Catalogs and MAF

Note: If you have trouble during installation, community.lsst.org is a Stack Overflow-like forum for all things LSST. We are doing our best to answer all of the questions posted there as helpfully as possible. You might consider searching for your bug or, if you have discovered a new one, creating a post there so that others can benefit from your experience.

Stack requirements:

There are some general LSST software stack requirements before installation: please check that your system has the pre-requisites installed.

There are some additional requirements for the simulations packages, mostly python dependencies that are so ubiquitous in scientific computing environments that it is more difficult for users for us to provide new versions than it is to supply them with the stack. These requirements are (there is some overlap here with the basic stack requirements):

- python 3.6
- cmake - available from conda if required
- gfortran (which is necessary to build the ephemeris-generating software package oorb – this is ONLY needed if you are installing sims_movingObjects in addition to lsst_sims)
    - On linux systems, gfortran is probably preinstalled. Otherwise, you can get gfortran binaries from https://gcc.gnu.org/wiki/GFortranBinariesMacOS or (on a mac) from the homebrew distribution (brew install gcc).

Installing from Source

The recommended (simplest) approach is to use the newest LSST software stack, and use the LSST-provided version miniconda python. (See here if you want to use your own python).

1. Start by installing the necessary parts of the LSST software stack and the LSST-installed anaconda.

In these instructions we assume you are installing in ~/lsst, however the install directory can be any place in the file system, including a place visible to all users. To install in another location, replace ~/lsst with the desired path in the following instructions. For multi user stacks, permissions are typically restricted to read only for the main stack so packages are not accidentally installed in the main stack.

For the installation of the basic LSST packages, please check for additional requirements or known problems here: https://pipelines.lsst.io/install/newinstall.html and then follow a similar process to start the installation of the LSST build system.

Note: In the past we have warned you against running newinstall.sh with the `-t` flag, which enables installation of the DM binary builds. That is no longer a concern. `lsst_sims` will build against DM binaries. Feel free to run newinstall.sh with `-t` enabled.

```
cd ~/lsst
curl -OL https://raw.githubusercontent.com/lsst/lsst/master/scripts/newinstall.sh
bash newinstall.sh -ct
```

You do not need to do "eups distrib install lsst_distrib", as described on the pipelines page, as we will install the required **simulations** packages below.

Note that the https://pipelines.lsst.io/install/newinstall.html pages go on to describe LSST Data Management packages (lsst_apps and lsst_distrib) that include additional software not necessary for simulations (although it may be very useful for you!). This includes the LSST data management demo package. You may find it interesting for image processing and catalog purposes, but it is not directly related to installation and use of the simulations packages.

2. Set up the environment and install the simulations code and data.

Source the appropriate shell script and use eups (see here for more info) to install the software and data.

```
source ~/lsst/loadLSST.csh
eups distrib install lsst_sims -t sims_weekly_tag
curl -sSL https://raw.githubusercontent.com/lsst/shebangtron/master/shebangtron | python
```

This will install all packages currently in the catalogs simulations framework (CatSim) and metrics analysis framework (MAF) and all dependencies. The installation should take on the order of 1 hour, with a final required installation size of 10GB. Note that sims_weekly_tag will change depending on which weekly you want to install. Sims weekly tags are of the form sims_w_YYYY_WW. There is no good place to find what weekly has been most recently published, unfortunately. If you look at the release tab on the lsst/afw repository you should see what the latest release is. If, for example, the latest release is w.2019.14, then sims_weekly_tag will be sims_w_2019_14.

The `shebangtron` command is there to correct the paths to installed binaries of Data Management software packages. Every time you install new binary packages, you will need to run the `shebangtron` command.

Any of the individual packages and all their dependencies can be installed by replacing lsst_sims with the appropriate package name in the above code snippet (e.g. sims_maf). Installation is now complete. See package specific pages for documentation.

4. Setup installed packages

You have now downloaded and built all of the packages in lsst_sims, with the command "eups distrib install <packagename> -t <tag>". These packages are designed to be totally self-contained. They have been built in the directories

```
$LSST_HOME/yourOperatingSystem/yourPackageName/
```

where yourOperatingSystem is 'DarwinX86' for Mac users and 'Linux64' for Linux users. In order to use the packages you installed, you must 'activate' them in your environment, which is essentially adding these directories to your $PYTHONPATH. This is done using

```
setup your_package_name -t your_package_tag
```

"setup <packagename> -t <tag>" is an eups command that activates the packages and their dependencies. your_package_name is the name of one of the packages you have installed (e.g. sims_maf or sims_catUtils). your_package_tag is a tag eups uses to keep track of the versions of each package you have built on your machine. This corresponds to the argument of '-t' in the 'eups distrib install' command above. So, if you wanted to setup the version of MAF you just downloaded, you would use

```
setup sims_maf -t sims_weekly_tag
```

When you setup a package, eups inspects it and determines what other packages it depends on. Eups will also setup those prerequisite packages, preferring versions tagged with the tag you specified and defaulting to versions with the tag 'current'. If you do not want to default to 'current', you can specify more than one tag.

```
setup sims_maf -t $USER -t sims_weekly_tag
```

will setup sims_maf and all of its dependencies, and then using versions matching the tags (resolved from left to right). This system of eups tags allows you to have multiple versions of the stack built on your system simultaneously. You will only ever be using the one that eups has setup. To see which versions of a package exist on your system (and which has been setup) use

```
eups list -v your_package_name
```

or

```
eups list -s
```

Known issues:

- If you have issues with installation, first check that your system meets the minimum requirements listed here: https://pipelines.lsst.io/install/newinstall.html#prerequisites. Note that you need cmake – if installation of 'mariadb' fails, you probably do not have cmake installed ('conda install cmake' is an easy way to get it).
- Check for other known issues here: https://pipelines.lsst.io/known-issues.html#installation-issues
- You can also search for similar problems on https://community.lsst.org/
- If you are having issues specifically with pyephem or healpy on a Mac, check for the existence of a /Developer directory. This directory is obsolete after upgrading to newer versions of XCode, but not removed by the XCode installer. Rename the /Developer directory and pyephem will install.
- If you are using your own python, be sure to check the Using Your Own Python page. In particular, on Linux, some lsst_apps packages will currently fail to build if the "nomkl" package is not installed in anaconda.
- On a Mac, make sure you have accepted the terms on XCode. You can do this by opening the Xcode.app (should be in your Applications folder).
- git can fail, complaining about not having an https helper. If your native git version is > 1.7, you can probably use that rather than the LSST installed git.
- If all else fails, it's usually an issue with some environment variables interfering with the installation. You can create a new user and install the stack there. You can quickly login/out of a new user account as follows: First make a new admin-level user in System Preferences->Users and Groups, and then click on your name in the top right hand corner of the screen. A drop-down menu should appear, offering you a choice of other users to log in as. You might have to toggle the check box in System Preferences->Users and Groups->Login Options first though.

Mixing Installed Stack with Development Repositories

If you are going beyond simply using software packages provided by LSST and need a direct git clone of a particular repository (because you wish to contribute development work directly back into repository or because a new feature is available but has not been released officially yet), you can mix an installed stack with development repositories. For pure python packages, this is straightforward. The following steps will put a local copy of the sims_maf git repository into a pre-existing stack. **Note you do not have to do this just to install and use any sims package, such as MAF.

All packages may be declared with a version and a tag. The version distinguishes on instance of a particular package from another. The eups tag can be used to define a coherent set of packages. The philosophy is to tag all personal package (packages downloaded via git) with a custom tag using the username. Using this workflow allows packages to be set up with the -t $USER switch which means that custom packages are used if they are declared and main stack packages are used otherwise.

1. Move to a directory to hold the working repository and clone it:

```
mkdir ~/lsstRepos; cd ~/lsstRepos
# if you are using ssh
git clone git@github.com:lsst/sims_maf.git
# or if you are using https
git clone https://github.com/lsst/sims_maf.git
```

Note that you will need a password on the stash server (or have set up ssh keys) to push to the server.

2. Declare and build the package:

```
cd sims_maf
eups declare -r . -t $USER
setup sims_maf -t $USER
scons
```

You can check running the following that your dev version is being used by the following method.

```
> eups list sims_maf
...
tag: username  username setup
```

See here for the confluence question dealing with how to be polite in a shared stack.

Note that when you are trying to simply use your git cloned version of the package, rather than install it, you simply need to 'setup sims_maf -t $USER'.

3. Code, commit and push

Code reviews should be handled by branching the repository and issuing a pull request through github.