FOSS4G SEOUL 2015

Building Continuous Integration within your open source project

BY STEVEN D. LANDER, RGi ®

ABOUT ME

- https://github.com/stevendlander
- Software Engineer @ RGi
- Experienced in caching and storing large raster images
- Early implementer of the OGC GeoPackage specification
- Work in Java, Android, Python, and some others

ABOUT OUR PROJECTS

- Software to Aggregate Geospatial Data (SWAGD)
  - Full implementation of the GeoPackage raster spec
  - Java 1.8
  - Uses GDAL 1.11.1
- geopackage-python
  - Naïve implementation of the GeoPackage raster spec
  - Python 2.7 & 3.4
  - Improvements to gdal2tiles.py along with separate script to package tiles

OTHER PRESENTATIONS AT FOSS4G SEOUL

- *Geopackage and how open source is changing the way governments think about standards*
  - (2015/09/16) Nathan Frantz, Ben Tuttle, 11:25 PT1-05
- *OGC GeoPackage in practice: Implementing a new OGC specification with open source tools*
  - (2015/09/17) Steven Lander, 11:25 PT4-08

AT A GLANCE

- Why do we need CI? What is the problem?
- How do we solve those problems?
- Our best practices for implementing CI within an open source project

THE PROBLEM

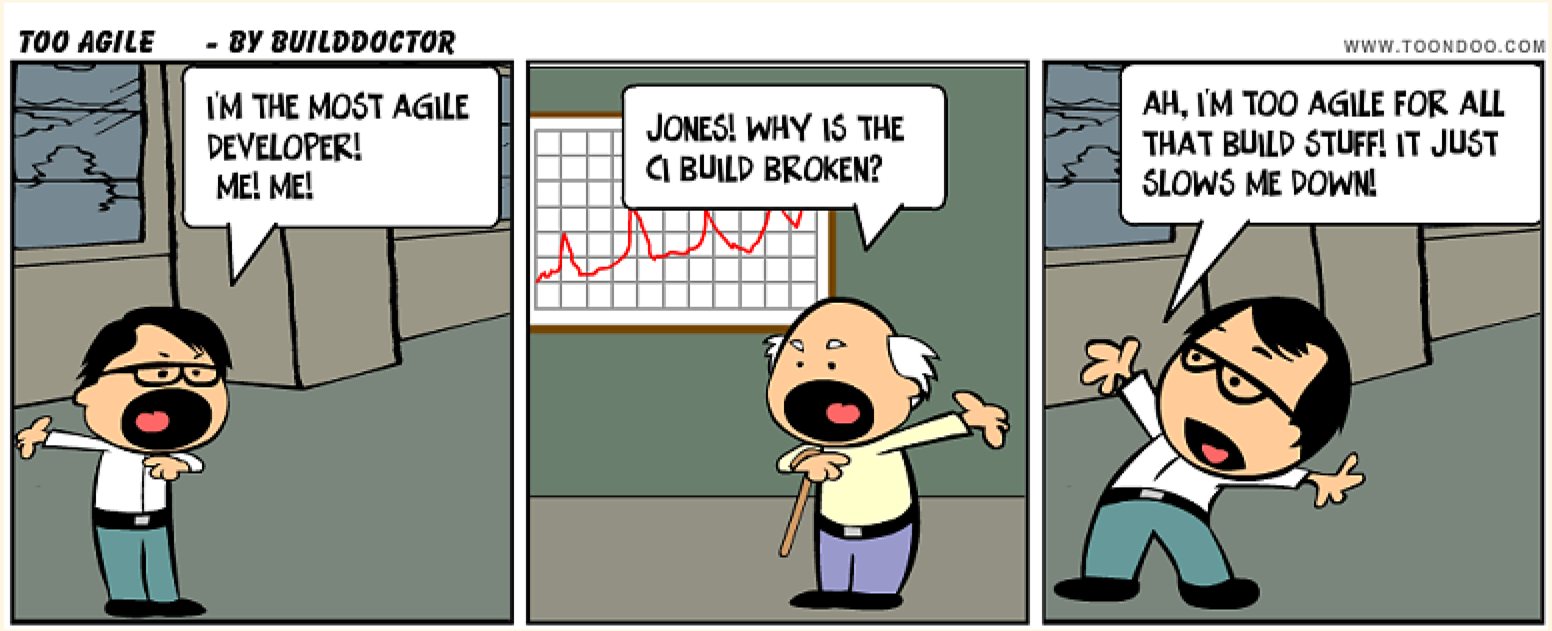
(broadly)

MODERN BUILD SYSTEMS ARE COMPLICATED

- Build framework
- Dependencies
- Testing
- Installers?

HUMAN FACTORS

- Typing and logic errors
- Poor understanding of code impacts
- Improper software development practices


TOO AGILE - BY BUILDDOCTOR
WWW.TOONDOO.COM
I'M THE MOST AGILE DEVELOPER! ME! ME!
JONES! WHY IS THE CI BUILD BROKEN?
AH, I'M TOO AGILE FOR ALL THAT BUILD STUFF! IT JUST SLOWS ME DOWN!

SOURCE: HTTP://BUILD-DOCTOR.COM/2008/11/22/TOO-AGILE/

REGRESSIONS

- Side effects of development or refactoring
- A change to **Area A** breaks functionality in **Area B**
- Elimination of resources necessary in other areas

BARRIERS

- Too many login credentials to manage
- **The Right Process**™ is onerous to follow
- Manual steps where automation makes more sense

SELF-HOSTED RESOURCES

- Require administration (patches, updates, etc.)
- Downtimes and scheduled maintenance
- Access outside of intranet

OUR CI SOLUTION

MANY ZERO-COST TOOLS FOR BEING OPEN SOURCE!

- Travis-CI - build automation
- Scrutinizer - build automation, code quality analysis
- WaffleIO - issue management
- Coveralls - unit test coverage
- Coverity - static analysis

OTHER CI TOOLS & APPROACHES

- IDE (Eclipse, IntelliJ, PyCharm)
  - Enforce rigorous code standards and common syntax convention
- Design reviews
- Code reviews

MITIGATION

	complicated build system	human factors	regression	barriers	sel host
Source control	✓		✓	✓	
IDE		✓			
Testing		✓	✓		
Services	✓	✓	✓	✓	✓
Static analysis		✓	✓		

SERVICE COMPARISON

	build process	code quality	static analysis	issue mgmt	test coverage
Travis-CI	✓				
Scrutinizer	✓	✓			✓
Coveralls					✓
Coverity			✓		
WaffleIO				✓	

TRAVIS-CI

- Simple workflow:
  - Install your dependencies (script or local)
  - Test if your code compiles
  - Run your test suite
  - Create a code coverage report from your test suite
  - Submit your coverage report to a service

.TRAVIS.YML (FOR JAVA)

```
language: java
jdk:
    - oraclejdk8
install:
    - ./install-deps-linux.sh
    - mvn test-compile -DskipTests=true -Dmaven.javadoc.skip=true -B -V
script:
    - mvn test
after_success:
    - mvn clean cobertura:cobertura coveralls:report
cache:
    directories:
        - $HOME/.m2
```

.TRAVIS.YML (FOR PYTHON)

```
language: python
python:
    - "2.7"
    - "3.4"
install:
    - pip install -r dependencies.txt
    - pip install coveralls
script:
    - py.test Testing/test_tiles2gpkg.py --doctest-modules -v --cov Packa
after_success:
    - coveralls
```

SCRUTINIZER

- Point the service to your open source repo
- Optionally configure how to run the test suite

WAFFLE.IO

- Tracks Github issues in a friendly interface
- Customizable "swim lanes" track issues for the team
- Can aggregate multiple Waffle boards into a single view

COVERALLS

- Code-coverage report visualization and tracking
- Will post to Github about the impact a pull request will have upon code coverage

COVERITY

- Keep a separate branch to limit requests to service
- Can track defects introduced and ones that are still unaddressed

OUR BEST PRACTICES FOR IMPLEMENTING CI WITHIN AN OPEN SOURCE PROJECT

YOUR INTEGRATED DEV ENVIRONMENT (IDE)

- First line of defense
  - Share your code inspection files in VCS
  - Text editor plugins (JSLint/Hint, pylint, etc)
  - Custom builders tailored to your workflow

EFFECTIVE USE OF GITHUB

- Do **not** commit to master; commit to a branch
- Issue pull requests from branches:
  - Easy code reviews
  - Travis will tell you if **a) the push** or **b) the merge** will break
- Use labels, releases, and milestones

TESTING

- Unit test
  - As much as you can (goal = 100%)
  - Exclude code that is impractical or impossible to test
- Integration test
  - Reach across code to other systems (database, web, etc)
  - Clean up after themselves
- Acceptance test
  - Simulate a user or a workflow
  - Check for logic issues
  - Robot Framework

CODE COVERAGE REPORT

- Set a threshold for a failed build if test coverage drops below an acceptable level
- Use as a guide for future test case creation

STATIC ANALYSIS

- Catch bad coding practices early
- Discover logical errors
- Either as an IDE plugin (Findbugs) or a service (Coverity)
- Different analyzers can catch different bugs

DOCUMENTATION AND WIKI

- Github wiki
- Markdown files per folder
- Or, just documentation only

SUMMARY

- Build systems can be quite complicated
- Open source software gets solid tools for **no cost**
- Follow good CI practices

QUESTIONS?

THANKS