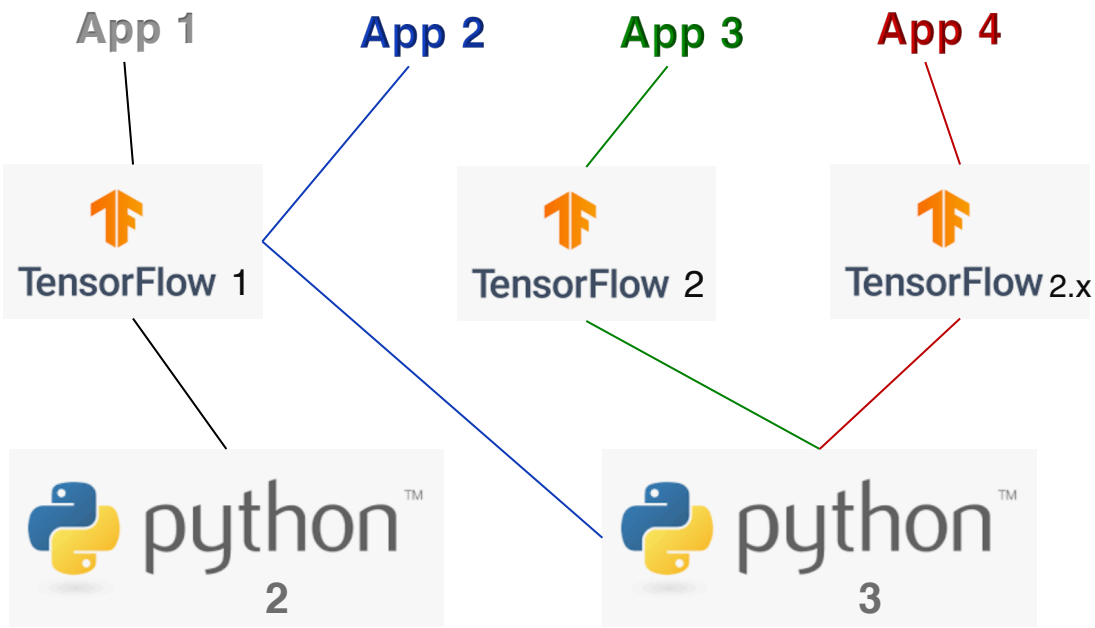
How some view the Python ecosystem



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987/

Seriously, why customize Python environment?



Some Tools to Manage Python environment

Miniconda

Anaconda

conda-pack

A tool for packaging and distributing conda environments.

Watch

build passing

Some Tools to Manage Python environment

Miniconda

Anaconda

Why Anaconda

- Users want us to provide a stable, yet flexible Python environment
- From a support point of view, this can be challenging
  - Users will want different Python versions (some still need Python 2!)
  - The number of packages and modules seems almost infinite
  - Some users will require access to old versions of modules. This seems to be common among grad students nearing the end of their thesis projects who don't want subtle changes in results from bioinformatics pipelines.
  - Some Python packages can have dependencies on systems libraries
- Using Anaconda, users can easily manage their own Python installations, with the added bonus that they can maintain consistency from laptop to cluster

Anaconda installation

Installing Anaconda on the cluster is easy and doesn't require root access or any special knowledge.

- Go to https://www.anaconda.com/products/individual
- Follow link to Download page and select Linux
  https://www.anaconda.com/products/individual#Downloads
- Right click download button to copy link location then use wget to install in your home directory on TSCC
  wget https://repo.anaconda.com/archive/Anaconda3-2021.05-Linux-x86_64.sh

Anaconda installation

Download to cluster (TSCC) is fast and takes about 5 seconds

```
[manu1729@tscc-login12 tscc-examples]$ wget
https://repo.anaconda.com/archive/Anaconda3-2021.05-Linux-x86_64.sh

--2021-09-02 08:45:20--  https://repo.anaconda.com/archive/Anaconda3-2021.05-
Linux-x86_64.sh

Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3,
104.16.131.3, 2606:4700::6810:8203, ...

Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443...
connected.

HTTP request sent, awaiting response... 200 OK

Length: 570853747 (544M) [application/x-sh]

Saving to: 'Anaconda3-2021.05-Linux-x86_64.sh'

100%[=============================================================
=============>] 570,853,747  173MB/s   in 3.2s

2021-09-02 08:45:25 (168 MB/s) - 'Anaconda3-2021.05-Linux-x86_64.sh' saved
[570853747/570853747]
```

Anaconda installation

Full anaconda installation in home directory takes just under six minutes and requires 3.4 GB of disk space

```
$ time bash Anaconda3-5.1.0-Linux-x86_64.sh
Welcome to Anaconda3 5.1.0
                …
Thank you for installing Anaconda3!
real 5m50.775s
user 1m39.212s
sys 0m42.248s

$ du -hs anaconda3/
3.4G anaconda3/
```

```
Thank you for installing Anaconda3!

real     64m13.925s
user     2m22.940s
sys      3m15.737s

[manu1729@tscc-login12 tscc-examples]$ du -hs anaconda3/
2.6G     anaconda3/
```

Anaconda installation

The Anaconda installer can automatically modify the .bashrc to point to the newly installed Python location. Beware of possible conflicts due to existing PYTHONPATH.

```
WARNING:
    You currently have a PYTHONPATH environment variable set. This may cause
    unexpected behavior when running the Python interpreter in Anaconda3.
    For best results, please verify that your PYTHONPATH only points to
    directories of packages that are compatible with the Python interpreter
    in Anaconda3: /home/manu1729/tscc-examples/anaconda3
Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]
[no] >>>
You have chosen to not have conda modify your shell scripts at all.
To activate conda's base environment in your current shell session:

eval "$(/home/manu1729/tscc-examples/anaconda3/bin/conda shell.YOUR_SHELL_NAME hook)"
To install conda's shell functions for easier access, first activate, then:
conda init

If you'd prefer that conda's base environment not be activated on startup,
   set the auto_activate_base parameter to false:

conda config --set auto_activate_base false

Thank you for installing Anaconda3!
```

Installing packages

Although Anaconda comes with a large number of standard packages, users may still need to install additional packages. This can be done using conda

```
[manu1729@tscc-login12 tscc-examples]$ source anaconda3/bin/activate
(base) [manu1729@tscc-login12 tscc-examples]$ which python
~/tscc-examples/anaconda3/bin/python
```

```
(base) [manu1729@tscc-login12 tscc-examples]$ python3
Python 3.8.8 (default, Apr 13 2021, 19:58:26)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from Bio import SeqIO
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'Bio'

$ conda install BioPython
(base) [manu1729@tscc-login12 tscc-examples]$ python3
Python 3.8.8 (default, Apr 13 2021, 19:58:26)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from Bio import SeqIO
>>>
```

Installing packages

But beware that sometimes special installations may be needed
https://stackoverflow.com/questions/33433274/anaconda-graphviz-cant-import-after-installation

```
$ python3
$ conda install graphviz
Solving environment: done
    graphviz:        2.40.1-h25d223c_0
Downloading and Extracting Packages
graphviz

$ python3
>>> import graphviz
Traceback (most recent call last):
ModuleNotFoundError: No module named 'graphviz'

$ pip install graphviz
Installing collected packages: graphviz
Successfully installed graphviz-0.8.3

$ python3
>>> import graphviz
>>> graphviz.__version__
'0.8.3'
```

outdated

Miniconda vs. Anaconda

Anaconda contains all of the most common packages (tools) a data scientist needs and can be considered the hardware store of data science tools. Miniconda is more like a workbench, you can customise it with the tools you want. Conda is the assistant underlying Anaconda and Miniconda. It helps you order new tools and organise them when you need.

Ref: https://towardsdatascience.com/get-your-computer-ready-for-machine-learning-how-what-and-why-you-should-use-anaconda-miniconda-d213444f36d6

Miniconda vs. Anaconda

- Miniconda includes only conda and its dependencies. Anaconda includes conda plus over 7,000 widely used packages (over 250 packages are installed by default).
- Installing Miniconda is quick (~ one minute) and requires little space (348 MB)
- Which one is right for you? It depends. According to https://stackoverflow.com/questions/45421163/anaconda-vs-miniconda

Choose Anaconda if you:

- Are new to conda or Python
- Like the convenience of having Python and over 150 scientific packages automatically installed at once
- Have the time and disk space (a few minutes and 3 GB), and/or
- Don't want to install each of the packages you want to use individually.

Choose Miniconda if you:

- Do not mind installing each of the packages you want to use individually.
- Do not have time or disk space to install over 150 packages at once, and/or
- Just want fast access to Python and the conda commands, and wish to sort out the other programs later.

I use Miniconda myself. Anaconda is bloated. Many of the packages are never used and could still be easily installed if and when needed.

Listing installed packages

Installed packages and their versions can be listed using “conda list”. Of course, you can still get this information from the Python prompt using the appropriate dunder (e.g. numpy.__version__)

```
(base) [manu1729@tscc-login12 tscc-examples]$ conda list
# packages in environment at /home/manu1729/tscc-
examples/anaconda3:
#
# Name                    Version                   Build  Channel
_ipyw_jlab_nb_ext_conf    0.1.0                    py38_0
_libgcc_mutex             0.1                        main
alabaster                 0.7.12                 pyhd3eb1b0_0
anaconda-project          0.9.1                  pyhd3eb1b0_1
anyio                     2.2.0                py38h06a4308_1
appdirs                   1.4.4                      py_0
argh                      0.26.2
.
.
.
(base) [manu1729@tscc-login12 tscc-examples]$ conda list | wc -l
348
```

Working with environments

Environments give you the flexibility to use different versions of Python and/or Python packages without altering your default environment (i.e. the one created when installing Anaconda or Miniconda). Before creating new environments, you may need to update conda to the latest version

```
(base) [manu1729@tscc-login12 tscc-examples]$ conda update -n base
conda
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.
```

Creating environments

Environments are created using the conda create command. A few examples are shown below for creating environments based on different Python versions and/or packages

```
# Environment with different Python version (3.7)
$ conda create -n py37 python=3.7

# Environment with older yaml version
$ conda create -n yaml24 yaml=0.2.4 -c conda-forge
```

Listing and removing environments

Environments are listed and removed with the conda info and remove commands.

```
$ (base) [manu1729@tscc-login12 tscc-examples]$ conda info --envs
# conda environments:
base                  *  /home/manu1729/tscc-examples/anaconda3
py37                     /home/manu1729/tscc-examples/anaconda3/envs/py37
yaml24                   /home/manu1729/tscc-examples/anaconda3/envs/yaml24
/anaconda3/envs/py34

$ conda remove -n <env_name> --all
```

Activating/deactivating environments

Use source activate and source deactivate to change environments. Note that the prompt changes to reflect the new environment and that repeated calls to activate create a stack of environments.

```
[manu1729@tscc-login12 ~]$ source tscc-examples/anaconda3/bin/activate
(base) [manu1729@tscc-login12 ~]$ python -V
Python 3.8.8
(base) [manu1729@tscc-login12 ~]$ conda activate py37
(py37) [manu1729@tscc-login12 ~]$ python -V
Python 3.7.11
(py37) [manu1729@tscc-login12 ~]$ conda deactivate
(base) [manu1729@tscc-login12 ~]$ python -V
Python 3.8.8
```

Some Tools to Manage Python environment

conda-pack

A tool for packaging and distributing conda environments.

Watch

build passing

Miniconda

Anaconda

Python pip

- pip is an inbuilt installer program to install python modules (starting Python 3.4)
- Installing a module using pip
  - python3 -m pip install <some_package>
    installs the library at the system level
  - python3 -m pip install --user <some_package>
    installs the library for the user locally
- Some useful pip commands
  - pip3 show <package> :
  - pip3 list
  - pip3 install pipreqs (Generate requirements.txt file for any project based on imports)
  - pipreqs . (from the source directory)

Python venv

- I don't want to use conda, what should I do?
- Python packages can be installed at different locations in your system.
  - system packages are typically stored in the path stored in sys.prefix

```
>>> import sys
>>> sys.prefix
'/usr'
```

- third party packages installed using pip are typically stored in one of the paths pointed to by site.getsitepackages

```
>>> import site
>>> site.getsitepackages()
['/usr/local/lib64/python3.6/site-packages', '/usr/local/lib/python3.6/site-packages', '/usr/lib64/python3.6/site-packages', '/usr/lib/python3.6/site-packages']
```

Python venv

- Python virtual environment creates an isolated environment for Python projects
  - each project can have its own dependencies
  - no site packages are included by default
  - the path located in sys.prefix is then used for locating the site-packages directory by searching the relative path
  - using pip within a virtual environment installs the packages within the environment (no need for --user flag)
- Some useful commands to create a virtual environment

```
# Python 3
$ python3 -m venv env

# Python 2
pip install virtualenv  (< python3, python3 + has venv)
$ virtualenv env
```

Some Tools to Manage Python environment

python pip

python CONDA

Miniconda

Anaconda

conda-pack

A tool for packaging and distributing conda environments.

Watch

build passing

python venv

Conda-Pack

- conda install / activate too slow (use of shared file systems / login nodes / …)
- conda environments are not relocatable
  - moving an environment to a different directory can render it partially or completely inoperable
- command line tool for creating archives of conda environments that can be installed on other systems and locations
- Useful commands

  conda install conda-pack
  conda install -c conda-forge conda-pack

- Example

home / login node

```
Thank you for installing Anaconda3!

real    64m13.925s
user    2m22.940s
sys     3m15.737s
```

scratch / compute node

```
Thank you for installing Anaconda3!

real    1m35.836s
user    1m9.883s
sys     0m13.447s
```

Thank You!

Managing R with conda

Although Anaconda and Jupyter are usually associated with the Python community, keep in mind that conda is language agnostic.

```
$ which R
/opt/R/bin/R

$ conda install -c r r-irkernel
Solving environment: done
## Package Plan ##
  environment location: /home/etrain109/anaconda3

$ which R
/home/etrain109/anaconda3/bin/R
```

Managing R with conda

R notebooks executed in Jupyter will use the R version found in Anaconda and packages will be installed in anaconda3/lib/R/library. Personally, I find this more convenient than dealing with a mix of system wide and local installations.