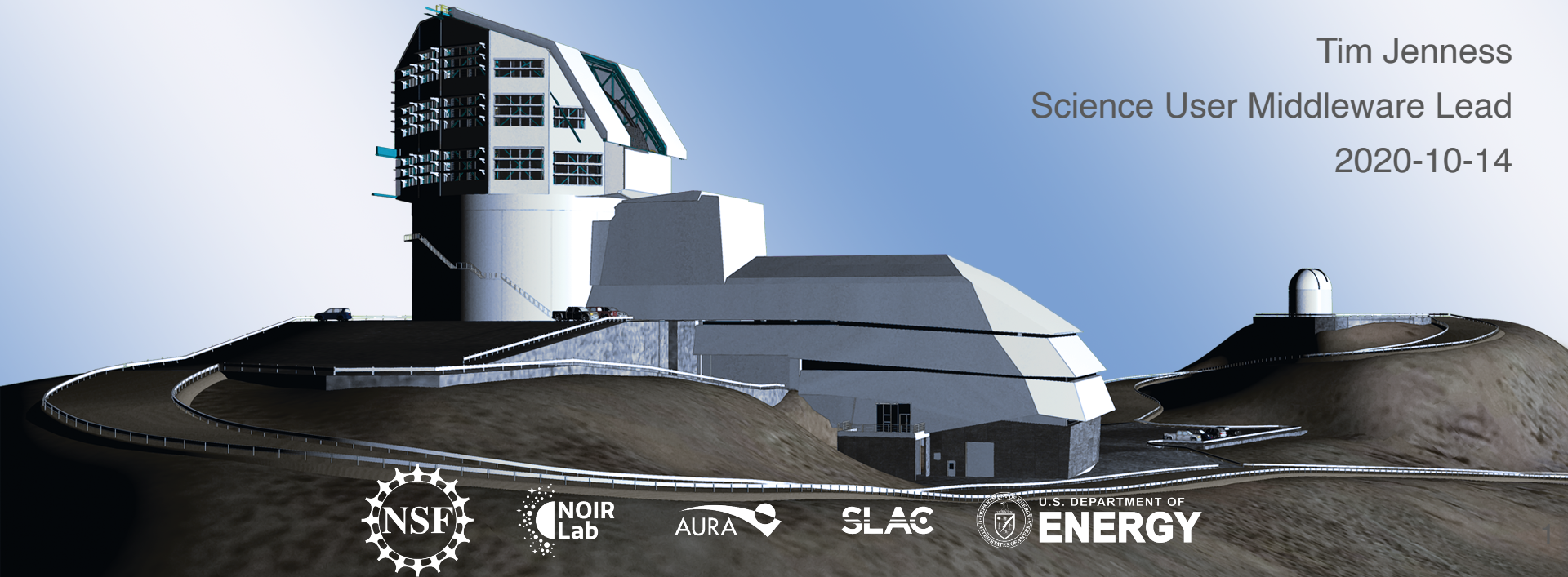
Rubin Observatory

Developing Rubin Software

Tim Jenness
Science User Middleware Lead
2020-10-14



Background

Rubin Observatory

- Developer Guide is source of truth (https://developer.lsst.io)
  - Has the coding style
  - Describes the development process.

- For a concise overview of the development process see 2018 SPIE paper:
  - Document-28452
  - aka doi:10.1117/12.2312157

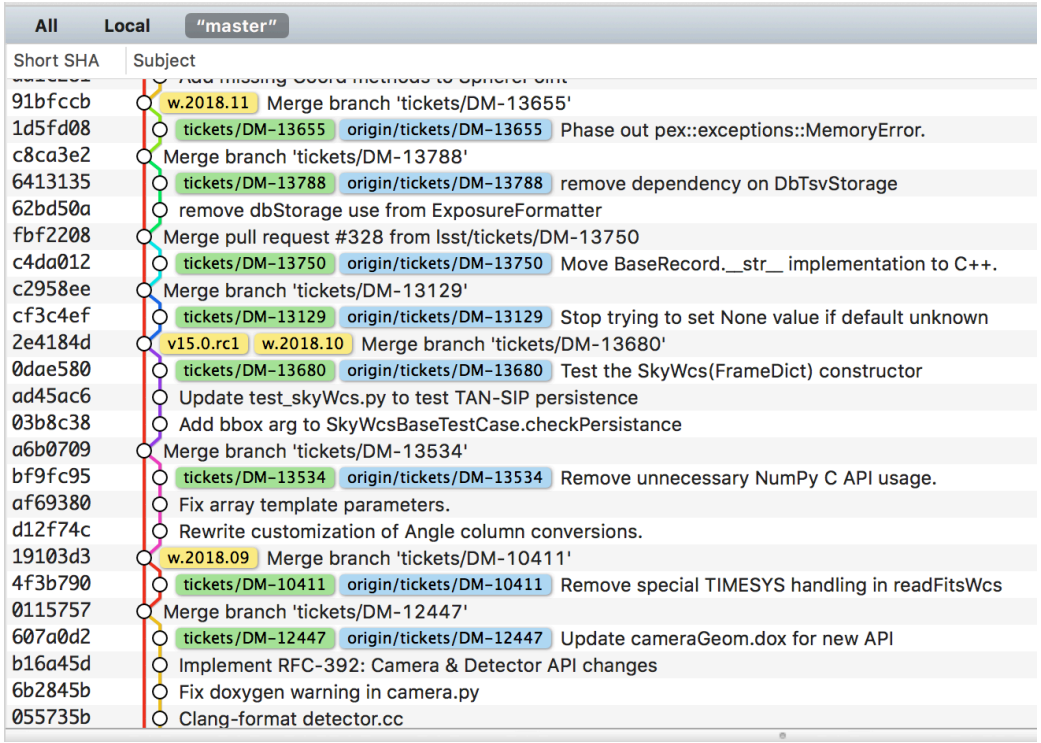
Development Basics

Rubin Observatory

- Python >=3.7 (3.8 soon) and C++14.
- Manage work through Jira with bugs, stories, and epics.
- Git repositories on GitHub in two organizations (lsst, and lsst-dm).
- All work is done on branches named after the Jira ticket (`tickets/DM-1234`).
- All code must be reviewed by at least one person (some exceptions).
- Reviews done on GitHub pull requests but approved on Jira
- We use master as our deployment branch.
- Ticket branches are rebased before merging to master.

Why do we rebase?

Rubin Observatory



EUPS and SCons

Rubin Observatory

- eups is an environment manager used by Science Pipelines developers.
- Don’t panic!
- Like git, a few commands will take you a long way:
  - `eups list lsst_distrib`
  - `setup lsst_distrib`
  - `cd $MY_WORK_DIR`
  - `setup -k -r .`
  - `scons -j4`

Continuous Integration

Rubin Observatory

- Use CI integration with GitHub (migrating from Travis to GitHub Actions)
  - Run `flake8` tool to ensure style compliance.
  - Can also do other checks if required (eg run `mypy`).
  - Enable branch protections to require that branches are up to date before merging. (never hit the “update branch” button on GitHub!).
- Use Jenkins for integration/unit testing.
  - Use `pytest` as the test runner (but `unittest` for tests themselves).
  - ci.lsst.codes lists all the options. `stack-os-matrix` is the common one for testing science pipelines.
  - Must pass before merging branch. (usually `lsst_distrib` + `lsst_ci` for science pipelines).

DEMO

Rubin Observatory