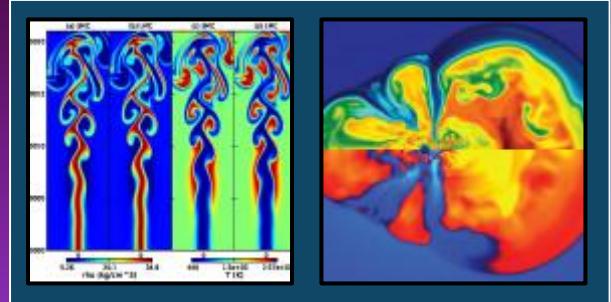


An Automated Approach to Continuous Acceptance Testing of HPC Systems at NERSC

HPCSYSPRO22 at SC22 Nov 14th, 2022



Shahzeb Siddiqui, LBNL, shahzebsiddiqui@lbl.gov

Erik Palmer, LBNL

Sameer Shende, University of Oregon

Wyatt Spear, University of Oregon

Prathmesh Sambrekar, Arizona State University

Sijie Xiang, Carnegie Mellon University



Our Software Testing and Integration Team

NERSC



Shahzeb Siddiqui



Erik Palmer



Justin Cook

Univ. of Oregon



Sameer Shende



Wyatt Spear



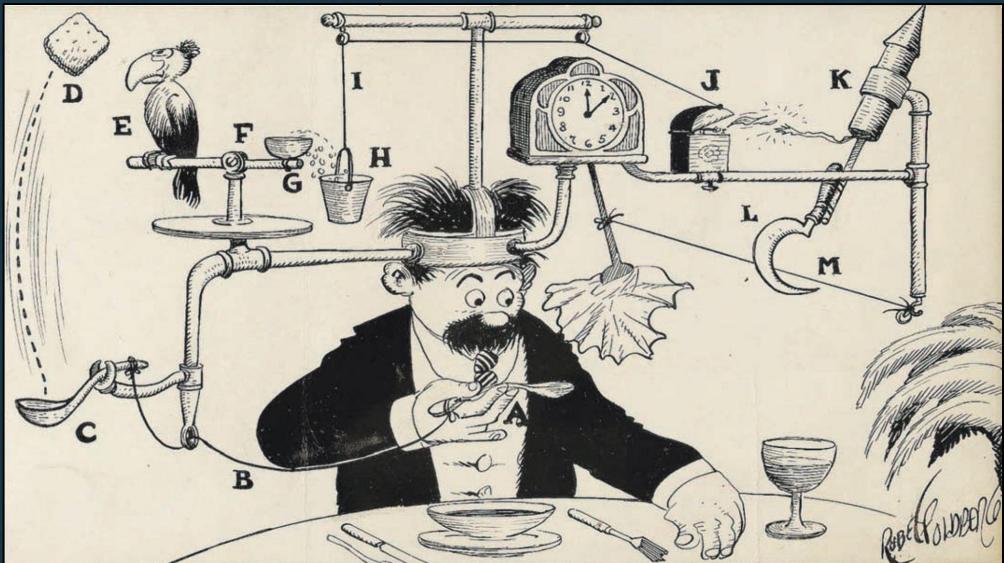
Prathmesh Sambrekar



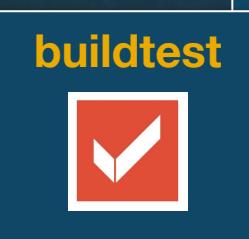
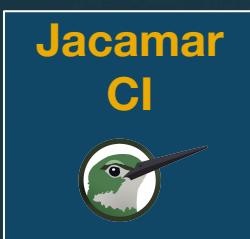
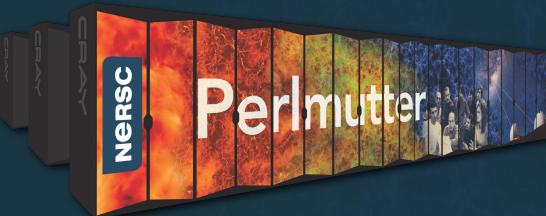
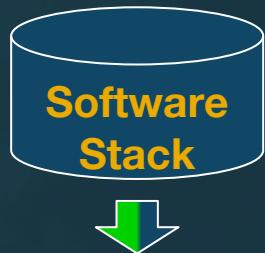
Sijie Xiang

Deploying Software in HPC can be Complex

- Tight, complex dependencies between software packages in a stack, and the environment
- Wide range of software needed by a wide range of users
- Research software is often not written with “product” qualities



Workflow



Deploy





GitLab

The screenshot shows the GitLab interface for the project 'buildtest-nersc'. At the top, it displays basic project statistics: 520 Commits, 4 Branches, 0 Tags, and 262.1 MB Project Storage. Below this is a search bar with dropdowns for 'devel' and 'buildtest-nersc /'. To the right are buttons for 'Find file', 'Web IDE', and 'Clone'. A timeline bar indicates the progress of a build. Below the search bar is a commit message from Shahzeb Siddiqui: 'updates to CONTRIBUTING guide,'. A timestamp shows it was authored 3 hours ago. Below the commit message are links for 'README', 'MIT License', 'CONTRIBUTING', 'CI/CD configuration', 'Add CHANGELOG', and 'Add Kubernetes cluster'. A 'Configure Integrations' button is also present. The main area shows a table of commits, with columns for 'Name', 'Last commit', and 'Last update'. The commits listed include .github, buildspecs, .gitignore, .gitlab-ci.yml, CONTRIBUTING.md, LEGAL.txt, LICENSE, README.md, config.yml, and tutorial.md.

- NERSC hosted GitLab CI/CD to develop and deploy our software stack
- Infrastructure and test development separated into two projects
- Pipelines initiated by merge and schedule
- GitLab runner runs as single user on a login node

Description	Target	Last Pipeline	Next Run	Owner	Actions
Muller System Check	devel	#74628	in 14 hours	Shahzeb Siddiqui	<button>Take ownership</button> <button>Delete</button>
Muller E4S Test	devel	#74630	in 16 hours	Shahzeb Siddiqui	<button>Take ownership</button> <button>Delete</button>
Perlmutter GPU Check	devel	#74605	in 57 minutes	Shahzeb Siddiqui	<button>Take ownership</button> <button>Delete</button>
Perlmutter System Check	devel	#74464	in 2 days	Shahzeb Siddiqui	<button>Take ownership</button> <button>Delete</button>



Jacamar CI

passed Job muller:check_spack_dependencies triggered 5 days ago by Erik T Palmer

```
1 Running with gitlab-runner 15.1.1 (bfccfab3)
2 on gitlab shell runner for muller for e4s user a2U1wJNq
3 Resolving secrets
4
5 Preparing the "custom" executor
6 Using Custom executor with driver Jacamar CI 0.11.0.pre.eeddb8...
7
8 Preparing environment
9 Targeting shell executor
10 Static builds directory in use, all jobs results kept. Please be aware of large
    disk space requirements when using this runner (minimum days: 7)
11 Running as epalmer UID: 93315 GID: 93315
12 Local time: 2022-10-25 08:16:04
13 Running on login01 via login01...
14
15 Getting source from Git repository
16 Fetching changes with git depth set to 1...
17 Initialized empty Git repository in /global/cfs/cdirs/m3503/ci-data-dir/epalmer/
    static/00172607/NERSC/spack-infrastructure/.git/
18 Created fresh repository.
19 Checking out fb064950 as main...
20 Skipping Git submodules setup
21
22 Executing "step_script" stage of the job script
23 WARNING: Starting with version 14.0 the 'build script' stage will be replaced wi
```

- Custom GitLab executor for HPC systems
- Provides a static directory structure for CI jobs to preserve previous job artifacts
- Integrates with batch schedulers
- Provides security features for HPC systems:
 - downscoping as a non-root user
 - server logs for job tracing



- Buildtest is a testing framework for HPC systems
- Intended for HPC Staff, developers, end-users who want to test the HPC system
- Designed to:
 - Facilitate test creation
 - Automate test execution
 - Aggregate test results
- Tests are written declaratively in YAML known as **buildspecs** which buildtest will process into a shell script.
- Integration with several batch schedulers including LSF, Slurm, PBS and Cobalt

A screenshot of a web browser displaying the buildtest documentation on ReadTheDocs. The URL in the address bar is "buildtest.readthedocs.io/en/devel/index.html".

The page title is "Docs > buildtest". On the right side, there is a "Edit on GitHub" button and a "Next" button. Below the title, there are several status indicators: "license MIT", "docs passing", "codecov 82%", "slack 0/58", and "regressiontest passing".

buildtest

This documentation was last rebuilt on Oct 14, 2022 and is intended for version 1.0.

If you are working off a latest release please see <https://buildtest.readthedocs.io/en/latest/> for documentation. If you are working off [devel](#) branch then please refer to <https://buildtest.readthedocs.io/en/devel/> which references the *devel* branch.

Useful Links

- Source Code: <https://github.com/buildtesters/buildtest>
- Documentation: <http://buildtest.rtfd.io/>
- Schema Docs: <https://buildtesters.github.io/buildtest/>
- ReadTheDocs: <https://readthedocs.org/projects/buildtest/>
- CodeCov: <https://codecov.io/gh/buildtesters/buildtest>
- Slack Channel: <http://hpcbuildtest.slack.com>
- Slack Invite: <https://hpcbuildtest.herokuapp.com/>
- CodeFactor: <https://www.codefactor.io/repository/github/buildtesters/buildtest>



buildtest

Example buildtest test “buildspec”

```
1 buildspecs:
2   amrex_single_vortex:
3     executor: '(perlmutter|muller).slurm.regular'
4     type: script
5     description: "AmrLevel SingleVortex Build and Run"
6     tags: ["e4s"]
7     sbatch:
8       - "-A m█████"
9       - "-N 1"
10      - "-t 00:05:00"
11      - "-C gpu"
12      # Note: This test doesn't use GPUs.
13     run: |
14       module load e4s/21.11-tcl
15
16       # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
17       module load PrgEnv-gnu
18
19       spack load cmake
20       spack load amrex
21
22       # Store AMReX install location
23       export AMR_DIR=$(spack location -i amrex)
24
25       cd test/Exec
26       mkdir build
27       cd build
28
29       cmake .. \
30       -DAMReX_FORTRAN=ON \
31       -DAMReX_FORTRAN_INTERFACES=ON \
32       -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
33       -DAMReX_SPACEDIM=3
34
35       cmake --build . -j4
36       srun -n 4 ./SingleVortex ../inputs max_step=1
37
38       status:
39         regex:
40           stream: stdout
41           exp: "finalized"
42       maintainers:
43         - epalmer
```

BERKELEY
LAB



NERSC



U.S. DEPARTMENT OF
ENERGY

Office of
Science



buildtest

Example buildtest test “buildspec”

```
1 buildspecs:
2   amrex_single_vortex:
3     executor: '(perlmutter|muller).slurm.regular'
4     type: script
5     description: "AmrLevel SingleVortex Build and Run"
6     tags: ["e4s"]
7     sbatch:
8       - "-A m█████"
9       - "-N 1"
10      - "-t 00:05:00"
11      - "-C gpu"
12      # Note: This test doesn't use GPUs.
13     run: |
14       module load e4s/21.11-tcl
15
16       # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
17       module load PrgEnv-gnu
18
19       spack load cmake
20       spack load amrex
21
22       # Store AMReX install location
23       export AMR_DIR=$(spack location -i amrex)
24
25       cd test/Exec
26       mkdir build
27       cd build
28
29       cmake .. \
30       -DAMReX_FORTRAN=ON \
31       -DAMReX_FORTRAN_INTERFACES=ON \
32       -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
33       -DAMReX_SPACEDIM=3
34
35       cmake --build . -j4
36       srun -n 4 ./SingleVortex ../inputs max_step=1
37
38       status:
39         regex:
40           stream: stdout
41           exp: "finalized"
42       maintainers:
43         - epalmer
```

- Declarative YAML format
- Validated by buildtest

BERKELEY
LAB



NERSC



Office of
Science



buildtest

Example buildtest test “buildspec”

```
1 buildspecs:
2   amrex_single_vortex:
3     executor: '(perlmutter|muller).slurm.regular'
4     type: script
5     description: "AmrLevel SingleVortex Build and Run"
6     tags: ["e4s"]
7     sbatch:
8       - "-A m█████"
9       - "-N 1"
10      - "-t 00:05:00"
11      - "-C gpu"
12      # Note: This test doesn't use GPUs.
13     run: |
14       module load e4s/21.11-tcl
15
16       # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
17       module load PrgEnv-gnu
18
19       spack load cmake
20       spack load amrex
21
22       # Store AMReX install location
23       export AMR_DIR=$(spack location -i amrex)
24
25       cd test/Exec
26       mkdir build
27       cd build
28
29       cmake .. \
30       -DAMReX_FORTRAN=ON \
31       -DAMReX_FORTRAN_INTERFACES=ON \
32       -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
33       -DAMReX_SPACEDIM=3
34
35       cmake --build . -j4
36       srun -n 4 ./SingleVortex ../inputs max_step=1
37
38       status:
39         regex:
40           stream: stdout
41           exp: "finalized"
42       maintainers:
43         - epalmer
```

- Declarative YAML format
- Validated by [buildtest](#)
- [buildtest preamble](#)



buildtest

Example buildtest test “buildspec”

```
1 buildspecs:
2   amrex_single_vortex:
3     executor: '(perlmutter|muller).slurm.regular'
4     type: script
5     description: "AmrLevel SingleVortex Build and Run"
6     tags: ["e4s"]
7     sbatch:
8       - "-A m█████"
9       - "-N 1"
10      - "-t 00:05:00"
11      - "-C gpu"
12      # Note: This test doesn't use GPUs.
13     run: |
14       module load e4s/21.11-tcl
15
16       # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
17       module load PrgEnv-gnu
18
19       spack load cmake
20       spack load amrex
21
22       # Store AMReX install location
23       export AMR_DIR=$(spack location -i amrex)
24
25       cd test/Exec
26       mkdir build
27       cd build
28
29       cmake .. \
30       -DAMReX_FORTRAN=ON \
31       -DAMReX_FORTRAN_INTERFACES=ON \
32       -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
33       -DAMReX_SPACEDIM=3
34
35       cmake --build . -j4
36       srun -n 4 ./SingleVortex ../inputs max_step=1
37
38       status:
39         regex:
40           stream: stdout
41           exp: "finalized"
42       maintainers:
43         - epalmer
```

- Declarative YAML format
- Validated by buildtest
- buildtest preamble
- scheduler commands

BERKELEY
LAB



NERSC



Office of
Science



buildtest

Example buildtest test “buildspec”

```
1 buildspecs:
2   amrex_single_vortex:
3     executor: '(perlmutter|muller).slurm.regular'
4     type: script
5     description: "AmrLevel SingleVortex Build and Run"
6     tags: ["e4s"]
7     sbatch:
8       - "-A m█████"
9       - "-N 1"
10      - "-t 00:05:00"
11      - "-C gpu"
12      # Note: This test doesn't use GPUs.
13    run: |
14      module load e4s/21.11-tcl
15
16      # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
17      module load PrgEnv-gnu
18
19      spack load cmake
20      spack load amrex
21
22      # Store AMReX install location
23      export AMR_DIR=$(spack location -i amrex)
24
25      cd test/Exec
26      mkdir build
27      cd build
28
29      cmake .. \
30      -DAMReX_FORTRAN=ON \
31      -DAMReX_FORTRAN_INTERFACES=ON \
32      -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
33      -DAMReX_SPACEDIM=3
34
35      cmake --build . -j4
36      srun -n 4 ./SingleVortex ../inputs max_step=1
37
38      status:
39        regex:
40          stream: stdout
41          exp: "finalized"
42      maintainers:
43        - epalmer
```

- Declarative YAML format
- Validated by buildtest
- buildtest preamble
- scheduler commands
- shell script:
 - modules
 - define job with SLURM commands

BERKELEY
LAB



NERSC



U.S. DEPARTMENT OF
ENERGY

Office of
Science



buildtest

Example buildtest test “buildspec”

```
1 buildspecs:
2   amrex_single_vortex:
3     executor: '(perlmutter|muller).slurm.regular'
4     type: script
5     description: "AmrLevel SingleVortex Build and Run"
6     tags: ["e4s"]
7     sbatch:
8       - "-A m█████"
9       - "-N 1"
10      - "-t 00:05:00"
11      - "-C gpu"
12      # Note: This test doesn't use GPUs.
13     run: |
14       module load e4s/21.11-tcl
15
16       # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
17       module load PrgEnv-gnu
18
19       spack load cmake
20       spack load amrex
21
22       # Store AMReX install location
23       export AMR_DIR=$(spack location -i amrex)
24
25       cd test/Exec
26       mkdir build
27       cd build
28
29       cmake .. \
30       -DAMReX_FORTRAN=ON \
31       -DAMReX_FORTRAN_INTERFACES=ON \
32       -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
33       -DAMReX_SPACEDIM=3
34
35       cmake --build . -j4
36       srun -n 4 ./SingleVortex ../inputs max_step=1
37
38       status:
39         regex:
40           stream: stdout
41           exp: "finalized"
42     maintainers:
43       - epalmer
```

- Declarative YAML format
- Validated by buildtest
- buildtest preamble
- scheduler commands
- shell script:
 - modules
 - define job with SLURM commands
- Flexible testing criteria:
 - regex of stdout

BERKELEY
LAB



NERSC



U.S. DEPARTMENT OF
ENERGY

Office of
Science



buildtest

Example buildtest test “buildspec”

```
1 buildspecs:
2   amrex_single_vortex:
3     executor: '(perlmutter|muller).slurm.regular'
4     type: script
5     description: "AmrLevel SingleVortex Build and Run"
6     tags: ["e4s"]
7     sbatch:
8       - "-A m█████"
9       - "-N 1"
10      - "-t 00:05:00"
11      - "-C gpu"
12      # Note: This test doesn't use GPUs.
13     run: |
14       module load e4s/21.11-tcl
15
16       # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
17       module load PrgEnv-gnu
18
19       spack load cmake
20       spack load amrex
21
22       # Store AMReX install location
23       export AMR_DIR=$(spack location -i amrex)
24
25       cd test/Exec
26       mkdir build
27       cd build
28
29       cmake .. \
30       -DAMReX_FORTRAN=ON \
31       -DAMReX_FORTRAN_INTERFACES=ON \
32       -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
33       -DAMReX_SPACEDIM=3
34
35       cmake --build . -j4
36       srun -n 4 ./SingleVortex ../inputs max_step=1
37
38       status:
39         regex:
40           stream: stdout
41           exp: "finalized"
42     maintainers:
43       - epalmer
```

```
[LEVEL 2 Step 20] Advanced 247668 cells
Coarse STEP 5 ends. TIME = 0.05475189088 DT = 0.01096740925
Writing plotfile plt00005

Total Time: 3.564830664
Unused ParmParse Variables:
[TOP]::amr.do_tracers(nvals = 1) :: [1]

AMReX (22.10) finalized
```

- Declarative YAML format
- Validated by buildtest
- buildtest preamble
- scheduler commands
- shell script:
 - modules
 - define job with SLURM commands
- Flexible testing criteria:
 - regex of stdout





buildtest

Example buildtest test “buildspec”

```
1 buildspecs:
2   amrex_single_vortex:
3     executor: '(perlmutter|muller).slurm.regular'
4     type: script
5     description: "AmrLevel SingleVortex Build and Run"
6     tags: ["e4s"]
7     sbatch:
8       - "-A m█████"
9       - "-N 1"
10      - "-t 00:05:00"
11      - "-C gpu"
12      # Note: This test doesn't use GPUs.
13     run: |
14       module load e4s/21.11-tcl
15
16       # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
17       module load PrgEnv-gnu
18
19       spack load cmake
20       spack load amrex
21
22       # Store AMReX install location
23       export AMR_DIR=$(spack location -i amrex)
24
25       cd test/Exec
26       mkdir build
27       cd build
28
29       cmake .. \
30       -DAMReX_FORTRAN=ON \
31       -DAMReX_FORTRAN_INTERFACES=ON \
32       -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
33       -DAMReX_SPACEDIM=3
34
35       cmake --build . -j4
36       srun -n 4 ./SingleVortex ../inputs max_step=1
37
38       status:
39         regex:
40           stream: stdout
41           exp: "finalized"
42       maintainers:
43         - epalmer
```

- Declarative YAML format
- Validated by buildtest
- buildtest preamble
- scheduler commands
- shell script:
 - modules
 - define job with SLURM commands
- Flexible testing criteria:
 - regex of stdout



Spack

```
1 buildspecs:
2   spack_test_amrex_e4s_22.05:
3     type: spack
4     executor: '(perlmutter|muller).local.sh'
5     description: "Test AMReX for e4s/22.05 stack for Perlmutter via spack test"
6     tags: e4s
7     pre_cmds: |
8       module swap PrgEnv-nvidia PrgEnv-gnu
9       module load e4s/22.05
10    spack:
11      root: /global/common/software/spackcp/perlmutter/e4s-22.05/spack/
12      verify_spack: false
13      test:
14        run:
15          specs: ['amrex']
16          results:
17            option: '-l'
18            specs: ['amrex']
```

- Spack is a package manager for supercomputers
- Used to deploy our software stack
- Spack supports stand-alone testing which buildtest can leverage



Testsuite

```
1 #This buildtest was generated from e4s-template.yaml
2 buildspecs:
3   hdf5_e4s_testsuite_22.05:
4     type: script
5     executor: perlmutter.slurm.regular
6     description: Run E4S Testsuite hdf5 test for e4s/22.05 stack
7     tags: [e4s]
8     sbatch: ["-t 30", "-N 1", "-G 1", "-A █", "-C gpu"]
9     run: |
10       module load e4s/22.05
11       spack load --first cmake
12       git clone https://github.com/E4S-Project/testsuite
13       cd testsuite
14
15       sh test-all.sh --color-off validation_tests/hdf5 --print-logs
16 maintainers:
17 - shahzebsiddiqui
18 - wspear
```

- E4S Testsuite includes over 100 tests, with a goal of full test coverage for the E4S ecosystem
- Test code largely extracted from package sources
- Uses a simple, extensible bash-script test driver



5 passed, 3 failed, 0 not run, 3 missing.

Name	Status	Time	Details	Labels	History	Summary	compiler	description	endtime	hostname	starttime	user
e4s_hdf5	Failed	1h 46m 32s 940ms	e4s		Broken	Broken		Run hdf5 test from E4S Testsuite. This test is expected to fail	2022/10/18 08:50:44	cori02	2022/10/18 07:04:11	e4s
e4s_testsuite_e4s_22.02_gnu	Failed	2h 4m 6s 160ms	e4s		Broken	Broken		Run E4S Testsuite for e4s/22.02, gnu stack	2022/10/18 09:08:17	cori02	2022/10/18 07:04:11	e4s
e4s_testsuite_e4s_22.02_intel	Failed	2h 4m 36s 360ms	e4s		Broken	Broken		Run E4S Testsuite for e4s/22.02, intel stack	2022/10/18 09:08:47	cori02	2022/10/18 07:04:11	e4s
ascent_e4stestsuite_e4s_21.05	Missing											
e4s_adios2	Missing											
e4s_superlu-dist	Missing											
e4s_22_02_spack	Passed	6s 660ms	e4s spack	Unstable	Stable			Check e4s/21.05 spack instance	2022/10/18 07:04:17	cori02	2022/10/18 07:04:11	e4s
showquota	Passed	2s 170ms		Stable	Stable			run the showquota utility	2022/10/18 07:04:13	cori02	2022/10/18 07:04:11	e4s
showquota_check_m3503	Passed	830ms	e4s	Stable	Stable			run the showquota check on m3503 project directory	2022/10/18 07:04:12	cori02	2022/10/18 07:04:11	e4s
showquota_check_spackcp	Passed	680ms	e4s	Stable	Stable			run showquota on /global/common/software/spackcp	2022/10/18 07:04:11	cori02	2022/10/18 07:04:11	e4s
spack_e4s_directory_checks	Passed	80ms	e4s	Stable	Stable			Check for production directory paths for spack e4s stack	2022/10/18 07:04:11	cori02	2022/10/18 07:04:11	e4s



- Buildtest can upload results into CDash
- CDash produces color coded charts, timelines (track progress over time), and links to test output
- Comprehensible presentation of test results posted to the web for users and developers

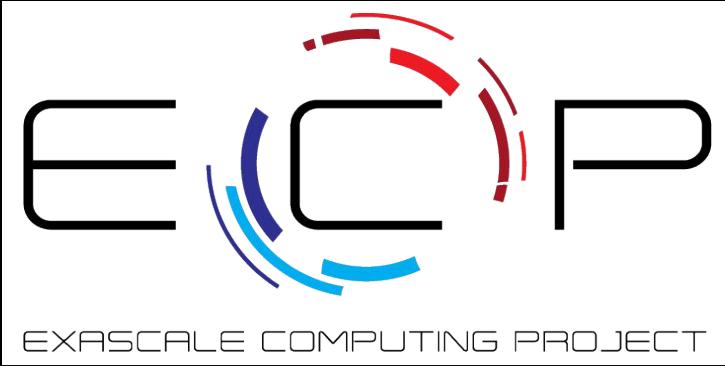
Why do we test?



- Detect bugs in an environment as close to our user experience as possible
- Tests and results are publicly available and digestible for non-experts

Increase confidence in our system

Acknowledgements



This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

References

Buildtest

- Documentation: <https://buildtest.rtfd.io/>
- Source Code: <https://github.com/buildtesters/buildtest>
- Slack: <http://hpcbuildtest.slack.com/>
- Buildtest at NERSC: <https://github.com/buildtesters/buildtest-nersc>
- CDASH for buildtest-nersc:
<https://my.cdash.org/index.php?project=buildtest-nersc>



E4S

- E4S Home Page: <https://e4s.io>
- E4S Docs: <https://e4s.rtfd.io/>
- E4S Testsuite: <https://github.com/E4S-Project/testsuite>



Spack

- Documentation <https://spack.rtfd.io>
- Source Code: <https://github.com/spack/spack>



GitLab

- GitLab Runner Docs: <https://docs.gitlab.com/runner/>
- GitLab CI/CD: <https://docs.gitlab.com/ee/ci/>



Jacamar CI

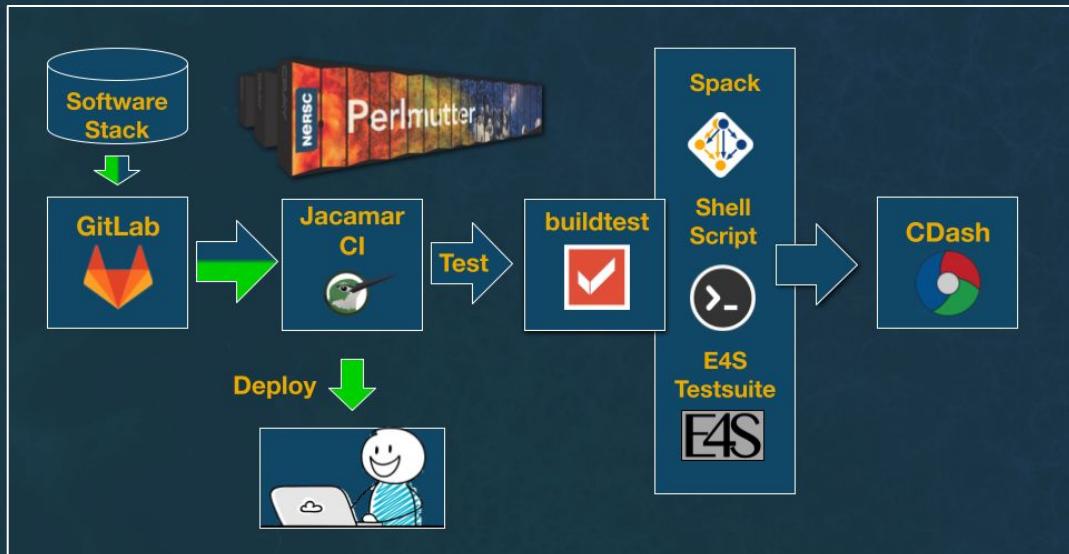
- Documentation: <https://ecp-ci.gitlab.io/>
- Source Code: <https://gitlab.com/ecp-ci/jacamar-ci>



Thanks for listening!

Key Points:

- Automated pipeline to build, test, and deploy a scientific software stack
- Jacamar CI and buildtest, bridge the gap between GitLab and our HPC systems with several unique features.
- Tests and test results are easily accessible and digestible by users to increase system confidence.



Shahzeb Siddiqui, LBNL, shahzebsiddiqui@lbl.gov
Erik Palmer, LBNL
Sameer Shende, University of Oregon
Wyatt Spear, University of Oregon
Prathmesh Sambrekar, Arizona State University
Sijie Xiang, Carnegie Mellon University