Continuous Integration

David E. Bernholdt
Oak Ridge National Laboratory

Mark C. Miller
Lawrence Livermore National Laboratory

Better Scientific Software Tutorial, SC20, November 2020

See slide 2 for license details

License, Citation and Acknowledgements

License and Citation

- This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
- **The requested citation the overall tutorial is: David E. Bernholdt, Anshu Dubey, Patricia A. Grubel, Rinku K. Gupta, Better Scientific Software tutorial, in SC ‘20: International Conference for High Performance Computing, Networking, Storage and Analysis, online, 2020. DOI: 10.6084/m9.figshare.12994376**
- Individual modules may be cited as *Speaker, Module Title*, in Better Scientific Software tutorial…

Acknowledgements

- Additional contributors include: Mike Heroux, Alicia Klinvex, Mark Miller, Jared O’Neal, Katherine Riley, David Rogers, Deborah Stevens, James Willenbring
- 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.
- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
- This work was performed in part at the Lawrence Livermore National Laboratory, which is managed by Lawrence Livermore National Security, LLC for the U.S. Department of Energy under Contract No. DE-AC52-07NA27344.
- This work was performed in part at the Los Alamos National Laboratory, which is managed by Triad National Security, LLC for the U.S. Department of Energy under Contract No.89233218CNA000001
- This work was performed in part at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525.

What is Continuous Integration (CI) *Testing*

- Testing
  - Focused, critical functionality (infrastructure), fast, independent, orthogonal, complete, …
  - Existing test suites often require re-design/refactoring for CI

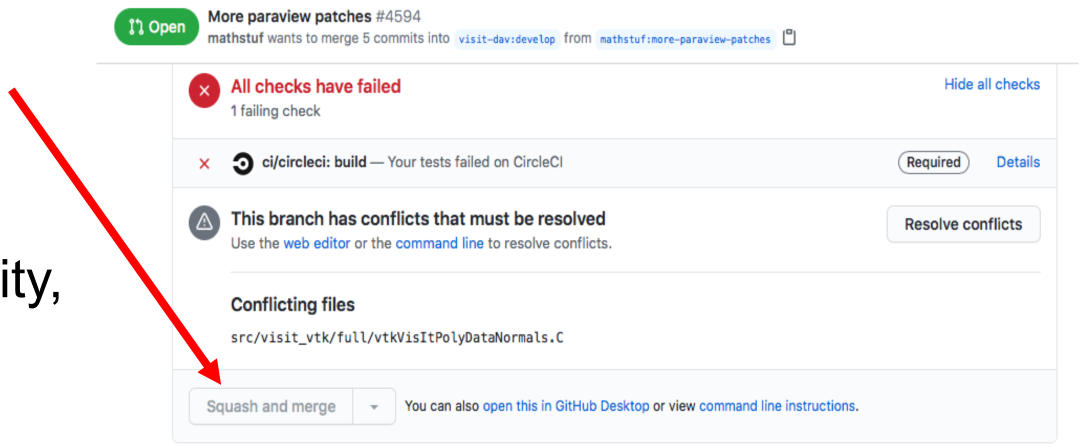
- Integration
  - Changes across key branches merged & tested to ensure the “whole” still works
  - Develop, develop, develop, merge, merge, merge, test, test, test…NO!
  - Develop, merge, test, develop, merge, test, develop, merge, test…YES!

- Continuous
  - Changes tested every commit and/or pull-request (like auto-correct)

- CI generally implies a lot of automation

Automated Testing vs. Continuous Integration (CI) Testing

- ***Automated Testing***: Software that automatically performs tests and reliably detects and reports anomalous behaviors/outcomes.
  - Examples: Auto-test, CTest/CDash, nightly testing, `make check’
  - Lives “next to” your development workflow
  - Potential issues: change attribution, timeliness of results, multiple branches of development
- ***Continuous Integration (CI)***: automated testing performed at high frequency and fine granularity aimed at *preventing* code changes from breaking key branches of development (e.g. *main*)
  - Example: Disabled/enabled “Merge Pull Request” button on GitHub
  - Lives “within” your development workflow
  - Potential issues: extreme automation, test granularity, coverage, 3rd-party services/resources



More paraview patches #4594
mathstuf wants to merge 5 commits into `visit-dav:develop` from `mathstuf:more-paraview-patches`

All checks have failed
1 failing check

ci/circleci: build — Your tests failed on CircleCI

This branch has conflicts that must be resolved
Use the web editor or the command line to resolve conflicts.

Conflicting files
`src/visit_vtk/full/vtkVisItPolyDataNormals.C`

You can also open this in GitHub Desktop or view command line instructions.

What can make CI Difficult

Common situations

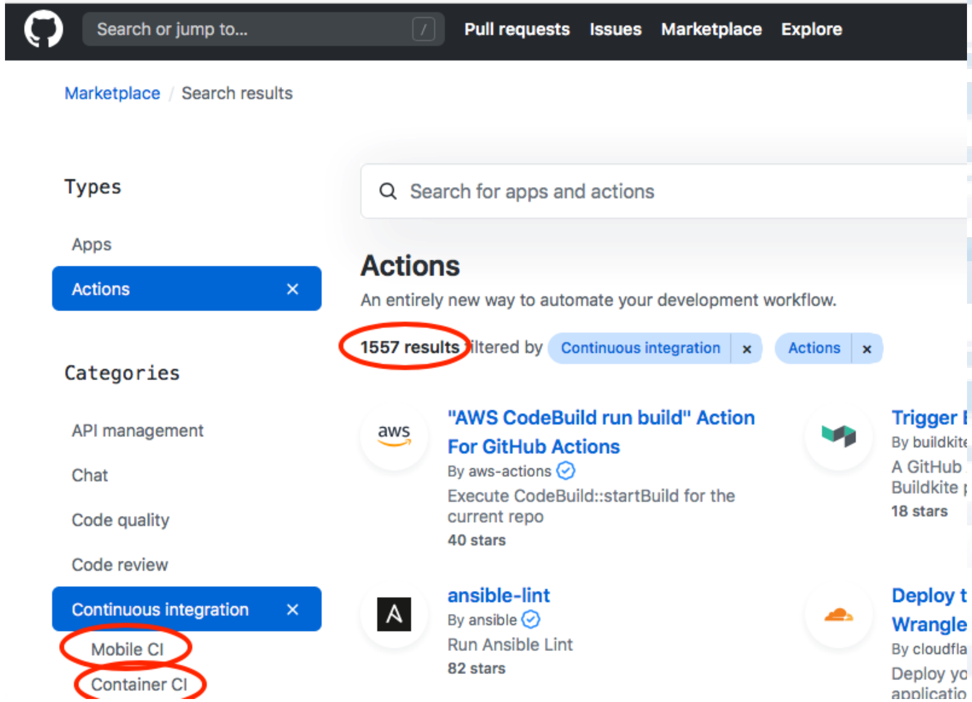
- Just getting started
  - Many technologies/choices; often in the ”cloud”
  - Solution: start small, simple, build up
- Developing suitable tests
  - Many project’s existing tests not suitable for CI
  - CI testing is a balance of thoroughness and responsiveness
  - Solution: Simplify/refactor and/or sub-setting test suite
- Ensuring sufficient coverage
  - Some changes to code never get tested – CI can provide a false sense of security
  - Solution: tools to measure it, enforce always increasing

Advanced situations

- Defining failure for *many* configurations
  - Bit-for-bit (exact) match vs. fuzzy match
  - Solution: absolute/relative tolerances → AI/ML
- Numerous 3rd party libraries (TPLs)
  - Compiling takes too long
  - Solution: cache pre-built TPLs, containers
- Performance testing
  - Avoid time-, space-, scaling-performance degradation
  - Solution: Perf. instrumentation and *scheduled* testing

CI Resources (Where do jobs run?)

- Free Cloud Resources (many free on GitHub, BitBucket, GitLab, etc.)
  - Travis-CI, Circle-CI, AppVeyor, Azure Pipelines,…
  - All launch a VM (Linux variants, Windows and OSX)
    - Constrained in time/size, hardware (e.g. GPU type/count)
    - Not a complete solution for many HPC/scientific codes, but a useful starting point.



Actions

An entirely new way to automate your development workflow.

1557 results filtered by Continuous integration × Actions ×

"AWS CodeBuild run build" Action For GitHub Actions
By aws-actions
Execute CodeBuild::startBuild for the current repo
40 stars

Trigger I
By buildkite
A GitHub
Buildkite
18 stars

ansible-lint
By ansible
Run Ansible Lint
82 stars

Deploy t
Wrangle
By cloudfla
Deploy yo
applicatio

- Site-local Resources
  - Group, department, institution, computing facility
  - Examples: Bamboo @ LLNL, Jenkins @ ANL, Travis+CDash @ NERSC, etc.
  - ECP Program: GitLab-CI @ ANL, LANL, LLNL, NERSC, ORNL, SNL
- Create your own by setting up resources/services

Getting started with CI

- What *configuration* is most important?
  - Examples: gcc, icc, xlc? MPI-2 or MPI-3? Python 2, 3 or 2 & 3?
- What *functionality* is most important?
  - Examples: vanilla numerical kernels? OpenMP kernels? GPU kernels? All of these?
- Good candidates…
  - A “hello world” example for your project
  - Once you’ve got the basics working, its easy to build up from there

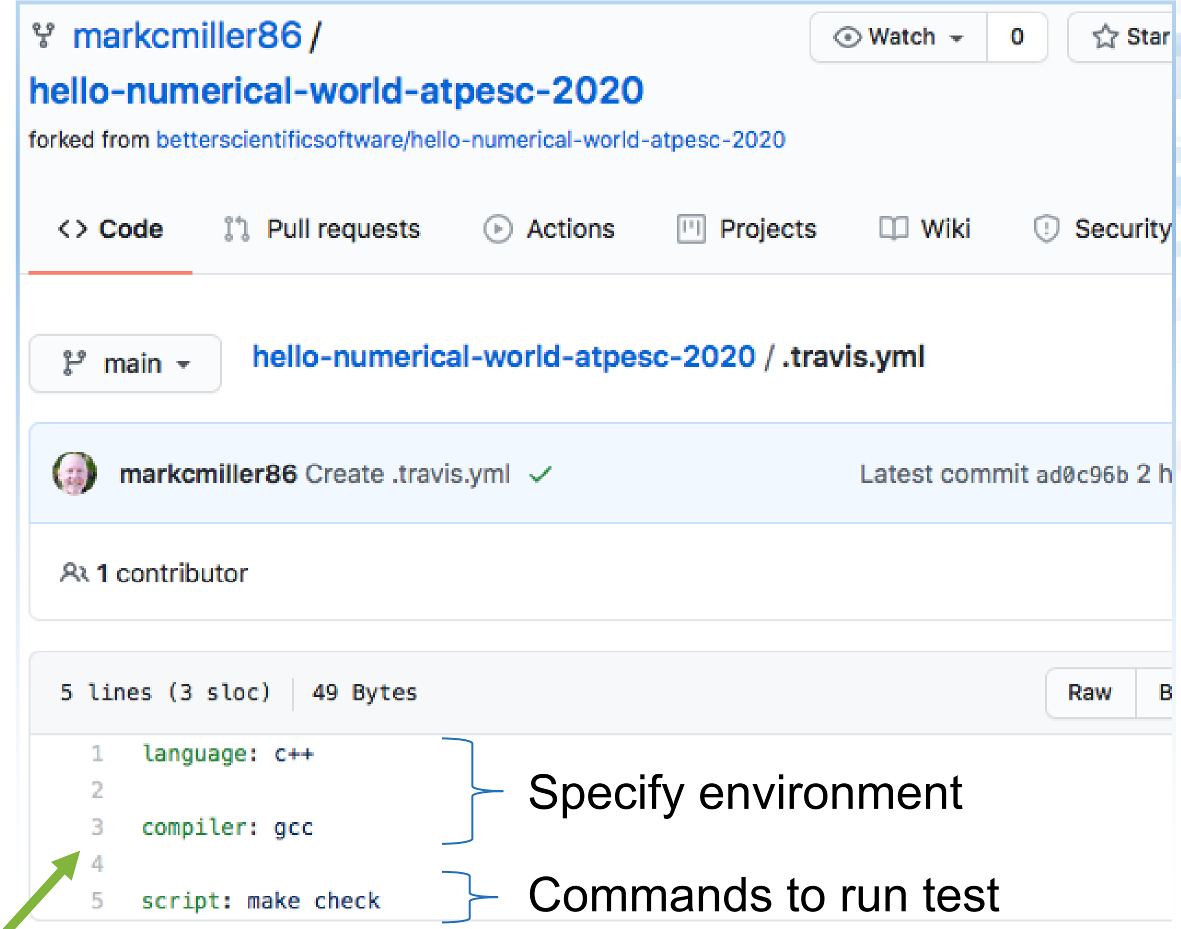
Getting started with CI:

Setting up CI

Service	Interface	
Travis	repo YAML file [& repo scripts]	/.travis.yml in root of repo
GitLab	Web page configurator + repo YAML file [& repo scripts]	/.gitlab-ci.yml in root of repo
Bamboo	Web page configurator + repo scripts	
. . .		

Keywords defined by service provider's YAML docs

Example .travis.yml file (also doing coverage analysis)

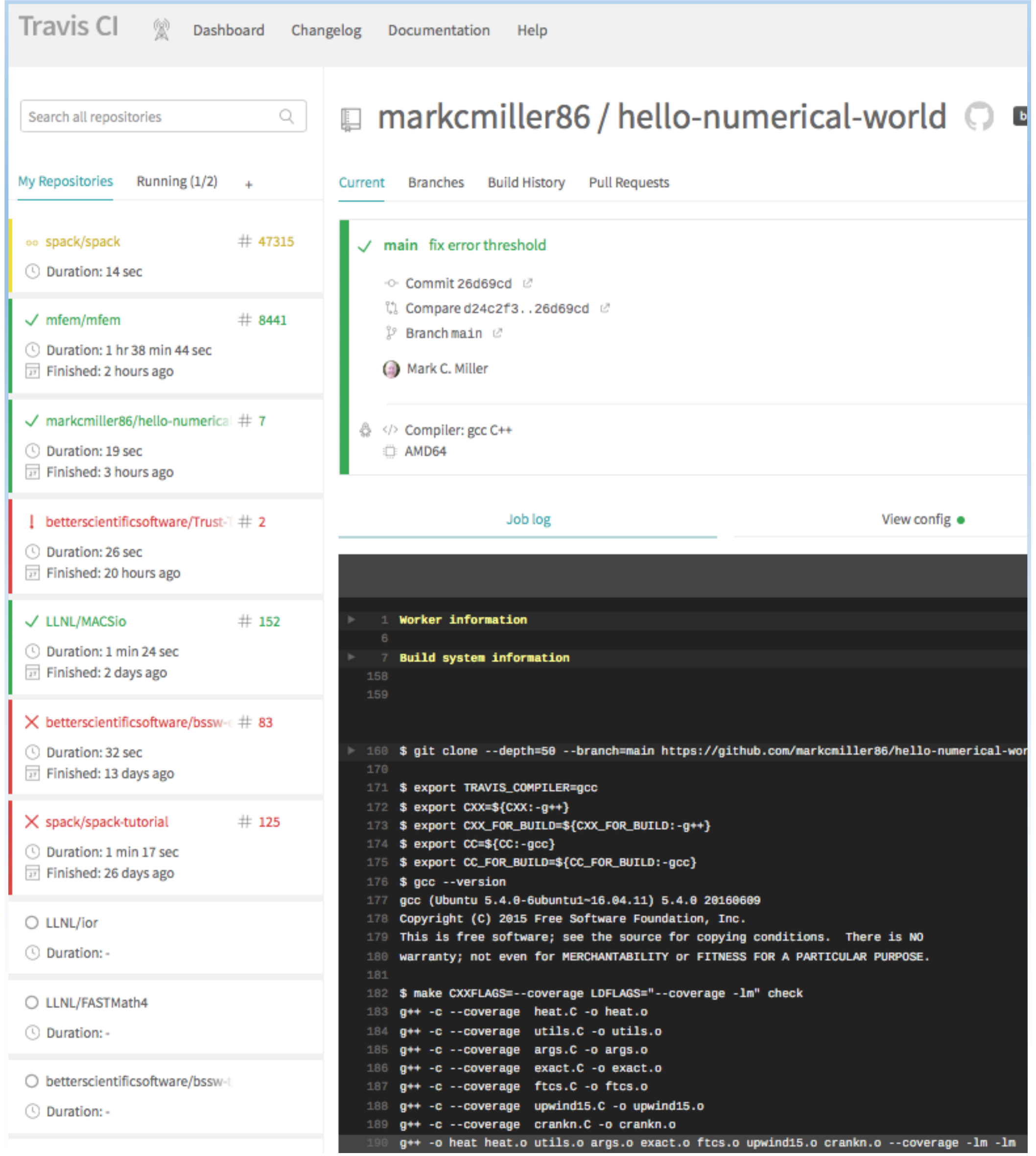

markcmiller86 / hello-numerical-world-atpesc-2020
forked from betterscientificsoftware/hello-numerical-world-atpesc-2020
Watch 0
Star
Code
Pull requests
Actions
Projects
Wiki
Security
main
hello-numerical-world-atpesc-2020 / .travis.yml
markcmiller86 Create .travis.yml ✓
Latest commit ad0c96b 2 h
1 contributor
5 lines (3 sloc) 49 Bytes
Raw
B

```
language: c++

compiler: gcc

script: make check
```

Specify environment

Commands to run test

travis-ci.com



Travis CI Dashboard Changelog Documentation Help

Search all repositories

My Repositories Running (1/2) +

- spack/spack # 47315
  Duration: 14 sec
- ✓ mfem/mfem # 8441
  Duration: 1 hr 38 min 44 sec
  Finished: 2 hours ago
- ✓ markcmiller86/hello-numerica # 7
  Duration: 19 sec
  Finished: 3 hours ago
- ! betterscientificsoftware/Trust- # 2
  Duration: 26 sec
  Finished: 20 hours ago
- ✓ LLNL/MACSio # 152
  Duration: 1 min 24 sec
  Finished: 2 days ago
- ✗ betterscientificsoftware/bssw- # 83
  Duration: 32 sec
  Finished: 13 days ago
- ✗ spack/spack-tutorial # 125
  Duration: 1 min 17 sec
  Finished: 26 days ago
- ○ LLNL/ior
  Duration: -
- ○ LLNL/FASTMath4
  Duration: -
- ○ betterscientificsoftware/bssw-
  Duration: -

markcmiller86 / hello-numerical-world

Current Branches Build History Pull Requests

✓ main fix error threshold

Commit 26d69cd
Compare d24c2f3..26d69cd
Branch main

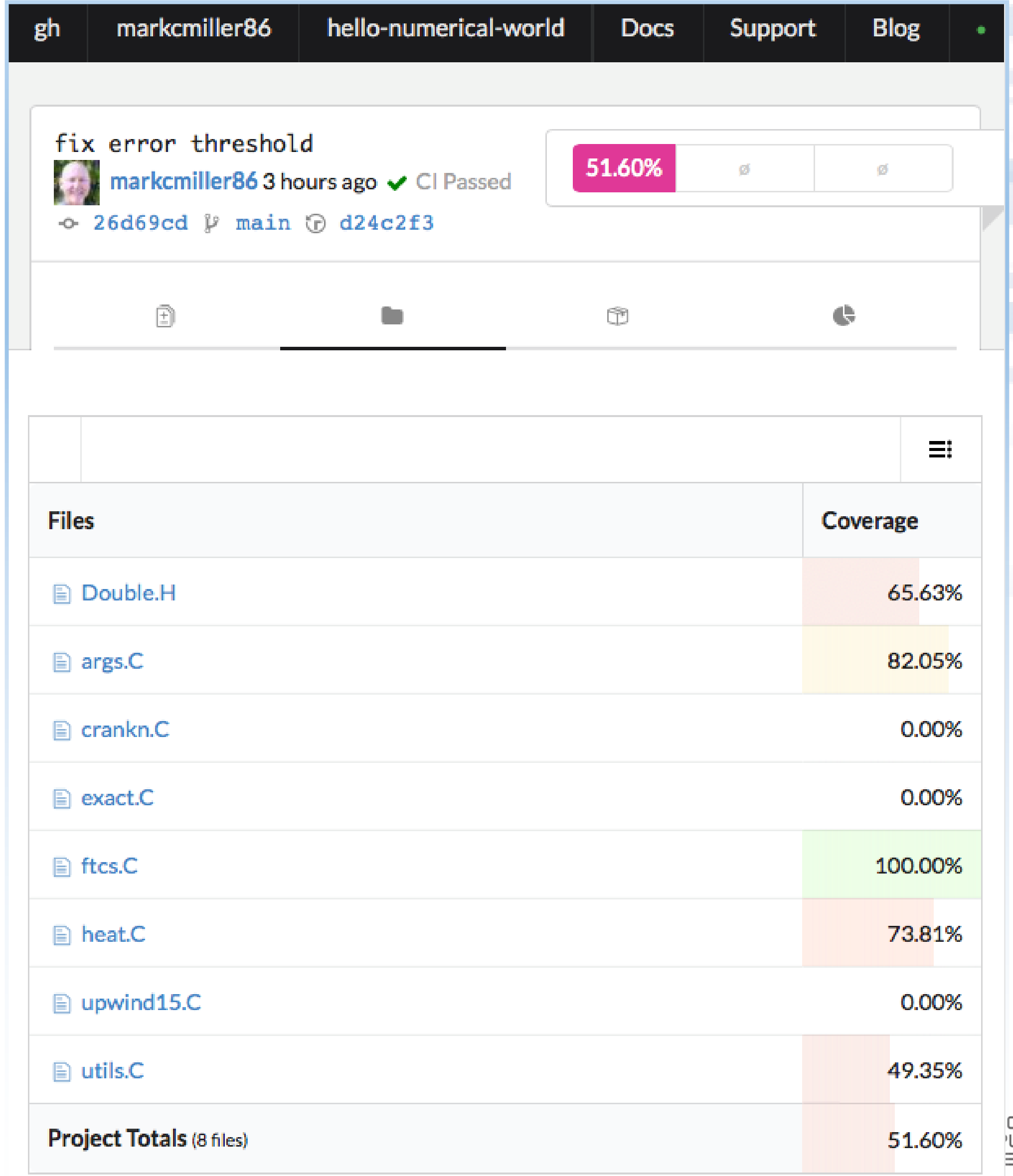
Mark C. Miller

Compiler: gcc C++
AMD64

Job log View config

```
Worker information

Build system information



$ git clone --depth=50 --branch=main https://github.com/markcmiller86/hello-numerical-wor

$ export TRAVIS_COMPILER=gcc
$ export CXX=${CXX:-g++}
$ export CXX_FOR_BUILD=${CXX_FOR_BUILD:-g++}
$ export CC=${CC:-gcc}
$ export CC_FOR_BUILD=${CC_FOR_BUILD:-gcc}
$ gcc --version
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.11) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

$ make CXXFLAGS=--coverage LDFLAGS="--coverage -lm" check
g++ -c --coverage  heat.C -o heat.o
g++ -c --coverage  utils.C -o utils.o
g++ -c --coverage  args.C -o args.o
g++ -c --coverage  exact.C -o exact.o
g++ -c --coverage  ftcs.C -o ftcs.o
g++ -c --coverage  upwind15.C -o upwind15.o
g++ -c --coverage  crankn.C -o crankn.o
g++ -o heat heat.o utils.o args.o exact.o ftcs.o upwind15.o crankn.o --coverage -lm -lm
```

codecov.io



gh markcmiller86 hello-numerical-world Docs Support Blog

fix error threshold

markcmiller86 3 hours ago ✓ CI Passed

26d69cd main d24c2f3

51.60% ø ø

Files	Coverage
Double.H	65.63%
args.C	82.05%
crankn.C	0.00%
exact.C	0.00%
ftcs.C	100.00%
heat.C	73.81%
upwind15.C	0.00%
utils.C	49.35%
Project Totals (8 files)	51.60%

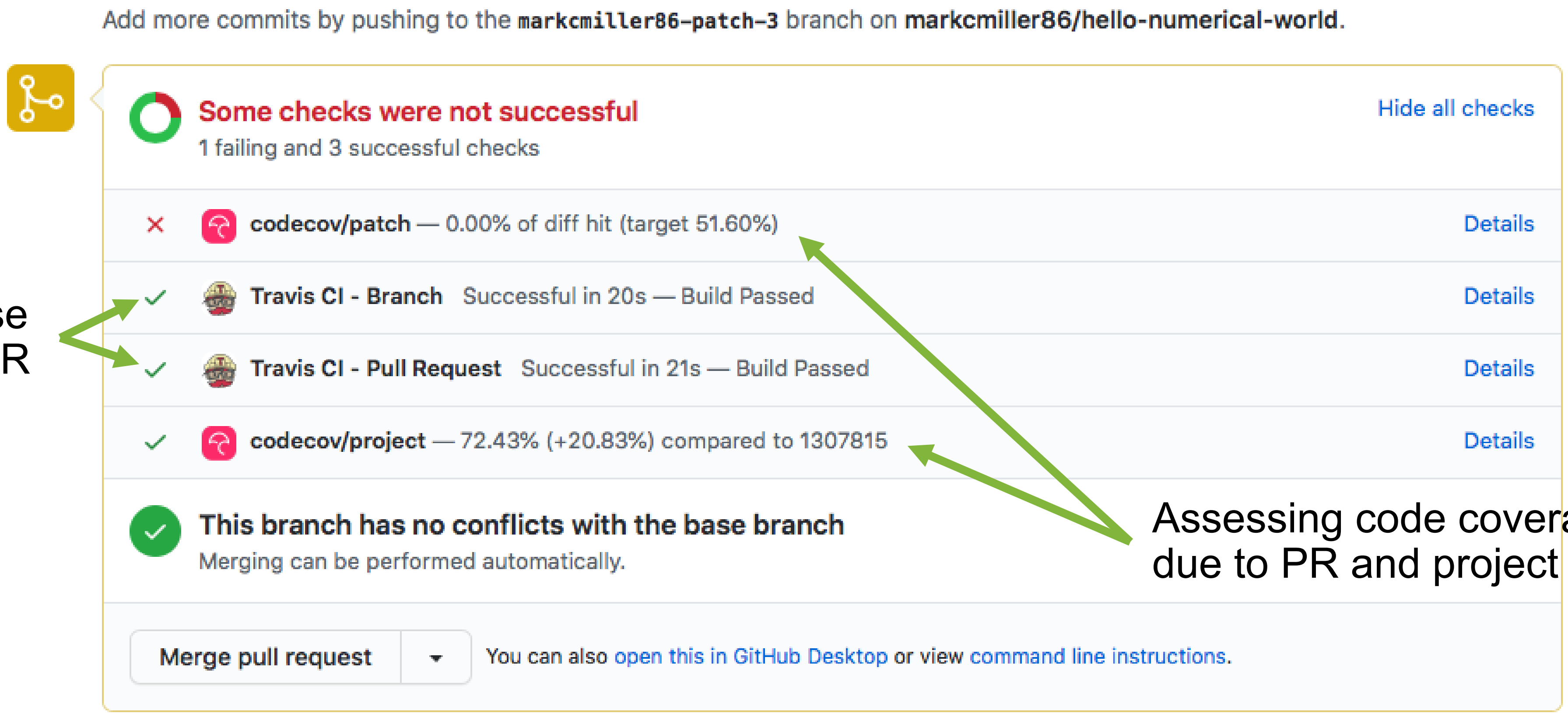
Homework Assignment

- See tutorial web site for details
  - https://betterscientificsoftware.github.io/bssw-tutorial-sc20/

1. Fork the repository
2. Configure Travis-CI to run ‘make check’ as a CI check
3. Add code coverage checking using Codecov.io
4. Expand the testing by using ‘make check_all’ instead of ‘make check’
5. Extra credit: make the CI check fail if the code coverage decreases

(Possible) Results of Homework*

https://github.com/betterscientificsoftware/hello-numerical-world-sc20
with testing via Travis CI and coverage analysis via Codecov.io



Add more commits by pushing to the **markcmiller86-patch-3** branch on **markcmiller86/hello-numerical-world**.

Some checks were not successful
1 failing and 3 successful checks

✗ **codecov/patch** — 0.00% of diff hit (target 51.60%) Details

✓ **Travis CI - Branch** Successful in 20s — Build Passed Details

✓ **Travis CI - Pull Request** Successful in 21s — Build Passed Details

✓ **codecov/project** — 72.43% (+20.83%) compared to 1307815 Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

Checking base branch and PR

Assessing code coverage changes due to PR and project overall

* Your development issues may vary!

Summary

- The purpose of Continuous Integration Testing is to identify problems early
  - Catch things that would “break the build” or adversely impact other developers
  - Need to provide sufficient confidence, but run quickly – balance varies by project
- CI testing should complement (not replace) more extensive automated “nightly” testing
  - Use scheduled testing for more and more detailed tests, more configurations and platforms, performance testing, etc.
- Many options for where to execute CI tests
  - Free services are a good (easy) place to start
  - But may not be sufficient in the long run (especially large HPC/scientific codes)
- Start simple to get automation working, then build out what you need
  - Focus initially on key software configurations and aspects of the code
  - Make sure your testing expands to cover new code