Installing Packages Using Conda

Conda is a **package** manager, which helps you find and install packages such as numpy or scipy. It also serves as an **environment** manager, and allows you to have multiple isolated environments for different projects on a single machine. Each environment has its own installation directory, and does not share packages with other environments.

For example, you may need python 2.7 and Biopython 1.60 in project A, while you also work on another project B, which needs python 3.5 and Biopython 1.68. You can use conda to create a separate environment for each project. You can then switch between different versions of packages easily to run your project code.

Installing Python packages without Conda

pip, which stands for Pip Install Packages, is Python's official package manager. If you do not need to manage multiple environments for different projects you can install python packages through pip. You can find the list of available packages from Python Package Index (PyPI) https://pypi.python.org/pypi.

You already have pip, if you are using Python 2 >= 2.7.9 or Python 3 >= 3.4. Otherwise you need to install pip, following the instructions at https://packaging.python.org/installing/#requirements-for-installing-packages). In a terminal, you enter the following to install a package.

```
pip search <package>
pip install --user <package>
```

The `--user` option here tells pip to install the package inside your home directory, rather than the system python library location. On BioHPC you do not have permission to add packages to the system location, so you must use the `--user` option.

Why conda?

Even if you have simple requirements, you may wish to use conda to manage packages instead of pip. The two greatest advantages of conda are:

1) Comprehensive Dependency Management
Often, installing a package is not as straight forward as you first think. Imagine a case like this: You want to install package A. When installing, it asks you to install B, and C, because A needs B and C to work. B and C are called the **dependencies** of A. In turn B and C might have dependencies D and E, which may require even more package to run. Installing a single package quickly becomes a complex task that requires many installations.

Conda provide a solution for this situation: when you install package A, it will automatically install all the dependencies of A. So you don't have to install them one by one, manually. This save you great amount of time.

Pip has similar functionality, but more limited. It is only able to install python dependencies – if a package depends on another piece of software that is not python based then pip will not install this dependency. Conda can manage dependencies of any type.

2) Multiple environments for different projects. As mentioned at the beginning of this guide, it's becoming common to have different package requirements for different projects. Also, as reproducible computational science becomes a bigger concern, having a defined environment for a project – which contains exact versions of all tools – can make replicating results or extending long-running projects more manageable.

Conda vs Anaconda

On the web, and on BioHPC, you will see a lot of documentation referring to both conda and Anaconda. Anaconda is a complete distribution of python and many common packages, created by Continuum Analytics. Conda is the package management tool which was developed for Anaconda. You can use conda without Anaconda, but using Anaconda always involves the conda tool.

Using conda environments on BioHPC

1. Load the anaconda python module on BioHPC

To access the conda tool you need to load one of our anaconda python modules, either version 2.7.x or 3.4.x:

```
module load python/2.7.x-anaconda
Or
module load python/3.4.x-anaconda
```

Python 3 is the latest version of the language and python 2 is considered legacy. Generally you should choose Python 3 for new projects whenever possible. However, if you know some packages you want are not compatible with Python 3, then you will use Python 2.7.x.

See the link below for a discussion of Python 2 vs Python 3

https://wiki.python.org/moin/Python2orPython3#Should_I_use_Python_2_or_Python_3_for_my_development_activity.3F

2. Create new conda environment

When you load a python/anaconda module on BioHPC you have access to the standard Anaconda distribution

Anaconda is a collection of scientific and data analytic Python packages, it has over 200 packages for python 2.7.x-anaconda including NumPy, Pandas, SciPy, Matplotlib, and iPython.

Create a conda environment named test1 with latest anaconda package.

```
conda create -n test1 anaconda
```

Usually, the conda environment is installed in your home directory on BioHPC, /home2/<your username>. The newly created environment <test1> will be installed in the directory /home2/<your username>/.conda/envs/. You can see all the packages under folder "bin".

```
[s173217@Nucleus005 bin]$ pwd
/home2/s173217/.conda/envs/test1/bin
[s173217@Nucleus005 bin]$ ls
2to3                              glib-genmarshal          python
activate                          glib-gettextize          python2
assistant                         glib-mkenums             python2.7
bokeh                             gobject-query            python-config
cairo-trace                       gresource                pyuic5
cjpeg                             gsettings                qcollectiongenerator
conda                             gst-device-monitor-1.0   qdbus
c_rehash                          gst-discoverer-1.0       qdbuscpp2xml
dbus-cleanup-sockets              gst-inspect-1.0          qdbusviewer
dbus-daemon                       gst-launch-1.0           qdbusxml2cpp
dbus-launch                       gst-play-1.0             qdoc
dbus-monitor                      gst-stats-1.0            qhelpconverter
dbus-run-session                  gst-typefind-1.0         qhelpgenerator
```

3. Use the environment you just created

```
source activate test1
```

It will show your environment name at the beginning of the prompt.

```
(test1) [s173217@Nucleus005 ~]$ conda list
```

4. Install packages in the conda environment

Install using conda

You can search if your package is in the default source from Anaconda collection. Besides the 200 pre-built Anaconda packages, it contains over 600 extra scientific and analytic packages. All the dependencies will also be installed automatically.

```
conda search <package>
conda install <package>
```

```
(test1) [s173217@Nucleus005 ~]$ conda install matplotlib
Fetching package metadata .............
Solving package specifications: .

Package plan for installation in environment /home2/s173217/.conda/envs/test1:

The following NEW packages will be INSTALLED:

    functools32:  3.2.3.2-py27_0    bioconda
    matplotlib:   2.0.0-np111py27_0
    subprocess32: 3.2.7-py27_0

Proceed ([y]/n)? y

subprocess32-3 100% |################################| Time: 0:00:00  18.03 MB/s
matplotlib-2.0 100% |################################| Time: 0:00:00  45.85 MB/s
```

Install using conda-forge (example: emacs)

Conda **channels** are the remote repository that conda takes to search or download the packages. If you want to install a package that is not in the default Anaconda channel, you can tell conda which channel containing the package, so that conda can find and install.

Conda-forge is a GitHub community-led conda channel, containing general packages which are not in the default Anaconda source. All the packages from conda-forge is listed at https://conda-forge.github.io/feedstocks.

```
conda install -c conda-forge emacs
```

Install using Bioconda (example: stringtie)

Bioconda is another channel of conda, focusing on bioinformatics software. Instead of adding "`-c`" to search a channel only one time, "`add channels`" tells Conda to always search in this channel, so you don't need to specify the channel every time. Remember to add channel in this order, so that Bioconda channel has the highest priority. Channel orders will be explained in next part.

```
conda config --add channels conda-forge
conda config --add channels defaults
conda config --add channels r
conda config --add channels bioconda
```

Adding channels will not generate any command line output.
Then, you can install Stringtie from the Bioconda channel

```
conda install stringtie
```

All the bioconda packages can be found here: https://bioconda.github.io/recipes.html

Channel order

If you add multiple channels in one environment using (`conda config --add channels <new_channel>`), The latest or most recent added one have the highest priority. That means, if there is a same package in different channels, the package version from highest priority channel will overwrite other versions, to either higher or lower.

For example, if you add the channels in different order in the Stringtie example, by switching channel r and channel bioconda. Say, channel R has package A version 0.8 and bioconda has A version 1.0. The environment will have A 0.8 now from channel R, since it's the highest priority. Then, Stringtie might not work if it need package A 1.0.

If the packages can be found in different channels, the package from the highest priority channel will be installed even if the version of it isn't newest. The command <conda config --append channels new_channel> puts the new channel at the bottom of the channel list, making it the lowest priority.

Install R and R packages

The Conda package manager is not limited to Python. R and R packages are well supported by a conda channel maintained by the developers of Conda. The R-essentials include all the popular R packages with all of their dependencies. The command below opens R channel by "-c r", and then install the r-essentials using R channel.

```
module load python/2.7.x-anaconda
conda install -c r r-essentials
```

```
(test1) [s173217@Nucleus005 bin]$ conda install -c r r-essentials
Fetching package metadata ...............
Solving package specifications: .

Package plan for installation in environment /home2/s173217/.conda/envs/test1:

The following NEW packages will be INSTALLED:

    bzip2:          1.0.6-3
    configparser:   3.5.0-py27_0
    curl:           7.45.0-2          bioconda
    entrypoints:    0.2.2-py27_0
    gsl:            2.2.1-0
    harfbuzz:       0.9.39-2
    ipykernel:      4.5.2-py27_0
    jbig:           2.1-0
    jsonschema:     2.5.1-py27_0
    jupyter_client: 4.4.0-py27_0
    jupyter_core:   4.2.1-py27_0
    libsodium:      1.0.10-0
    libtiff:        4.0.6-3
    mistune:        0.7.3-py27_0
    nbconvert:      4.2.0-py27_0
    nbformat:       4.2.0-py27_0
```

Update R packages

```
conda update -c r r-essentials
conda update -c r r-<package name>
```

Install from online source (Example: Theano from GitHub)

If some packages are not in any conda channels, you can still use pip to install them in the conda environment. Here's an example: Install Theano from Github (https://github.com/Theano/Theano).

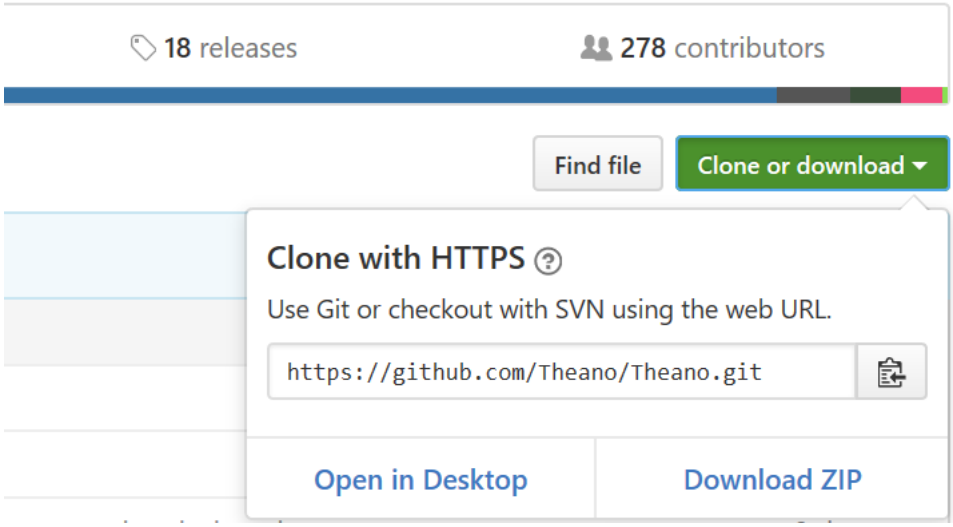
We need to add git module first and install the package.

```
module load git/v2.5.3
```

You can find the package URL from top right of the Theano git website, by clicking the green button "Clone or downloaded".

```
#pip install git+<package web URL>

pip install git+https://github.com/Theano/Theano.git
```



More conda commands:

1. See all available environments

You can check the list of all separate environments, and it will show * at your current environment. In the figure below, it shows root, since I'm not in any conda environment.

```
conda env list
```

```
[s173217@Nucleus005 ~]$ conda env list
# conda environments:
#
epic                     /home2/s173217/.conda/envs/epic
fusion                   /home2/s173217/.conda/envs/fusion
test1                    /home2/s173217/.conda/envs/test1
root                  *  /cm/shared/apps/python/2.7.x-anaconda
```

2. Exit current environment:

You can exit, when you finish your work in the current environment.

```
source deactivate
```

3. List all package installed

This will show all the packages and versions you've installed.

```
conda list
```

```
(test1) [s173217@Nucleus005 ~]$ conda list
# packages in environment at /home2/s173217/.conda/envs/test1:
#
backports                 1.0                      py27_0
backports_abc             0.5                      py27_0
bokeh                     0.12.4                   py27_0
cairo                     1.14.8                        0
cycler                    0.10.0                   py27_0
dbus                      1.10.10                       0
decorator                 4.0.11                   py27_0
enum34                    1.1.6                    py27_0
expat                     2.1.0                         0
fontconfig                2.12.1                        1
freetype                  2.5.5                         2
futures                   3.0.5                    py27_0
get_terminal_size         1.0.0                    py27_0
glib                      2.50.2                        1
gst-plugins-base          1.8.0                         0
```

4. Update packages or conda itself

This will update to the newest version of one package, or conda itself.

```
#update package
conda update <package>
#update conda self
conda update conda
```

5. Uninstall package from the environment

```
conda uninstall <package name>
```

```
(test1) [s173217@Nucleus005 ~]$ conda uninstall matplotlib
Fetching package metadata .............
Solving package specifications: .

Package plan for package removal in environment /home2/s173217/.conda/envs/test1:

The following packages will be REMOVED:

    matplotlib: 1.5.3-np111py27_1

Proceed ([y]/n)?
```

6. Remove environment

```
conda env remove --name test2
```

When you finish your project, you might want to remove the environment. However, it is not recommended because you might want to update some work in this project in the future.

Reference

Conda documentation http://conda.pydata.org/docs/get-started.html

Conda-forge https://conda-forge.github.io/

BioConda https://bioconda.github.io/