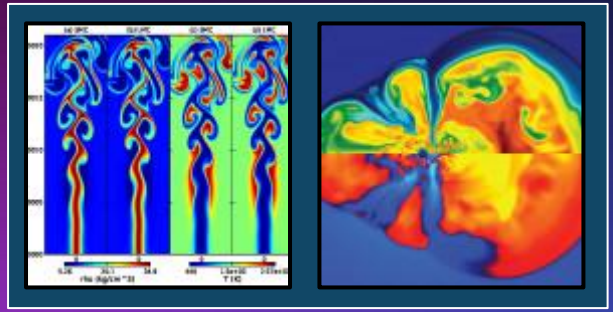
An Automated Approach to Continuous Acceptance Testing of HPC Systems at NERSC

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

SC22 Dallas, TX | hpc accelerates.

BERKELEY LAB

NERSC

U.S. DEPARTMENT OF ENERGY

Our Software Testing and Integration Team

NERSC

Shahzeb Siddiqui

Erik Palmer

Justin Cook

Prathmesh Sambrekar

Sijie Xiang

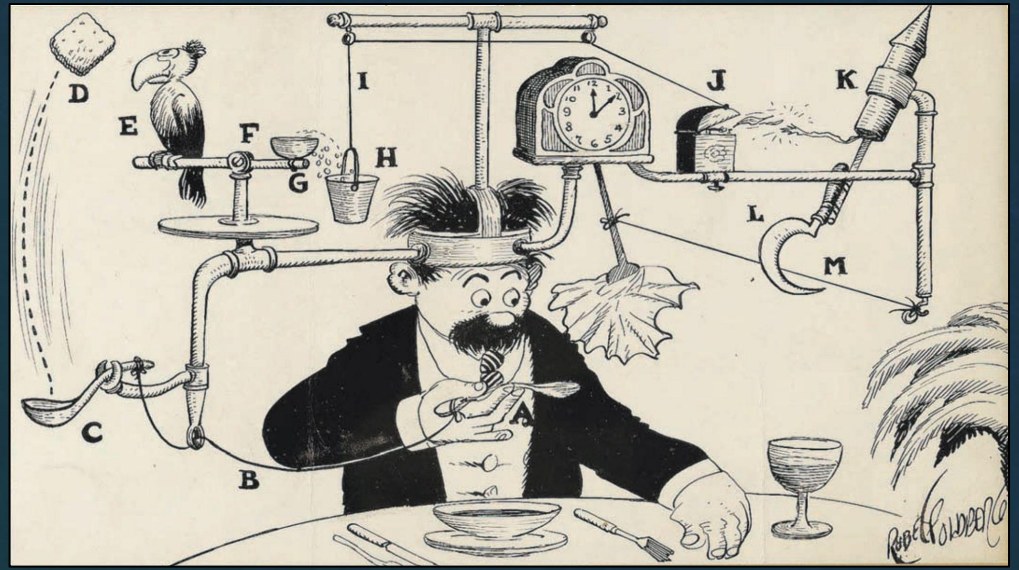
Univ. of Oregon

Sameer Shende

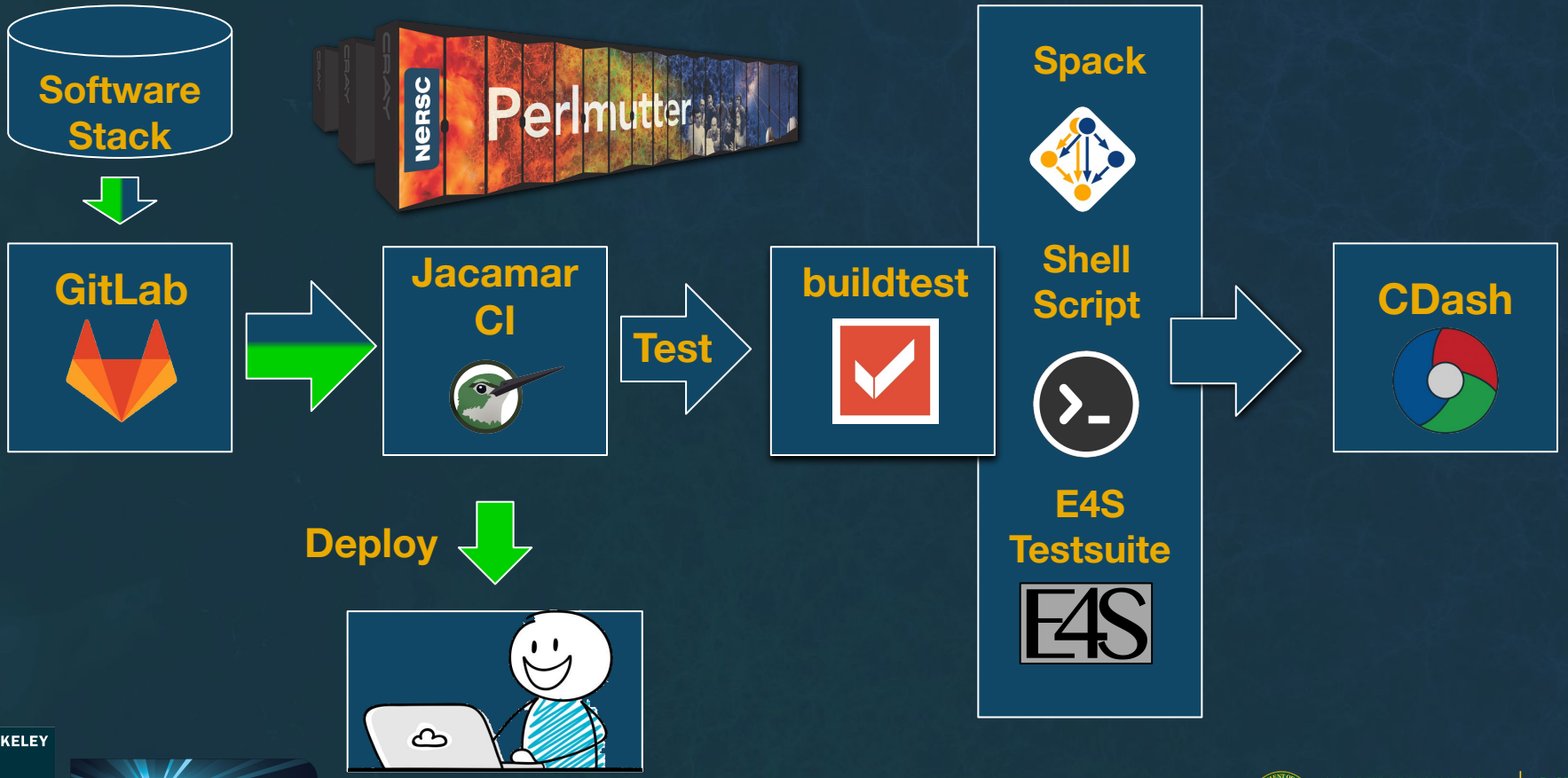
Wyatt Spear

Deploying Software in HPC can be Complex

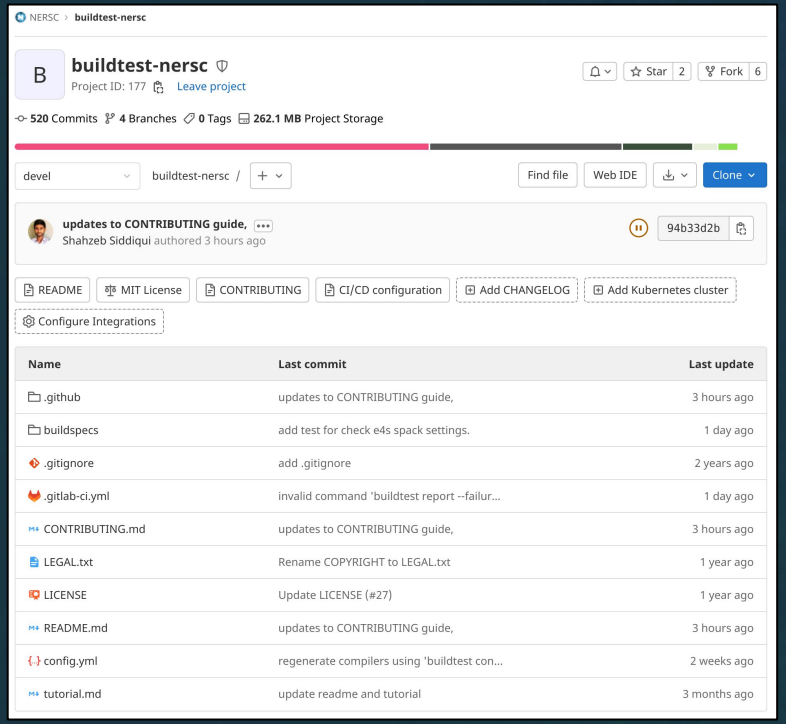
- Tight, complex dependencies between software packages in a stack, and the environment
- Wide range of software needed by a wide range of users
- Research software is often not written with “product” qualities



Workflow


Software Stack
NERSC
Perlmutter
Spack
GitLab
Jacamar CI
Test
buildtest
Shell Script
CDash
Deploy
E4S Testsuite
E4S

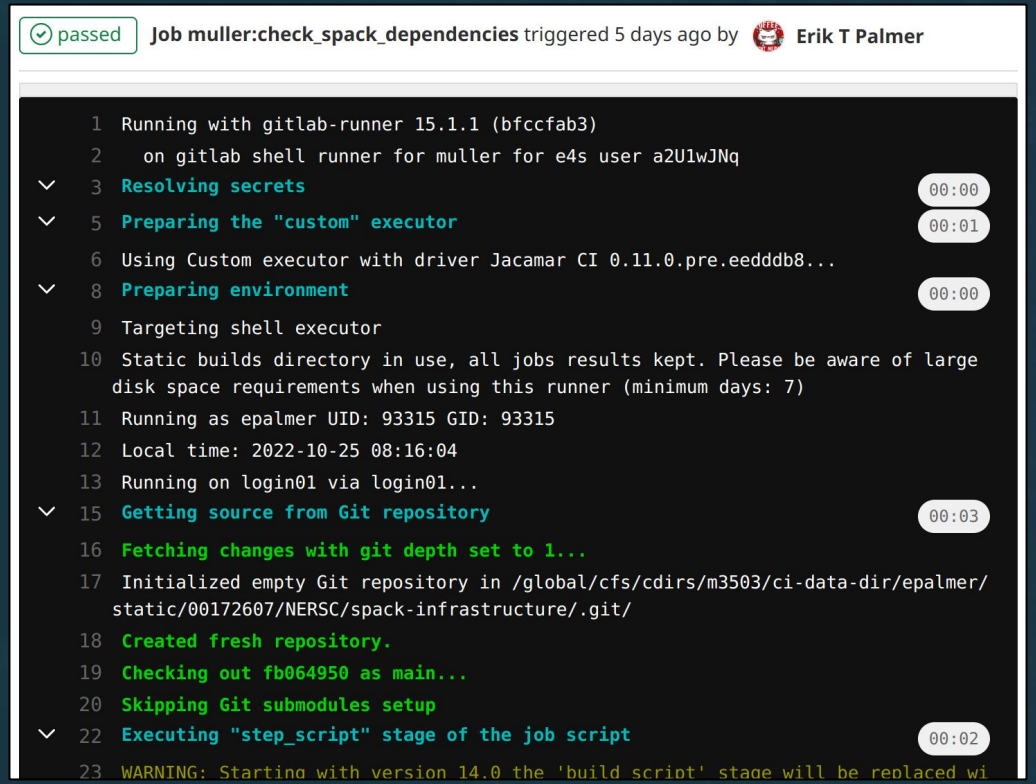
GitLab



- 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	
Muller System Check	devel	#74628	in 14 hours	Shahzeb Siddiqui	Take ownership
Muller E4S Test	devel	#74630	in 16 hours	Shahzeb Siddiqui	Take ownership
Perlmutter GPU Check	devel	#74605	in 57 minutes	Shahzeb Siddiqui	Take ownership
Perlmutter System Check	devel	#74464	in 2 days	Shahzeb Siddiqui	Take ownership

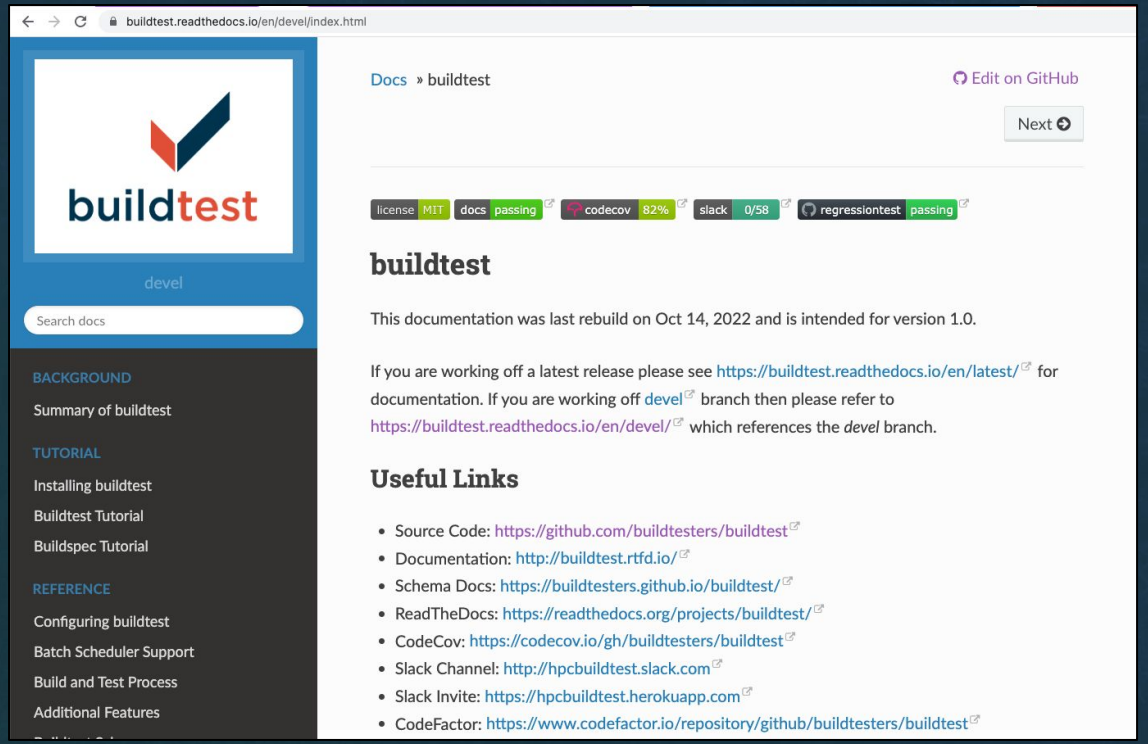
Jacamar CI



- 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

- 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



buildtest

Example buildtest test “buildspec”

```yaml
buildspecs:
  amrex_single_vortex:
    executor: '(perlmutter|muller).slurm.regular'
    type: script
    description: "AmrLevel SingleVortex Build and Run"
    tags: ["e4s"]
    sbatch:
      - "-A m[redacted]"
      - "-N 1"
      - "-t 00:05:00"
      - "-C gpu"
      # Note: This test doesn't use GPUs.
    run: |
      module load e4s/21.11-tcl

      # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
      module load PrgEnv-gnu

      spack load cmake
      spack load amrex

      # Store AMReX install location
      export AMR_DIR=$(spack location -i amrex)

      cd test/Exec
      mkdir build
      cd build

      cmake .. \
      -DAMReX_FORTRAN=ON \
      -DAMReX_FORTRAN_INTERFACES=ON \
      -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
      -DAMReX_SPACEDIM=3

      cmake --build . -j4
      srun -n 4 ./SingleVortex ../inputs max_step=1

    status:
      regex:
        stream: stdout
        exp: "finalized"
maintainers:
  - epalmer
```

BERKELEY LAB

NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

buildtest

Example buildtest test "buildspec"

```yaml
buildspecs:
  amrex_single_vortex:
    executor: '(perlmutter|muller).slurm.regular'
    type: script
    description: "AmrLevel SingleVortex Build and Run"
    tags: ["e4s"]
    sbatch:
      - "-A m[redacted]"
      - "-N 1"
      - "-t 00:05:00"
      - "-C gpu"
      # Note: This test doesn't use GPUs.
    run: |
      module load e4s/21.11-tcl

      # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
      module load PrgEnv-gnu

      spack load cmake
      spack load amrex

      # Store AMReX install location
      export AMR_DIR=$(spack location -i amrex)

      cd test/Exec
      mkdir build
      cd build

      cmake .. \
      -DAMReX_FORTRAN=ON \
      -DAMReX_FORTRAN_INTERFACES=ON \
      -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
      -DAMReX_SPACEDIM=3

      cmake --build . -j4
      srun -n 4 ./SingleVortex ../inputs max_step=1

    status:
      regex:
        stream: stdout
        exp: "finalized"
maintainers:
  - epalmer
```

- Declarative YAML format
- Validated by buildtest

BERKELEY LAB

NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

buildtest

Example buildtest test “buildspec”

```yaml
buildspecs:
  amrex_single_vortex:
    executor: '(perlmutter|muller).slurm.regular'
    type: script
    description: "AmrLevel SingleVortex Build and Run"
    tags: ["e4s"]
    sbatch:
      - "-A m[redacted]"
      - "-N 1"
      - "-t 00:05:00"
      - "-C gpu"
      # Note: This test doesn't use GPUs.
    run: |
      module load e4s/21.11-tcl

      # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
      module load PrgEnv-gnu

      spack load cmake
      spack load amrex

      # Store AMReX install location
      export AMR_DIR=$(spack location -i amrex)

      cd test/Exec
      mkdir build
      cd build

      cmake .. \
      -DAMReX_FORTRAN=ON \
      -DAMReX_FORTRAN_INTERFACES=ON \
      -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
      -DAMReX_SPACEDIM=3

      cmake --build . -j4
      srun -n 4 ./SingleVortex ../inputs max_step=1

    status:
      regex:
        stream: stdout
        exp: "finalized"
maintainers:
  - epalmer
```

- Declarative YAML format
- Validated by buildtest
- buildtest preamble

BERKELEY LAB

NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

buildtest

Example buildtest test "buildspec"

```yaml
buildspecs:
  amrex_single_vortex:
    executor: '(perlmutter|muller).slurm.regular'
    type: script
    description: "AmrLevel SingleVortex Build and Run"
    tags: ["e4s"]
    sbatch:
      - "-A m[redacted]"
      - "-N 1"
      - "-t 00:05:00"
      - "-C gpu"
      # Note: This test doesn't use GPUs.
    run: |
      module load e4s/21.11-tcl

      # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
      module load PrgEnv-gnu

      spack load cmake
      spack load amrex

      # Store AMReX install location
      export AMR_DIR=$(spack location -i amrex)

      cd test/Exec
      mkdir build
      cd build

      cmake .. \
      -DAMReX_FORTRAN=ON \
      -DAMReX_FORTRAN_INTERFACES=ON \
      -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
      -DAMReX_SPACEDIM=3

      cmake --build . -j4
      srun -n 4 ./SingleVortex ../inputs max_step=1

    status:
      regex:
        stream: stdout
        exp: "finalized"
maintainers:
  - epalmer
```

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

BERKELEY LAB

NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

buildtest

Example buildtest test "buildspec"

```yaml
buildspecs:
  amrex_single_vortex:
    executor: '(perlmutter|muller).slurm.regular'
    type: script
    description: "AmrLevel SingleVortex Build and Run"
    tags: ["e4s"]
    sbatch:
      - "-A m[redacted]"
      - "-N 1"
      - "-t 00:05:00"
      - "-C gpu"
      # Note: This test doesn't use GPUs.
    run: |
      module load e4s/21.11-tcl

      # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
      module load PrgEnv-gnu

      spack load cmake
      spack load amrex

      # Store AMReX install location
      export AMR_DIR=$(spack location -i amrex)

      cd test/Exec
      mkdir build
      cd build

      cmake .. \
      -DAMReX_FORTRAN=ON \
      -DAMReX_FORTRAN_INTERFACES=ON \
      -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
      -DAMReX_SPACEDIM=3

      cmake --build . -j4
      srun -n 4 ./SingleVortex ../inputs max_step=1

    status:
      regex:
        stream: stdout
        exp: "finalized"
maintainers:
  - 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”

```yaml
buildspecs:
  amrex_single_vortex:
    executor: '(perlmutter|muller).slurm.regular'
    type: script
    description: "AmrLevel SingleVortex Build and Run"
    tags: ["e4s"]
    sbatch:
      - "-A m[redacted]"
      - "-N 1"
      - "-t 00:05:00"
      - "-C gpu"
      # Note: This test doesn't use GPUs.
    run: |
      module load e4s/21.11-tcl

      # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
      module load PrgEnv-gnu

      spack load cmake
      spack load amrex

      # Store AMReX install location
      export AMR_DIR=$(spack location -i amrex)

      cd test/Exec
      mkdir build
      cd build

      cmake .. \
      -DAMReX_FORTRAN=ON \
      -DAMReX_FORTRAN_INTERFACES=ON \
      -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
      -DAMReX_SPACEDIM=3

      cmake --build . -j4
      srun -n 4 ./SingleVortex ../inputs max_step=1

    status:
      regex:
        stream: stdout
        exp: "finalized"
maintainers:
  - 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"

```yaml
buildspecs:
  amrex_single_vortex:
    executor: '(perlmutter|muller).slurm.regular'
    type: script
    description: "AmrLevel SingleVortex Build and Run"
    tags: ["e4s"]
    sbatch:
      - "-A m[redacted]"
      - "-N 1"
      - "-t 00:05:00"
      - "-C gpu"
      # Note: This test doesn't use GPUs.
    run: |
      module load e4s/21.11-tcl

      # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
      module load PrgEnv-gnu

      spack load cmake
      spack load amrex

      # Store AMReX install location
      export AMR_DIR=$(spack location -i amrex)

      cd test/Exec
      mkdir build
      cd build

      cmake .. \
      -DAMReX_FORTRAN=ON \
      -DAMReX_FORTRAN_INTERFACES=ON \
      -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
      -DAMReX_SPACEDIM=3

      cmake --build . -j4
      srun -n 4 ./SingleVortex ../inputs max_step=1

    status:
      regex:
        stream: stdout
        exp: "finalized"
maintainers:
  - 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

```
[Level 2 step 20] Advanced 247868 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
```

BERKELEY LAB

NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

buildtest

Example buildtest test “buildspec”

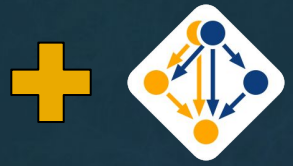
```yaml
buildspecs:
  amrex_single_vortex:
    executor: '(perlmutter|muller).slurm.regular'
    type: script
    description: "AmrLevel SingleVortex Build and Run"
    tags: ["e4s"]
    sbatch:
      - "-A m[redacted]"
      - "-N 1"
      - "-t 00:05:00"
      - "-C gpu"
      # Note: This test doesn't use GPUs.
    run: |
      module load e4s/21.11-tcl

      # Use the PrgEnv-gnu to avoid issues with NVHPC and Fortran
      module load PrgEnv-gnu

      spack load cmake
      spack load amrex

      # Store AMReX install location
      export AMR_DIR=$(spack location -i amrex)

      cd test/Exec
      mkdir build
      cd build

      cmake .. \
      -DAMReX_FORTRAN=ON \
      -DAMReX_FORTRAN_INTERFACES=ON \
      -DAMReX_DIR=${AMR_DIR}/lib/cmake/AMReX \
      -DAMReX_SPACEDIM=3

      cmake --build . -j4
      srun -n 4 ./SingleVortex ../inputs max_step=1

    status:
      regex:
        stream: stdout
        exp: "finalized"
maintainers:
  - 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



Spack

```
buildspecs:
  spack_test_amrex_e4s_22.05:
    type: spack
    executor: '(perlmutter|muller).local.sh'
    description: "Test AMReX for e4s/22.05 stack for Perlmutter via spack test"
    tags: e4s
    pre_cmds: |
      module swap PrgEnv-nvidia PrgEnv-gnu
      module load e4s/22.05
    spack:
      root: /global/common/software/spackecp/perlmutter/e4s-22.05/spack/
      verify_spack: false
      test:
        run:
          specs: ['amrex']
        results:
          option: '-l'
          specs: ['amrex']
```

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

buildtest + E4S Testsuite

```yaml
#This buildtest was generated from e4s-template.yaml
buildspecs:
  hdf5_e4s_testsuite_22.05:
    type: script
    executor: perlmutter.slurm.regular
    description: Run E4S Testsuite hdf5 test for e4s/22.05 stack
    tags: [e4s]
    sbatch: ["-t 30", "-N 1", "-G 1", "-A [redacted]", "-C gpu"]
    run: |
      module load e4s/22.05
      spack load --first cmake
      git clone https://github.com/E4S-Project/testsuite
      cd testsuite

      sh test-all.sh --color-off validation_tests/hdf5 --print-logs
maintainers:
  - shahzebsiddiqui
  - 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

CDash

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_spackecp	Passed	680ms		e4s	Stable	Stable		run showquota on /global/common/software/spackecp	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

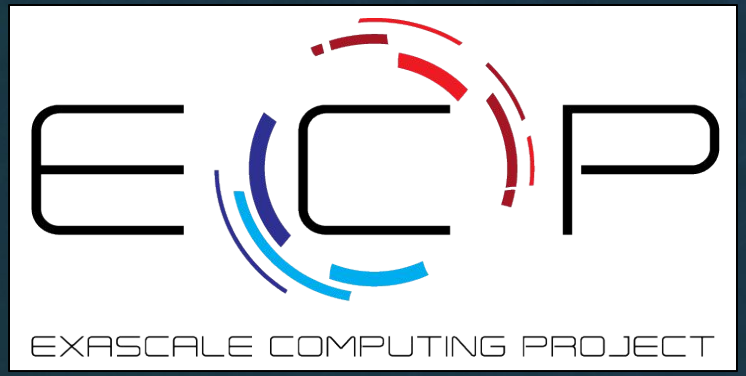
- Buildtest can upload results into CDash
- CDash produces color coded charts, timelines (track progress over time), and links to test output
- Comprehensible presentation of test results posted to the web for users and developers

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.

BERKELEY LAB

NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

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/

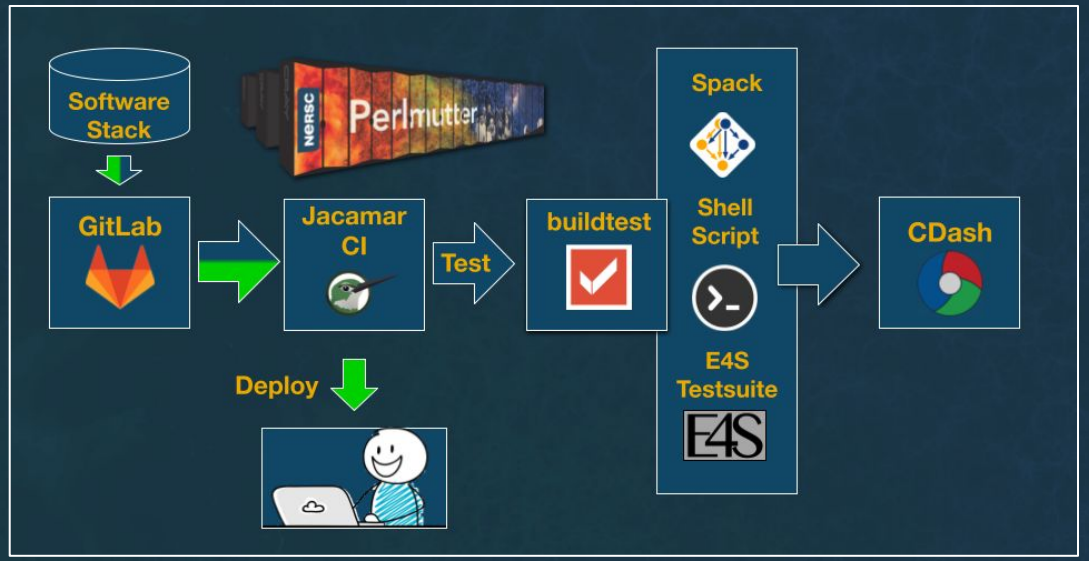
Jacamar CI

- 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