Code Coverage & Continuous Integration

ATPESC 2019

Jared O’Neal
Mathematics and Computer Science Division
Argonne National Laboratory

Q Center, St. Charles, IL (USA)
July 28 – August 9, 2019
License, citation, and acknowledgments

License and Citation

• This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).


Acknowledgements

• This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

• This work was performed in part at the Argonne National Laboratory, which is managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357

• Alicia Klinvex
CODE COVERAGE
How do we determine what other tests are needed?

Code coverage tools

- Expose parts of the code that aren’t being tested
  - gcov
    - standard utility with the GNU compiler collection suite
    - Compile/link with –coverage & turn off optimization
    - counts the number of times each statement is executed
  - lcov
    - a graphical front-end for gcov
- Hosted servers (e.g. coveralls, codecov)
  - graphical visualization of results
  - push results to server through continuous integration server
Code Coverage Output

Overall Analysis

Detailed Analysis

https://github.com/jrdoneal/infrastructure
Code Coverage is Popular

- gcov also works for C and Fortran
- Other tools exist for other languages
  - JCoV for Java
  - Coverage.py for python
  - Devel::Cover for perl
  - profile for MATLAB
  - etc.
Limitations

- 100% coverage by line
- Checks 2 of 4 pathways only
- Possibility for bugs
Other Code Coverage

Test Driven Development
• Covers functionality coevolved with tests
• Limited if we have only unit tests

Requirements & Verification
• Covers higher-level functionality and constraints
• Depends on completeness
CONTINUOUS INTEGRATION
The Short & Sweet of Continuous Integration

A master branch that always works

- DVCS workflow isolate master from integration environment
- Extend workflow to address difficulties of integrating
  - Minimize likelihood of merge conflict
  - Detect bugs immediately
  - Make debugging process quick and easy
Work Decomposition

Commit and integrate often

- Limit divergence between feature and master branches
- Decreased probability of conflict
- Conflict resolution is simpler and less risky

![Diagram showing the difference between integrating often and integrating less frequently.](image)
Error Detection

Test at integration to identify failures immediately

- Control quality of code
- Isolate failure to few commits
- No context switching for programmer

We want a system that

- triggers automated builds/tests on target environments when code changes and
- ideally tests on proposed merge product without finalizing merge.
Test Servers

Servers that

• automate the execution of a test suite or a subset of a test suite,
• allow for running tests on different environments,
• host an interface for viewing results, and
• allows for configuring when the tests are run.

Examples

• CTest/CDash
• Jenkins
• Travis CI and GitLab CI
Cloud-based Test Servers

- Linked to VCS hosts
  - GitHub & Travis CI
  - GitLab CI
  - BitBucket Pipelines

- Automated builds/tests triggered *via* pushes and pull requests
- Builds/tests can be run on cloud systems
- Test results are reported in repository’s web interface
- Can trigger code coverage analysis & documentation build
Continuous integration (CI)

- Has existed for some time and interest is growing
- HPC community working to adapt CI for HPC machines
- Setup, maintenance, and monitoring required
- Prerequisites
  - A reasonably automated build system
  - An automated test system with significant test coverage & useful feedback
  - Builds/tests must finish in reasonable about of time
  - Ability to bundle subset of tests
Simplest CI example
https://github.com/jrdoneal/CI_HelloWorld
https://travis-ci.org/jrdoneal/CI_HelloWorld

CI example w/ multiple platforms and specific compiler versions
https://github.com/jrdoneal/CI_Multiplatform

Code coverage, testing and CI tutorial (C++)
https://github.com/amklinv/morpheus

Code coverage, testing, and CI example (Fortran, C++)
https://github.com/jrdoneal/infrastructure
<table>
<thead>
<tr>
<th>Time</th>
<th>Module</th>
<th>Topic</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:30am-10:15am</td>
<td>01</td>
<td>Objectives, Motivation, &amp; Overview</td>
<td>Katherine Riley, ANL</td>
</tr>
<tr>
<td>10:15am-10:45am</td>
<td>Break</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:45am-11:30am</td>
<td>02</td>
<td>Requirements &amp; Test-Driven Development</td>
<td>Jared O’Neal, ANL</td>
</tr>
<tr>
<td>11:30am-12:30pm</td>
<td>03</td>
<td>Software Design &amp; Testing</td>
<td>Anshu Dubey, ANL</td>
</tr>
<tr>
<td>12:30pm-1:30pm</td>
<td>Lunch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1:30pm-2:15pm</td>
<td>04</td>
<td>Licensing</td>
<td>James Willenbring, SNL</td>
</tr>
<tr>
<td>2:15pm-3:15pm</td>
<td>05</td>
<td>Agile Methodologies &amp; Useful GitHub Tools</td>
<td>James Willenbring, SNL</td>
</tr>
<tr>
<td>3:15pm-3:45pm</td>
<td>Break</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:45pm-4:15pm</td>
<td>06</td>
<td>Git Workflows</td>
<td>Jared O’Neal, ANL</td>
</tr>
<tr>
<td>4:15pm-4:55pm</td>
<td>07</td>
<td>Code Coverage &amp; Continuous Integration</td>
<td>Jared O’Neal, ANL</td>
</tr>
<tr>
<td>4:55pm-5:30pm</td>
<td>08</td>
<td>Software Refactoring &amp; Documentation</td>
<td>Anshu Dubey, ANL</td>
</tr>
</tbody>
</table>
CI HELLO WORLD – BACKUP SLIDES
GitHub Repository Page

https://github.com/jrdoneal/CI_HelloWorld
Travis CI Configuration File

.travis.yml

```
env:
  - TRAVIS_CI_ENV="Hello, World"

#before_install:
#- Put commands here to prepare for executing builds/install
#- Examples would be using apt-get to install dependencies not
#  included in the Travis CI build environment by default.

#install:
#- Put build commands here
#- In each phase, you can execute multiple commands
#- Travis CI stops if any single command fails in this phase

before_script:
  - echo $TRAVIS_CI_ENV

script:
  - $TRAVIS_BUILD_DIR/hello_world.sh
#- Travis CI will run each command in this phase even if a previous command
#  terminated in failure

after_success:
  - echo "You should see that Hello, World was printed by before_script"

after_failure:
  - echo "Hello, World should not have been printed by before_script"
```
The Script Phase

```bash
#!/bin/bash

if [ -z "${TRAVIS_CI_ENV}" ]; then
    echo "Please set the TRAVIS_CI_ENV environment variable"
    exit 1
elif [ "${TRAVIS_CI_ENV}" != "Hello, World" ]; then
    echo "TRAVIS_CI_ENV value is ill-suited for this tutorial"
    exit 2
fi
```
Connecting GitHub & Travis CI

We're only showing your public repositories. You can find your private projects on travis-ci.com.

Legacy Services Integration

Filter repositories

CI_HelloWorld

CI_Multiplatform

infrastructure
Repository in Travis CI

https://travis-ci.org/jrdoneal/CI_HelloWorld

tutorial. Travis CI builds should now be successful.

Commit 93a75c4
Compare ff52718..93a75c4
Branch master

jrdoneal

Ruby

TRAVIS_CI_ENV="Hello, World"
Commit History

```
.travis.yml added
```

This change should lead to a correct build environment for the purposes...

Update Travis CI configuration file so that it is a step closer to..."...

Add Travis CI configuration file. With the present content, the build..."...

Add the script that tests that the build environment is correctly con...

Add README file to explain the intent and eventual content of this tu..."...
Travis CI Build History

```
1 Worker information
2 Build system information

Setting APT mirror in /etc/apt/sources.list: http://us-east-1.ec2.archive.ubuntu.com/ubuntu/

$ git clone --depth=50 --branch=master https://github.com/jrDoneal/CI_HelloWorld.git jrDoneal/CI_HelloWorld
$ rvm use default
$ ruby --version
No Gemfile found, skipping bundle install
$ echo $TRAVIS_CI_ENV

$ $TRAVIS_BUILD_DIR/hello_world.sh
Please set the TRAVIS_CI_ENV environment variable

The command "$TRAVIS_BUILD_DIR/hello_world.sh" exited with 1.

$ echo "Hello, World should not have been printed by before_script"

Done. Your build exited with 1.
```
Travis CI Build History

Update Travis CI configuration file so that it is a step closer to se... ...

1. Worker information
2. Build system information
3. Setting APT mirror in /etc/apt/sources.list: http://us-east-1.ec2.archive.ubuntu.com/ubuntu/
4. $ git clone --depth=50 --branch=master https://github.com/jrdoneal/CI_HelloWorld.git jrdoneal/CI_HelloWorld
5. Setting environment variables from .travis.yml
6. $ export TRAVIS_CI_ENV="This content will result in failure"
7. $ rvm use default
8. $ ruby --version
9. No Gemfile found, skipping bundle install
10. $ echo $TRAVIS_CI_ENV
11. This content will result in failure
12. $ $TRAVIS_BUILD_DIR/hello_world.sh
13. TRAVIS_CI_ENV value is ill-suited for this tutorial
14. The command "$TRAVIS_BUILD_DIR/hello_world.sh" exited with 2.
15. $ echo "Hello, World should not have been printed by before_script"
Travis CI Build History

This change should lead to a correct build environment for the purposes...

Developer D. Develop committed 2 days ago.

```
1 Worker information
2 Build system information

Setting APT mirror in /etc/apt/sources.list: http://us-east-1.ec2.archive.ubuntu.com/ubuntu/

$ git clone --depth=50 --branch=master https://github.com/jrdoneal/CI_HelloWorld.git jrdoneal/CI_HelloWorld

Setting environment variables from .travis.yml

$ export TRAVIS_CI_ENV="Hello, World"

$ rvm use default
$ ruby --version

No Gemfile found, skipping bundle install

$ echo "$TRAVIS_CI_ENV"
Hello, World

$ $TRAVIS_BUILD_DIR/hello_world.sh

The command "$TRAVIS_BUILD_DIR/hello_world.sh" exited with 0.

$ echo "You should see that Hello, World was printed by before_script"

Done. Your build exited with 0.
```
Special Notes for Morpheus Tutorial

• A code coverage and testing tutorial can be found at the Morpheus repository doxygen pages

• **STEP 1**: These exercises must be run on your own local machine or on a remote machine that you have access to.

• If you cannot generate your own gcov output, the associated lcov output is online