Continuous Integration

David M. Rogers
Oak Ridge National Laboratory

Software Productivity and Sustainability track, ATPESC 2021

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

U.S. DEPARTMENT OF ENERGY | Office of Science

National Nuclear Security Administration
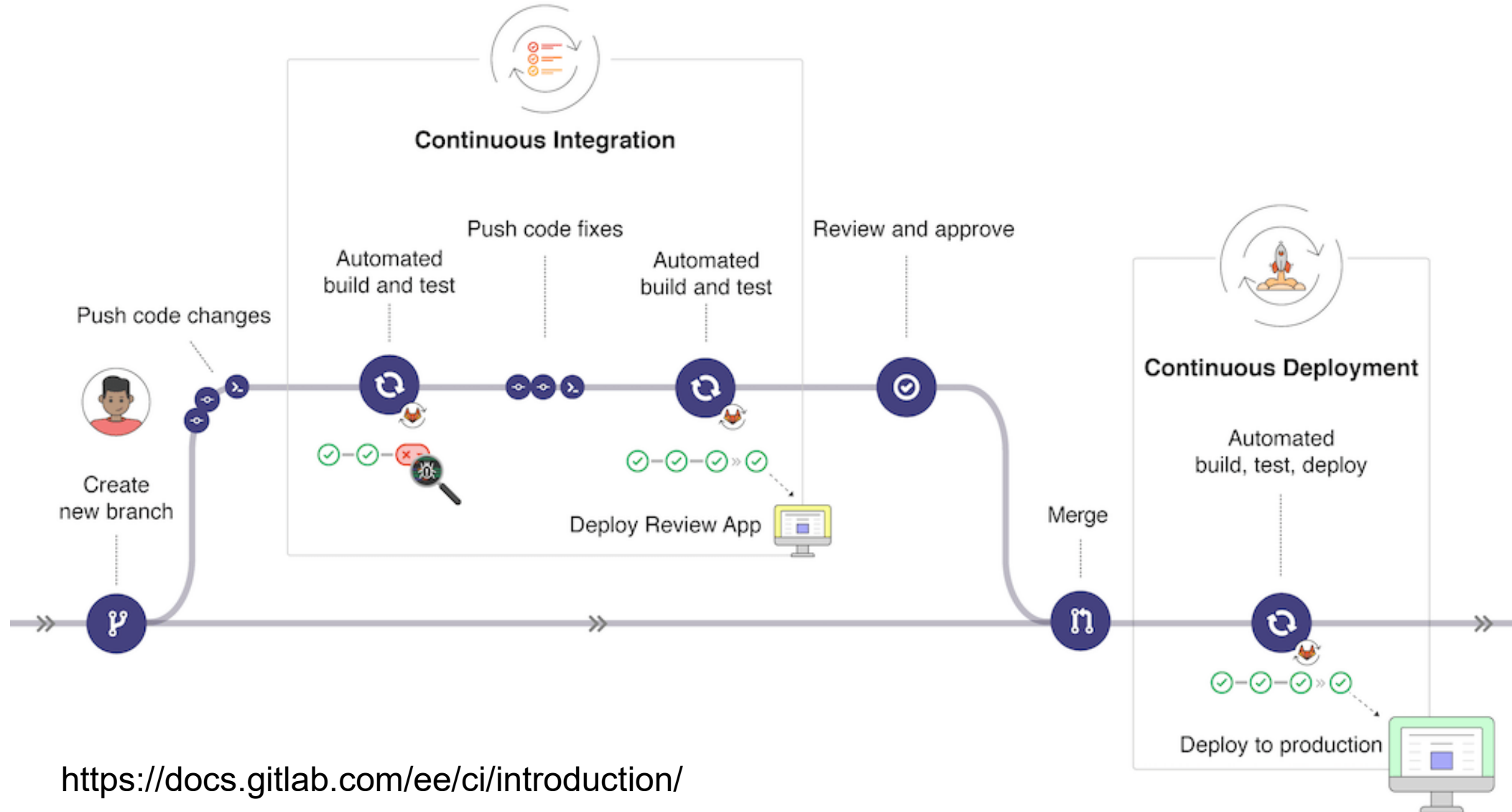
License, Citation and Acknowledgements

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, Rinku K. Gupta, and David M. Rogers, Software Productivity and Sustainability track, in Argonne Training Program on Extreme-Scale Computing (ATPESC), online, 2021. DOI: 10.6084/m9.figshare.15130590**
- Individual modules may be cited as *Speaker, Module Title*, in Better Scientific Software tutorial…

Acknowledgements

What is Continuous Integration (CI)

https://docs.gitlab.com/ee/ci/introduction/

3

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 <u>automation</u>

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'
  - May live "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
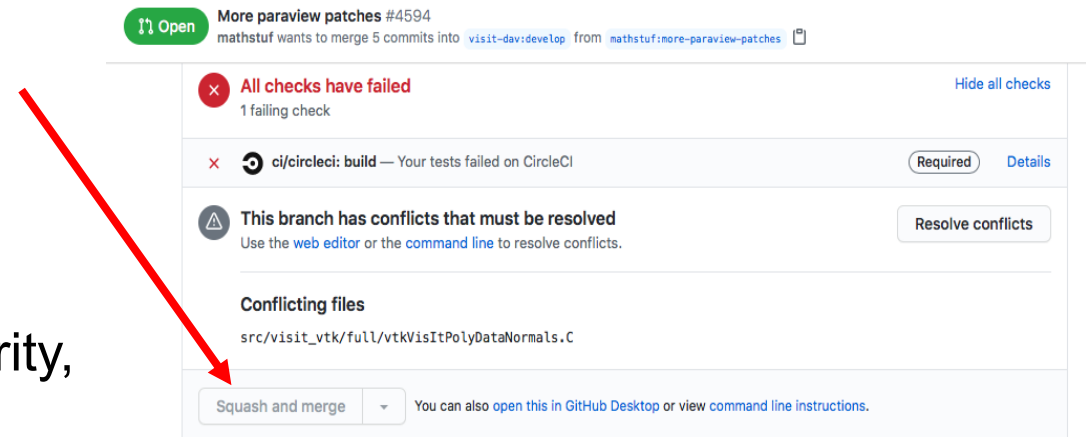
Examples…

Automated Nightly Testing Dashboard Lives "next to" your development work	CI Testing Lives embedded in 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

Add more commits by pushing to the **ebidel-patch-1** branch on **ebidel/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 |

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

What can make CI Difficult

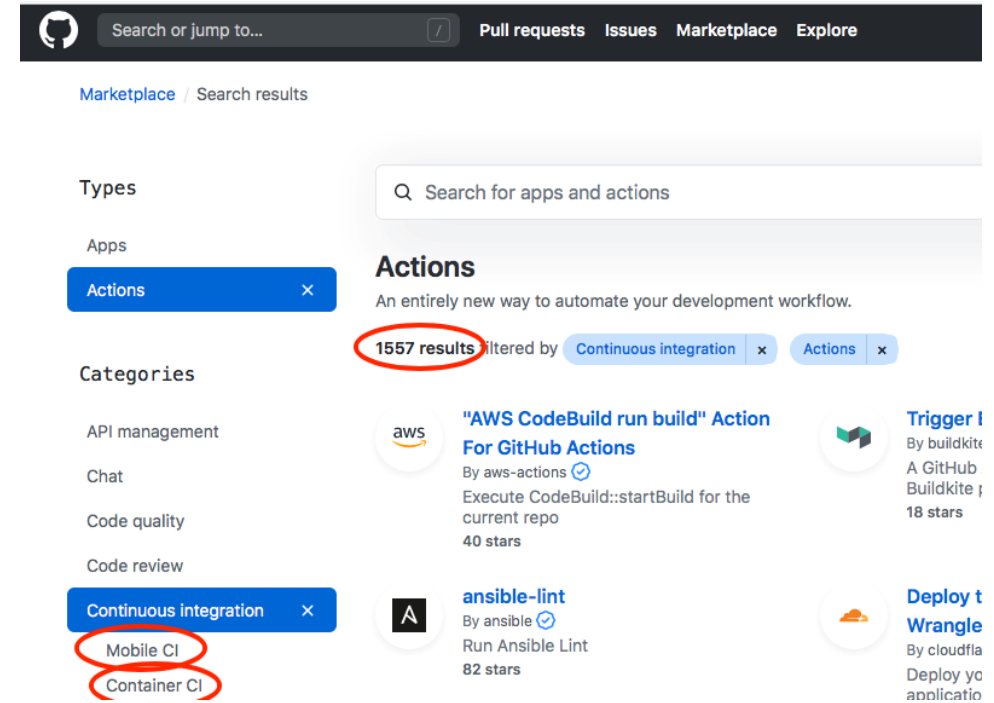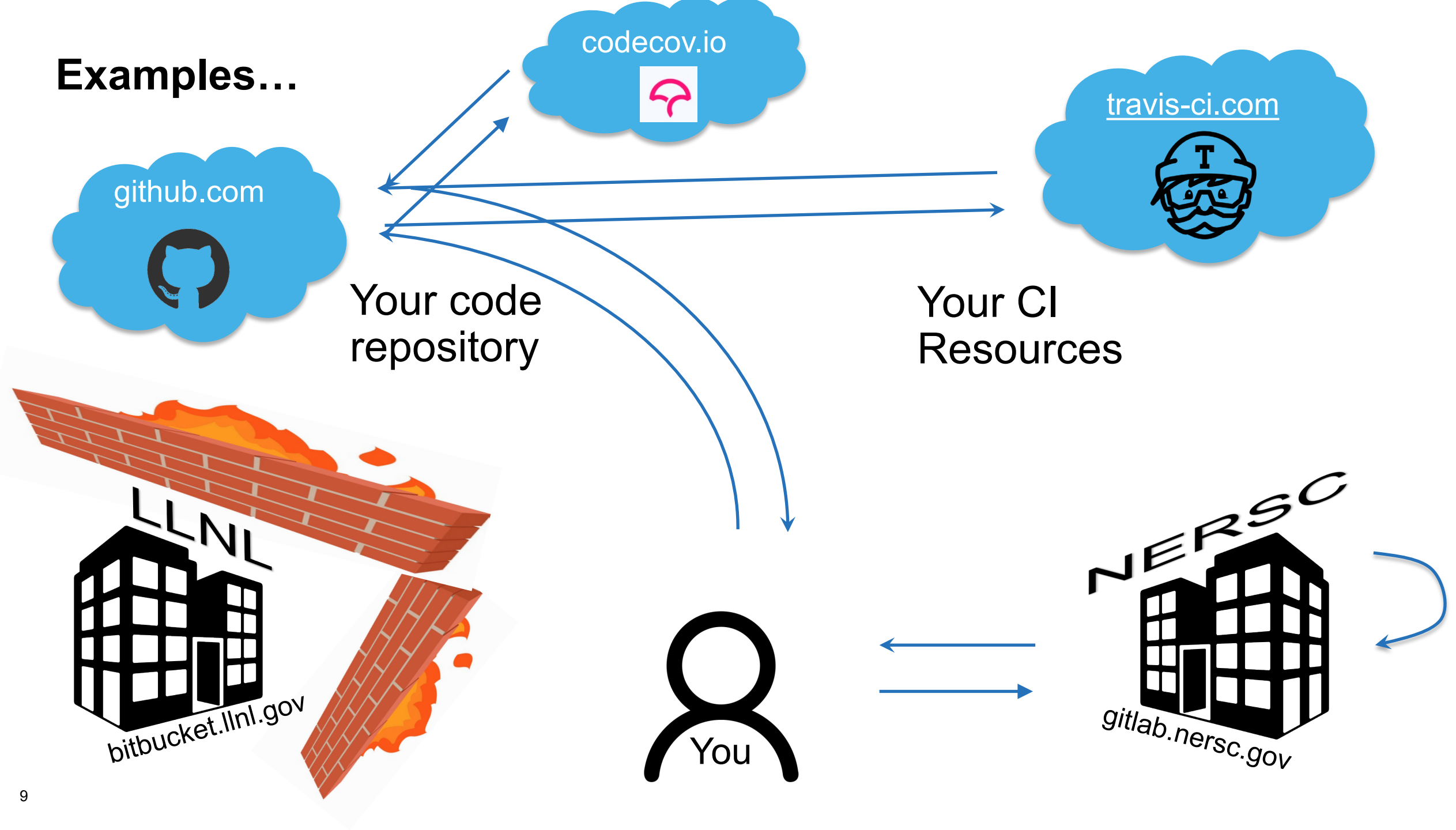| Common situations	Advanced situations

• 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
	• 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

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

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…

codecov.io

travis-ci.com

github.com

Your code repository

Your CI Resources

LLNL

bitbucket.llnl.gov

You

NERSC

gitlab.nersc.gov

9

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:

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

 1    name: Check Results
 2
 3    on:
 4      push:
 5        branches: [ main ]
 6      pull_request:
 7        branches: [ main ]
 8
 9    jobs:
10      build:
11
12        runs-on: ubuntu-latest
13
14        steps:
15        - uses: actions/checkout@v2
16        - name: make coverage
17          run: make CXXFLAGS=--coverage LDFLAGS="--coverage -lm" check_all
18        - name: upload coverage
19          run: bash <(curl -s https://codecov.io/bash)
```

github.com   codecov.io

GitHub Actions – results of workflow test runs

Workflows

All workflows

(TEST) Pyomo Windows Tests ...

(WIP) Pyomo Windows Test (P...

(WIP) Pyomo Windows Test (P...

(WIP) Pyomo Windows Tests (...

(WIP) Windows Pip Cmd Pyom...

GitHub Branch CI

GitHub CI

Pyomo Release Distribution Cr...

Python package

Ubuntu Pyomo Single Python ...

Ubuntu Pyomo Workflow (Slim,...

GitHub CI

Showing runs from all workflows named GitHub CI

event:push workflow:"GitHub CI"

461 workflow run results   Event ▾   Status ▾   Branch ▾   Actor ▾

✅ **Merge pull request #1902 from jsiirola/fix-unittest-rc**   `main`
GitHub CI #121: Commit 901b487 pushed by blnicho
📅 20 hours ago ⏱ 1h 3m 55s ...

✅ **Merge pull request #1901 from mrmundt/remove-six**   `main`
GitHub CI #117: Commit a101b6d pushed by mrmundt
📅 2 days ago ⏱ 1h 3m 12s ...

✅ **Merge pull request #1896 from jsiirola/abstract-disa...**   `main`
GitHub CI #112: Commit 1f9dd19 pushed by jsiirola
📅 2 days ago ⏱ 1h 5m 39s ...

❌ **Merge pull request #1898 from mrmundt/py-unittest**   `main`
GitHub CI #109: Commit 1beb848 pushed by michaelbynum
📅 3 days ago ⏱ 1h 3m 33s ...

✅ **Merge pull request #1893 from jsiirola/config-enum**   `main`
GitHub CI #105: Commit 9aa1186 pushed by jsiirola
📅 3 days ago ⏱ 1h 11m 3s ...

GitHub Pull Request Status Indicators

GitHub Actions – Key Elements of Defining Tests

```
1    name: GitHub CI
2
3    on:
4      push:
5        branches:
6          - main
7      pull_request:
8        branches:
9          - main
10     workflow_dispatch:
11       inputs:
12         git-ref:
13           description: Git Hash (Optional)
14           required: false
15
```

← Name

← Trigger

GitHub Actions – Key Elements of Defining Tests

```
35   jobs:
36     build:
37       name: ${{ matrix.TARGET }}/${{ matrix.python }}${{ matrix.other }}
38       runs-on: ${{ matrix.os }}
39       timeout-minutes: 90
40       strategy:
41         fail-fast: false
42        matrix:
43           os: [ubuntu-18.04, macos-latest, windows-latest]
44           python: [3.6, 3.7, 3.8, 3.9, pypy3]
45           other: [""]
46           category: ["nightly"]
47           # Ubuntu-18.04 should be replaced with ubuntu-latest once PyNumero
48           # build error is resolved:
49           # https://github.com/Pyomo/pyomo/issues/1710
50
51           include:
52           - os: ubuntu-18.04
53             TARGET: linux
54             PYENV: pip
55
56           - os: macos-latest
57             TARGET: osx
58             PYENV: pip
```
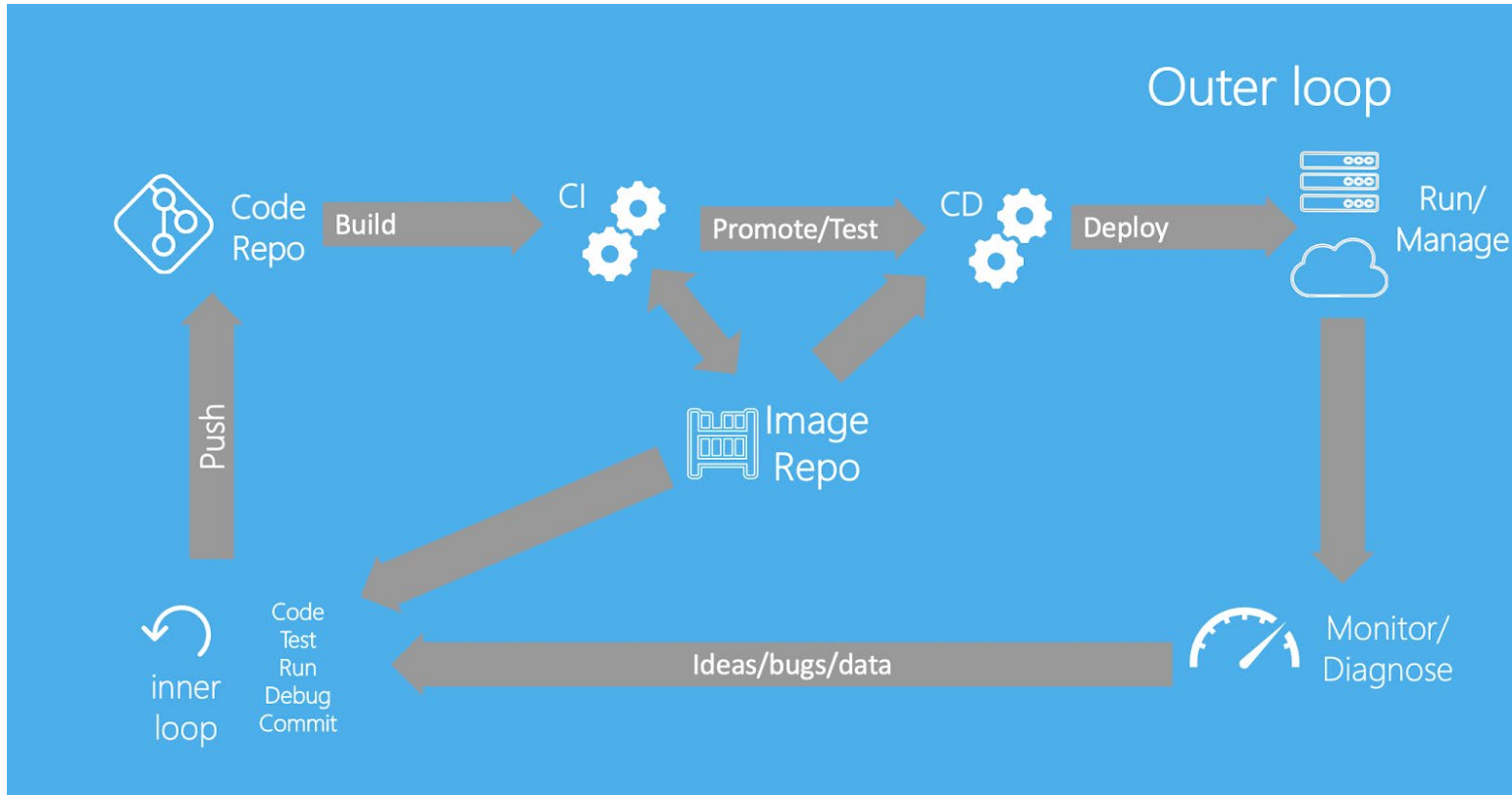
Job setup

17

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

Going Further with CI

Outer loop

Code Repo → Build → CI → Promote/Test → CD → Deploy → Run/Manage

Push

Image Repo

inner loop — Code Test Run Debug Commit

Ideas/bugs/data → Monitor/Diagnose

https://docs.docker.com/ci-cd/best-practices/

- Mirror inner and outer test loops
- High-level build systems
  - Makefile
  - autoconf / cmake
  - spack (spack.readthedocs.io)
  - containers? (e.g. in e4s-project)
- Enable caching
  - uses: actions/cache@v2 (github)
  - cache: (gitlab)
- Automate Releases

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

GitHub Actions - Release Automation

Merge pull request #1807 from Pyomo/finalize-release Pyomo Release Distribution Creation #9

Summary

Jobs

✅ manylinux/wheel_creation

✅ generic_tarball

✅ osxpy2.7/wheel_creation

✅ osxpy3.5/wheel_creation

✅ osxpy3.6/wheel_creation

✅ osxpy3.7/wheel_creation

✅ osxpy3.8/wheel_creation

✅ osxpy3.9/wheel_creation

✅ winpy3.6/wheel_creation

✅ winpy3.7/wheel_creation

✅ winpy3.8/wheel_creation

✅ winpy3.9/wheel_creation

Triggered via push 2 months ago
jsiirola pushed 46e336e 5.7.3

Status	Total duration	Artifacts
Success	**49m 30s**	**4**

release_wheel_creation.yml
on: push

Matrix: generictarball
✅ 1 job completed
Show all jobs

Matrix: manylinux
✅ 1 job completed
Show all jobs

Matrix: osx
✅ 6 jobs completed
Show all jobs

Matrix: windows
✅ 4 jobs completed
Show all jobs

Artifacts
Produced during runtime

Name	Size
generictarball	2.11 MB
manylinux-wheels	117 MB
osx-wheels	28.9 MB
win-wheels	18.5 MB

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

```
10 lines (7 sloc)    166 Bytes          .travis.yml

  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)

- <u>Start simple</u> 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

CI Testing is one part of the "Shift Left" movement in DevOps

ECP CI Resources
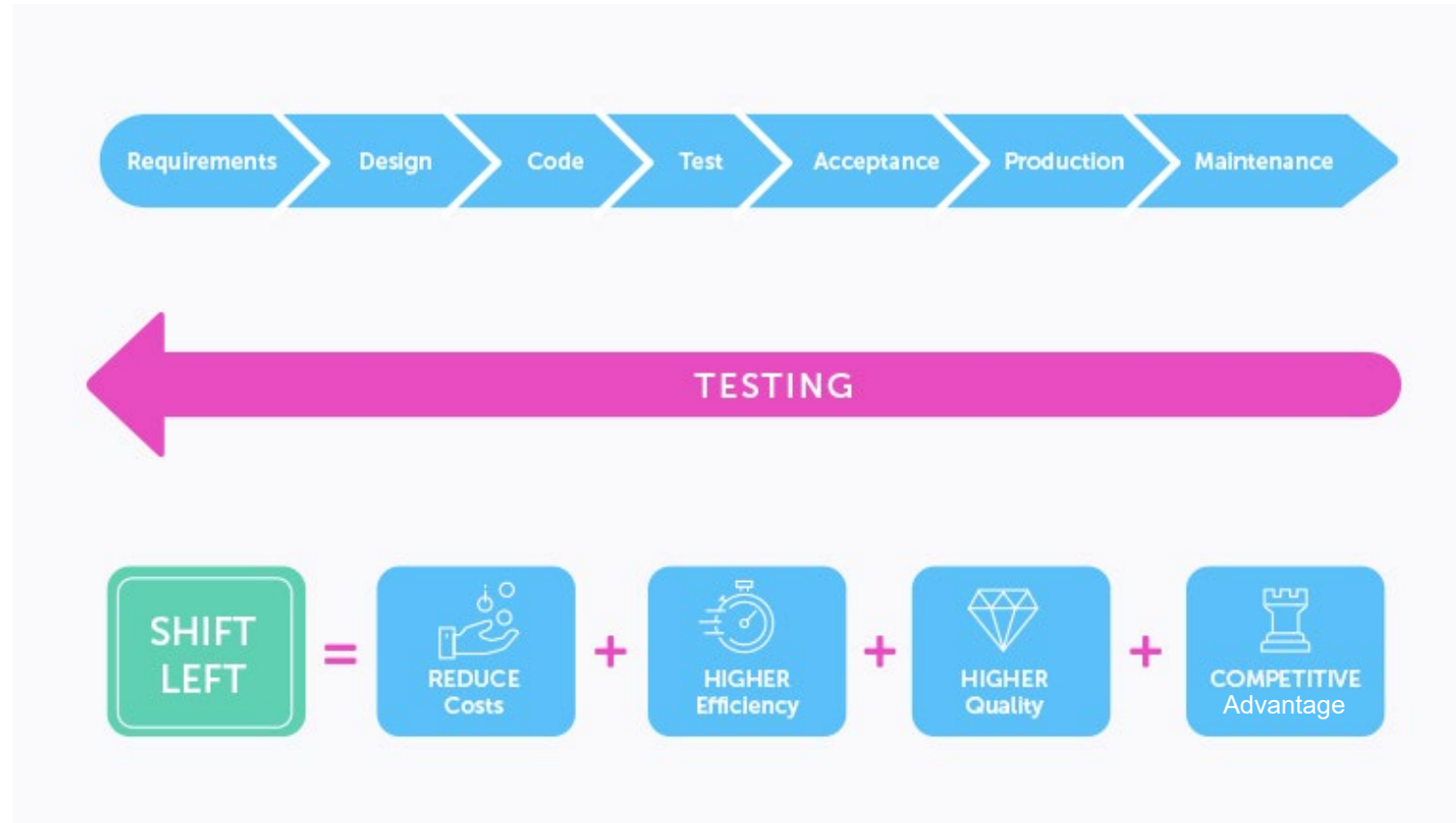
- ECP investing in GitLab for complex-wide CI

- Complex-wide Federation
  - Many hurdles still to overcome

- More about ECP CI at the Annual Meeting
  - Benefitting From ECP CI
    - Wednesday 2:30-3:30 Eastern
  - ECP CI Startup Tutorial
    - Friday 2:30-6 Eastern

- Documentation and on-boarding help
  - https://ecp-ci.gitlab.io

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

Keywords defined by service provider's YAML docs

markcmiller86 /
hello-numerical-world-atpesc-2020
forked from betterscientificsoftware/hello-numerical-world-atpesc-2020

<> Code   Pull requests   Actions   Projects   Wiki   Security

Watch ▾   0   ⭐ Star

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

```
1  language: c++
2
3  compiler: gcc
4
5  script: make check
```

Specify environment

Commands to run test

25

travis-ci.com   codecov.io