Code Testing and Continuous Integration

Brigitta Sipőcz
DIRAC Fellow, University of Washington
@bsipocz @AstroBrigi

Most astronomers in fact DO test.

However they may do them as manual tests rather than in a formalized, reproducible way.

Why test - How to test

- Does the code run from start to end, do the examples work?
- Result oriented: running on known data, do I get the expected science out of the whole pipeline
- Want to ensure nothing breaks along the line

What testing is

- Code that runs features with known expected result or behavior
- Known expected result can be e.g. a mathematical relation, trivial case for an algorithm or previously buggy but now fixed behavior.

What testing is

- Code that runs features with known expected result or behavior
- Known expected result can be e.g. a mathematical relation, trivial case for an algorithm or previously buggy but now fixed behavior.

Caveats

- Writing tests takes requires significant human resources, but
- Indispensable especially for collaborative projects where developers are joining and leaving and no single person is able to oversee all the details

Unit tests

- Lightweight, code is broken up for the smallest sensible pieces
  - Modular code is critical
- Run them early and often
- Make sure the most basic parts work correctly, cover edge cases

Regression tests

- A way to collectively remember the development history of old bugs.

Other types of tests

- Benchmarking
  - Speed
  - Load
- Style
- Security
- Integration
- ...

Example

```
def test_constellations():

    sc = SkyCoord(135*u.deg, 65*u.deg)
    assert sc.get_constellation() == 'Ursa Major'
    assert sc.get_constellation(short_name=True) == 'UMa'

    scs = SkyCoord([135]*2*u.deg, [65]*2*u.deg)
    npt.assert_equal(scs.get_constellation(), ['Ursa Major']*2)
    npt.assert_equal(scs.get_constellation(short_name=True), ['UMa']*2)
```

Code coverage - The illusion of 100% coverage

- 100% coverage:
  - All code was run
  - it doesn’t mean all code was tested, all cases may not been covered
  - **But**: Code not covered was certainly not tested
- Passing a test doesn’t mean the code is correct
- Incorrect code can pass a tests suite, and correct code can fail
- Tests can be incomplete, tests can be wrong

In practice

- Pytest
  Nose
  unittest
  mock

- Test driven development:
  - write the tests based on the API design
  - test results of algorithms based on extra knowledge
    - from analytic solutions of simple cases
    - test data design (test data are drawn from known distribution)

Continuous integration

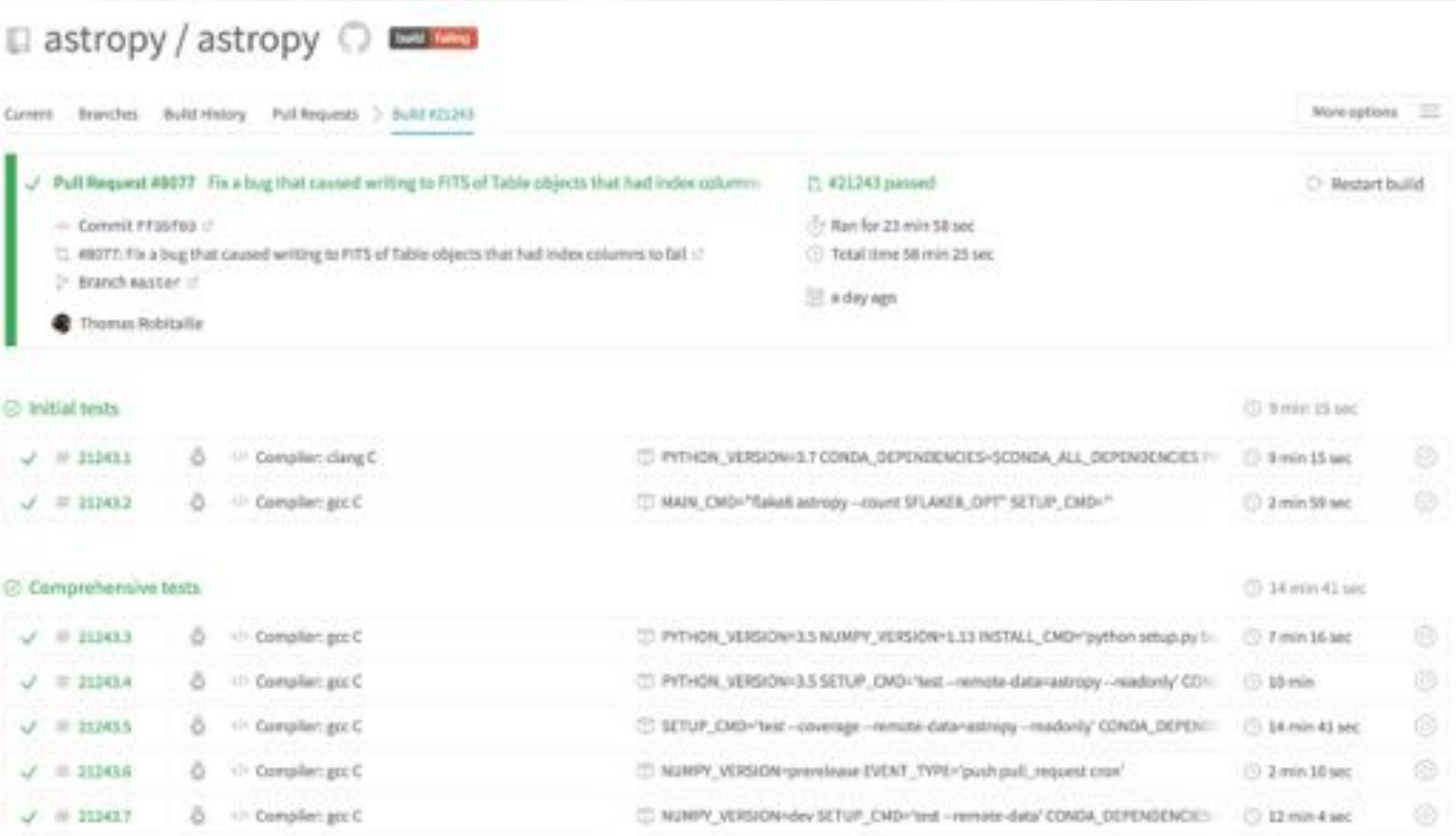
- Running tests automatically for all proposed code changes.
- Goal: Catch problems, incompatibility with the existing codebase introduced by new modifications as soon as possible.

Continuous Integration

- Self hosted, e.g. 

- Cloud hosted providers, integrated with GitHub:
  - Travis CI 
  - Appveyor 
  - Azure Pipeline 
  - CircleCI 
  - Codecov 

Continuous Integration



Continuous Integration