

# Better Software Development Through Automated Tooling

Vaughan Kitchen

April 28, 2017

# Structure

- ▶ Background
- ▶ Survey of Tools
- ▶ Live Demonstration
- ▶ Project Examples
- ▶ Related Topics

# Background

## What

Plug as many automatic tools as possible into the development of JASSv2 (A search engine being developed by Andrew Trotman)

## When

A summer studentship at the start of 2017

## How

The approach was to first figure out what tools we might want. Then what tools were available to us. Which of the available ones were worth integrating. And then get them integrated

# An Overview of Tools

- ▶ Unit Tests
- ▶ Integration Tests
- ▶ System Tests
- ▶ Regression Tests
- ▶ Code Coverage
- ▶ Static Analysis
- ▶ Memory & Leak Checks
- ▶ Performance Profiling
- ▶ Code Hygiene
- ▶ Code Documentation

# Continuous Integration pt.1

## Travis CI

- ▶ Automates Build
- ▶ Integrates with Github (or BitBucket for some CI systems)
- ▶ Runs on every push
- ▶ An hour of compute time per VM instance
- ▶ What else can it do besides build?

# Unit Testing 1

- ▶ Part of your test suite
- ▶ Runs from your Continuous Integration system
- ▶ Assures the quality of individual units
- ▶ Smallest testable unit (Function, or Class)
- ▶ Simplifies integration
- ▶ Test-Driven Development
- ▶ JUnit, CUnit, Unity, tinytest, Jasmine...

## Unit Testing 2

---

```
1 #include "unity.h"
2 #include "string2.h"
3
4 void test_string_append(void)
5 {
6     /* test the normal use case */
7     struct string *s = string_new_c("cat");
8     string_append_c(s, "dog");
9     TEST_ASSERT_EQUAL_STRING(s->str, "catdog");
10    TEST_ASSERT_EQUAL_UINT(s->bytes, 6);
11
12    string_free(s);
13 }
```

---

# Integration Testing

- ▶ Part of your test suite
- ▶ Runs from your Continuous Integration system
- ▶ Next step up from unit tests
- ▶ Tests whether groups of modules work together correctly
- ▶ Doesn't test the correctness of the system as a whole



# System Testing

- ▶ Does the application do what it should?
- ▶ Hard to automate entirely
- ▶ Can automate some code paths (CI has full power of Linux)
- ▶ bats (Bash Automated Testing System)

# Regression Testing

- ▶ Does the previously functional software still function correctly?
- ▶ Computing power is now cheap
- ▶ Run the entire test suite for each push
- ▶ Continuous Integration system will notify on breakage
- ▶ Regression tests are now free

# Code Coverage

- ▶ How much of the source code does the test suite execute?
- ▶ Do tests cover succeeding and failing paths?
- ▶ Function coverage
- ▶ Statement coverage
- ▶ Branch coverage
- ▶ Condition coverage
- ▶ Codecov, Coveralls

## Code Coverage Badge



# Static Analysis

- ▶ Verify behaviour
- ▶ Lint for best practices
- ▶ Prove correctness
- ▶ Sometimes freely available to Open Source projects
- ▶ Can be integrated into CI

## Coverity Scan Badge



# memcheck

- ▶ Dynamic analysis
- ▶ Memory leaks
- ▶ Corrupted memory
- ▶ Valgrind

# Performance Profiling

- ▶ Algorithmic performance
- ▶ Web performance
- ▶ Responsive vs Throughput
- ▶ Part of regression testing
- ▶ Instrumentation, Sampling
- ▶ Gprof, Callgrind

# Code Hygiene

- ▶ Code smell
- ▶ Linting
- ▶ Best Practices
- ▶ Anti-patterns
- ▶ CodeClimate, CodeLingo

# Code Documentation

- ▶ Doxygen, Javadoc
- ▶ CodeDocs.xyz
- ▶ GitHub Pages



## Continuous Integration pt.2

- ▶ Merge all working copies to mainline several times a day
- ▶ Prevents "integration hell"
- ▶ Feature toggles for partially complete code
- ▶ Continuous Delivery (mainline is always in a deployable state)
- ▶ Continuous Deployment (automatic deployment to production)

# YAML 1

- ▶ Superset of JSON
- ▶ Supports comments
- ▶ Often used as a configuration format

## YAML 2

---

```
1  "object": {
2      "key": "value",
3      "array": [
4          {
5              "null_value": null
6          },
7          {
8              "boolean": true
9          },
10         {
11             "integer": 1
12         },
13         {
14             "string": "rope"
15         }
16     ]
17 }
```

# YAML 3

---

```
1 object:
2     key: value
3     array:
4         - null_value:
5         - boolean: true
6         - integer: 1
7         - string: rope
```

---

# Our Set Up

- ▶ Travis CI (OSX, and Linux)
- ▶ AppVeyor (Windows)
- ▶ Unit Testing in a custom framework
- ▶ Coverity (coverity\_scan branch)
- ▶ Valgrind
- ▶ CodeCov (Gcov)
- ▶ CodeDocs.xyz (integrates directly with GitHub)

# Live Demonstration

# Other Set Ups

- ▶ <https://github.com/DandyHQ/mace-prototype>
- ▶ <https://github.com/rubinius/rubinius>
- ▶ <https://github.com/IronLanguages/main>

# Continuous Deployment

empty



# Property Based Testing

empty

# Trunk Development and Feature Toggles

empty

# Roll Your Own

empty

# How Integrations Work

empty