Anaconda Python

This page describes the configuration of Anaconda Python on the Debian Grid nodes.

Contents

Overview

Use of (anaconda) python in batch scripts

Use of (anaconda) python in interactive sessions

How about Anaconda2 ?!

User Installation of Python modules

Using "pip"

Private environments (Experienced users only)

Overview

We're adopting a simplified structure of application versions that retains only two repositories: one for Python vers3 and one for Python vers2 intuitively named:

- /apps/anaconda3
- /apps/anaconda2

This approach seems not only simpler but more reflective of the content, as stamping a version label on the folder is accurate only until one of the contained packages gets updated. As we frequently add&update various packages, the folder's "version" indicates only the Anaconda distribution version that *started* that repo -- completely irrelevant information which is mistaken for a true "release" tag...

The two repos will be updated as frequently as reasonable in order to provide the newest set of packages part of the Anaconda distribution. In addition they'll contain the sum of all other packages that have been added in response to user requests.

Note:

The frequent consequence of these package additions, let alone the programatic updates, is the updating of various dependent packages -- therefore the repositories are constantly changing. If these changes are affecting negatively the functioning of your code the only option that freezes the individual application environment is the use of *private virtual environments*. The repositories above, as shared resources, cannot be maintained constant in spite of the false image created by the version stamped anaconda repositories deployed on the previous RHEL6 nodes.

Use of (anaconda) python in batch scripts

The previous recommendation remains in effect.

To prepare a python script that employs the interpreter found in one of these repositories, as well as the packages/modules therein (!!), use as first line of your script the string:

```
#!/apps/anaconda3/bin/python
```

or, for version 2:

```
#!/apps/anaconda2/bin/python
```

Submit the script using "sge_run" with the proper options and arguments (as before).

Use of (anaconda) python in interactive sessions

This mode of operation requires the understanding of "Python Environments" and the use of "conda". You may refresh your knowledge with public sources and reading the sections below (to be written).

Upon opening an interactive session to a Debian node (by running "sge_run" on the head node), the system automatically made available the "conda" command from the "Anaconda3" repository. However, no environment was activated. Consequently the system python is default, but we don't want to use it !!! (none of the packages we need are available in that installation)

```
[rp2927@researchgrid ~]$ sge_run
Warning: Permanently added '[research10.gsb.columbia.edu]:35955,[128.59.199.150]:35955' (ECDSA) to the list of
known hosts.
Linux research10 4.19.0-4-amd64 #1 SMP Debian 4.19.28-2 (2019-03-15) x86_64

-------  CBS Research Support -- researchsupport@gsb.columbia.edu  -------

(live)rp2927@research10:~$
(live)rp2927@research10:~$ which conda
/apps/anaconda3/condabin/conda                   <<<==== Conda from Anaconda3 is available
(live)rp2927@research10:~$
(live)rp2927@research10:~$ conda info

     active environment : None                   <<<==== No Active Environment
            shell level : 0
       user config file : /user/user1/rp2927/.condarc
 populated config files : /user/user1/rp2927/.condarc
          conda version : 4.6.14
    conda-build version : 3.17.8
         python version : 3.7.3.final.0
       base environment : /apps/anaconda3  (read only)
           channel URLs : https://repo.anaconda.com/pkgs/main/linux-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/free/linux-64
                          https://repo.anaconda.com/pkgs/free/noarch
                          https://repo.anaconda.com/pkgs/r/linux-64
                          https://repo.anaconda.com/pkgs/r/noarch
          package cache : /apps/anaconda3/pkgs
                          /user/user1/rp2927/.conda/pkgs
       envs directories : /user/user1/rp2927/.conda/envs
                          /apps/anaconda3/envs
               platform : linux-64
             user-agent : conda/4.6.14 requests/2.21.0 CPython/3.7.3 Linux/4.19.0-4-amd64 debian/testing
glibc/2.28
                UID:GID : 578103723:578103723
             netrc file : None
           offline mode : False

(live)rp2927@research10:~$ which python
/usr/bin/python         <<<==== As there is no Active Environment, the system python is default.   WE DON'T WANT
TO RUN THAT PYTHON !!!!!
(live)rp2927@research10:~$
```

To activate the "base" environment (the interpreter and packages installed in /apps/anaconda3) run: "**conda activate**". Observe the change of the prompt, and more importantly the new default python interpreter (the one from /apps/anaconda3, which is the one that has access to the data

science packages from the Anaconda3 install)

```
(live)rp2927@research10:~$ conda activate
(base) (live)rp2927@research10:~$ conda info

     active environment : base         <<<==== The Active Environment has changed, it's /apps/anaconda3...
    active env location : /apps/anaconda3
            shell level : 1
       user config file : /user/user1/rp2927/.condarc
 populated config files : /user/user1/rp2927/.condarc
          conda version : 4.6.14
    conda-build version : 3.17.8
         python version : 3.7.3.final.0
       base environment : /apps/anaconda3  (read only)   <<<==== Observe that the Environment /apps/anaconda3 is
NOT USER WRITABLE.  Only system administrators can write in there.
           channel URLs : https://repo.anaconda.com/pkgs/main/linux-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/free/linux-64
                          https://repo.anaconda.com/pkgs/free/noarch
                          https://repo.anaconda.com/pkgs/r/linux-64
                          https://repo.anaconda.com/pkgs/r/noarch
          package cache : /apps/anaconda3/pkgs
                          /user/user1/rp2927/.conda/pkgs
       envs directories : /user/user1/rp2927/.conda/envs
                          /apps/anaconda3/envs
               platform : linux-64
             user-agent : conda/4.6.14 requests/2.21.0 CPython/3.7.3 Linux/4.19.0-4-amd64 debian/testing
glibc/2.28
                UID:GID : 578103723:578103723
             netrc file : None
           offline mode : False

(base) (live)rp2927@research10:~$ which python
/apps/anaconda3/bin/python         <<<==== The python interpreter from Anaconda's installation is now the default.
That's the one we want!!
(base) (live)rp2927@research10:~$
```

Observe that **/apps/anaconda3 is not user writable** -- attempting to install a package now, with the "base" environment active, will result in failure with an error code indicating the lack of write privileges. Only system administrators can add packages to the shared /apps/anaconda3. **Users need to create their own "private" environment first!!**

How about Anaconda2 ?!

If conda from Anaconda3 is default how do I activate the /apps/anaconda2 base environment?!

Well, if you have **legacy code** that requires a python2.x interpreter and the associated packages, activate the Anaconda2 base environment by **sourcing** this file:

```
source  /apps/bin/activate_anaconda2
```

Try running "conda info" now to see the differences.

"What if I want to **go back to the Anaconda3**?..." Easy! Source the equivalent file:

```
source  /apps/bin/activate_anaconda3
```

User Installation of Python modules

Using "pip"

At startup, pip is not in your PATH -- get it in by activating an Anaconda environment:

```
conda activate
```

Make a "user installation" of the desired package/module:

```
pip install <packageName>
```

Private environments (Experienced users only)

There are many web pages with friendly instructions on how to manage virtual environments, and I strongly recommend searching terms like "conda create environment" or similar, but here I'll link the authoritative source from conda's own docs: https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html It has the advantage of always being up to date.

Let's run through an example based on the instructions above.

```
conda create -n demo python
conda info
conda env list
conda activate demo
which python
conda list -n demo
conda deactivate
conda env remove -n demo
```

Using pip...

```
conda activate <myEnv>
pip install ...
```

[to be continued...]