Continuous Integration

David M. Rogers
Oak Ridge National Laboratory

Better Scientific Software tutorial, ISC, June 2021

Contributors: David E. Bernholdt (ORNL), Mark C. Miller (LLNL), David M. Rogers (ORNL), James M. Willenbring (SNL)

License, Citation and Acknowledgements

License and Citation

- **The requested citation the overall tutorial is: David E. Bernholdt, Anshu Dubey, Patricia A. Grubel, Rinku K. Gupta, and David M. Rogers, Better Scientific Software tutorial, in ISC High Performance, online, 2021. DOI: 10.6084/m9.figshare.14642520**
- Individual modules may be cited as *Speaker, Module Title*, in Better Scientific Software tutorial…

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.
- 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.

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

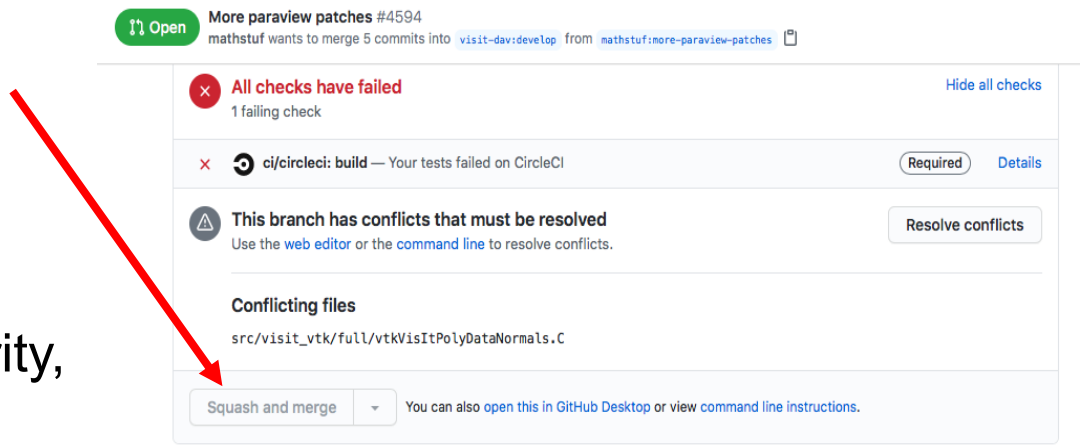
What is Continuous Integration (CI) *Testing*

- 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
    - Integration can take place at multiple levels
      - Individual project
      - Spack
      - E4S
  - 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
    - Can also be run post merge
  - Lives "within" your development workflow
  - Potential issues: extreme automation, test granularity, coverage, 3rd-party services/resources



Examples…

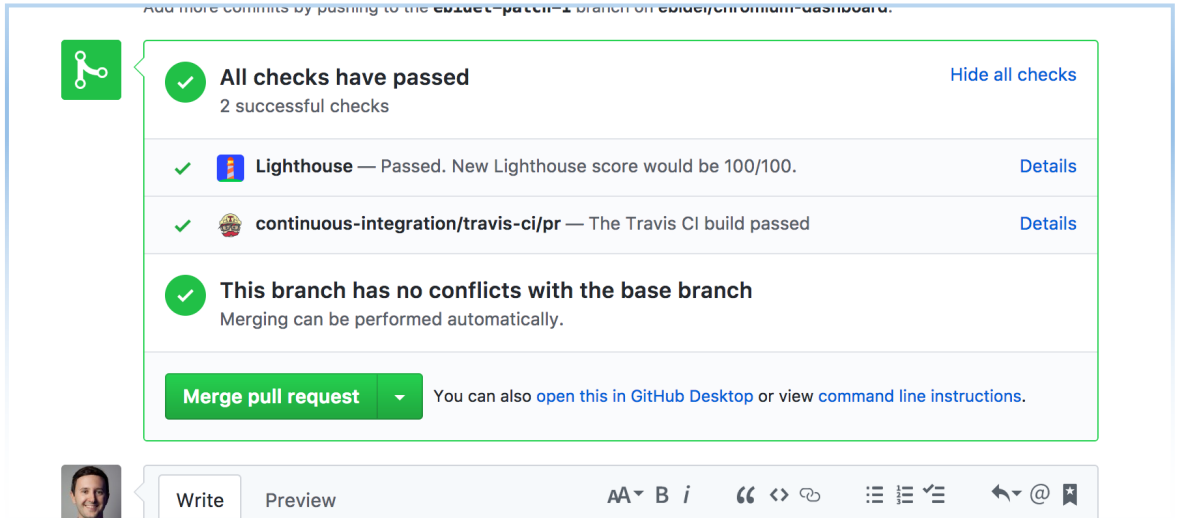
Automated Nightly Testing Dashboard
Lives "next to" your development work

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	chgcar.py	Succeeded With Skips	11.0
32	databases	exodus.py	Succeeded With Skips	14.0
66	databases	silo.py	Succeeded With Skips	50.0
67	databases	silo_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



IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

What can make CI Difficult

Common situations

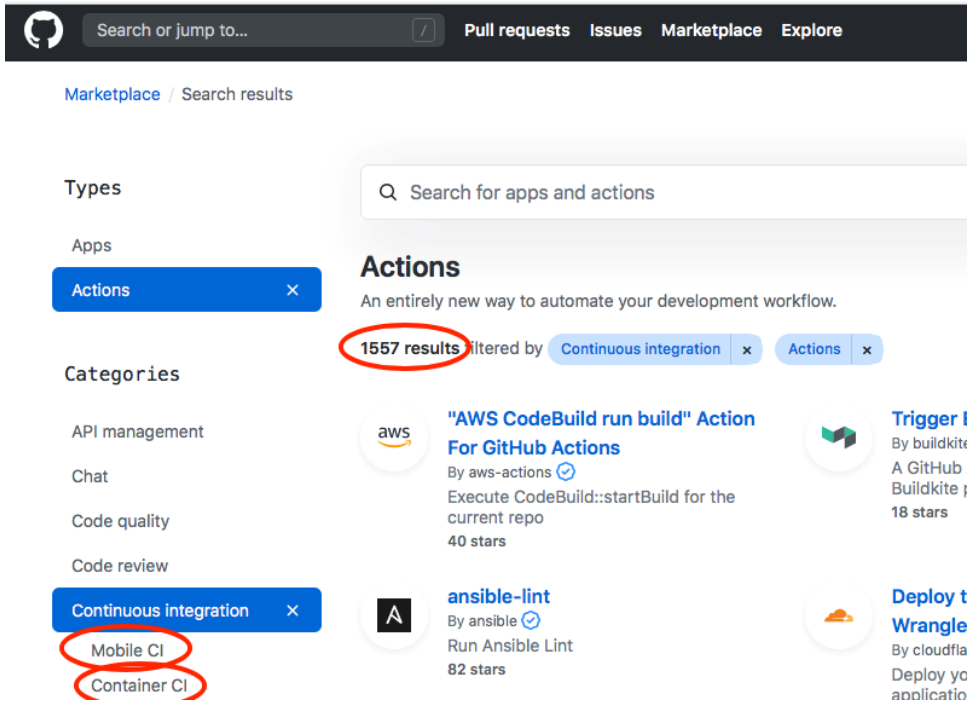
- Just getting started
  - Many technologies/choices; often in the ”cloud”
  - Solution: start small, simple, build up
- Developing suitable tests
  - Many project’s existing tests not suitable for CI
  - CI testing is a balance of thoroughness and responsiveness
  - Solution: Simplify/refactor and/or 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 / inconsistent failures
  - 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.



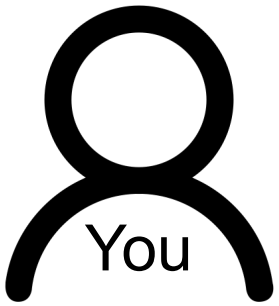
- Site-local Resources
  - Group, department, institution, computing facility
  - Examples: Bamboo @ LLNL, Jenkins @ ANL, Travis+CDash @ NERSC, etc.
  - ECP Program: GitLab-CI @ ANL, LANL, LLNL, NERSC, ORNL, SNL

- Create your own by setting up resources/services

Examples…

Your code repository

Your CI Resources



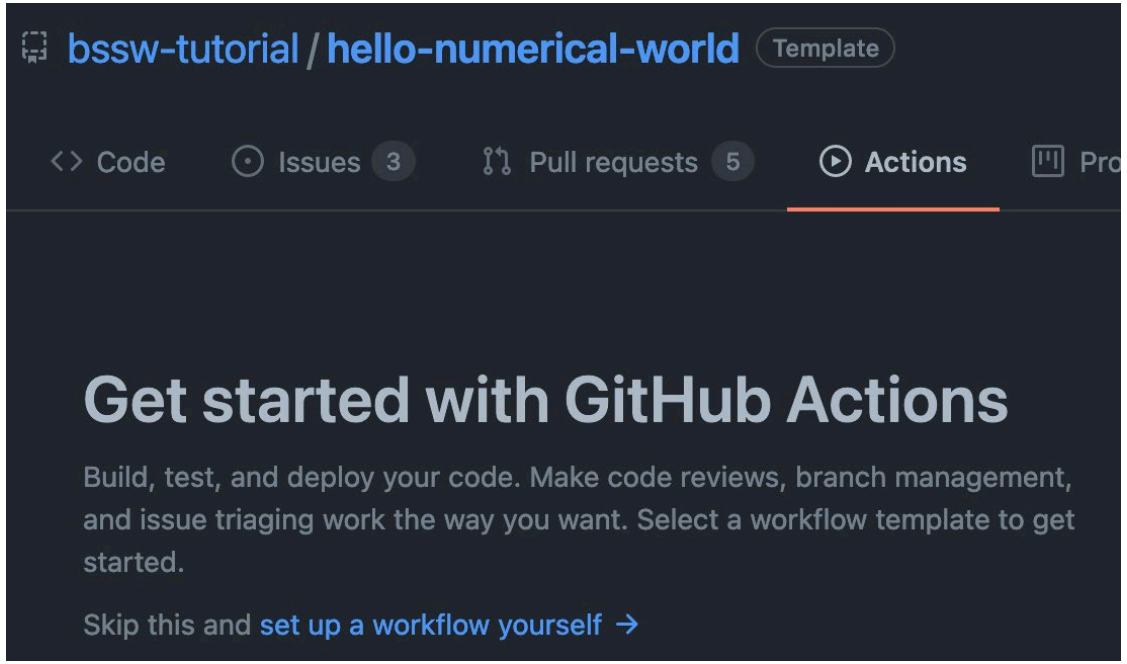
Getting started with CI

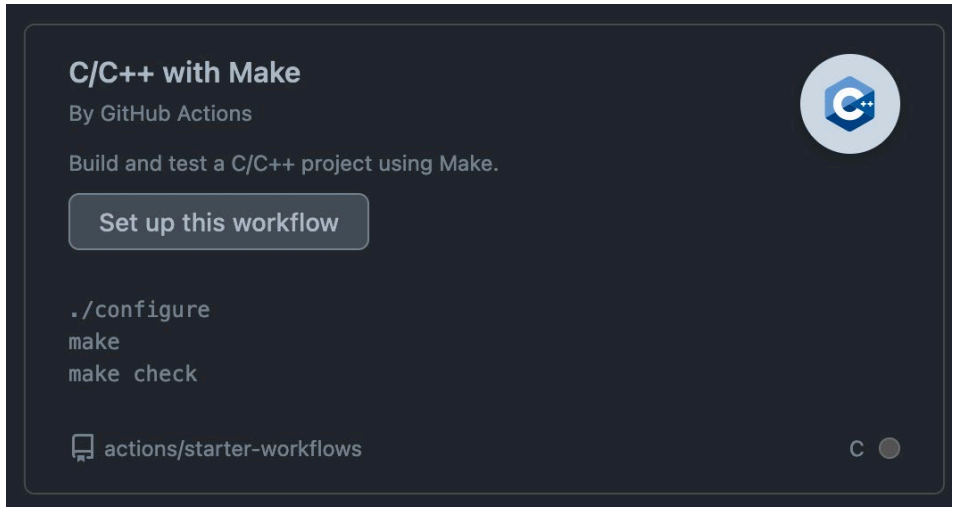
- What *configuration* is most important?
  - Examples: gcc, icc, xlc? MPI-2 or MPI-3? Python 2, 3 or 2 & 3?
- What *functionality* is most important?
  - Examples: vanilla numerical kernels? OpenMP kernels? GPU kernels? All of these?
- Good candidates…
  - A “hello world” example for your project
  - At a minimum, even just building the code can be a place to start!
  - 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	
GitHub Actions . .	Repo YAML file	.github/workflows/<test_name>.yml





Getting started with Github Actions:

19 lines (15 sloc) | 359 Bytes

```yaml
name: Check Results

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: make coverage
      run: make CXXFLAGS=--coverage LDFLAGS="--coverage -lm" check_all
    - name: upload coverage
      run: bash <(curl -s https://codecov.io/bash)
```

github.com

Code Issues Pull requests **Actions** Projects ...

Added workflows.

Check Results #1

build

succeeded 4 minutes ago in 7s

- Set up job 2s
- Run actions/checkout@v2 1s
- make coverage 3s

```
1  ▶ Run make CXXFLAGS=--coverage LDFLAGS="--coverage -lm"
   check_all
4  g++ -c --coverage  heat.C -o heat.o
5  g++ -c --coverage  utils.C -o utils.o
6  g++ -c --coverage  args.C -o args.o
7  g++ -c --coverage  exact.C -o exact.o
```

codecov.io

gh | markcmiller86 | hello-numerical-world | Docs | Support | Blog

fix error threshold

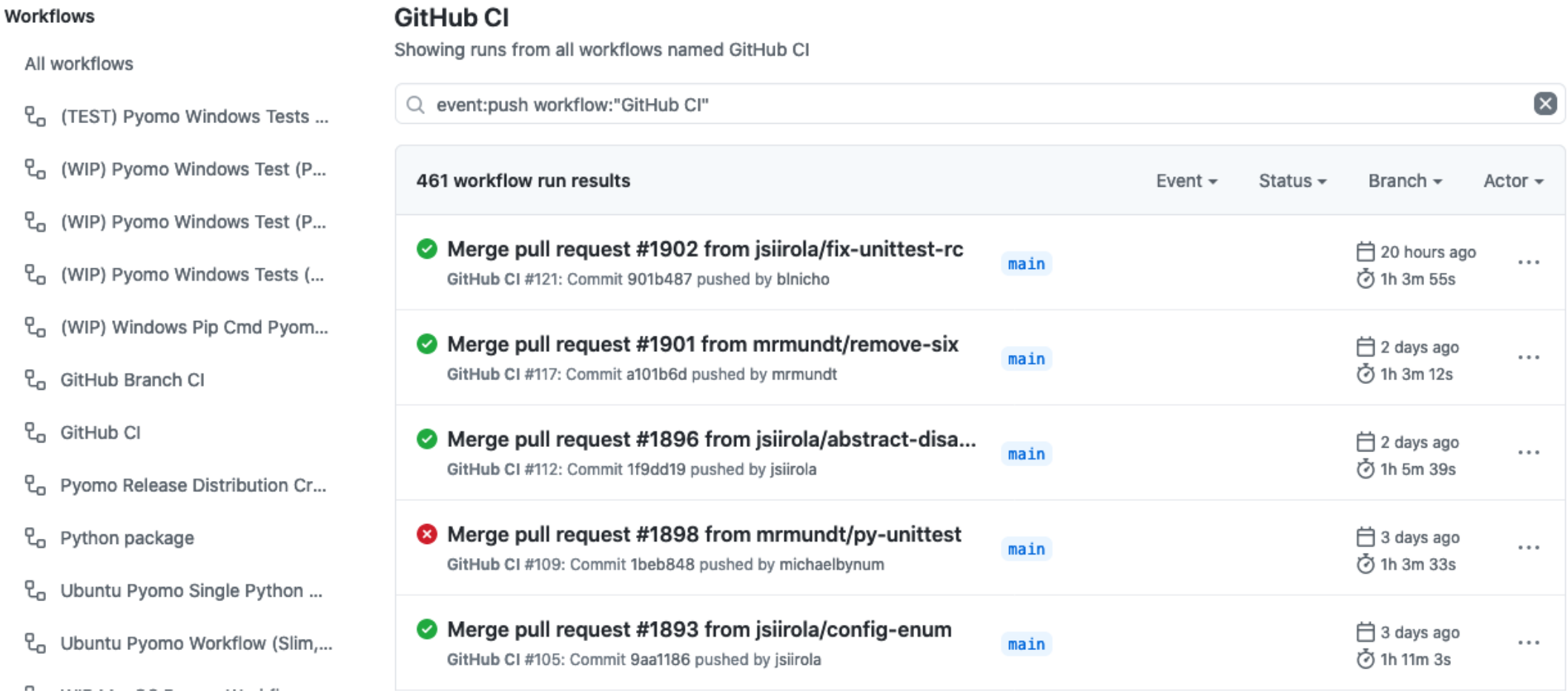
markcmiller86 3 hours ago ✔ CI Passed

26d69cd main d24c2f3

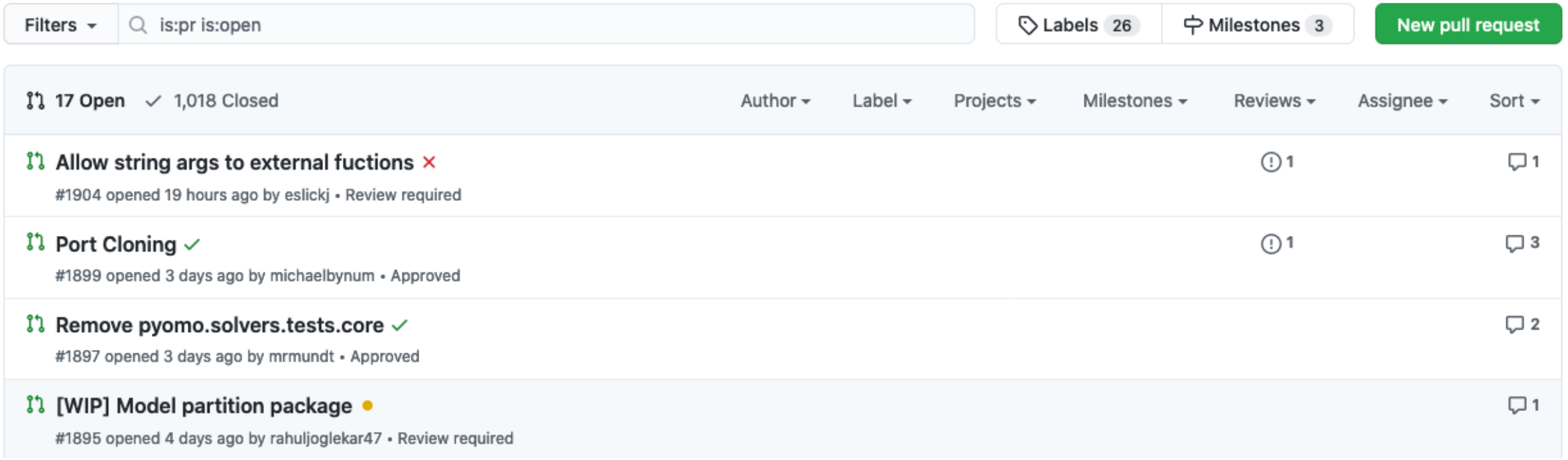
51.60%

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%

GitHub Actions – results of workflow test runs



GitHub Pull Request Status Indicators



GitHub Actions – Key Elements of Defining Tests

```yaml
name: GitHub CI

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main
  workflow_dispatch:
    inputs:
      git-ref:
        description: Git Hash (Optional)
        required: false

```

Name

Trigger

GitHub Actions – Key Elements of Defining Tests

```yaml
jobs:
  build:
    name: ${{ matrix.TARGET }}/${{ matrix.python }}${{ matrix.other }}
    runs-on: ${{ matrix.os }}
    timeout-minutes: 90
    strategy:
      fail-fast: false
      matrix:
        os: [ubuntu-18.04, macos-latest, windows-latest]
        python: [3.6, 3.7, 3.8, 3.9, pypy3]
        other: [""]
        category: ["nightly"]
        # Ubuntu-18.04 should be replaced with ubuntu-latest once PyNumero
        # build error is resolved:
        # https://github.com/Pyomo/pyomo/issues/1710

        include:
        - os: ubuntu-18.04
          TARGET: linux
          PYENV: pip

        - os: macos-latest
          TARGET: osx
          PYENV: pip
```

Job setup

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

GitHub Actions – Key Elements of Defining Tests

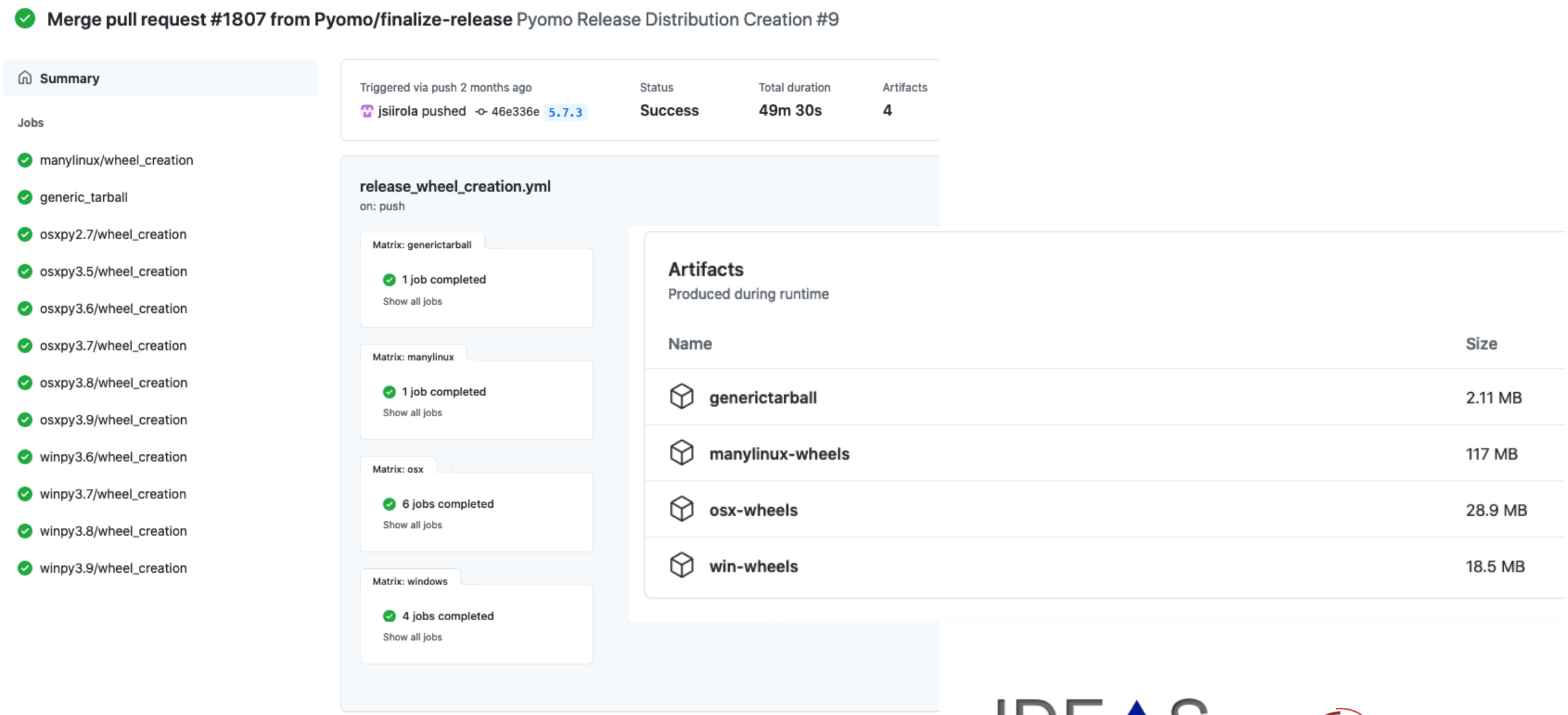
```
111     steps:
112     - name: Checkout Pyomo source
113       uses: actions/checkout@v2
114
115     - name: Configure job parameters
116       run: |
117         JOB="${{matrix.TARGET}}/${{matrix.python}}${{matrix.other}}"
118         echo "GHA_JOBNAME=$JOB" | sed 's|/|_|g' >> $GITHUB_ENV
119         if test -z "${{matrix.other}}"; then
120             echo "GHA_JOBGROUP=${{matrix.TARGET}}" >> $GITHUB_ENV
121         else
122             echo "GHA_JOBGROUP=other" >> $GITHUB_ENV
123         fi
124         # Note: pandas 1.0.3 causes gams 29.1.0 import to fail in python 3.8
125         PYTHON_PACKAGES="${PYTHON_REQUIRED_PKGS}"
126         if test -z "${{matrix.slim}}"; then
127             PYTHON_PACKAGES="$PYTHON_PACKAGES ${PYTHON_BASE_PKGS}"
128         fi
129         if [[ ${{matrix.python}} != pypy* && ! "${{matrix.slim}}" ]]; then
130             # NumPy and derivatives either don't build under pypy, or if
131             # they do, the builds take forever.
132             PYTHON_PACKAGES="$PYTHON_PACKAGES ${PYTHON_NUMPY_PKGS}"
133         fi
134         PYTHON_PACKAGES="$PYTHON_PACKAGES ${{matrix.PACKAGES}}"
135         echo "PYTHON_PACKAGES=$PYTHON_PACKAGES" \
136             | tr '\n' ' ' | sed 's/ \+/ /g' >> $GITHUB_ENV
```

Job steps

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

GitHub Actions - Release Automation



Many Alternatives, e.g. gitlab, circle-ci, and Travis CI:

10 lines (7 sloc) | 166 Bytes

```
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)
```

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