TraP with Conda

Conda is a tool which lies somewhere between a Virtual Environment and a Virtual Machine. It has advantages over the Python virtualenv – particularly that you can specify which Python you wish to use. These instructions will help you to set up and use TraP with Conda. Once setup, you just need to run step 4 and then you can skip to step 17.

1. Download miniconda to your home area using:
   ```
   $ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
   ```

2. Install miniconda using:
   ```
   $ bash Miniconda3-latest-Linux-x86_64.sh
   ```
   Say yes to all of the questions during install.
   Then log out of the system and then log in again to confirm changes.

3. Create a new conda environment called "TraP_env" in python 3 using:
   ```
   $ conda create --name TraP_env python=3
   ```
   You can specify whatever version of Python you wish. For TraP releases including r5.0 and lower, you need Python 2.7. For TraP releases r6.0 and higher, you want the latest Python 3.

4. Activate your Conda environment using:
   ```
   $ conda activate TraP_env
   ```

5. Clone the TraP repository into your home area, using:
   ```
   $ git clone https://github.com/transientskp/tkp.git
   ```

6. Move into the tkp folder:
   ```
   $ cd tkp
   ```

7. Fetch all the branches:
   ```
   $ git fetch –all
   ```

8. Check out the TraP release required. The current stable version in Python 2 is r6.0.
   ```
   $ git checkout r6.0
   ```

9. Use conda install to install all the requirements (see setup.py).
   ```
   $ conda install astropy colorlog Jinja2 numpy psycopg2 python-dateutil python-casacore pytz scipy sqlalchemy<2 alembic monotonic psutil pytest dask dask[array]
   ```

10. n.b. the curremt version of python-casacore sometimes does not install correctly as it misses dependencies. Run this and then try the above command again.
    ```
    $ conda install libcBlas
    ```

11. Install TraP using:
```
$ pip install -e ".[pixelstore]"
```

12. Install PostgreSQL following the instructions for your system, https://www.postgresql.org/download/ You will probably be asked if you want the database server to start automatically when it boots. Otherwise, you can start it with
```
$ pg_ctl start -l logfile
```

13. Now you can create a new database for your TraP results.
```
$ createdb -h localhost -p 5432 -U $USER TraP_db
```

14. Start up a new TraP project folder. You will run all your TraP jobs from this folder.
```
$ trap-manage.py initproject TraP_jobs
```

15. Move into the TraP_jobs folder and edit the pipeline.cfg file that you find there. Specifically, you need to edit these rows:
```
[database]
engine = postgresql
host = localhost
username = $USER
password = $USER_PASSWORD
port = 5432
```

16. Setup your new TraP database with the TraP Schema
```
$ trap-manage.py initdb
```

Running TraP

17. Start up a new TraP job inside your TraP_jobs folder.
```
$ trap-manage.py initjob test_job
```

18. Edit the files in the new job folder. Specifically, images_to_process.py needs to be edited to point to your image folder and the job_params.cfg needs to have the settings you want to use for TraP

19. Run TraP using:
```
$ trap-manage.py run test_job
```

Filtering TraP Results

20. The easiest way to get started with searching for transients and variables in the TraP database is by using a Jupyter Notebook. An example Jupyter Notebook, with the key filtering steps developed over the past years is available here on GitHub: https://github.com/AntoniaR/TraP_filter_demo

21. Install Jupyter Notebook in your conda environment:

```
$ conda install jupyter notebook
```

22. Start your Jupyter Notebook kernel by typing:

```
$ jupyter notebook
```

23. Then use Jupyter Notebooks as usual. The website is https://jupyter.org for more information and useful documentation. Also, you can install new python modules using the typical `conda install` command above.