Publishing a new version of lsst_sims

DEPRECATED!!! We no longer support or publish lsst_sims as a package. Please look at rubin_sim (https://github.com/lsst/rubin_sim) instead.

Historical content only

Broadly, the steps involved in publishing a new release of lsst_sims are

1. tag all of the sims git repositories with a version number formatted like X.Y.Z.sims where X, Y, Z are integers
2. use the LSST continuous integration system to build a version of the sims stack using the tagged sims repositories as well as a tagged DM weekly
3. use the LSST continuous integration system to publish the build from step (2) above

Tag the sims git repositories

Add an annotated tag to each of the sims repositories by cloning the repositories, navigating to the desired commit, and running

```
git tag -a X.Y.Z.sims
```

git will then ask you to add a descriptive message to go with the tag. Usually, this is where we keep track of all of the changes being included in this release.

Note: because it is possible that the first tag you issue will not build, it is safest to first tag the repositories X.Y.Z.sims.rc1 (for "release candidate 1"). Once you have a combination of tags that build successfully, you can go back and apply the X.Y.Z.sims tag. This is cumbersome, but it is the safest way to ensure a sensible release.

The repositories that need to be tagged for a sims release are

- sims_utils
- sims_photUtils
- sims_catalogs
- sims_coordUtils
- sims_catUtils
- sims_GalSimInterface
- sims_maf
- sims_skybrightness
- sims_skybrightness_pre
- sims_alertsim (this is a part of the lsst-sims group on git)
- sims_survey_fields
- sims_movingObjects
- lsst_sims

Now, select the DM weekly tag against which you want to build. This will look something like w.2017.26. I usually just check list of available releases on the afw repository to see which DM weeklies are available.

Use the LSST continuous integration system to build the tagged repositories

1. Go to the run-rebuild job
2. Click the 'Run' button with the blue arrow
3. In the first field enter all of the git tags you want to build. If you enter more than one branch, the build system will resolve them such that the leftmost tag has the highest priority. Thus, if I were trying to build 2.4.7.sims against the DM weekly w.2017.28, I would enter `2.4.7.sims w.2017.28` into the first field. Any repository that has a 2.4.7.sims tag would use that tag. Any repository that does not will use w.2017.28. Any repository that has neither will use master. This can be extremely helpful if you end up having to amend your release, since you can enter something like `2.4.7.sims.rc2 2.4.7.sims.rc1 w.2017.28` in the event that some repositories need a 2.4.7.sims.rc2 tag to fix bugs discovered when building 2.4.7.sims.rc1.
4. In the second field, enter the software products that you want to build. The products that need to be built for a sims release are (in no particular order) `lsst_apps lsst_sims sims_movingObjects sims_operations`. sims_movingObjects and sims_operations are not official components of lsst_sims. We build them, though, to make sure that it is possible for users to install versions of these products that are built against the same version of the DM stack as lsst_sims.
5. Check `Skip_Demo` and `Skip_Docs`
6. Click `Run`

This will start a `run_rebuild` job which you should be able to see at the top of the page (the list of green and red dots with numbers next to them). You will need to refresh your browser to see your build. When the dot next to your build stops flashing, the build is done. If the steady dot is green, the build succeeded. If the steady dot is red, the build failed. You can click on the number next to the dot to see the stdout report of why the build failed.

Note: This build will probably fail due to Doxygen with a message like

```
eups list: Unable to find product datarel tagged "b3130"
*** Failed to find datarel "b3130" version.
*** FAILURE: Doxygen document was not installed.
```

that doesn't actually matter. Look for a message like

```
# BUILD b3130 completed.
The DM stack has been installed at /home/jenkins-slave/snowflake/release/lsstsw with tag: b3130.
```

in the stdout. That will mean that build succeeded (or, at least, that it succeeded enough for our purposes).

Use the LSST continuous integration system to publish the build

Once your `run-rebuild` job succeeds, go back and tag the repositories with a final X.Y.Z.sims tag (no .rc1, .rc2, etc.) and run `run_rebuild` again (sorry). After that job completes, navigate to the `Console Output` for that job. There should be a `BUILD ID` of the form bWXYZ associated with the build (W, X, Y, Z are all integers). Make a note of that `BUILD ID`

1. Go to the run publish section of the continuous integration system
2. Click the 'Run' button with the blue arrow on it
3. Set `package` (as opposed to `git`); this will control where the build is stored
4. Enter the `BUILD ID` from `run rebuild` into the first field (this is the bNNNN number)
5. Enter the eups tag you want applied to the release (probably `sims`) into the second field
6. Enter the list of products from `run rebuild` into the third field
7. Click `Run`

This will start a `run publish` job. When that job is finished, you should be able to `eups distrib install your_product -t your_tag` and get the product and tag that you entered.

Note: `run publish` should be fast, however, `run rebuild` and `run publish` share the same (single) Jenkins node, so if your `run publish` job is unfortunate enough to get stuck behind someone else's `run rebuild` job, you could be waiting for a while.

Note: traditionally, we run `run publish` twice, once with the tag `sims` and once with the tag `sims_X_Y_Z` with X, Y, Z taken from the versioned release. That way, `eups distrib install lsst_sims -t sims` always gets the most up-to-date sims stack, while `eups distrib install lsst_sims -t sims_X_Y_Z` allows users to install a historical version.