



# Continuous Integration



Mark C. Miller

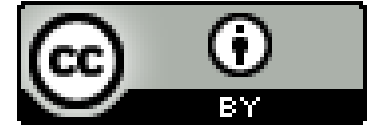
Lawrence Livermore National Laboratory

Software Productivity Track, ATPESC 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](https://creativecommons.org/licenses/by/4.0/) (CC BY 4.0).
- **The requested citation the overall tutorial is: David E. Bernholdt, Anshu Dubey, Mark C. Miller, Katherine M. Riley, and James M. Willenbring, Software Productivity Track, in Argonne Training Program for Extreme Scale Computing (ATPESC), August 2020, online. DOI: [10.6084/m9.figshare.12719834](https://doi.org/10.6084/m9.figshare.12719834)**
- Individual modules may be cited as *Speaker, Module Title*, in Software Productivity Track...

## Acknowledgements

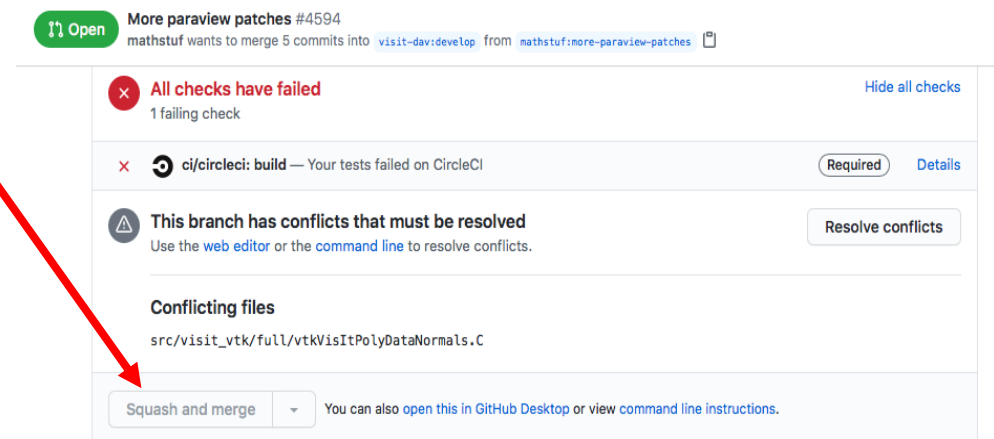
- Additional contributors include: Patricia Grubel, Rinku Gupta, Mike Heroux, Alicia Klinvex, Jared O'Neal, David Rogers, Deborah Stevens
- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing 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. IM Release #LLNL-PRES-813357
- 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`
  - 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
  - Potential issues: extreme automation, test granularity, coverage, 3<sup>rd</sup>-party services/resources



# Examples...

## Automated Nightly Testing Dashboard Lives “next to” your development work


### Results of Visit Regression Test ( pascal,trunk,serial )

Test suite run started at 2020:07:09:22:49:46.  
(Click on table header to sort)

Index	Category	Test File	Status	Runtime (sec)
243	rendering	ospray.py	Unacceptable	5.0
273	simulation	batch.py	Unacceptable	38.0
24	databases	chgcarr.py	Succeeded With Skips	11.0
32	databases	exodus.py	Succeeded With Skips	14.0
66	databases	silos.py	Succeeded With Skips	50.0
67	databases	silos_altdriver.py	Succeeded With Skips	87.0
75	databases	xdmf.py	Succeeded With Skips	14.0
109	hybrid	merge_tree.py	Succeeded With Skips	11.0
136	meshtype	emptydomains.py	Succeeded With Skips	7.0
256	rendering	view.py	Succeeded With Skips	17.0
275	simulation	curve.py	Succeeded With Skips	8.0
281	simulation	life.py	Succeeded With Skips	8.0
296	simulation	zerocopy.py	Succeeded With Skips	32.0
0	databases	ANALYZE.py	Succeeded	10.0
1	databases	ANSYS.py	Succeeded	9.0
2	databases	CGNS.py	Succeeded	11.0
3	databases	Cale.py	Succeeded	6.0
4	databases	Chombo.py	Succeeded	7.0
5	databases	EnSight.py	Succeeded	9.0
6	databases	FITS.py	Succeeded	8.0
7	databases	Fluent.py	Succeeded	7.0
8	databases	GDAL.py	Succeeded	20.0
9	databases	NASTRAN.py	Succeeded	15.0


## CI Testing Lives embedded in your development work


Add more commits by pushing to the `exodus-patch-1` branch on `exodus/chromium-dashboard`.



✓ All checks have passed [Hide all checks](#)

2 successful checks


✓  **Lighthouse** — Passed. New Lighthouse score would be 100/100. [Details](#)

✓  **continuous-integration/travis-ci/pr** — The Travis CI build passed [Details](#)

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

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

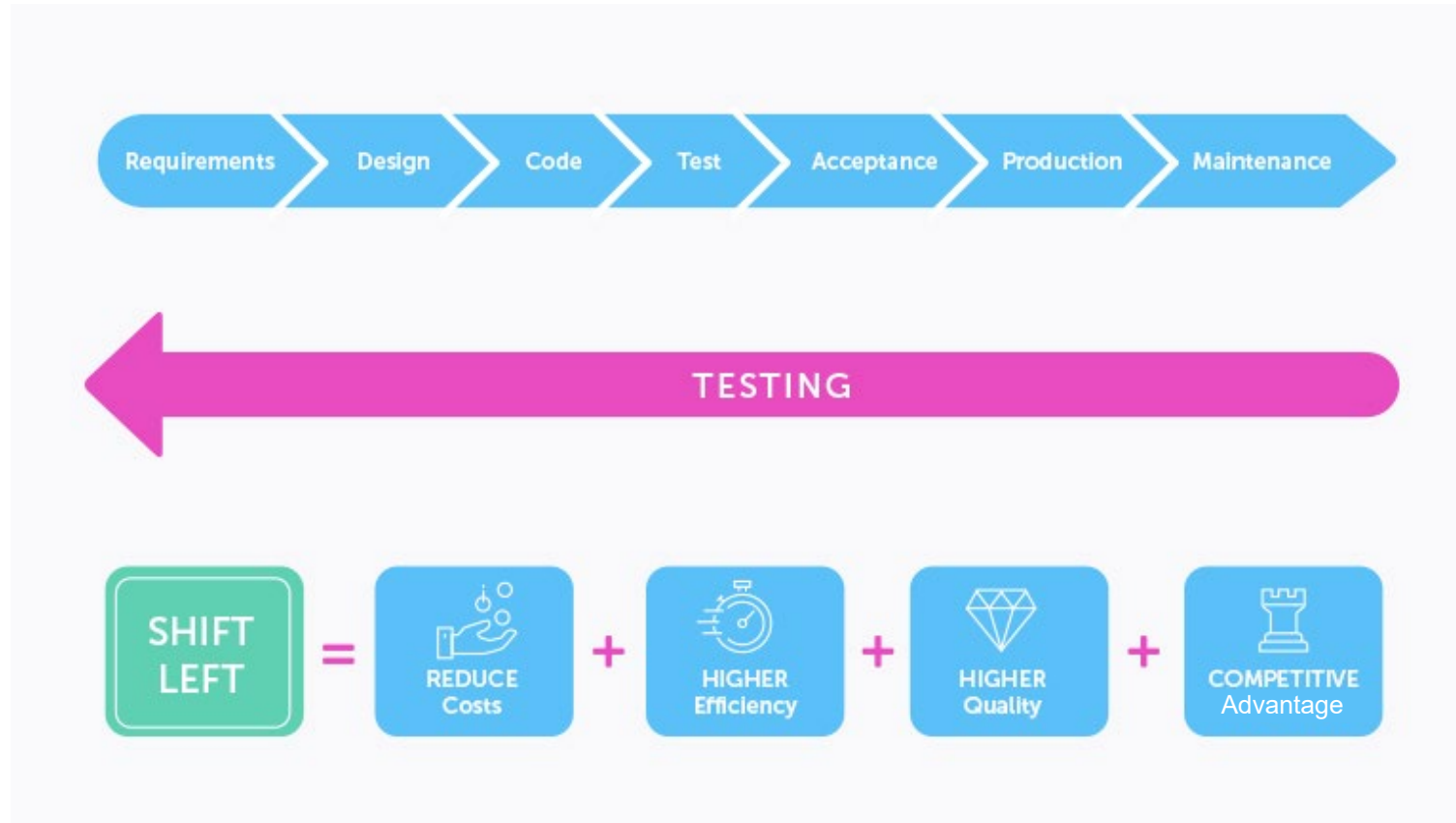


Write

Preview

AA B i “ < > ⌂ ⋮ ≡ ≡ ↶ @ 📌

# CI Testing is one part of the “Shift Left” movement in DevOps



# What can make CI Difficult

## Common situations

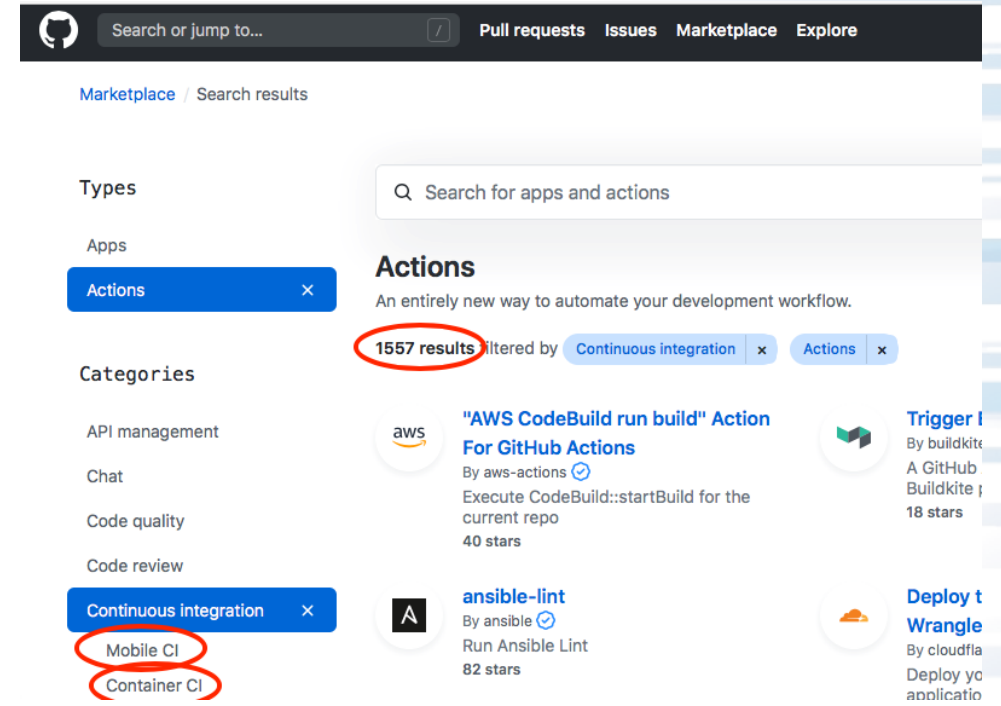
- Just getting started
  - Many technologies/choices; often in the "cloud"
  - Solution: start small, simple, build up
- Developing suitable tests
  - Many project's existing tests not suitable for CI
  - Solution: Simplify/refactor and/or sub-setting test suite
- Ensuring sufficient coverage
  - Some changes to code never get tested
  - 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 3<sup>rd</sup> 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, config. (e.g. GPU type/count)
    - Not always suitable for large, HPC projects due to need for longer than usual time to run
- Site-local Resources
  - 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

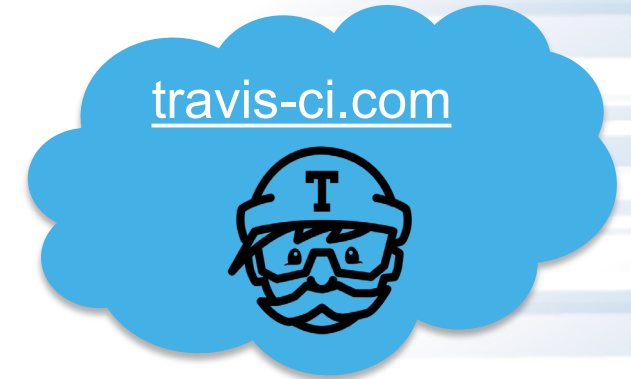




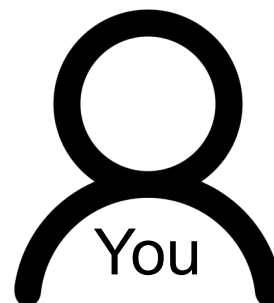
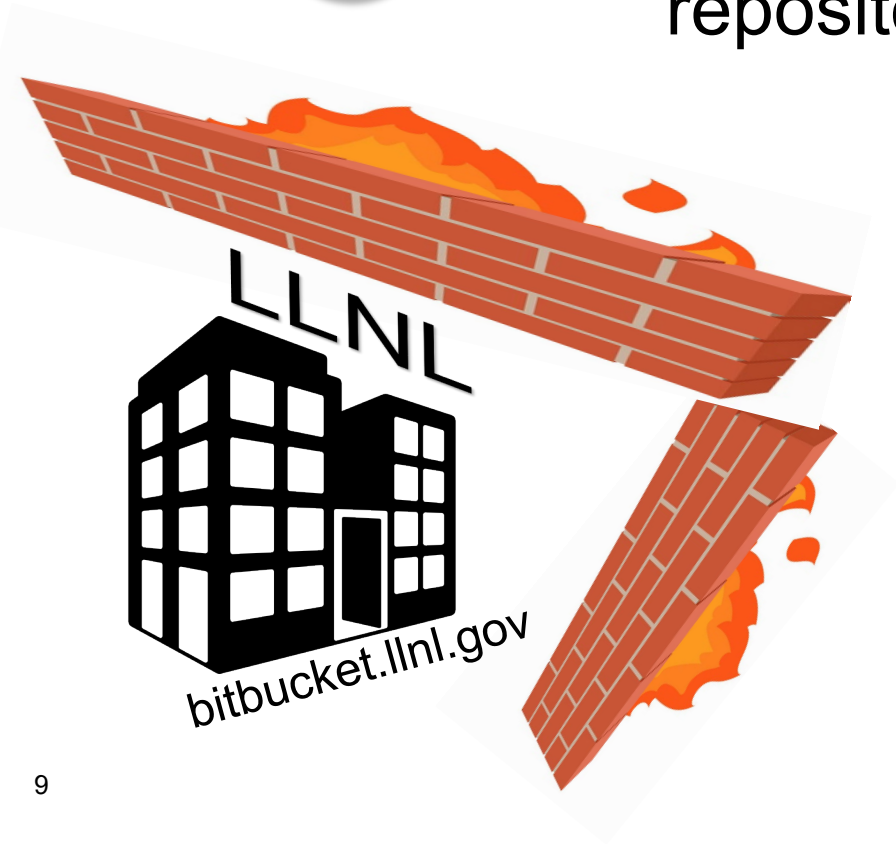
# Examples...



Your code repository

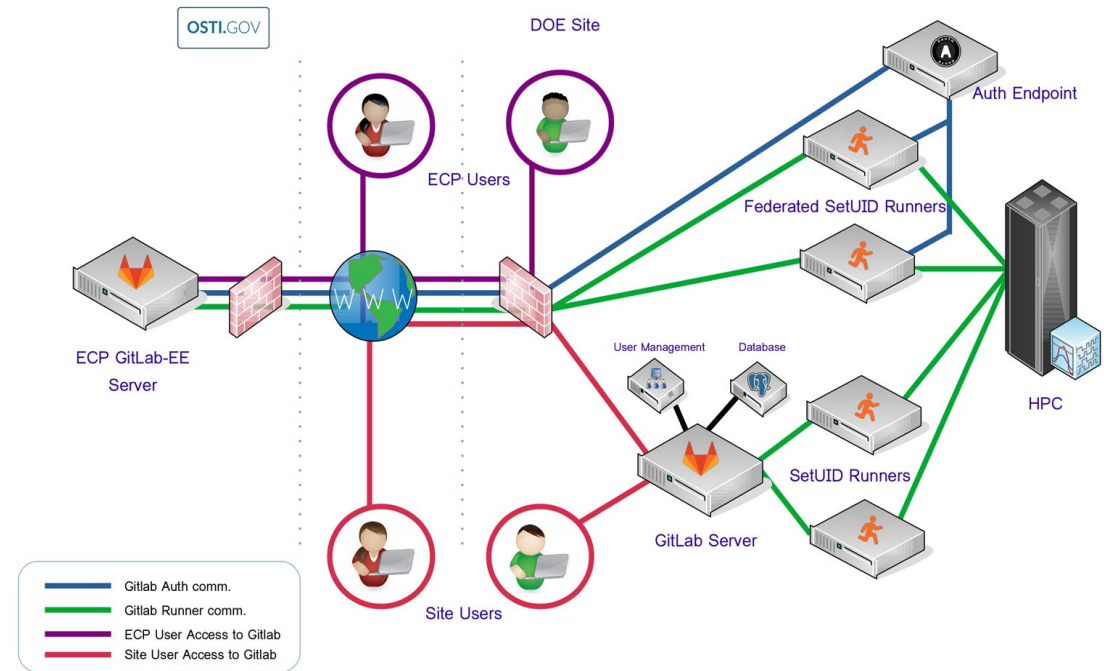


Your CI Resources



# ECP CI Resources

- ECP investing in GitLab for complex-wide CI
- Non-GitLab projects *mirror* into GitLab
- Complex-wide Federation via OSTI
  - Many hurdles still to overcome
  - Manual federation possible...but non-trivial
- Documentation and on-boarding help
  - <https://ecp-ci.gitlab.io>
  - email me, [miller86@llnl.gov](mailto:miller86@llnl.gov) for on-boarding contacts





# 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










# https://github.com/betterscientificsoftware/hello-numerical-world

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

[Hide all checks](#)

	 <b>codecov/patch</b> — 0.00% of diff hit (target 51.60%)	<a href="#">Details</a>
	 <b>Travis CI - Branch</b> Successful in 20s — Build Passed	<a href="#">Details</a>
	 <b>Travis CI - Pull Request</b> Successful in 21s — Build Passed	<a href="#">Details</a>
	 <b>codecov/project</b> — 72.43% (+20.83%) compared to 1307815	<a href="#">Details</a>
	<b>This branch has no conflicts with the base branch</b> Merging can be performed automatically.	

Merge pull request

▼

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# 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	
.		
.		
.		

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

```
1  language: c++
2
3  compiler:
4    - gcc
5
6  script:
7    - make CXXFLAGS=--coverage LDFLAGS="--coverage -lm" check_all
8
9  after_success:
10    - bash <(curl -s https://codecov.io/bash)
```

# Getting started with CI:

Keywords specific to service being used

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

Specify environment

Commands to run

```
1  language: c++
2
3  compiler:
4    - gcc
5
6  script:
7    - make CXXFLAGS=--coverage LDFLAGS="--coverage -lm" check
8
9  after_success:
10   - bash <(curl -s https://codecov.io/bash)
```

# travis-ci.com

# codecov.io

Travis CI

[Dashboard](#)[Changelog](#)[Documentation](#)[Help](#)

Search all repositories

markcmiller86 / hello-numerical-world

My RepositoriesRunning (1/2)

spack/spack# 47315

Duration: 14 sec

mfem/mfem# 8441

Duration: 1 hr 38 min 44 sec

Finished: 2 hours ago

markcmiller86/hello-numerical-world# 7

Duration: 19 sec

Finished: 3 hours ago

beterscientificsoftware/Trust# 2

Duration: 26 sec

Finished: 20 hours ago

LLNL/MACSIo# 152

Duration: 1 min 24 sec

Finished: 2 days ago

beterscientificsoftware/bssw# 83

Duration: 32 sec

Finished: 13 days ago

spack/spack-tutorial# 125

Duration: 1 min 17 sec

Finished: 26 days ago

LLNL/lor

Duration: -

LLNL/FASTMath4

Duration: -

beterscientificsoftware/bssw

Duration: -

CurrentBranchesBuild HistoryPull Requests

main fix error threshold

Commit 26d69cd

Compare d24c2f3...26d69cd

Branch main

Mark C. Miller

Compiler: gcc C++

AMD64

Job logView config

```
1 Worker information
6
7 Build system information
158
159
160 $ git clone --depth=50 --branch=main https://github.com/markcmiller86/hello-numerical-world
170
171 $ export TRAVIS_COMPILER=gcc
172 $ export CXX=${CXX:-g++}
173 $ export CXX_FOR_BUILD=${CXX_FOR_BUILD:-g++}
174 $ export CC=${CC:-gcc}
175 $ export CC_FOR_BUILD=${CC_FOR_BUILD:-gcc}
176 $ gcc --version
177 gcc (Ubuntu 5.4.0-6ubuntu1-16.04.11) 5.4.0 20160609
178 Copyright (C) 2015 Free Software Foundation, Inc.
179 This is free software; see the source for copying conditions. There is NO
180 warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
181
182 $ make CXXFLAGS=-coverage LDFLAGS=-coverage -lm check
183 g++ -c --coverage heat.C -o heat.o
184 g++ -c --coverage utils.C -o utils.o
185 g++ -c --coverage args.C -o args.o
186 g++ -c --coverage exact.C -o exact.o
187 g++ -c --coverage ftcs.C -o ftcs.o
188 g++ -c --coverage upwind15.C -o upwind15.o
189 g++ -c --coverage crankn.C -o crankn.o
190 g++ -o heat heat.o utils.o args.o exact.o ftcs.o upwind15.o crankn.o --coverage -lm -lm
```

ghmarkcmiller86hello-numerical-worldDocsSupportBlog

fix error threshold

markcmiller863 hours ago

CI Passed

26d69cdmaind24c2f3

51.60%

FilesCoverage

Double.H65.63%

args.C82.05%

crankn.C0.00%

exact.C0.00%

ftcs.C100.00%

heat.C73.81%

upwind15.C0.00%

utils.C49.35%

Project Totals (8 files)51.60%

# After Hours Hands-on Lesson – YouTube Video

- Follow QR code to [GitHub repository](#)
  - You can do this exercise entirely in your browser on GitHub
- Fork the repo
- Create .travis.yml using
- Submit Pull Request (PR)
- Increase coverage
  - Change 'check' to "check\_all"
- Update the PR and observe coverage change
- Extra credit...fail PR if coverage drops
  - Hint: read codecov.io docs

