




# How to be properly lazy

*Like, really lazy!*

Ellert van der Velden

ADACS Astro Hack Week 2020



Code coverage: What?

# Code coverage: What?

```
----- coverage: platform win32, python 3.6.6-final-0 -----
Name                               Stmts  Miss  Cover   Missing
-----
example_scripts\__init__.py         0      0   100%
example_scripts\downsampler.py      91     69    24%    39-103, 141-172, 215-217, 222-225, 228-231, 263-273
example_scripts\galaxy.py           58     21    64%    51-52, 149, 168-169, 206-219, 245-254
-----
TOTAL                               149     90    40%
```



# Code coverage: Why?

# Code coverage: Why?

- Write near-exhaustive tests;

# Code coverage: Why?

- Write near-exhaustive tests;
- Check for code redundancy;

# Code coverage: Why?

- Write near-exhaustive tests;
- Check for code redundancy;
- Find non-covered code;

# Code coverage: Why?

- Write near-exhaustive tests;
- Check for code redundancy;
- Find non-covered code;
- Special test-cases help in the future.





Code coverage: How?

# Code coverage: How?

- Aim for 100% coverage, including branch coverage;

# Code coverage: How?

- Aim for 100% coverage, including branch coverage;
- If that is not possible, ask yourself why (maybe use *pragma: no cover*);

# Code coverage: How?

- Aim for 100% coverage, including branch coverage;
- If that is not possible, ask yourself why (maybe use *pragma: no cover*);

```
main_code()
if flag: # pragma: no cover
    do_action()
    do_another_action()
main_code_continued()
```

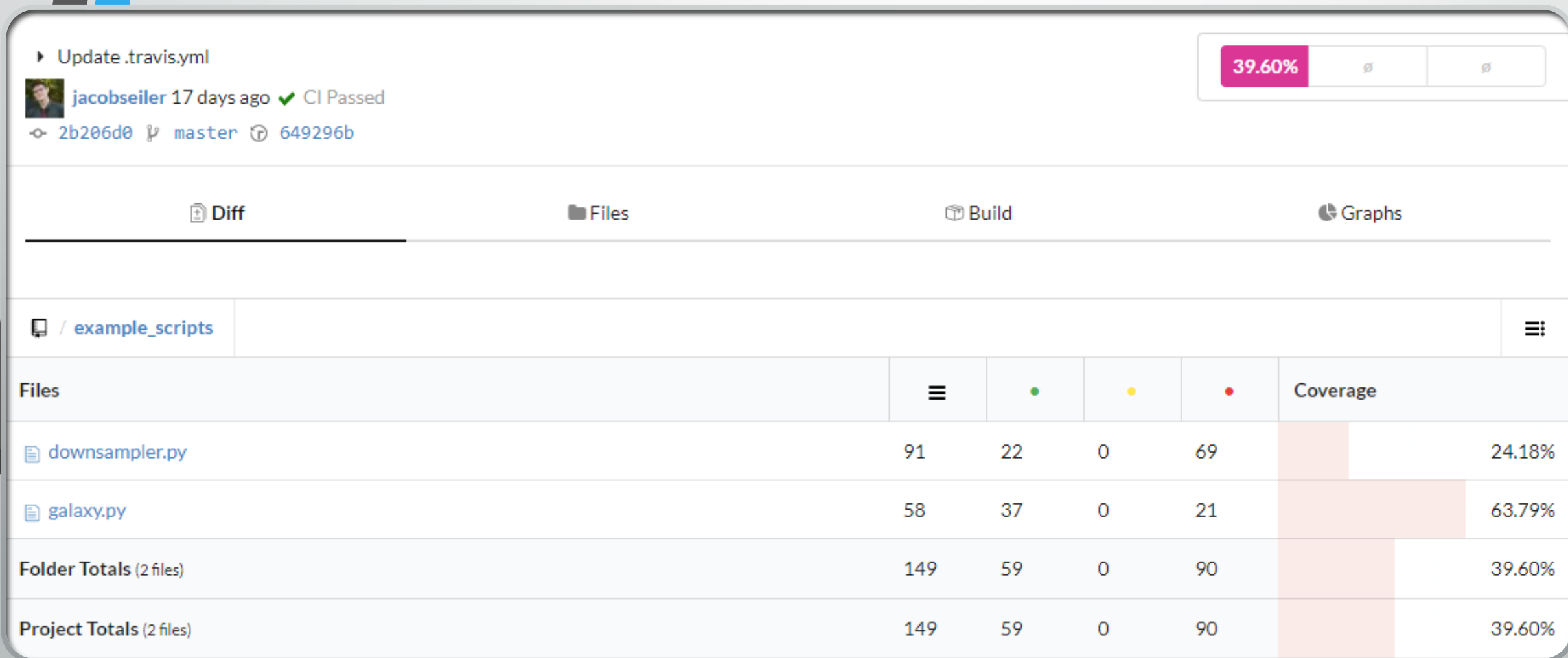
# Code coverage: How?

- Aim for 100% coverage, including branch coverage;
- If that is not possible, ask yourself why (maybe use *pragma: no cover*);
- Make sure to write a single test for a single coverage case (e.g., do not cover multiple exception cases in the same test).



Code coverage: Where?

# Code coverage: Where?





Code coverage: Where?



# Code coverage: Where?

- Once you have your coverage reports, you can upload them to CodeCov;

# Code coverage: Where?

- Once you have your coverage reports, you can upload them to CodeCov;
- CodeCov keeps track of your code coverage;

# Code coverage: Where?

- Once you have your coverage reports, you can upload them to CodeCov;
- CodeCov keeps track of your code coverage;
- It can also provide commit status messages.

# Code coverage: Where?

Listing all repositories sort by **most recent commit**



**PRISM**

Latest commit 2 days ago by 1313e

79.27%

< 0.00% >



**e13Tools**

Latest commit 3 days ago by 1313e

100.00%

< 100.00% >



**software-testing**

Latest commit 17 days ago by jacobseiler

39.60%



**mpi4pyd**

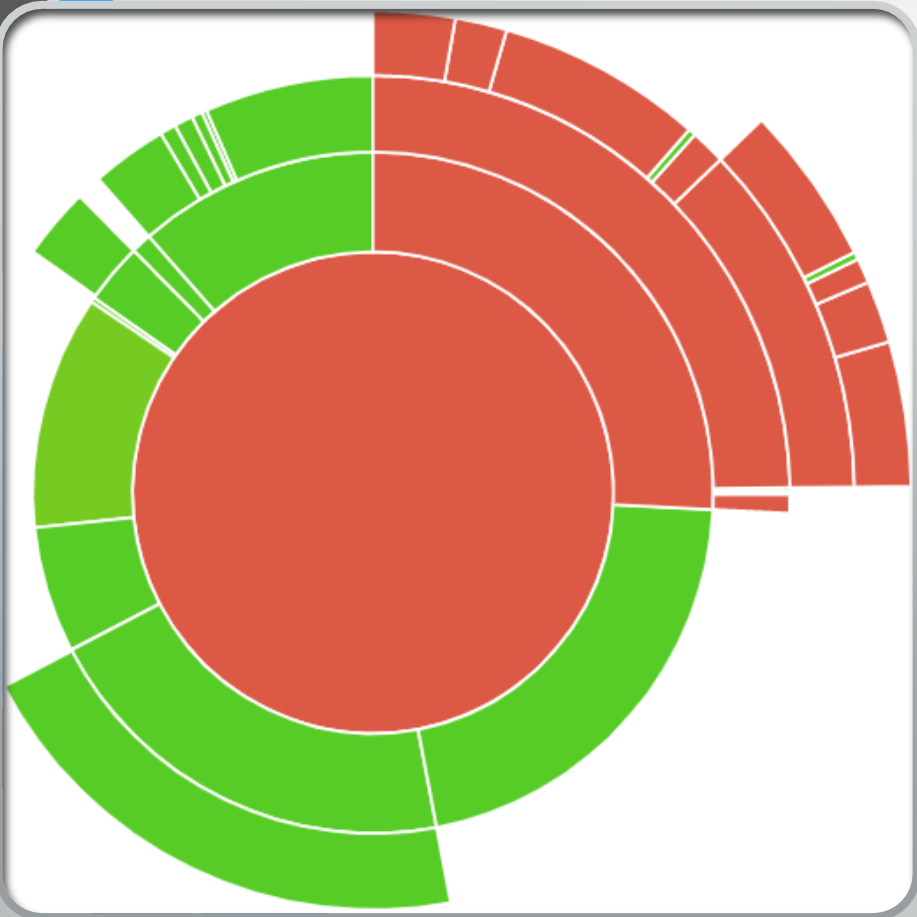
Latest commit 3 months ago by 1313e

76.27%

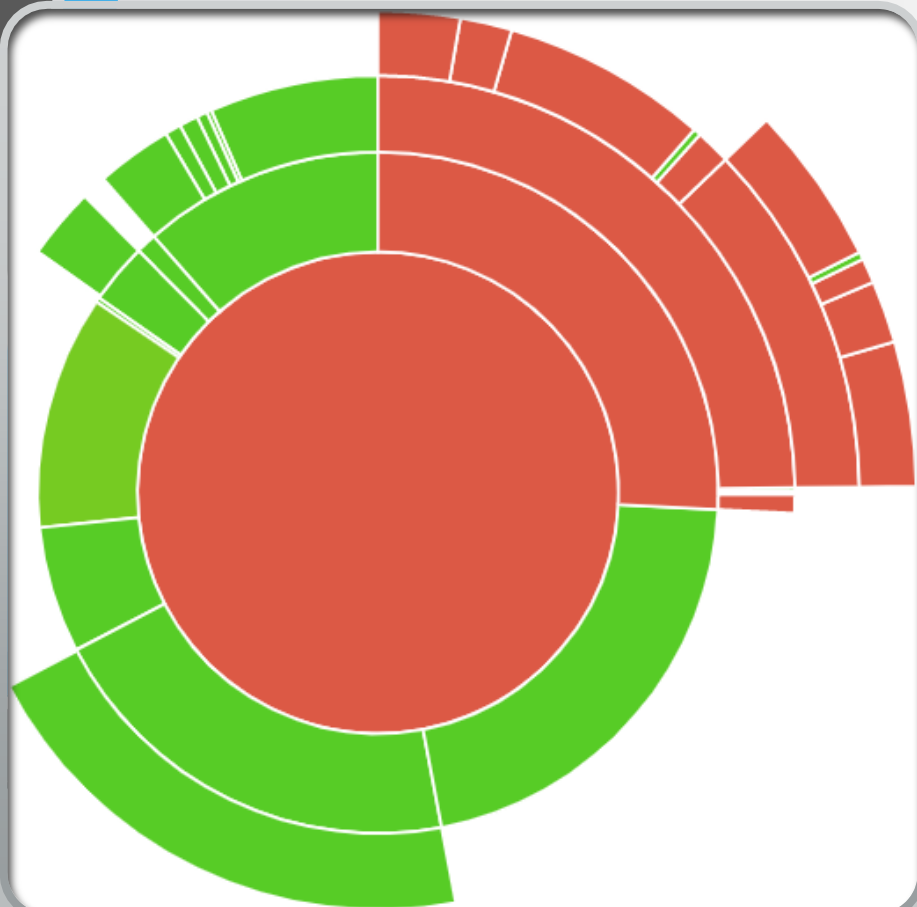
< 94.12% >









# Code coverage: Where?



# Code coverage: Where?



+115 +24 +91		
	[WIP #27] Added a details option that gives the details overview of an emulator iteration. 1313e 12 days ago dev a9c4a4d ✓ CI Passed	85.16% < 20.93% > (-2.55%)
+119 +8 +111		
	[WIP #27] Fixed that the options menu sometimes did not open in the center of the GUI. 1313e 13 days ago dev 04f1ac8	87.40% < 9.68% > 0
+16 +1 +15		
	[WIP #27] Made some modifications to the way figures are saved. 1313e 14 days ago dev 60d677c ✓ CI Passed	87.71% < 6.78% > (-0.31%)
+25 +1 +24		
	[WIP #27] More improvements to the Projection GUI. 1313e 14 days ago dev 46ad99c ✓ CI Passed	88.21% < 20.15% > (-2.90%)
+159 +25 +134		
	[WIP #27] Changed the two main parts of the GUI to be dock widgets, allowing them to be moved by the user. 1313e 14 days ago dev 5e9f1a3	91.12% < 4.00% > 0
+42 +2 +40		
	Enforce that PyQt5 is used.	92.04% < 25.00% > 0

Continuous Integration

Or

Let the Damn Computer Worry  
About It

Jacob Seiler

# State of the Game

- Have a repo which contains tests run using the command 'pytest'.



# State of the Game

- Have a repo which contains tests run using the command 'pytest'.
- Ensures your code can run on your machine!

# State of the Game

- Have a repo which contains tests run using the command 'pytest'.
- Ensures your code can run on your machine!
- Make sure you run 'pytest' before you push a commit!

# I'm Lazy and Don't Like Thinking

- Remembering to run PyTest every single time before you push is an absolute nightmare.

# I'm Lazy and Don't Like Thinking

- Remembering to run PyTest every single time before you push is an absolute nightmare.

# I'm Lazy and Don't Like Thinking

- Remembering to run PyTest every single time before you push is an absolute nightmare.
- What happens if you forget and your code is sitting on GitHub broke...?

# I'm Lazy and Don't Like Thinking

- Remembering to run PyTest every single time before you push is an absolute nightmare.
- What happens if you forget and your code is sitting on GitHub broke...?
- Also, the tests run only on your machine. What happens if someone is running Windows? Or a different Python version?

# I'm Lazy and Don't Like Thinking

- Remembering to run PyTest every single time before you push is an absolute nightmare.
- What happens if you forget and your code is sitting on GitHub broke...?
- Also, the tests run only on your machine. What happens if someone is running Windows? Or a different Python version?
- What if we set it up so our tests automatically run every time we push? And what if we could decide what environment to run these tests on...?

# I'm Lazy and Don't Want to Push

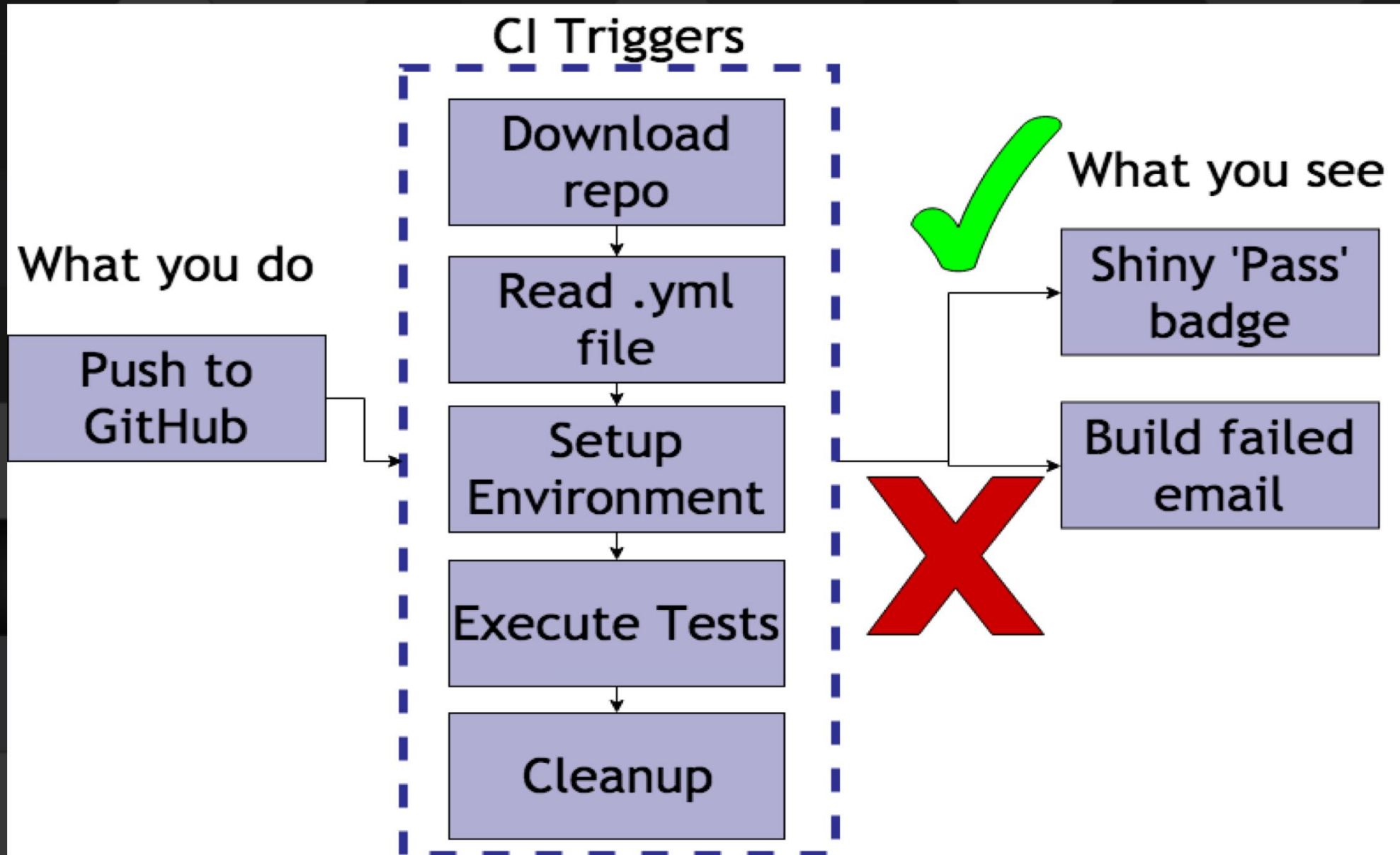
- Remembering to run Deployment Push is an absolute nightmare

## Continuous Integration

- What if someone pushes code to the repository? This automatically run every time we push code and decide what environment to run these tests



# Continuous Integration



# The .travis.yml File

- What Python versions or OS should be tested?

# The .travis.yml File

- What Python versions or OS should be tested?
- What should Travis do once it's cloned your repo?

# The .travis.yml File

- What Python versions or OS should be tested?
- What should Travis do once it's cloned your repo?
- What happens if the tests pass/fail?

# The .travis.yml File

- What Python versions or OS should be tested?
- What should Travis do once it's cloned your repo?
- What happens if the tests pass/fail?
- What happens after your tests have been executed?

# Your Turn!

- Log into Travis CI and enable builds for your fork of the workshop repo (<https://github.com/1313e/software-dev>).

# Your Turn!

- Log into Travis CI and enable builds for your fork of the workshop repo (<https://github.com/1313e/software-dev>).
- Take the template .yml file and adjust it to be 'correct'.

# Your Turn!

- Log into Travis CI and enable builds for your fork of the workshop repo (<https://github.com/1313e/software-dev>).
- Take the template .yml file and adjust it to be 'correct'.
- Track the .yml file. Commit. Push!



# Your Turn!

- Log into Travis CI and enable builds for your fork of the workshop repo (<https://github.com/1313e/software-dev>).
- Take the template .yml file and adjust it to be 'correct'.
- Track the .yml file. Commit. Push!
- Did your tests pass?
  - No? Change the .yml file and commit/push until it does!
  - Yes? Add the shiny 'pass' badge to the README!

# Your Turn!

- Log into Travis CI and enable builds for your fork of the workshop repo (<https://github.com/1313e/software-dev>).
- Take the template .yml file and adjust it to be 'correct'.
- Track the .yml file. Commit. Push!
- Did your tests pass?
  - No? Change the .yml file and commit/push until it does!
  - Yes? Add the shiny 'pass' badge to the README!
- All done? Work on getting 'codecov' to automatically run and update your results to 'codecov.io'. Then get the codecov badge!