Continuous Integration

Frossie Economou

1st ASTERICS-OBELICS International School

6-9 June 2017, Annecy, France.

H2020-Astronomy ESFRI and Research Infrastructure Cluster
(Grant Agreement number: 653477).

… OR: HOW TO SAVE ASTRONOMY ONE GIT COMMIT AT A TIME

Frossie Economou (frossie@lsst.org)
Science Quality and Reliability Engineering (SQuaRE)
Data Management
Large Synoptic Survey Telescope (LSST)

One is way better than none

UNIT TESTING

What is unit testing?

In *computer programming,* ***unit testing is a software testing method by which individual units of source code****, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures,* ***are tested to determine whether they are fit for use***

Wikipedia on Unit Testing

Terminology

- Unit test(s): a piece of code whose purpose is to test a single feature or behaviour of the code (a "unit" of functionality)
- Test framework or test runner: a harness for running tests and reporting success or failure (eg. pytest)
- Code coverage: The percentage of code that is "covered" by unit tests

Example 0: Using pytest

- See https://github.com/Asterics2020-Obelics/simple_pytest
- Note the simple test we have written that we can run using py.test
- We'll get back to this later

Don't fixate on coverage

The easiest way that I have found to start unit testing an application is as part of fixing a bug. The method goes something like this.

- *Find a bug*
- *Write a test that will pass when the bug is fixed*
- *Change your code until the test passes*

AJ Todd "Why I use pytest" (2013)

L’enfer, c’est les autres. Their code, anyway.

CONTINUOUS INTEGRATION

"Is this thing on?"

I wrote some code. It works! Or does it?

- Does this work at all (compile/run)
- Does it work as I intended (run unit tests)
- Does it **integrate** (work with other code, in different environments, with a production system)

➠ The process of checking integration every time a code change is made is **Continuous Integration**

What is CI ?

CI is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. ***CI*** *most often refers to the build or integration stage of the software release process and* ***entails both an automation component*** *(e.g. a CI or build service)* ***and a cultural component*** *(e.g. learning to integrate frequently).*

AWS documentation

Why CI?

- It is a great incentive to write tests
- People forget to run tests
- Tests often will pass only because of your environment (that PYTHONPATH that you set…)
- CI encourages Agile software development practices
- CI can test for other dependency chains and portability (eg. Py3)
- CI can even deploy your code (later)
- CI makes it easier to accept code contributions

Example 1: Setting up Travis

- Back to our previous example
- https://github.com/Asterics2020-Obelics/simple_pytest

Example 2: CI as portability testing

- See https://github.com/Asterics2020-Obelics/github-demo

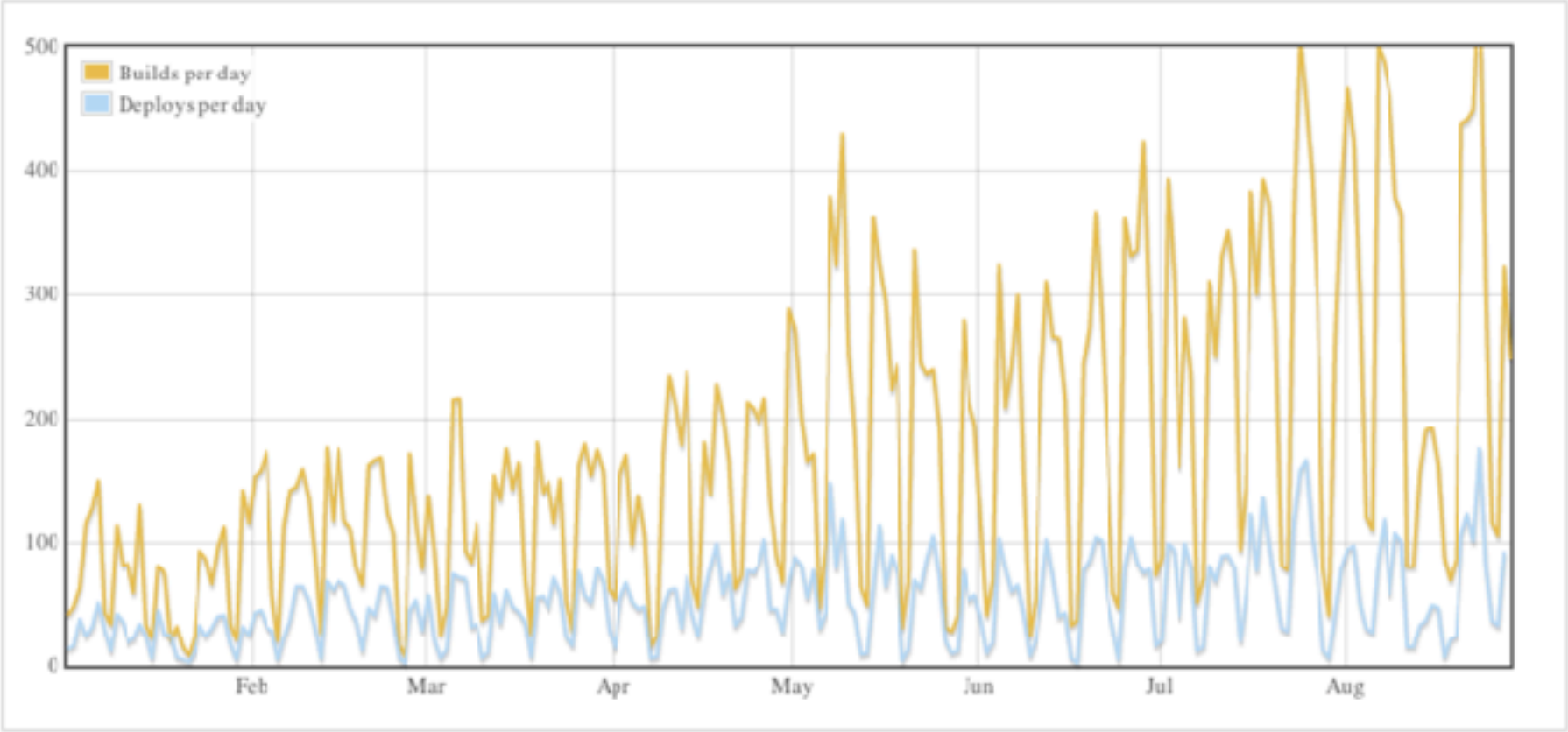
So an astronomer walked into a bar in Silicon Valley and…

CONTINUOUS DEPLOYMENT

Terminology

- Continuous Deployment (sometimes Continuous Delivery) is the practice of releasing / deploying into production code as soon as it has passed CI

- The lull in mid-August was our company summit, which kicked off the following week with a big dose of inspiration. Our busiest day yet, Aug. 23, saw 563 builds and 175 deploys.



Github Blog (2012)

Example 3: CI for deployment

- See https://github.com/lsst-dm/dm_dev_guide/blob/master/.travis.yml
- Used to generate developer.lsst.io

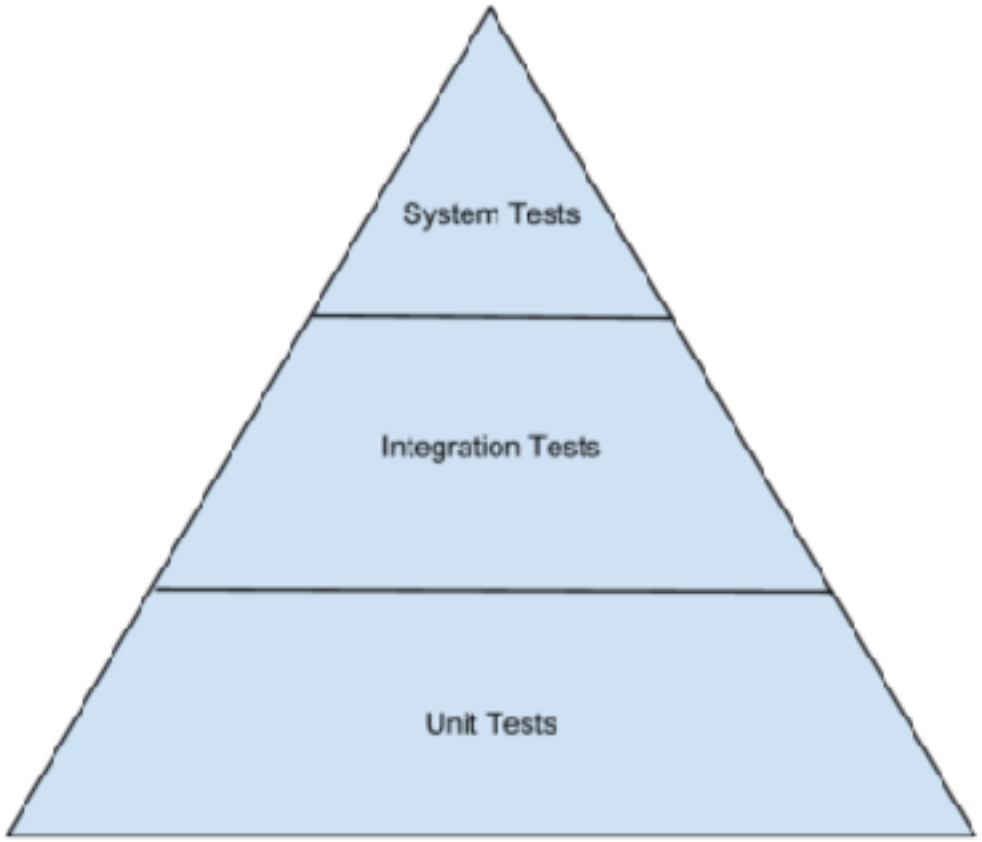


Figure 20.1: The hierarchy of traditional tests.

Site Reliability Engineering (2016)

Day 1 Recap

or, why bother?

Software Engineering skills

- Working with git, especially branches
- Collaborating on Github
- Writing clear, maintainable code
- Testing and debugging
- Using CI

➠ These are great real-world skills for working on sustainable software with other people

Your future self is people too

Almost there

THE END ?

=Grand Groupe 83/1947 , le 4 avril 1947, de 13h30 à 15h.=

(Troisième retour du gr.23/58)

Longueur du Groupe: 215.000 Km.

Merveilleux aspect d'une complicati
invraisemblable. Voir les zones dan
noyau de la tache suivante.

Saving astronomy one git commit at a time

- Effective software engineering is now the critical pathway to scientific discovery. We don't analyse data: we write software to analyse data.
- Writing code does not make you a software engineer. We have a lot to learn from "real-world" software engineering.
- Code for tomorrow and **code together**
- Your professors are not the answer. **You** are the answer

Ask Me Anything

I'm here all week - catch me at ☕ or if you are shy, talk to me on the gitter chat!

- What's it like to work for a telescope / in technical astronomy instead of academia
- Agile/DevOps/Cloud tech and practices
- Women/minorities in astronomy/computing
- Open source and team culture (and why you need a "no-jerks" rule in your collaborations)
- LSST !

Merci!

- H2020-Astronomy ESFRI and Research Infrastructure Cluster (Grant Agreement number: 653477) for making this amazing workshop possible 🇪🇺
- LSST 🔭
- The organizers (esp. Jayesh) 😘
- … and of course you! Keep coding!👇